Incremental-Eclat Model: An Implementation via Benchmark Case Study

Wan Aezwani Bt Wan Abu Bakar, Zailani B. Abdullah, Md. Yazid B. Md Saman, Masita@Masila Bt Abd Jalil, Mustafa B. Man, Tutut Herawan and Abdul Razak Hamdan

Abstract Association Rule Mining (ARM) is one of the most prominent areas in detecting pattern analysis especially for crucial business decision making. With the aims to extract interesting correlations, frequent patterns, association or casual structures among set of items in the transaction databases or other data repositories, the end product of association rule mining is the analysis of pattern that could be a major contributor especially in managerial decision making. Most of previous frequent mining techniques are dealing with horizontal format of their data repositories. However, the current and emerging trend exists where some of the research works are focusing on dealing with vertical data format and the rule mining results are quite promising. One example of vertical rule mining technique is called Eclat which is the abbreviation of Equivalence Class Transformation.

W.A.B.W.A. Bakar (✉) · Z.B. Abdullah · Md.Y.B. Md Saman · Masita@Masila B.A. Jalil · M.B. Man
Department of Computer Science, School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu, Malaysia
e-mail: beny2194@yahoo.com

Z.B. Abdullah
e-mail: zailania@umt.edu.my

Md.Y.B. Md Saman
e-mail: yazid@umt.edu.my

Masita@Masila B.A. Jalil
e-mail: masita@umt.edu.my

M.B. Man
e-mail: mustafaman@umt.edu.my

T. Herawan
Department of Information Systems, Faculty of Computer Science and Information Technology, University of Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia
e-mail: tutut@um.edu.my

A.R. Hamdan
Data Mining and Optimization Research Group, Fakulti Teknologi & Sains Maklumat, Universiti Kebangsaan Malaysia, 43650 Bangi, Selangor, Malaysia
e-mail: arh@ukm.edu.my

© Springer International Publishing Switzerland 2016
P.J. Soh et al. (eds.), *Advances in Machine Learning and Signal Processing*, Lecture Notes in Electrical Engineering 387, DOI 10.1007/978-3-319-32213-1_4

In response to the promising results of the vertical format and mining in a higher volume of data, in this study we propose a new model called an Incremental-Eclat adopting via relational database management system, MySQL (My Structured Query Language) that serves as our association rule mining database engine in testing benchmark Frequent Itemset Mining (FIMI) datasets from online repository. The experimental results of our proposed model outperform the traditional Eclat with certain order of magnitude.

Keywords Association rule mining · Relational database · Mysql · Frequent itemset · Eclat algorithm

1 Introduction

Association rules mining (ARM) is first defined by [1] remains as one of the prominent and advance techniques in data mining. With the objectives to find the correlations, associations or casual structures among sets of items, association rules are the if-then statements that uncover relationships between unrelated data in transactional database, relational database or other types of data repositories. There are two (2) major aspects of ARM i.e. mining frequent itemset and generate interesting rules or also called positive association rule (PAR) and also mining infrequent itemset and generate interesting rules or also called negative association rule (NAR) [2, 3].

Most of the previous efforts on ARM have utilized the traditional horizontal transactional database layout format such as in [2, 4]. However, recently a number of vertical association rules mining algorithms have been proposed in works done by [5–8]. A general survey and comparison for association rule mining algorithms has been comprehensively initiated by [9] where a group of researchers have systemized the approaches of ARM algorithms into detailed schematic diagram. Since the introduction of frequent itemset mining, it has received a major attention among researchers [3, 10–14] and various efficient and sophisticated algorithms have been proposed to do frequent itemset mining. Among the three basic frequent itemset mining and best-known algorithms are Apriori [1, 2], Eclat [5, 15] and FP-Growth [4, 16, 17]. The state of the art in association rule mining algorithm is dealing with the extraction of frequent itemsets that occur with high frequency of support, s% in transactional database. The prerequisite feature that must taking into major account is the database format or sometimes called as database layout. The database format (either in horizontal or vertical) might be a major determinant of how far and how fast association rule is mined prior to generation of frequent itemsets from a database. Among existing works on vertical data association rules mining [4–7, 9–11, 15], Eclat algorithm is known for its ‘fast’ intersection of its tidlist whereby the resulting number of tids is actually the support (frequency) of each itemsets [5, 9]. That is, we should break off each intersection as soon as the resulting number of tids is below minimum support threshold that we have set.

Studies on Eclat algorithm has attracted many development including the works of [6, 8, 18]. Motivated to its 'fast intersection', this paper proposed a new Incremental-Eclat model by taking Eclat as well as Eclat-variants as the based models. This new model proposes a new incremental mechanism in Eclat dimension model.

The rest of the paper is organized as follows. Section 2 describes the related works of vertical algorithm in ARM. Section 3 explains the theoretical background. Section 4 outlines on Traditional Eclat and Eclat-variants versus Incremental-Eclat concept. This is followed by the experimentation of Eclat and Incremental-Eclat in Sect. 5. Finally, conclusion and future direction is reported in Sect. 6.

2 Related Works

The first aspect in association rule mining is looking on the frequent itemset mining as applied in FP-Growth and Eclat algorithms. The FP-Growth is defined in [4, 17] employs a divide-and-conquer strategy and a FP-tree data structure to achieve a condensed representation of the transaction database. It has become the fastest algorithms for frequent pattern mining. In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partitioned the database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases. The generation of FP-tree is done by counting occurrences and depth first search (refer to Fig. 2) in searching the nodes [9].

The Eclat is first initiated by [5] stands for *Equivalence Class Transformation* [15, 16] and as an acronym of *Equivalence Class Clustering and bottom up Lattice Traversal* [18]. It also takes a depth-first search in searching nodes and intersecting, in which each item is represented by a set of transaction IDs (called a tidset) whose transactions contain the item. The tidset of an itemset is generated by intersecting tidsets of its items. Because of the depth-first search, it is difficult to utilize the downward closure property like in Apriori [1, 2] that based on breadth-first searching. However, using tidsets has an advantage that there is no need for counting support, the support of an itemset is the size of the tidset representing it. The main operation of Eclat is intersecting tidsets, thus the size of tidsets is one of main factors affecting the running time and memory usage of Eclat. The bigger tidsets are, the more time and memory are needed.

Continuing the work by [5], a new vertical data representation, called Diffset is proposed in [6], and introduced dEclat, an Eclat-based algorithm using diffset. Instead of using tidsets, they use the difference of tidsets (called diffsets). Using diffsets has drastically reduced the set size representing itemsets and thus operations on sets are much faster. The dEclat has shown to achieve significant improvements in performance as well as memory usage over Eclat, especially on dense databases. However, when the dataset is sparse, diffset loses its advantage over tidset. Therefore, the researchers suggested using tidset format at the start for sparse databases and then switching to diffset format later when a switching condition is met.

The VIPER (Vertical Itemset Partitioning for Efficient Rule Extraction) is established by [7] uses compressed vertical bitmaps for association rule mining has shown to outperform an optimal horizontal algorithm that has complete apriori knowledge of the identities of all frequent itemsets and only need to find their frequency.

Following the efforts in [5, 6], a novel approach for vertical representation wherein the authors used the combination of tidset and diffset and sorted the diffset in descending order to represent databases [8] which is called sortdiffset. The technique is claimed to eliminate the need of checking the switching condition and converting tidset to diffset format regardless of database condition either sparse or dense. Besides, the combination can fully exploit the advantages of both tidset and diffset format where the prelim results have shown a reduction in average diffset size and speed of database processing.

Motivating on the support measure in frequent item mining in [6], an improvement work by [11] is done whereby a conjecture of support count and improvement of traditional Eclat are proposed. The new Bi-Eclat algorithm sorted on support is introduced such that items are in descending order according to frequencies in transaction cache while itemsets use ascending order of support during support count. As compared to traditional Eclat, it has gained better performance when tested on several public selected datasets.

3 Theoretical Background

Following is the formal definition of the problem defined in [11]. Let $I = \{\mathrm{i}_1, \mathrm{i}_2, \ldots \mathrm{i}_m\}$ for $|m| > 0$ be the set of items. D is a database of transactions where each transaction has a unique identifier called tid. Each transaction T is a set of items such that $T \subseteq I$. An association rule is an implication of the form $X \subseteq Y$ where X represent the antecedent part of the rule and Y represents the consequent part of the rule where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. A set $X \subseteq I$ is called an *itemset*. An itemset with k-items is called a k—*itemset*. The itemset that satisfies minimum support is called frequent itemset. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has support s in the transaction set D if s% of transaction in D contains $X \cup Y$. A rule is *frequent* if its support is greater than minimum support (min_supp) threshold. The rules which satisfy minimum confidence (min_conf) threshold is called *strong rule* and both min_supp and min_conf are user specified values [15]. An association rule is considered *interesting* if it satisfies both min_supp and min_conf thresholds [18].

4 Traditional Eclat and Eclat-Variants Versus Incremental-Eclat Concept

A. Traditional Eclat

An Eclat algorithm is first proposed by [13, 18, 19] for discovering frequent itemsets from a vertical database layout of a transaction database. It uses prefix-based equivalence relation, θ_1 along with bottom up search. It enumerates all frequent itemsets. There are two main steps: candidate generation and pruning.

1. Candidate Generation

In candidate generation, each *k-itemset* candidate is generated from two frequent (k-1)-itemsets and its support is counted, if its support is lower than the threshold, then it will be discarded, otherwise it is frequent itemsets and used to generate (k + 1)-itemsets. Since Eclat uses the vertical layout, counting support is trivial. Depth-first searching strategy is done where it starts with frequent items in the item base and then 2-itemsets from 1-itemsets, 3-itemsets from 2-itemsets and so on.

The first scan of the database builds the transaction id (tids) of each single items. Starting with single item (k = 1), then the frequent (k + 1)-itemset will grow from the previous k-itemset will be generated with a depth first computation order similar to FP-Growth [15]. The computation is done by intersecting tids of the frequent k-itemsets to compute the tidsets of the corresponding (k + 1)-itemsets. The process is repeated until no more frequent candidate itemsets can be found.

2. Equivalence Class Clustering

An equivalence class $E = \{(i_1, t(i_1 \cup P)), \ldots, (i_k, t(i_k \cup P))|P\}$, considering the set $\{i_1, \ldots, i_k\}$ as an item base, it will have a tree of itemsets over this item base and if the prefix P is appended to all itemsets in this new tree, it will have a set of all itemsets sharing the prefix P in the search tree over the item base B. In other word, from this equivalence class, a set of all itemsets sharing the prefix P could be generated and this set forms a sub tree of the initial search tree.

Eclat starts with prefix {} and the search tree is actually the initial search tree. To divide the initial search tree, it picks the prefix {a}, generate the corresponding equivalence class and does frequent itemset mining in the sub tree of all itemsets containing {a}, in this sub tree it divides further into two sub trees by picking the prefix {ab}: the first sub tree consists of all itemset containing {ab}, the other consists of all itemsets containing {a} but not {b}, and this process is recursive until all itemsets in the initial search tree are visited. The search tree of an item base {a,b, c,d,e} is represented by the tree as shown in Fig. 1.

In the vertical layout, each item i_k in the item base B is represented as $i_k : \{i_k, t(i_k)\}$ and the initial transaction database consists of all items in the item base.

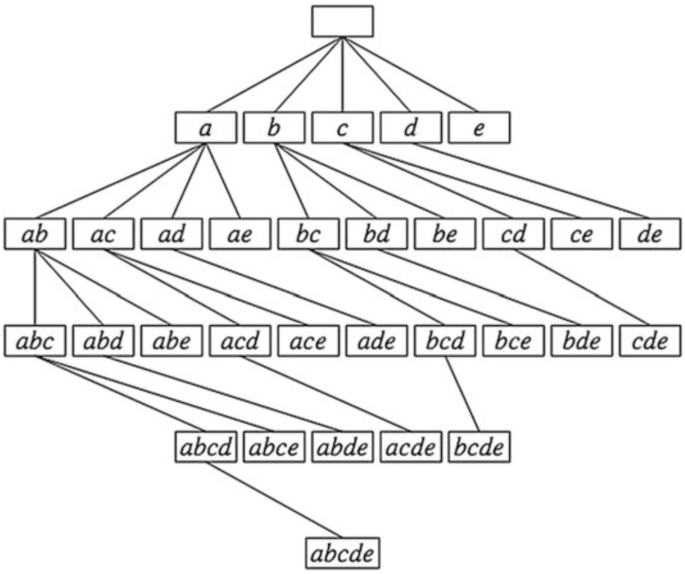
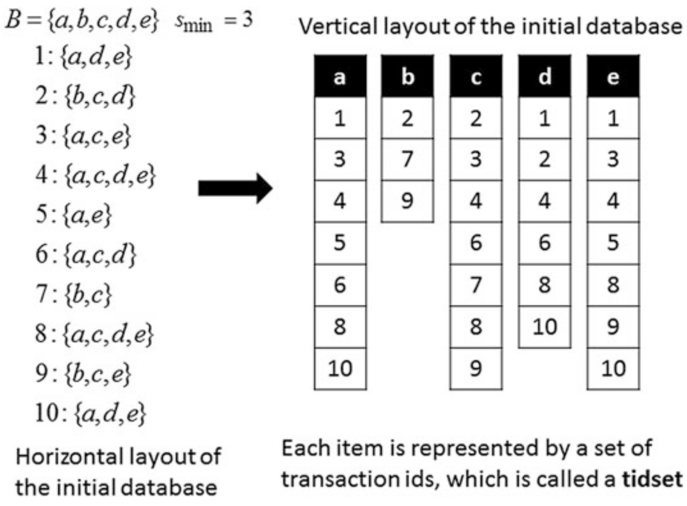


Fig. 1 Prefix Tree for 5 items {a,b,c,d,e} with null set



$B = \{a,b,c,d,e\}$ $s_{\min} = 3$

1: $\{a,d,e\}$
2: $\{b,c,d\}$
3: $\{a,c,e\}$
4: $\{a,c,d,e\}$
5: $\{a,e\}$
6: $\{a,c,d\}$
7: $\{b,c\}$
8: $\{a,c,d,e\}$
9: $\{b,c,e\}$
10: $\{a,d,e\}$

Horizontal layout of the initial database

Vertical layout of the initial database

a	b	c	d	e
1	2	2	1	1
3	7	3	2	3
4	9	4	4	4
5		6	6	5
6		7	8	8
8		8	10	9
10		9		10

Each item is represented by a set of transaction ids, which is called a **tidset**

Fig. 2 Transformation from horizontal to vertical layout

For both layouts, it is possible to use the bit format to encode tids and also a combination of both layouts can be used [20, 21]. Figure 2 illustrates how data in horizontal layout is transformed by a set of transaction ids or tidset in vertical layout [20].

In Fig. 2, the items in B consist of {a,b,c,d,e} and each itemsets are allocated with unique identifiers (tids) for each transactions. This is clearly visualized in horizontal format. To switch to vertical format, every items {a,b,c,d,e} are then organized where all items are allocated with their corresponding tids. When this is done, it is clearly visualized the support of each items through the counting number of every item's tids.

B. Traditional Eclat (Tidset) and Eclat-Variants (Diffset and SortDiffset)

A detail steps taken in Eclat-tidset algorithm when assuming that the initial transaction database is in vertical layout and represented by an equivalence class E with prefix {} is shown in Fig. 3 (refer to steps *1,2,3,4,5,6,7,8,9,10*). The itemset in the database table is first sorted in ascending order into each column and then, the looping value is getting based on the number of columns occupied by the itemset. Starting with the first column and second column, the intersection of items in both columns is done and save the intersecting tidlist into database. This process is repeated until all frequent itemsets have been enumerated. The looping number is determined by the number of attributes of the dataset read.

The Eclat-diffset or named as dEclat (different set or diffset) is proposed by [6] where the authors represent an itemset by tids that appear in the tidset of its prefix but do not appear in its tidsets. In other words, diffset is the difference between two (2) tidsets (i.e. tidset of the itemsets and its prefix). Using diffset, the cardinality of sets representing itemsets is reduced significantly and this results in faster intersection and less memory usage. The dEclat is shown to achieve significant improvements in performance and memory usage over traditional Eclat especially in dense database. However when database is sparse, it loses its advantages over tidsets. Then in [6] the authors suggested to use tidset format at starting for sparse database and later switch to diffset format when switching condition is met. The pseudocode of diffset is given in Fig. 3 (refer to steps *1,2,3,4,5,6.1,7,8,9,10*).

```
1.   start
2.   Sort data by itemset
2.1  Sort data by itemset with descending order of dataset value.
3.   looping=numberofcolumn;
     //process tidset
4.   for(i=0;i<looping;i++)
5.   {
6.   get intersect data for column[i] with  column[i+1];
6.1  get diffset data for column[i] with  column[i+1];
7.   save to db;
8.   add next transaction data;
9.   }
10.  end
```

Fig. 3 Eclat-tidset (in steps 1,2,3,4,5,6,7,8,9,10), Eclat-diffset (in steps 1,2,3,4,5,6.1,7,8,9,10), and Eclat-sortdiffset (in steps 1,2.1,3,4,5,6,7,8,9,10)

The Eclat-SortDiffset is established in [8] that applied the sorting of diffset in descending order. It claims to achieve a significant reduces in running time and memory usage. Since $p(PXY) = sup(PX) - |d(PXY)| = sup(PY) - |d(PYX)|$, both $d(PXY)$ and $d(PYX)$ could be used to calculate $sup(PXY)$. Therefore, the smaller one of the two should be used to calculate $sup(PXY)$ to reduce the memory usage and processing time. Because $d(PXY) = d(PY) - d(PX)$ and $d(PYX) = d(PX) - d(PY)$, if $d(PX)$ is smaller than $d(PY)$ then $d(PYX)$ is smaller than $d(PXY)$. In general, diffsets in an equivalence class should be sorted in descending order according to size to generate new itemsets represented by diffsets with smaller sizes. The portion of SortDiffset algorithm is given in Fig. 3 (refer to steps *1,2.1,3,4,5,6.1,7,8,9,10*) where the difference only in step 2 as compared to diffset algorithm.

C. Incremental Eclat Algorithm

The initial objective of Incremental Eclat is to handle the issues of big and dynamic data. In real application, data is becoming bigger prior to non-stop transaction being done in many of real world application domain. With respect to association rule mining, items may incur either in two (2) different ratios i.e. increment of itemset or increment of records in typical database. To mine frequent items, it may require higher specification of memory and spaces of the computer hardware. Incurring itemsets result in bigger cardinality of data in equivalence class clustering whereas incurring records consumes higher volume of data. Thus, Incremental Eclat attends to reduce a memory and spaces requirement by implementing flushing of memory prior to each itemset being visited before intersecting the next itemsets. The current or last transaction data that is in-memory will be flushed before proceeding into next transaction data. Adopting in a structured and relational MySQL database, the incremental of either itemset or records of transaction is easier and more efficient in structuring the data. The pseudocode of the proposed algorithm is denoted in Fig. 4. The only difference in Incremental-Eclat engine is depicted in step 9.

```
1.   start
2.   Sort data by itemset
2.1  Sort data by itemset with descending order of dataset value.
3.   looping=numberofcolumn;
4.   for(i=0;i<looping;i++)
5.   {
6.   get intersect data for column[i] with  column[i+1];
6.1  get diffset data for column[i] with  column[i+1];
7.   save to db;
8.   add next transaction data;
9.   flush value for current/last transaction data;
10.  }
11.  end
```

Fig. 4 Incremental-Eclat approach

5 Experimentation

A. Database Platform

All experiments are performed on a Dell N5050, Intel ® Pentium ® CPU B960 @ 2.20 GHz with 2 GB RAM in a Win 7 64-bit platform. The software specification used is MySQL version 5.5.27—MySQL community server (GPL) for our database server, Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7 for our web server and phpMyAdmin with version 3.5.2.2. For the kick-off experimentation, we start with simple synthetic dataset. In addition, we have retrieved benchmark datasets from http://fimi.ua.ac.be/data/ in a *.dat file format. For the ease of use in MySQL, we convert datasets into comma separated value (csv) format. The characteristics of benchmark datasets with the average size include chess and mushroom is depicted in Table 1. For the faster results of a depth first with intersection searching strategy in database mining, we have split chess, connect and mushroom datasets into benchmark trained datasets. There are three (3) sub divisions of each i.e. chess1000 × 12, chess2000 × 12, chess3000 × 12. The same sub division is done in connect.

B. Empirical Results

The experimentation is done with regards to Eclat algorithm (tidset) in [5], dEclat algorithm (diffset) in [6] and sortdiffset algorithm in [8]. Figure 5, 6, 7, 8 show the graph of performance result in execution time between Eclat and Eclat-variants versus Incremental-Eclat within chess and connect datasets prior to running with Eclat engine versus Incremental-Eclat engine. The graphs indicate the result of sortdiffset algorithm with an order of magnitude outperforms diffset and tidset algorithm in Eclat engine. The execution time shows a slight decreased in connect dataset for about 0.31 % in Incremental-Eclat as compared to Eclat engine. However, in chess, the execution of Incremental-Eclat decreases tremendously for 21.03 % as compared to Eclat engine. As overall, Incremental-Eclat performs better than Eclat in certain order of magnitude. The incremental process either in itemsets

Table 1 Database characteristics

Database	Size (KB)	Average length	Records
Chess (original)	335	37	3196
Chess1000 × 12	32	12	1000
Chess2000 × 12	63	12	2000
Chess3000 × 12	95	12	3000
Connect (original)	9039	43	67,557
Connect1000 × 12	34	12	1000
Connect2000 × 12	67	12	2000
Connect3000 × 12	100	12	3000

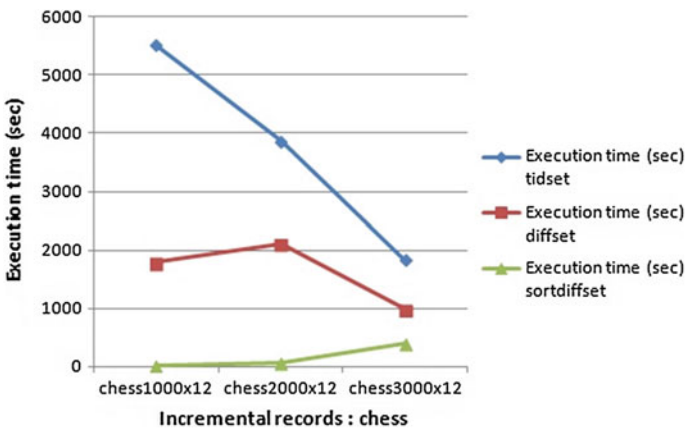


Fig. 5 Chess with eclat

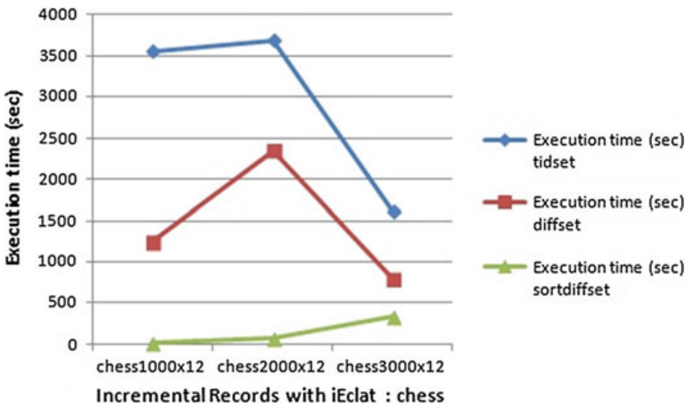


Fig. 6 Chess with incremental-Eclat

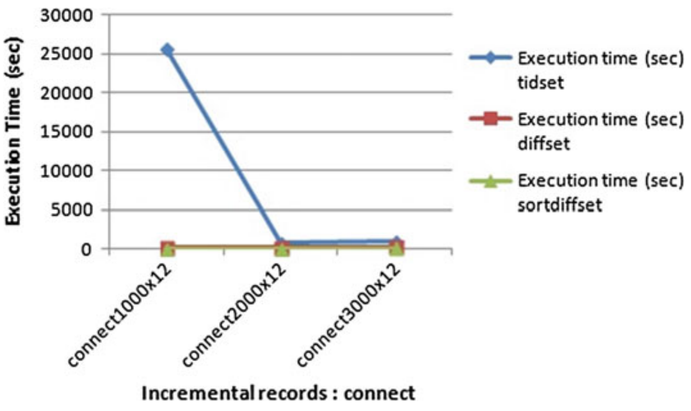


Fig. 7 Connect with eclat

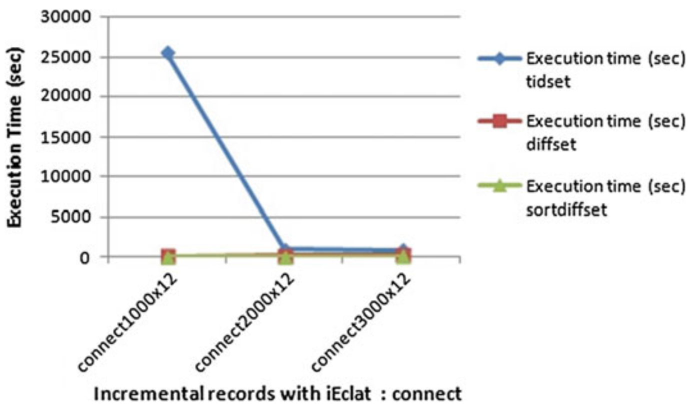


Fig. 8 Connect with incremental-eclat

or in transaction records are efficiently conducted with the adoption of MySQL database where the adhock query either addition or deletion of data can be efficiently manipulated through SQL query in phpMyAdmin software.

6 Conclusion and Future Direction

Experimenting ourselves in association rule database mining with the selected benchmark datasets conforms to what the other previous researchers have proven. In this paper, we have successfully adopted a depth first search (DFS) with intersection strategy through Eclat and our proposed algorithms within a benchmark transaction database in mining association rules. The important advantages in database mining that we disclose here are firstly, the ease of indexing mechanism. Secondly the ad hoc query support mechanism and thirdly, is the interoperability and flexibility of data storage to facilitate the altering (either adding or deleting of row/column) in a data table.Our proposed algorithm seems to benefit with dynamic database where data is always incur in volume from time to time. In conjunction with big data explosion and when the database integration method as in [21] needs to be adopted, then the use of this incremental method will give benefits to end users.

Acknowledgment We express our gratitude to MyPhD scholarship under MyBrain15 of Kementerian Pendidikan Malaysia (KPM) and also to UM research grant and UKM research grant from Research Acceleration Center Excellence (RACE) for the financial foundation of this work.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of 20th international conference on very large data bases (VLDB), vol 1215, pp 487–499
2. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. ACM SIGMOD Record 22(2):207–216
3. Abdullah Z, Herawan T, Deris MM (2010) Scalable model for mining critical least association rules. In: Information computing and applications. Springer Berlin Heidelberg, pp 509–516
4. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. ACM SIGMOD Record 29(2):1–12
5. Zaki MJ, Parthasarathy S, Ogihara M, Li W et al (1997) New algorithms for fast discovery of association rules. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (KDD'97), pp 283–286
6. Zaki MJ, Gouda K (2003) Fast vertical mining using diffsets. In: In Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 326–335
7. Shenoy P, Haritsa JR, Sudarshan S, Bhalotia G, Bawa M, Shah D (2000) Turbo-charging vertical mining of large databases. ACM SIGMOD Record 29(2):22–33
8. Trieu TA, Kunieda Y (2012) An improvement for declat algorithm. In: Proceedings of the 6th international conference on ubiquitous information management and communication (ICUIMC'12), vol 54, pp 1–6
9. Hipp J, Güntzer U, Nakhaeizadeh G (2000) Algorithms for association rule mining: a general survey and comparison. ACM SIGKDD Explor Newslett 2(1):58–64
10. Borgelt C (2003) Efficient implementations of apriori and eclat. In: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations (FIMI03)
11. Schmidt-Thieme L (2004) Algorithmic features of eclat. In: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations (FIMI04)
12. Goethals B (2010) Frequent set mining. In: Data mining and knowledge discovery handbook. Springer, pp 321–338
13. Borgelt C, Kruse R (2002) Induction of association rules: apriori implementation. In: Compstat. Springer, pp 395–400
14. Bakar WAWA, Saman MYM, Jalil MA (2014) Mining educational data: a review on student's pattern of behaviours and performances. Int J Adv Comput Sci Appl 4:247–252
15. Zaki MJ (2000) Scalable algorithms for association mining. IEEE Trans Knowl Data Eng 12(3):372–390
16. Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Disc 15(1):55–86
17. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min Knowl Disc 8(1):53–87
18. Yu X, Wang H (2014) Improvement of eclat algorithm based on support in frequent itemset mining. J Comput 9(9):2116–2123
19. Toivonen H (1996) Sampling large databases for association rules. In: Proceeding of the 22nd international conference on very large data bases (VLDB '96), pp 134–145
20. Slimani T, Lazzez A (2014) Efficient analysis of pattern and association rule mining approaches. Int J Inf Technol Comput Sci 6(3):70–81
21. Man M, Rahim MSM, Zakaria MZ, Bakar WAWA (2011) Spatial information databases integration model. In: Manaf AA et al (eds) ICIEIS 2011. Springer, Informatics Engineering and Information Science, pp 77–90
22. Savasere A, Omiecinski ER, Navathe SB (1995) An efficient algorithm for mining association rules in large databases. In: Proceeding of the 21th international conference on very large data bases (VLDB '95), pp 432–444