Facultad 5

Trabajo de Diploma para optar por el título de Ingeniera en Ciencias Informáticas

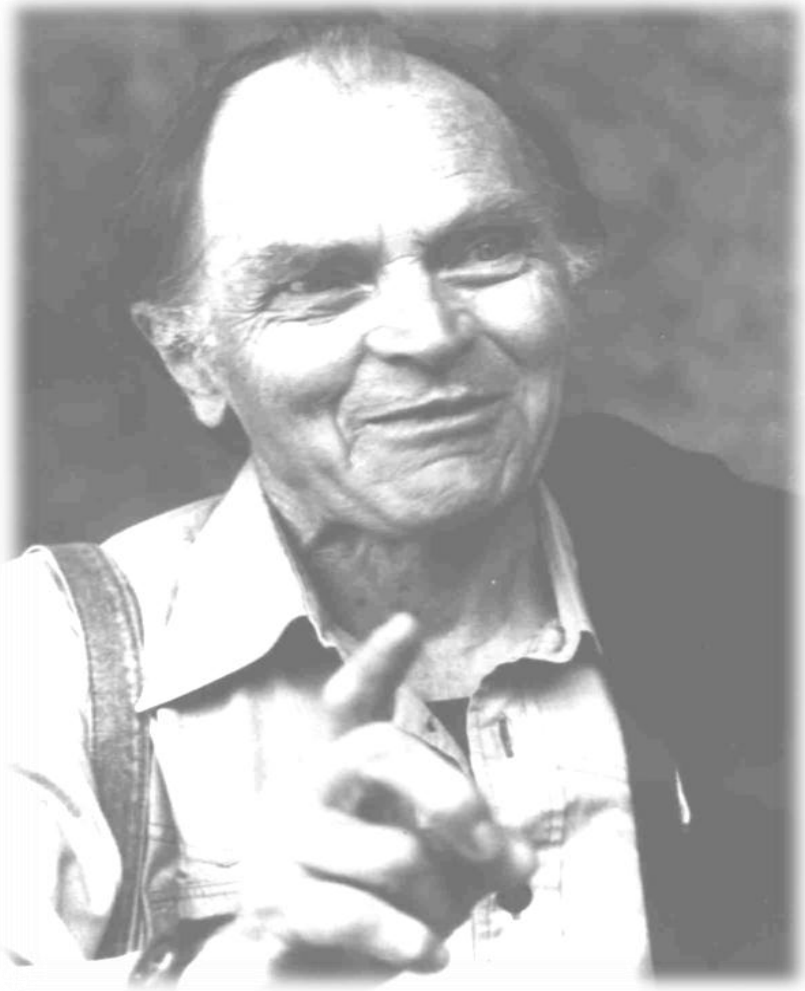
Título: Desarrollo de una base de datos difusa para el Sistema de Evaluación de Competencias del proyecto Talenmático

Autora: Evelyn López Gámez

Tutores: Ing. Luis Manuel Teijón Acosta

Msc. Manuel Villanueva Betancourt

La Habana, junio de 2013



“La razón por la cual el lenguaje natural se expresa en términos difusos no es porque el pensamiento humano sea difuso, sino porque el mundo es difuso”.

John F. Sowa

DATOS DE CONTACTO DE LOS TUTORES

Nombre y Apellidos: Luis M. Teijón Acosta
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Categoría Docente: Asistente
E-mail: lmteijon@uci.cu
Ocupación Actual: Profesor de Sistemas de Bases de Datos

Nombre y Apellidos: Manuel Villanueva Betancourt
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Máster en Educación
Categoría Docente: Profesor Auxiliar
E-mail: manuelvb@uci.cu
Ocupación Actual: Coordinador del Proyecto TALENMÁTICO

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de _____del año _____.

_____

Firma del Autor

Evelyn López Gámez

_____ _____

Firma del Tutor | Firma del Tutor

Msc. Manuel Villanueva Betancourt | **Ing. Luis M. Teijón Acosta.**

AGRADECIMIENTOS

Quisiera agradecer en general a todos los que de una forma u otra me han ayudado en la realización de este trabajo, en especial:

Al comandante en jefe Fidel Castro por permitirme participar en este maravilloso programa de la Revolución.

A mis padres y a mi hermana Elianne, por todo su cariño y apoyo durante estos cinco años, ya que sin ellos no hubiera podido llegar a ser lo que soy hoy.

A Ramsés, gracias por estar a mi lado en cada uno de los buenos y malos momentos, por darme fuerzas para continuar, por su comprensión y sobre todo por su amor.

A toda mi familia en general, principalmente a los de San José por ayudarme siempre en todo lo que necesité.

A mi suegra, gracias por acogerme como un miembro más de la familia.

A Prevot, Adriel y Papo por ser tan especiales y estar siempre ahí cuando los he necesitado.

A todas las chicas del apartamento muchas gracias por soportarme durante estos cinco años, especialmente a Adriana por brindarme su amistad que tanto aprecio.

A mis tutores por sus consejos, paciencia y apoyo incondicional.

Al tribunal y oponente por sus revisiones y señalamientos.

A todos muchas gracias…

DEDICATORIA

Este trabajo está dedicado en su totalidad:

A mis queridos padres, por ser tan especiales, por apoyarme siempre, por cada uno de los sacrificios que enfrentaron para que pudiera llegar hasta aquí. Por darme las fuerzas necesarias para superar cada momento difícil y por ser mi guía a cada paso, ya que sin ellos nunca hubiera logrado este sueño que es tanto mío como de ellos; los quiero.

RESUMEN

El propósito de este trabajo es diseñar e implementar una base de datos difusa para un sistema evaluador de competencias. La base de datos tendrá como principal objetivo el almacenamiento y procesamiento de la información resultante de la evaluación de estudiantes universitarios. El procesamiento se hará teniendo en cuenta una serie de competencias y/o responsabilidades previamente identificadas, cuya evaluación arrojará resultados imprecisos o cargados de subjetividad que deberán ser procesados teniendo en cuenta el grado de incertidumbre presente en cada uno de ellos. Dicha base de datos será usada por un sistema automático de evaluación de competencias que realizará todo el proceso de gestión de la información.

PALABRAS CLAVE:
Base de datos difusa, evaluación de competencias, FSQL, información imprecisa

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El avance de la sociedad en cada una de las esferas ha traído consigo grandes cambios, en su mayoría bastantes complejos, los cuales requieren de personas preparadas y capaces. Estas personas deben poseer conocimientos y habilidades, en función de crear e innovar, para superar con éxito cada uno de los retos que se presentan día a día, con el fin de promover el desarrollo económico, social y cultural del país. Es por ello que en la actualidad se le otorga un papel importante a la formación de profesionales, al desarrollo de las potencialidades humanas y a los talentos.

El desarrollo del talento en las instituciones docentes cubanas es un reto que asume la educación contemporánea y constituye una necesidad para enfrentar y aumentar el desarrollo de la ciencia y la técnica en la sociedad actual (1).

En casi todos los niveles escolares, fundamentalmente en el nivel superior, se realiza un trabajo diferenciado principalmente con los estudiantes que presentan problemas docentes, y no así con los que tienen mayor desempeño académico que la media estudiantil, a los cuales se les conoce como potencialmente talentosos. Estos estudiantes presentan necesidades educativas especiales que deben ser atendidas debidamente, por lo que resulta imprescindible realizar un trabajo diferenciado con ellos para estimular al máximo sus conocimientos, habilidades y motivación. Para ello el profesor debe conocer las fortalezas y debilidades de sus alumnos, para saber dónde enfatizar el trabajo de formación y así desarrollar las competencias más avanzadas de cada uno, con el fin de que puedan hacer uso de ellas, alcanzando un buen desempeño en el medio donde se desenvuelvan, ya sea como estudiantes, o una vez graduados como profesionales.

En la Universidad de las Ciencias Informáticas (UCI), desde su surgimiento en el 2002, se concibió el Centro de Innovación y Calidad de la Educación (CICE), con la misión de elevar la calidad del proceso de formación del egresado en la universidad, a través de acciones de investigación y formación posgraduada del claustro docente. Este centro se encuentra actualmente trabajando en la evaluación de competencias de los estudiantes de la UCI, principalmente en aquellos que se encuentran en el 1er y 4to año de la carrera. Dicha evaluación se realiza a través de la herramienta LimeSurvey, que es una plataforma web para la administración de encuestas en línea, la cual genera una serie de instrumentos que permiten evaluar a los estudiantes de la universidad.

Este mecanismo presenta varios inconvenientes, y es que dicha herramienta no guarda el resultado de la habilidad evaluada, sino una información general sobre las respuestas que un estudiante determinado dio a un conjunto de preguntas específicas. Esta herramienta es útil para realizar estudios a la masa de estudiantes en general, pero carece de los elementos necesarios para evaluar a los estudiantes de forma individual, de ahí que el mecanismo que usa el CICE no es suficiente para realizar trabajo diferenciado en los estudiantes.

Es por ello que en la facultad 5 de la UCI se creó un proyecto investigativo de innovación pedagógica, cuyo acrónimo es Talenmático, que trabaja en un modelo de atención educativa a los estudiantes potencialmente talentosos, y para lo cual la identificación de potencialidades constituye un aspecto decisivo. Como parte de este proyecto actualmente se trabaja en un sistema que permita la evaluación de competencias en los estudiantes. Este sistema los evalúa midiendo cada una de sus competencias o habilidades tales como: persistencia, motivación, creatividad, aprendizaje, entre otras.

Por tanto, la **situación problémica** de este trabajo es la siguiente:

Las competencias a evaluar en los estudiantes, no siempre podrán medirse de una manera cuantitativa o exacta, en ocasiones será necesario asignarle valores cualitativos o imprecisos que deberán procesarse para completar el proceso de identificación. Este sistema de evaluación de competencias, aún no cuenta con un mecanismo para el almacenamiento y procesamiento de información imprecisa o con cierto grado de incertidumbre, por lo que el proceso de evaluación no es lo suficientemente flexible y es hasta cierto punto limitado.

Teniendo en cuenta la situación problémica planteada anteriormente, se hace necesario desarrollar una base de datos para el sistema de evaluación de competencias, que permita el procesamiento de información imprecisa.

Para ello se ha formulado como **problema a resolver:** ¿Cómo almacenar y procesar información imprecisa que permita evaluar competencias en estudiantes universitarios? Quedando definido como **objeto de estudio:** sistemas de bases de datos que permitan procesar información imprecisa o con cierto grado de incertidumbre.

Se define como **objetivo general:** desarrollar una base de datos para el Sistema de Evaluación de Competencias que permita almacenar y procesar información imprecisa. Enmarcado en el **campo de acción:** sistemas de bases de datos difusas para el procesamiento de información imprecisa.

Para darle cumplimiento al objetivo propuesto se han planificado las tareas de investigación que se relacionan a continuación:

- Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema en la actualidad.
- Realización del levantamiento de requisitos de la BDD.
- Realización de un estudio sobre bases de datos difusas para seleccionar el servidor difuso a utilizar.
- Selección de las herramientas a utilizar para el desarrollo de la base de datos.
- Realización de estudios sobre los patrones de diseño de bases de datos difusas para lograr un mejor diseño.
- Realización del diseño de la base de datos.
- Implementación de la base de datos.
- Validación del resultado obtenido.

Para la realización de estas tareas se han empleado los métodos de investigación que se describen a continuación:

Métodos Teóricos: Permiten conocer las relaciones que fluyen alrededor del objeto de estudio.

- **Análisis-Síntesis:** se requiere para el estudio de la información sobre bases de datos difusas, que permita arribar a ideas generales sobre la base de datos que se desarrollará.
- **Inductivo-Deductivo:** es utilizado partiendo del estudio de bases de datos específicas para poder diseñar la que satisface el propósito del trabajo.
- **Histórico-lógico:** es utilizado para analizar la evolución cronológica y funcional de las soluciones existentes vinculadas al campo de acción, para de esta forma alcanzar el objetivo definido.

Métodos Empíricos: Permiten la observación y el análisis inicial de la información.

- **Consulta de la información en todo tipo de base:** es utilizado para la elaboración del marco teórico de la investigación.
- **Consulta de especialistas:** se realizarán consultas a los integrantes del Proyecto Talenmático, como una vía de validación de los resultados, por ser especialistas y clientes del trabajo.

- **Pruebas informáticas:** permitirán verificar la funcionabilidad de la base de datos.

La **idea a defender** será que aplicada la base de datos difusa desarrollada, que permita el procesamiento de información imprecisa, se facilitará el proceso de evaluación de competencias en estudiantes universitarios.
El **resultado que se espera** de este trabajo es una base de datos difusa para el Sistema de Evaluación de Competencias, que permita a los integrantes del proyecto Talenmático el almacenamiento y procesamiento de información imprecisa.

Estructura del contenido:
Para un mejor entendimiento del trabajo se decide estructurar el mismo de la siguiente manera:

- **Capítulo 1. "Fundamentación Teórica":** En este capítulo se abordan los conceptos necesarios para la comprensión del presente trabajo. Se presenta además la descripción del estado de arte de las bases de datos difusas y las soluciones que existen alrededor del objeto de estudio.
- **Capítulo 2. "Descripción de la solución propuesta":** Se realiza de forma general el análisis, diseño e implementación de la base de datos difusa. Se especifican los requisitos no funcionales de la base de datos, además se incluyen los modelos lógico y físico de la misma, así como la descripción de cada una de las tablas, entre otros aspectos.
- **Capítulo 3. "Validación de la solución propuesta":** En este último capítulo se realiza una validación teórica y funcional de la solución propuesta. Se exponen aspectos como la integridad, seguridad y redundancia de la información. Además se tratan aspectos referentes a las pruebas de rendimiento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo, se definen los conceptos asociados al dominio del problema, se sustenta el problema científico y el propósito de este trabajo mediante el estado del arte, formada por un grupo de referencias seleccionadas, análisis y valoraciones del tema en cuestión. Se abordan aspectos fundamentales sobre las base de datos difusas y se especifican cada una de las herramientas y lenguajes necesarios para su diseño e implementación.

1.2. Conceptos asociados al dominio del problema

Para lograr un mejor entendimiento y desenvolvimiento de los temas que se abordarán en el presente y posteriores capítulos, se mostrarán un grupo de conceptos identificados durante la investigación realizada. Estos conceptos son los siguientes:

Base de Datos (BD)

Se entiende por base de datos un repositorio o conjunto de datos almacenados, normalmente en dispositivos electrónicos de un ordenador, y que son gestionados (para lectura y/o escritura) por un programa llamado Sistema Gestor de Bases de Datos (2).

Una base de datos es una colección de archivos relacionados que permite el manejo de la información de alguna compañía. Cada uno de dichos archivos puede ser visto como una colección de registros y cada registro está compuesto por una colección de campos. Cada uno de los campos de cada registro permite llevar información de algún atributo de una entidad del mundo real (3).

Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto (4).

Una BD es un “almacén” que permite al usuario guardar grandes cantidades de información de forma organizada, para que resulte mucho más fácil la búsqueda y el uso de la misma.

Sistema Gestor de Base de Datos (SGBD)

El Sistema Gestor de Base de Datos es una aplicación que permite a los usuarios definir, crear y mantener la BD, además proporciona un acceso controlado a la misma (5).

El SGBD maneja las solicitudes de acceso a la base de datos por parte de los usuarios, ocultando a estos, detalles sobre el hardware donde los datos están almacenados (2).

Entre las ventajas de los SGBD se encuentran las siguientes (6):

- Proveen facilidades para la manipulación de grandes volúmenes de datos.
  - Garantizan la privacidad de los datos y la eficiencia en su acceso.
  - Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la BD serán siempre consistentes sin importar si hay errores.
  - Organizan los datos con un impacto mínimo en el código de los programas.
  - Bajan drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

Entre los SGBD más utilizados y difundidos en la actualidad se encuentra el MySQL, PostgreSQL, Oracle y Microsoft SQL Server (6).

Difuso (Fuzzy)

Originalmente el término *fuzzy* procede de *fuzz*, que sirve para denominar la pelusa que recubre el cuerpo de los polluelos al poco tiempo de salir del huevo. Este término significa “confuso, borroso, no definido o desenfocado”. La traducción de esta palabra al castellano es difuso o borroso, aunque *fuzzy*, en los ámbitos académicos y tecnológicos, está aceptado tal cual, de forma similar a como los es *“bit”*. *Fuzzy* significa ambiguo o vago, en el sentido del razonamiento humano, más que en la acepción de probabilidad de algo (7).

Lógica difusa (Fuzzy Logic)

La lógica difusa es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada vaga, ambigua, imprecisa, con ruido o incompleta, en general la lógica difusa imita como una persona toma decisiones basada en información (7).

Según el profesor L.A. Zadeh en un artículo titulado *“Fuzzy Sets”* (Conjuntos Difusos), las características principales de la lógica difusa son (8):

- En la lógica difusa, el razonamiento exacto es considerado como un caso particular del razonamiento aproximado.
- Cualquier sistema lógico puede ser trasladado a términos de lógica difusa.
- En lógica difusa, el conocimiento es interpretado como un conjunto de restricciones flexibles, es decir, difusas, sobre un conjunto de variables.
- La inferencia es considerada como un proceso de propagación de dichas restricciones.

La lógica difusa permite trabajar no sólo con métodos cuantitativos sino también cualitativos, se trata pues de un intento de aplicar una forma más humana de pensar en la programación de computadoras (7).

Base de Datos Difusa (BDD)

Las bases de datos difusas nacen de unir la teoría de bases de datos, principalmente del modelo relacional, con la teoría de conjuntos difusos, para permitir básicamente dos objetivos (2):

- El almacenamiento de información difusa (además de información no difusa)
- El tratamiento y consulta de esta información de forma difusa o flexible.

Es una base de datos dotada de imprecisión e incertidumbre. Esto quiere decir que se da redundancia de datos en las relaciones. Un sistema basado en lógica difusa es utilizado como herramienta para presentar diferentes maneras de resolver el problema a tratar (2).

Este tipo de bases de datos, surge ante la necesidad de procesar información imprecisa e incierta, ya que las bases de datos tradicionales no permiten el almacenamiento de conceptos difusos, conceptos que los humanos manejan de forma cotidiana y natural (2).

Por ejemplo, *alto* o *caro*, tienen significados diferentes de acuerdo al contexto en el que se esté utilizando, e incluso dentro del mismo contexto pueden significar cosas diferentes para diferentes individuos.

1.3. Estado del arte de las BDD

El estudio de las bases de datos difusas surge como una extensión del estudio de las bases de datos relacionales. En los últimos años, algunos investigadores han lidiado con el problema de relajar el modelo relacional para permitirle admitir algunas imprecisiones; lo que conduce a sistemas de bases de datos difusas, ya que permiten el manejo de información con una terminología que es muy similar a la del lenguaje natural (7).

A continuación se describen los aspectos más significativos de este tipo específico de base de datos.

Lenguajes de consultas para BDD

Como bien se planteó anteriormente, el estudio de las bases de datos difusas surge con el objetivo de permitir el tratamiento de información imprecisa o difusa. Para ello algunos investigadores han propuesto varias extensiones del lenguaje SQL (Lenguaje de Consulta Estructurado, Structured Query Language), un ejemplo de estas propuestas es el lenguaje FSQL y el SQLF.

1.3.1. FSQL

FSQL son las siglas de *Fuzzy Structured Query Language*, o *Fuzzy SQL*, es decir, lenguaje SQL difuso. Surge de modificar el lenguaje SQL para adaptarlo a las necesidades de una BDD, de forma que permita expresar información difusa (14).

El lenguaje FSQL es una auténtica extensión de SQL. Esto significa que todas las sentencias válidas en SQL lo son también en FSQL (2).

De FSQL se extienden:

- **DML (Lenguaje de Manipulación de Datos):** Las sentencias de este lenguaje permiten la consulta y la modificación de los datos almacenados en la BD. Un ejemplo de comandos del DML SQL son: SELECT, INSERT, DELETE y UPDATE (2).
- **DDL (Lenguaje de Definición de Datos):** Las sentencias de este lenguaje permiten la creación y modificación de las estructuras en las que se almacenarán los datos. Los comandos del DDL SQL son por ejemplo: CREATE, DROP, ALTER y sentencias para controles de seguridad y control del almacenamiento físico de los datos (2).

FSQL fue ampliamente definido e implementado por el doctor José Galindo en su tesis de doctorado, donde implementó algunas extensiones al lenguaje SQL, que pueden ser usadas a través de un cliente para el trabajo con consultas difusas.

FSQL puede describirse como la adición de nuevas tablas y paquetes al catálogo del SGBD. Permite además el uso de operadores difusos, atributos difusos, grados difusos, así como el

trabajo con etiquetas lingüísticas y trapecios probabilísticos; cada uno de estos aspectos serán tratados posteriormente.

1.3.2. SQLF

SQLF, es una extensión del lenguaje SQL, publicado por Patrick Bosc y Oliver Pivert en 1995. Entre las extensiones planteadas por este lenguaje se encuentra la cuantificación difusa, esta extensión brinda la definición para operar en una base de datos con etiquetas lingüísticas. El SQLF representa una síntesis de las características y funcionalidades sugeridas en otras propuestas de consulta flexible en bases de datos clásicas (14).

El formato de una consulta SQLF se basa en el de una consulta SQL (15):

```
SELECT [ N | T | N,T ] <lista de selección>
FROM <lista de tablas>
WHERE <condición difusa>
```

Para recuperar la relación resultante, se aplica la condición difusa al producto cartesiano de todas las tablas de la lista, se proyecta sobre los atributos de la lista de selección y se devuelven solo las N mejores tuplas y/o aquellas que tengan su umbral (threshold) de cumplimiento mayor que T. Los valores N y T son opcionales y se puede especificar uno de ellos o los dos (15).

1.3.3. Modelos teóricos de BDD

Se han encontrado algunos modelos para el tratamiento de la incertidumbre como por ejemplo: La aproximación de Codd (valores nulos), Modelos estadísticos y probabilísticos, Modelos básicos de bases de datos difusas (añaden simplemente un grado difuso), Modelo de Buckles-Petry, Modelo de Prade Testemale, Modelo de Umano-Fukami, Modelo de Zemankova-Kaendel, Modelo GEFRED de Medina.

Estos modelos permiten almacenar y/o tratar información imprecisa o incierta y han sido utilizados en aplicaciones de bases de datos difusas (7).

Consideran además el concepto de relaciones difusas debido a su implementación en bases de datos relacionales. Sin embargo, existen varias formas de permitir la información imprecisa o incierta en entidades o interrelaciones difusas. Además, esas formas pueden ser mezcladas para permitir una mayor flexibilidad (7).

Algunas de las más importantes formas de modelar información difusa en las entidades o interrelación son (2):

- **Valores difusos en los atributos:** Esto se refiere a que en los atributos de una entidad se pueden contener valores difusos y también operar con ellos. Los tipos de estos valores pueden ser muy variados, tales como: valor nulo, valor desconocido, valor indefinido, distribución de posibilidad, función de similitud, entre otros (2).
- **Grado en cada valor de un atributo:** Esto implica que cada valor de un atributo puede tener asociado un grado, generalmente en un intervalo [0,1], que mida el nivel de difuminado de dicho valor. El significado de estos grados puede ser variado (7).
- **Grado en toda la instancia de la entidad**: Esto es similar al caso anterior, pero aquí el grado está asociado a toda la instancia de la entidad y no exclusivamente a un valor particular de una instancia. Puede medir el grado con que esa instancia (o tupla) pertenece a esa relación (o tabla) de la base de datos (7).
- **Grado en un conjunto de valores de diversos atributos**: Este es un caso intermedio entre los casos anteriores. Aquí el grado está asociado a algunos atributos. Este puede ser un caso poco usual pero puede ser a veces muy útil (2).

El dominio de estos grados se puede encontrar en un intervalo [0,1], pero también permiten otros valores, como por ejemplo una distribución de posibilidad. Además el significado de esos grados es variado. Dependiendo de este significado el tratamiento de los datos podrá ser diferente (7).

Los posibles significados más importantes de los grados son los siguientes:

- **Grado de cumplimiento (satisfacción):** Una propiedad puede cumplirse con cierto grado entre dos extremos. La propiedad se cumple totalmente (usualmente grado 1), y la propiedad no se cumple en absoluto (usualmente grado 0). Esto suele emplearse tras establecer alguna condición sobre los valores de una entidad o interrelación y los grados expresarán en qué medida esa condición ha sido satisfecha (7).
- **Grado de incertidumbre**: El grado de incertidumbre expresa la seguridad con que se conoce un dato determinado. Si se está seguro de la veracidad de dicho grado, este será 1 y si se está seguro de su falsedad dicho grado será 0. Los valores entre 0 y 1 expresan distintos niveles de incertidumbre, indicando que no se está completamente seguro. Este significado es en cierta forma bastante parecido al anterior, pues puede

extenderse esa incertidumbre como expresada sobre la pertenencia de una instancia (tupla) a la relación (tabla) de la base de datos. (7).

- **Grado de posibilidad**: Mide la posibilidad de la información utilizada. Este significado es similar al anterior, pero se ve este grado como más débil, ya que contiene información que es más o menos posible y no más o menos cierta (7).
- **Grado de importancia**: Distintos objetos (instancias, atributos, etc.) pueden tener diferentes importancias, de forma que existan objetos más importantes que otros (7).

1.3.4. BDD: FIRST

El Dr. Medina en 1994, expuso un módulo para permitir extender la capacidad de un SGBD clásico para que pueda representar y manipular información "imprecisa". Este módulo, llamado FIRST (Fuzzy Interface for Relational SysTems, Interfaz Difuso para Sistemas Relacionales), utiliza GEFRED como modelo teórico y los recursos del modelo relacional clásico para poder representar este tipo de información (9).

Fundamentos de FIRST

- **FMB (Fuzzy Metaknowledge Base, Base de Metaconocimiento Difuso):** Es el "diccionario" o "catálogo del sistema" de un SGBD representa aquella parte del sistema que almacena información sobre los datos recogidos en la base de datos, así como otro tipo de informaciones: usuarios, permisos, accesos, datos de control, entre otros (2).
- **Servidor FSQL:** Su objetivo es captar las sentencias en lenguaje FSQL y traducirlas a un lenguaje que entienda el SGBD, el lenguaje SQL. Para efectuar esta traducción utilizará la información almacenada en la FMB (2).
- **Cliente FSQL:** Se trata de un programa que hace de interfaz entre el hombre (u otro programa) y el Servidor FSQL. Este programa puede ser muy simple, pues el trabajo principal de una operación FSQL será efectuado por el Servidor FSQL. El programa Cliente FSQL puede ser programado para cualquier Sistema Operativo y en cualquier lenguaje de programación (2).

1.3.4.1. Tablas de la FMB

La FMB almacena información sobre los datos difusos en forma relacional.

OBJ#, COL#: Son dos atributos que almacenan números que identifican una columna concreta dentro de una tabla concreta (2).

- Todas las tablas de la FMB tienen esos dos atributos como llave primaria (o como parte de ella).
- Oracle identifica cada tabla del sistema con un número OBJ#, que se puede obtener consultando la tabla USER_OBJECTS.
- Dentro de una tabla se identifica cada columna con un número COL# que puede consultarse en USER_TAB_COLUMNS.

Entre las tablas más importantes de la FMB, se encuentran las siguientes (Anexo 1):

- **FUZZY_COL_LIST:** Almacena la descripción de los atributos difusos:
  - OBJ#, COL#: Ubicación del atributo difuso.
  - F_TYPE: Tipo de atributo difuso (1, 2 ó 3).
  - LEN: Longitud de una distribución de posibilidad en atributos Tipo 3 (máximo 10).
  - COM: Comentario (usualmente se pone el nombre del atributo).

- **FUZZY_OBJECT_LIST:** Almacena la descripción de los objetos difusos definidos para cada atributo (OBJ#, COL#):
  - FUZZY_ID: Identificador del objeto difuso.
  - FUZZY_NAME: Nombre del objeto (sin espacios).
  - FUZZY_TYPE: Tipo del objeto (etiquetas trapezoidales, etiquetas de Tipo 3, cualificadores, cuantificadores relativos y absolutos).

- **FUZZY_LABEL_DEF:** Se definen cada una de las etiquetas trapezoidales.
  - FUZZY_ID: Identificador del objeto difuso.
  - ALFA, BETA, GAMMA, DELTA: Los cuatro valores del trapecio.

- **FUZZY_APPROX_MUCH:** Valores para el margen (MARGEN) y el valor mínimo o distancia mínima, para considerar dos valores (MUCH) para atributos difusos Tipo 1 ó 2.

- **FUZZY_NEARNESS_DEF:** Medidas de proximidad, semejanza o similitud (DEGREE) entre cada dos etiquetas de un atributo difuso Tipo 3 (FUZZY_ID1, FUZZY_ID2).

Esquema de las Tablas de la FMB (ver Figura 1):

- **Llaves o Claves:**
  - **Primarias:** Subrayadas.
  - **Externas:** Con flechas.
- **OBJ#:** Identifica un objeto concreto de la BD (como una tabla).
- **COL#:** Identifica un atributo concreto dentro del objeto OBJ#.
- **FUZZY_ID:** Identifica distintos objetos definidos para un atributo particular (como una etiqueta).

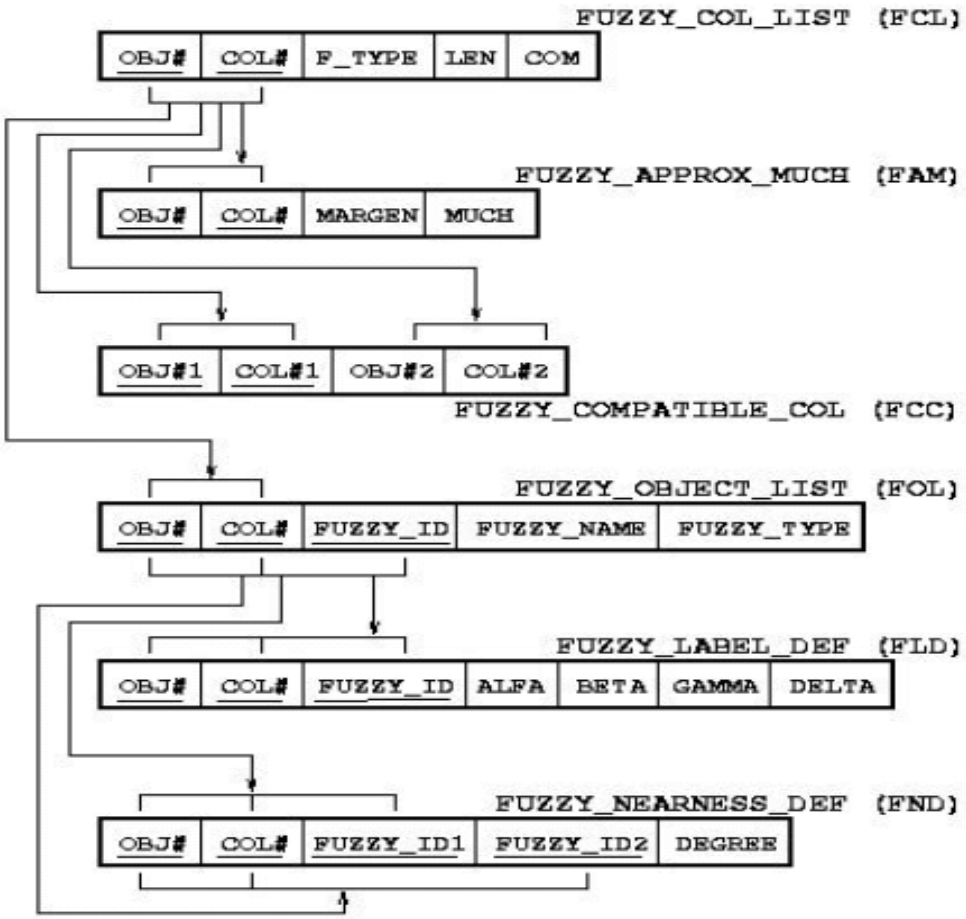


Figura. 1 Esquema gráfico de las tablas de la FMB (2).

Los atributos de esta tabla tienen el siguiente significado (2):

- **OBJ#:** Almacena el número de objeto de la tabla (o vista) que tiene un atributo difuso.
- **COL#:** Almacena el número de columna dentro de la tabla que admitirá un tratamiento difuso.

- **F TYPE:** Almacena el Tipo de atributo difuso de la columna identificada por (OBJ#, COL#). Este Tipo puede tomar el valor 1, 2 ó 3, y ello es comprobado por la restricción FUZZY_TYPE_MUST_BE.
- **LEN:** Almacena la longitud (length) máxima de una distribución de posibilidad en atributos Tipo 3, el número máximo de parejas (valor de posibilidad, etiqueta) que admite una distribución de posibilidad en este atributo.
- **COM:** Almacena un comentario opcional sobre el atributo. En general es útil para poner, por ejemplo, el nombre de la tabla y el atributo, de forma que sea fácil descubrir qué Tipo difuso tiene un atributo particular, sin tener que saber su pareja de números (OBJ#, COL#).

1.3.5. Clasificación de los Atributos Difusos

Los atributos difusos se consideran datos con dominio de referencial ordenado y no ordenado. La clasificación adoptada se basa en criterios de representación y de tratamiento de los datos "imprecisos", se clasifica según el tipo de dominio que les subyace y por si permiten representar la información imprecisa o sólo permiten el tratamiento impreciso de datos sin imprecisión (9).

Los atributos difusos pueden ser de 3 tipos (2):

- **Tipo 1 (crisp):** Estos son atributos con "*datos precisos"*, clásicos o *crisp* (tradicionales, sin imprecisión), que pueden tener etiquetas lingüísticas definidas sobre sus dominios. Estos tipos de atributos reciben una representación igual que los datos precisos, pero admiten que se puedan utilizar en consultas difusas, utilizando o no las etiquetas definidas en su dominio.
- **Tipo 2 (probabilísticos):** Admiten datos tradicionales y difusos como distribuciones de probabilidad sobre un dominio ordenado. Los valores de este tipo de atributos difusos pueden ser las propias etiquetas lingüísticas, distribuciones de posibilidad trapezoidales, entre otras.

Para un atributo difuso Tipo 2 llamado F, la representación usa 5 atributos: FT para almacenar el código de tipo que le corresponde a cada valor y los atributos F1, F2, F3 y F4 para almacenar los parámetros de cada dato. Los valores NULL que aparecen en los atributos tienen el significado de valor "no-aplicable" en el SGBD anfitrión (2).

Tipos de Valores	Atributos de la BD para cada Tipo 2				
	FT	F1	F2	F3	F4
UNKNOWN	0	NULL	NULL	NULL	NULL
UNDEFINED	1	NULL	NULL	NULL	NULL
NULL	2	NULL	NULL	NULL	NULL
CRISP	3	d	NULL	NULL	NULL
LABEL	4	FUZZY_ID	NULL	NULL	NULL
INTERVALO [n,m]	5	n	NULL	NULL	m
APROXIMADAMENTE(d)	6	d	d - margen	d + margen	margen
TRAPECIO[α,β,γ,δ]	7	α	β - α	γ - δ	δ

Tabla. 1 Representación interna de atributos difusos Tipo 2 (2).

En la Tabla.1 se muestra el sistema utilizado para representar a los atributos difusos Tipo 2. Por ejemplo, un atributo difuso Tipo 2, llamado F, está compuesto, de hecho, por 5 atributos clásicos:

- **FT:** Almacena el tipo de valor que corresponde al dato que se quiere almacenar, indicando su representación. Según lo visto, puede ser: UNKNOWN (0), UNDEFINED (1), NULL (2), CRISP (3), LABEL (4), INTERVALO (5), APROXIMADAMENTE (6) o TRAPEZOIDAL (7). Observe que se añadió una T al nombre del atributo (2).
- **F1, F2, F3 y F4:** Los atributos cuyo nombre se forma añadiendo los números 1, 2, 3 y 4 al nombre del atributo almacenan la descripción de los parámetros que definen el dato y que depende del tipo de valor (FT) al que pertenezca (2):
  - **UNKNOWN, UNDEFINED, NULL:** Estos 3 valores no necesitan ningún parámetro, por lo que todos ellos permanecen a NULL.
  - **CRISP:** Un valor de tipo crisp, necesita tan solo un parámetro, F1, en el cual se almacenará el valor crisp en cuestión.
  - **LABEL:** Igualmente, un valor de tipo etiqueta solo necesita un parámetro para almacenar el identificador asociado a dicha etiqueta (FUZZY_ID). Ese indicador es útil para poder acceder a la FMB y obtener la descripción asociada a esta etiqueta.
  - **INTERVALO:** Necesita los dos valores extremos del intervalo [n,m], que son almacenados en F1 y F4 respectivamente.

- **APROXIMADAMENTE:** Este valor solo necesita un valor que se almacena en F1 y que es el valor central de la distribución de posibilidad triangular, d. Sin embargo, para reducir operaciones (tanto matemáticas como de acceso a datos), se aprovechan los atributos F2, F3 y F4 para almacenar los valores siguientes: d-margen, d+margen y margen, respectivamente. El valor margen es un valor almacenado en la FMB para cada atributo difuso, y su valor depende del significado de dicho atributo.
- **TRAPECIO:** Necesita forzosamente almacenar los 4 valores que identifican a un trapecio: [α, β, γ, δ]. En F2 y F3 se almacenan unas operaciones que simplifican las ecuaciones cuando se opera con este tipo de dato.

- **Tipo 3 (escalares):** Datos definidos sobre un dominio discreto no ordenado con analogía. En estos atributos se definen algunas etiquetas (“rubio”, “moreno”, “pelirrojo”, etc.) que son escalares con una relación de similitud (proximidad, μ) definida sobre ellas, de forma que esta relación indique en qué medida se parecen entre sí cada par de etiquetas, ejemplo: μ(rubio, pelirrojo)=0.8, μ(rubio, moreno)=0.

A continuación se muestran algunos ejemplos de atributos difusos:

A	Un escalar simple (ej. Tamaño = Grande)
B	Un número simple (Ej. Edad = 28)
C	Un conjunto de posibles asignaciones excluyentes de escalares (Ej. Aptitud = {Mala, Buena})
D	Un conjunto de posibles asignaciones excluyentes de números (Ej. Edad = {20,21})
E	Una distribución de posibilidad en el dominio de los escalares (Ej. Aptitud = {0.6/Mala, 1.0/Regular}.
F	Una distribución de posibilidad en el dominio de los números (Ej. Alto = {0/185, 1.0/195, 1.0/205, 0/215}, números difusos, etiquetas lingüísticas, etc.).
G	Un número real perteneciente a [0, 1] representando el grado de cumplimiento (Ej. Calidad = 0.9).
H	Un valor desconocido UNKNOWN dado por la distribución de posibilidad

	UNKNOWN = {1/*u* : *u* ∈ *U*} sobre el dominio, *U,* considerado.
I	Un valor indefinido UNDEFINED dado por la distribución de posibilidad UNDEFINED = {0/*u* : *u* ∈ *U*}sobre el dominio U, considerado.
J	Un valor nulo dado por la expresión Null = {1/UNKNOWN, 1/ UNDEFINED}.

Tabla. 2 Ejemplo de datos imprecisos (7).

1.3.6. Distribución de Probabilidad Trapezoidal

Esta representación determina la función de pertenencia asociada al dato mediante el uso de cuatro parámetros [α, β, γ, δ]. Se utilizan funciones de pertenencia normalizadas, es decir, aquellas que poseen un núcleo no vacío. Se hace uso de la función de pertenencia trapezoidal (figura 2), ya que se adapta bastante bien a la definición de cualquier concepto, con la ventaja de su fácil definición, representación y simplicidad de cálculos (11).

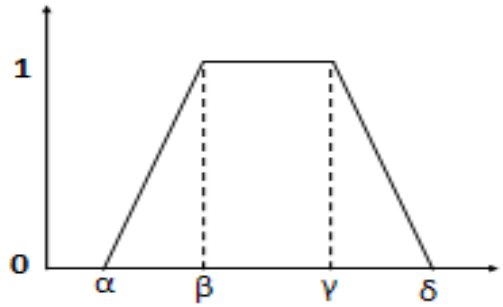


Figura. 2 Formato de una distribución de posibilidad trapezoidal (2).

1.3.7. Etiquetas Lingüísticas

Se le llama *etiqueta lingüística* a aquella palabra, en lenguaje natural, que expresa un conjunto difuso, que puede estar formalmente definido o no (2).

Las etiquetas lingüísticas es uno de los conceptos básicos en lógica difusa. Si un atributo es susceptible de tratamiento difuso entonces pueden definirse etiquetas sobre él. Estas etiquetas son precedidas, por convenio, por el símbolo $ para distinguirlas fácilmente de otros identificadores (11).

Existen 2 tipos de etiquetas que serán usadas en cada uno de los tipos de atributos difusos:

- **Etiquetas para atributos con un dominio subyacente ordenado:** Cada etiqueta de este tipo tiene asociada en la FMB, una distribución de posibilidad trapezoidal como la de la Figura 2.

  Ejemplo, el atributo $Alto puede ser definido como una distribución de posibilidad con los siguientes valores (en centímetros) α=185, β=195, γ=205 y δ=215, como se muestra en la Figura 3.

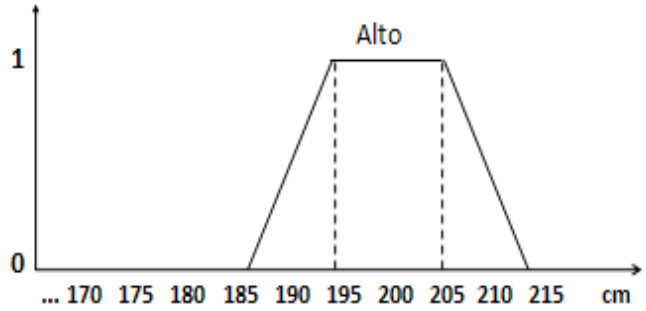


Figura. 3 Ejemplo de una etiqueta lingüística para el concepto "Alto".

- **Etiquetas para atributos con un dominio subyacente no ordenado (escalares):** Puede existir una relación de similitud definida entre cada dos etiquetas del dominio y almacenada en la FMB. El grado de similitud será siempre un valor del intervalo [0,1].

1.3.8. Comparadores Difusos

Además de los comparadores clásicos típicos (>, =, <.), FSQL incluye los *comparadores difusos de posibilidad* (Tabla 3) y los *comparadores difusos de necesidad* (Tabla 4).

Comparador difuso	Significado
FEQ	Fuzzy EQual: Posiblemente Igual.
FGT	Fuzzy Greater Than: Posiblemente Mayor.
FGEQ	Fuzzy Greater or EQual: Posiblemente Mayor o Igual.
FLT	Fuzzy Less Than: Posiblemente Menor.
FLEQ	Fuzzy Less or EQual: Posiblemente Menor o Igual.
MGT	Much Greater Than: Posiblemente Mucho Mayor.
MLT	Much Less Than: Posiblemente Mucho Menor.

Tabla. 3 Comparadores difusos de posibilidad de FSQL (11).

Comparador difuso	Significado
NFEQ	Necessarily Fuzzy EQual: Necesariamente Igual.
NFGT	Necessarily Fuzzy Greater Than: Necesariamente Mayor.
NFGEQ	Necessarily Fuzzy Greater or EQual: Necesariamente Mayor o Igual.
NFLT	Necessarily Fuzzy Less Than: Necesariamente Menor.
NFLEQ	Necessarily Fuzzy Less or EQual: Necesariamente Menor o Igual.
NMGT	Necessarily Much Greater Than: Necesariamente Mucho Mayor.
NMLT	Necessarily Much Less Than: Necesariamente Mucho Menor.

Tabla. 4 Comparadores difusos de necesidad de FSQL (11).

Como en SQL, los comparadores difusos pueden comparar una columna de una tabla con una constante o dos columnas del mismo tipo o de tipos compatibles (2).
Los *comparadores de posibilidad* son más generales (menos restrictivos) que los de necesidad. Por tanto, los *comparadores de necesidad* recuperan menos instancias y estas instancias cumplirán "necesariamente" con las condiciones impuestas en la consulta (2).

El comparador difuso de "desigualdad" o "posiblemente distinto" no se ha considerado porque puede modelarse negando una comparación con FEQ (2):

```
NOT <Atributo_Difuso> FEQ <Atributo_o_Cte>
```

En atributos con dominio subyacente no ordenado o sea, en los atributos difusos Tipo 3, sólo puede usarse el comparador difuso FEQ, puesto que carecen de orden (7).

1.3.9. Umbral de Cumplimiento

Para cada condición simple se puede establecer un umbral de cumplimiento (por defecto será 1) con el formato siguiente:

```
<Condición_simple> THOLD τ
```

De esta forma se indica que la condición debe ser satisfecha con un grado mínimo de $\tau \in [0; 1]$ para que la tupla en cuestión aparezca en la relación resultante.
La palabra reservada THOLD es opcional y puede ser sustituida por cualquier comparador crisp tradicional (=, ≤, ...), modificando así, lógicamente, el significado de la consulta. La palabra THOLD es equivalente a usar el comparador tradicional ≥ (11).

Ejemplo 1.1: Devolver todas las personas con pelo rubio (en grado mínimo 0.5) que son posiblemente más altas que la etiqueta $Alta (en grado mínimo 0.8):

```
SELECT * FROM Personas
WHERE Pelo FEQ $Rubio THOLD 0.5 AND Altura FGT $Alta THOLD 0.8
```

Si se buscan las personas que son “necesariamente” más altas que la etiqueta $Alta, entonces se debe usar el comparador difuso NFGT en vez de FGT.

1.3.10. Cuantificadores difusos

Los cuantificadores difusos o lingüísticos, permiten expresar cantidades o proporciones difusas para dar una idea aproximada del número de elementos de un subconjunto (o que cumplen cierta condición), o de la proporción de ese número en relación con el total de elementos posibles (7).

Los cuantificadores pueden clasificarse en absolutos o relativos:

- **Cuantificadores absolutos:** expresan cantidades sobre el número total de elementos de un determinado conjunto, diciendo si este número es “grande”, “muchísimos”, “aproximadamente entre 5 y 10”, etcétera (7).
- **Cuantificadores relativos:** expresan mediciones sobre el número total de elementos que cumplen cierta característica dependiendo del total de elementos posibles, por lo que la verdad del cuantificador depende de dos cantidades. Este tipo de cuantificadores se usa en expresiones como “la mayoría”, “la minoría”, “aproximadamente 40 años” (7).

1.4. Lenguajes y herramientas a utilizar

Para el desarrollo de la BDD es necesaria la utilización de varias herramientas y lenguajes, cada una de ellos ocupará un lugar importante en este proceso de desarrollo, ya que con su ayuda podrá ser posible el cumplimiento del objetivo propuesto. Estas herramientas y lenguajes son los siguientes:

1.4.1. Lenguaje de Modelado. UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software (12).

UML captura decisiones y conocimiento sobre los sistemas que se deben construir, su objetivo es lograr que, además de describir con cierto grado de formalismo tales sistemas, puedan ser entendidos por los usuarios aquello que se modela. Es un lenguaje gráfico con sintaxis y semántica bien definidas. Este lenguaje reúne las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos (13).

1.4.2. Lenguaje de consulta difuso. FSQL

Como bien se mencionó anteriormente, dicho lenguaje permite el tratamiento de bases de datos difusas, usando la lógica difusa creada por Lofti A. Zadeh (2).

Por todas sus características antes expuestas, se escoge el lenguaje FSQL como lenguaje de consulta estructurado difuso a utilizar en el desarrollo de la base de datos, ya que dicho lenguaje es bastante similar al SQL, pues incluye todas las sentencias propias de este lenguaje y además permite el manejo de comparadores difusos, funciones difusas, diversos tipos de datos difusos (incluyendo fechas), entre otros aspectos que facilitarán y harán posible el desarrollo de la base de datos difusa para el Sistema de Evaluación de Competencias.

1.4.3. Herramienta CASE. DBDesigner

DBDesigner es un sistema de diseño de base de datos disponible, gratuita y libre, que integra diseño, modelado, creación y mantenimiento de bases de datos en un ambiente de desarrollo único y compacto (16).

El DBDesigner combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos. Actualmente se encuentra en la versión 4, y permite ser descargado gratuitamente desde su web oficial. Esta herramienta soporta los sistemas operativos Windows 2000, XP y GNU/Linux y se puede acceder a la descarga gratis bajo la licencia GNU GPL. Esta herramienta dispone además de una interfaz de usuario clara y sencilla para ofrecer de manera más eficiente el manejo de las BD (17).

Algunas características de DBDesigner (16):

- Guarda los proyectos en XML.
- Brinda la posibilidad de conectividad con otros SGBD a través de plug-ins añadibles (por defecto MySQL y PostgreSQL)

- Conectividad con el "backend" de la base de datos
- Exportar / Importar scripts.

1.4.4. Servidor difuso a utilizar. Servidor FSQL

El doctor José Galindo, en su tesis doctoral, implementó un Servidor FSQL multiusuario, que está disponible para bases de datos Oracle y permite consultar tanto bases de datos relacionales clásicas (tradicionales) como difusas, con el lenguaje FSQL.

El Servidor FSQL es un conjunto de tablas y procedimientos almacenados (programados en PL/SQL) que permite acceder a la Base de Datos con el lenguaje de consulta difuso FSQL.

El objetivo de este servidor es captar y ejecutar las sentencias en el lenguaje FSQL y traducirlas a un lenguaje que entienda el SGBD, el lenguaje SQL. Para efectuar esta traducción utilizará la información almacenada en la FMB programado en PL/SQL para dicho SGBD, y que se encuentra implantado como una colección de módulos almacenados en el Servidor Oracle (2).

Este servidor ha sido diseñado para trabajar tanto con BDD como con bases de datos tradicionales, de forma que permite la incorporación rápida de las ventajas de las consultas flexibles en bases de datos tradicionales preexistentes. El Servidor FSQL podrá ser instalado en cualquier plataforma donde exista una implementación de Oracle (10).

Arquitectura de la BDD con el Servidor FSQL (2):

1. **Base de datos:**
   - Tradicional (crisp)
   - FMB
2. **Servidor FSQL:** Hace posible el uso del lenguaje FSQL en el SGBD tradicional.
   - Está programado en lenguaje PL/SQL, un lenguaje inmerso del SGBD Oracle que permite programar aplicaciones eficientes.
3. **Cliente FSQL:** Es un programa independiente que sirve de interfaz entre el usuario y el Servidor FSQL.

El servidor utilizará el modelo teórico antes mencionado, GEFRED, que es un modelo de bases de datos difusas. La implementación de este modelo se encargará de gestionar las consultas FSQL y traducirlas a SQL (mediante funciones implementadas en el servidor y llamadas desde el cliente).

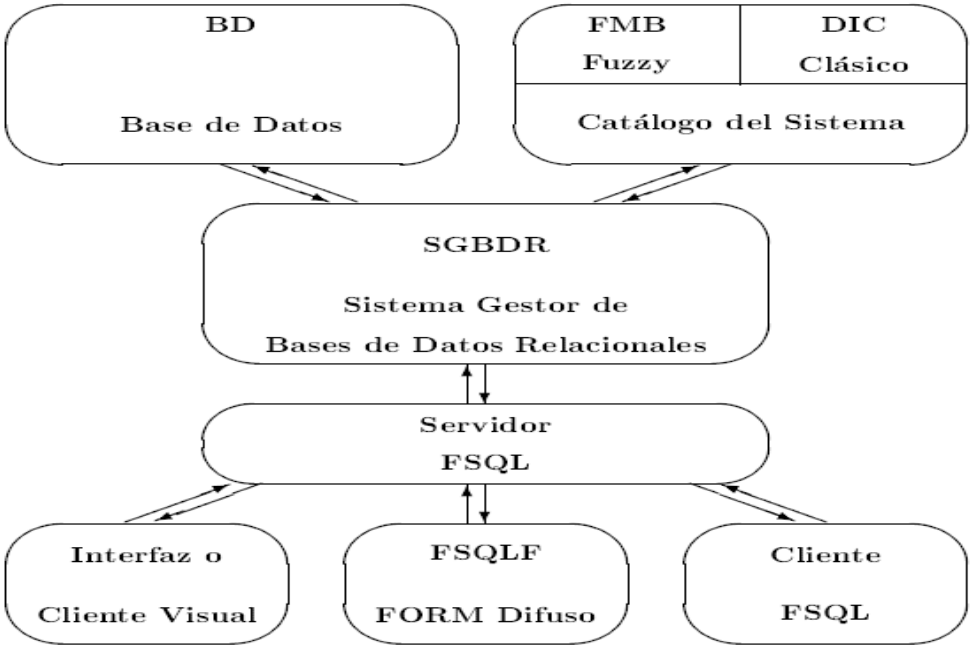


Figura. 4 Arquitectura del Servidor FSQL.

1.4.5. Sistema Gestor de Base de Datos. Oracle 10g

Es un sistema de gestión creado por Oracle Corporation. Se considera uno de los SGBD más completos existentes. Oracle está implementado en cerca de 100 plataformas distintas incluyendo MS-DOS, Windows, OS/2, Macintosh, Sun, MIPS y muchas otras basadas en sistemas Unix, como Linux. Oracle siempre ha sido el SGBD con mejor tiempo de ejecución, pero además, Oracle incorpora un lenguaje procedural, el PL/SQL, que es una extensión de SQL y que permite crear procedimientos y funciones almacenadas en el SGBD. Los paquetes PL/SQL son conjuntos de funciones y/o procedimientos. Oracle dispone de acceso a través de ODBC y multitud de aplicaciones para los más diversos fines SQL*Net, SQL*Plus, Developer 2000, etc (18).

Entre las múltiples ventajas de este SGBD se encuentran las siguientes (2):

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Uno de los principales motivos por el cual fue escogido Oracle como SGBD a usar, es que el Dr. José Galindo para poder hacer uso del lenguaje creado por él mismo, el FSQL, se vio en la necesidad de crear un Servidor FSQL, implantado de acuerdo con el modelo Cliente/Servidor para este SGBD, por ser uno de los más potentes y difundidos en la actualidad.
Se probó también realizar la BDD usando PostgreSQL y se comprobó que resultó ser muy difícil la instalación y manejo de la misma. Además muchas de las funciones para este tipo de BD se encontraban incompletas, y sin ellas el trabajo hubiese resultado más complicado.

1.4.6. Cliente para Oracle. PLSQL

El PLSQL es una extensión de programación a SQL, además de ser el lenguaje de programación de 5ta generación para base de datos Oracle, soporta todas las consultas, ya que la manipulación de datos que se usa es la misma que en SQL (19).
Todo código PLSQL se compone de código PLSQL + sentencias SQL. Donde este código es ejecutado en un motor llamado PLSQL y las secciones que son sentencias SQL son ejecutadas en el SQL Statement Executor (Oracle Database Server). Toda base de datos Oracle tiene un motor PLSQL de forma inherente. Existen otros productos que cuentan con este motor como el Oracle Application Server en sus productos (Oracle Forms, Oracle Reports) (19).

Algunas de las ventajas del PLSQL son las siguientes (19):

- Permite crear programas modulares.
- Integración con herramientas de Oracle.
- Portabilidad.
- Maneja Excepciones.

1.4.7. Cliente FSQL. FQ

FQ es un sencillo programa Cliente FSQL para el SGBD Oracle, creado por el Dr. José Galindo. Está diseñado para efectuar consultas clásicas o difusas a una BD Oracle (20).
FQ permite utilizar el lenguaje FSQL para expresar consultas difusas o flexibles (con atributos difusos, condiciones difusas, umbrales de cumplimiento, grados de cumplimiento de cada tupla, etc.). También admite consultas en SQL estándar. Incorpora menús para insertar en las consultas elementos como: operadores lógicos, operadores de conjuntos, comparadores, etc. Muestra fácilmente las etiquetas lingüísticas definidas para un atributo (columna) concreto.

Las consultas en FQ se expresan en modo texto y en lenguaje SQL o FSQL. Dispone de un icono que informa rápidamente al usuario si se está o no conectado al SGBD (20).
FQ tiene diversas opciones de configuración, teclas de acceso rápido para las acciones más comunes y multitud de ToolTipText (textos explicativos que aparecen al situar el cursor sobre algún objeto u opción concreta). Ofrece la posibilidad de usar el portapapeles (cortar, copiar y pegar) y de limitar el número de tuplas a recuperar a un número máximo.
FQ es un programa gratuito para investigadores, particulares y empresas, siempre que sea utilizado sin ánimo de lucro (20).

1.5. Metodologías de desarrollo de software

El desarrollo de software es una tarea compleja que involucra un gran equipo de personas trabajando en común por lo que se hace necesario mantener un estricto control sobre los procesos, de manera que se garantice la organización y coordinación de todo el trabajo.
Para garantizar la gestión del desarrollo de un sistema están definidas varias metodologías, pues de no ser guiado el proceso lo que se obtiene finalmente son clientes insatisfechos y productos inoperables (21).
Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las llamadas metodologías tradicionales y las metodologías ágiles. Las tradicionales se enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, son recomendadas para los proyectos de grandes dimensiones y con grandes equipos de desarrollo. En tanto las metodologías ágiles dan mayor importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueven el trabajo en equipo (22).
A continuación se analizan dos de las metodologías más utilizadas en la actualidad, una metodología ágil y otra tradicional, para escoger la más adecuada para el proceso de desarrollo del Sistema Evaluación de Competencias de manera general, y por tanto de la BDD. Estas metodologías son las siguientes:

1.5.1. Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo (RUP) es una metodología de desarrollo para proyectos grandes, lo que no quiere decir que no sea también usada en proyectos pequeños, por esto se le denomina una metodología pesada. El ciclo de vida de RUP se divide en cuatro fases de desarrollo (Inicio o Concepción, Elaboración, Construcción, Transición) (21).

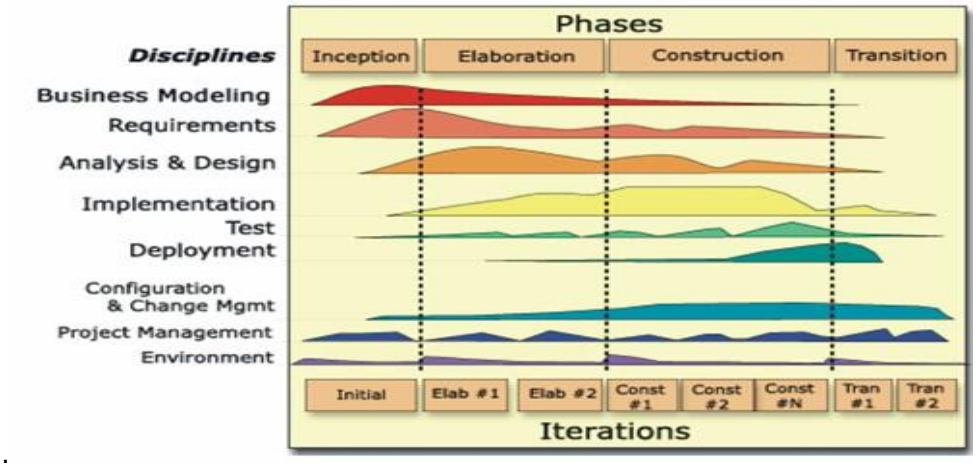


Figura. 5 Ciclo de vida de la metodología RUP.

RUP se caracteriza principalmente por (21):

- Estar dirigido por casos de uso.
- Centrado en la arquitectura
- Ser iterativo e incremental.

RUP realiza un levantamiento exhaustivo de requisitos, busca detectar defectos en las fases iníciales con el fin de realizar un proyecto eficiente con calidad y rentable. Intenta reducir el número de cambios que ocurren en el transcurso del software y realiza el análisis y diseño, tan completo como sea posible con un diseño genérico anticipándose a futuras necesidades. El cliente interactúa con el equipo de desarrollo mediante reuniones (22).

1.5.2. Programación Extrema (XP)

XP (Extreme Programming) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (23).

Esta metodología propone dividir el desarrollo en cuatro fases (23):

- Fase de planificación
- Fase de diseño
- Fase de codificación
- Fase de pruebas

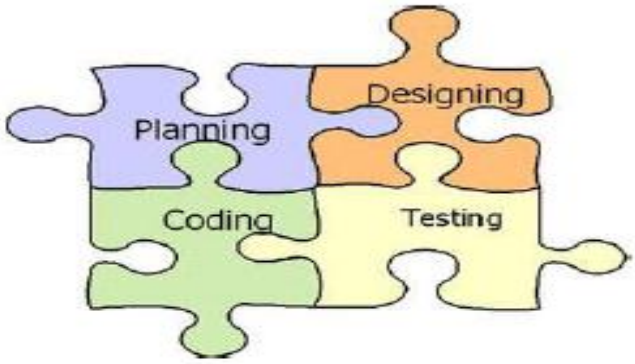


Figura. 6 Ciclo de vida de la metodología XP.

Las principales características de XP son las siguientes (23):

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas, frecuentemente repetidas y automáticas.
- Programación en parejas.
- Frecuentemente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Hacer entregas frecuentes.
- Refactorización del código.
- Propiedad del código compartida: promueve el que todo personal pueda corregir y extender cualquier parte del proyecto.
- Simplicidad en el código

La metodología XP fue concebida y desarrollada para direccionar las necesidades específicas del desarrollo de software llevado a cabo por pequeños equipos en aras de satisfacer requisitos vagos y cambiantes (22).

1.5.3. Fundamentación de la metodología a utilizar

Después de realizado el estudio de algunas metodologías de desarrollo de software, se selecciona la metodología XP para el desarrollo del Sistema de Evaluación de Competencias y para el desarrollo de la BDD. Fue escogida dicha metodología por ser ágil, diseñada para equipos de trabajo pequeños, centrada en vincular al cliente en el ciclo de desarrollo, incrementando la posibilidad de éxito, minimizando los riesgos de no conformidades y de obtener un producto final rechazado por el cliente por no cumplir con los objetivos y especificaciones trazadas. La metodología XP permite de manera eficiente los cambios que se puedan presentar durante todo el desarrollo de la herramienta, proponiendo un ciclo de vida dinámico. El proceso de prueba de XP posibilita probar cada funcionalidad al finalizar cada iteración comprobando si cumple con los requisitos de la herramienta.

1.6. Conclusiones del capítulo

En este capítulo fueron abordados diferentes temas con fundamento teórico, entre estos tema se encuentra un amplio estado del arte de las BDD, donde se especifican y se describen algunos de los aspectos más importantes relacionados con las mismas, los cuales propiciaron esclarecer las dudas existentes sobre el tema en cuestión y a la vez se comprobó que las BDD son la solución perfecta para resolver la situación problémica de este trabajo.

Se realizó un cuidadoso análisis y selección de los lenguajes y herramientas que serán utilizados posteriormente para el desarrollo de la BDD.

A través de cada uno de los contenidos que fueron tratados en este capítulo se le dio cumplimiento a tres de las tareas investigativas que fueron planificadas inicialmente para cumplir con el objetivo general propuesto.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1. Introducción

En el presente capítulo se describe de forma general la base de datos difusa a desarrollar, guiando el proceso de desarrollo a través de las tres primeras fases que propone la metodología XP, para de esta forma realizar el análisis, diseño e implementación de la BDD de forma organizada y en el menor tiempo posible.

2.2. Fase de planificación

Como bien se especificó en el capítulo anterior, la primera fase que propone la metodología XP es la fase de planificación, en ella se recopilan todos los requerimientos del proyecto, se interactúa con el usuario, y se planifica que es lo que se quiere realizar para lograr el objetivo final (22).

2.2.1. Requisitos no funcionales de la BDD

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable (24).

En muchos casos los RNF son fundamentales en el éxito del producto además de formar una parte significativa de las especificaciones. También ayudan a marcar la diferencia entre un producto bien aceptado por los usuarios y uno con poca aceptación (25).

Para el desarrollo de BDD es necesario tener en cuenta los siguientes requisitos no funcionales:

RNF 1. Usabilidad

RNF 1.1 La base de datos debe ser de fácil comprensión, navegación, configuración y utilización, tanto para usuarios con un nivel alto de experiencia como de niveles inferiores en el campo de la informática.

RNF 1.2 La información debe ser mostrada de forma lógica y organizada.

RNF 1.3 La base de datos resultante, debe satisfacer las necesidades del usuario.

RNF 2. Confiabilidad

RNF 2.1 La BD debe estar protegida mediante una política de usuarios y roles que no permitan el acceso no autorizado a la misma. De esta forma se asegura la integridad y confiabilidad de los datos que en ella se procesan.

RNF 2.2 La aplicación deberá estar disponible siempre, asegurando el acceso en todo momento desde cualquier lugar de red a todos los usuarios que estén autorizados.

RNF 3. Rendimiento

RNF 3.1 La BD debe tener un eficiente tiempo de respuesta, de acuerdo con la funcionalidad que se esté usando en cualquier momento, no excediendo los 10 segundos para las funcionalidades que no sean muy complejas.

RNF 4. Requisitos de Software

RNF 4.1 Debe estar instalado el SGBD Oracle 10g o una versión superior.

RNF 5. Requisitos de Hardware

RNF 5.1 Se requiere un procesador de 500 MHz (mínimo).

RNF 5.2 Se requiere 512 MB mínimo de RAM.

RNF 5.3 Se requiere un disco duro 80 GB o superior por el volumen de datos que se genera.

RNF 6. Portabilidad

RNF 6.1 La BD debe ser compatible con el sistema operativo Windows XP y Windows7.

2.3. Fase de Diseño

Durante esta fase se sugiere que hay que conseguir diseños simples y sencillos, por lo que hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible, que a la larga costará menos tiempo y esfuerzo desarrollar (26).

2.3.1. Diseño lógico de la BDD

El diseño lógico es el proceso de construir un esquema de la información, basándose en un modelo de BD específico independiente del SGBD y de cualquier otra consideración física. El

esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos (27).

El resultado de las evaluaciones realizadas a los estudiantes, presentará cierto grado de incertidumbre, por lo que resulta necesario que el modelo lógico de la BD que se desarrolla, permita almacenar cada uno de los aspectos relacionados con las BDD, para poder realizar luego el análisis difuso correspondiente.

De acuerdo a lo anterior se propone el siguiente modelo lógico de la base de datos difusa:

Figura. 7 Modelo lógico de la BDD.

2.3.2. Diseño físico de la BDD

El diseño físico es el proceso de producir la descripción de la implementación de la BD en memoria secundaria, donde las estructuras de almacenamiento y los métodos de acceso garanticen un acceso eficiente a los datos. En esta etapa se decidió cuál es el SGBD a utilizar, o sea Oracle, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico (27).

El modelo físico obtenido a partir del modelo lógico representado anteriormente es el siguiente:

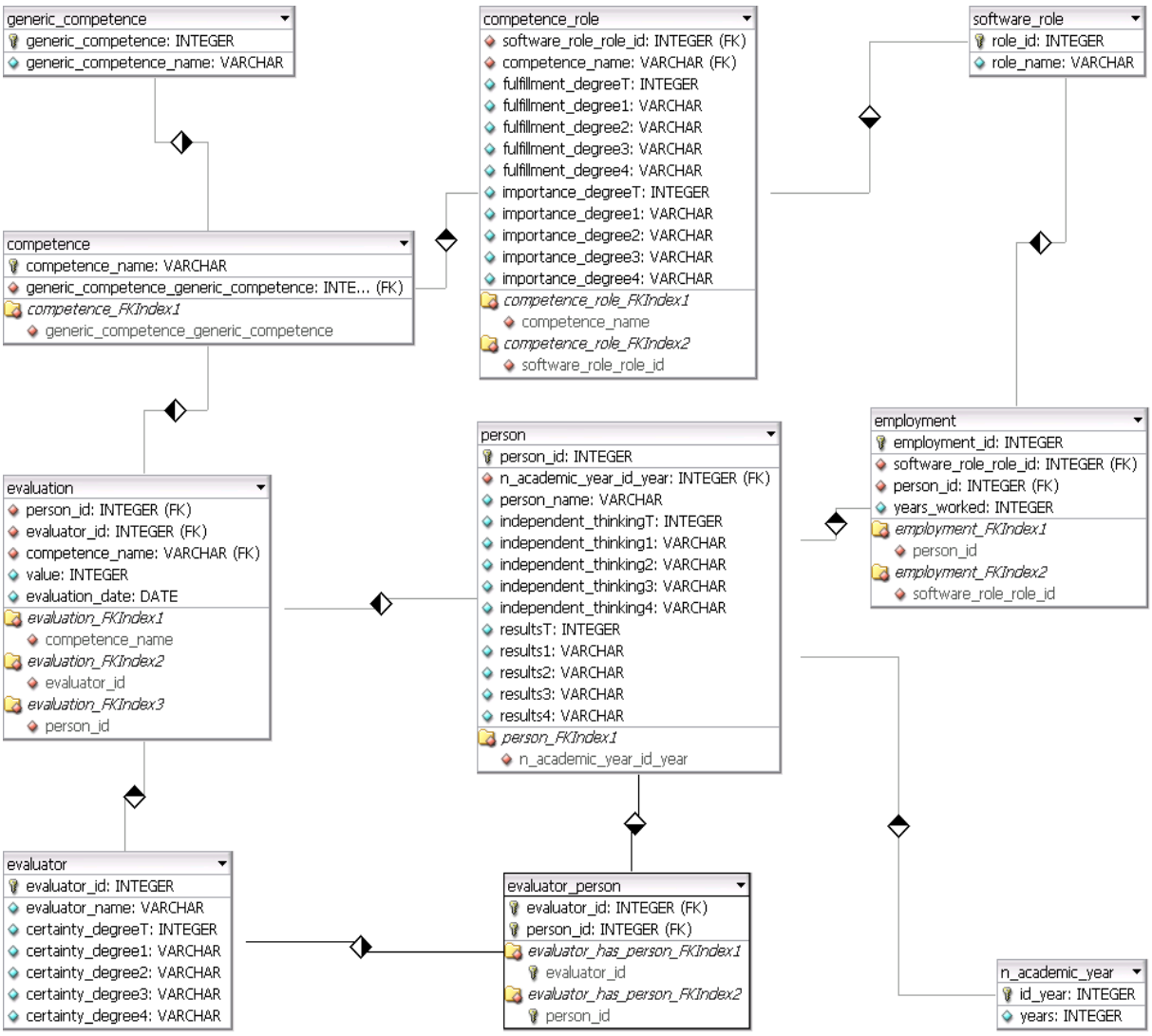


Figura. 8 Modelo físico de la BDD.

2.3.3. Descripción de las principales tablas

Nombre:	evaluation		
Descripción:	En esta tabla se almacenarán los resultados de las evaluaciones realizadas a personas determinadas, utilizando instrumentos evaluadores específicos.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	evaluator_id	NUMBER	Llave primaria de la tabla "*evaluator*". Almacena el identificador del evaluador utilizado.
PK	person_id	NUMBER	Llave primaria de la tabla "*person*". Almacena el identificador de la persona evaluada.
PK	competence_name	VARCHAR	Llave primaria de la tabla "*competence*". Nombre de la competencia evaluada.
	value	NUMBER	Almacena el resultado obtenido en la evaluación.
	evaluation_date	DATE	Fecha de realización de la evaluación.

Tabla. 5 Descripción de la tabla "*evaluation*".

Nombre:	evaluator		
Descripción:	En esta tabla se almacenan cada uno de los instrumentos que se utilizarán en el proceso de evaluación, dígase test, encuestas, u otros.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	evaluator_id	NUMBER	Llave primaria de la tabla "*evaluator*". Almacena el identificador de cada evaluador.
	evaluator_name	VARCHAR	Nombre del evaluador o instrumento.
	certainty_degreeT	NUMBER	Atributo difuso Tipo2 (ver Tabla 17)
	certainty_degree1	NUMBER	Atributo difuso Tipo2
	certainty_degree2	NUMBER	Atributo difuso Tipo2
	certainty_degree3	NUMBER	Atributo difuso Tipo2
	certainty_degree4	NUMBER	Atributo difuso Tipo2

Tabla. 6 Descripción de la tabla "*evaluator*".

Nombre:	person		
Descripción:	En esta tabla se almacenan los datos de las personas, así como los atributos difusos (competencias) que posean cada una de ellas, para poder realizar la evaluación correspondiente.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	person_id	NUMBER	Llave primaria de la tabla "*person*". Almacena el identificador de cada una de los estudiantes.

	person_name	VARCHAR	Nombre de la persona evaluada.
	independent_thinkingT	NUMBER	Atributo difuso Tipo2 (ver Tabla 18)
	independent_thinking1	NUMBER	Atributo difuso Tipo2
	independent_thinking2	NUMBER	Atributo difuso Tipo2
	independent_thinking3	NUMBER	Atributo difuso Tipo2
	independent_thinking4	NUMBER	Atributo difuso Tipo2
	resultsT	NUMBER	Atributo difuso Tipo2 (ver Tabla. 18)
	results1	NUMBER	Atributo difuso Tipo2
	results2	NUMBER	Atributo difuso Tipo2
	results3	NUMBER	Atributo difuso Tipo2
	results4	NUMBER	Atributo difuso Tipo2
FK	Id_years	NUMBER	Llave primaria de la tabla "n_academic_year".

Tabla. 7 Descripción de la tabla "*person*".

Nombre:	competence		
Descripción:	En esta tabla se almacenarán cada una de las competencias que se medirán en el proceso de evaluación.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	competence _name	VARCHAR	Llave primaria de la tabla "*competence*". Almacena el nombre de cada una de las competencias.
FK	generic_competence_id	NUMBER	Llave primaria de la tabla "*generic_competence*". Almacena el identificador de la competencia genérica a la que está asociada cada una de las competencias.
	obj#	NUMBER	Almacena el identificador del objeto de la tabla a la que pertenece el atributo difuso asociado a la competencia.
	col#	NUMBER	Almacena el identificador de la columna correspondiente al atributo difuso asociado a la competencia.

Tabla. 8 Descripción de la tabla "*competence*".

Nombre:	competence_role		
Descripción:	Almacena los grados difusos que caracterizan la relación existente entre un rol específico y una competencia determinada.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	role_id	NUMBER	Llave primaria de la tabla "*soft_role*". Almacena el identificador de cada uno de los roles informáticos.

PK	competence_name	VARCHAR	Llave primaria de la tabla "competence". Almacena el nombre de las competencias.
	fulfillment_degreeT	NUMBER	Atributo difuso Tipo2 (ver Tabla 16)
	fulfillment_degree1	NUMBER	Atributo difuso Tipo2
	fulfillment_degree2	NUMBER	Atributo difuso Tipo2
	fulfillment_degree3	NUMBER	Atributo difuso Tipo2
	fulfillment_degree4	NUMBER	Atributo difuso Tipo2
	importance_degreeT	VARCHAR	Atributo difuso Tipo2 (ver Tabla 16)
	importance_degree1	NUMBER	Atributo difuso Tipo2
	importance_degree2	NUMBER	Atributo difuso Tipo2
	importance_degree3	NUMBER	Atributo difuso Tipo2
	importance_degree4	NUMBER	Atributo difuso Tipo2

Tabla. 9 Descripción de la tabla "*competence_role*".

Nombre:	software_role		
Descripción:	En esta tabla se almacenarán los roles del proceso de desarrollo de software que se tendrán en cuenta a la hora de asignarlos a los estudiantes una vez realizada la evaluación.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	role_id	NUMBER	Llave primaria de la tabla "*software_role*". Almacena el identificador de cada uno de los roles informáticos.
	role_name	VARCHAR	Nombre del rol.

Tabla. 10 Descripción de la tabla "*soft_role*".

Nombre:	evaluator_person		
Descripción:	Es la tabla generada por la relación de mucho a mucho existente entre las tablas "*evaluator"* y "*person"*.		
Llave	**Columna**	**Tipo**	**Descripción**
FK	evaluator_id	NUMBER	Llave primaria de la tabla "*evaluator*". Almacena el identificador de cada uno de los instrumentos.
FK	person_id	NUMBER	Llave primaria de la tabla "*person*". Almacena el identificador de cada una de las personas.

Tabla. 11 Descripción de la tabla "*evaluator_person*".

Nombre:	generic_competence		
Descripción:	Es la tabla donde se almacenarán cada una de las competencias genéricas que serán evaluadas.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	generic_competence_id	NUMBER	Llave primaria de la tabla "*generic_competence*". Almacena el identificador de cada competencia genérica.
	generic_competence_name	VARCHAR	Nombre de la competencia genérica.

Tabla. 12 Descripción de la tabla "*generic_competence*".

Nombre:	employment		
Descripción:	En esta tabla se almacena la cantidad de años en que una persona ha trabajado con un rol específico.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	employment_id	NUMBER	Llave primaria de la tabla "*employment*".
FK	role_id	NUMBER	Llave primaria de la tabla "*software_role*". Almacena el identificador de cada uno de los roles.
FK	person_id	NUMBER	Llave primaria de la tabla "*person*". Almacena el identificador de cada una de las personas.
	years_worked	NUMBER	Cantidad de años de experiencia.

Tabla. 13 Descripción de la tabla "*employment*".

Nombre:	n_academic_year		
Descripción:	En esta tabla se almacena el año de estudio en que se encuentra cada uno de los estudiantes que serán evaluados.		
Llave	**Columna**	**Tipo**	**Descripción**
PK	id_year	NUMBER	Llave primaria de la tabla "*n_academic_year*".
	year	NUMBER	Año de estudio que cursa el estudiante.

Tabla. 14 Descripción de la tabla "*n_academic_year*".

2.3.4. Patrones de diseño de BD

El diseño y construcción de una base de datos requiere del mayor esfuerzo y análisis posible, ya que a partir de este diseño es que se crean las mismas. En la actualidad las bases de datos suelen ser muy grandes y a veces el uso de los patrones de diseño hace que el trabajo resulte mucho más sencillo, además aseguran un resultado correcto (28).

Los *patrones de diseño de bases de datos*, es una plantilla que ya ha sido evaluada como la responsable de resolver un problema. Son recomendaciones probadas y sirven como modelo para dar soluciones a problemas comunes en el desarrollo del software. En la actualidad son consideradas como buenas prácticas en la ingeniería de software (28).

2.3.4.1. Patrón utilizado. Llaves subrogadas

Este patrón decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Normalmente se usa enteros en columnas *identity* o GUID (Global UniqueIdentifier) que están demostrados que no se repiten o con una probabilidad extremadamente baja. Permite que las tablas sean más fáciles de consultar por el identificador dado que se conoce el mismo tipo en cada tabla (28).

2.4. Fase de codificación

En esta fase se planifican y ejecutan las tareas de programación, durante la misma se codifica todo el sistema diseñado, obteniendo como resultado la herramienta, en este caso, la base de datos difusa.

2.4.1. Sintaxis y semántica del lenguaje FSQL

Cada sentencia FSQL que se desea ejecutar debe ser previamente analizada por el sistema, para asegurar por un lado que la sentencia está correctamente escrita y por otro, que tiene sentido efectuarla; para ello se usa los tres típicos analizadores siguientes:

- **Analizador Léxico:** Asegura que todos los elementos de la sentencia están permitidos, agrupando los caracteres en palabras (tokens) (7).
- **Analizador Sintáctico:** Asegura que los tokens están en un orden adecuado y que la construcción de la sentencia es correcta sintácticamente (7).

- **Analizador Semántico:** Asegura que el significado de la sentencia es correcto y que por tanto tiene sentido efectuarla (7).

A continuación se relacionan los comandos más útiles e importantes, explicando las novedades que estos incorporan en FSQL para permitir manejar información difusa.

2.4.1.1. SELECT

La sentencia SELECT es una sentencia tan potente como compleja y flexible, muy fácil de usar en consultas simples y no tan fácil de usar en consultas complejas debido a su potencia y versatilidad. Esta sentencia es tan potente que rara vez se llega a utilizar todo su poder expresivo para realizar una consulta, pues lo normal es efectuar consultas mucho más simples de lo que SELECT permite. A veces, para simplificar la escritura y el entendimiento de consultas complicadas se usan vistas intermedias que son creadas como subconsultas a la BD (2).

2.4.1.2. Otros Comandos: INSERT, DELETE y UPDATE

Los comandos INSERT, DELETE y UPDATE también pertenecen con el comando SELECT al DML y su sintaxis es muy similar a la que tienen en SQL, modificando las expresiones, las subconsultas y las condiciones por expresiones difusas, subconsultas difusas y condiciones difusas respectivamente.

En síntesis, las modificaciones de FSQL son las siguientes para cada uno (2):

- **INSERT:** Se pueden utilizar expresiones difusas como valores para la inserción, así como subconsultas difusas, tanto en los valores a insertar como en las relaciones en las que se insertan.
- **DELETE:** En la cláusula WHERE de este comando se pueden utilizar las mismas condiciones difusas que en la cláusula WHERE del comando SELECT de FSQL, explicado anteriormente.
- **UPDATE:** Los valores a actualizar podrán ser expresiones difusas o subconsultas difusas. Además, en la cláusula WHERE de este comando se pueden utilizar las mismas condiciones difusas que en la cláusula WHERE del comando SELECT de FSQL, explicado anteriormente.

2.4.1.3. Carácter comodín %

El empleo de este carácter es similar al del utilísimo carácter comodín * de SQL, pero este, además de incluir todas las columnas de las tablas indicadas en la parte FROM de la consulta, también incluye las columnas con el grado de cumplimiento de aquellos atributos relevantes. O sea, en el resultado también se encuentran columnas donde la función CDEG está aplicada a cada uno de los atributos que aparecen en la condición (2).

Este carácter puede ser también usado con el formato [[scheme.]table.]%, como por ejemplo: Personas.%.

Si un atributo difuso no aparece en la cláusula WHERE, su CDEG no es aplicable y por tanto no aparecerá su CDEG si se usa el comodín %.

2.4.1.4. Condición con IS

Otra clase de condición que se pueden usar en FSQL tiene el siguiente formato (2):

```
                                    ⎧ UNKNOWN
<Atributo_difuso> IS [NOT]          ⎨ UNDEFINED
                                    ⎩ NULL
```

Observaciones sobre la condición con IS (7):

- Este tipo de condición (sin NOT) será cierta si el valor del atributo difuso de la izquierda (<Atributo_difuso>) es la constante situada a la derecha.
- Si el atributo no es difuso y la constante de la derecha es NULL, esta constante será entendida de la forma que lo haga el SGBD (si este lo permite). En particular Oracle permite valores NULL como valor posible de los atributos, aunque esto puede ser evitado estableciendo la restricción NOT NULL en los comandos CREATE TABLE o ALTER TABLE.
- Si FEQ es usado en vez de IS el significado es distinto. Con FEQ se compara el grado de compatibilidad entre el atributo y la constante, y no simplemente si el atributo es igual a la constante.

2.4.1.5. Función CDEG

Una llamada a la función CDEG (Compatibility DEGree) puede ser colocada en la lista de selección (expresiones tras la palabra reservada SELECT). Su argumento es un atributo y muestra una columna con el grado de compatibilidad o cumplimiento de la condición de la consulta para el atributo que se indica (2).

Si el argumento de la función CDEG es un atributo, entonces la función CDEG sólo usa las condiciones que incluyen a ese atributo. Si el atributo indicado como argumento de CDEG no aparece en la condición, entonces, esta función no es aplicable a dicho atributo, pero en vez de dar un error se procede a devolver grado 1 para todas las tuplas. También se puede usar un asterisco como argumento como se explica a continuación (9).

CDEG(*): Si esta función tiene un asterisco como argumento, entonces calcula y muestra el grado de cumplimiento de la condición para cada tupla. Este cálculo es efectuado como se explicó anteriormente pero teniendo en cuenta todos los atributos empleados en la condición, y no sólo uno de ellos (9).

<Condición> (*con operadores lógicos*)	**CDEG(<Condición>)**
<cond1> AND <cond2>	min(CDEG(<cond1>),CDEG(<cond2>))
<cond1> OR <cond2>	max(CDEG(<cond1>),CDEG(<cond2>))
NOT <cond1>	1 - CDEG(<cond1>)

Tabla. 15 Operaciones usadas por defecto para el cálculo de la función CDEG de FSQL (11).

2.4.1.6. Comentarios

FSQL permite incorporar comentarios en las sentencias, de forma que ellos no serán tomados en cuenta al analizarla y ejecutarla. Los comentarios pueden ser de 3 tipos:

- **Comentario hasta fin de línea:** Con -- (dos guiones) señala que desde ese punto hasta el final de esa línea es un comentario. Este tipo de comentarios es también válido en SQL y PL/SQL (2).
- **Comentario de un bloque:** Todo lo que esté incluido entre las marcas /* y */ será considerado como un comentario. Este tipo de comentarios (estándar en el lenguaje C) es también válido en SQL y PL/SQL (2).

- **Comentario hasta fin de sentencia:** Con /* señala que desde ese punto hasta el final de la sentencia es un comentario. Es decir, si no se cierra el comentario se supone que el comentario termina al final, siendo ignorado totalmente el resto de la sentencia. Este tipo de comentarios es propio de FSQL y generan un error si se usan en SQL o PL/SQL (2).

2.4.2. Atributos difusos utilizados en la BDD

Para la implementación de la BDD fue necesario crear una serie de atributos difusos en algunas de las tablas de la misma, cada uno de estos atributos difusos utilizados son de Tipo 2. Los mismos permiten almacenar y representar valores difusos como trapecios, labels (etiquetas), crisp, entre otros.

Como bien se explicó en el primer capítulo, este tipo de atributo difuso en específico internamente se representa a través de cinco atributos (T, 1, 2, 3, 4), donde cada uno de ellos toman varios valores para cada tipo de constante difusa (ver Tabla. 1).

Estos atributos son los siguientes:

Atributos difusos de la tabla "*competence_role*":

Atributos difusos	Descripción
fulfillment_degreeT fulfillment_degree1 fulfillment_degree2 fulfillment_degree3 fulfillment_degree4	Grado de cumplimiento: El grado de cumplimiento se puede definir como el nivel requerido en una competencia para poder desempeñar un rol determinado de manera exitosa. Este atributo podrá ser medido a través de las siguientes etiquetas lingüísticas: "muy_bajo", "bajo", "normal", "alto" y "muy_alto" (ver figura 12).
importance_degreeT importance_degree1 importance_degree2 importance_degree3 importance_degree4	Grado de importancia: En este atributo se almacena el grado de importancia que posea una competencia determinada para un rol específico. Este atributo podrá ser medido a través de las siguientes etiquetas lingüísticas: "muy_baja", "baja", "normal", "alta" y "muy_alta" (ver figura 11).

Tabla. 16 Atributos difusos de la tabla "*competence_role*".

Atributos difusos de la tabla "*evaluator*":

Atributos difusos	Descripción
certainty_degreeT certainty_degree1 certainty_degree2 certainty_degree3 certainty_degree4	Grado de confiabilidad: En este atributo se almacena el grado de confiabilidad que posee un evaluador determinado (dígase test, encuestas, entre otros), para de esta forma tener una idea de hasta qué punto podría ser confiable dicho evaluador a la hora de realizar las evaluaciones a los estudiantes. Este atributo podrá ser medido a través de las siguientes etiquetas lingüísticas: "muy_bajo" "bajo", "medio", "alto" y "muy_alto" (ver figura 13).

Tabla. 17 Atributos difusos de la tabla "*evaluator*".

Atributos difusos de la tabla *"person":*

La tabla *"person"* contiene los atributos difusos (competencias) que se medirán en el proceso de evaluación. Estos atributos podrán ser gestionados por la aplicación que utilice la base de datos difusa, de ahí que estos tienen un carácter dinámico, ya que se pueden agregar nuevos atributos difusos así como eliminar o editar los existentes.

A continuación se muestra un ejemplo de cómo se representan estos atributos en dicha tabla.

Atributos difusos	Descripción
independent_thinkingT independent_thinking1 independent_thinking2 independent_thinking3 independent_thinking4	Pensamiento independiente (competencia): Esta competencia podrá ser medida a través de las siguientes etiquetas: "muy_malo", "malo", "normal", "bueno" y "muy_bueno".
resultsT results1 results2 results3 results4	Resultados sociales (competencia): Esta competencia podrá ser medida a través de las siguientes etiquetas: "muy_bajos", "bajos", "normales", "buenos" y "muy_buenos".

Tabla. 18 Ejemplo de atributos difusos de la tabla "*person*".

2.4.3. Etiquetas lingüísticas de la BDD

A continuación se definen las etiquetas lingüísticas asociadas a cada uno de los grados difusos que fueron identificados y descritos anteriormente. Estas etiquetas se encuentran representadas a través de la distribución de posibilidad trapezoidal o del trapecio probabilístico mencionado en el capítulo 1.

Se omitirán las etiquetas lingüísticas asociadas a los atributos difusos de la tabla "*person*" debido a su naturaleza dinámica, lo que implica que estas se crearán a través del sistema de gestión que utiliza la base de datos.

Los valores mostrados en los trapecios probabilísticos son valores arbitrarios que podrán ser modificados por el administrador del sistema en cualquier momento.

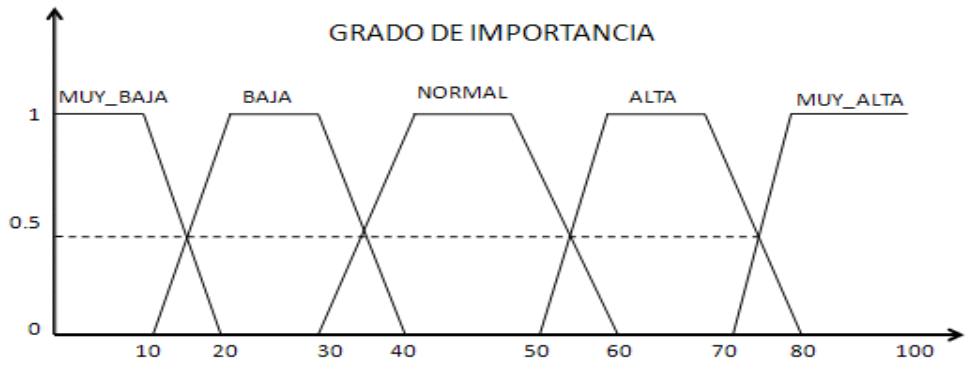


Figura. 9 Definición de etiquetas para el atributo "*importance_degree*".

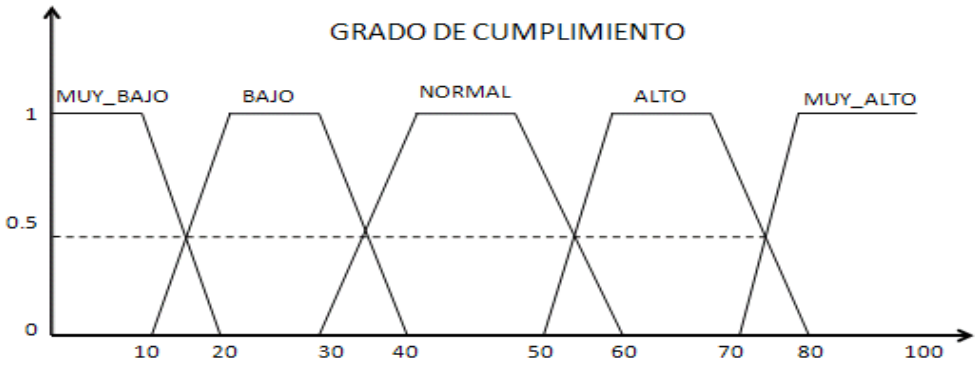


Figura. 10 Definición de etiquetas para el atributo "*fulfillment_degree*".

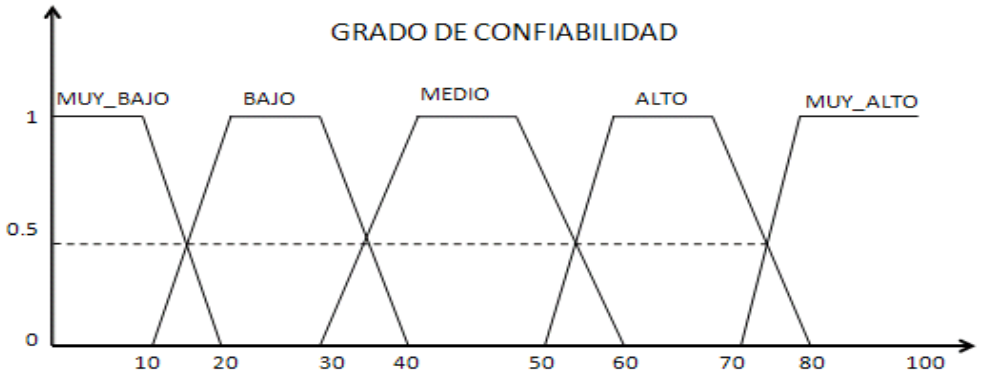


Figura. 11 Definición de etiquetas para el atributo *"certainty_degree"*.

2.4.4. Consultas difusas

Seguidamente se muestran algunos ejemplos de consultas difusas realizadas a la BDD que se desarrolla, probando así cada uno de los aspectos mencionados en el epígrafe 2.4.1. Estas consultas fueron implementadas en el lenguaje escogido, el lenguaje FSQL.

Para la tabla *"person"* que se muestra a continuación, se han implementado algunas consultas difusas, algunas de ellas son las siguientes:

FQ: SELECT person.%, cdeg(*)□□from person□□

Archivo Opciones

Núm. de Columnas : **6**

Filas Recuperadas: **15 (todas)**

Tiempo Traducción: 0,0 sgs.

Tiempo Consulta : 0,2 sgs.

Cerrar

Nº Fila	PERSON_ID	PERSON_NAME	INDEPENDENT_THINKING	RESULTS	ID_YEAR	CDE
1	1	Evelyn	9	8	2	
2	2	Luis	3	5	7	
3	3	Adriana	[5,10]	[40,80]	3	
4	4	Rafael	MALO	NORMALES	4	
5	5	Lorena	MUY_BUENO	BUENOS	10	
6	6	Maria	[1,5]	[5,10]	5	
7	7	Carlos	40	33	7	
8	8	Pedro	[15,25]	[5,25]	5	
9	9	Manuel	25	NORMALES	7	
10	10	Tatiana	[12,30]	BAJOS	4	
11	11	Barbara	[5,10]	[15,35]	8	
12	12	Juan	BUENO	MUY_BUENOS	7	
13	13	Yanet	55	15	2	
14	14	Teijon	MUY_BUENO	50	2	
15	15	Adrian	[45,80]	[20,25]	NULL	

Figura. 12 Tabla *"person" de la BDD*.

2.4.4.1. Ejemplo 2.1

Devolver todos los datos de aquellas personas que tengan pensamiento independiente malo, y que a su vez posean resultados sociales posiblemente mayor que bajos (en grado mínimo 0.25).

La consulta FSQL y el resultado obtenido se muestran a continuación.

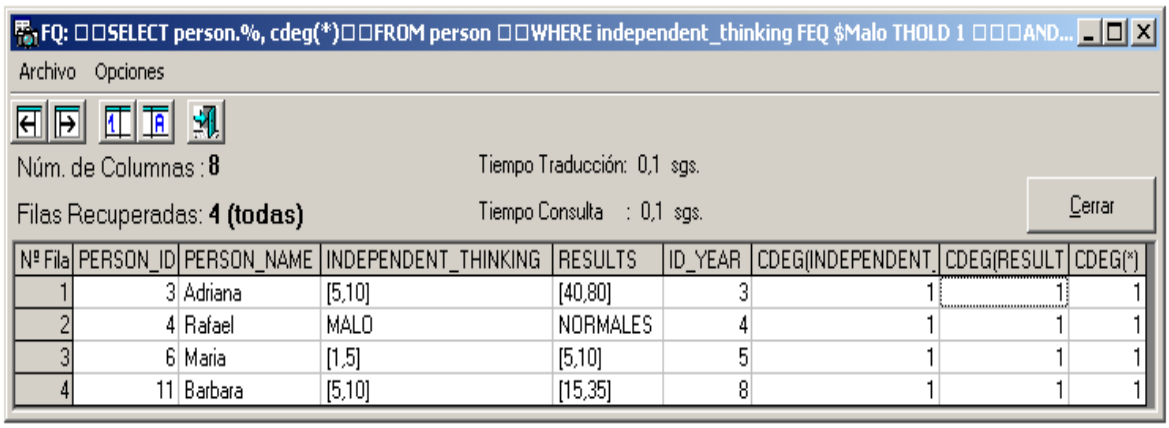
Consulta difusa en el lenguaje FSQL:

```
SELECT person.%, cdeg(*)
FROM person
WHERE independent_thinking FEQ $Malo THOLD 1
          AND results FGT $Bajos THOLD 0.25
```

Traducción de la consulta difusa anterior al lenguaje SQL:

```
SELECT
   person.person_id,person.person_name,FSQL_FUNCTIONS.FSHOW2(4994
   8,3,person.independent_thinkingt,person.independent_thinking1,
   person.independent_thinking2,person.independent_thinking3,pers
   on.independent_thinking4)
   independent_thinking,FSQL_FUNCTIONS.FSHOW2(49948,8,person.resu
   ltst,person.results1,person.results2,person.results3,person.re
   sults4)results,person.id_year,FSQL_FUNCTIONS.FEQ_trape(0,0,5,1
   0,49948,3,independent_thinkingt,independent_thinking1,independ
   ent_thinking2,independent_thinking3,independent_thinking4)
   "CDEG(independent_thinking)",FSQL_FUNCTIONS.FGT_trape(0,0,5,10
   ,49948,8,resultst,results1,results2,results3,results4)
   "CDEG(results)",LEAST(FSQL_FUNCTIONS.FEQ_trape(0,0,5,10,49948,
   3,independent_thinkingt,independent_thinking1,independent_thin
   king2,independent_thinking3,independent_thinking4),FSQL_FUNCTI
   ONS.FGT_trape(0,0,5,10,49948,8,resultst,results1,results2,resu
   lts3,results4))
   "CDEG(*)"
FROM person
```

```
WHERE
    FSQL_FUNCTIONS.FEQ_trape(0,0,5,10,49948,3,independent_thinking
    t,independent_thinking1,independent_thinking2,independent_thin
    king3,independent_thinking4) >= 1
    AND
    FSQL_FUNCTIONS.FGT_trape(0,0,5,10,49948,8,resultst,results1,re
    sults2,results3,results4) >= 0.25
```



Nº Fila	PERSON_ID	PERSON_NAME	INDEPENDENT_THINKING	RESULTS	ID_YEAR	CDEG(INDEPENDENT_	CDEG(RESULT	CDEG(*)
1	3	Adriana	[5,10]	[40,80]	3	1	1	1
2	4	Rafael	MALO	NORMALES	4	1	1	1
3	6	Maria	[1,5]	[5,10]	5	1	1	1
4	11	Barbara	[5,10]	[15,35]	8	1	1	1

Figura. 13 Relación resultante del Ejemplo 2.1.

2.4.4.2. Ejemplo 2.2

Devolver todas las personas que tengan un pensamiento independiente mayor o igual que 9 (con un grado mínimo de 0.75).

La consulta FSQL y el resultado obtenido se muestran a continuación.

Consulta difusa en el lenguaje FSQL:

```
SELECT person.%, cdeg(results)
FROM person
WHERE independent_thinking FGEQ #9 thold 0.5
```

Traducción de la consulta difusa anterior al lenguaje SQL estándar:

```
SELECT
      person.person_id,person.person_name,FSQL_FUNCTIONS.FSHOW2(49
      948,3,person.independent_thinkingt,person.independent_thinki
```

```
        ng1,person.independent_thinking2,person.independent_thinking
        3,person.independent_thinking4)independent_thinking,FSQL_FUN
        CTIONS.FSHOW2(49948,8,person.resultst,person.results1,person
        .results2,person.results3,person.results4)results,person.id_
        year,FSQL_FUNCTIONS.FGEQ_aprox(9,6,12,3,49948,3,independent_
        thinkingt,independent_thinking1,independent_thinking2,indepe
        ndent_thinking3,independent_thinking4)
         "CDEG(independent_thinking)",1 "CDEG(RESULTS)"
FROM person
WHERE
        FSQL_FUNCTIONS.FGEQ_aprox(9,6,12,3,49948,3,independent_think
        ingt,independent_thinking1,independent_thinking2,independent
        _thinking3,independent_thinking4) >= 0.5
```

FQ: SELECT person.%, cdeg(results) □□FROM person □□WHERE independent_thinking FGEQ #9 thold ...

Archivo Opciones

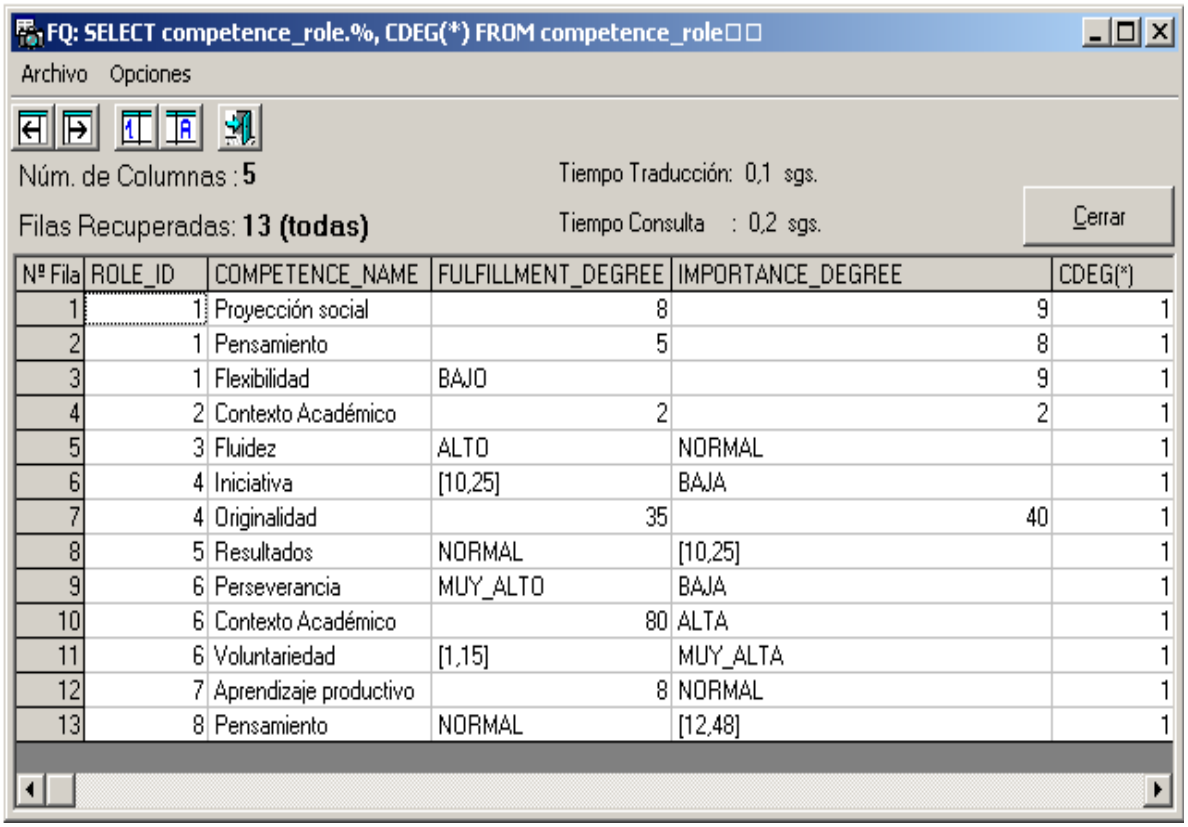
Núm. de Columnas : 7 Tiempo Traducción: 0,1 sgs.

Filas Recuperadas: 11 (todas) Tiempo Consulta : 0,1 sgs. Cerrar

Nº Fila	PERSON_ID	PERSON_N/	INDEPENDENT_THINKING	RESULTS	ID_YEAR	CDEG(INDEI	CDEG(RESULTS)
1	1	Evelyn	9	8	2	1	1
2	4	Rafael	MALO	NORMALES	4	0,5	1
3	5	Lorena	MUY_BUENO	BUENOS	10	1	1
4	7	Carlos	40	33	7	1	1
5	8	Pedro	[15,25]	[5,25]	5	1	1
6	9	Manuel	25	NORMALES	7	1	1
7	10	Tatiana	[12,30]	BAJOS	4	1	1
8	12	Juan	BUENO	MUY_BUENOS	7	1	1
9	13	Yanet	55	15	2	1	1
10	14	Teijon	MUY_BUENO	50	2	1	1
11	15	Adrian	[45,80]	[20,25]	NULL	1	1

Figura. 14 Relación resultante del Ejemplo 2.2.

Ejemplos de consultas para la tabla "*competence_role*" que se muestra a continuación:


FQ: SELECT competence_role.%, CDEG(*) FROM competence_role

Archivo Opciones

Núm. de Columnas : 5 Tiempo Traducción: 0,1 sgs.

Filas Recuperadas: 13 (todas) Tiempo Consulta : 0,2 sgs. Cerrar

Nº Fila	ROLE_ID	COMPETENCE_NAME	FULFILLMENT_DEGREE	IMPORTANCE_DEGREE	CDEG(*)
1	1	Proyección social	8	9	1
2	1	Pensamiento	5	8	1
3	1	Flexibilidad	BAJO	9	1
4	2	Contexto Académico	2	2	1
5	3	Fluidez	ALTO	NORMAL	1
6	4	Iniciativa	[10,25]	BAJA	1
7	4	Originalidad	35	40	1
8	5	Resultados	NORMAL	[10,25]	1
9	6	Perseverancia	MUY_ALTO	BAJA	1
10	6	Contexto Académico	80	ALTA	1
11	6	Voluntariedad	[1,15]	MUY_ALTA	1
12	7	Aprendizaje productivo	8	NORMAL	1
13	8	Pensamiento	NORMAL	[12,48]	1

Figura. 15 Tabla "*competence_role*" de la BDD.

2.4.4.3. Ejemplo 2.3

Devolver todos los datos de las competencias que posean un grado de cumplimiento bajo (en grado mínimo 0.75) o las competencias que a su vez tengan un alto grado de cumplimiento y que cumplan con un grado de importancia normal.

La consulta FSQL y el resultado obtenido se muestran a continuación.

Consulta difusa en el lenguaje FSQL:

```
SELECT competence_role.%, CDEG(*) FROM competence_role
WHERE fulfillment_degree NFEQ $bajo THOLD 0.75
      OR    (fulfillment_degree    NFEQ    $alto    THOLD    1    AND
      importance_degree FEQ $Normal THOLD 1)
```

Traducción de la consulta difusa anterior al lenguaje SQL:

```
SELECT
    competence_role.role_id,competence_role.competence_name,FSQL_
    FUNCTIONS.FSHOW2(49952,3,competence_role.fulfillment_degreet,
    competence_role.fulfillment_degree1,competence_role.fulfillme
    nt_degree2,competence_role.fulfillment_degree3,competence_rol
    e.fulfillment_degree4)fulfillment_degree,FSQL_FUNCTIONS.FSHOW
    2(49952,8,competence_role.importance_degreet,competence_role.
    importance_degree1,competence_role.importance_degree2,compete
    nce_role.importance_degree3,competence_role.importance_degree
    4)importance_degree,greatest(FSQL_FUNCTIONS.NFEQ_trape(0,0,5,
    180,49952,3,fulfillment_degreet,fulfillment_degree1,fulfillme
    nt_degree2,fulfillment_degree3,fulfillment_degree4),FSQL_FUNC
    TIONS.NFEQ_trape(185,10,10,210,49952,3,fulfillment_degreet,fu
    lfillment_degree1,fulfillment_degree2,fulfillment_degree3,ful
    fillment_degree4))
    "CDEG(fulfillment_degree)",FSQL_FUNCTIONS.FEQ_trape(175,5,10,
    195,49952,8,importance_degreet,importance_degree1,importance_
    degree2,importance_degree3,importance_degree4)
    "CDEG(importance_degree)",GREATEST(FSQL_FUNCTIONS.NFEQ_trape(
    0,0,5,180,49952,3,fulfillment_degreet,fulfillment_degree1,ful
    fillment_degree2,fulfillment_degree3,fulfillment_degree4),LEA
    ST(FSQL_FUNCTIONS.NFEQ_trape(185,10,10,210,49952,3,fulfillmen
    t_degreet,fulfillment_degree1,fulfillment_degree2,fulfillment
    _degree3,fulfillment_degree4),FSQL_FUNCTIONS.FEQ_trape(175,5,
    10,195,49952,8,importance_degreet,importance_degree1,importan
    ce_degree2,importance_degree3,importance_degree4))) "CDEG(*)"
FROM competence_role
WHERE
    FSQL_FUNCTIONS.NFEQ_trape(0,0,5,180,49952,3,fulfillment_degre
    et,fulfillment_degree1,fulfillment_degree2,fulfillment_degree
    3,fulfillment_degree4)>=0.75
```

```
OR(FSQL_FUNCTIONS.NFEQ_trape(185,10,10,210,49952,3,fulfillmen
t_degreet,fulfillment_degree1,fulfillment_degree2,fulfillment
_degree3,fulfillment_degree4) >=1
AND
FSQL_FUNCTIONS.FEQ_trape(175,5,10,195,49952,8,importance_degr
eet,importance_degree1,importance_degree2,importance_degree3,
importance_degree4)>=1)
```

FQ: □□SELECT competence_role.%, CDEG(*) FROM competence_role□□WHERE fulfillment_degree FGT $baj...

Archivo Opciones

Núm. de Columnas : 7 Tiempo Traducción: 0,1 sgs.

Filas Recuperadas: 4 (todas) Tiempo Consulta : 0,1 sgs. Cerrar

Nº Fila	ROLE_ID	COMPETENCE_NAME	FULFILLMENT_DEGREE	IMPORTANCE_DEGREE	CDEG(FULFI	CDEG(IMPO	CDEG(*)
1	3	Fluidez	ALTO	NORMAL	1	0,5	1
2	5	Resultados	NORMAL	[10,25]	1	0	1
3	6	Perseverancia	MUY_ALTO	BAJA	1	0	1
4	8	Pensamiento	NORMAL	[12,48]	1	0	1

Figura. 16 Relación resultante del Ejemplo 2.3.

2.5. Conclusiones del capítulo

En este capítulo se realizó el análisis, diseño e implementación de la base de datos difusa para el Sistema de Evaluación de Competencias. Para ello se tuvo en cuenta cada uno de los elementos que fueron expuestos en la fundamentación teórica del presente trabajo.

Se representó el modelo lógico y físico de la base de datos, así como la descripción de cada una de las tablas y del patrón de diseño utilizado. Por último se definieron cada uno de los atributos difusos y se mostraron algunos ejemplos de consultas difusas en el lenguaje FSQL sobre la base de datos diseñada.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

En el presente capítulo se lleva a cabo el desarrollo de la última fase de la metodología XP, realizando para ello una validación de la solución propuesta para el diseño de la base de datos difusa, a través de validaciones teóricas y funcionales. Se definen además aspectos tan importantes como son la integridad, redundancia de la información y seguridad de la base de datos.

3.2. Fase de Prueba

Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de prueba, el cual se realiza de manera continua para asegurar durante todo el proceso de desarrollo, el éxito del producto que se está elaborando. Esto permite elaborar un software de mayor calidad ya que los errores son detectados en un plazo de tiempo corto y se corrigen de una manera más sencilla (23).

3.2.1. Validación teórica del diseño

La etapa de diseño constituye un punto crucial en el proceso de creación de la base de datos. Aspectos como la integridad de los datos, la normalización del diseño, la redundancia de información y la seguridad en la base de datos son de vital importancia no solo en la confección de un buen diseño de base de datos, sino para garantizar la consistencia e integridad de la información, asegurando la calidad de almacenamiento y disponibilidad de los datos.

3.2.1.1. Análisis de la integridad de los datos

La integridad en una base de datos se refiere a la corrección y exactitud de la información contenida. Una base de datos determinada podría estar sujeta a cualquier cantidad de restricciones de integridad de una complejidad arbitraria (29).

Cuando se utilizan sentencias INSERT, DELETE o UPDATE, la integridad de datos puede verse afectada; ya que se puede añadir datos no válidos o modificarse datos existentes de forma incorrecta, con lo que la integridad podría no cumplirse (30).

La integridad en las bases de datos puede clasificarse de las siguientes maneras:

Integridad de dominio:

La integridad de *dominio* (o columna) especifica un conjunto de valores de datos que son válidos para una columna y determina si se permiten valores nulos. La integridad de dominio se suele implementar mediante el uso de comprobaciones de validez y también, mediante la restricción del tipo de datos, el formato o el intervalo de los valores posibles permitidos en una columna (31).

Para lograr la integridad de dominio en la base de datos difusa, se precisaron correctamente cada uno de los tipos de datos para cada uno de los atributos. Además se revisaron los valores que no pueden clasificarse de nulos haciendo uso de la restricción NOT NULL.

Integridad de Entidades:

La integridad de *entidad* (o tabla) requiere que todas las filas de una tabla tengan un identificador exclusivo, conocido como *clave principal*. El que se pueda modificar el valor de la clave principal o eliminar la fila entera depende del nivel de integridad requerido entre la clave principal y cualquier otra tabla (30).

En el caso de la base de datos que se está analizando, cada entidad tiene definida su llave primaria, que no puede ser nula ni se puede repetir. Definir correctamente el identificador principal para cada registro, es de vital importancia, ya que esto representa la principal vía que utiliza el servidor de base de datos para seleccionar la información que se necesite.

Integridad Referencial:

La integridad *referencial* asegura que siempre se mantienen las relaciones entre las claves principales (en la tabla a la que se hace referencia) y las *claves externas* (en las tablas que hacen referencia). No se puede eliminar una fila de una tabla a la que se hace referencia, ni se puede modificar la clave principal, si una clave externa hace referencia a la fila, salvo que se permita la acción en cascada. Puede definir relaciones de integridad referencial dentro de la misma tabla o entre tablas diferentes (30).

Los conceptos de actualizar registros en cascada y eliminar registros en cascada están asociados a la integridad referencial. El primero hace referencia a que cuando se cambie un valor del campo clave de la tabla principal, automáticamente cambiará el valor de la clave foránea de los registros relacionados en la tabla secundaria. El segundo indica al SGBD que cuando se elimina un registro de la tabla principal automáticamente se borran también los registros relacionados en la tabla secundaria (30).

Para mantener la integridad referencial en la base de datos, se utilizaron llaves foráneas que obligan a los valores introducidos en las columnas marcadas por esta restricción a que correspondan con los valores en las tablas referenciadas. Además se definió que la realización de la actualización y eliminación de datos sería en cascada.

3.2.1.2. Normalización de la base de datos difusa

La normalización es un proceso mediante el cual se descomponen relaciones que no cumplen con una serie de requisitos, en otras más pequeñas que lo hagan. Tiene como objetivo la eliminación de la redundancia y anomalías a la hora de efectuar la inserción, actualización y eliminación de campos en tablas (32).

En el presente trabajo, se realizó el diseño de una base de datos difusa, de ahí que se tuvo en cuenta una serie de elementos que normalmente no se analizan cuando se trata del diseño de una base de datos tradicional. Galindo, en su tesis doctoral, describe el procedimiento para la modelación de los atributos difusos, los cuales aparecerán en tablas que no están normalizadas.

La tabla que se muestra a continuación**,** por ejemplo, no está normalizada hasta la 3FN ya que presentaría dependencias transitivas con respecto a la llave primaria, tal y como se muestra en la figura 19.

Figura. 17 Tabla "competence_role" de la BDD.

Si se representara parcialmente la tabla "*competence_role*" como una relación del modelo físico correspondiente a la base de datos, se tendría lo siguiente:

```
Competence_Role(role_id,competence_name,fulfillment_degreeT,
                fulfillment_degree1,fulfillment_degree2,
                fulfillment_degree3,fulfillment_degree4)
```

En este caso los atributos *fulfillment_degree1*, *fulfillment_degree2*, *fulfillment_degree3* y *fulfillment_degree4* dependen funcionalmente de *fulfillment_degreeT* quien a su vez depende de la llave primaria de la tabla. Lo mismo sucede con el atributo difuso *importance_degree* de dicha tabla*.*

Si se decidiera normalizar esta relación, se tendría entonces una nueva relación como la que sigue:

```
Competence_Role(role_id,competence_name,fulfillment_degreeT)
Atributo_Difuso (fulfillment_degreeT,fulfillment_degree1,
                 fulfillment_degree2,fulfillment_degree3,
                 fulfillment_degree4)
```

Anteriormente se había explicado que los atributos difusos se identifican por el identificador de la tabla a la que pertenece (obj) y el identificador de la columna que esta ocupa en dicha tabla (col). Quiere decir esto, que si la relación *Atributo_Difuso* existiera, no fuera posible identificar a los atributos difusos ya que todos pertenecerían a la misma tabla, al tiempo que ocuparían la misma columna. Por esta razón, es necesario pasar por alto el proceso de normalización en dicha base de datos.

3.2.1.3. Análisis de redundancia de información

Al diseñar una BD se debe tener en cuenta que inevitablemente ocupará espacio en memoria. Es por ello la importancia de evitar tener datos repetidos, a este almacenamiento de información repetida se le conoce como redundancia de la información (33).

Como bien se explicó con anterioridad la BDD desarrollada, posee tablas que no están normalizadas, lo que significa que dichas tablas poseen redundancias de información. Ante esta situación solo se pudo eliminar las redundancias en el resto de las tablas que no poseen atributos difusos.

A continuación se representa gráficamente el por ciento de tablas de la BDD libres de redundancia (por tanto normalizadas) y las que no lo están.

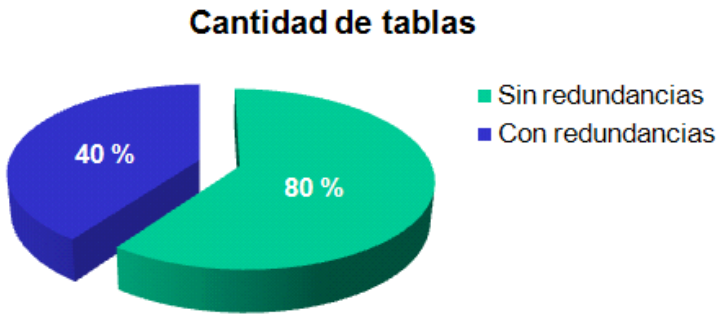


Figura. 18 Representación gráfica de las tablas con redundancias y sin ellas.

3.2.1.4. Análisis de la seguridad de la base de datos

La seguridad es un punto esencial en las base de datos para evitar ataques, impedir cualquier acceso no autorizado con la intención de modificar, usar y/o difundir información almacenada en las BD (34).

El SGBD utilizado para el diseño de la BDD, Oracle, incluye formas de restringir el acceso a la misma. Esta función se denomina control de acceso y se pone en práctica creando cuentas de usuarios y contraseñas para que el SGBD controle el proceso de entrada al sistema.

El administrador de la BD juega un papel importante en la seguridad de la BD porque es quien otorga privilegios a los usuarios, los clasifica, así como a los datos. Las órdenes de los administradores incluyen las siguientes acciones (34):

- Creación de roles o usuarios.
- Concesión de privilegios.
- Revocación de privilegios.

3.2.2. Validación funcional del diseño

Para comprobar el correcto funcionamiento de la base de datos difusa desarrollada, es necesario validarla funcionalmente, para ello en un primer momento, se probó usando la BDD con el Sistema de Evaluación de Competencias para el que fue creada, donde se comprobó la flexibilidad y el correcto funcionamiento de la misma, permitiendo evaluar correctamente a los estudiantes a través del uso de conceptos difusos. Se verificó además

el comportamiento de la BDD bajo una gran carga de información, como se demuestra posteriormente.

3.2.2.1. Prueba de volumen

Las pruebas de volumen centran su trabajo en poblar a la base de datos con volúmenes de datos similares a los esperados en la explotación real, para determinar si existen límites de almacenamiento, identificándose así la carga máxima que puede manejar la base de datos difusa en un período dado (35).

En la Tabla 19, se muestran los resultados alcanzados por la prueba de llenado voluminoso de la base de datos difusa. El llenado de la misma se realizó de forma manual con el uso de los llamados *campos autoincrementales*. En Oracle no existe un atributo para definir este tipo de campos, pero para suplir esta "carencia", dicho SGBD dispone de un objeto denominado secuencia (SEQUENCE), donde la misma posee un valor inicial, un valor máximo y un valor de secuencia que incrementará cada vez que se realice una llamada a dicha secuencia, poblando así cada tabla de la BD con la cantidad de tuplas que se desee.
Las tablas fueron llenadas con un rango de datos equivalentes al que tendrá la base de datos durante el proceso de evaluación de competencias.

Resultados de la prueba de volumen		
Registros insertados por tablas	**Tiempo de ejecución de la BDD**	**Registro total en la BDD**
332	0.10 segundos	2320
765	0.30 segundos	7650
1692	1.00 minuto	16920
2189	2.00 minutos	21890

Tabla. 19 Resultados de la prueba de volumen a la BDD.

En la tabla se muestran cuatro momentos de carga de datos, identificándose el número de registros insertados por tabla, el tiempo de ejecución que tardó la BDD para el llenado de los datos y la cantidad de registros totales en la BD después de terminada cada carga.
En los cuatro momentos se completó la prueba de carga intensiva, lo que demuestra que el gestor utilizado y el diseño de las estructuras de la base de datos soportan completamente el

almacenamiento de los niveles de información requeridos para la puesta en funcionamiento de la base de datos difusa.

3.3. Conclusiones parciales

En este capítulo se realizó la validación teórica y funcional del diseño de la BDD, para demostrar que el mismo ha sido construido correctamente y cumple el objetivo propuesto, dándole cumplimiento a la última de las tareas de investigación.

Con la validación teórica se comprobaron un conjunto de aspectos fundamentales, tales como la integridad, la redundancia de información y la seguridad de la base de datos difusa.

Para la validación funcional se realizó pruebas de rendimiento, específicamente las pruebas de volumen, permitiendo conocer el comportamiento de la base de datos difusa ante una sobrecarga de información, obteniéndose en la misma resultados favorables.

CONCLUSIONES GENERALES

Con la realización de la presente investigación se ha dado cumplimiento al objetivo general propuesto, arribando a las siguientes conclusiones:

- Los sistemas gestores de BDD desarrollados en una plataforma libre presentaban inconsistencias a la vez que carecían de funcionalidades básicas para la ejecución de consultas difusas, de ahí que se hizo necesario utilizar un paquete de funciones implementadas para Oracle, el cual brinda todas las funciones básicas necesarias para el presente trabajo.
- Se evidenció la flexibilidad del lenguaje FSQL a partir de consultas capaces de filtrar información imprecisa o con cierto grado de incertidumbre.
- El proceso de normalización se vio afectado debido a la naturaleza de las BDD, las cuales identifican a los atributos difusos por el identificador de la tabla a la que pertenece y el identificador de la columna que esta ocupa en dicha tabla. Es por ello que fue necesario repetir información para poder tratarla como filas en algunos casos y como columnas en otros.
- La validación realizada al diseño propuesto de la base de datos difusa, garantiza la integridad, el correcto funcionamiento y el rendimiento de la misma.

De manera general, se puede concluir que este trabajo ha llevado a la realización satisfactoria de una base de datos difusa, capaz de satisfacer los requerimientos del sistema. Cumpliendo exitosamente cada una de las tareas de investigación formuladas inicialmente, y de esta forma contribuir en el proceso de evaluación de competencias en los estudiantes universitarios.

RECOMENDACIONES

Al culminar el desarrollo del presente trabajo de diploma se recomienda:

- Implementar la BDD en el SGBD PostgreSQL, el cual es libre y las versiones de bases de datos difusas que existen están incompletos.
- Realizar modificaciones a la estructura interna de las bases de datos difusas con el objetivo de garantizar la normalización en todas las tablas de la base de datos.
- Implementar los cuantificadores difusos para lograr una mayor flexibilidad en las consultas difusas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Lorenzo García, Raquel and Martínez Llantada, Marta.** *Talento para la Ciencia: estrategia para su desarrollo.* La Habana : Academia, 1999.

2. **Galindo Gómez, José.** *Tratamiento de la Imprecisión en Base de Datos Relacionales: Extensión del Modelo de Adaptación de los SGBD actuales.* Granada : s.n., 1999.

3. **Cruz Chávez, Marco Antonio.** *Conceptos básicos de bases de datos.*

4. Definición ABC. [Online] 2013. http://www.definicionabc.com/tecnologia/base-de-datos.php.

5. **Ramos Martín, Alicia and Ramos Martín, Jesús.** *Operaciones con base de datos ofimáticas y corporativas.* Madrid : s.n., 2008.

6. **Martínez Romero, Julio Alfonso.** *Gestores de Base de Datos.* México : s.n., 2011.

7. **Urrutia Sepúlveda, Angélica.** *Definición de un modelo conceptual para base de datos difusas.* Madrid : s.n., 2003.

8. **Lotfi A., Zadeh.** *Fuzzy Sets.*

9. **Oliva Moreno, Rafael Francisco and Galindo Gómez, José.** *Visual FSQL: Gestión visual de bases de datos difusas en Oracle a través de Internet usando FSQL.* Málaga : s.n., 2003.

10. **Urrutia, Angélica, Galindo, José and Sepúlveda, Alejandro.** *Implementación de una base de datos difusa con FIRST-2 y PostgreSQL.* Huelva : s.n., 2010.

11. **Wong Cruz, Christian J. and Flores Velarde, Miluska Yamilé.** *Fuzzy Queries. Un framework para realizar consultas difusas.* 2008.

12. **Larman, Craig.** *UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Vancouver : s.n.

13. **Urrutia S, Angélica, Varas C, Marcela and Galindo G., José.** *Diseño de una Base de Datos Difusa Modelada con UML.* 2010.

14. **Patrick, Bosc and Oliver, Pivert.** *SQLf: A Relational Database Language for Fuzzy Querying.*

15. **Urrutia, Angélica, Tineo, Leonid and González, Claudia.** *FSQL and SQLf:Towards a Standard in Fuzzy Databases.* Chile : s.n., 2008.

16. **Ancajima Miñán, Víctor.** *Modelamiento de Bases de Datos con Software Libre. DBDesigner.* 2010.

17. Free Download Manager. *DBDesigner (Diseñador de Bases de Datos).* [Online] diciembre 2003. [Cited: mayo 30, 2013.] http://www.freedownloadmanager.org/es/downloads/DBDesigner_4_3178_p/.

18. *Oracle database 10g standard edition.* 2010.

19. **Faci, Santiago.** *Lenguaje PL/SQL.* 2004.

20. ABCdatos. Programas y tutoriales en castellano. [Online] 2000. http://www.abcdatos.com/programas/programa/l537.html.

21. **G. Figueroa, Roberth, J. Solís, Camilo and A. Cabrera, Armando.** *Metodologías tradicionales vs. Metodologías ágiles.* Loja : s.n., 2011.

22. **Alonso, Fernando, Martínez, Loic and Segovia Pérez, Javier.** *Introducción a la Ingeniería del Software.* Zaragosa : s.n., 2005.

23. **H. Canós, , José, Letelier, Patricio and Penadés, Maria Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n., 2010.

24. **Sommerville, Ian.** *Ingeniería del Software. Séptima edición.* Madrid : s.n., 2006.

25. **Campderrich Falgueras, Benet.** *Ingeniería del Software.* Barcelona : UOC, 2003.

26. **Guerrero, Jedutún.** Metodologías Ágiles de desarrollo de software (XP) Fases. [Online] 2008. http://boards5.melodysoft.com/UBV_INGS/metodologias-agiles-de-desarrollo-43.html.

27. **Gil, Fidel, Albrigo, Javier and Do Rosario, Javier.** *Sistemas de Gestión de Bases de Datos. SGBD/DBMS.* Valencia : s.n., 2005.

28. EcuRed. Conocimiento con todos y para todos. [Online] [Cited: abril 25, 2013.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

29. **Savedra, J.** *Integridad Informática enfocada a las bases de datos.* 2009.

30. **Microsoft.** Microsoft SQL Server. Integridad de los datos. [Online] 2008. [Cited: mayo 13, 2013.] http://technet.microsoft.com/es-es/library/ms184276%28v=sql.105%29.aspx.

31. **Wordpress.** Definición.de. [Online] 2008. [Cited: mayo 15, 2013.] http://definicion.de/integridad/.

32. **MySQL-hispano.** *Normalización de bases de datos.* 2003.

33. **García, R.M.M.** *Sistemas de Bases de Datos. Segunda Edición.* La Habana : Editorial Pueblo y Educación, 2005.

34. **Consultores S.A.** TestHouse. *Pruebas de rendimiento.* [Online] 2006. [Cited: mayo 17, 2013.] http://www.es.testhouse.net/pruebas-de-rendimiento/.

35. **Pozo Coronado, Salvador.** MySQL con clase. [Online] 2005. [Cited: mayo 19, 2013.] http://mysql.conclase.net/curso/?cap=001.

BIBLIOGRAFÍA

Lorenzo García, Raquel y Martínez Llantada, Marta. *Talento para la Ciencia: estrategia para su desarrollo.* La Habana : Academia, 1999.

Ortíz Torres, Emilio, Aguilera Pupo, Eleanne y González Pérez del Villar, Ana María. *Los estilos de aprendizaje, la superdotación intelectual y el talento en estudiantes universitarios.* Holguín : s.n., 2010.

Poza Latorre, Irene. *A la búsqueda del talento universitario.* 2004.

Galindo Gómez, José. *Tratamiento de la Imprecisión en Base de Datos Relacionales: Extensión del Modelo de Adaptación de los SGBD actuales.* Granada : s.n., 1999. Tesis doctoral.

Jiménez, Leoncio, Valdés, Yolanda y Vistoso, Jorge. *Protocolos de Representación de Conocimiento Impreciso e Incierto en una Base.* Talca : s.n., 2003.

T. Cadenas, J., Aguilera, A. y Tineo, L. *Benchmarking de Consultas Difusas utilizando PostgreSQL.* Caracas : s.n., 2012.

Galindo Gómez, Jose. *FSQL, un Lenguaje para Consultas Difusas: Definición e Implementación de una BDRD.* Málaga : s.n.

Urrutia Sepúlveda, Angélica. *Definición de un modelo conceptual para base de datos difusas.* Madrid : s.n., 2003.

Lotfi A., Zadeh. *Fuzzy Sets.* Tesis doctoral.

Oliva Moreno, Rafael Francisco y Galindo Gómez, José. *Visual FSQL: Gestión visual de bases de datos difusas en Oracle a través de Internet usando FSQL.* Málaga : s.n., 2003.

Urrutia, Angélica, Galindo, José y Sepúlveda, Alejandro. *Implementación de una base de datos difusa con FIRST-2 y PostgreSQL.* Huelva : s.n., 2010.

Wong Cruz, Christian J. y Flores Velarde, Miluska Yamilé. *Fuzzy Queries. Un framework para realizar consultas difusas.* 2008.

Urrutia, Angélica, Tineo, Leonid y Gonzalez, Claudia. *FSQL and SQLf:Towards a Standard in Fuzzy Databases.* Chile : s.n., 2008.

Patrick, Bosc y Oliver, Pivert. *SQLf: A Relational Database Language for Fuzzy Querying.*

Urrutia S, Angélica, Varas C, Marcela y Galindo G., José. *Diseño de una Base de Datos Difusa Modelada con UML.* 2010.

Cruz Chávez, Marco Antonio. *Conceptos básicos de bases de datos.*

Definición ABC. [En línea] 2013. http://www.definicionabc.com/tecnologia/base-de-datos.php.

Ramos Martín, Alicia y Ramos Martín, Jesús. *Operaciones con base de datos ofimáticas y corporativas.* Madrid : s.n., 2008.

Martínez Romero, Julio Alfonso. *Gestores de Base de Datos.* México : s.n., 2011.

Mota Herranz, Laura , Casamayor Ródenas, Juan Carlos and Celma Giménez, Matilde. *Bases de datos relacionales: teoría y diseño.* 1999.

Date, C. J. *Introducción a los Sistemas de Base de Datos.* 1993.

EcuRed. [En línea] [Citado el: 24 de abril de 2013.] http://www.ecured.cu/index.php/Oracle.

Silberschatz, Abraham. *Fundamentos de Bases de Datos. Cuarta.* 2002.

Pons, Olga. *Introducción a las bases de datos: el modelo relacional.* 2005.

ABCdatos. Programas y tutoriales en castellano. [En línea] 2000. http://www.abcdatos.com/programas/programa/l537.html.

EcuRed. Conocimiento con todos y para todos. [En línea] [Citado el: 25 de abril de 2013.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

Ancajima Miñán, Víctor. *Modelamiento de Bases de Datos con Software Libre. DBDesigner.* 2010.

Free Download Manager. *DBDesigner (Diseñador de Bases de Datos).* [Online] diciembre 2003. [Cited: mayo 30, 2013.] http://www.freedownloadmanager.org/es/downloads/DBDesigner_4_3178_p/.

EcuRed. Conocimiento con todos y para todos. [En línea] 2010. http://www.ecured.cu/index.php/Herramienta_CASE.

Murillo Alfaro, Félix. *Herramientas CASE.* 1999.

Campderrich Falgueras, Benet. *Ingeniería del Software.* Barcelona : UOC, 2003.

Sommerville, Ian. *Ingeniería del Software. Séptima edición.* Madrid : s.n., 2006.

Fernández Escribano, Gerardo. *Introducción a Extreme Programming.* 2002.

Guerrero, Jedutún. Metodologías Ágiles de desarrollo de software (XP) Fases. [En línea] 2008. http://boards5.melodysoft.com/UBV_INGS/metodologias-agiles-de-desarrollo-43.html.

H. Canós, , José, Letelier, Patricio y Penadés, María Carmen. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n., 2010.

G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando. *Metodologías tradicionales vs. Metodologías ágiles.* Loja : s.n., 2011.

Alonso, Fernando, Martínez, Loic y Segovia Pérez, Javier. *Introducción a la Ingeniería del Software.* Zaragosa : s.n., 2005.

Gil, Fidel, Albrigo, Javier y Do Rosario, Javier. *Sistemas de Gestión de Bases de Datos. SGBD/DBMS.* Valencia : s.n., 2005.

Faci, Santiago. *Lenguaje PL/SQL.* 2004.

Pozo Coronado, Salvador. MySQL con clase. [En línea] 2005. [Citado el: 19 de mayo de 2013.] http://mysql.conclase.net/curso/?cap=001.

Oracle database 10g standard edition. 2010.

Larman, Craig. *UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Vancouver : s.n.

Mota Herranz, Laura y Celma Giménez, Matilde. *Métodos para la comprobación de la integridad en base de datos deductivas.* Valencia : s.n.

Microsoft. Microsoft SQL Server. Integridad de los datos. [En línea] 2008. [Citado el: 13 de mayo de 2013.] http://technet.microsoft.com/es-es/library/ms184276%28v=sql.105%29.aspx.

Olmeda Matos, Ignacio y Moratilla Ocaña, Antonio. *Restricciones de Integridad. Base de Datos.* Alcalá : s.n., 2007.

Wordpress. Definición.de. [En línea] 2008. [Citado el: 15 de mayo de 2013.] http://definicion.de/integridad/.

Savedra, J. *Integridad Informática enfocada a las bases de datos.* 2009.

GLOSARIO DE TÉRMINOS

UCI: Universidad de las Ciencias Informáticas.

CICE: Centro de Innovación y Calidad de la Educación.

Difuso: Ambiguo, vago, impreciso.

BDD: Base de Datos Difusa.

SQL: Structured Query Language, Lenguaje de Consulta Estructurado.

FSQL: Fuzzy Structured Query Language, Lenguaje de Consulta Estructurado Difuso.

DML: Lenguaje de Manipulación de Datos.

DDL: Lenguaje de Definición de Datos.

FIRST: Fuzzy Interface for Relational SysTems, Interfaz Difuso para Sistemas Relacionales.

FMB: Fuzzy Metaknowledge Base, Base de Metaconocimiento Difuso.

Tupla: Conjunto de todos los valores concretos de todos los atributos de una determinada entidad u objeto.

Atributo: Es cada una de las cualidades, propiedades o características de un elemento.

Clase: Es la declaración o abstracción de objetos, lo que significa, que una clase es la definición de un objeto.

SGBD: Sistema Gestor de Base de Datos.

UML: Lenguaje Unificado de Modelado.

Normalización: Proceso de reducción sobre una estructura de datos que procura aumentar la integridad, disminuir la redundancia y las dependencias funcionales de esa estructura.

Redundancia: Repetición de información.

ANEXOS

Anexo 1. Tablas principales de la FMB de la BDD desarrollada para el Sistema de Evaluación de Competencias:

	OBJ#	COL#	F_TYPE	LEN	COM
1	49948	3	2	1	EVE.PERSON.PENSAMIENTO_INDEPENDIENTET
2	49948	8	2	1	EVE.PERSON.RESULTADOS_SOCIALEST
3	49952	8	2	1	EVE.PROFICIENCY_ROLE.IMPORTANCE_DEGREET
4	49952	3	2	1	EVE.PROFICIENCY_ROLE.FULFILLMENT_DEGREET
5	49942	3	2	1	EVE.EVALUATOR.CERTAINTYT

Figura. 19 FCL (FUZZY_COL_LIST)

	OBJ#	COL#	FUZZY_ID	FUZZY_NAME	FUZZY_TYPE
1	49952	8	0	BAJA	0
2	49952	8	1	NORMAL	0
3	49952	8	2	ALTA	0
4	49952	8	3	MUY_ALTA	0
5	49952	3	0	BAJO	0
6	49952	3	1	NORMAL	0
7	49952	3	2	ALTO	0
8	49952	3	3	MUY_ALTO	0
9	49942	3	0	BAJO	0
10	49942	3	1	MEDIO	0
11	49942	3	2	ALTO	0
12	49948	3	0	MALO	0
13	49948	3	1	NORMAL	0
14	49948	3	2	BUENO	0
15	49948	3	3	MUY_BUENO	0
16	49948	8	0	BAJOS	0
17	49948	8	1	NORMALES	0
18	49948	8	2	BUENOS	0
19	49948	8	3	MUY_BUENOS	0

Figura. 20 FOL (FUZZY_OBJECT_LIST)

	OBJ#	COL#	FUZZY_ID	ALFA	BETA	GAMMA	DELTA
▶ 1	49952	8	0	0	0	175	180
2	49952	8	1	175	180	185	195
3	49952	8	2	185	195	200	210
4	49952	8	3	200	210	300	1000
5	49952	3	0	0	0	175	180
6	49952	3	1	175	180	185	195
7	49952	3	2	185	195	200	210
8	49952	3	3	200	210	300	500
9	49942	3	0	0	3	7	10
10	49942	3	1	7	20	60	80
11	49942	3	2	70	80	95	100
12	49948	3	0	0	0	5	10
13	49948	3	1	5	10	15	20
14	49948	3	2	10	20	30	45
15	49948	3	3	30	35	100	100
16	49948	8	0	0	0	5	10
17	49948	8	1	5	10	15	20
18	49948	8	2	10	20	30	45
19	49948	8	3	30	35	1000	1000

Figura. 21 FLD (FUZZY_LABEL_DEF)

	OBJ#	COL#	MARGEN	MUCH
▶ 1	49952	8	6	12
2	49952	3	10	15
3	49942	3	10	30
4	49948	3	3	9
5	49948	8	5	12

Figura. 22 FAM (FUZZY_APPROX_MUCH)