Extensión del Lenguaje SQL con Nuevas Primitivas para el Descubrimiento de Reglas de Asociación en una Arquitectura Fuertemente Acoplada con un SGBD

Ricardo Timarán-Pereira*§ **Marta Millán****

* Universidad de Nariño, Pasto, Colombia ** Universidad del Valle, Cali, Colombia
§ *e-mail: ritimar@udenar.edu.co*

(Recibido: Octubre 5 de 2005 - Aceptado: Noviembre 29 de 2005)

Resumen

Los actuales sistemas de bases de datos han sido diseñados, principalmente, para dar soporte a aplicaciones de negocios. El éxito del lenguaje de consulta SQL se debe al reducido número de primitivas suficientes para soportar una vasta mayoría de este tipo de aplicaciones. Desafortunadamente, estas primitivas no son suficientes cuando se trata de dar soporte a la emergente familia de nuevas aplicaciones que involucran el descubrimiento de conocimiento en bases de datos. Por esta razón, se propone en este artículo, extender el lenguaje SQL con nuevas primitivas que soporten eficientemente la tarea de minería de datos asociación, al interior de un Sistema Gestor de Bases de Datos(SGBD). Para garantizar la eficiencia de las operaciones de minería de datos, el álgebra relacional se extiende con nuevos operadores algebraicos que se requieren en tareas de Asociación.

Palabras Clave: Descubrimiento de Conocimiento en Bases de Datos, Nuevas Primitivas SQL para Minería de Datos, Nuevos Operadores Relacionales.

Abstract

Current database systems have been designed mainly to support business applications. The success of Structured Query Language SQL has capitalized on a small number of primitives sufficient to support a vast majority of such applications. However, these primitives are not enough to support the emergent family of the new applications dealing with Knowledge Discovery in Databases. In this paper, both the relational algebra and the SQL language are extended with new algebraic operators and primitives, to support efficiently association data mining tasks.

Keywords: Knowledge Discovery in Databases, New SQL Primitives for Data Mining, New Relational Algebraic Operators.

1. Introducción

Los actuales sistemas de bases de datos se han diseñado, principalmente, para soportar aplicaciones de negocios. Una buena parte del éxito del lenguaje de consulta SQL se debe al reducido número de primitivas suficientes para soportar una vasta mayoría de este tipo de aplicaciones. Desafortunadamente, estas primitivas no son suficientes para soportar la emergente familia de nuevas aplicaciones que tratan con el Descubrimiento de Conocimiento en Bases de Datos (DCBD) [ImMa96].

Algunos investigadores [HoSw95], [SuTA98], [ThSa98], [WaIS98], [PSTK99], [RCIC99], [ThCh99], [SuTA00], [YoPK00] han propuesto implementar tareas de minería de datos tales como

asociación, patrones secuenciales y clasificación, entre otras, utilizando las operaciones tradicionales del lenguaje SQL como joins, agrupamiento y agregaciones que implementan los operadores del álgebra relacional [Codd70][Codd72][Codd79]. Sin embargo, su mayor desventaja es el bajo desempeño.

Para solucionar este problema se propone, en este artículo, extender el lenguaje SQL con nuevas primitivas que soporten eficientemente la tarea de minería de datos de asociación, al interior de un Sistema Gestor de Bases de Datos (SGBD) [TiMM03] [TiMi05a] [TiMi05b]. Para garantizar la eficiencia en las operaciones de minería de datos, el álgebra relacional se extiende con nuevos operadores algebraicos que facilitan los procesos más costosos.

En asociación [AgIS93] [AgSr94], el cálculo de los *itemsets* frecuentes, i.e. todos aquellos conjuntos de items cuyo soporte es mayor o igual al soporte mínimo definido por el usuario, determina el rendimiento total del proceso de encontrar reglas de asociación [ChHY96]. En este artículo se proponen nuevos operadores algebraicos para facilitar este proceso.

El resto del artículo está organizado en 5 secciones. En la sección 2, se presentan los trabajos relacionados con las nuevas primitivas propuestas. En la sección 3, se definen los nuevos operadores con los que se extiende el álgebra relacional para soportar la tarea asociación. En la sección 4, las nuevas primitivas con las que se extiende el lenguaje SQL para el conteo eficiente de los *itemsets* frecuentes que se introducen. En la sección 5, se define un nuevo operador SQL para la extracción de reglas y finalmente, en la última sección, se presentan algunas conclusiones del trabajo.

2. Trabajos relacionados

Extender el motor de un SGBD con nuevos operadores y primitivas, es una de las estrategias más importantes para soportar eficientemente el descubrimiento de conocimiento en bases de datos, bajo una arquitectura fuertemente acoplada [Tima01]. Algunas de las más importantes propuestas, que discuten la manera cómo implementar estos sistemas, se introducen en esta sección.

En Meo et al. [MePC96] se propone un modelo unificado para descubrir reglas de asociación, basado en el operador, *MINE RULE*, diseñado como una extensión del lenguaje *SQL*. Se propone también una semántica formal para este operador con base en nuevos operadores y en la extensión del álgebra relacional, que permite transformar una relación con el fin de descubrir reglas de asociación. Se propone también una arquitectura, para soportar el operador *MINE RULE* [MePC98a], implementada en un SGBD objeto-relacional, que a juicio de los autores, es fuertemente acoplada.

Las ventajas de esta propuesta hacen referencia al poder expresivo del operador *MINE RULE*, cuando se trata de formular cualquier problema de reglas de asociación, y a su sintaxis formal. Sin embargo, *MINE RULE* soporta únicamente una tarea de minería de datos, sus operadores algebraicos no conservan la propiedad de cierre del modelo relacional al operar con relaciones con atributos multivaluados, y por lo tanto, *MINE RULE* es aplicable solo en un SGBD objeto-relacional. La arquitectura en la cual se ha implementado no está fuertemente acoplada a un SGBD en su totalidad, debido a que los algoritmos de minería de datos forman parte del *kernel,* un módulo independiente implementado por encima del SGBD. En este módulo, se realiza todo el proceso de descubrimiento de reglas y el SGBD se utiliza únicamente, para almacenar los datos iniciales, las reglas de salida, los resultados intermedios y para realizar ciertas tareas como evaluación de subconsultas y agrupamientos, entre otros [MePC98b].

Por su parte, *Microsoft Corporation*, propone *OLE DB for Data Mining* [Micr00] [NCBF00], un lenguaje de minería de datos, como un esfuerzo hacia la estandarización de primitivas de los lenguajes de minería de datos. *OLE DB i*ncluye las primitivas para la creación y utilización de las tareas de minería de datos de asociación, clasificación, predicción y clustering. La desventaja del lenguaje radica en que no se trata de un sistema fuertemente acoplado, y por lo tanto, las herramientas de

minería de datos están por fuera del SGBD y se acoplan a él a través de una API (*Application Programming Interface*) [BoMa05]. *OLE DB* es un API diseñada, principalmente, para trabajar con *SQL Server* al que se puede conectar herramientas de minería de datos de diferentes proveedores que cumplan con las especificaciones del API *OLE DB for DM* [NCBF00], [NCFB00].

En [Clear et al. 99] se reporta la implementación de un conjunto de primitivas, al interior de NonStop SQL/MX, un SGBD objeto-relacional, paralelo, de la División Tandem de Compaq. El conjunto de primitivas soporta, de manera eficiente y escalable, algunas tareas básicas de descubrimiento de conocimiento, sacando provecho y potencia del motor paralelo. Este tipo de integración se enmarca, por lo tanto, en una solución muy específica, de la que otros motores no paralelos seguramente no pueden sacar estas ventajas.

3. Nuevos operadores del álgebra relacional para asociación

En esta sección se introducen al álgebra relacional nuevos operadores para soportar eficientemente la tarea de Asociación [TiMM03] [TiMi05a] [TiMi05b]:

3.1 Operador Associator (α)

El operador algebraico unario *Associator*(α) es un operador, que a diferencia del operador Selección o Restricción (σ), aumenta la cardinalidad de una relación. *Associator* genera, a partir de cada tupla de una relación, todas las posibles combinaciones de los valores de sus atributos, como tuplas de una nueva relación, conservando el esquema de la relación inicial. Los atributos de la relación inicial pueden pertenecer a diferentes dominios.

Sintáxis:

$$\alpha_{\text{tamaño\_inicial, tamaño\_final}}(R)$$

donde, <*tamano_inicial* > y <*tamaño-final*> son dos parámetros de entrada que determinan el tamaño inicial y final de las combinaciones.

Sea $A=\{A_1, ..., A_n\}$ el conjunto de atributos de la relación R de grado n y cardinalidad m, IS y ES el tamaño inicial y final, respectivamente, de los subconjuntos a calcular. El operador α aplicado a R se define como:

$$\alpha_{IS,\,ES}(R) = \{ \cup_{all}\ X_i \mid X_i \subseteq t_i,\ t_i \in R\ ,\forall_i\ \forall_k\ (X_i = <v_i(A_1), v_i(A_2), null, .., v_i(A_k), .., null_n>,\ v_i(A_k)\ ?\ null),\ (i = (2^n - 1) * m),\ (IS = k = ES),\ A_1 < A_2 < ... < A_k\ ,\ IS = 1,\ ES = n \}$$

El esquema de la relación resultante de $a_{IS,\,ES}(R)$, tiene el mismo grado n de R, su cardinalidad es $m' = (2^n - 1) * m$ y su extensión $r(A)$ está formada por todos los subconjuntos X_i generados a partir de todas las combinaciones posibles de los valores no nulos $v_i(A_k)$ de los atributos de cada tupla t_i de R. En cada tupla X_i, a todos los atributos se les asigna el valor nulo, excepto a aquellos cuyo valor esté entre IS y ES.

A pesar de que generar todas las posibles combinaciones de los valores, para cada tupla de una relación, es un proceso computacionalmente costoso, *Associator* lo hace en una sola pasada sobre la base de datos. De esta manera, *Associator* facilita el cálculo de *itemsets* frecuentes para el descubrimiento de reglas de asociación multidimensionales (i.e. una regla de Asociación es multidimensional si los ítems o atributos de la regla hacen referencia a dos o más criterios o dimensiones [HaKa01]).

Ejemplo 1. Sea la relación $R(A,B,C)$, de la figura 1. La figura 2 muestra el resultado de calcular $\alpha_{2,3}(R)$, para producir las diferentes combinaciones de tamaño 2 hasta tamaño 3.

3.1 Operador Assorow ($\alpha\rho$)

Assorow, al igual que *Associator*, es un operador unario que genera, por cada tupla de la relación R, todos los posibles subconjuntos (*itemsets*), como tuplas de una nueva relación, conservando el esquema de la relación inicial. Se diferencia de *Associator*, en el tipo de relación de entrada R, ya

que todos los atributos de R deben tomar valores en el mismo dominio.

A	B	C
a1	b1	c1
a1	b2	c1

Figura 1. Relación R.

A	B	C
a1	b1	null
a1	null	c1
a1	b1	c1
a1	b2	null
a1	null	c1
a1	b2	c1

Figura 2. $R1=\alpha_{2,3}(R)$

La sintáxis de *Assorow* es la siguiente:

$$\alpha\tilde{n}_{\text{tamaño\_inicial, tamaño\_final}}(R)$$

donde <*tamano_inicial* > y <*tamaño-final*> son dos parámetros de entrada que determinan el tamaño inicial y tamaño final de las combinaciones.

Formalmente, sea $A=\{A_1, \ldots, A_n\}$, $dom(A_1)=dom(A_2)=\ldots=dom(A_n)$, el conjunto de atributos de la relación *R* de grado *n* y cardinalidad *m*, *IS* y *ES* el tamaño inicial y final, respectivamente, de los subconjuntos. El operador $\alpha\tilde{n}$ aplicado a *R* se define como:

$$\alpha\tilde{n}_{IS,\,ES}(R) = \{ \cup_{all}\ X_i \mid X_i \subseteq t_i,\ t_i \in R\ ,\forall_i\ \forall_k\ (X_i = <v_i(A_1),\ldots,v_i(A_k),\ null_{k+1},\ldots,null_n>,\ v_i(A_k)\ ?\ null),\ (i = (2^n - 1) * m),\ (IS = k = .ES),\ A_1 < A_2 < \ldots < A_k\ ,\ IS = 1,\ ES = n\ \}$$

$\alpha\tilde{n}_{IS,\,ES}(R)$ produce una nueva relación con el mismo esquema de *R*, de grado *n* y cardinalidad $m' = (2^n - 1) * m$, y su extensión *r(A)* está formada por todas las tuplas X_i generadas a partir de todas las combinaciones posibles de los valores no nulos $v_i(A_k)$ de los atributos de cada tupla t_i de R. En una tupla X_i los valores de los atributos mayores que el valor que en ese momento tiene *k* (*k=IS..ES*) se hacen nulos.

Ejemplo 2. Sea la relación *R*(*A,B,C*) de la figura 1 y sea *dom*(A)=*dom*(B)=*dom*(C). La figura 3 muestra el resultado calcular R(A,B,C) siendo $R1=\alpha p_{2,3}(R)$.

A	B	C
a1	b1	null
a1	c1	null
a1	b1	c1
a1	b2	null
a1	c1	null
a1	b2	c1

Figura 3. $R1=\alpha p_{2,3}(R)$

3.1 Operador Assocol ($\alpha\rho$)

El operador *Assocol* es un operador unario que requiere que la relación de entrada *R* sea una relación de modelo simple SC [RCIC99] (i.e. una relación cuyo esquema es del tipo *R*(*TID,ITEM*), comúnmente usada en análisis de canasta de mercado), donde *TID* e *ITEM* representan, respectivamente, el identificador de la transacción y el productos adquirido.

Sintáxis:

$$\alpha\hat{e}_{\text{atributo\_identificador, atributo\_valor, tamaño\_inicial, tamaño\_final}}(R)$$

donde, <*atributo_identificador*> es el atributo común para un conjunto de tuplas de la SC-relación, <*atributo_valor*> es el atributo cuyos valores se agrupan por el <*atributo_identificador*> y <*tamano_inicial* >, <*tamaño-final*> son dos parámetros que determinan el tamaño inicial y tamaño final de las combinaciones.

El operador *Assocol* genera como tuplas, por cada grupo de valores iguales que forma el atributo identificador (*<atributo_identificador>)* de la relación SC-relación *R*, todos los posibles subconjuntos (*itemsets*) de diferente tamaño, en una nueva relación de modelo multicolumna MC (i.e.una relación con esquema $R(A_1, A_2, \ldots, A_n)$) [RCIC99].

Sea *A*={*AI, AV*} el conjunto de atributos de la SC-relación *R,* de esquema *R(A),* de grado *n* y cardinalidad *m*, $AI \in A$ el atributo identificador, $AV \in A$ el atributo valor, *IS* y *ES* el tamaño inicial y final respectivamente de los subconjuntos a obtener . El operador $\alpha\hat{e}$ aplicado a *R* se define:

$$\alpha\hat{e}_{AI,AV,\ IS,\ ES}(R)=\{\ \cup_{all}\ X_i \mid \forall_i\ \forall_k\ (X_i =<v_i(AV_1),v_i(AV_2),...,v_i(AV_k),\ null_{k+1},\ldots,null_{ES}>\ ),\ (i = 2^{tg1}+2^{tg2}+..+2^{tgj}\ \ J,\ J=count(distinct(AI)),\ (IS= k=ES),\ \ v_i(AV_1)<v_i(AV_2)<...<v_i(AV_k)\ \}$$

El resultado de $\alpha\hat{e}_{AI,AV,\ IS,\ ES}(R)$ es una MC-relación de esquema $R(Y)$, de grado $n'=ES$, diferente al esquema de R, i.e. $R(Y)\neq R(A)$, donde $Y=\{AV_1, AV_2,\ldots,AV_{ES}\}$ es el conjunto de n' atributos necesarios para formar las combinaciones posibles desde IS hasta ES.

La cardinalidad de la MC- relación es m', $m' = 2^{tg1}+2^{tg2}+..+2^{tgj}\ \ J,$ donde tgj es el número de tuplas en cada grupo de valores comunes del atributo AI y J es el número de distintos valores de AI, $J=count(distinct(AI))$. Su extensión $r(Y)$ está formada por todas las tuplas X_i generadas a partir de todas las combinaciones posibles de los valores $v_i(AV)$ del atributo $AV \in R,$ en cada grupo en el cual el valor de atributo AI es común. En una tupla X_i los valores de los atributos mayores que el valor que en ese momento tiene k ($IS= k=ES$) se hacen nulos.

Ejemplo 3. Sea $R(TID,ITEM)$ la SC-relación de la figura 4. La figura 5 describe las diferentes combinaciones de tamaño 2 hasta tamaño 3, que se producen al aplicar el operador Assocol, es decir, $R1=\alpha\hat{e}_{TID,\ ITEM,\ 2,3}(R)$.

TID	ITEM
100	I1
100	I2
100	I5
200	I2
200	I5
200	I6
300	I5
300	I7

Figura 4. Relación R

ITEM1	ITEM2	ITEM3
i1	i2	null
i1	i5	null
i2	i5	null
i1	i2	i5
i2	i5	null
i2	i6	null
i5	i6	null
i2	i5	i6
i5	i7	Null

Figura 4. $R1=\alpha\hat{e}_{TID,ITEM,2,3}(R)$

3.1 Operador EquiKeep (χ)

EquiKeep (χ) es un operador unario, que se asemeja a la *Selección* o *Restricción* por tener una expresión lógica que evaluar sobre una relación R, y por conservar el esquema. Se diferencia de la Restricción (s) en que en lugar de aplicar la condición a las filas (tuplas) de la relación, *EquiKeep* la aplica a las columnas (atributos) de R. Es decir, restringe los valores de los atributos de cada una de las tuplas de la relación R, a únicamente aquellos que satisfacen una condición determinada, haciendo nulo el resto de valores y conserva el esquema de la relación.

Sintáxis:

$$\chi_{expresión\_lógica}(R)$$

donde *<expresión lógica>* es la condición que deben cumplir los valores de los atributos de la relación R para no hacerse nulos.

El operador *EquiKeep*, restringe los valores de los atributos de cada una de las tuplas de la relación R a únicamente los valores de aquellos atributos que satisfacen una expresión lógica *<expresión_lógica>*. Una expresión lógica se construye a partir de un conjunto de cláusulas de la forma *Atributo=Valor* y conectivos lógicos *AND, OR y NOT*. En cada tupla, los valores de los atributos que no cumplen la *<expresión_lógica>* se hacen nulos. *EquiKeep* elimina las tuplas vacías, es decir, aquellas tuplas en las cuales los valores de todos sus atributos son nulos.

Formalmente, sea $A= \{A_1, ..., A_n\}$ el conjunto de atributos de la relación R de esquema $R(A)$, de grado n y cardinalidad m. Sea p una expresión lógica integrada por cláusulas de la forma $A_i = const,$ donde $const$ y por los operadores booleanos AND (^), OR (v), NOT (¬). El operador χ aplicado a la relación R con la expresión lógica $p,$ produce:

$$\chi_p(R)=\{\ t_i(A) \mid \forall i \forall j\ ((p(v_i(A_j))= v_i(A_j)\ si\ p =true)\ o\ (\ p(v_i(A_j))= null\ si\ p =false)),\ 1= i=m', 1= j=n,\ \ m' \leq m\}$$

$\chi_p(R)$ genera una relación de igual esquema $R(A)$ de

grado n y cardinalidad m', donde $m' \leq m$. En su extensión, cada n-tupla ti, está formada por los valores de los atributos de R, $v_i(A_j)$, que cumplan la expresión lógica p, cuando $p(v_i(Y_j))$ es verdadero, y por valores nulos, si $p(v_i(Y_j))$ es falso.

Ejemplo 4. Sea la relación $R(A,B,C,D)$ de la figura 6. Restringir los valores de los atributos A=a1, B=b1, C=c2 y D=d1, es decir, $R1=\chi_{A=a1 \vee B=b1 \vee C=c2 \vee D=d1}(R)$ produce como resultado la figura 7.

Note en la figura 7, que la tupla {*a2,b2,c1,d2*} de la relación R se elimina porque todos sus valores son nulos.

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b1	c1	d1
a2	b2	c1	d2
a1	b2	c2	d1

Figura 6. Relación R

A	B	C	D
a1	b1	null	d1
a1	null	null	null
null	null	c2	null
null	b1	null	d1
null	null	null	null
a1	null	c2	d1

Figura 7. $R1=\chi_{A=a1 \vee B=b1 \vee C=c2 \vee D=d1}(R)$

3.1 Operador Describe Associator ($\beta\alpha$)

Describe Associator es un operador unario que toma como entrada la relación resultante de los operadores *associator*, *assorow* o *assocol*. Por cada tupla de esta relación, *Describe Associator* genera, a partir de los atributos *l* no nulos de la tupla, todos los diferentes subconjuntos de un tamaño específico de la forma {{a}, {l-a},s}, donde {a} se denomina subconjunto antecedente y {l-a} subconjunto consecuente. {a}y {l-a}, son subconjuntos de atributos de *l* y *s* es el tamaño del subconjunto antecedente {a}.

La sintáxis del operador *Describe Associator* es la siguiente:

$$\beta\alpha_{longitud\_regla}(R)$$

donde <*longitud_regla*> es la longitud máxima de atributos no nulos por tupla.

Formalmente, sea $A=\{A_1, ..., A_n\}$ el conjunto de atributos de la relación *R* de grado *n* y cardinalidad *m*, *LR* es el tamaño de los subconjuntos a obtener . El operador $\beta\alpha$ aplicado a R se define como:

$$\beta\alpha_{LR}(R) = \{ \cup_{all} X_i(Y) \mid Y=\{Y_1,Y_2,..Y_{LR},S\}, X_i \subseteq t_i, t_i \in R, \forall_i \ (X_i=<v_i(A_1),..,v_i(A_k), s>, v_i(A_k)\neq null), (i = (2^n -2) * m), LR = n \}$$

$\beta\alpha_{LR}(R)$ produce una nueva relación de grado $LR+1$ y cardinalidad $i = (2^n -2)$, con esquema $R(Y)$, $Y=\{Y_1, Y_2,..Y_{LR}, S\}$ donde *LR* es el tamaño de los subconjuntos a obtener y S la longitud del subconjunto antecedente. Su extensión $r(Y)$, está formada por todos los subconjuntos X_i, generados a partir de todas las combinaciones posibles de los valores no nulos $v_i(A_k)$ de los atributos de cada tupla t_i de R. Cada tupla X_i se forma siguiendo la regla ({a}=> {l-a},s), donde *l* es el conjunto de atributos no nulos de la tupla *t*, {*a*} es un subconjunto de *l* denominado *antecedente*, {l-a}se denomina subconjunto *consecuente* y *s* es el tamaño del subconjunto {*a*}.

Describe Associator facilita la generación de reglas de asociación tanto unidimensionales como multidimensionales [HaKa01].

Ejemplo 5. Sea la relación *R*(*A,B,C*) de la figura 8. El resultado de calcular $R1= \beta\alpha_3(R)$. se muestra en la figura 9.

A	B	C
a1	b1	c1
a2	b2	c2

Figura 8. Relación R

A1	A2	A3	S
a1	b1	c1	1
b1	a1	c1	1
c1	a1	b1	1
a1	b1	c1	2
a1	c1	b1	2
b1	c1	a1	2
a2	b2	c2	1
b2	a2	c2	1
c2	a2	b2	1
a2	b2	c2	2
a2	c2	b2	2
c2	b2	a2	2

Figura 9. Relación R1=$\beta\alpha_3$(R).

4. Nuevas primitivas SQL para asociación

Los nuevos operadores algebraicos *Associator, Assorow, Assocol, EquiKeep* extienden el álgebra relacional para soportar la tarea de Asociación. Para que el lenguaje de consultas SQL pueda soportar eficientemente el descubrimiento de reglas de Asociación, es necesario implementar estos operadores como nuevas primitivas SQL.

4.1 Primitiva Associator Range

Esta primitiva implementa el operador algebraico *Associator* en la cláusula SQL SELECT. *Associator* permite obtener por cada tupla de una tabla, todos los posibles subconjuntos desde un tamaño inicial hasta un tamaño final determinado en la cláusula RANGE [TiMM03] [TiMi05a] [TiMi05b].

La sintáxis de la primitiva *Associator Range* dentro de la cláusula SELECT es:

SELECT <*ListaAtributosTablaDatos*> [INTO <*NombreTablaAssociator*>]
FROM <*NombreTablaDatos*>
WHERE <*CláusulaWhere*>
ASSOCIATOR RANGE <*valor1*> UNTIL <*valor2*>
GROUP BY <*ListaAtributosTablaDatos*>

La primitiva *ASSOCIATOR RANGE* facilita el cálculo de los *itemsets* frecuentes para el descubrimiento de Reglas de Asociación en tablas multicolumna [RCIC99].

Ejemplo 6. Obtener, de la tabla ESTUDIANTES (PROGRAMA, EDAD, SEXO, ESTRATO, PROMEDIO), los *itemsets* frecuentes de tamaño 2 y 3, formados por lo atributos *programa, sexo, estrato* que cumplan con un soporte mínimo mayor o igual a 2 y almacenarlos en la tabla ASSOSTUDENT:

SELECT *programa,sexo, estrato, count(*)* AS *soporte*
INTO *assostudent*
FROM *estudiantes*
ASSOCIATOR RANGE *2* UNTIL *3*
GROUP BY *programa,sexo,estrato* HAVING *count(*)*>=2

4.2 Primitiva Assorow Range

Esta primitiva implementa el operador algebraico *Assorow* en la cláusula SELECT. *Assorow* permite obtener, por cada tupla de una tabla, todos los posibles subconjuntos desde un tamaño inicial hasta un tamaño final determinado en la cláusula RANGE siempre y cuando todos los atributos de la tabla de datos sean del mismo tipo. Dentro de la cláusula SELECT, *Assorow* tiene una sintáxis equivalente a Associator.

La primitiva *Assorow* facilita el cálculo de los *itemsets* frecuentes para el descubrimiento de Reglas de Asociación en tablas multicolumna [RCIC99] especialmente en análisis de canasta de mercado.

4.3 Primitiva Assocolgroup Range

Esta primitiva implementa el operador algebraico *Assocol* en la cláusula SELECT. *AssocolGroup* obtiene a partir de una tabla de columna simple [RCIC99] una tabla multicolumna con todos los posibles *itemsets* desde un tamaño inicial hasta un tamaño final determinado por la cláusula RANGE, como tuplas.

La siguiente sintáxis de *AssocolGroup* dentro de la cláusula SELECT tiene:

SELECT <*ListaAtributosTablaDatos*> INTO <*NombreTablaAssocol*>
FROM <*NombreTablaDatos*> WHERE <*CláusulaWhere*>
ASSOCOLGROUP <*AtributoIdentificador*> REPLACE <*AtributoDatos*> WITH
<*ListaAtributosTabla Assocol*> RANGE <*valor1*> UNTIL <*valor2*>

Ejemplo 7. Sea la tabla de modelo de columna simple TRANSACCION (TID, ITEM). Calcular los *itemsets* frecuentes de tamaño 2 y 3 que cumplan con un soporte mínimo de 2 y almacenar los resultados en

la tabla ASSOTRANCOL.

Para obtener esta consulta se necesitan las siguientes dos sentencias SQL:

```
SELECT tid, item INTO tempassotrancol
FROM transaccion
ASSOCOLGROUP tid REPLACE item WITH
item1,item2,item3 RANGE 2 TO 3

SELECT item1,item2,item3, count() AS soporte INTO
assotrancol
FROM tempassotrancol
GROUP BY item1,item2,item3 HAVING count(*)>=2
```

4.4 Primitiva EquiKeep On

Esta primitiva implementa el operador algebraico *EquiKeep* en la cláusula SQL SELECT. *EquiKeep On* conserva, en cada registro de una tabla, los valores de los atributos que cumplen una condición determinada. El resto de valores de los atributos se hacen nulos.

La sintáxis de *EquiKeep On* dentro de la cláusula SELECT es:

```
SELECT <ListaAtributosTablaDatos> [INTO
<NombreTablaEquiKeep>]
FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
EQUIKEEP ON <CondiciónValoresAtributos>
```

La primitiva *Equikeep On* facilita la generación de los *itemsets* frecuentes en el descubrimiento de Reglas de Asociación, al permitir conservar, en cada registro de una tabla, únicamente los valores de los atributos frecuentes o *itemsets* frecuentes. Se puede utilizar en el algoritmo *Apriori* [AgSr94*], en el *FP-Tree* [HaPe00*], [HaPY00*] o conjuntamente con cualquiera de la primitivas de Asociación (*Associator, Assorow, Assocol*).

Ejemplo 8. Sea la tabla *TRANSACCION* (*TID, ID_ITEM1, ID_ITEM2, ID_ITEM3*). Encontrar los *itemsets* frecuentes de tamaño 2 y 3 formados por lo atributos *id_item1, id_item2, id_item3 ,* que cumplan un soporte mínimo mayor o igual a 3. Se supone que los *itemsets* frecuentes de tamaño 1 son : {i2:4}, {i3:4}, {i4:3}. Almacenar los resultados en la tabla EQUITRAN.

La sentencia SQL que permite obtener esta consulta utilizando las primitivas *EquiKeep On* y *Assorow Range* es la siguiente:

```
SELECT id_item1, id_item2, id_item3, count(*) AS soporte
INTO equitran
FROM transaccion
EQUIKEEP ON id_item1 IN (i2,i3,i4) OR id_item2 IN
(i2,i3,i4) OR id_item3 IN (i2,i3,i4)
ASSOROW RANGE 2 UNTIL 3
GROUP BY id_item1, id_item2,id_item3 HAVING
count(*)>=3
```

5. Nuevo operador SQL para asociación

El lenguaje SQL se ha extendido con nuevas primitivas para soportar la tarea de Asociación al interior de un SGBD. Estas primitivas facilitan el cálculo de los *itemsets* frecuentes. Ahora, es necesario unificar estas primitivas en un nuevo operador SQL *Describe Association Rules* que permita extraer reglas de Asociación que cumplan con una confianza mínima especificada por el usuario

El operador *Describe Association Rules*, implementa el operador algebraico *Describe Associator* en una nueva cláusula SQL. *Describe Association Rules* puede generar reglas de asociación unidimensionales o multidimensionales de una longitud específica, una vez se hayan calculado los *itemsets* frecuentes.

Describe Association Rules tiene la siguiente sintáxis:

```
DESCRIBE [UNIDIMENSIONAL][MULTIDIMENSIONAL]
                ASSOCIATION RULES
        FROM <NombreTablaItemsetsFrecuentes> [INTO
                <TablaReglasAsociación>]
            WITH CONFIDENCE <valor1>
                LENGTH <valor2 >
        [AS <SubconsultaItemsetsFrecuentes>]

                <valor 1> ::=1,2,3,4,….
                <valor 2> ::=1,2,3,4,….
< SubconsultaItemsetsFrecuentes >::=<SELECT FROM WHERE
                                    EQUIKEEP ASSOROW
                                    GROUP BY> |
                                   <SELECT FROM WHERE
                                    EQUIKEEP ASSOCIATOR
                                    GROUP BY>
```

6. Conclusiones

Para extender el lenguaje SQL con nuevas primitivas y operadores que soporten eficientemente cualquier

tarea de minería de datos al interior de un SGBD, se recomienda seguir el método propuesto para Asociación. Inicialmente, se debe extender el álgebra relacional con nuevos operadores que faciliten los procesos computacionalmente más costosos de la tarea. Luego, se deben definir las nuevas primitivas SQL que implementen estos nuevos operadores algebraicos en la cláusula *SQL SELECT* y finalmente, se deben unificar estas nuevas primitivas SQL en un nuevo operador SQL que realice la tarea de minería de datos.

Actualmente, estas primitivas se han implementado al interior del SGBD PostgreSQL [Timarán et al 05]. Este proceso se encuentra en la etapa de pruebas y análisis de desempeño.

El contar con nuevos operadores algebraicos para minería de datos, facilita la definición de nuevas estrategias de optimización que le permitan al SGBD ejecutar eficientemente consultas que involucren descubrimiento de conocimiento.

7. Referencias bibliográficas

[AgIS93] Agrawal R., Imielinski T., Swami A., Mining Association Rules between Sets of Items in Large Databases, ACM SIGMOD, Washington DC, USA, 1993.

[AgSr94] Agrawal R., Srikant R., Fast Algorithms for Mining Association Rules, VLDB Conference, Santiago, Chile, 1994.

[BoMa05] Boulicaut J-F., Masson C., Data Mining Query Languages, Data Mining and Knowledge Discovery Handbook : A Complete Guide for Practitioners and Researchers, O. Maimon and L. Rokach (Eds), Kluwer Academic Publishers, 14 pages, 2005.

[ChHY96] Chen M., Han J., Yu P., Data Mining: An Overview from Database Perspective, IEEE Transactions on Knowledge and Data Engineering, 1996.

[Clear et al. 99] Clear, J., Dunn, D., Harvey, B., Heytens, M., Lohman, P., Mehta, A., Melton, M., Rohrberg, L., Savasere, A., Wehrmeister, R., Xu, M., NonStop SQL/MX Primitives for Knowledge Discovery, KDD-99, San Diego, USA, 1999.

[Codd70] Codd, E., F., A Relational Model of Data for Large Shared Data Banks, CACM, Vol. 13, No. 6, June, 1970.

[Codd72] Codd, E., F., Relational Completeness of Data Base Sublanguages, Courant Computer Science Symposium 6, Data base Systems, Englewood Cliffs, NJ:Prentice Hall, 1972.

[Codd79] Codd, E.,F., Extending the Database Relational Model to Capture More Meaning, ACM TODS, Vol. 4, No. 4, December, 1979.

[DePi99] De Miguel A., Piattini, M., Fundamentos y modelos de bases de datos, 2ª edición, Alfaomega Grupo Editor, 1999.

[HaKa01] Han, J., Kamber, M., Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, 2001.

[HoSw95] Houtsma, M., Swami, A., Set Oriented Mining for Association Rules in Relational Databases, ICDE, 1995.

[ImMa96] Imielinski, T., Mannila, H., A Database Perspective on Knowledge Discovery, Communications of the ACM, Vol 39, No. 11, November, 1996.

[MePC96] Meo R., Psaila G., Ceri S., A New SQL-like Operator for Mining Association Rules, VLDB Conference, Bombay, India, 1996.

[MePC98a] Meo R., Psaila G., Ceri S., An Extension to SQL for Mining Association Rules, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 2, pp .195-224, Boston, 1998.

[MePC98b] Meo R., Psaila G., Ceri S., A Tightly-Coupled Architecture for Data Mining, 14th. International Conference on Data Engineering ICDE98, 1998.

[Micr00] Microsoft Corporation, OLE DB for Data Mining Draft Specification, version 0.9, in http://www.microsoft.com/data/oledb/dm.html, 2000.

[NCBF00] Netz A., Chaudhuri S., Bernhardt J., Fayyad U., Integration of Data Mining and Relational Databases, Proceedings of the 26thInternational Conference on Very Large Databases, Cairo, Egypt, 2000.

[NCFB00] Netz A., Chaudhuri S., Fayyad U.,Bernhardt J., Integrating Data Mining with SQL Databases: OLE DB for Data Mining, technical report,2000.

[PSTK99] Pramudiono, I., Shintani, T., Tamura, T., Kitsuregawa, M., Parallel SQL Based Association Rule Mining on Large Scale PC Cluster: Performance Comparison with Directly Coded C Implementation, In Proc. Of Third Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD, 1999.

[RCIC99] Rajamani, K., Cox, A., Iyer, B., Chadha, A., Efficient Mining for Association Rules with Relational Database Systems, International Database Engineering and Aplication Symposium, p. 148-155, 1999.

[SuTA98] Sarawagi S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, ACM SIGMOD, 1998.

[SuTA00] Sarawagi S., Thomas S., Agrawal R., Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, Vol 4, 2000.

[Tima01] Timarán,R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: un Estado del Arte, Ingeniería y Competitividad, Volumen 3, No.2, Cali, Colombia, 2001.

[TiMM03] Timarán, R., Millán, M., Machuca, F., New Algebraic Operators and SQL Primitives for Mining Association Rules, in Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence (NCI 2003), International Association of Science and Technology for Development, Cancun, Mexico, mayo 2003.

[TiMi05a] Timarán, R., Millán, M., EquipAsso: An algorithm based on New Relational Algebraic Operators for Association Rules Discovery, in Proceedings of The Fourth IASTED International Conference on Computational Intelligence (CI 2005), International Association of Science and Technology for Development, Calgary, Canada, julio 2005.

[TiMi05b] Timarán, R., Millán, M., EquipAsso: Un algoritmo para el Descubrimiento de reglas de Asociación basado en Operadores Algebraicos, en Proceedings de la Cuarta Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2005), Orlando, Florida, USA, julio 2005.

[Timarán et al.05] Timarán, R., Guerrero, M., Diaz M., Cerquera, C., Armero, S., Implementación de Primitivas SQL para reglas de Asociación en una Arquitectura Fuertemente Acoplada, en Proceedings de la XXXI Conferencia Latinoamericana de Informatica (CLEI 2005), Santiago de Cali, Colombia, octubre 2005.

[ThCh99] Thomas, S., Chakravarthy, S., Performance Evaluation and Optimization of Join Queries for Association Rule Mining, in Proc. Of First International Conference on Data Warehousing and Knowledge Discovery , DAWAK, 1999.

[ThSa98] Thomas, S., Sarawgi, S., Mining Generalized Association Rules and Sequential Patterns Using SQL Queries, in Fourth International Conference on Knowledge Discovery and Data Mining, KDD, 1998.

[YoPK00] Yoshizawa, T., Pramudiono, I., Kitsuregawa, M, SQL Based Association Rule Mining using Commercial RDBMS (IBM DB2 UDB EEE), Data Warehousing and Knowledge Discovery, pag. 301-306, 2000.

[WaIS98] Wang, M., Iyer, B., Scott, V.,J., Scalable Mining for Classification Rules in Relational Databases, International Database Engineering and Application Symposium, pages 58-67, 1998.