# EL TRATAMIENTO DE LA INFORMACIÓN DIFUSA

**Autor: Lic. Patricia Silvia Mac Gaul**
**Director: Mag. Gustavo Daniel Gil**
**Co-Director: Dr. Gustavo Rossi**

**TRABAJO FINAL INTEGRADOR**
**presentado para obtener el grado de**
**ESPECIALISTA EN INGENIERÍA DE SOFTWARE**

**Facultad de Informática**
**Universidad Nacional de La Plata**

**Junio de 2.012**

*A María José, Facundo,*
*Camila, Agustín,*
*Milagros, Lucas*
*y Felipe*

# ÍNDICE

# INTRODUCCIÓN

El almacenamiento y la manipulación de los datos que han sido almacenados según una estructura formal como la que brinda, por ejemplo, el Modelo Relacional, funciona de manera altamente eficiente.

Las consultas que se pueden efectuar cuando se trabaja con este tipo de modelos, tipo de datos y estructuras de almacenamiento, poseen una forma “cerrada”. Por ejemplo: “lista de alumnos que se encuentran en condiciones de rendir la asignatura Bases de Datos”, o, “lista de personas que tienen 25 años”.

Sin embargo, este tipo de consultas no es suficiente a la hora de encarar una solicitud en un Sistema para la Toma de Decisiones. ¿Qué pasaría si se formulara la siguiente petición: “apellido y nombre de las personas jóvenes”?. Si uno de los atributos almacenados en alguna relación fuese la edad, su valor sería tipo numérico, de modo que ¿cómo compararíamos un valor numérico con “joven”. Con la estructura mencionada: imposible.

¿Cuál es la solución?. Para responder esta pregunta tenemos que pensar en dos aspectos. El primero es cómo almacenar esa información imprecisa: ¿qué tipo de datos es “joven”?. El segundo es cómo hacer para que el motor entienda, ¡y resuelva!, una consulta como la enunciada: “apellido y nombre de las personas jóvenes”.

Una propuesta de solución son las Bases de Datos Difusas.

Un concepto difuso es un concepto que encierra alguna imprecisión o incertidumbre. El problema de la representación y/o manipulación de conceptos difusos no es sencillo de resolver puesto que implica una modificación de las estructuras sugeridas para cada modelo de datos y para las operaciones definidas sobre tales estructuras.

En el presente trabajo se presentan dos modelos diferentes de Bases de Datos Difusas y se pretende, a partir de la inserción de conceptos de uno de ellos en el otro modelo, mejorar la perfomance de una consulta difusa.

# CAPÍTULO 1

# DE LOS FICHEROS SECUENCIALES A LAS BASES DE DATOS DIFUSAS

El primer intento por guardar físicamente datos persistentes, esto es, datos cuyo almacenamiento debe mantenerse a través del tiempo, está relacionado con el uso de los archivos secuenciales. Si bien este uso permitió tal almacenamiento, las aplicaciones que manipulaban estos archivos eran un conjunto de programas en algún lenguaje de alto nivel pero sin capacidad para control de aspectos fundamentales en el logro de información confiable: redundancia, seguridad e inconsistencia, para no ahondar en la ineficiencia en la recuperación de los datos.

Pero el descontrol y la ineficiencia no se debían a fallas en el lenguaje ni a falta de datos almacenados. El punto crítico era la forma en que se almacenaban estos datos, en realidad, a una falta total de *alguna* forma de almacenar los datos. Si existiese esa forma, estaría solucionado el tema de la eficiencia y quedaría por resolver el de los controles.

A partir de los años 70, la investigación se centró en la obtención de modelos de datos, justamente con el fin de encontrar *alguna* forma de almacenarlos con el objetivo de lograr información en forma eficiente.

Un modelo es una representación simplificada de un sistema real. En particular, un modelo de datos no solo debe representar las características estáticas del sistema que se pretende modelar, sino también debe reflejar sus características dinámicas.

Para efectuar este modelo, se utilizan las siguientes componentes:

- un conjunto de objetos y sus interrelaciones, que representan las características estáticas del sistema, incluyendo las propiedades de los objetos, que se denomina Modelo Conceptual

- un conjunto de operaciones o lenguaje, que representa las características dinámicas

- un conjunto de restricciones sobre los objetos, sus interrelaciones y las operaciones que actúan sobre ellos.

En el año 1.971, el Comité CODASYL: Conference of Data Systems Languages, define el Modelo de Red y sugiere Cobol como lenguaje anfitrión. Para representar el Modelo Conceptual utiliza dos estructuras básicas: los registros y los conjuntos. Se instrumenta también un Sistema Administrador de una Base de Datos de Red que posee un lenguaje que permite definir y manipular los datos.

Con posterioridad, teniendo en cuenta que la mayoría de las estructuras del mundo real que deben modelarse son de naturaleza jerárquica, surge el Modelo Jerárquico. La estructura de la base de datos jerárquica se formaliza a través del uso de vínculos padre-hijo, sobre los cuales se definen un conjunto de restricciones. Se pone en funcionamiento un motor, IMS: Information Management System, que manipula datos pero que no cuenta con un lenguaje de consulta integrado.

En el año 1.976, Chen presenta el Modelo Entidad-Relación. En este modelo, los datos se describen como entidades, atributos y vínculos. Estos últimos poseen como característica una restricción de cardinalidad, que limita la cantidad de entidades que pueden participar en una relación. Los motores que se construyen para este tipo de modelo funcionan con eficiencia sobre el modelo conceptual definido, sin embargo, este último carece totalmente de sustentación formal y, por lo tanto, las excepciones se siguen resolviendo “a como se presenten”.

A fines de los años 70, Codd introduce el Modelo Relacional de datos. El mérito de este modelo fue la propuesta de una sólida base matemática, simple y bien definida, para el estudio de los problemas de las bases de datos.

El Modelo Relacional representa conceptualmente a la base de datos como una colección de relaciones, cada una de las cuales puede percibirse como una tabla con una cierta cantidad de filas, representando cada fila un conjunto de valores relacionados entre sí. Las restricciones del Modelo Relacional apuntan al manejo eficiente del tipo de datos, a la identificación unívoca de cada conjunto de datos y al enlace eficiente de los datos entre sí.

Las relaciones que participan en el Modelo Conceptual deben implementarse respetando un proceso de normalización, sustentado en bases teóricas firmes que asegura la mejor representación del conjunto de datos: por primera vez había *alguna* manera formal, sustentable y aplicable de representación conceptual, lógica y física de los datos.

Los motores relacionales se construyen para implementar las operaciones entre los datos. Desde el Modelo Relacional, estas operaciones están definidas, también, en base a un marco teórico: el Álgebra Relacional, que provee un conjunto de operadores de alto nivel que actúan sobre las relaciones. Estas operaciones se implementan a través de un lenguaje de consulta netamente estructurado: SQL.

Es por ello y, además, por los años que lleva la implementación física de las bases de datos relacionales y su gestión a través de un DBMS Relacional, que ha hecho que en la actualidad, el gestión de datos utilizando este modelo alcance un 100 % de eficiencia.

¿Entonces por qué surgen los modelos que le siguen al Relacional?. Porque, como en todo, la eficiencia llega hasta la limitación: los esquemas relacionales funcionan muy bien en las aplicaciones tradicionales como, por ejemplo, el procesamiento de datos en negocios y en la industria.

De modo que los modelos posteriores al Modelo Relacional surgen como necesidad de dar una solución al procesamiento de datos para aplicaciones de bases de datos como diseño en ingeniería, procesamiento de imágenes y gráficos, cartografías, multimedia e inteligencia artificial, por nombrar algunas pocas. Para el diseño de este tipo de bases de datos, deben utilizarse conceptos adicionales de modelado semántico.

Los sistemas de bases de datos orientados a objetos tienen sus orígenes en la programación orientada a objetos. Básicamente, la idea es manejar objetos que definan a sus equivalentes, o modelados, del mundo real y establecer las operaciones que cada objeto puede soportar.

¿Y los motores orientados a objetos?. Varios fabricantes han lanzado al mercado diversos productos denominados “motores objeto-relacionales”, también conocidos como “servidores universales” como, por ejemplo, Universal Database de DB2 Universal Data Option para Informix y Oracle.

¿Por qué usar un motor relacional para implementar un modelo conceptual orientado a objetos?. Porque los motores relacionales, como consecuencia de más de 30 años de avances en la investigación del tema, son tecnológicamente sólidos, en cambio, el modelo conceptual orientado a objetos no tiene aún la suficiente formalización como para que su implementación sea robusta.

Hasta acá, podríamos decir que los sistemas de bases de datos, a través de sus distintos enfoques, han resuelto el problema de la organización de los datos persistentes necesarios para los sistemas de información. Sin embargo no es así: lamentablemente, los sistemas de bases de datos que hemos descripto no producen la información necesaria a los requerimientos de la toma de decisión gerencial.

Para cubrir esa deficiencia, aparecen los conceptos del Data Warehouse. W.H.Inmon, padre del Data Warehouse, ha definido: “Un Data Warehouse es un conjunto de datos integrados, orientados a una materia, que varían con el tiempo y que son transitorios, los cuales soportan el proceso de toma de decisiones de una administración.”

Un Data Warehouse es un depósito semánticamente consistente de datos, que llenan completamente los requerimientos de acceso y de reporte. Los datos son obtenidos de fuentes heterogéneas y soportan la constante necesidad de consultas estructuradas y no estructuradas, reportes analíticos y soporte de decisiones.

Otra alternativa a la organización de los datos para los Sistemas de Información para la Toma de Decisiones, DSS, son las Bases de Datos Difusas.

Un concepto difuso es un concepto que encierra alguna imprecisión o incertidumbre. Los conceptos difusos son manejados por los seres humanos de manera habitual, cotidiana. Por ejemplo, el concepto de lugar “cercano”, claramente nos deja entender que no se trata de un lugar situado a 5.000 Km. del lugar en donde estamos, pero, un lugar que esté a 33 Km. ¿será cercano? o, para ser cercano, ¿el lugar tendrá que estar a menos de 5 Km. de distancia?.

Es imposible, al menos muy ineficiente usando un sistema de bases de datos relacional o uno orientado a objetos, resolver una consulta simple como: “lista de lugares cercanos”.

Existe una gran cantidad de modelos sugeridos para dar solución al problema de la representación y/o manipulación de la información difusa. El problema no es sencillo de resolver puesto que implica una modificación de las estructuras sugeridas por cada uno de los modelos de datos y, por ende, de las operaciones definidas sobre tales estructuras.

Los modelos sugeridos para el tratamiento de la información difusa, se clasifican en dos grupos:

- aproximaciones sin utilizar Lógica Difusa
- modelos de Bases de Datos Difusas.

Las primeras consisten en extensiones del Modelo Relacional que permiten almacenar información complementaria que se usa a la hora de resolver una consulta difusa. En cambio, las Bases de Datos Difusas unen la teoría de las bases de datos, especialmente la enunciada por el Modelo Relacional, con la teoría de los Conjuntos Difusos.

Cualquiera de las dos líneas tienen los mismos objetivos:

- el almacenamiento de la información difusa, conviviendo con la información no difusa, que, en este contexto, se denomina crisp

- el tratamiento y consulta difusa de la información almacenada

Las propuestas para el logro de estos dos objetivos son múltiples, incluyendo las consultas difusas a las bases de datos clásicas, sin información difusa.

# CAPÍTULO 2

# LA INFORMACIÓN IMPERFECTA

Un modelo es una versión abstracta del mundo real. Como en cualquier modelo, lo importante es el grado de consistencia entre tal modelo y la realidad que se pretendía modelar. Desafortunadamente, nuestro conocimiento del mundo real es imperfecto de modo que es muy difícil asegurar tal consistencia.

Hay dos soluciones para este problema: la primera de ella es restringir el modelado a solo aquélla parte del mundo real de la cual poseamos toda la información. La segunda es permitir que los modelos puedan representar tanto la información perfecta como la imperfecta.

En el caso particular del modelo sea un sistema de información, tenemos dos componentes que permitirán realizar la representación:

- una componente declarativa para describir al mundo real y
- una componente operacional para manipular esa descripción.

A su vez, la manipulación puede ser de dos clases diferentes:

- modificación de la descripción, para reflejar los cambios ocurridos en una componente descripta o
- transformación de la descripción, para reflejar los cambios estructurales en la descripción.

Así, un sistema de información está compuesto por un elemento abstracto, D, un conjunto de modificaciones y un conjunto de transformaciones. Cada modificación, m, reemplaza la descripción presente por una nueva descripción. Cada transformación, t, introduce una modificación estructural de la descripción.

Por ejemplo, en una base de datos relacional, la descripción es un conjunto de tablas, una modificación tendrá efecto sobre los valores almacenados en una tabla y una transformación podría reducir un conjunto de tablas a una sola de ellas.

La información que se provee a través de un sistema será, entonces, el resultado de aplicar una transformación sobre una componente declarativa, t(D). Pero el resultado t(D) puede ser imperfecto por una o varias de las siguientes razones:

- la descripción inicial de D es imperfecta
- la especificación de la transformación t a aplicar resultó imperfecta
- una modificación, m, efectuada sobre D, posterior a la creación de D, resultó imperfecta
- el procesamiento de t sobre D resultó imperfecto.

## Descripciones imperfectas

La descripción contiene, básicamente, tres tipos de imperfecciones diferentes:

- Error: la información almacenada es errónea cuando es diferente de la información verdadera.

- Imprecisión: la información almacenada es imprecisa cuando viene dada por un conjunto de valores posibles y el valor real es un valor del conjunto.

  La imprecisión, a su vez, puede ser de de varios tipos:

  - disyuntivo: María José tiene 10 u 11 años

  - negativo: Facundo no tiene 20 años,

  - en rango: Camila tiene entre 3 y 7 años, o bien, Camila tiene más de 4 años,

  - con errores marginales: Agustín tiene $3 \pm 1$ año.

  La información imprecisa tiene dos extremos:

  - valor preciso: es un valor cuyo conjunto de valores posibles es unitario

  - valor nulo: es un valor no disponible

  Como se puede apreciar, la información imprecisa abarca el dominio completo de valores posibles.

- Incertidumbre: la información almacenada es incierta cuando viene dada por un valor no preciso pero probable.

  Por ejemplo: Milagros tiene, probablemente, 3 años es información incierta y es distinto, desde el punto de vista de la información imperfecta, decir Milagros tiene 3 ó 4 años, que es información imprecisa.

<u>El valor nulo y el valor disyuntivo</u>

En todos los sistemas de información ocurre con frecuencia que algún elemento de la descripción no puede señalarse con total precisión y certidumbre.

La solución propuesta a ese tema es aceptar que parte del modelo puede tener una pseudo-descripción y que ésta estará representada por un valor particular llamado valor nulo.

Un valor nulo denota imposibilidad de definir perfectamente cuál es el valor real del elemento descriptivo. O sea, semánticamente, un valor nulo es alguno de los valores definidos el dominio, pero cualquiera de ellos tiene igual probabilidad de ser el valor verdadero.

Pero, una vez admitido el nulo como un valor del elemento de descripción, el modelo debe definir sin ambigüedades el comportamiento del elemento de descripción que contiene el nulo ante las transformaciones y ante las modificaciones.

En otros casos, el valor nulo puede tomar valores dentro de un subconjunto del dominio, valores entre los cuales, obviamente, está el valor verdadero. Esta información parcial se modela a través de los llamados valores disyuntivos. Un valor disyuntivo es un conjunto de valores entre los cuales se encuentra el valor verdadero del elemento de descripción.

Notemos que, entonces, un valor nulo es un caso particular del valor disyuntivo en donde el conjunto de donde toma valores es todo el dominio.

Obviamente, tanto los valores nulos como los disyuntivos representan información imprecisa.

Factores de confianza

Un factor de confianza, o peso, o coeficiente de relevancia es un valor que indica con qué grado de certeza el valor del elemento de descripción es el valor verdadero.

¿En qué caso usamos factores de confianza?. Supongamos que tenemos un elemento de descripción que tiene valor disyuntivo dado por el siguiente conjunto:

d = { 10, 11, 20, 43, 80 }

Si, en vez de definir el conjunto d como un conjunto de valores lo definimos como un conjunto de pares ordenados

$< v_i , p_i >$

donde:

$v_i$ es un valor
$p_i$ es la posibilidad de que el valor $v_i$ sea el verdadero valor
$p_i \in ( 0 , 1 )$

siguiendo el ejemplo, tenemos:

d = { (10,0.3), (11,0.4), (20,0.8), (43,0.5), (80,0.6) }

el valor que suponemos *menos impreciso* del elemento de descripción es 20, porque es el más posible.

El concepto de distancia

Una propuesta de tratamiento de la información imprecisa que ha dado muy buenos resultados en los sistemas de información es el concepto de distancia entre dos elementos de descripción, lo que deriva en la creación de elementos de descripción *vecinos*.

Extendiendo este concepto de vecindad, ante una consulta será posible no solo informar los elementos que tienen el valor consignado sino todos aquellos elementos que tengan un valor vecino al solicitado.

Supongamos el caso en que un sistema de información haya sido modelado relacional. Ante una consulta que debe devolver las tuplas que tengan un valor *v* determinado en un atributo, devolverá ese conjunto de tuplas pero, además, agregará a la solución de la consulta todas aquellas tuplas en las cuales el valor del atributo en cuestión diste *poco* del verdadero valor *v*. ¿Qué es *poco*?, es un valor que debe consignarse al formular la consulta. Es decir, la solución de la consulta es el conjunto de tuplas con valor *v* más todas las tuplas vecinas respecto al atributo implicado.

El gran avance de la incorporación del concepto de vecindad, es que de la misma forma en que se resuelve una consulta en donde los valores almacenados son imprecisos y los de la consulta precisos, también se resuelve la consulta en donde los valores almacenados son precisos y los parámetros de la consulta son imprecisos.

## Transformaciones imperfectas

Una transformación es una operación que produce una descripción a partir de una o varias descripciones almacenadas.

La forma más usual de una transformación es una consulta y los errores en ellas se producen por diferentes factores:

- falta de conocimiento sobre exactamente qué información está almacenada y/o de cómo está almacenada
- idea vaga de qué es exactamente lo que se desea consultar, por ejemplo, se desea una lista de productos “económicos”
- conocimiento insuficiente de las herramientas y/o del lenguaje de consulta.

## Modificaciones imperfectas

Al igual que las transformaciones, la modificaciones están definidas por el usuario, quien podría introducir factores de imperfección debidos a:

- conocimiento insuficiente del sistema y/o de las herramientas
- conocimiento insuficiente de los datos que contiene el sistema
- desconocimiento de la información contenida en una modificación.

## Procesamientos imperfectos

Aunque una descripción, D, y una transformación, t, sean perfectas, t(D) podría resultar imperfecta debido a los métodos usados para la obtención de dicha t(D).

Este tipo de imperfecciones en t(D) se dan generalmente en casos de consultas recursivas que están asociadas a un período de tiempo para su finalización y, cuando el tiempo es el determinante, el resultado de t(D) es incompleto. Otro tipo de consulta que arroja resultados imperfectos es la que involucra randomizaciones.

# CAPÍTULO 3

# EL MODELO RELACIONAL

El Modelo Relacional representa a la base de datos como una colección de relaciones, cada una de las cuales puede percibirse como una tabla con una cierta cantidad de filas, representando cada fila una colección de valores relacionados entre sí.

A fin de formalizar el concepto de *relación*, vamos a establecer las siguientes

## Definiciones

*Base de datos*: una base de datos es un conjunto de datos almacenados, normalmente en dispositivos electrónicos de un computador, que son manipulados por un programa denominado Sistema Administrador de la Base de Datos, DBMS, o Motor de la Base de Datos.

*Dominio*: un dominio es un conjunto de valores atómicos, todos de un mismo tipo.

Cada elemento de este conjunto constituye la menor unidad semántica de datos, esto es, no existe una descomposición del valor que aporte significado.

*Atributo*: es una característica descriptiva de un objeto que debe ser almacenada en la base de datos con el fin de contribuir al reconocimiento del objeto que describe.

Cada atributo $A_i$ toma valores es un dominio y, por tanto, diremos que cada atributo $A_i$ tiene un dominio asociado $D_i$, lo que representamos:

$dom(A_i) = D_i$

*Tupla*: una tupla es una lista ordenada de valores $< v_1, v_2, v_3, \dots , v_n >$ donde

$v_i \in dom (A_i)$, $i = 1, \dots , n$

o bien,

$v_i$ es un valor nulo.

*Valor de dominio*: es un valor concreto que toma, de entre los valores posibles del dominio que le corresponde, un atributo de una tupla.

## Relación

Una relación, R, se compone de una cabecera de relación y de un cuerpo de relación.

La cabecera de relación, que denotaremos como

$R(A_1, A_2, A_3, \ldots, A_n)$,

está compuesta por el nombre de la relación, R, y una lista de atributos $A_1,A_2,A_3,..., A_n$.

La cantidad de atributos, n, de la cabecera de la relación, se denomina *grado* de la relación.

El cuerpo de relación, que denotaremos como

r(R),

es un conjunto de m tuplas:

$r = \{ t_1, t_2, t_3, \ldots, t_m \}$.

La cantidad de tuplas, m, que posee el cuerpo de una relación se conoce como *cardinalidad* de la relación.

## Propiedades de las relaciones

Las relaciones tienen tres propiedades que definen, justamente, las características que las distinguen de un archivo tradicional:

- No existen tuplas repetidas.

  El cuerpo de una relación R, está definido como un conjunto de tuplas. Matemáticamente, los conjuntos no tienen elementos repetidos, por lo tanto, el cuerpo de una relación no contiene tuplas repetidas.

  Una consecuencia inmediata de esta propiedad, es la necesidad de que cada tupla posea un valor particular, o un conjunto de valores, que solamente aparezca en ella y que permita su individualización dentro del conjunto de tuplas.

- Las tuplas no están ordenadas.

  Esta propiedad también se deriva de la definición de conjunto. El conjunto de tuplas puede representarse en cualquier orden, lo que no altera los valores de ninguno de los datos persistentes.

  De hecho, un índice sobre alguno de los atributos implica una referencia a un ordenamiento de las tuplas de acuerdo a algún criterio pero, definitivamente, no implica la alteración de los valores que contienen ni ese atributo ni ninguno de los otros.

- Los atributos no están ordenados.

  Esta propiedad se desprende también de la definición de cabecera de una relación. Al haber sido definida como un conjunto, no interviene en ella la propiedad de orden.

## Las restricciones del Modelo Relacional

En el Modelo Relacional, las restricciones inherentes se clasifican de la siguiente manera:

1. Restricción de dominio.
2. Restricción de clave.
3. Restricciones de integridad.
    3.1. Integridad de las entidades.
    3.2. Integridad referencial.

Restricción de dominio

Enunciado de la restricción: el valor de cada atributo $A_i$ debe ser un único valor atómico del dominio dom($A_i$) de ese atributo.

O sea, en cada posición definida por la intersección de un atributo con una tupla, solo puede haber un valor de dominio y nunca un conjunto de valores.

Restricción de clave

Como hemos mencionado en la primera propiedad de las relaciones, no existen tuplas repetidas. Esto significa que no puede haber dos tuplas que tengan la misma combinación de valores para todos sus atributos.

Por lo tanto debe existir, al menos, un subconjunto de atributos para el cual no existan dos tuplas con la misma combinación de valores para esos atributos. Cada uno de estos subconjuntos de denominan *superclave* de la relación y, por lo tanto, debe poder asegurarse que toda relación debe tener, al menos, una superclave.

Es decir, una superclave asegura que cada tupla es única con respecto a esa superclave. Decimos en este caso que una superclave cumple con la propiedad de unicidad. Sin embargo, una superclave puede tener atributos redundantes, así que definiremos un concepto más útil: el de clave candidata.

Una *clave candidata*, K, de una relación R es una superclave que cumple con la propiedad de minimalidad. Esta propiedad de minimalidad es la que otorga a una clave candidata la posibilidad de mantener su propiedad de unicidad pero utilizando la menor cantidad posible de atributos.

Enunciado de la restricción: no puede haber relaciones sin clave primaria.

Como una relación puede tener más de una clave candidata, de entre ellas se selecciona una que será la *clave primaria* de la relación y es la que se utilizará para la identificación unívoca de cada tupla. La elección de la clave primaria entre las claves candidatas es arbitraria.

Restricciones de integridad

Integridad de las entidades

Enunciado del a restricción: ningún valor de clave primaria puede ser nulo.

Esta restricción no necesita explicación por su obviedad. Si una tupla tiene un valor nulo en su clave primaria no hay como identificarla unívocamente. Es más, si más de una tupla tuviese valor nulo en su clave primaria, no serían distinguibles entre ellas.

Integridad referencial

Las restricciones de dominio, de clave y de integridad de las entidades se consignan sobre relaciones individuales. En cambio, la restricción de integridad referencial se especifica entre dos relaciones y sirve para mantener la consistencia entre las tuplas de esas relaciones.

Enunciado de la restricción: cuando una tupla de una relación haga referencia a otra relación, en esta segunda deberá existir una tupla a la cual se refiere.

Para formalizar esta restricción, definiremos un nuevo concepto: clave ajena, o clave externa, o clave foránea. La definición de clave foránea implica por sí misma la definición de integridad referencial.

Un conjunto de atributos de una relación $R_1$ es una clave foránea de $R_1$ si:

- los atributos de la clave foránea tienen el mismo dominio que los atributos de la clave primaria de una relación $R_2$

- un valor de clave foránea en una tupla $t_1$ de $R_1$ ocurre como valor en alguna tupla $t_2$ de $R_2$, o bien es nulo. Se dice que la tupla $t_1$ hace referencia, o se refiere a la tupla $t_2$.

## La manipulación de los datos en el Modelo Relacional: el Álgebra Relacional y el Cálculo Relacional

Álgebra Relacional

En una base de datos hay información guardada en lugares físicos diferentes. Mediante una consulta, es posible acceder a esta información y a las vinculaciones entre ella.

Para la manipulación de los datos, Codd [11] diseñó dos niveles de lenguaje: el Álgebra Relacional y el Cálculo Relacional.

El Álgebra Relacional es un conjunto de operadores de alto nivel que actúan sobre las relaciones, en cambio, el Cálculo Relacional proporciona una sintaxis para formular las consultas sin necesidad de especificar en qué forma debe operarse para resolver dicha consulta. Es decir: en el Álgebra Relacional es necesario indicar “cómo”

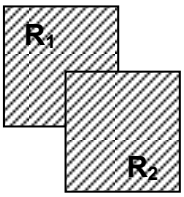
recuperar los datos a través del uso de los operadores, en cambio, en el Cálculo Relacional hay que indicar solamente "qué" datos queremos recuperar.
El Álgebra Relacional definida por Codd consta de ocho operaciones cerradas separadas en dos grupos:

- las operaciones tradicionales: unión, intersección, diferencia y producto.

  La unión de dos relaciones da como resultado una nueva relación que contiene todas las tuplas que pertenecen a una, a otra o a ambas relaciones:

  $R_1 \cup R_2 = \{ t_i \mid ( t_i \in R_1 ) \vee ( t_i \in R_2 ) \}$

  Gráficamente:

  Ejemplo:

  Si

  $R_1$

| Id | Nombre |
|---|---|
| 1 | Juan |
| 3 | Pedro |
| 7 | María |

  $R_2$

| Id | Nombre |
|---|---|
| 1 | Juan |
| 2 | Diego |
| 4 | Rosa |
| 5 | José |

  entonces

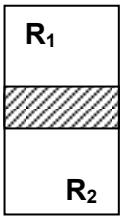
  $R_1 \cup R_2$

| Id | Nombre |
|---|---|
| 1 | Juan |
| 3 | Pedro |
| 7 | María |
| 2 | Diego |
| 4 | Rosa |
| 5 | José |

  La intersección de dos relaciones da como resultado una nueva relación que contiene las tuplas que pertenecen a ambas relaciones:

  $R_1 \cap R_2 = \{ t_i \mid ( t_i \in R_1 ) \wedge ( t_i \in R_2 ) \}$

Gráficamente:




Ejemplo: si $R_1$ y $R_2$ son las relaciones del ejemplo anterior, entonces:

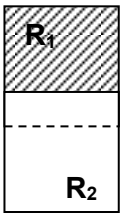
$R_1 \cap R_2$

| Id | Nombre |
|---|---|
| 1 | Juan |

La diferencia entre dos relaciones devuelve una nueva relación en la que aparecen todas las tuplas de la primera que no pertenecen a la segunda:

$R_1 - R_2 = \{ t_i \mid ( t_i \in R_1 ) \wedge ( t_i \notin R_2 ) \}$

Gráficamente:




Ejemplo: para las relaciones $R_1$ y $R_2$ dadas:

$R_1 - R_2$

| Id | Nombre |
|---|---|
| 3 | Pedro |
| 7 | María |

El producto entre dos relaciones devuelve una relación que contiene un conjunto de tuplas resultantes de la combinación de todas las tuplas de una con todas las tuplas de la otra relación.

Si $R_1(A_1, A_2, A_3, \ldots, A_m)$, $R_2(B_1, B_2, B_3, \ldots, B_n)$, $< a_1, a_2, a_3, \ldots, a_m >$ es una tupla de la relación $R_1$ y $< b_1, b_2, b_3, \ldots, b_n >$ es una tupla de la relación $R_2$ , la tupla resultante del producto entre ellas es:

$< a_1, a_2, a_3, \ldots, a_m, b_1, b_2, b_3, \ldots, b_n >$

de modo que, si $R_1$ tiene cardinalidad $c_1$ y $R_2$ tiene cardinalidad $c_2$:

$$R_1 \times R_2 = \{ < a_{1i}, a_{2i}, a_{3i}, \ldots, a_{mi}, b_{1j}, b_{2j}, b_{3j}, \ldots, b_{nj} > \mid < a_{1i}, a_{2i}, a_{3i}, \ldots, a_{mi} > \in R_1, < b_{1j}, b_{2j}, b_{3j}, \ldots, b_{nj} > \in R_2, i = 1, \ldots, c_1 , j = 1, \ldots, c_2 \}$$

Ejemplo:

$R_1$

| **$a_1$** | **$a_2$** |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |

$R_2$

| **$b_1$** |
|---|
| 10 |
| 11 |
| 12 |
| 13 |

entonces:

$R_1 \times R_2$

| **$a_1$** | **$a_2$** | **$b_1$** |
|---|---|---|
| 1 | 1 | 10 |
| 1 | 1 | 11 |
| 1 | 1 | 12 |
| 1 | 1 | 13 |
| 1 | 2 | 10 |
| 1 | 2 | 11 |
| 1 | 2 | 12 |
| 1 | 2 | 13 |
| 1 | 3 | 10 |
| 1 | 3 | 11 |
| 1 | 3 | 12 |
| 1 | 3 | 13 |

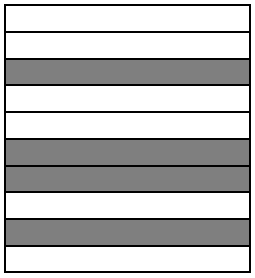
- las operaciones especiales: selección, proyección, reunión y división.

  La selección, $\sigma$, devuelve una relación que contiene todas las tuplas de una relación dada que cumplan con una condición impuesta.

  Si c es la condición que deben cumplir las tuplas seleccionadas, entonces

  $\sigma_c(R) = \{ t_i \mid ( t_i \in R ) \wedge ( \text{para } t_i \; c = \text{Verdadero} ) \}$

  Gráficamente:



Ejemplo:

Si

R

| Id | Nombre |
|---|---|
| 1 | Juan |
| 2 | Diego |
| 4 | Rosa |
| 5 | José |

entonces:

$\sigma_{Id<4}(R)$

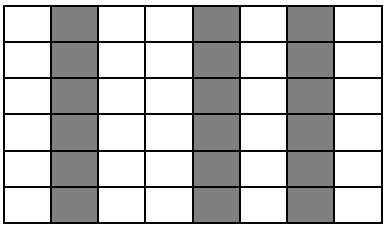
| Id | Nombre |
|---|---|
| 1 | Juan |
| 2 | Diego |

La proyección, $\Pi$, devuelve una relación que contiene algunos de los atributos de todas las tuplas de la relación dada.

Si $R(A_1, A_2, A_3, \ldots, A_m)$, entonces

$\Pi_{A1,\ A3}(R) = \{ t_i \mid ( t_i \in R ) \wedge t_i = < a_1, a_3 > \}$

Gráficamente:



Ejemplo:

Si

R

| Id | Nombre |
|---|---|
| 1 | Juan |
| 2 | Diego |
| 4 | Rosa |
| 5 | José |

entonces

$\Pi_{Nombre}(R)$

| Nombre |
|---|
| Juan |
| Diego |
| Rosa |
| José |

La reunión, $\bowtie$, de dos relaciones es una relación que contiene un subconjunto del producto entre las relaciones dadas, subconjunto formado solo por las combinaciones de tuplas en las que ambas satisfacen una condición dada.

Si c es la condición que deben cumplir tanto la tupla de la relación $R_1$ como la de $R_2$, entonces:

$R_1 \bowtie_c R_2 = \sigma_c(R_1 \times R_2)$

Un caso especial de la operación reunión, es aquél en el que la condición c de reunión es la igualdad de los valores entre un atributo de una tupla de $R_1$y un atributo de una tupla de $R_2$. Este caso recibe el nombre de reunión natural y se simboliza $R_1 \bowtie R_2$ .

Ejemplo:

$R_1$

| Id | Nombre | CPostal |
|---|---|---|
| 1 | Juan | 4400 |
| 2 | Pedro | 4402 |
| 3 | Diego | 4401 |
| 4 | María | 4401 |
| 5 | Rosa | 4400 |

$R_2$

| CPostal | Localidad |
|---|---|
| 4000 | Salta |
| 4001 | San Lorenzo |
| 4002 | Cerrillos |

entonces

$R_1 \bowtie R_2$

| Id | Nombre | CPostal | Localidad |
|---|---|---|---|
| 1 | Juan | 4400 | Salta |
| 2 | Pedro | 4402 | Cerrillos |
| 3 | Diego | 4401 | San Lorenzo |
| 4 | María | 4401 | San Lorenzo |
| 5 | Rosa | 4400 | Salta |

La división, ÷ , entre las relaciones $R_1$ y $R_2$ da como resultado una relación cuya cabecera es la cabecera de la relación $R_1$ y cuyo cuerpo son las tuplas de $R_1$ en las que aparecen alguna combinación de tuplas de $R_2$.

Ejemplo:

Si

$R_1$

| **A1** | **A2** |
|---|---|
| N1 | 1 |
| N1 | 2 |
| N1 | 3 |
| N1 | 4 |
| N1 | 5 |
| N1 | 6 |
| N2 | 1 |
| N2 | 2 |
| N3 | 2 |
| N4 | 2 |
| N4 | 4 |
| N4 | 5 |

$R_2$

| **A2** |
|---|
| 1 |

$R_3$

| **A2** |
|---|
| 2 |
| 4 |

$R_4$

| **A2** |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |

entonces:

$R_1 \div R_2$

| **A1** |
|---|
| N1 |
| N2 |

$R_1 \div R_3$

| **A1** |
|---|
| N1 |
| N4 |

$R_1 \div R_4$

| **A1** |
|---|
| N1 |

Cálculo Relacional

En su artículo “Relational Completeness of Data Base Sublanguajes”, Codd [08] propone un cálculo de predicados de primer orden orientado a las bases de datos relacionales.

Una de las principales características que posee este cálculo definido por Codd, es que utiliza un lenguaje *relacionalmente completo*, esto es, un lenguaje que posee la propiedad de que cualquier consulta definible por medio de expresiones del lenguaje se puede resolver utilizando solo las proposiciones del mismo lenguaje.

El cálculo de Codd se conoce como *cálculo de tuplas* porque se basa en el concepto de *variable de tupla*, que es una variable que toma los valores de una tupla de una relación, es decir, los únicos valores permitidos para una variable de tupla son las tuplas de una relación.

En contraposición al cálculo de tuplas, Lacroix y Pirotte [22] propusieron el *cálculo de dominios*, en donde la variable solo toma valores en los conjuntos que conforman los dominios subyacentes de los atributos de las relaciones.

Una expresión del cálculo relacional orientado a las tuplas tiene la siguiente forma:

$$\{ t \mid \psi(t) \}$$

donde:

t es una variable de tupla
$\psi(t)$ es una fórmula construida por átomos y operadores.

Un átomo tiene una de las siguientes formas:

- Pertenencia: R(t), donde R es una relación y t una variable tupla, este átomo expresa la afirmación de que la tupla t pertenece a la relación R

- Comparación: x $\theta$ y, donde $\theta$ es un comparador aritmético, x e y son constantes o valores particulares de los atributos de una variable tupla.

Ejemplo: sea R una relación que contiene los nombres de los alumnos de una determinada asignatura y la nota que obtuvieron en la misma. La consulta: "nombre de los alumnos cuya nota es al menos 7", se define de la siguiente manera:

$\{ t \mid \exists u (R(u) \wedge u.nota \geq 7 \}$

donde u.nota simboliza al atributo nota de la tupla u.

## El lenguaje SQL

SQL: Structured Query Language, es el lenguaje estándar para trabajar con bases de datos relacionales. Fue originalmente desarrollado por IBM Research y fue implementado por primera vez a gran escala en un prototipo de la mencionada firma.

Está basado en el lenguaje SEQUEL: Structured English Query Language, desarrollado por Chamberlain et al., con el objetivo de manejar el Modelo Relacional de Codd.

Los comandos básicos de SQL pueden clasificarse en dos grandes grupos:

- DDL: Data Definition Language, Lenguaje de Definición de datos.
  Las sentencias del DDL permiten la creación y modificación de las estructuras en las que se almacenan los datos.
  Ejemplos de estos comandos son:
  CREATE
  DROP
  ALTER

- DML: Data Manipulation Language, Lenguaje de Manipulación de datos.
  Las sentencias del DML permiten la consulta y la modificación de los datos persistentes.
  Ejemplos de estos comandos son:
  SELECT
  INSERT
  DELETE
  UPDATE

Revisaremos, brevemente, la acción de algunos de ellos:

Comando CREATE

Este comando se utiliza para crear objetos de la base de datos, sean estos tablas, vistas o la estructura de la base de datos en sí.

En el primer caso, el formato general del comando es:

```
CREATE TABLE      < nombre de una tabla >
[ ( < definición de columnas y/o restricciones > )  ]
[< otras cláusulas >]
```

Las restricciones pueden ser: NOT NULL, NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY o REFERENCES, CHECK < condición >.

Ejemplo:

```
CREATE TABLE      Pisos (
                  Número    Numérico(9)
                            NOT NULL
                            PRIMARY KEY
                  Dueño     Numérico(9)
                            NOT NULL
                            REFERENCES Personas(NumeroDocumento)
```

Comando SELECT

Es el comando de consulta más importante y tiene el siguiente formato:

```
SELECT      < lista de selección >
FROM        < lista de tablas >
[WHERE      < condición >]
[< otras cláusulas >]
```

La condición que acompaña a WHERE puede incluir atributos de una relación, constantes, comparadores atritméticos, funciones definidas para el lenguaje como: LIKE, IN, IS, BETWEEN, u operadores lógicos.

Las cláusulas más importantes son: ORDER BY, GROUP BY, GOUP BY HAVING. Cuando se trabaja con grupos, pueden utilizarse las denominadas funciones de grupo: COUNT, SUM, AVG, MAX, MIN, STDDEV, VARIANCE.

Ejemplo: Lista de empleados con su salario, ordenada por salario decreciente, que cumplan que el nombre del empleado tenga una letra X en el medio y su salario supere la media:

```
SELECT      Nombre, Salario
FROM        Empleados
WHERE       Nombre LIKE '%X%'
AND         Salario >   (SELECT AVG (Salario)
                        FROM Empleados)
```

Comando INSERT

Se utiliza para introducir nuevos datos en la base de datos y tiene el siguiente formato general:

INSERT INTO < nombre de una tabla >
[< lista de atributos >]
VALUES < lista de expresiones >

Comando DELETE

Se utiliza para borra tuplas de una base de datos y tiene el siguiente formato general:

DELETE < nombre de una tabla >
WHERE < condición >

Comando UPDATE

Se utiliza para actualizar valores almacenados en una base de datos y tiene el siguiente formato general:

UPDATE < nombre de una tabla >
SET < asignación >
[WHERE < condición >]

# CAPÍTULO 4

# APROXIMACIONES SIN UTILIZAR LÓGICA DIFUSA: LA EXTENSIÓN DEL MODELO RELACIONAL

En general, estos modelos aportan una estrategia para el tratamiento de la imprecisión en bases de datos convencionales.

El primer intento de representar datos imprecisos en una base de datos fue la introducción de valores nulos, NULL, por parte de Codd [09] en el año 1.979.

Cualquier comparación con el valor NULL origina un resultado que no ni Verdadero ni Falso sino, digamos, es un Quizá (Q). El motor SQL u Oracle lo devuelven como UNKNOWN.

Obviamente, las tablas de verdad de las operaciones NOT, AND y OR del Álgebra Relacional se modifican ante la presencia de NULL, apareciendo el mencionado resultado Q:

| NOT | |
|---|---|
| T | F |
| Q | Q |
| F | T |

| AND | T | Q | F |
|---|---|---|---|
| T | T | Q | F |
| Q | Q | Q | F |
| F | F | F | F |

| OR | T | Q | F |
|---|---|---|---|
| T | T | T | T |
| Q | T | Q | Q |
| F | T | Q | F |

Después se añadió una distinción: si el valor NULL representaba un valor ausente o desconocido, pero aplicable, se denominaba A_NULL pero si, en cambio, representaba un valor ausente por no aplicable, se denominaba I_NULL.

Por ejemplo, un A_NULL es un valor de un atributo NúmeroDePatente de una persona que tiene auto pero al cargar los datos no se cuenta con la información sobre la matrícula de dicho auto. En cambio, el valor del mismo atributo sería I_NULL si la persona no tuviese auto.

Esta distinción llevó consigo la necesidad de transformar la lógica tri-valuada mostrada en las tablas C anteriores y definir una lógica tetra-valuada de la siguiente forma:

| NOT | |
|---|---|
| T | F |
| A | A |
| I | I |
| F | T |

| AND | T | A | I | F |
|---|---|---|---|---|
| T | T | A | I | F |
| A | A | A | I | F |
| I | I | I | I | F |
| F | F | F | F | F |

| OR | T | A | I | F |
|---|---|---|---|---|
| T | T | T | T | T |
| A | T | A | A | A |
| I | T | A | I | F |
| F | T | A | F | F |

En el año 1.980, Grant [21] propone otra extensión del Modelo Relacional en la que, además del valor preciso o grip y del valor NULL, puede almacenarse un intervalo de valores posibles para un atributo.

Para solucionar el tema de las consultas, éstas arrojaban en la resolución tres categorías de tuplas: las que seguro pertenecen al resultado, las que posiblemente pertenecen al resultado y las que seguro no pertenecen al resultado.

Por la misma época que Grant, Wong [36] propone el manejo de los casos de incertidumbre por inferencia estadística. Este método ve las consultas como experimentos estadísticos en los que la información es incompleta y se centra en calcular la respuesta como el conjunto de aquellas tuplas que minimizan los errores estadísticos.

## La extensión del Modelo Relacional de Motro

Dentro de esta línea de tratamiento de la información difusa sin utilizar Lógica Difusa, Amihai Motro [28], en el año 1.987, introduce el concepto de *distancia* entre valores de un mismo atributo y propone una extensión del Modelo Relacional para representar esa información.

Las consultas se clasifican en dos tipos:

- específicas

y

- de objetivos

Las específicas se satisfacen con datos que pertenecen a una calificación rígida y poseen valores definidos o exactos. Por ejemplo, una consulta específica es: "¿Cuál es el sueldo de Juan?", o, "¿Cuál es el horario de salida del ómnibus a Córdoba?". Si en la base de datos no se encuentra almacenado el sueldo de Juan o el horario de salida de los ómnibus, ninguna de las consultas podrá ser resuelta.

Las de objetivos, en cambio, pueden ser satisfechas por valores exactos pero también por valores que se encuentren cercanos a los exactos correspondientes. Por ejemplo: "Lista de parrilladas en el área céntrica". La resolución de la consulta podría incluir una serie de parrilladas que no se encuentren exactamente en el área céntrica pero sí en un área cercana al centro. También podrían satisfacer la consulta una lista de restaurantes que, sin ser específicamente parrillas, sirvan menúes similares a ellas y se encuentren en el área céntrica o cerca de ella.

Para satisfacer una consulta de objetivos debemos, en primera medida, saber qué interpretaremos como un valor cercano, esto es, cuáles serán las tuplas que se agregarán como resultado de la consulta que tienen una alta posibilidad de satisfacer la misma. Para ello, se introduce el concepto de *distancia*.

La definición de distancia contiene información importante acerca de la semántica de los atributos, información que permite la manipulación de los datos. Esta distancia se incorpora al a base de datos de la misma forma que se incorpora la información sobre el ordenamiento y el dominio de cada atributo.

## Distancia: conceptos

Toda relación de una base de datos posee un atributo o conjunto de atributos denominado clave primaria. Esta clave provee una identificación única para cada tupla de la relación.

Todo atributo no clave, por otra parte, provee una descripción del atributo clave.

Por ejemplo, consideremos la siguiente relación:

Emisora FM (Nombre, Tipo, Ubicación, Calidad de Sonido, Popularidad, Teléfono)

cuya PK – clave primaria – es el atributo Nombre y sea la tupla:

(Balcarce, Folclore, Zona Oeste, Media, Muy Mala, 4363636)

Por ser Nombre la clave primaria, el resto de los atributos son solo calificativos de ese atributo clave, por lo tanto, si queremos saber la distancia entre esa tupla y cualquier otra, debemos preguntarnos cuál es la distancia entre los correspondientes valores del atributo Nombre. Obviamente, esta distancia estará influenciada por los valores de los atributos no clave.

Supongamos otra tupla:

(De La Mujer, Melódico, Zona Norte, Muy Baja, Buena, 4252525)

La distancia entre las emisoras Balcarce y De La Mujer, será la combinación de las distancias entre Folclore y Melódico, Zona Oeste y Zona Norte, Media y Muy Baja, Muy Mala y Buena y 4363636 y 4252525.

Pero, ¿cómo calcularíamos la distancia entre Folclore y Melódico?, ¿y las demás?. Tendremos necesidad de definir nuevas relaciones que nos permitan acceder a estos cálculos. Además, cualquier relación que se defina para poder efectuar estos cálculos deberá contener todos los valores de los atributos que aparezcan en cualquiera de las otras relaciones, porque debe cumplirse la restricción de integridad referencial.

Pero no es necesario definir una nueva relación para el cálculo de las distancias entre los valores de todos los atributos. Por ejemplo, si el dominio del atributo es numérico, la distancia entre dos valores se puede calcular con la función matemática. De igual manera, si el atributo tuviese su dominio en un Alfanumérico, la distancia entre dos valores será 1 si se trata de la misma cadena y 0 si son dos diferentes.

Las funciones, estándares o definidas, para el cálculo de distancia entre pares de valores correspondientes, se denomina *Medida* de un atributo. Adicionalmente, se asocian a cada atributo tres parámetros más: *Factor de Escala*, *Peso Relativo* y *Radio de Vecindario.*

Formalmente:

Medida: es el nombre de la expresión con que será calculada la distancia entre dos valores de atributos correspondientes. Puede ser el nombre de una función estándar o el nombre de una relación definida para efectuar el cálculo.

Factor de Escala (s): es un número que se usa para corregir el valor de la distancia entre valores de un mismo atributo luego de que este valor es obtenido a partir de la correspondiente Medida.

Peso Relativo (w): es un número que asigna a cada atributo el peso que tendrá en el cálculo de la distancia entre dos tuplas.

Radio de Vecindario (r): es un valor que establece cuánto es el máximo que pueden separarse dos valores de un atributo para que no dejen de considerarse cercanos.

## Distancia: definición

Sean x e y dos valores de un mismo atributo A que tiene Medida M. La distancia entre x e y, medida según M, $d_M(x,y)$, se define como:

- Si M es numérico:

  $d_M(x,y) = d_{Numérico}(x,y) = | x - y|$

- Si M es alfanumérico:

  $$d_M(x,y) = d_{Alfanumérico}(x,y) = \begin{cases} 0 \text{ si } x = y \\ 1 \text{ si } x \neq y \end{cases}$$

- Si M es una relación:

  $$d_M(x,y) = \sum_{i=1}^{n} d_{M_i}(x_i, y_i) \left(\frac{1}{s_i}\right) \left(\frac{w_i}{w}\right)$$

  donde:

  M es una relación con atributos $A_0, A_1, A_2, \dots, A_n$
  $A_0$ es la clave primaria de M
  $M_i$ es la Medida de $A_i$
  $s_i$ es el Factor de Escala de $A_i$
  $w_i$ es el Peso Relativo de $A_i$
  $w = w_1 + w_2 + \dots + w_n$
  $x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n$ son valores de $A_1, A_2, \dots, A_n$, respectivamente.

## Un ejemplo de aplicación

Consideremos el siguiente esquema de una base de datos, esquema que llamaremos Emisoras FM, en donde se guarda información sobre emisoras radiales de frecuencia FM:

| Relación | Atributo | PK | Medida | s | w | r |
|---|---|---|---|---|---|---|
| Emisora FM | Nombre | * | Emisora FM | 1 | - | 1 |
| | Tipo | | Ofrece | 1 | 2 | 1 |
| | Ubicación | | Vecino | 10 | 1 | 1 |
| | Calidad de Sonido | | Calidad | 1 | 1 | 1 |
| | Popularidad | | Rating | 1 | 2 | 1 |
| | Teléfono | | Alfanumérico | 1 | 0 | 0 |

| Relación | Atributo | PK | Medida | s | w | r |
|---|---|---|---|---|---|---|
| Ofrece | Nombre | * | Ofrece | 1 | - | 2 |
| | Categoría | | Alfanumérico | 1 | 2 | 1 |
| | Aceptación | | Numérico | 500 | 1 | 0.5 |

| Relación | Atributo | PK | Medida | s | w | r |
|---|---|---|---|---|---|---|
| Vecino | Nombre (A,B) | * | Vecino | 1 | - | 1 |
| | Kilómetros | | Numérico | 1 | 1 | 5 |

| Relación | Atributo | PK | Medida | s | w | r |
|---|---|---|---|---|---|---|
| Calidad | Descripción | * | Calidad | 1 | - | 1 |
| | Valoración | | Numérico | 1 | 1 | 1 |

| Relación | Atributo | PK | Medida | s | w | r |
|---|---|---|---|---|---|---|
| Rating | Descripción | * | Rating | 1 | - | 1 |
| | Ranking | | Numérico | 1 | 1 | 1 |

En este esquema podemos observar:

- las distancias entre valores del atributo Teléfono, se calcularán a partir dela función estándar para alfanuméricos
- las relaciones Ofrece, Vecino, Calidad y Rating se han definido para poder efectuar el cálculo de las distancias entre valores de los atributos Tipo, Ubicación, Calidad de Sonido y Cantidad de Oyentes, respectivamente

- las distancias entre los valores de los atributos Tipo y Cantidad de Oyentes, tienen más peso (w = 2) que las demás (w = 1) en el cálculo de la distancia entre tuplas

- el radio de vecindario definido es de 5 Km. máximo, como lo indica r = 5 en la relación Vecino, esto es, si la distancia entre dos ubicaciones fuere mayor que 5 Km, las ubicaciones no pueden considerarse vecinas

Supongamos ahora que tenemos los siguientes valores:

Relación: **Emisora FM**

| **Nombre** | **Tipo** | **Ubicación** | **Calidad de Sonido** | **Popularidad** | **Teléfono** |
|---|---|---|---|---|---|
| Balcarce | Folclore | Zona Oeste | Media | Muy Mala | 4363636 |
| Nativa | Tango | Centro | Baja | Mala | 4313131 |
| De La Mujer | Melódico | Zona Norte | Muy Baja | Buena | 4252525 |
| Todo Deporte | Deportes | Centro | Baja | Mala | 4310000 |
| La Mañana | Melódico | Zona Sur | Media | Muy Buena | 4242424 |
| FM 840 | Metálica | Zona Oeste | Muy Alta | Excelente | 4360000 |
| La Compañía | Rock | Zona Sur | Muy Baja | Buena | 4240000 |
| TFM | Noticias | Zona Este | Baja | Mala | 4210000 |
| Solo Fútbol | Fútbol | Zona Este | Baja | Muy Buena | 4211010 |
| Mi Ciudad | Soul | Centro | Media | Excelente | 4311111 |

Relación: **Ofrece**

| **Nombre** | **Categoría** | **Aceptación** |
|---|---|---|
| Melódico | Música | 800 |
| Noticias | Noticias | 1800 |
| Folclore | Música | 2900 |
| Soul | Música | 950 |
| Metálica | Música | 1500 |
| Fútbol | Deportes | 1250 |
| Rock | Música | 1800 |
| Deportes | Deportes | 2000 |
| Tango | Música | 2500 |

Relación: **Vecino**

| **Nombre A** | **Nombre B** | **Kilómetros** |
|---|---|---|
| Zona Oeste | Zona Norte | 24 |
| Zona Oeste | Centro | 20 |
| Centro | Zona Este | 12 |
| Zona Oeste | Zona Oeste | 0 |
| Zona Norte | Centro | 4 |
| Zona Norte | Zona Este | 14 |
| Zona Oeste | Zona Sur | 8 |
| Zona Norte | Zona Norte | 0 |
| Centro | Zona Sur | 10 |

Relación: **Calidad**

| Descripción | Valoración |
|---|---|
| Muy Alta | 5 |
| Alta | 4 |
| Media | 3 |
| Baja | 2 |
| Muy Baja | 1 |

Relación: **Rating**

| Descripción | Ranking |
|---|---|
| Excepcional | 5 |
| Excelente | 4 |
| Muy Buena | 3 |
| Buena | 2 |
| Mala | 1 |
| Muy Mala | 0 |

¿Cuál será la distancia entre la emisora FM Balcarce y la emisora FM Nativa?, o sea,

$d_{Emisora}$ FM (Balcarce, Nativa) = ?

Para calcular esta distancia, debemos calcular las distancias parciales entre los valores de los atributos y, finalmente calcular la distancia total, la que está dada por la fórmula de distancia para una relación, en nuestro caso, la relación Emisora FM.

Paso 1: calcular la distancia entre Folclore y Tango.

La Medida del atributo Tipo es Ofrece, esto es, debemos calcular entonces:

$d_{Ofrece}$(Folclore, Tango)

A su vez, la relación Ofrece tiene dos atributos que participarán en el cálculo de la distancia: Categoría y Aceptación.

La Medida de Categoría es Alfanumérico y:

valor de Categoría para Folclore: Música
Valor de Categoría para Tango: Música

entonces, por las definiciones de distancias precedentes:

$d_{Alfanumérico}$ (Música, Música) = 0

La Medida de Aceptación es Numérico y:

valor de Aceptación para Folclore = 2900
valor de Aceptación pata Tango = 2500

de modo que:

$d_{Numérico}$ (2900, 2500) = 400

Entonces tenemos

| $d_{M_i}(x_i, y_i)$ | $s_i$ | $w_i$ |
|---|---|---|
| 0 | 1 | 2 |
| 400 | 500 | 1 |

con: w = w1 + w2 = 2 + 1 = 3.

Llevando estos valores a la expresión respectiva, obtenemos:

$d_{Ofrece}(x,y) = \sum_{i=1}^{n} d_{M_i}(x_i, y_i) \left(\frac{1}{s_i}\right) \left(\frac{w_i}{w}\right) = \sum_{i=1}^{2} d_{M_i}(x_i, y_i) \left(\frac{1}{s_i}\right) \left(\frac{w_i}{w}\right) =$

$= (0) \left(\frac{1}{1}\right) \left(\frac{2}{3}\right) + (400) \left(\frac{1}{500}\right) \left(\frac{1}{3}\right) = 0{,}27$

Paso 2: calcular la distancia entre Zona Oeste y Centro.

La Medida del atributo Ubicación es Vecino, que tiene un único atributo, Kilómetros, con un $s_i$ = 1 y un $w_i$ = 1por lo tanto:

$d_{Vecino}$(Zona Oeste, Centro) = $(20) \left(\frac{1}{1}\right) \left(\frac{1}{1}\right) = 20$

Paso 3: calcular la distancia entre Media y Baja.

La Medida del atributo Calidad de Sonido es Calidad, que es de tipo Numérico, entonces:

$d_{Numérico}$(3, 2) = 1

además: $s_i$ = 1, $w_i$ = 1, de modo que:

$d_{Calidad}$(Media, Baja) = $(1) \left(\frac{1}{1}\right) \left(\frac{1}{1}\right) = 1$

Paso 4: calcular la distancia entre Muy Mala y Mala.

La Medida del atributo Popularidad es Rating, que es de tipo Numérico, entonces:

$d_{Numérico}$(0, 1) = 1

además: $s_i$ = 1, $w_i$ = 1, de modo que:

$d_{Rating}$(Muy Mala, Mala) = $(1) \left(\frac{1}{1}\right) \left(\frac{1}{1}\right) = 1$

Paso 5: calcular la distancia entre 4363636 y 4313131

El atributo Teléfono es Alfanumérico y los valores correspondientes para cada tupla son distintos así que, por la definición de distancia:

$d_{Alfanumérico}(4363636, 4313131) = 1$

Ahora que tenemos las distancias de entre cada uno de los atributos de la relación Emisora FM, podemos calcular la distancia entre las dos tuplas.

$w = w_1 + w_2 + w_3 + w_4 + w_5 = 2 + 1 + 1 + 2 + 0 = 6$

$d_{Emisora\ FM}$ (Balcarce, Nativa) = $\sum_{i=1}^{n} d_{M_i}(x_i, y_i) \left(\frac{1}{s_i}\right) \left(\frac{w_i}{w}\right) =$

$= \sum_{i=1}^{5} d_{M_i}(x_i, y_i) \left(\frac{1}{s_i}\right) \left(\frac{w_i}{w}\right) = (0{,}27) \left(\frac{1}{1}\right) \left(\frac{2}{6}\right) + (20) \left(\frac{1}{10}\right) \left(\frac{1}{6}\right) +$

$+ (1) \left(\frac{1}{1}\right) \left(\frac{1}{6}\right) + (1) \left(\frac{1}{1}\right) \left(\frac{2}{6}\right) + (1) \left(\frac{1}{1}\right) \left(\frac{0}{6}\right) = 0{,}92$

## ¿Cuál es el conjunto de tuplas que satisface la consulta?

Hemos encontrado que la distancia entre la emisora Balcarce y la emisora Nativa es de 0,92 y ahora la pregunta es: dado este valor de la distancia, ¿debemos incluir la emisora Balcarce cuando consultemos sobre emisoras cuyo tipo de música sea tango, o a la emisora Nativa cuando consultemos emisoras cuyo tipo de música sea folclore?.

Definición: comparador aritmético “similar a”

Sean x e y dos valores a los cuales se les puede aplicar una medida M y sean s y r dos números reales. El comparador aritmético “similar a”, denotado por $\sim$, se define como:

$x \sim y$ si $d_M(x, y) \left(\frac{1}{s}\right) \leq r$

Consideremos la siguiente consulta:

Q = { t | (∃ e) (∃ o) (r ∈ Emisora FM) ∧ (o ∈ Ofrece) ∧ (t = e.Ubicación)
∧ (e.Calidad de Sonido = Muy Baja) ∧ (e.Tipo = o.Nombre) ∧
(o.Aceptación = 2000) }

Las únicas emisoras con Calidad de Sonido = Muy Baja son De La Mujer y La Compañía y estas emisoras tienen Aceptación = 800 y Aceptación = 1800, respectivamente, con lo cual obtendríamos:

Q = ∅

Pero supongamos ahora que

e.Calidad de Sonido = Muy Baja

se sustituye por

e.Calidad de Sonido ~ Muy Baja

El atributo Calidad de Sonido tiene:

Medida = Calidad
s = 1
w = 1
r = 1

de modo que

e.Calidad de Sonido ~ Muy Baja

será satisfecha por el conjunto de tuplas donde

$d_{Calidad}$ (e.Calidad de Sonido) ≤ 1

que es el conjunto de tuplas para las cuales el valor de Calidad de Sonido sea o Muy Baja o Baja. Por lo tanto, el resultado de la consulta incluirá la tupla:

(Todo Deporte, Deporte, Centro, Baja, Mala, 4310000)

¿Qué tuplas incluiría el resultado de la consulta si en Q se hubiese puesto o.Aceptación ~ 2000 en lugar de o.Aceptación = 2000?.

El atributo Aceptación de la relación Ofrece tiene:

Medida = Numérico
s = 500
w 0 1
r = 0,5

de modo que se deberá cumplir:

$$d_M(x, y)\left(\frac{1}{s}\right) \leq r$$

de donde

$$d_M(x, y) \leq (s)(r) \leq (500)\ (0{,}5) \leq 250$$

O sea:

1750 ≤ Ofrece.Aceptación ≤ 2250

de modo que se incluirían en el resultado de Q:

(La Compañía, Rock, Zona Sur, Muy Baja, Buena, 4240000)

Notemos que si transformamos Q en:

Q = { t | (∃ e) (∃ o) (r ∈ Emisora FM) ∧ (o ∈ Ofrece) ∧ (t = e.Ubicación) ∧ (e.Calidad de Sonido ~ Muy Baja) ∧ (e.Tipo = o.Nombre) ∧ (o.Aceptación ~ 2000) }

las tuplas que aparecerán en el resultado son:

(Todo Deporte, Deporte, Centro, Baja, Mala, 4310000)
(La Compañía, Rock, Zona Sur, Muy Baja, Buena, 4240000)
(TFM, Noticias, Zona Este, Baja, Mala, 4210000).

# CAPÍTULO 5

# MODELOS DE BASES DE DATOS RELACIONALES DIFUSAS

La propuesta inicial para el modelo de una Base de Datos Relacional Difusa fue el agregado de un atributo a cada relación. Este atributo contenía un valor dentro del intervalo [0,1] y se denominaba *grado* de la tupla. El valor del atributo indicaba el grado de pertenencia de la tupla a la relación que se formaba a consecuencia de la resolución de la consulta.

Si bien esta propuesta permitía incorporar tuplas "similares" a las buscadas al resultado de una consulta, no permitía la representación de la información imprecisa, como los valores "joven" o "viejo" para el atributo Edad. Además, el valor contenido en el atributo grado, semánticamente, otorgaba un valor global a la tupla y por lo tanto era ésta la que poseía el carácter de difusa y no uno o más atributos en particular.

Surgen entonces los Modelos de Bases de Datos Relacionales Difusas que utilizan relaciones de similitud. Típicos exponentes de estos modelos son los de Buckles-Petry [04, 05, 06], Prade-Testemale [32, 33, 34], Umano-Fukami [35] y Zemankova-Kandel [40, 41]. Todos estos modelos proponen formas de definiciones de los conjuntos dominio y/o cálculos en base a la Teoría de la Posibilidad para asignar valores al atributo grado.

En el año 1.994, Medina, Pons y Vila [26] presentan en Modelo GEFRED, que presenta una innovación que permite un gran avance en el tema: define lo que se llama *Dominio Difuso Generalizado* (D) y *Relación Difusa Generalizada* (R).

## Teoría de Conjunto Difusos

La Teoría de Conjuntos Difusos fue introducida por L.A. Zadeh [37] en el año 1.965, ante la necesidad de ampliar la definición clásica de conjunto para ampliarlo a la descripción de nociones vagas o imprecisas.

Esta generalización se realiza como sigue:

1. la pertenencia de un elemento a un conjunto pasa a ser un concepto difuso o borroso: para algunos elementos puede no estar clara su pertenencia o no al conjunto

2. la pertenencia puede ser cuantificada por un grado, que se denomina habitualmente "grado de pertenencia" del elemento al conjunto y toma un valor en el intervalo [0,1].

Conjunto Difuso: definición

Un Conjunto Difuso, $\widetilde{A}$, sobre un universo de discurso, $\Omega$, es un conjunto de pares

$\widetilde{A} = \{ \mu_A(x)/x : x \in \Omega, \mu_A(x) \in [0,1] \}$

donde:

$\mu_A(x)$ se denomina grado de pertenencia del elemento x al conjunto difuso $\widetilde{A}$ y $\mu_A$ se denomina función de pertenencia.

Por ejemplo: consideremos que la "edad", en años enteros, es el universo de discurso, entonces, el conjunto difuso "joven" podría representarse como:

joven = {1/20,1/25,0.9/26,0.8/27,0.7/28,0.6/29,0.5/30,0.4/31,0.3/32,0.2/33,0.1/34}

¿Qué debemos interpretar ante este conjunto?. Diremos que el valor "20" tiene una pertenencia de grado = 1 al conjunto "joven", o que el valor "31" también pertenece al conjunto pero con un grado menor, con grado = 0.4.

El identificador "joven" lleva en sí toda la semántica asociada a un conjunto difuso y recibe el nombre de etiqueta lingüística.

Etiqueta lingüística: definición

Llamaremos etiqueta lingüística a aquella palabra en lenguaje natural que exprese un conjunto difuso.

Distribución de Posibilidad: definición

Como vimos en las definiciones y en los ejemplos anteriores, los elementos pertenecen a un conjunto difuso en mayor o en menor grado.

Pero supongamos ahora que tenemos un conjunto difuso, $\widetilde{A}$ , y queremos saber si un determinado elemento, X, pertenece o no a ese conjunto. Esto es, deseamos saber cuál es la *posibilidad* de que un determinado elemento, X, pertenezca al conjunto difuso $\widetilde{A}$.

El valor de verdad de la proposición:

$X \in \widetilde{A}$

viene dado por una función que se denomina Distribución de Posibilidad.

Formalmente:

Sea $\widetilde{A}$ un conjunto difuso definido sobre el universo de discurso $\Omega$, con función de pertenencia $\mu_A(x)$ y una variable X definida también sobre $\Omega$. Entonces, la proposición

$X \in \widetilde{A}$

define una Distribución de Posibilidad de forma que se dice que la posibilidad de que

X = u

vale

$\mu_A(u), \forall\ u \in \Omega$

Dominio Difuso Generalizado: definición

Si U es el universo o dominio de discurso, $\tilde{P}(U)$ el conjunto de todas las distribuciones de posibilidad definidas sobre U, incluidas las que definen los tipos Unknown y Undefined, se define Dominio Difuso Generalizado, D, como:

$D \subseteq \tilde{P}(U) \cup$ Null

Con esta definición, se pueden representar los siguientes tipos de datos:

1 Un escalar simple
Tamaño = Grande, se representa mediante la distribución de posibilidad 1/Grande

2 Un número simple
Edad = 28, se representa mediante la distribución de posibilidad 1/28

3 Un conjunto de posibles asignaciones excluyentes de escalares
Aptitud = {Mala, Buena}, se expresa {1/Mala, 1/Buena}

4 Un conjunto de posibles asignaciones excluyentes de números
Edad = {20, 21}, se representa {1/20, 1/21}

5 Una distribución de posibilidad en el dominio de los escalares
Aptitud = {0.6/Mala, 1.0/Regular}

6 Una distribución de posibilidad en el dominio de los números
Edad = {0.4/23, 1.0/24, 0.8/25)

7 Un número real en el intervalo [0,1], que representa grado de cumplimiento
Calidad = 0.9

8 Un valor desconocido
Unknown = {1/u, u $\in$ U}

9 Un valor idefinido
Undefined = {0/u, u $\in$ U}

10 Un valor nulo
Null = {1/Unknown, 1/Undefined}

Relación Difusa Generalizada: definición

Una Relación Difusa Generalizada es un par ordenado cabecera (H) y cuerpo (B),

$R = < H , B >$

La cabecera es un conjunto de ternas atributo-dominio-compatibilidad

$H = \{ (A_1 : D_1 [,C_1]), (A_2 : D_2 [,C_2]), ……., (A_n : D_n [,C_n]) \}$

donde cada $D_j$ es un Dominio Difuso Generalizado y cada $C_j \in [0,1]$.

El cuerpo es un conjunto de tuplas difusas generalizadas distintas, cada una de ellas compuesta por un conjunto de ternas atributo-valor-grado

$B = \{ (A_1 : \tilde{d}_{i1} [,c_{i1}]), (A_2 : \tilde{d}_{i2} [,c_{i2}]), ……., (A_n : \tilde{d}_{in} [,c_{in}]) \}$

con i = 1, 2, …, m.

El grado de complejidad será utilizado cuando se efectúe alguna operación, por ejemplo, una consulta y servirá para almacenar el grado con el que el valor de un atributo concreto de una tupla concreta ha satisfecho dicha operación..

## Álgebra Relacional Difuso Generalizado y Cálculo Relacional Difuso

Sobre las definiciones anteriores, GEFRED redefine las operaciones del Álgebra Relacional: Unión, Intersección Diferencia, Producto, Proyección, Selección, Reunión y División. La redefinición de estas operaciones conforman el Álgebra Relacional Difusa.

Como sabemos, el Cálculo Relacional usa el cálculo de predicados de primero orden y utiliza un lenguaje relacionalmente completo, esto es, un lenguaje que posee la propiedad de que cualquier consulta definible por medio de expresiones del Cálculo Relacional se puede recuperar mediante proposiciones pertinentes de ese lenguaje.

El Cálculo Relacional tiene dos orientaciones:

- Cálculo Relacional orientado a las tuplas: cuando se basa en el concepto de variable de tupla, que es una variable que toma los valores de las tuplas de alguna relación o de la unión de dos o más relaciones.
  Una implementación del cálculo orientado a las tuplas fue QUEL, usado en el sistema INGRES. Actualmente, SQL aplica el cálculo con este tipo de orientación.

- Cálculo Relacional orientado a los dominios: las variables, llamadas variables de dominio, toman valores en los dominios subyacentes de los atributos de las relaciones.
  Este tipo de cálculo fue propuesto por Lacroix [22] en 1.977. ILL, FQL y QBE son lenguajes que permiten su aplicación.

El Modelo GEFRED redefine también el Cálculo Relacional, optando por trabajar sobre el orientado al dominio, creando de esta manera el Cálculo Relacional Difuso orientado al Dominio o, simplemente, Cálculo Difuso de Dominios. ¿Por qué orientado al dominio y no a las tuplas?. Porque el cálculo relacional orientado al dominio tiende a estar más cerca del lenguaje natural que el orientado a las tuplas.

En el Cálculo Difuso de Dominios, las expresiones son de la forma:

$\{x_1, x_2, \ldots, x_n \mid \psi(x_1, x_2, \ldots, x_n)\}$

donde:

$x_1, x_2, \ldots, x_n$ son variables de dominio, o sea, variables cuyos valores pertenecen a un dominio difuso generalizado predefinido.

$\psi(x_1, x_2, \ldots, x_n)$ es una Fórmula Bien Formada, o WFF, que se construye a partir de elementos que se llaman átomos difusos y de operadores específicos. Las únicas variables de estas fórmulas son $x_1, x_2, \ldots, x_n$ y expresa un predicado, $\psi$, que debe ser cumplido por las tuplas resultantes y que solo tiene valor de Verdadero o Falso.

Un átomo difuso está formado por dos partes:

1) grado de cumplimiento $\Delta$

2) umbral de cumplimiento $\gamma$

y debe cumplir la condición:

$\Delta \geq \gamma$

Obviamente, las operaciones que exigen el cálculo relacional también se han extendido, incorporándose las necesarias para operar en el Cálculo Relacional Difuso.

Así, se han definido los comparadores difusos generalizados. Estos comparadores pueden comparar valores de un atributo con constantes o valores de dos atributos definidos en el mismo dominio.

En adelante, denotaremos con $\Theta$ a cualquier comparador difuso generalizado y le pondremos, a modo de exponente, cuál es la comparación que efectúa. Así, por ejemplo:

$\Theta^{>}$

simboliza al comparador difuso generalizado “mayor que”.

Veamos el siguiente ejemplo: sean

- R y S son relaciones difusas generalizadas de grado 4 y 3, respectivamente
- $x_1$, $x_2$ y $x_3$, son variables de dominio
- “Bueno” es una etiqueta lingüística con una distribución de posibilidad asociada

- #n es un número difuso que expresa "aproximadamente n", siendo n una constante numérica

- $\Theta^{>}$ es el comparador difuso generalizado "mayor que"

- $\Theta^{>>}$ es el comparador difuso generalizado "mucho mayor que"

entonces, los siguientes son ejemplos de átomos difusos:

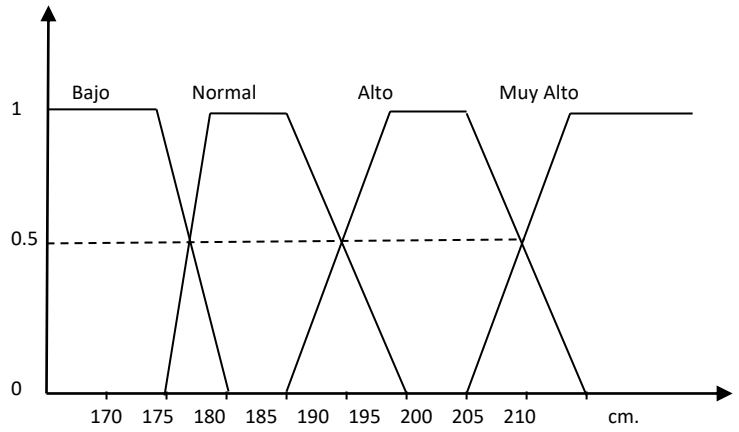
| Átomo | Significado |
|---|---|
| $R(x_1, x_2, 13, 8) \geq 0.6$ | La tupla ($x_1$, $x_2$, 13, 8) pertenece a R con un grado mínimo de 0.6 |
| $R(x_1, x_2, x_3, \#8) \geq 0.5$ | La tupla ($x_1$, $x_2$, $x_3$, #8) pertenece a R con grado mínimo 0.5 |
| $S(x_1, Bueno, x_3)$ | La tupla ($x_1$, Bueno, $x_3$) pertenece a S con grado no nulo |
| $\Theta^{=}(x_1, Bueno) \geq 0.9$ | $x_1$ es "Bueno" con grado mínimo 0.9 |
| $\Theta^{>}(x_1, \#15) \geq 0.25$ | $x_1$ es mayor que "aproximadamente 15" con un grado mínimo de 0.25 |

## Un ejemplo de aplicación

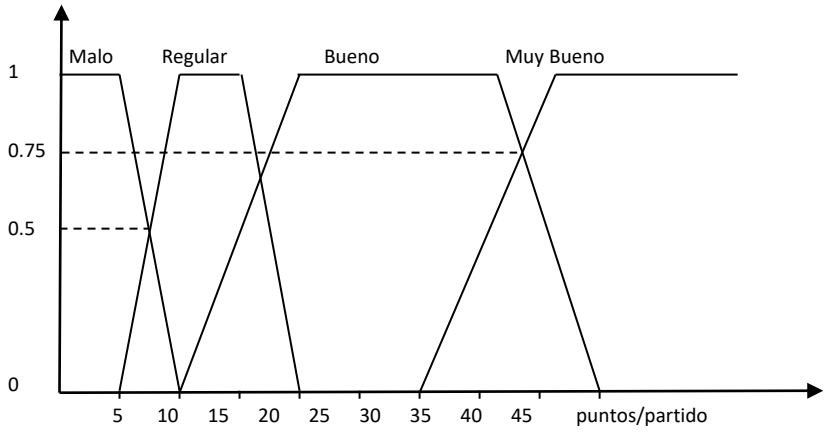
Supongamos una Base de Datos Relacional Difusa sobre jugadores de basquet y supongamos también una proyección de una relación, R, de esa base de datos, que sigue tiene como datos:

| Cabecera | Jugador | Equipo | Altura | Calidad |
|---|---|---|---|---|
| Cuerpo | J1 | Argentina | Alto | Muy Bueno |
| | J2 | Argentina | Bajo | Regular |
| | J3 | Canadá | Muy Alto | Muy Bueno |
| | J4 | Canadá | Bajo | Bueno |
| | J5 | Colombia | Bajo | Muy Bueno |
| | J6 | Colombia | Muy Alto | Malo |
| | J7 | Guatemala | Bajo | Malo |
| | J8 | Guatemala | Muy Alto | Malo |
| | J9 | Guatemala | Alto | Regular |
| | J10 | Holanda | Alto | Muy Bueno |
| | J11 | Jamaica | Bajo | Muy Bueno |
| | J12 | Jamaica | Normal | Regular |
| | J13 | Jamaica | Alto | Muy Bueno |
| | J14 | Jamaica | Alto | Muy Bueno |
| | J15 | Malasia | Bajo | Muy Bueno |
| | J16 | Malasia | Alto | Regular |
| | J17 | Malasia | Muy Alto | Muy Bueno |
| | J18 | Suiza | Bajo | Bueno |
| | J19 | Suiza | Muy Alto | Bueno |
| | J20 | Suiza | Normal | Bueno |

Los atributos Altura y Calidad permiten almacenar valores difusos. Para definirlos utilizaremos las siguientes etiquetas lingüísticas:




Etiqueta lingüística para el atributo Altura




Etiqueta lingüística para el atributo Calidad

Consulta: mostrar los jugadores, con sus equipos y alturas, de aquellos jugadores Bajos, con un grado mínimo de 0.5.

Si usamos j como nombre de variable para el atributo Jugador, e para Equipo y a para Altura, la expresión que resuelve la consulta es:

$\{ j, e, a \mid \exists c (R(j, e, a, c) \wedge \Theta^{=} (a, Bajo) \geq 0.5 \}$

Primero se calculan cuáles son las tuplas (j, e, a) que cumplen con el predicado. En este caso, existen 9 tuplas, puesto que las comparaciones se hacen en función de la etiqueta lingüística para el atributo Altura. Los resultados se muestran en la siguiente tabla, donde $c_{Altura}$ es la componente de valor calculado en cada caso:

| Cabecera | j | e | a | $c_{Altura}$ |
|---|---|---|---|---|
| Cuerpo | J2 | Argentina | Bajo | 1 |
| | J4 | Canadá | Bajo | 1 |
| | J5 | Colombia | Bajo | 1 |
| | J7 | Guatemala | Bajo | 1 |
| | J11 | Jamaica | Bajo | 1 |
| | J12 | Jamaica | Normal | 0.5 |
| | J15 | Malasia | Bajo | 1 |
| | J18 | Suiza | Bajo | 1 |
| | J20 | Suiza | Normal | 0.5 |

El atributo $C_{Altura}$ para cada una de las tuplas del resultado se calcula como:

$c_{Altura} = \Delta_a ( \psi ( j, e, a ) )$

Por ejemplo, en la novena tupla tenemos:

$c_{Altura} = \Delta_a$ (J20, Suiza, Normal) ( $\psi$ ( j, e, a ) )
= (J20, Suiza, Normal) $\wedge$ $\Theta^{=}$ (Normal, Bajo)
= 1 $\wedge$ 0.5
= 0.5

## FSQL: un lenguaje para consultas difusas

Siguiendo la misma línea de razonamiento, el modelo GEFRED ha propuesto una serie de criterios para efectuar la representación del conocimiento impreciso y lo ha hecho a través de la extensión del SQL relacional, lo que ha dado lugar a FSQL: Fuzzy Structured Query Language, o bien, SQL Difuso.

Para simplificar y agrupar características comunes que faciliten la representación de los diferentes tipos de datos que se definieron en el dominio difuso generalizado, se propone la siguiente clasificación:

1. Datos precisos

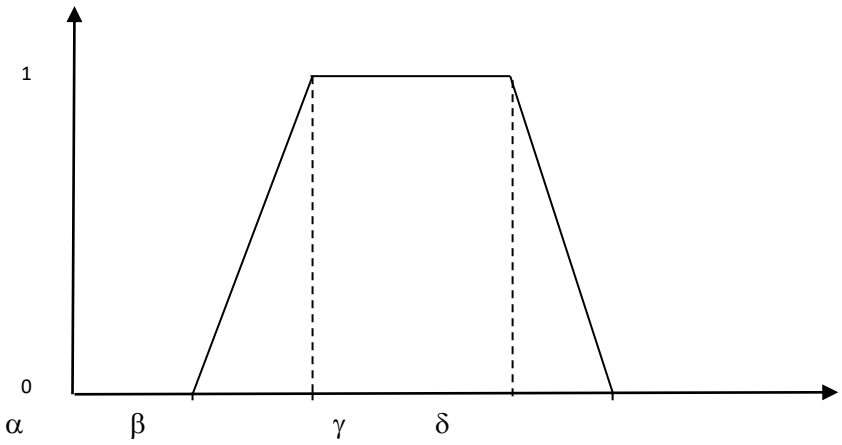
   También llamados crisp o clásicos, toman la forma que maneja el motor relacional sobre el cual se extenderán las definiciones y consultas.

2. Datos imprecisos

   Los datos imprecisos se clasifican en dos tipos:

   - Datos imprecisos definidos sobre un referencial, o dominio subyacente, ordenado. Este grupo contiene:

     – las distribuciones de posibilidad sobre dominios continuos o discretos que contienen los parámetros con los cuales deben evaluarse las etiquetas lingüísticas, como la que se muestra en la siguiente figura:




Formato de una distribución de posibilidad con parámetros $\alpha$, $\beta$, $\gamma$ y $\delta$

- los valores aproximados
- los intervalos de posibilidad.

- Datos con analogía sobre un referencial no ordenado

  Este grupo está construido sobre dominios subyacentes discretos no ordenados, en los que se encuentran definidas relaciones de semejanza, de modo que debemos almacenar tanto los valores como dichas relaciones:

  - Escalares simples, que se representan como pares ordenados cuyo valor de posibilidad es 1: { (1, d) }
  - Distribución de Posibilidad sobre escalares: son pares ordenados en donde a cada valor, d, se le asigna una p: ({ (p, d) }
  - Valores especiales: Unknown, Undefined y Null

¿Cómo se representan estos tipos en FSQL?. Se representan mediante la descripción de un conjunto de atributos cuyos valores, o carencia de los mismos, indican el tipo de datos que contiene cada dominio subyacente.

Observemos, a modo de ejemplo, la siguiente tabla:

| Tipo de valor | Atributos | | | | |
|---|---|---|---|---|---|
| | 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 |
| Etiqueta | 4 | Nombre | Null | Null | Null |
| Intervalo [m,n] | 5 | n | Null | Null | m |
| Aproximadamente (d) | 6 | d | d – margen | d + margen | margen |
| Trapecio [$\alpha$,$\beta$,$\gamma$,$\delta$] | 7 | $\alpha$ | $\beta$ - $\alpha$ | $\gamma$ - $\delta$ | $\delta$ |

Para un atributo difuso, 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.

## ¿Y el motor?

¿Tenemos un motor que manipule este tipo de datos, acepte una consulta difusa y la resuelva?.

En el año 1.994, J.M. Medina [25] propuso un módulo a adicionar a un motor relacional, para extender su capacidad permitiendo así la representación y manipulación de la información imprecisa.

Este módulo, llamado FIRST – Fuzzy Interface for Relational SysTems, utiliza GEFRED como modelo teórico y los recursos del Modelo Relacional para representar la información imprecisa.

Para permitir las operaciones típicas de la base de datos: creación de tablas, etiquetas, etc., la implementación de FIRST ha extendido el SQL para el tratamiento de los nuevos datos.

# CAPÍTULO 6

# CONCLUSIONES

## El diseño y la implementación en la extensión del Modelo Relacional

El proceso de definición de una nueva base de datos se refiere, usualmente, al proceso de diseño de la nueva base de datos.

En este aspecto, el proceso de diseño de una base de datos Relacional Extendida no difiere absolutamente en nada del proceso que se sigue para diseñar una base de datos Relacional, salvo en que se debe tener en cuenta la incorporación de las relaciones que definirán las medidas y los parámetros que contendrán los valores de los factores de escala, de los pesos relativos y de los radios de vecindario respectivos.

Sin embargo, la definición del modelo lógico y del modelo físico no son tan triviales.

Para implementar consultas de objetivos se deben efectuar un conjunto de modificaciones en el DBMS:

- el DDL debe aceptar la definición de las extensiones de los atributos
- debe poseer un procesador de las consultas que acepte un lenguaje extendido que permita expresar la consulta en lenguaje simbólico
- debe poder permitir la definición de un procedimiento recursivo, llamado Distancia, que acepte dos valores, el nombre de la medida, el factor de escala y el radio de vecindario y determine si el valor de la distancia que arroja es menor que el radio de vecindario

El cálculo recursivo de la distancia puede hacer lenta la resolución de la consulta. Se ha demostrado que el orden de complejidad cuando intervienen n relaciones en el cálculo de la distancia, es del orden de 2n, por lo tanto la forma de minimizar el tiempo es hacer n lo menor posible.

## El diseño y la implementación en una Base de Datos Relacional Difusa

El Modelo GEFRED no solo proporciona todas la herramientas que permitirán el modelado conceptual de una Base de Datos Difusa sino que, además presenta, con todo el rigor de la formalización, el Álgebra Relacional Difusa y el Cálculo Relacional Difuso de Dominios asociado.

El Cálculo Relacional Difuso es una extensión del Cálculo Relacional clásico, por lo que es también posible usarlo en una Base de Datos Relacional clásica.

Cuando los valores de los atributos difusos están asociados a distribuciones de posibilidad o a etiquetas lingüísticas definidas por extensión, al resultado de cada consulta se le incorpora un conjunto de tuplas que satisface la condición con un cierto grado en el intervalo [0,1]. Cada uno de estos grados se calcula aplicando la función $\Delta$.

El inconveniente de las Bases de Datos Relacionales Difusas es la influencia que pueda tener la "subjetividad" en la definición y uso de los conceptos difusos.

## La convergencia de ambas propuestas

Las consultas difusas sobre una Base de Datos Relacional Difusa son posibles porque FSQL permite la definición de etiquetas. Estas etiquetas vienen dadas por funciones de distribución que se definen a través de sus parámetros o bien contienen un conjunto difuso que se define por extensión asignando a cada elemento del dominio su correspondiente valor de posibilidad.

Como mencionamos anteriormente, sería deseable eliminar de los modelos de una Base de Datos Relacional Difusa toda componente de subjetividad. Por ejemplo, si a dos personas diferentes le damos la tarea de definir la etiqueta lingüística "joven", la definición de ambas etiquetas puede ser similar pero, difícilmente, será igual. Ahí el problema de la subjetividad.

Y, a consecuencia de ello, nos debemos preguntar cuál de las dos será la "mejor etiqueta", porque dependiendo de cual elijamos serán las tuplas que incluirá el resultado de la consulta.

Si bien GEFRED soluciona, o palea, este inconveniente con el uso de umbrales de cumplimiento, estos en sí mismos ya poseen un grado de subjetividad, pequeño, pero lo poseen.

Por otra parte, hemos visto la definición de distancia entre tuplas en la extensión del Modelo Relacional propuesta por Motro para el tratamiento de la información imprecisa.

Los valores contenidos en este modelo de Motro son todos crisp, no hay subjetividad, todos responden a las restricciones del Modelo Relacional, aún las relaciones adicionales que llevan en sí las medidas de cada atributo cuando éstas no son las estándares.

Así que la pregunta es: en GEFRED, ¿no podemos sustituir la etiqueta lingüística que define una función de posibilidad por el nombre de la relación que contendrá la medida para el cálculo de la distancia?. Porque si la respuesta es sí, hemos eliminado la subjetividad que implica la definición de los parámetros de la distribución de posibilidad.

Esto conllevaría, por otra parte, a una ampliación de los tres modelos involucrados en el proceso de diseño de una base de datos difusa: el conceptual, el físico y el lógico.

GEFRED, debido a su fuerte fundamentación, permitiría efectuar estas modificaciones y, así, sería posible optimizar un aspecto que redundaría en la obtención de un resultado más objetivo al momento de efectuar una consulta difusa.

# BIBLIOGRAFÍA

[01] J. Arias Figueroa, Apuntes "Sistemas de Bases de Datos". Licenciatura en Análisis de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional de Salta. 1.996.

[02] J. Arias Figueroa, Seminario de Sistemas "Lenguaje de Modelado Semántico". Licenciatura en Análisis de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional de Salta. 1.996.

[03] J.S. Bowman, S.L. Emerson, M. Darnovsky, "The Practical SQL Handboock", Tercera Edición. Addison-Wesley. 1.997.

[04] B.P. Buckles, F.E. Petry, "A Fuzzy Representation of Data for Relational Databases". Fuzzy Sets and Systems, 7, pp.213-226. 1.982.

[05] B.P. Buckles, F.E. Petry, "Fuzzy Databases and their Aplications", en "Fuzzy Information and Decition Processes", Vol. 2, Eds. M. Grupta, E. Sanchez. North Holand, Amsterdam, pp. 361-371. 1.982.

[06] B.P. Buckles, F.E. Petry, "Extending the Fuzzy Database with Fuzzy Numbers". Information Science 34, pp. 45-55. 1.984.

[07] D.D. Chamberlain, R.F. Boyce, "SEQUEL: A Structured English Query Language". Proc. ACM SIGMOD Wiorkshop on Data Description. Access and Control. 1.974.

[08] E.F. Codd, "Relational Completeness of Data Base Sublanguajes" en Randall J. Rustin, Data Base Systems, Courant Computer Science Symposia Series 6. Prentice Hall. 1.972.

[09] E. F. Codd, "Extending the Database Relational Model to Capture More Meaning". IBM Research Laboratory. ACM Transactions on Database Systems, Vol. 4, No. 4, 397-434. Diciembre de 1.979.

[10] Codd, E.F.. Missing Information (Aplicable and Inapplicable) in Relational Databases. ACM SIGMOND record. Vol 15. No. 4. 1.986.

[11] E.F. Codd, "The relational model for databases management". Addison-Wesley. 1.990.

[12] C.J. Date, "An introduction to Database Systems, Volume II". Addison Wesley. 1.983.

[13] C.J. Date, H. Darwen, "A Guide to SQL Standard", Cuarta Edición. Addison-Wesley. 1.997.

[14] C.J. Date, "Introducción a los Sistemas de Bases de Datos", Séptima Edición. Edison Wesley. 2.001.

[15] R. Elmasri, S.B. Navathe, “Sistemas de Bases de Datos”. Addison Wesley. 1.997.

[16] J. Galindo Gómez, Tesis Doctoral “Tratamiento de la imprecisión en Bases de Datos Relacionales: Extensión del modelo y adaptación de los SGBD actuales”. Universidad de Granada. España. 1.999.

[17] J. Galindo, J.M. Medina, M.C. Aranda Garrido, “Querying Fuzzy Relational Databases Treogh Fuzzy Domain Calculus”, International Journal of Intelligent Systems, Vol. 14, issue 3, 1.999.

[18] J. Galindo, J.M. Medina, A. Villa, O. Pons, “Fuzzy Comparators for Flexible Queries to Databases”. Iberoamerican Conference on Artificial Intelligence. IBERAMIA’98, Lisbon (portugal), pp. 29-41, Octubre 1.998.

[19] H. Gil, P. Rao, “La integración de la información para la mejor toma de decisiones. Data Warehousing”. Prentice Hall Hispanoamericana. 1.996.

[20] S. Ginsburg, R. Hull, “Order Dependency in the Relational Model”. Theoretical Computer Science 26:149-195. 1.993.

[21] J.Grant, “Incomplete Information in a Relational Database”. FundamentaInformaticae, 3, pp. 363-378, 1.980.

[22] M. Lacroix, A. Pirotte, “Domain-Oriented Relational Languages”. Proc. 3ed. International Conference on Very Large Data Bases. Octubre 1.977.

[23] P. Mac Gaul, M. Díaz, Apuntes “Bases de Datos”. Cátedra de Bases de Datos. Licenciatura en Análisis de Sistemas. Facultad de Ciencias Exactas. Universidad Nacional de Salta. 2.001.

[24] D.J. McLeod, “High Level Definitions of Abstract Domain in the Relational Data Base System”. Journal of Computer Languages 2(3): 61-73, July, 1.977.

[25] J.M. Medina, “Bases de Datos Relacionales Difusas. Modelo teórico y aspectos de su implementación”. Ph. Doctoral Thesis, University of Granada (Spain), May 1.994.

[26] J.M. Medina, O. Pons, M.A. Vila, “GEFRED. a Generalized Model of Fuzzy Relational Data Bases”. Information Sciences, 76 (1-2), 87-109, 1.994.

[27] J.M. Medina, O. Pons, M.A. Vila, “FIRST. A Fuzzy Interface for Relational SysTems”. VI International Fuzzy Systems Association World Congress (IFSA 1995). Sao Pablo (Brasil), 1.995.

[28] A. Motro, “Extending the Relational Database Model to Support Goal Queries”. Expert Database Systems. 1.987.

[29] A. Motro, “ VAGUE A user interface to relational databases that permits vague queries”. ACM Transactions on Office Information Systems, Vol. 6, No. 3, 187-214. 1.988.

[30] A. Motro, “Accommodating imprecision database systems issues and solutions”. SIGMOD RECORD, Vol. 19, No. 4. 1.990

[31] A. Motro, “Uncertain Management in Information Systems. From needs to solutions”. Kluwer Academic Publishers. Capítulo. 2, Páginas 9–34. 1.996.

[32] H. Prade, C. Testemale, “Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries”. Information Science 34, pp. 115-143. 1.984.

[33] H. Prade, C. Testemale, “Fuzzy Relational Databases: Representational issues and Reduction Using Similarity Measures”. J. Am. Soc. Information Science 38 (2), pp. 118-126. 1.987.

[34] H. Prade, C. Testemale, “Representation of Soft Constraints and Fuzzy Atributes Values by Means of Possibility Distributions in Databases”. En “Analysis of Fuzzy Information”, Vol II: “Artificial Intelligence and Decision Systems”. Ed. J. Bezdek, CRC Press., pp. 213-229. 1.987.

[35] M. Umano, S. Fukami, “Fuzzy Relational Algebra for Possibility-Distribution-Fuzzy-Relational Model of Fuzzy Data”. Journal of Intelligent Information Systems, 3, pp. 7-28. 1.994.

[36] E.A. Wong, “Statistical Approach to Incomplete Informations in Database Systems”. ACM Trans. on Database Systems 7, pp.479-488, 1.982.

[37] L.A. Zadeh, “Fuzzy Sets”. Information Control, 8, pp. 338-353, 1.965.

[38] L.A. Zadeh, “Similarity Relations and Fuzzy Orderings”. Information Sciences, 3, pp.177-200, 1971.

[39] L.A. Zadeh, “Fuzzy Logic, Neural Networks and Soft Computing. Fuzzy System”. 1.994.

[40] M. Zemankova-Leech, A. Kandel, “Fuzzy Relational Databases – A Kay to Expert Systems”. Koln, Germany, Verlag. 1.984.

[41] M. Zemankova-Leech, A. Kandel, “Implementing Imprecision in Information Systems”. Information Scie4nces, 37, pp. 107-141. 1.985.

# AGRADECIMIENTOS

Gracias ...

**...** a mi director, Gustavo Gil, por guiarme, con paciencia pero con firmeza, durante todo el proceso que concluyó con la elaboración de este trabajo,

**...** a mi co-director, Gustavo Rossi, quien me asesoró en la elección de este tema, que resultó atrapante y abierto a nuevas aplicaciones,

**...** a Julio Arias, mi maestro, el que me enseñó a hacer los primeros DER's y me introdujo así en el fascinante mundo de las bases de datos,

**...** a Marcelo, mi marido, que salió airoso de las embestidas de mis malhumores debidos a varios "no me sale", "no encuentro el paper que busco", "no me gusta cómo va quedando",….

**...** a mis hijos, Martín, Claudio, Julia, Eugenia y Ana, porque cuando los miro recuerdo el motivo por el cuál vale la pena seguir avanzando.

Lic. Patricia Silvia Mac Gaul  Mag. Gustavo Daniel Gil  Dr. Gustavo Rossi