SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining

[1]Somboon Anekritmongkol, [2]Kulthon Kasemsan
[1,]Faculty of information Technology, Rangsit University, Pathumtani 12000, Thailand, somboon_a@hotmail.com
[2]Faculty of information Technology, Rangsit University, Pathumtani 12000, Thailand, kasemsan@rangsit.rsu.ac.th

Abstract

Discovery of association rules is an important for Data mining. One of the most famous association rule learning algorithms is Apriori rule. Apriori algorithm is one of algorithms for generation of association rules. The drawback of Apriori Rule algorithm is the number of time to read data in the database equal number of each candidate is generate. Many research papers have been published trying to reduce the amount of time needed to read data from the database. In this paper, we propose a new algorithm that will work rapidly and without frequency tree or temporary candidate itemsets in RAM or Hard disk. SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM). This algorithm will generate candidates are greater than minimum support on-the-fly by SQL. This algorithm is based on two major ideas. Firstly, compress data. Secondly, generation of candidate itemsets on-the-fly by SQL. Based on the experimental results, an increase in the number of transactions or the number of items did not affect the speed at which candidates were generated by this algorithm. The construction method of SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM) technique has twenty times higher mining efficiency in execution time than Apriori Rule.

Keywords: *Data Mining, Association Rule, Apriori Rule, Frequent Itemset*

1. Introduction

Data mining is the process of extracting patterns from data. Data mining is seen as an increasingly important tool by modern business to transform data into an informational advantage. Association rule mining is searches for recurring relationships in a database. One of the most popular technique in association rule mining is Apriori rule [1][2]. Association rule mining is usually associated with huge information. Association rules exhaustively look for hidden patterns, making them suitable for discovering predictive rules involving subsets of data set attributes. Association rule learners are used to discover elements that co-occur frequently within a data set consisting of multiple independent selections of elements (such as purchasing transactions), and to discover rules. In my point of view, Firstly, most of information in data set is same pattern. Secondly, amount of time to read the whole database. Thirdly, the pruning candidate in each step of process. This paper proposes the development of algorithm to discover association rules from large amounts of information that is faster than Apriori rule by using SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining(SMILE-ARM). The improvement focuses on compress data and improve performance without generate temporary frequency of Itemsets and reducing the number of times to read data from the database

2. Basic in Association Rule

Let $D = \{T_1, T_2, . . . , T_n\}$ [2] be a set of n transactions and let I be a set of items, $I = \{I_1, I_2, . . ., I_m\}$. Each transaction is a set of items, i.e. $Ti \subseteq I$. An association rule is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \emptyset$; X is called the antecedent and Y is called the consequent of the rule. In general, a set of items, such as X or Y, is called an itemset. In this work, a transaction record transformed into a binary format where only positive binary values are included as items. This is done for efficiency purposes because transactions represent sparse binary vectors. Let $P(X)$ be the probability of appearance of itemset X in D and let $P(Y|X)$ be the conditional probability of appearance of itemset Y given itemset X appears. For an itemset $X \subseteq I$, *support*(X) is defined as the fraction of transactions $Ti \in D$ such that $X \subseteq Ti$. That is, $P(X)$ = *support*(X). The support of a rule $X \Rightarrow Y$ is defined as *support*($X \Rightarrow Y$) = $P(X \cup Y)$. An association rule $X \Rightarrow Y$ has a measure of reliability called *confidence* ($X \Rightarrow Y$) defined as $P(Y|X)$ = $P(X \cup Y)/P(X)$ = *support*($X \cup Y$)/*support*(X). The standard problem of mining association rules [1] is to find all rules whose metrics are equal to or greater than some specified minimum support and minimum confidence thresholds. A k-itemset with support above the minimum threshold is called frequent. We use a third significance metric for association rules called *lift, lift*($X \Rightarrow Y$) = $P(Y|X)/P(Y)$ = *confidence* ($X \Rightarrow Y$)/*support*(Y). Lift quantifies the predictive power of $X \Rightarrow Y$; we are interested in rules such that *lift*($X \Rightarrow Y$) > 1.

3. Apriori rule

Apriori is an algorithm proposed by R. Agrwal and R. Srikant in 1994. Apriori rule employs an iterative approach know as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequency 1-itemsets is found by scanning the database to accumulate the count of each time and collecting those items satisfy minimum support. The resulting set is L_1. Next L_1 used to find the set of frequency 2-itemsets, which is used to find, and so on, until no more frequency k-itemsets can be found. The finding of each L_k requires one full scan of database.
Algorithm: Apriori rule. Find frequent itemsets using an iterative level-wide approach based on candidate generation.

Input: 1. D, a database of transaction;
    2. min_sup, The minimum support count threshold.

Output: L, frequent itemsets in D
Method:

```
(1)  L1 = Find_Frequent_1-Itemset(D);
(2)  for (k=2;$L_{k-1} \neq$ 0;k++){
(3)      $C_k$ = apriori_gen($L_K$-1);
(4)      for each transaction $T \in D$ { // scan D for count
(5)          $C_t$= subset($C_k$,t);// Get subset of t that are candidate
(6)          for each candidate $c \in C_t$
(7)              c.count++;
(8)      }
(9)      $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10) }
(11) Return L = $\cup_K L_K$;
```

Procedure apriori_gen(L_{k-1}frequent (k-1)-itemsets)

```
(1)  for each itemset $l_1 \in L_{k-1}$
(2)      for each itemset $l_2 \in L_{k-1}$
(3)          if($l_1$[1] = $l_2$[1]^($l_1$[2]=$l_2$[2]^…^($l_1$[k-2]=
                 $l_2$[k-2])^( $l_1$[k-1])< $l_2$[k-1] then {
(4)              C= $l_1 \rhd\lhd l_2$; // join step : generate candidates
(5)              if has_infrequent_subset(c,$L_{k-1}$) the
(6)                  delete c; // prune step : remove unfruitful candidate
(7)              else add c to $C_k$;
(8)          }
(9)  return $C_k$;
```

Procedure has_infrequent_subset(c:candidate k-itemset;
L_{k-1} : frequent(k-1)-itemsets); // use prior knowledge

```
(1)  for each (k-1)-subset s of c
(2)      if s $\notin L_{k-1}$ then
(3)          return True;
(4)  return False;
(5)
```

4. Boolean Algebra

Boolean algebra, developed in 1854 by George Boole in his book An Investigation of the Laws of Thought. Some operations of ordinary algebra, in particular multiplication xy, addition $x+y$, and negation $-x$, have their counterparts in Boolean algebra, respectively the Boolean Operations AND, OR, and NOT also called conjunction $x\wedge y$, disjunction $x\vee y$, and negation or complement $\neg x$ sometime $!x$. Some authors use instead the same arithmetic operations as ordinary algebra reinterpreted for Boolean algebra, treating xy as synonymous with $x\wedge y$ and $x+y$ with $x\vee y$.

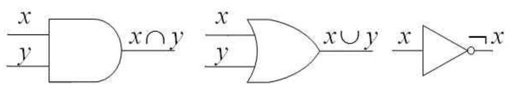


Figure 1. Logic Gate

5. Structured Query Language(SQL)

SQL was developed at IBM by Donald D. Chamberlin and Raymond F.Boyce in the early 1970. SQL was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970. SQL is a database computer language designed for managing data in relational database management systems (RDBMS). The Structured Query Language (SQL) defines the methods used to create and manipulate relational databases on all major platforms.

Table 1. Basic SQL Command

SQL Command	Description
SELECT	Retrieves data from a table.
INSERT	Add new Data to a table.
UPDATE	Modifies existing data in a table.
DELETE	Removes existing data from a table.
CREATE object	Create a new database object
ALTER object	Modifies the structure of an object.
DROP object	Removes an existing database object.

```
SQL query syntax
Select [Distinct] <column-name(s), arithmetic expression>
From <table-name(s)>
[Where <condition>]
[Group by <column-name(s)>]
[Having <condition>]
[Order by <column-name(s)> [ASC/DESC]]
```

```
SQL Insert syntax
Insert into <table-name>
[(column-name-1, column-name-2,…)]
Value (<value-1,value-2,…>)
```

6. SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining(SMILE-ARM)

An Algorithm of SMILE-ARM, The step of algorithm is 1. Find frequent itemsets minimum support. 2. Compress data by create a new structure are same pattern of transactions. 3. Generate Candidate by SQL from candidate 2-itemsets until k-itemsets by generate candidate based on actual data in pattern transactions. SMILE_ARM is able to generate any candidate itemsets without previous candidate such as create candidate 4-itemsets do not need to create candidate 3-itemsets or candidate 2-itemsets. SMILE-ARM is able to apply to do classification.

Algorithm: Pseudo-code of SMILE-ARM.
Tid_cl, a table of transaction
Tid_Pattern, a table contain Pattern Data
Final_Candidate , the result of candidates   Min_sup
Tid_Item, Itemset
Tid, transaction ID
Min_sup, Minimum support count all transactions

```
Procedure Find L1
(1)     for each itemset count  Tk    Tid_cl;
(2)     Delete Tk < Min_sup;
(3)     add to Final_Candidate;

Procedure Compress Structure
(1)     for each Tid , Tidk   L1 {
(2)         if Tid <> Tid_Patternk  then
(3)             Create Tid_Patternk;\\  Create Pattern;
(4)         else
(5)             Tid_Patternk++;
(6)     }

Procedure SQL_Command(k)
(1)     Insert into Final_Candidate (Candidate, Count_Items, xdim)
(2)     Select Tk.Tid_Item+Tk-1.Tid_Item+…+T1.Tid_Item, sum(T1.Feq),k
(3)     From Tid_Pattern as T1, Tid_Pattern as  T2,…,Tid_Pattern as Tk
(4)     Where (T1.Tid_Item < T2.Tid_Item and T2.Tid_Item < T3.Tid_Item and … Tk-1.Tid_Item <
                Tk.Tid_Item ) and  (T1.Tid=T2.Tid and T2.Tid=T3.Tid and …Tk-1.Tid =Tk.Tid)
(5)     Group by Tk.Tid_item, Tk-1.Tid_item,…, T1.Tid_item
(6)     Having sum(T1.Feq) >= min_Sup

Procedure Generate Associate Data
(1)     Find L1;
(2)     Compress Structure;
(3)     for each (k=2; k = Max_Itemk; k++){
(4)         SQL_Model(k);
(5)     }
```

Example: Minimum support 10 percent = 1.5

Table 2. Transaction Data

Trans#	Item		Trans#	Item		Trans#	Item
T001	I1		T006	I2		T011	I1
T001	I2		T006	I3		T011	I2
T001	I5		T007	I2		T011	I3
T002	I2		T007	I3		T012	I2
T002	I4		T008	I1		T012	I3
T003	I2		T008	I2		T013	I1
T003	I3		T008	I3		T013	I2
T004	I1		T009	I1		T013	I4
T004	I2		T009	I2		T014	I1
T004	I3		T009	I3		T014	I2
T004	I4		T010	I1		T014	I3
T005	I1		T010	I2		T015	I2
T005	I3		T010	I4		T015	I3

Step 1: Find L1
The algorithm scans all of transactions in order to find frequency item of each itemsets and remove frequency item less than minimum support.

Table 3. Support Count

Itemsets	Support count
{I1}	9
{I2}	14
{I3}	11
{I4}	4
{I5}	1

Eliminate {I5} lower Minimum Support

Table 4. Result of candidate-1 Itemset

Itemsets	Support count
{I1}	9
{I2}	14
{I3}	11
{I4}	4

Step 2: Compress Data
Count transactions are same items such as T003 = {I2, I3}, T006 = {I2, I3}, T007 = {I2, I3}, T012 = {I2, I3} and T015 = {I2, I3} or {I2, I3} = {(T003), (T006), (T007), (T012), (T015)} = 5. Create pattern is same structure such as T001 $\rightarrow$ create Pattern-1, T002 $\rightarrow$ Pattern-2, {T003, T006, T007, T012, T015} $\rightarrow$ Pattern-3, T004 $\rightarrow$ Pattern-4, {T010, T013 } $\rightarrow$ Pattern-5, T005 $\rightarrow$ Pattern-6 and {T008, T009, T011, T014} $\rightarrow$ Pattern-7.

Table 5. Data Compression

	I1	I2	I3	I4	Count
Pattern-1	X	X			1
Pattern-2		X		X	1
Pattern-3		X	X		5
Pattern-4	X	X	X	X	1
Pattern-5	X	X		X	2
Pattern-6	X		X		1
Pattern-7	X	X	X		4

Step 3: Find Candidate 2-Itemsets to k-Itemsets
To discover the set of frequency 2-itemsets to k-Itemsets by SQL Model, Generate candidate from Pattern table.

SQL Command for candidate 2-Itemsets.

```
Insert into CL_Final (Candidate,Count_Items,xdim)
Select T2.Tid_Item+T1.Tid_Item, sum(T1.Feq),2
From Tid_Pattern as T1, Tid_Pattern as  T2
Where (T1.Tid_Item < T2.Tid_Item)
    and (T1.Tid=T2.Tid)
Group by T2.Tid_item, T1.Tid_item
Having sum (T1.Feq) >= Min_Support
```

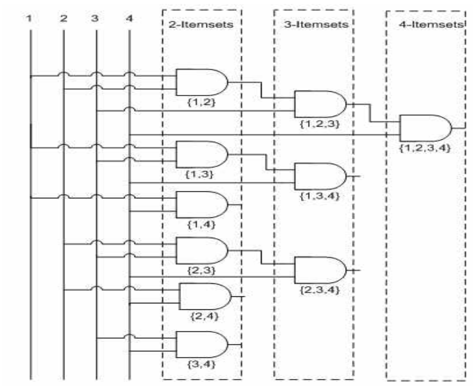


Figure 2. Generate Candidate K-Itemsets

Table 6. Result of candidate=2 Itemsets

Pattern	Data	Count	Final Candidate	Candiates	xdim
1	{I1,I2}	1	{I1,I2}	8	2
2	{I2,I4}	1	{I1,I3}	6	2
3	{I2,I3}	5	{I1,I4}	3	2
4	{I1,I2,I3,I4}	1	{I2,I3}	10	2
5	{I1,I2,I4}	2	{I2,I4}	4	2
6	{I1,I3}	1			
7	{I1,I2,I3}	4			

```
SQL Command for candidate 3-itemsets.
Insert into Final_Candidate (Candidate,Count_Items,3)
Select T3.Tid_Item +T2.Tid_Item+T1.Tid_Item,
        sum(T1.Feq),2
From Tid_Pattern as T1, Tid_Pattern as  T2, Tid_Pattern as  T3
Where (T1.Tid_Item < T2.Tid_Item and T2.Tid_Item <
         T3.Tid_Item) and (T1.Tid=T2.Tid and
         T2.Tid=T3.Tid)
Group by T3.Tid_item, T2.Tid_item, T1.Tid_item
Having sum (T1.Feq) >= Min_Support
```

Attribute of <xdim> is for control to generate candidates on each step.

Table 7. Generate from Candidate-3 Itemsets

Pattern	Data	Count	Final Candidate	Candidate	xdim
1	{I1,I2}	1	{I1,I2,I3}	5	3
2	{I2,I4}	1	{I1,I3,I4}	3	3
3	{I2,I3}	5			
4	{I1,I2,I3,I4}	1			
5	{I1,I2,I4}	2			
6	{I1,I3}	1			
7	{I1,I2,I3}	4			

Final Result: Candidate-Itemsets $\geq$ Min_Support

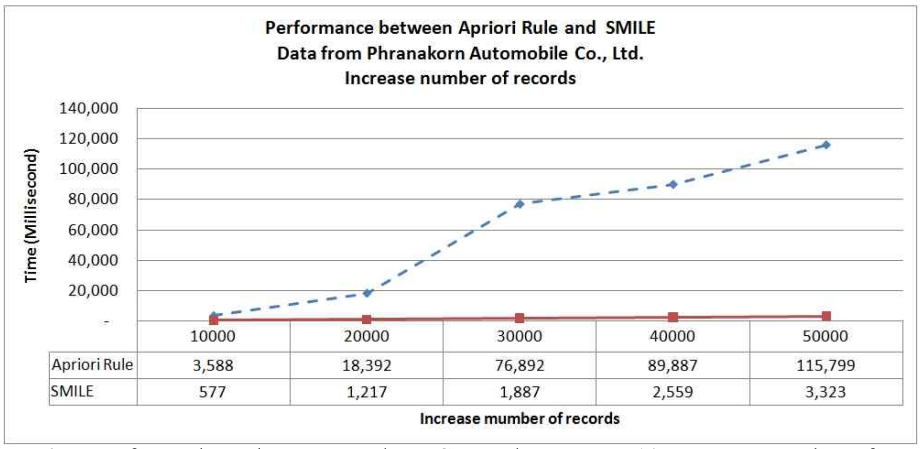
Table 8. Final Result

Itemsets	Final Candidate
{I1}	9
{I2}	14
{I3}	11
{I4}	4
{I1,I2}	8
{I1,I3}	6
{I1,I4}	3
{I2,I3}	10
{I2,I4}	4
{I1,I2,I3}	5
{I1,I3,I4}	3

7. Experimental results

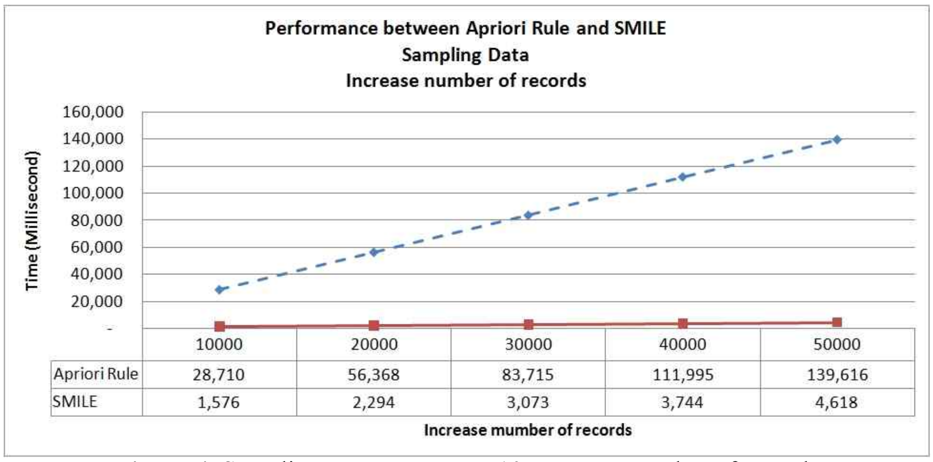
In this section, we performed a set of experiments to evaluate the effectiveness of SMILE-ARM. The experiment dataset consists of two kinds of data. First, data from Phranakorn Yontrakarn Co., Ltd. This company sales and offer car services to discover association data. Second, generate sampling data. The experiment of four criteria, Firstly, Increase amount of records from 10,000 to 50,000 records and fixed 10 items (Figure3, Figure4). Secondly, increase of item and fixed amount of records = 50,000 (Figure5, Figure6). Thirdly, increase amount of records from 10,000 to 190,000 records (Figure7). Fourthly, decrease of minimum support and fixed items and amount of records (Figure8).

Experiment 1: Increase number of records. Step 10,000 records. Fixed 10 items. Compare Apriori Rule with SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM). Apriori rule, with increasing amount of record will take longer time. SMILE-ARM has 35 times higher mining efficiency in execution time than Apriori Rule (10 Itemsets, 50,000 records).



	10000	20000	30000	40000	50000
Apriori Rule	3,588	18,392	76,892	89,887	115,799
SMILE	577	1,217	1,887	2,559	3,323

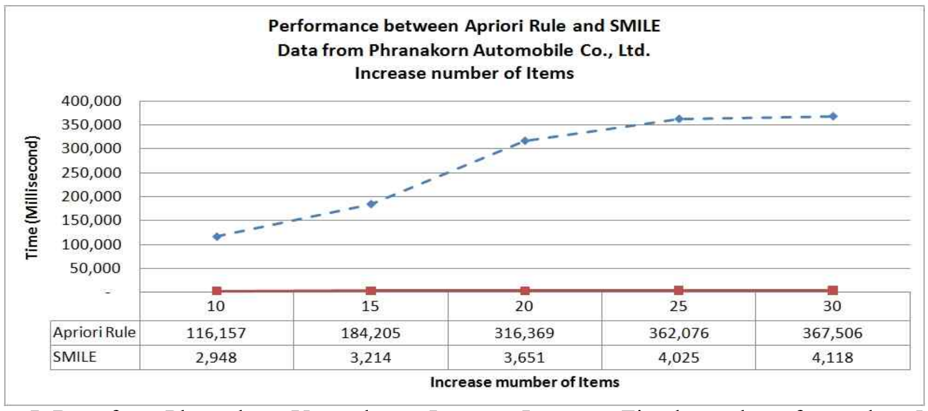
Figure 3. Data from Phranakorn Yontrakarn Co., Ltd. Itemsets=10, Increase number of records



	10000	20000	30000	40000	50000
Apriori Rule	28,710	56,368	83,715	111,995	139,616
SMILE	1,576	2,294	3,073	3,744	4,618

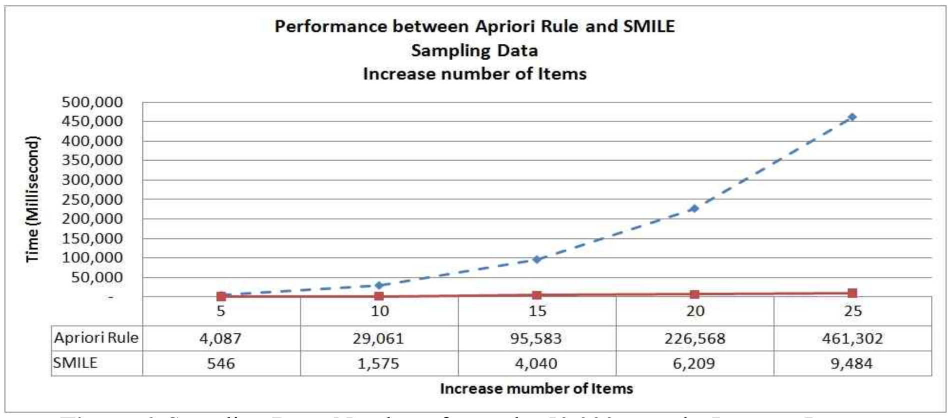
Figure 4. Sampling Data, Itemsets =10, Increase number of records

Experiment 2: Increase items. Fixed number of records 50,000 records. Compare Apriori Rule with SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM). Apriori rule, with increasing itemsets will take longer time. SMILE-ARM has 30 times higher mining efficiency in execution time than Apriori Rule (30 Itemsets, 50,000 records). That it mean amount of records will be slightly affected SMILE-ARM but Apriori rule performance takes a lot of time.



	10	15	20	25	30
Apriori Rule	116,157	184,205	316,369	362,076	367,506
SMILE	2,948	3,214	3,651	4,025	4,118

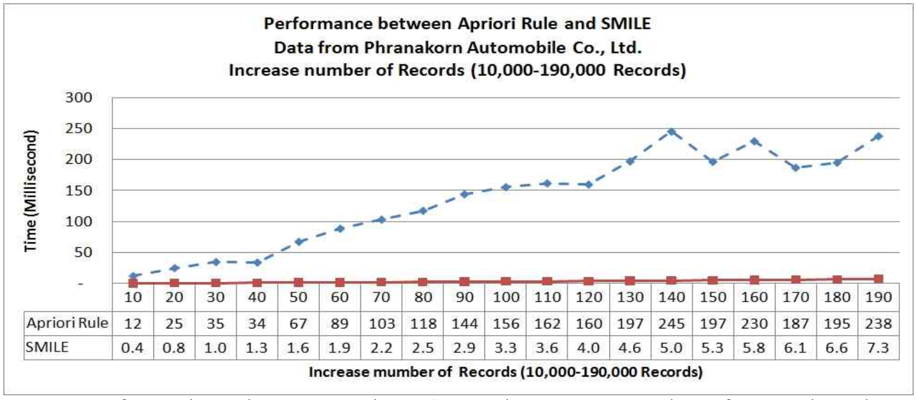
Figure 5. Data from Phranakorn Yontrakarn, Increase Itemsets, Fixed number of records = 50,000 records



	5	10	15	20	25
Apriori Rule	4,087	29,061	95,583	226,568	461,302
SMILE	546	1,575	4,040	6,209	9,484

Figure 6. Sampling Data, Number of records=50,000 records, Increase Itemsets

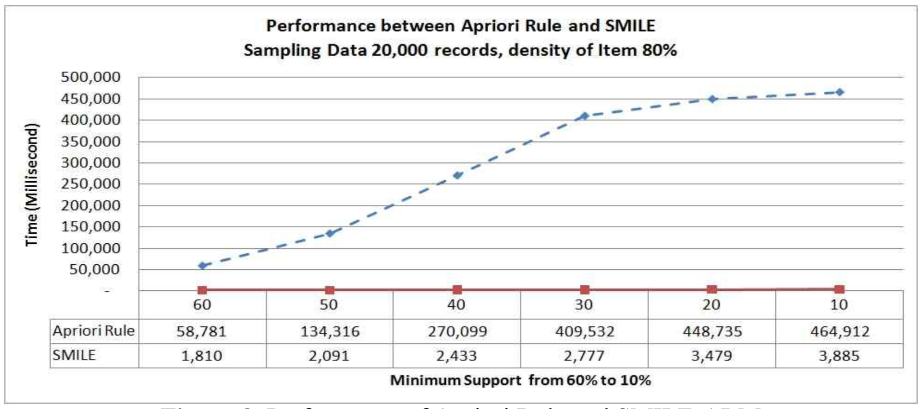
Experiment 3: Increase number of records from 10,000 to 190,000 records. Compare Apriori Rule with SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM). Real-life data from Phranakorn Yontrakarn Co., Ltd. shows effective performance between Apriori rule and SMILE-ARM as 190,000 records, Apriori rule takes 238 seconds but SMILE-ARM takes only 6 seconds on processing time. SMILE-ARM has 39 times higher mining efficiency in execution time than Apriori Rule (10 Itemsets, 190,000 records).



	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190
Apriori Rule	12	25	35	34	67	89	103	118	144	156	162	160	197	245	197	230	187	195	238
SMILE	0.4	0.8	1.0	1.3	1.6	1.9	2.2	2.5	2.9	3.3	3.6	4.0	4.6	5.0	5.3	5.8	6.1	6.6	7.3

Figure 7. Data from Phranakorn Yontrakarn Co., Ltd. Increase number of recoords and Increase Itemsets

Experiment 4: Decrease of minimum support from 60% to 10%. The step to change minimum support, Apriori rule low minimum support takes time to process 464 sec. but SMILE-ARM takes time to process 3 sec. SMILE-ARM has 154 times higher mining efficiency in execution time than Apriori Rule (10 Itemsets, 20,000 records, density 80%). SMILE-ARM slightly affects the performance because SMILE-ARM compresses data, SQL is generate candidate direct to database without temporary candidates.



	60	50	40	30	20	10
Apriori Rule	58,781	134,316	270,099	409,532	448,735	464,912
SMILE	1,810	2,091	2,433	2,777	3,479	3,885

Figure 8. Performance of Apriori Rule and SMILE-ARM

Table 9. Number of candidates

Minimum Support (%)	Number of Candidates
10	1023
20	1012
30	847
40	509
50	175
60	56

The result of experiments, SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM) discovers an association is faster than

Apriori rule. If increasing the number of records, Apriori rule will take time to read the whole data. If increasing the number of items, Apriori rule will create more candidates depending on the number of items but SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM) takes shorter time because it will compress data and SQL is generate candidate on-the-fly direct to database without temporary candidates.

8. Conclusion

The paper proposes a new association rule mining theoretic models and designs a new algorithm based on established theories. SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM) is compresses data and processes only the actual data by SQL command. SQL is very effective in terms of performance. SMILE-ARM is able to discover data more than twenty times faster than Apriori rule.

From the experiments, we found that the number of records and number of Items would not affect the performance of SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining (SMILE-ARM). The results is a especially amplified in itemsets from experiment 4 where we used data from 10,000 records, 10 Items and density 80%. SMILE-ARM has proven to have the best performance. We believe that SMILE-ARM is an important way to find Association data or Data mining and the best of algorithm for classification.

9. References

[1] Agrawal R., Imielinski T. and Swami A., “Mining association rules between sets of items in large databases”, In Proceedings of ACM SIGMOD International Conference on Management of Data Washington D.C., USA, pp. 207–216, 1993.
[2] Ramakrishnan Srikant and Rakesh Agawal, “Mining Generalized Association Rules”, In Proceedings of 21th VLDB Conference Zurich, Swizerland, pp. 407-419, 1995.
[3] Fukuda, T., Morimoto, Y. Morishita, S. and Tokuyama T., “Data Mining using Two-Dimensional Optimized Association Rules Scheme, Algorithms, and Visualization”, In Proceedings of ACM - SIGMOD International Conference on the Management of Data, pp. 13-23, 1996.
[4] M. Zaki, S. Parthasarathy, M. Ogihara and W. Li. "New Algorithms for Fast Discovery of Association Rules". 3rd International Conference on Knowledge Discovery and Data, Mining KDD’97, Newport Beach, CA, 283–296, 1997.
[5] Christian Borgelt, “Simple Algorithms for Frequent Item Set Mining”, Springer-Verlag, Berlin, Germany, pp.351-369, 2010.
[6] Zuling Chen and Guoqing Chen, “Building an Association Classifier Based on Fuzzy Association Rules”, International Journal of Computational Intelligence Systems, Vol. 1-3, pp. 262-272, 2008.
[7] HUANG Liusheng, CHEN Huaping, WANG Xun, CHEN Guoliang, “A Fast Algorithm for Mining Association Rules. Journal of Computer Science and Technology”, Vol.15, pp. 619-624, 2000.
[8] DU XiaoPing, TANG SgiWei and Akifumi Makinouchi, “Maintaining Discovered Frequent Itemsets: Cases for Changeable Database and Support”, Journal of Computer Science and Technology, Vol.18, pp. 648-658, 2003.
[9] S.Prakash and R.M.S.Parvathi, “An Enhanced Scaling Apriori for Association Rule Mining Efficiency. European Journal of Scientific Research”, Vol.39, pp.257-264, 2010.
[10] LI Qingzhong, WANG Haiyang, YAN Zhongmin and MA Shaohan, “Efficient Mining of Association Rules by Reducing the Number of Passes over the Database”, Journal of Computer Science and Technology, Vol.16, pp. 182-188, 2001.
[11] Somboon Anekritmongkol and M.L. Kulthon Kasemsan, “The Comparative of Boolean Algebra Compress and Apriori Rule Techniques for New Theoretic Association Rule Mining Model”, International Journal of Advancements in Computing Technology, Vol. 3 No. 1, pp. 58-67, 2011.

[12] Shui Wang, Le Wang, "An Implementation of FP-Growth Algorithm Based on High Level Data Structures of Weka-JUNG Framework", Journal of JCIT, Vol. 5, No. 9, pp. 287-294, 2010.
[13] S. Roy , D. K. Bhattacharyya , "OPAM: An Efficient One Pass Association Mining Technique without Candidate Generation", Journal of JCIT, Vol. 3, No. 3, pp. 32-38, 2008.