Análisis de desempeño de EquipAsso: Un algoritmo para el cálculo de *Itemsets* frecuentes basado en operadores algebraicos relacionales.

Ricardo Timarán Pereira, Ph. D.*
ritimar@udenar.edu.co

Andrés O. Calderón Romero.*
and@mail.udenar.edu.co

Iván Ramírez Freyre.*
ivan.ramirezf@gmail.com

Juan Carlos Alvarado.*
endimeon777@yahoo.es

Fernando Guevara.*
drus3@latinmail.com

*Universidad de Nariño, Departamento de Ingeniería,
Ciudad Universitaria Torobajo
San Juan de Pasto, Colombia

Abstract

The task of searching for interesting relationships among data has been always a researching focus in data mining. The overall performance of mining association rules is determined by discovering large itemsets, i.e., sets of itemsets that have their support above a pre-determined minimum support. The different algorithms proposed for association rules task show different approaches to generate all large itemsets: Apriori, AprioriTid, AprioriHybrid, DHP, DIC, Partition, FP-Growth and EquipAsso.

In this paper, the performance of EquipAsso, an algorithm for discovering large itemsets, based on two new operators of relational algebra, is evaluated in relation with Apriori and FP-Growth algorithms, on Tariy, a tool for the Association task loosely coupled with a DBMS.

Keywords: Data Mining, Algorithms for Association Task, Data Mining Tools, Performance Analisys.

Resumen

La tarea de generar asociaciones a partir de los datos ha sido siempre un foco de investigación en el área de Minería de Datos. El cálculo de conjuntos de ítems frecuentes es la etapa computacionalmente más costosa cuando se aborda la tarea de asociación. Los distintos algoritmos propuestos para implementarla se han concentrado en el cálculo de conjuntos frecuentes: Apriori, AprioriTid, Apriori Hibrido, DHP, DIC, Partition, FP-Growth y EquipAsso.

En este artículo, se evalúa el rendimiento del algoritmo EquipAsso, un algoritmo para el cálculo de conjuntos de ítems frecuentes, basado en dos nuevos operadores del álgebra relacional, con respecto a los algoritmos Apriori y FP-Growth en Tariy, una herramienta para la tarea de Asociación débilmente acoplada con un SGBD.

Palabras claves: Minería de Datos, Algoritmos para Asociación, Herramientas de Minería de Datos, Análisis de Rendimiento.

1. Introducción

El Descubrimiento de Conocimiento en Bases de Datos (DCBD) es básicamente un proceso automático en el que se combinan desarrollo y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos y presentar resultados [5] [8].

Una herramienta DCBD debe integrar una variedad de componentes (técnicas de minería de datos, consultas, métodos de visualización, interfaces, etc.), que juntos puedan eficientemente identificar y extraer patrones interesantes y útiles de los datos almacenados en las bases de datos [9]. Las arquitecturas de estas herramientas se pueden ubicar en una de tres tipos: sistemas débilmente acoplados, medianamente acoplados y fuertemente acoplados con un Sistema Gestor de Bases de Datos (SGBD) [13]. Por otra parte, de acuerdo a las tareas que desarrollen, las herramientas DCBD se clasifican en tres grupos: herramientas genéricas de tareas sencillas, herramientas genéricas de tareas múltiples y herramientas de dominio específico [11].

En el área de minería de datos, el problema de derivar asociaciones de los datos ha sido un centro de gran atención e importancia. El problema fue formulado por Agrawal et al [1] [2] y a menudo se referencia como el problema de canasta de mercado (*market-basket*). En este problema se da un conjunto de ítems y una colección de transacciones que son subconjuntos (canastas) de estos ítems. La tarea es encontrar relaciones entre la presencia de varios ítems con esas canastas. 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* [1]. Un conjunto de ítems, se denomina frecuente si su soporte excede un umbral dado.

El cálculo de conjuntos de ítems frecuentes es la etapa computacionalmente más costosa cuando se aborda la tarea de asociación. Los distintos algoritmos propuestos para implementarla se han concentrado en el cálculo de conjuntos frecuentes: Apriori, AprioriTid, Apriori Hibrido [2], DHP (Standing for Direct Hashing and Pruning) [10], DIC (Dynamic Itemset Counting) [3], Partition [12], Sampling [17], FP-Growth [6] [7] y EquipAsso [15] [16].

EquipAsso [15] [16], es un algoritmo para el cálculo de los conjuntos de ítems frecuentes basado en dos nuevos operadores del álgebra Relacional para Asociación: *Associator* y *EquiKeep* [14] e implementado en el lenguaje SQL mediante las nuevas primitivas *Associator Range* y *EquiKeep On* [14].

En este artículo, se evalúa el rendimiento del algoritmo EquipAsso [15] [16] con relación a los algoritmos Apriori [2] y FP-Growth [6] [7] en Tariy, una herramienta de tarea sencilla para el descubrimiento de reglas de asociación débilmente acoplada con un SGBD.

El resto del artículo está organiza en secciones. En la sección 2, se dan algunos conceptos sobre Asociación y sobre los nuevos operadores del álgebra relacional que soportan esta tarea. En la sección 3, se describe la arquitectura de la herramienta Tariy, así como también algunos aspectos de la implementación de los diferentes módulos. En la sección 4, se hace el análisis del rendimiento de los algoritmos Apriori, FP-Growth y EquipAsso, implementados en Tariy. Finalmente, en la sección 5, se presentan las conclusiones.

2. Conceptos Preliminares

2.1 Reglas de Asociación

El problema de encontrar reglas de asociación fue formulado por Agrawal et al [1] [5] y a menudo se referencia como el problema de canasta de mercado (market-basket). En este problema se da un conjunto de ítems y una colección de transacciones que son subconjuntos (canastas) de estos ítems. La tarea es encontrar relaciones entre la presencia de varios ítems en esas canastas.

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 $T \subseteq I$. 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 $X \subset I$, $Y \subset I$ 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 $X \cup Y$.

Un ejemplo de una regla de asociación es: "el 30% de las transacciones que contienen cerveza también contienen pañales; el 2% de todas las transacciones contienen ambos ítems" [2]. Aquí el 30% es la confianza de la regla y el 2%, el soporte de la regla.
La confianza denota la fuerza de la implicación y el soporte indica la frecuencia de ocurrencia de los patrones en la regla. Las reglas con una confianza alta y soporte fuerte son referidas como reglas fuertes (strong rules) [1]. El problema de encontrar reglas de asociación se descompone en los siguientes pasos:

- Descubrir los itemsets frecuentes, i.e., el conjunto de itemsets que tienen el soporte de transacciones por encima de un predeterminado soporte ***s*** mínimo.

- Usar los itemsets frecuente para generar las reglas de asociación.

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

2.2. Operadores algebraicos relacionales para Asociación

Timarán [14] extiende el álgebra relacional [10] con los siguientes operadores algebraicos:

Associator (α). Este operador genera, por cada tupla de la relación *R*, todos sus posibles subconjuntos (itemsets) de diferente tamaño, en una sola pasada.
EquiKeep (χ). Este operador 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.

2.3 Primitivas SQL para Asociación

Timarán [14] extiende el lenguaje SQL con las siguientes primitivas:

- *Primitiva Associator Range*. Esta primitiva implementa el operador algebraico *Associator* [14] 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. Dentro de la cláusula SELECT, *Associator* tiene la siguiente sintaxis:

  ```
  SELECT <ListaAtributosTablaDatos> [INTO <NombreTablaAssociator>]
  FROM <NombreTablaDatos>
  WHERE <CláusulaWhere>
  ASSOCIATOR RANGE <numero1> UNTIL <numero2>
  GROUP BY <ListaAtributosTablaDatos>
  ```

- *Primitiva Equikeep On*. Esta primitiva implementa el operador algebraico *EquiKeep* [14] 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 >
  ```

3. Algoritmo Equipasso

Se han diseñado diferentes algoritmos para obtener los conjuntos de ítems frecuentes en la tarea de Asociación, dentro de los cuales están: Apriori, AprioriTid, Apriori Hibrido [2], DHP (Standing for Direct Hashing and Pruning) [9], DIC (Dynamic Itemset Counting) [3], Partition [15] y FP-Growth [6] [7]. Sin embargo, todos ellos se han implementado aisladamente y por fuera de los motores de bases de datos.

EquipAsso es un algoritmo que se basa en los nuevos operadores del álgebra relacional *Associator* y *EquiKeep* 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. A continuación se describe este algoritmo.

3.1 Descripción 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 *L1*. 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 items 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 *R'*. En la figura 1 se muestra el algoritmo *Equipasso*.

```
Leer g // tamaño de la regla
L1 = (1 – conjuntos de ítems frecuentes);
Forall transacciones t D do begin
  // se aplica el operador EquiKeep
  R = χL1(D)
  k=2
  // genera todos los conjuntos de ítems posibles
  R' = αk,q (R) = { ∪all Xi | Xi ⊆ ti }
End
// conjuntos de ítems frecuentes
L = { count(R') | count ≤ minsup }
```

Figura 1. Algoritmo EquipAsso

3.2 Algoritmo SQL-Equipasso

Utilizando las nuevas primitivas SQL para minar reglas de Asociación, el algoritmo *EquipAsso* de la figura 1, se implementa con la cláusula SELECT de la siguiente manera:

SELECT <ListaAtributosTablaDatos>, count(*) *AS* soporte *INTO R'*
FROM D
EQUIKEEP ON L1
ASSOCIATOR RANGE k UNTIL g
GROUP BY <ListaAtributosTablaDatos> *HAVING* count(*) >= minsup

3.3 Ejemplo de Aplicación del Algoritmo Equipasso

Sea la tabla booleana *Transacción* (Tid, Item_1, Item_2, Item_3, Item_4), la cual se indica en la tabla 1.

Tid	Item_1	Item_2	Item_3	Item_4
100	1	1	0	0
200	0	1	1	1
300	1	1	1	0
400	0	1	1	1
500	0	0	1	1

Tabla 1. Relación Transacción

Suponiendo que ya se han calculado los conjuntos de ítems frecuentes de tamaño 1 con un soporte igual a 3, los cuales son {Item_2}, {Item_3}, {Item_4}. Encontrar los conjuntos de ítems frecuentes de tamaño 2 y 3 formados por lo atributos Item_1, Item_2, Item_3, Item_4 y almacenar los resultados en la tabla *Frecuentes*, utilizando el algoritmo *EquipAsso*:

La sentencia SQL que permite obtener esta consulta es la siguiente:

SELECT Item_1, Item_2, Item_3, Item_4, count(*) AS soporte *INTO* Frecuentes
FROM Transaccion
EQUIKEEP ON Item_2 =1 *OR* Item_3=1 *OR* Item_4=1
ASSOCIATOR RANGE 2 *UNTIL* 3
GROUP BY Item_1, Item_2, Item_3, Item_4 *HAVING* count(*)>=3

Inicialmente se ejecuta la cláusula *SELECT*, en este caso se seleccionan los atributos Item_1, Item_2, Item_3, Item_4 de la tabla *Transacción*. Luego sobre la tabla resultante, se ejecuta la primitiva *EQUIKEEP ON* que conserva los valores de los atributos que cumplan la condición I*tem_2 =1 OR Item_3=1 OR Item_4=1* asi:

Item_1	Item_2	Item_3	Item_4
null	1	null	null
null	1	1	1
null	1	1	null
null	1	1	1
null	null	1	1

Tabla 2. Resultado primitiva SQL EquiKeep

Posteriormente, se generan los conjuntos de ítems de tamaño 2 y 3 con la primitiva *ASSOCIATOR RANGE* 2 *UNTIL* 3 así:

Item_1	Item_2	Item_3	Item_4
null	1	1	null
null	1	null	1
null	null	1	1
null	1	1	1
null	1	1	null
null	1	1	null
null	1	null	1
null	null	1	1
null	1	1	1
null	null	1	1

Tabla 3. Resultado primitiva SQL Associator Range

El resultado del agrupamiento y el conteo se presenta en la tabla 4.

En la tabla 5 se muestra la tabla *Frecuentes* donde se almacenan los conjuntos de items frecuentes cuyo soporte ≥ 3, que es el resultado final del algoritmo *SQL EquipAsso.*

Item_1	Item_2	Item_3	Item_4	soporte
null	1	1	null	3
null	1	null	1	2
null	null	1	1	3
null	1	1	1	2

Tabla 4. Resultado Agrupamiento y conteo

Item_1	Item_2	Item_3	Item_4	Soporte
null	1	1	null	3
null	null	1	1	3

Tabla 5. Resultado algoritmo SQL-EquipAsso

4. Aspectos de Implementación de EquipAsso en Tariy

El algoritmo EquipAsso se implementó conjuntamente con los algoritmos Apriori y FP-Growth en TARIY: una herramienta para minería de datos débilmente acoplada con el SGBD PostgreSQL, con el fin de evaluar el rendimiento de este algoritmo con respecto a los dos últimos.

4.1 Arquitectura de Tariy

Tariy se desarrolló bajo el sistema operativo Fedora Core 3 64 bits, lenguaje Java 1.5.0_06, con un computador con procesador AMD 64 bits a 2 Ghz, memoria RAM de 1Gb, con disco duro Serial ATA de 80Gb con una tasa de transferencia de 150 Mb/seg., entorno de desarrollo NetBeans 5.0 y SGBD utilizado para la conexión fue PostgreSQL 7.4.

La Herramienta Tariy esta compuesta por 4 módulos principales (figura 2):

- Módulo de conexión a SGBD, el cual permite la comunicación de la herramienta con el SGBD y los datos.
- Algoritmos, donde se ejecutan el cálculo de los conjuntos frecuentes con los tres algoritmos propuestos.
- Módulo de generación de reglas, encargado de generar las reglas que cumplan las restricciones de confianza.
- Interfaz Grafica, que permite la interacción del usuario con la herramienta.

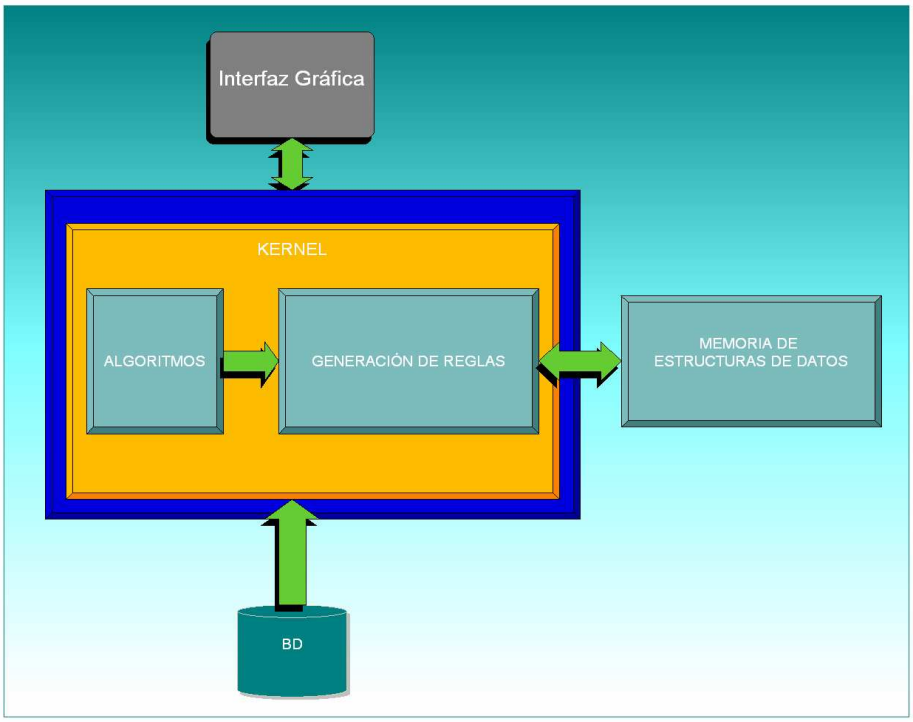

Figura 2. Arquitectura Herramienta Tariy.

4.2 Módulo de algoritmos

Los algoritmos EquipAsso, A-priori y FP-Growth se implementaron bajo el lenguaje de programación Java v.1.5.0_06.

Por otra parte para el almacenamiento de los datos provenientes del SGBD se utiliza un árbol n-ario el cual sirve para almacenarlos de forma compacta aprovechando que diferentes transacciones pueden compartir ítems repetidos, cada algoritmo obtiene las transacciones del árbol y se procede a la aplicación de los mismos. Los conjuntos de itemsets frecuentes obtenidos son almacenados en un árbol AVL balanceado.

5. Pruebas y Evaluación

Las pruebas y evaluación del rendimiento de los algoritmos Apriori, FP-Growth y EquipAsso, se realizaron en un computador con un procesador AMD 64 bits a 2 Ghz, con una memoria RAM de 1Gb, con disco duro Serial ATA de 80Gb con una tasa de transferencia de 150 Mb/seg.

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 minados con la herramienta Tariy se muestran en la tabla 6.

Para cada conjunto de datos se realizó preprocesamiento y transformación de datos con el fin de eliminar los productos repetidos en cada transacción y transformar las tablas objeto a un modelo simple (i.e. una tabla con esquema Tid, Item).

Nomenclatura	*Número de registros*	*Número de transacciones*	*Promedio ítems por Transacción*
BD85KT7	555.123	85.692	7
BD40KT5	194.337	40.256	5
BD10KT10	97.824	10.731	10

Tabla 6. Descripción de los conjuntos de datos minados.

Se evaluó 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, BD40KT5 y BD10KT10, se pueden observar en las figuras 3, 4 y 5 respectivamente.

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.

BD85KT7			
Soporte (%)	Tiempo (ms)		
	Apriori	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

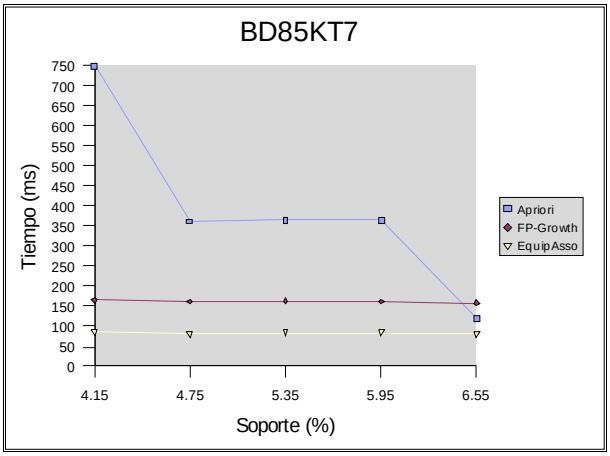

Figura 3. Resultados obtenidos para BD85KT7.

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

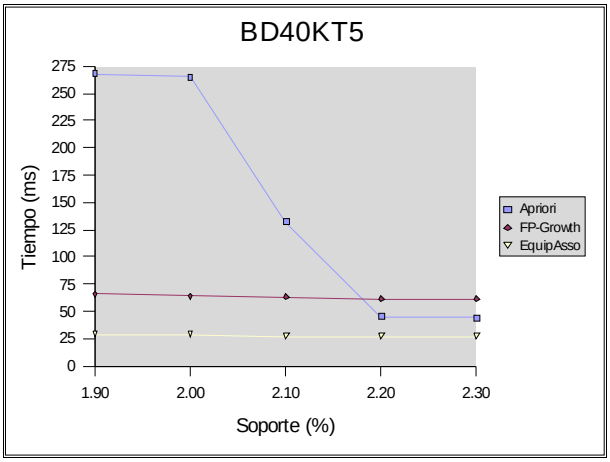

Figura 4. Resultados obtenidos para BD40KT5.

BD10KT10			
Soporte (%)	Tiempo (ms)		
	Apriori	FP-Growth	EquipAsso
3.00	525	28	15
4.00	182	26	14
5.00	105	25	13
6.00	53	24	13
7.00	52	24	13

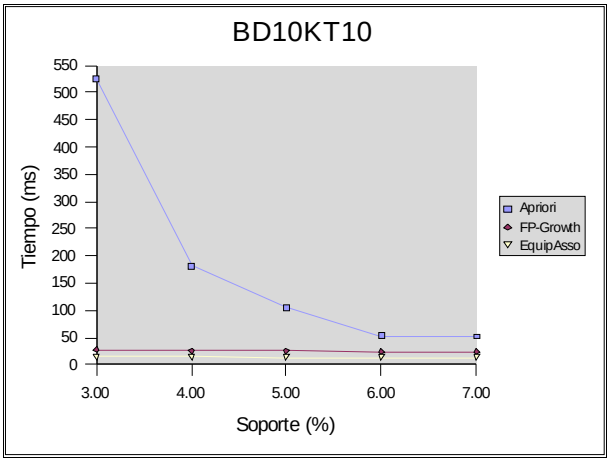

Figura 5. Resultados obtenidos para BD10KT10.

Analizando el tiempo de ejecución de únicamente los algoritmos FP-Growth y EquipAsso (figuras 6, 7 y 8) para los conjuntos de datos BD85KT7, BD40KT5 y BD10KT10 (figuras 6 y 7), con soportes mas bajos, el comportamiento de estos algoritmos sigue siendo similar.

BD85KT7			
Soporte (%)	Tiempo (ms)		
	Apriori	FP-Growth	EquipAsso
1.00	-	5130	1225
1.50	-	730	270
2.00	-	212	205
2.50	-	202	202
3.00	-	185	187

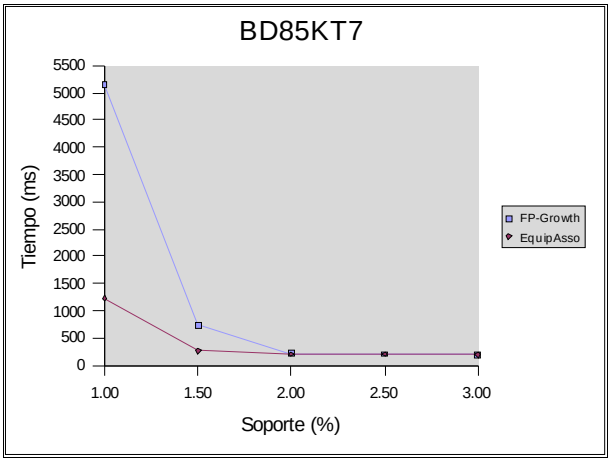

Figura 6. Resultados obtenidos para BD85KT7.

BD40KT5			
Soporte (%)	Tiempo (ms)		
	Apriori	FP-Growth	EquipAsso
0.10	-	965	741
0.20	-	425	290
0.30	-	240	168
0.40	-	156	121
0.50	-	124	105

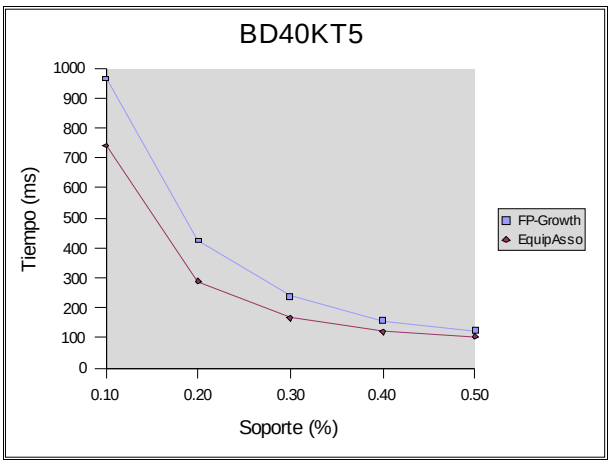

Figura 7. Resultados obtenidos para BD40KT5

BD10KT10			
Soporte (%)	Tiempo (ms)		
	Apriori	FP-Growth	EquipAsso
0.50	-	257	181
0.75	-	133	93
1.00	-	78	60
1.25	-	61	50
1.50	-	46	42

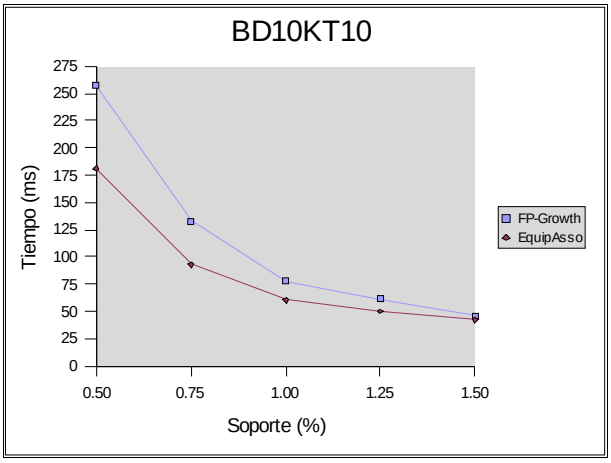

Figura 8. Resultados obtenidos para BD10KT10

6. Conclusiones

En este artículo se evaluó el desempeño del algoritmo EquipAsso [15] [16] con respecto a Apriori [2] y FP-Growth [7], para la tarea de minería de datos asociación, implementándolos en Tariy, una herramienta de minería de datos débilmente acoplada con PostgreSQL y desarrollada en el lenguaje JAVA 2. Para las pruebas, se utilizaron conjuntos de datos reales, pertenecientes a las ventas de un periodo determinado, de uno de los supermercados más importantes del departamento de Nariño (Colombia).

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 Tariy, 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.

En el futuro, se complementará esta investigación realizando pruebas con diferentes conjuntos de datos reales y se continuará con el desarrollo de la herramienta Tariy con una interfaz gráfica y nuevos módulos que soporten otras tareas y algoritmos de minería de datos.

Referencias

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

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

[3] Brin S., Motwani R., Ullman J., Tsur S., Dynamic Itemset Counting and Implication Rules for Market Basket Data, ACM SIGMOD, USA, 1997.

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

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

[6] Han, J., Pei, J., Yin, Y., Mining Frequent Patterns without candidate Generation, Proc. ACM SIGMOD, Dallas, TX, 2000.
[7] Han, J., Pei, J., Mining Frequent Patterns by Pattern-Growth: Methodology and Implications, SIGKDD Explorations, 2:14-20, 2000
[8] Han, J., Kamber, M. Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, SanFrancisco,2001.
[9] Imielnski T., Mannila, H., A Database Perspective on Knowledge Discovery, Communications of the ACM, Vol 39, No. 11, November, 1996.
[10] Park J.S., Chen M., Yu P., An Effective Hash-Based Algorithm for Mining Association Rules, ACM SIGMOD, San Jose, Ca USA, 1995.
[11] Piatetsky-Shapiro G., Brachman R., Khabaza T., An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. 1996.
[12] Savasere A., Omiecinski E., Navathe S., An Efficient Algorithm for Mining Association Rules in Large Databases, VLDB Conference, Zurich, Suiza,1995.
[13] Timarán, R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de datos: un Estado del Arte, en revista Ingeniería y Competitividad, Universidad del Valle, Volumen 3, No. 2, Cali, diciembre de 2001.
[14] Timarán R., Millán M., Machuca F., New AlgebraicOperators and SQL Primitives for Mining Association Rules, Proceedings of the IASTED International Conference Neural Networks and Computational Intelligence, Cancun, Mexico, 2003.
[15] Timarán, R., Millán, M., EquipAsso: un algoritmo para el descubrimiento de Reglas de Asociación basado en operadores algebraicos, en memorias de la 4ª Conferencia Iberoamericana en Sistemas, Cibernética e Informática CICI 2005, Orlando, Florida, EE.UU., julio 2005.
[16] 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, Calgary, Alberta, Canada, july 2005.
[17] Toivonen H., Sampling large databases for association rules, VLDB Conference, Bombay, India, 1996.