EquipAsso: un algoritmo para el descubrimiento de conjuntos de ítems frecuentes sin generación de candidatos*

[EquipAsso: an algorithm for discovery large itemsets without candidate generation]

Ricardo TIMARÁN-PEREIRA[1]

***Recibo:** 20.08.2011 **- Ajuste:** 20.09.2011 **- Ajuste:** 25.11.2011 **- Aprobación:** 10.12.2011*

Resumen

Con las grandes cantidades de datos continuamente recolectadas y almacenadas en las bases de datos, el problema de derivar asociaciones a partir de los datos ha sido siempre un foco de investigación en Minería de Datos. Todo el rendimiento de generar reglas de asociación es determinado por el descubrimiento de conjuntos de ítems frecuentes. Un conjunto de ítems es frecuente si su soporte excede un predeterminado soporte mínimo. La gran mayoría de algoritmos propuestos para reglas de Asociación (Apriori, AprioriTid, AprioriHibrido, DHP y DIC entre otros), se basan en la generación de conjuntos de ítems candidatos, para el cálculo de los conjuntos frecuentes. En este artículo se propone el algoritmo EquipAsso. Este algoritmo encuentra directamente los conjuntos de ítems frecuentes sin la generación de conjuntos candidatos, utilizando los operadores algebraicos relacionales EquiKeep y Associator.

Palabras Clave*: Minería de Datos, Reglas de Asociación, Conjunto de Ítems Frecuentes, Operadores Algebraicos.*

* Modelo para citación de este reporte de caso
TIMARÁN PEREIRA, Ricardo (2011). EquipAsso: Un algoritmo para el descubrimiento de conjuntos de ítems frecuentes sin generación de candidatos. En: Ventana Informática. No. 25 (jul. – dic., 2011). Manizales (Colombia): Facultad de Ciencias e Ingeniería, Universidad de Manizales. p. 63-82. ISSN: 0123-9678

1 PhD. en Ingeniería; MSc. en Ingeniería; Especialista en Multimedia; Ingeniero de Sistemas y Computación. Director grupo de investigación GRIAS, Profesor Asociado, Departamento de Sistemas, Facultad de Ingeniería, Universidad de Nariño, Pasto (Colombia).Correo electrónico: ritimar@udenar.edu.co

Abstract

With the huge amounts of continually gathered data and stored in the databases, the problem of deriving associations from the data has been always a research focus in data mining. The overall performance of mining association rules is determined by the discover the large itemsets. An ítemset is large if has his support above a pre-determined minimum support. The most proposed algorithms for association rules (Apriori, AprioriTid, AprioriHibrido, DHP, DIC and others), are based on the generation of candidate itemsets for the calculation of large itemsets. In this paper, the algorithm EquipAsso is proposed. This algorithm finds the large itemsets directly without the candidate generation, using the relational algebraic operators Associator and EquiKeep.

Keywords: *Data Mining, Association Rules, Large Itemsets, Algebraic Operators.*

Introducción

La minería de datos es una de las etapas del proceso de Descubrimiento de Conocimiento en Bases de Datos -DCBD que permite descubrir información oculta y útil para la toma de decisiones entre grandes volúmenes de datos, según Cabena et al. (1997), Fayyad et al. (1996a) y Han y Kamber (2001). La tarea de Reglas de Asociación ha sido siempre un centro de gran atención, importancia e investigación en el área de minería de datos, afirman Fayyad (1996b) y Hernández, Ramírez y Ferri (2004). Esta fue formulada por Agrawal, Imielinski y Swami (1993) y a menudo se referencia como el problema de canasta de mercado (*market-basket*). La entrada del problema consiste de un conjunto de ítems y una colección de transacciones que son subconjuntos de ellos. La tarea es encontrar relaciones entre los ítems de esas transacciones. La formalización de este problema es encontrar reglas de asociación que cumplan unas especificaciones mínimas dadas por el usuario, expresadas en forma de soporte y confianza. Un conjunto de ítems, se denomina frecuente si su soporte excede un umbral dado.

Todo el rendimiento de generar reglas de asociación es determinado por el cálculo de los conjuntos de ítems frecuentes. Una vez identificados los conjuntos frecuentes, las correspondientes reglas de asociación pueden ser derivadas de una manera directa, de acuerdo con Agrawal, Imielinski y Swami (1993) y Savasere, Omiecinski y Navathe (1995). La investigación en este campo se ha centrado en diseñar algoritmos eficientes para encontrar los conjuntos frecuentes: *Apriori*

(Agrawal y Srikant, 1994), *Standing for Direct Hashing and Pruning DHP* (Park, Chen y Yu, 1995), *Dynamic* Ítems*et Counting DIC* (Brin et al., 1997), *FP-growth* (Han y Pei (2000) y Han, Pei y Yin (2000)), entre otros.

Para el cálculo de los *itemsets* frecuentes, los algoritmos *Apriori*, *DHP* y *DIC* se basan en la generación de *itemsets* candidatos. El algoritmo *FP-growth* encuentra los *itemsets* frecuentes sin generar conjuntos de ítems candidatos. El algoritmo *FP-growth* construye un árbol denominado *FP-tree* para calcular los *itemsets* frecuentes.

En este artículo se propone el algoritmo *EquipAsso*, algoritmo que encuentra directamente los conjuntos de ítems frecuentes sin la generación de conjuntos candidatos, utilizando los operadores algebraicos relacionales *EquiKeep* y *Associator*, presentados por Timarán, Millán y Machuca (2003).

Para cada tupla de una relación, el operador algebraico *Associator* genera todas las posibles combinaciones (conjuntos de ítems) de los valores de los atributos de esa tupla, de tamaños diferentes, determinados por dos parámetros: tamaño inicial y tamaño final. 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.

El operador algebraico *EquiKeep*, restringe los valores de los atributos de cada una de las tuplas de una relación a, únicamente, aquellos que satisfacen una expresión lógica. El resto de los valores se vuelven nulos. Una vez calculados los conjuntos de ítems frecuentes de tamaño uno, *EquipAsso* utiliza el operador *EquiKeep* para dejar, en cada registro de la relación, únicamente éstos conjuntos de ítems. Posteriormente, utiliza el operador *Associator* para generar, por cada tupla de la relación, todos los conjuntos de ítems posibles de tamaño 2 hasta máximo el grado de la relación. Estos conjuntos se agrupan y se cuentan, restringiendo la relación a únicamente aquellos que cumplan un soporte mínimo. De esta manera se calculan todos los conjuntos de ítems frecuentes.

La ventaja del algoritmo con respecto a los demás, es que al basarse en nuevos operadores algebraicos relacionales facilita su integración al interior de cualquier SGBD, acoplando la tarea de *Asociación* de una manera fuerte, según Timarán (2001). Debido a que es ejecutado conjuntamente con los datos en el SGBD, tiene la ventaja potencial de resolver los problemas de escalabilidad y rendimiento de las arquitecturas débilmente acopladas y facilita la aplicación de técnicas de optimización de consultas, de acuerdo con Sarawagi, Thomas y Agrawal (1998) y (2000). Al implementarse *EquipAsso* con nuevas primitivas

SQL permite la realización de consultas de minería de datos *ad hoc* y el desarrollo de aplicaciones en esta área.

El resto del artículo está organizado en secciones. En la sección 1, se da el marco teórico necesario para la implementación del algoritmo *EquipAsso*. En la sección 2, se describe dicho algoritmo y su implementación en el lenguaje SQL. En la sección 3, se muestran las pruebas de rendimiento del algoritmo *EquipAsso* con respecto a los algoritmos *Apriori* y *FP-growth*. Finalmente, en la sección 4, se presentan las conclusiones y futuros trabajos.

1. Conceptos preliminares

1.1 Reglas de asociación

Las asociaciones persiguen patrones en los que la presencia de algo implica la presencia de algo más. Dado una base de datos de ventas, se desea descubrir las asociaciones importantes entre ítems tal que la presencia de algún ítem en una transacción implicara la presencia de otros ítems en la misma transacción. El modelo matemático fue propuesto por Agrawal et al. (1993), para afrontar el problema de encontrar reglas de asociación.

Formalmente, sea $I = \{i_1, i_2, ..., i_m\}$ un conjunto de literales, llamados ítems. Sea D un conjunto de transacciones, donde cada transacción T es un conjunto de ítems tal que $T \subseteq I$. Las cantidades de ítems comprados en una transacción no son considerados, lo que significa que cada ítem es una variable binaria representando si un ítem fue comprado o no. Cada transacción se asocia con un identificador, llamado *TID*. Sea X un conjunto de ítems. Se dice que una transacción T contiene a X si y solo si $X \subseteq T$. Una regla de asociación es una implicación de la forma $X \Rightarrow Y$, donde X y Y son conjuntos de ítems tal que $X \subseteq I$, $Y \subseteq I$ y $X \cap Y = \phi$. El significado intuitivo de tal regla es que las transacciones de la base de datos que contienen X tienden a contener Y. La regla $X \Rightarrow Y$ se cumple en el conjunto de transacciones D con una confianza c si el c% de las transacciones en D que contienen X también contienen Y. La regla $X \Rightarrow Y$ tiene un soporte s en el conjunto de transacciones D si el s% de las transacciones en D contienen $X \cup Y$.

La confianza denota la fuerza de la implicación y el soporte indica la frecuencia de ocurrencia de los patrones en la regla. Es a menudo deseable poner atención solamente a esas reglas que pueden tener razonablemente un soporte largo. Tales reglas con una confianza alta y soporte fuerte son referidas como reglas fuertes (*strong rules*) por Agrawal et al. (1993). La tarea de encontrar reglas de asociación es

esencialmente para descubrir reglas de asociaciones fuertes en grandes bases de datos. Agrawal et al. (1993) y Agrawal y Srikant (1994), consideran que el problema de minar reglas de asociación es encontrar todas las reglas de asociación que satisfagan un soporte mínimo especificado por el usuario y una mínima restricción de confianza. El problema se descompone en los siguientes pasos:

- Descubrir los conjuntos de ítems frecuentes, por ejemplo, el conjunto de ítems que tienen el soporte de transacciones por encima de un predeterminado soporte *s* mínimo.
- Usar los conjuntos de ítems frecuentes para generar las reglas de asociación para la base de datos.

Después de que los conjuntos de ítems frecuentes son identificados, las correspondientes reglas de asociación se pueden derivar de una manera directa.

Un uso clásico de asociaciones es el análisis de la canasta de mercado, en donde la asociación es una lista de afinidades de productos. Por ejemplo, observar las compras de los clientes en un supermercado, puede generar una regla: *'el 30% de las transacciones que contienen cerveza también contienen pañales; el 2% de todas las transacciones contienen a ambos ítems'*. Aquí el 30% es la confianza de la regla y el 2%, el soporte de la regla, según Agrawal et al. (1996).

Otras aplicaciones de reglas de asociación son los análisis de demandas médicas para determinar procedimientos médicos que se realizan ya sea al mismo tiempo o a lo largo de un periodo de tiempo para un diagnóstico en particular. También se aplican para el análisis de textos, diseño de catálogos, segmentación de clientes basado en patrones de compra, en mercadeo, entre otros.

1.2 Operadores algebraicos para asociación

Para lograr eficiencia en las operaciones de minería de datos, un nuevo operador algebraico debe facilitar los procesos de minería de datos computacionalmente más costosos. Los operadores algebraicos propuestos por Timarán, Millán y Machuca (2003), con los que se extiende el álgebra relacional facilitan este proceso y los cuales se describen a continuación.

1.2.1 Operador Asociator (α). *Associator* es un operador algebraico unario 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, acorde con Timarán, Millán y Machuca (2003). Su sintaxis es la siguiente:

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

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

El operador *Associator* genera, por cada tupla de la relación R, todos sus posibles subconjuntos (conjuntos de ítems) de diferente tamaño. *Associator* toma cada tupla *t* de *R* y dos parámetros: <*tamaño_inicial*> y <*tamaño_final*> como entrada, y retorna, por cada tupla *t*, las diferentes combinaciones de atributos X_i, de tamaño <*tamaño_inicial*> hasta tamaño <*tamaño_final*>, como tuplas en una nueva relación. El orden de los atributos en el esquema de *R* determina los atributos en los subconjuntos con valores, el resto se hacen nulos. El tamaño máximo de un conjunto de ítems y por consiguiente el tamaño final máximo (<*tamaño_final*>) que se puede tomar como entrada es el correspondiente al valor del grado de la relación.

Formalmente, 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*

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

produce una nueva relación cuyo esquema *R(A)* es el mismo de *R* de grado *n* y cardinalidad $m'\leq(2n-1) * m$ y cuya 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 únicamente un grupo de atributos mayor o igual que *IS* y menor o igual que *ES* tienen valores, los demás atributos se hacen nulos.

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. *Associator* facilita el cálculo de conjuntos de ítems frecuentes para el descubrimiento de reglas de asociación multidimensionales, según Han y Kamber (2001).

Ejemplo 1. Sea la relación *R(A, B, C, D)* de la tabla 1. Encontrar las diferentes combinaciones de tamaño 2 hasta tamaño 4, es decir $R1=\alpha_{2,4}(R)$.

Tabla 1. Relación R

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

El resultado del operador *Associator R1=$\alpha_{2,4}$(R)* se muestra en la tabla 2.

Tabla 2. Resultado operación R1=$\alpha_{2,4}$(R)

A	B	C	D
a1	b1	null	null
a1	null	c1	null
a1	null	null	d1
null	b1	c1	null
null	b1	null	d1
null	null	c1	d1
a1	b1	c1	null
a1	b1	null	d1
a1	null	c1	d1
null	b1	c1	d1
a1	b1	c1	d1
a1	b2	null	null
a1	null	c1	null
a1	null	null	d2
null	b2	c1	null
null	b2	null	d2
null	null	c1	d2
a1	b2	c1	null
a1	b2	null	d2
a1	null	c1	d2
null	b2	c1	d2
a1	b2	c1	d2

1.2.2 Operador EquiKeep (χ). *EquiKeep* es un operador unario, que se asemeja a la Selección o Restricción por tener una expresión lógica que evaluar sobre una relación *R*, y conserva su esquema. Se diferencia de la Restricción (σ) en que en lugar de aplicar la condición a las filas (tuplas) de la relación, *EquiKeep* aplica la expresión lógica 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 nulos al resto de valores y conserva el esquema de la relación, según Timarán, Millán y Machuca (2003). Su sintaxis es la siguiente:

$$\chi_{expresión\_lógica}(R)$$

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 los atributos que satisfacen una expresión lógica <*expresión_lógica*>, la cual está formada por un conjunto de cláusulas de la forma *Atributo=Valor*, conectivos lógicos *AND, OR y NOT*. En cada tupla, los valores de los atributos que no cumplen la condición <*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,* unidas por los operadores booleanos *AND (^), OR (v), NOT (¬).* El operador c aplicado a la relación *R* con la expresión lógica *p*:

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

produce 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*, es decir $p(v_i(Y_j))$ es verdadero, y por valores nulos si $p(v_i(Y_j))$ es falso.

EquiKeep optimiza el trabajo del operador *Associator* en el cálculo de conjuntos de ítems frecuentes en la tarea de Asociación, al disminuir el número de combinaciones.

Ejemplo 2. Sea la relación *R(A, B, C, D)* de la tabla 3.

Restringir los valores de los atributos A=a1, B=b1, C=c2 y D=d1 es decir, $R1=\chi_{A=a1 \text{ v } B=b1 \text{ v } C=c2 \text{ v } D=d1}(R)$.

Tabla 3. Relación R

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

El resultado del operador *EquiKeep* $R1=c_{A=a1 \text{ v } B=b1 \text{ v } C=c2 \text{ v } D=d1}(R)$ se muestra en la tabla 4.

Tabla 4. Operación R1=c $_{A=a1\ v\ B=b1\ v\ C=c2\ v\ D=d1}$ (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

En este ejemplo, la tupla {*a2, b2, c1, d2*} es eliminada por resultar todos sus valores nulos.

1.3 Primitivas SQL para asociación

Los operadores algebraicos *Associator* y EquiKeep extienden el álgebra relacional para soportar la tarea de minería de datos Asociación. Con el fin de que estos operadores se puedan expresar en el lenguaje SQL, es necesario implementarlos como primitivas SQL. De esta forma, consideran Timarán, Millán y Machuca (2003), el lenguaje SQL será capaz de soportar eficientemente el descubrimiento de reglas de Asociación.

1.3.1 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 por la cláusula *RANGE,* de acuerdo con Timarán, Millán y Machuca (2003).

Dentro de la cláusula SELECT, la primitiva *Associator Range* tiene la siguiente sintaxis:

SELECT <*ListaAtributosTablaDatos*> [INTO<*NombreTablaAssociator*>]

FROM <*NombreTablaDatos*>

WHERE <*CláusulaWhere*>

ASSOCIATOR RANGE <*valor1*> UNTIL<*valor2*>

GROUP BY <*ListaAtributosTablaDatos*>

<*ListaAtributosTablaDatos*>::=*Atributo*>,<*ListaAtributos*>

<*ListaAtributos*>::=<*Atributo*>,<*ListaAtributos*>

<*ListaAtributos*>::=<*Atributo*>

<*valor1*>::=*1,2,3, 4 ..*

<*valor2*>::=*1,2,3,4 ...*

Donde:

- La cláusula SELECT <*ListaAtributosTablaDatos*> permite seleccionar los atributos de la tabla de datos <*NombreTablaDatos*> cuyos valores formarán los diferentes subconjuntos o conjuntos de ítems como tuplas de la nueva tabla <*NombreTablaAssociator*>.
- La cláusula INTO <*NombreTablaAssociator*> define el nombre de la tabla <*NombreTablaAssociator*> donde se almacenarán los resultados de ASSOCIATOR RANGE con los atributos especificados en la cláusula SELECT <*ListaAtributosTablaDatos*> como esquema. Si no se especifica la cláusula INTO, el resultado se almacenará en una tabla temporal como sucede normalmente en el lenguaje SQL.
- La cláusula FROM <*NombreTablaDatos*> WHERE <*CláusulaWhere*> determina el nombre de la tabla <*NombreTablaDatos*> de donde se extraerá el conjunto de tuplas que cumplan las restricciones de la <*CláusulaWhere*> para formar las diferentes combinaciones o conjuntos de ítems.
- La cláusula ASSOCIATOR RANGE <*número1*> UNTIL <*número2*> determina el tamaño inicial hasta el final de los diferentes subconjuntos o conjuntos de ítems que formará el operador *Associator.*
- La cláusula GROUP BY <*ListaAtributosTablaDatos*> agrupa la tabla <*NombreTablaDatos*> por los atributos especificados en <*ListaAtributosTablaDatos*>.

Ejemplo 3. Sea la tabla *Estudiantes* (PROGRAMA, EDAD, SEXO, ESTRATO, PROMEDIO) de la tabla 5, obtener los conjuntos de ítems frecuentes de tamaño 2 y 3 que cumplan con un soporte mínimo mayor o igual a 2.

Tabla 5. Tabla Estudiantes

Programa	Edad	Sexo	Estrato	Promedio
Sistemas	16..20	M	2	Medio alto
Idiomas	21..25	F	3	Regular
Sistemas	16..20	F	3	Regular
Física	21..25	M	2	Bajo
Sistemas	21..25	F	2	Alto
Derecho	25..30	F	1	Bajo

Los conjuntos de ítems frecuentes se obtienen con la siguiente sentencia SQL:

```
SELECT programa,sexo, estrato, count(*) AS soporte INTO
assostudent
```

```
FROM estudiantes
ASSOCIATOR RANGE 2 UNTIL 3
GROUP BY programa,sexo,estrato HAVING count(*)>=2
```

El proceso de obtención del resultado final se muestra en las tablas 6 y 7.

Tabla 6. Resultado primitiva Associator Range 2 until 3

Programa	Sexo	Estrato	Soporte
Sistemas	F		2
Sistemas	M		1
Sistemas	F	2	1
Sistemas	F	3	1
Sistemas	M	2	1
Sistemas		2	2
Sistemas		3	1
Idiomas	F		1
Idiomas	F	3	1
Idiomas		3	1
Física	M		1
Física		2	1
Física	M	2	1
Derecho	F		1
Derecho		1	1
Derecho	F	1	1
	F	1	1
	F	2	1
	M	2	2
	F	3	2

Tabla 7. Assostudent con el conjunto de Ítems frecuentes

Programa	Sexo	Estrato	Soporte
Sistemas	F		2
Sistemas		2	2
	F	3	2
	M	2	2

1.3.2 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.

Dentro de la cláusula SELECT, *EquiKeep On* tiene la siguiente sintaxis:

```
SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaEquiKeep>]
FROM <NombreTablaDatos>
WHERE <CláusulaWhere>
EQUIKEEP ON <CondiciónValoresAtributos >
<ListaAtributosTablaDatos>::=<Atributo>, <ListaAtributos>
<ListaAtributos>::=<Atributo>, <ListaAtributos>
<ListaAtributos>::=<Atributo>
<CondiciónValoresAtributos>::=<Atributo=Valor><operador>
<ListaCondición>
<CondiciónValoresAtributos>::=<Atributo><operador><(List
avalores)>
,<ListaCondición>
<ListaCondición>::=<Atributo=Valor>
<ListaCondición>::=<Atributo><operador><(Listavalores)>
<operador>::=AND,OR,NOT,IN
<Listavalores>::=1,2,3, 4 ...
<Valor>::=1,2,3, 4 ...
```

Donde:

- La cláusula SELECT <*ListaAtributosTablaDatos*> permite seleccionar los atributos de la tabla de datos <*NombreTablaDatos*> cuyos valores se conservarán si cumplen la condición <*CondiciónValoresAtributos*>, especificada en la primitiva *EquiKeep On*.
- La cláusula INTO <*NombreTablaEquiKeep*> define el nombre de la tabla <*NombreTablaEquiKeep*> donde se almacenarán los resultados de *EquiKeep On* con los atributos <*ListaAtributosTablaDatos*> como esquema. Si no se especifica la cláusula INTO, el resultado se almacenará en una tabla temporal.
- La cláusula FROM <*NombreTablaDatos*> WHERE <*CláusulaWhere*> determina el nombre de la tabla de datos <NombreTablaDatos> con el conjunto de registros que cumplan las restricciones de la <*CláusulaWhere*>.
- La primitiva EQUIKEEP ON <*CondiciónValoresAtributos*> conserva los valores de los atributos de <*ListaAtributosTablaDatos*> que cumplan la condición especificada en <*CondiciónValoresAtributos*>. El resto de valores de los atributos se hacen nulos.

Ejemplo 4. Sea la tabla *Estudiantes* (PROGRAMA, EDAD, SEXO, ESTRATO, PROMEDIO) de la tabla 5. Eliminar de la tabla Estudiantes los conjuntos de ítems tamaño 1 que no cumplan con el soporte mínimo

mayor o igual a 2, teniendo en cuenta que el conjunto de ítems frecuentes tamaño 1 son {sistemas: 3}, {F: 4}, {M: 2}, {2:3} y {3:2}

La sentencia SQL que permite obtener esta consulta utilizando la primitiva *EquiKeep On* es la siguiente:

```
SELECT programa,sexo, estrato INTO equistudent
FROM estudiantes
EQUIKEEP ON programa="Sistemas", sexo in (F, M), estrato in (2,3)
```

El resultado final se muestra en la tabla 8.

Tabla 8. Equistudent con el resultado de la primitiva Equikeep On

Programa	Sexo	Estrato
Sistemas	M	2
Sistemas	F	3
Sistemas	F	2
Idiomas	F	3
Física	M	2

2. Algoritmo EquipAsso

EquipAsso es un algoritmo que se basa en los operadores del álgebra relacional *Associator* y *EquiKeep,* aseguran Timarán, Millán y Machuca (2003) para el cálculo de conjuntos de ítems frecuentes. Este hecho facilita su integración al interior de cualquier SGBD, acoplando la tarea de Asociación de una manera fuerte y favorece la aplicación de técnicas de optimización de consultas para mejorar su rendimiento.

2.1 Descripción del algoritmo EquipAsso

En el primer paso del algoritmo se cuenta el número de ocurrencias de cada ítem para determinar los 1-conjuntos de ítems frecuentes L_1. En el subsiguiente paso, se aplica el operador *EquiKeep* para extraer de todas las transacciones en *D*, los conjuntos de ítems frecuentes tamaño 1, haciendo nulos el resto de valores. Luego, a la relación resultante *R*, se aplica el operador *Associator* para generar todos los conjuntos de ítems tamaño 2 (*Is = 2*) hasta máximo tamaño *n*, donde *n* es el grado de *R*. Finalmente, se calculan todos los conjuntos de ítems frecuentes *L*, contando el soporte de las diferentes combinaciones generadas por *Associator* en la relación R1. En la figura 1 se muestra el algoritmo *EquipAsso*.

2.2 Implementación del algoritmo EquipAsso en SQL

La figura 1 muestra el algoritmo *SQL-EquipAsso*. Utilizando las primitivas SQL que implementan los operadores algebraicos *Equikeep* y *Associator*, el algoritmo *EquipAsso* de la tabla 9, se implementa en el lenguaje SQL en la cláusula SELECT conjuntamente con las primitivas *Equikeep On* y *Associator Range.*

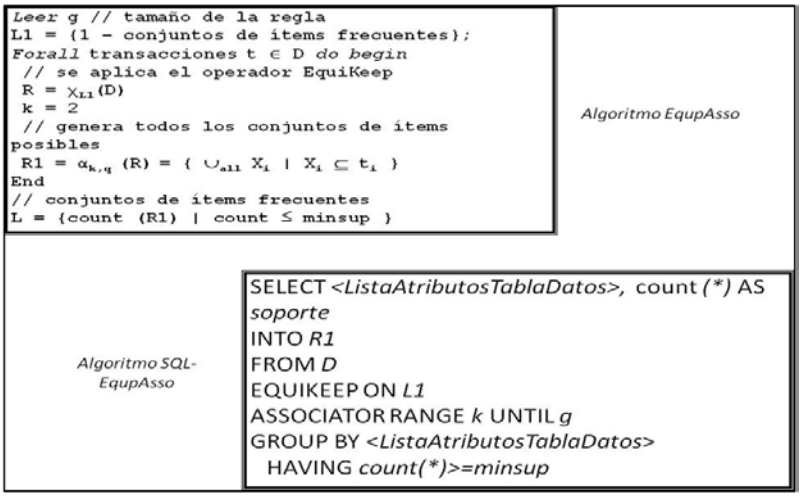

Figura 1. Algoritmo EquipAsso y Algoritmo SQL-EquipAsso

La ventaja de esta implementación es que facilita su integración al interior de cualquier SGBD, acoplando la tarea de Asociación de una manera fuerte, según Timarán (2001). Debido a que el algoritmo es ejecutado conjuntamente con los datos en el SGBD, la ventaja potencial de este enfoque es que resuelve los problemas de escalabilidad y rendimiento de las arquitecturas débilmente acopladas y facilita la aplicación de técnicas de optimización de consultas. Al implementarse este algoritmo con primitivas SQL permite la realización de consultas de minería de datos *ad hoc* y el desarrollo de aplicaciones en esta área.

Tabla 9. Resultado primitiva Equikeep On

Programa	Sexo	Estrato
Sistemas	M	2
Sistemas	F	3
Sistemas	F	2
Idiomas	F	3
Física	M	2

Ejemplo 5. Sea la tabla *Estudiantes* (PROGRAMA, EDAD, SEXO, ESTRATO, PROMEDIO) de la tabla 5. Con el algoritmo *EquipAsso*, obtener los conjuntos de ítems frecuentes de tamaño 2 y 3 que cumplan con un soporte mínimo mayor o igual a 2, teniendo en cuenta que el conjunto de ítems frecuentes tamaño 1 son {sistemas: 3}, {F: 4}, {M: 2}, {2:3} y {3:2}.

El algoritmo SQL-EquipAsso que permite obtener esta consulta utilizando conjuntamente las primitivas *EquiKeep On* y *Associator Range* es la siguiente:

```
SELECT programa,sexo, estrato INTO equistudent
FROM estudiantes
EQUIKEEP ON programa="Sistemas", sexo in (F, M), estrato in (2,3)
ASSOCIATOR RANGE 2 UNTIL 3
GROUP BY programa,sexo,estrato HAVING count(*)>=2
```

El proceso de obtención del resultado final se muestra en las tablas 9, 10 y 11.

Tabla 10. Resultado primitiva Associator Range 2 until 3

Programa	Sexo	Estrato	Soporte
Sistemas	F		2
Sistemas	M		1
Sistemas	F	2	1
Sistemas	F	3	1
Sistemas	M	2	1
Sistemas		2	2
Sistemas		3	1
	F	2	1
	M	2	2
	F	3	2

Tabla 11. Resultado final de EquipAsso

Programa	Sexo	Estrato	Soporte
Sistemas	F		2
Sistemas		2	2
	F	3	2
	M	2	2

Comparando las tablas 6 y 11 se nota que la primitiva *Equikeep On* disminuye el número de combinaciones que realiza *Associator Range*.

Con grandes volúmenes de datos esta disminución es supremamente importante para el rendimiento del algoritmo *EquipAsso*.

3. Evaluación de rendimiento del algoritmo *EquipAsso*

Debido a que no es posible implementar en SQL el resto de algoritmos de Asociación y con el fin de comparar el rendimiento del algoritmo *EquipAsso*, con respecto al rendimiento de los algoritmos *Apriori* (Agrawal y Srikant, 1994) y *FP-growth* (Han y Pei, 2000 y Han, Pei y Yin, 2000), estos se implementaron en una herramienta para minería de datos débilmente acoplada con el *SGBD PostgreSQL* denominada *TaryKDD* (Timarán et al., 2008).

El conjunto de datos utilizados en las pruebas pertenecen a las transacciones de uno de los supermercados más importantes del departamento de Nariño (Colombia) durante un periodo determinado. El conjunto de datos contiene 10.757 diferentes productos. Los conjuntos de datos utilizados con la herramienta TariyKDD se muestran en la tabla 12.

Tabla 12. Conjunto de datos pruebas

Nomenclatura	Número de registros	Número de transacciones	Promedio de ítems
BD85KT7	555.123	85.692	7
BD40KT5	197.337	40.256	5

Las pruebas consistieron en evaluar el rendimiento de los algoritmos *Apriori*, *FP-Growth* y *EquipAsso*, comparando los tiempos de respuesta para diferentes soportes mínimos. Los resultados de la evaluación del tiempo de ejecución de estos algoritmos, aplicados a los conjuntos de datos BD85KT7 y BD40KT5, se pueden observar en las tablas 13, 14 y la figura 2.

Tabla 13. Rendimiento algoritmos para el conjunto de datos BD85KT7

Soporte (%)	Tiempo (ms)		
	A priori	FP-Growth	EquipAsso
4.15	750	166	85
4.75	362	162	82
5.35	365	164	83
5.95	365	162	83
6.55	120	159	80

Tabla 14. Rendimiento algoritmos para el conjunto de datos BD40KT5

Soporte (%)	Tiempo (ms)		
	Apriori	FP-Growth	EquipAsso
1.90	268	66	29
2.00	265	64	29
2.10	132	63	27
2.20	45	61	27
2.30	44	61	27

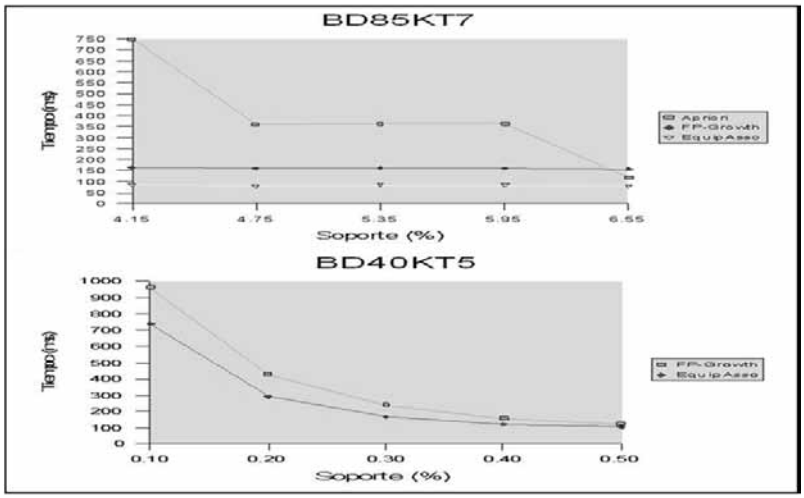

Figura 2. Rendimiento algoritmos para los conjuntos de datos BD85KT7 y BD40KT5

En general, observando el comportamiento de los algoritmos *FP-Growth* y *EquipAsso* con los diferentes conjuntos de datos, se puede decir que su rendimiento es similar, contrario al tiempo de ejecución de *Apriori*, que se ve afectado significativamente a medida que se disminuye el soporte.

4. Conclusiones y trabajos futuros

- Actualmente se cuenta con un algoritmo que permite calcular los conjuntos de ítems frecuentes en la tarea de Asociación sin generar conjuntos candidatos. Este algoritmo denominado *EquipAsso* genera el conjunto de ítems frecuentes directamente en cada tupla de una relación ya que se basa para su cálculo en dos operadores del álgebra relacional para minería de datos: *Associator* y *EquiKeep*. Esto facilita la integración fuerte de la tarea de Asociación con un Sistema de Gestión de Bases de Datos, al extender el lenguaje SQL con las primitivas *Associator Range* y *EquiKeep On* que implementan estos

operadores y favorece la aplicación de técnicas de optimización de consultas para mejorar su rendimiento.

- Como resultado de las pruebas realizadas, se puede observar que los algoritmos *EquipAsso* y *FP-Growth* tienen tiempos de ejecución semejantes. Con respecto a *Apriori* se puede apreciar que a medida que se disminuye el soporte sus tiempos de ejecución se incrementan hasta llegar a límites incomparables con respecto a *EquipAsso* y *FP-Growth*.
- Para soportes bajos, donde la cantidad de *itemsets* frecuentes aumenta y lo mismo su tamaño, *EquipAsso* tuvo un mejor desempeño que *FP-Growth*. En soportes altos, no existe una diferencia representativa entre el rendimiento de los algoritmos *FP-Growth* y *EquipAsso*.
- En la herramienta *TariyKDD*, el algoritmo *EquipAsso*, aunque presenta mejores tiempos de ejecución, consume más recursos del sistema, en especial la memoria principal.
- El rendimiento de cualquier algoritmo de minería de datos en una herramienta débilmente acoplada con un SGBD depende de la cantidad de memoria RAM disponible. Con grandes volúmenes de datos, el rendimiento de estos sistemas se ve afectado y es posible que se llegue a desbordar la memoria.
- Como trabajos futuros están el evaluar el rendimiento del algoritmo EquipAsso, utilizando grandes volúmenes de datos, en el sistema gestor de bases de datos PostgreSQL en el cual ya se encuentra implementado y acoplado de una manera fuerte.

Bibliografía

AGRAWAL, R. and SRIKANT, R. (1994). Fast Algorithms for Mining Association Rules. In: The 20th International Conference on Very Large Data Bases – VLDB, (12-15/09/1994), Santiago de Chile (Chile): Morgan Kaufmann. Proceedings, p. 487-499. ISBN: 1-55860-153-8.

AGRAWAL, R.; IMIELINSKI, T. and SWAMI, A. (1993). Mining Association Rules between Sets of Ítems in Large Databases. In: SIGMOD International Conference on Management of Data, (26-28/05/1993), Washington DC (USA): ACM. Proceedings, p. 207-216. ISBN: 0-89791-592-5

AGRAWAL, R.; MANNILA, H.; SRIKANT, R.; TOIVONEN, H. and VERKAMO, A.I. (1996). Fast Discovery of Association Rules. In: Advances in Knowledge Discovery and Data Mining. Menlo Park (USA): AAAI Press / The MIT Press. 595 p. ISBN: 0-262-56097-6.

BRIN, S.; MOTWANI, R.; ULLMAN, J. and TSUR S. (1997). Dynamic Ítemset Counting and Implication Rules for Market Basket Data. In: SIGMOD International Conference on Management of Data, (11-15/05/1997), Tucson (USA): ACM. Proceedings, p. 255-264. ISBN: 0-89791-911-4.

CABENA, P.; HADJINIAN, P.; STADLER, R.; VERHEES, J. and ZANASI, A. (1997). Discovering Data Mining from Concept to Implementation. New York (USA): Prentice Hall. 224 p. ISBN: 978-0137439805.

CHEN, M.; HAN, J. and YU, P. (1996). Data Mining: An Overview from Database Perspectiva. In: Journal IEEE Transactions on Knowledge Data Engineering. Vol. 8, No. 6 (dec.). New Jersey (USA): IEEE Educational Activities Department Piscataway. p. 866-883. ISSN: 1041-4347.

FAYYAD, U.; PIATETSKY-SHAPIRO, G. and SMYTH, P. (1996a). From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowlege Discovery and Data Mining. Menlo Park (USA): AAAI Pres/ The MIT Press. 595 p. ISBN: 0-262-56097-6.

FAYYAD, U.; PIATETSKY-SHAPIRO, G. and SMYTH, P. (1996b). The KDD Process for Extracting Useful Knowledge from Volumes of Data. En: Communications of the ACM. Vol. 39, No 11 (nov.). New York, NY (USA): ACM. p. 27-34. ISSN: 0001-0782.

HAN, J. and KAMBER, M. (2001). Data Mining: Concepts and Techniques. San Diego (USA): Morgan Kaufmann Publishers Inc. 745 p. ISBN: 978-1-55860-901-3.

HAN, J.; PEI, J. and YIN, Y. (2000). Mining Frequent Patterns without candidate Generation. En: SIGMOD International Conference on Management of Data, (16-18/05/2000), Dallas (USA): ACM. Proceedings, p. 1-12. ISBN: 1-58113-218-2.

HAN, L. and PEI, J. (2000). Mining Frequent Patterns by Pattern-Growth: Methodology and Implications, En: SIGKDD Explorations. Vol 2, No. 2 (dec.). New York (USA): ACM. p. 14-20. ISSN: 1931-0145

HERNÁNDEZ, O.J.; RAMIREZ, Q. M. y FERRI, R.C. (2004). Introducción a la Minería de Datos. Madrid (España): Pearson Prentice Hall. 656 p. ISBN: 84-205-4091-9.

IMIELNSKI, T. and MANNILA, H. (1996). A Database Perspective on Knowledge Discovery. En: Communications of the ACM. Vol. 39, No.11 (nov.): New York (USA): ACM. p. 58-64. ISSN: 0001-0782.

PARK, J.S.; CHEN, M. and YU, P. (1995). An Effective Hash-Based Algorithm for Mining Association Rules. In: SIGMOD International Conference on Management of Data, (22-25/05/1995), San José (USA): ACM. Proceedings, p. 175-186. ISBN: 0-89791-731-6.

SARAWAGI, S.; THOMAS, S. and AGRAWAL, R. (1998). Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications, In: SIGMOD International Conference on Management of Data, (02-05/06/1997), Seattle (USA): ACM. Proceedings, p. 343-354. ISBN: 0-89791-995-5.

SARAWAGI, S.; THOMAS, S. and AGRAWAL, R. (2000). Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In: Data Mining and Knowledge Discovery. Vol. 4, No. 2-3 (jul.): Hingham (USA): Springer. p. 89-125. ISSN: 1384-5810.

SAVASERE, A.; OMIECINSKI, E. and NAVATHE, S. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. In: The 21th International Conference on Very Large Data Bases – VLDB, (11-15/09/1995), Zurich (Switzerland): Morgan Kaufmann. Proceedings, p. 432-444. ISBN: 1-55860-379-4.

TIMARÁN, R. (2001). Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de Bases de Datos: un Estado del Arte. En: Revista Inge-

niería y Competitividad, Vol. 3, No. 2 (dic.). Cali (Colombia): Universidad del Valle. ISSN 0123-3033.

TIMARÁN, R.; CALDERÓN, A.; RAMÍREZ, I.; ALVARADO, J. y GUEVARA, F. (2008). TariyKDD una Herramienta de Minería de Datos Débilmente Acoplada con un SGBD. En: VII Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento - JIISIC, (30-1/02/2008), Guayaquil (Ecuador): Escuela Superior Politécnica del Litoral. Memoria Técnica, p 3-11. ISSN No. 1390-292X.

TIMARÁN, R.; MILLÁN, M. and MACHUCA, F. (2003). New Algebraic Operators and SQL Primitives for Mining Association Rules. In: IASTED International Conference Neural Networks and Computational Intelligence, (19-21/05/2003), Cancun (Mexico): The International Association of Science and Technology for Development. Proceedings, p. 227-232. ISBN: 0-88986-347-4.