Manual de Usuario

POSTGRESQLf: Implementación de Extensiones Difusas de manera Fuertemente Acoplada sobre el RDBMS PostgreSQL

22 de Enero del 2010

Ayudante de Investigador: Lic. Rodolfo Vegas

Gerentes de Proyecto: Ana Aguilera (phd)

Tineo Leonid (phd)

José Cadenas (msc)

Contenido

MANUAL DE USUARIO

1. **Introducción**

   El presente manual de usuario, tiene como finalidad dar a conocer de una manera detallada y sencilla, el proceso de instalación de la extensión de PostgreSQLf; además de mostrar las funcionalidades implementadas en esta investigación y como se manejan en dicha extensión, con el propósito de que los usuarios se familiaricen con la extensión de PostgreSQLf y de los grandes beneficios que le pueden aportar a sus actividades tan sólo por el hecho de estar basado en la lógica difusa.

   Con el desarrollo de esta extensión se esta poniendo en práctica el mecanismo llamado transferencia tecnológica, la cual da impulso al desarrollo y crecimiento de los diversos sectores de la sociedad y en especial aquellos que utilicen como manejador de base de datos PostgreSQLf, impulsando así la competencia y los beneficios económicos de las instituciones y organizaciones de Venezuela con el fin de prepararlos a las nuevas tecnologías desarrolladas en el mundo, además serían pioneros por usar la extensión de PostgreSQLf ya que es tecnología nueva, no desarrollada antes en nuestro país.

2. **Acerca de PostgreSQLf**

   Para la versión final del RFDBMS PostgreSQLf, se incluyen las funcionalidades de Creación de Comparadores Difusos trapezoidales y por extensión, y Selección de Datos de acuerdo a la extensión del lenguaje SQLf para el manejo de comparadores, consultas difusas en el from y operaciones conjuntistas difusas.

   A partir de esto, en el presente manual se definen entonces de forma detallada dos componentes de la última versión de PostgreSQLf:

   - Instalación de PostgreSQLf: especificación de pasos que se deben llevar a cabo para instalar esta versión del sistema.
   - Lenguaje de Manejo de Datos y Selección de Datos que involucren comparadores difusos, predicados difusos, cuantificadores difusos, modificadores difusos, conectores difusos, consultas difusas en el from, operaciones conjuntistas difusas, group by con having difuso y el uso de la calibración. A su vez se definen una serie de ejemplos que faciliten la comprensión de dicho lenguaje por parte del lector.

   2.1. **Instalación**

   A continuación se describe de manera detallada, la forma en la cual se debe instalar el RFDBMS PostgreSQLf en una plataforma de sistema operativo basada en LINUX. Así mismo, se hace referencias a consideraciones que se deban tomar en cuenta al llevar a cabo dicho proceso.

Requerimientos del Sistema

En general, cualquier sistema compatible con el UNIX moderno está en capacidad de soportar la ejecución del PostgreSQLf sobre la plataforma respectiva. De cualquier manera, para la instalación de este SMBD, es necesario constar con cada uno de los siguientes paquetes de software descritos a continuación:

GNU make:

Cualquier otro sistema de construcción "*make*" no funcionará a la hora de construir el sistema a partir de su código fuente. La mayoría de los sistemas basados en UNIX poseen este paquete como predeterminado, específicamente con el nombre "*gmake*". Para probar su existencia, sólo se debe hacer:

```
# gmake –version
```

Compilador ISO/ANSI C:

Son recomendadas las versiones recientes de GCC, pero en el caso de PostgreSQL, se ha demostrado que es posible el que sea construido a partir de una gran variedad de compiladores de distintos proveedores.

Herramienta de Compresión TAR:

Este paquete es necesario para descomprimir el código fuente del sistema PostgreSQLf.

Librería Readline de GNU:

Esta librería es usada por defecto por parte del sistema, aunque en los casos en los cuales no se desee utilizar la misma en conjunción con el RT-PostgreSQL, únicamente se debe especificar el comando *--without-readline* a la hora de establecer la configuración del sistema (*./configure --without-readline*). Esta librería aporta funcionalidades de edición de líneas y recuperación de historiales de ejecución de comandos.

Librería de Compresión ZLIB:

Al igual que la librería READLINE, esta librería es usada por defecto por parte del sistema, y de la misma manera, en los casos en los cuales no se desee utilizar la misma en conjunción con el RT-PostgreSQL, únicamente se debe especificar el comando *--without-zlib* a la hora de establecer la configuración del sistema (*./configure --without-zlib*). Esta librería permite el uso de archivos comprimidos a partir de los comandos *pg_dump* y *pg_restore*.

Requerimientos Adicionales:

En vista de que el sistema RT-PostgreSQL es una versión modificada del sistema base de PostgreSQL, es necesario constar con ciertos paquetes que permitan hacer uso de esta versión, para el soporte de variaciones en los archivos léxicos y de gramática. Para dar soporte a dichas variaciones, es necesario entonces de manera obligatoria, instalar los paquetes *GNU Flex* y *Bison*, en sus versiones iguales o superiores a la 2.5.4 y 1.875 respectivamente.

Detalles de Instalación

A continuación se describe paso a paso, la manera en la cual debe llevarse a cabo la instalación del RT-PostgreSQL.

Descomprimir el Archivo:

Suponiendo que el archivo *PostgreSQLf.tar.gz* fue descargado y colocado en el directorio */usr/local/*, debe desempacarse y descomprimirse así:

cd /usr/local

tar -xzf postgresqlf.tar.gz

Ahora debe accederse al directorio postgresql de la siguiente manera:

cd postgresql

Configuración:

El próximo paso en el procedimiento de instalación es la configuración del sistema base, así como la elección de las opciones adicionales de construcción del mismo. Esto debe llevarse a cabo mediante la ejecución del SCRIPT de configuración. Para la configuración por defecto solo se debe hacer *./configure* desde el directorio base del sistema.

A partir de esto, el *script* ejecutará una serie de pruebas para determinar el valor de distintas variables del sistema, así como comprobar a su vez la configuración actual del sistema en sí, permitiendo de esta manera saber si es posible o no instalar el RT-PostgreSQL sobre la plataforma en cuestión.

Para llevar a cabo una configuración personalizada, existen distintas opciones, de las cuales se describen a continuación las más importantes:

--prefix = PREFIX

El sistema se instalará por defecto en */usr/local/pgsql/*, pero en caso de que el mismo se desee instalar en alguna otra ubicación, *PREFIX*, será entonces la dirección del directorio a utilizar.

--with-pgport = NUMBER

El sistema por defecto posee como número de puerto, 5432. Si se desea utilizar algún otro número, el mismo estará identificado por *NUMBER.*

--enable-depend

Permite llevar un chequeo dinámico de dependencias. Por medio de esta opción los *makefiles* correspondientes a todos los archivos serán configurados de manera tal que todos los archivos serán reconstruidos en caso de que cualquiera de sus archivos de cabecera (o *header files*) sea modificado. Al igual que en la opción anterior, se recomienda utilizar esta opción cuando se están llevando a cabo modificaciones sobre el código fuente del sistema.

Para conocer todas las opciones de configuración disponibles, se recomienda revisar el archivo *INSTALL*, el cual se encuentra en el directorio base del RTDBMS.

Ahora, para llevar a cabo la configuración, se debe hacer entonces:

./configure

Construcción:

Para la construcción del sistema sólo se debe ejecutar *make* así:

make

Una vez que se haya construido por completo el sistema, se mostrará el siguiente mensaje: *All of PostgreSQL is successfully made. Ready to install.*

Instalación de Archivos:

Finalmente, para llevar a cabo la instalación de los archivos respectivos del sistema, se debe hacer:

make install

A partir de este comando, se colocarán de manera adecuada los archivos necesarios dentro del directorio especificado en el apartado *1.2.3-PREFIX*. Es necesario evaluar siempre que se posean los permisos necesarios para escribir datos sobre el área especificada.

Configuración del Usuario postgres:

Se debe crear una cuenta de usuario UNIX para poseer y gestionar los archivos de bases de datos de PostgreSQLf. Este usuario es denominado *postgres*, el cual será considerado el usuario *root* o *superusuario* de PostgreSQLf.

Para esto es necesario tener privilegios de *root*. En una máquina Linux, el usuario *postgres* debe añadirse por medio del comando *useradd*, así:

adduser postgres

Crear Cluster para la Base de Datos:

Este es el *cluster* que contendrá las bases de datos que vayamos creando. Al momento de inicializar un *cluster* de bases de datos de PostgreSQL, se tiene que especificar el directorio donde se creará el mismo. El propietario de ese directorio tiene que ser un usuario que no sea *root*, en este caso, el usuario será *postgres*. Como ya se ha indicado en el */etc/passwd* que el HOME del usuario postgres es */var/pgsql/data*, siguiendo con esa lógica, se indicará que ése será el directorio donde se localice el *cluster* de bases de datos. Entonces se crea el directorio y se cambia el propietario:

mkdir -p /var/pgsql/data

chown postgres /var/pgsql/data

Ahora se cambia al usuario *postgres* y se inicializa el *cluster* de las bases de datos con el comando initdb:

su – postgres

postgres# /usr/local/pgsql/bin/initdb -D /var/pgsql/data

Si la inicialización del *cluster* se llevó a cabo con éxito, el resultado final del comando será similar al que se muestra a continuación:

Success. You can now start the database server using:

/usr/local/pgsql/bin/postmaster -D /var/pgsql/data

or

/usr/local/pgsql/bin/pg_ctl -D /var/pgsql/data -l logfile start

Iniciar el Servidor:

Lo que sigue es iniciar el servidor de bases de datos.

Antes de esto, se recomienda agregar las próximas líneas al *profile* (archivo *.bash_profile* o *.bashrc*) del usuario *postgres*, el cual debe encontrarse en el directorio HOME de este usuario, */var/pgsql/data/*.

PGDATA=/var/pgsql/data
PATH=/usr/local/pgsql/bin:$PATH
MANPATH=/usr/local/pgsql/man:$MANPATH
export PATH PGDATA MANPATH

Esto debería funcionar para cualquier *shell* basada en *sh* (incluyendo *bash* y *ksh*). Esto se hace para evitar la molestia de poner toda la ruta del *cluster* de la base de datos cada vez que se inicializa el servidor, y a su vez para agregar al *PATH* del sistema el directorio donde están los comandos de PostgreSQL:

Nota: es necesario *loguearse* de nuevo a la consola del sistema una vez se haya actualizado el archivo correspondiente, de manera tal que la *shell* haga uso de la nueva configuración.

Ahora, como usuario *postgres* inicializar el servidor de bases de datos con el comando *pg_ctl*, así:

postgres# pg_ctl start –l nombre_log
postmaster starting

De esta manera ya se tiene PostgreSQL funcionando y cada uno de los registros del sistema (*logs*) se almacenarán en el archivo *nombre_log*.

Para conectarse a un servidor de base datos PostgreSQL se utiliza el comando *psql*. Se le pueden pasar muchos parámetros a este comando de la forma *psql* [OPTIONS]... [DBNAME [USERNAME]].

Para probar, puede conectarse al servidor de PostgreSQL sobre la base de datos base que se acaba de crear:

postgres# psql template1
Welcome to psql 8.1.4, the PostgreSQL interactive terminal. Type:
\copyright for distribution terms
\h for help with SQL commands
\? for help with psql commands
\g or terminate with semicolon to execute query
\q to quit
template1=#

Ahora, *template1* es el nombre de una de las dos bases de datos que por defecto se encuentran en el *cluster*. Entre otras cosas, *template1* se utiliza como plantilla base para la creación de una base de datos nueva.

Para mayor información acerca del uso y manejo de PostgreSQL se recomienda descargar los manuales de PostgreSQL desde la siguiente dirección:

[http://www.postgresql.org/files/documentation/pdf/8.2/postgresql-8.2-A4.pdf]

Para salir del *shell* del PostgreSQL se utiliza el comando *q*.

Parar el Servidor:

Ahora, para detener el servicio del servidor de PostgreSQL, se debe hacer lo siguiente: postgres# pg_ctl stop

postmaster stopped

2.2. Consideraciones Importantes:

El sistema PostgreSQLf ha sido diseñado tomando en cuenta la estructura y definiciones de la extensión del lenguaje de consultas difusas SQLf para el de consultas flexibles. Debido a que el lenguaje SQLf es una extensión realizada sobre el estándar SQL92, son muchos los aspectos del DDL (Data Definition Language) y el DML (Data Manipulation Language) de SQLf que se encuentran descritos por el estándar SQL92, por lo que no se hace mayor referencia a dichos aspectos dentro del presente manual. De la misma manera se cumple esto para aquellos aspectos específicos del lenguaje SQLf y su extensión.

Como consecuencia de esto, mucha de la sintaxis detallada en el manual se encuentra resumida, por lo que el lector debe tomar en cuenta que al ver puntos suspensivos (…) dentro de una descripción sintáctica, estará en presencia de un aspecto de la sintaxis de SQL92, el cual fue omitido.
Para mayor información acerca de cada uno de los aspectos descritos por el estándar SQL92, se recomienda visitar el siguiente enlace:
[http://www.postgresql.org/docs/8.2/interactive/sql-commands.html]

2.3. Lenguaje de Manejo de Datos

A continuación se describe la forma en la cual se deben llevar a cabo las operaciones de Creación de Tablas e Inserción y Selección de Datos de acuerdo a la extensión de SQLf para el Manejo de Datos imprecisos.

Comparadores Difusos

Creación de Comparadores Difusos Trapezoidales

CREATE COMPARATOR <nombre_comparador> as <expresión>

donde:
<expresión> es:
(INFINITE, INFINITE, expresión$_1$, expresión$_2$) para comparadores difusos decrecientes.
| (expresión$_1$, expresión$_2$, expresión$_3$, expresión$_4$) para comparadores difusos unimodales.
| (expresión$_1$, expresión$_2$, INFINITE, INFINITE) para comparadores difusos crecientes.

Donde expresión$_i$ es la expresión que permite calcular el valor de la comparación a partir de los dos elementos comparados, los cuales son

denotados con la variable x (para el término izquierdo) y la variable y (para el término derecho). Esta expresión puede utilizar conjuntos difusos.

Creación de Comparadores Difusos por Extensión

CREATE COMPARATOR name ON domdisc AS conj

donde:

i. name: nombre del comparador difuso.
ii. domdisc: nombre del dominio discreto.
iii. conj: lista de pares ordenados con su respectivo grado de membresía, su sintaxis es la siguiente:
((elementoa,elementob)/Gr.Memb.), …)

Los $elementos_i$ pertenecen al dominio discreto definido al momento de crear el comparador difuso por extensión, se presentan en pares ordenados junto con el grado de membresía que concierne a dicho par, separados por el símbolo "/".

La sintaxis definida anteriormente hace posible la creación de comparadores difusos crecientes, decrecientes y unimodales; además de los comparadores difusos por extensión.

Ejemplos

- Comparadores Difusos Trapezoidales
  CREATE COMPARATOR '~' ON 1 .. 100 AS ('x*y-5','y','y','x*y+5');
  CREATE COMPARATOR '<<' ON 1 .. 100 AS ('INFINITE','INFINITE','y/3','y');
  CREATE COMPARATOR '>>' ON 1 .. 100 AS ('y','y*3','INFINITE','INFINITE');
- Comparadores Difusos por Extensión
  CREATE COMPARATOR 'parecido1' ON plato AS ('(china,thailandesa)/0.8','(japonesa,china)/0.4','(portuguesa,espanola)/0.7','(espanola,china)/0.1','(venezolana,colombiana)/0.95','(venezolana,mexicana)/0.2','(mexicana,argentina)/0.4','(china,vietnamita)/1','(venezolana,peruana)/0.5','(colombiana,peruana)/0.36','(portuguesa,venezolana)/0.8','(espanola,colombiana)/0.25','(espanola,venezolana)/0.69','(portuguesa,peruana)/0.85','(japonesa,vietnamita)/0.92','(thailandesa,peruana)/0.02');
  CREATE COMPARATOR 'parecido2' ON piel AS ('(negra,india)/0.8','(negra,morena)/1.0','(blanca,negra)/0.01','(blanca,

india)/0.2','(india,triguena)/0.5','(morena,blanca)/0.01','(triguena,blanca)/0.01','(negra,triguena)/0.6');

Predicados Difusos

Creación de Predicados Difusos Atómicos

CREATE FUZZY PREDICATE term ON i_b .. i_e AS conj;

donde:

i. term es un string de caracteres con el nombre del predicado difuso,
ii. i_b, i_e números enteros que indica el dominio.
iii. conj es la definición de la función de membresía de un trapecio que puede ser de tres formas:
    a) (infinite, infinite, i_1, i_2),
    b) (i_1, i_2, infinite, infinite) ó
    c) (i_1, i_2, i_3, i_4),

Las cuales describen función de membresía monótona decreciente (al menos), monótona creciente (a lo sumo) y unimodal respectivamente.

Ejemplos

- CREATE FUZZY PREDICATE joven ON 0 .. 120 AS (infinite, infinite, 20, 40);
- CREATE FUZZY PREDICATE viejo ON 0 .. 120 AS (40, 80, infinite, infinite);
- CREATE FUZZY PREDICATE maduro ON 0 .. 120 AS (20, 40, 60, 80);
- CREATE FUZZY PREDICATE cerca_50 ON 0 .. 120 AS (20, 49, 51, 80);

Creación de Predicados por Extensión y por Expresión

CREATE FUZZY PREDICATE term ON dom AS conj;

donde:

i. term es un string de caracteres con el nombre del predicado difuso.
ii. dom es un string de caracteres que indica el dominio discreto.
iii. conj es la definición de la función de membresía que puede ser:
    a) (val1/grado1, val2/grado2,..., valn/gradon), para la función de membresía por extensión.
    b) (expr) siendo expr una expresión para la función de membresía por expresión.

Ejemplos

- CREATE FUZZY PREDICATE gordo ON contextura AS ('obeso/1', 'grueso/0.7', 'normal/0.4'); *Predicado por extensión*
- CREATE FUZZY PREDICATE aprox ON promedio AS ('x+0.5/10'); *Predicado por expresión*

Creación de Modificadores Difusos

CREATE MODIFIER nombre AS POWER n
CREATE MODIFIER nombre AS norm POWER n
CREATE MODIFIER nombre AS TRANSLATION d

donde:

i. n es la potencia a la cual se va a elevar el grado de membresía.
ii. norm es una norma o conorma triangular que se le aplica al grado de membresía
iii. d es un valor que se suma al argumento del predicado original para hacer la traslación.

Ejemplos

- CREATE MODIFIER muy AS POWER 2;
- CREATE MODIFIER extremadamente AS max ('x+y-1', '0') POWER 4;
- CREATE MODIFIER realmente AS TRANSLATION -10;

Creación de Conectores Difusos

CREATE CONNECTOR nombre AS expr

donde:

i. expr es la expresión que permite calcular el valor del predicado compuesto a partir de los valores de sus miembros.

Ejemplo

- CREATE CONNECTOR imp AS max ('1-x, y');

Cuantificadores Difusos

CREATE [ABSOLUTE/RELATIVE] QUANTIFIER <nombre> AS <conj>

donde:

i. ABSOLUTE/RELATIVE dependerá del tipo de cuantificador que se desee crear.
ii. nombre es un string de caracteres con el nombre del cuantificador difuso.
iii. conj es la definición de la función de membrecía de un trapecio que puede ser de tres formas:
    a) (infinite, infinite, f_1, f_2),
    b) (f_1, f_2, infinite, infinite) ó
    c) (f_1, f_2, f_3, f_4),

    Las cuales describen función de membrecía monótona decreciente (al menos), monótona creciente (a lo sumo) y unimodal respectivamente.

Ejemplos

- CREATE ABSOLUTE QUANTIFIER a_lo_sumo_2 AS (0.0, 0.0, 2.0, 3.0);
- CREATE ABSOLUTE QUANTIFIER al_menos3 AS (1.0, 3.0, infinite, infinite);
- CREATE ABSOLUTE QUANTIFIER aprox_5 AS (3.0, 5.0, 5.0, 7.0);
- CREATE RELATIVE QUANTIFIER la_minoria AS (0.0, 0.0, 0.25, 0.5);
- CREATE RELATIVE QUANTIFIER la_mayoria AS (0.5, 0.75, 1, 1);
- CREATE RELATIVE QUANTIFIER aprox_la_mitad AS (0.25, 0.5, 0.5, 0.75);

Vistas Difusas

CREATE VIEW <nombre_vista> AS <condicion_difusa>;

donde:
i. <nombre_vista> es: La etiqueta o nombre que se usa para identificar la vista.
ii. <condición_difusa> es: La consulta difusa empleada para crear la vista.

Ejemplos

- CREATE VIEW repuestos_baratos_vol_bajo AS SELECT ps_partkey FROM partsupp WHERE ps_supplycost = barato;

2.4. Selección de Datos.

A continuación se presenta la sintaxis para la Selección de Datos empleando predicados difusos, conectores difusos, cuantificadores difusos, modificadores difusos, comparadores difusos, consultas difusas en el from, operaciones conjuntistas difusas, consultas particionadas difusas, creación y consulta de Vistas difusas y el uso de los operadores CHECK y UPDATE difusos por la extensión de SQLf para el Manejo de Datos en base a consultas con preferencias.

- SELECT <ATRIBUTOS> FROM <TABLAS> WHERE <lista_de_comparadores_difusos>

  donde:
  <lista_de_comparadores_difusos> es (comparador$_1$, comparador$_2$, …, comparador$_n$).

  Ejemplo:
  select * from datos where edad ‘<<’ joven and edad ‘>>’ 10 or plato ‘parecido’ china;

- SELECT <ATRIBUTOS> FROM <TABLAS> WHERE <lista_de_predicados_difusos>

  donde:
  <lista_de_predicados_difusos> es (predicado$_1$, predicado$_2$, …, predicado$_n$).

  Ejemplo:
  select * from datos where edad = **joven** and edad = **aprox** or peso = **gordo**;

- SELECT <ATRIBUTOS> FROM <TABLAS> WHERE <lista_de_modificadores_difusos>

  donde:
  <lista_de_modificadores_difusos> es (modificador$_1$, modificador$_2$, …, modificador$_n$).

  Ejemplo:
  select * from datos where edad ‘**muy**’ joven and edad ‘**extremadamente**’ viejo;

- SELECT <ATRIBUTOS> FROM <TABLAS> WHERE <lista_de_conectores_difusos>

  donde:

<lista_de_conectores_difusos> es (conector$_1$, conector$_2$, …, conector$_n$).

Ejemplo:
select * from datos where edad = joven and **not** edad = aprox **or** peso = gordo;

- SELECT ... FROM … WHERE Q<lista_pred_difusos>...

  donde:
  Q es un cuantificador difuso
  <lista_pred_difusos> es:
  (FuzzyPred$_1$,..., FuzzyPred$_n$)

  Ejemplo:
  SELECT * FROM datos
  WHERE lamayoria(edad=joven, talla=grande);

- Para el caso de las vistas difusas

  SELECT <atributos> FROM <nombre_vista>;

  Ejemplo:
  SELECT * FROM repuestos_baratos_vol_bajo;

- Consultas Particionadas Difusas

  SELECT <atributos> FROM <tablas> WHERE <condicion> GROUP BY <atributos_grupo> HAVING <condicion_difusa>;

  donde:
  <atributos_grupo> es: El o los atributos que servirán como criterio de agrupamiento.
  <condición_difusa> es: La condición difusa impuesta sobre los grupos.

  Ejemplo:
  SELECT ps_partkey, avg(ps_supplycost) FROM partsupp GROUP BY ps_partkey having avg(ps_supplycost) = costeable;

- Update Difuso

  UPDATE <tabla> SET <atributo> = [<valor>] WHERE <condicion_difusa>;

Ejemplo:
UPDATE supplier SET s_nationkey = 26 WHERE s_nationkey = 20 AND s_acctbal = bajo;

- Check Difuso

CREATE TABLE <nombre_tabla> AS (nombre_atributo tipo_atributo (CHECK nombre_atributo = condición_difusa));

donde:
<nombre_atributo tipo_atributo> es: El nombre y tipo de atributo creado en la tabla, este puede ser uno o varios atributos dependiendo de las necesidades del usuario
(CHECK nombre_atributo = condición_difusa) es: La regla difusa que se verificara cada vez que se inserta un nuevo valor para el atributo en la tabla.

Ejemplo:
CREATE TABLE pacientes_jovenes (id_paciente integer, nomb_paciente varchar(15), edad integer CHECK(edad = joven));

INSERT INTO pacientes_jovenes VALUES (1,'Pedro',23);
INSERT 0 1
INSERT INTO pacientes_jovenes VALUES (3,'Jose',31);
ERROR: new row for relation "pacientes_jovenes" violates check constraint "pacientes_jovenes_edad_check"

- Para el caso de las consultas difusas en el from, tenemos:

SELECT <ATRIBUTOS> FROM (SELECT <ATRIBUTOSJ> FROM <TABLAS> WHERE <CONDICIÓN DIFUSA INTERNA>) AS ALIAS WHERE <CONDICION DIFUSA EXTERNA>

donde:
<CONDICION DIFUSA INTERNA>, <CONDICION DIFUSA EXTERNA> pueden presentar predicados difusos, comparadores difusos, cuantificadores difusos o grupo by con having difuso.

Ejemplo:
select * from (select * from (select * from datos where lamayoria(edad=joven, talla=grande)) as alias1 where edad ‘<<’ 10) as alias2 where edad=joven;

- Para el caso de consultas que presenten operación conjuntistas difusas, tenemos:

SELECT$_i$ [**INTERSECT**|**EXCEPT**|**UNION**] SELECT$_j$

donde:
SELECT$_k$ presentan condiciones difusas definidas por predicados difusos, comparadores difusos, cuantificadores difusos, modificadores difusos, conectores difusos o group by con having difuso.

Ejemplo:
select * from (select * from (select * from datos where lamayoria(edad=joven, talla=grande)) as alias1 where edad '<<' 10) as alias2 where edad=joven [**INTERSECT**|**EXCEPT**|**UNION**] select * from datos group by atributo$_i$ having(atributo$_i$) = viejo

- Función de Calibración

SELECT <lista de atributos>
FROM <lista de relaciones>
WHERE <condición difusa>
[WITH CALIBRATION <c>]

La cláusula WITH CALIBRATION es opcional, donde $< c >$ es el factor de calibración que permite al usuario determinar un filtro adicional a las tuplas de acuerdo al grado de membresía a la consulta o un número determinado de tuplas, ya sea un número real entre mayor a 0 y menor o igual a 1 ($0 < c \leq 1$) o un número entero positivo mayor a 0, respectivamente.