Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 44 No. 12
December 2023

A Novel Pattern Storage System to Preserve Patterns in a Pattern Database

Khedekar Vilas Baburao[1], Dharmendra Singh Rajput[2*]
[1] School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India
[2*] School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore, India

Abstract: Various data mining techniques nowadays can generate results popularly known as patterns from large data repositories. However, there is no facility or infrastructure, or model to preserve it as persistent storage of patterns. The proposed storage system provides a model to make these patterns persistent. Most organizations are interested in knowledge or patterns rather than raw data or many unprocessed data because extracted knowledge plays a vital role in making the right decision for the organization's growth. In this paper, Frequent Pattern Mining (FPM) algorithms are used to extract large data set patterns. Result comparison was done using Apriori, Fp-Growth, and Eclat. The proposed model of the Pattern Storage System uses a pattern database to make all patterns persistent. Pattern Storage System (PSS) is used to store FPM algorithms' results in the MongoDB NoSQL database. The work proposed in this paper reduces unnecessary processing on row datasets for which patterns are already available. Pattern retrieval comes very easy with minimum time compared to traditional ways of finding patterns. The proposed model uses a unique way of generation of pattern id which is useful for pattern storage as well as pattern retrieval.

Keywords: Frequent Pattern Mining, Pattern Database, NoSQL Database, MongoDB**.**

I. Introduction

Currently, a substantial volume of data is continuously generated by diverse sources. Owing to the exponential growth of data [1], it is estimated that the global data volume would increase twofold every two years. Since 2010, it has already experienced a 50-fold increase. Both data generated by humans and data generated by machines are expanding at a rate that is 10 times faster than typical enterprise data. Additionally, machine-generated data is growing even more rapidly at a rate of 50 times. In today's business environment, the majority of stakeholders are in need of a fast and precise system to extract valuable information from large datasets. This is done with the purpose of making well-informed decisions that will contribute to the growth of their businesses [2], [39].

Data mining and data analytics are extensively utilised in diverse industries, including healthcare [3], tourism [4], manufacturing, electricity supply [5], financial services [6], railway safety management [7], education [8], wireless networking [9], [41], and quality monitoring for web services [10], as well as not-for-profit organisations [11].

Users cannot successfully utilise the vast amount of data directly, and significant data insights may not be immediately extrapolated from casual observation [12]. Conventional databases are inadequate for storing and handling this vast dataset; therefore, it is stored in data warehouses for subsequent analysis and utilisation. Further investigation is required to uncover concealed insights and obtain valuable patterns from this data [13].

Data mining techniques play a crucial role in extracting knowledge, usually referred to as patterns, from unprocessed databases. These methodologies yield outcomes like as association rules, clusters, decision trees, and other frameworks that aid in describing unprocessed data [14], [34]. Frequent Pattern Mining (FPM) is an essential method that may detect repeating connections among various things in the data [40], and express them as association rules [15]. Functional pattern mining (FPM) is essential for executing a wide range of data mining tasks. [15]. Many of the researchers proposed algorithms to find association rules like Apriori [16], Fp-Growth [17], Eclat [18], Tree Projection [19], COFI [20], TM [19], P-Mine [21], LP-Growth [22], Can-Mining [23], EXTRACT [24].

The resulting patterns are succinct and convey a full semantic representation of the source material [25], [35]. In the realm of databases or data warehouses, the extracted patterns are not considered to be enduring entities. The current

Database Management Systems are insufficient in terms of their strength and adaptability to handle this novel form of knowledge. Therefore, a specialised management system is needed that can effectively model and store patterns" [12]. Extensive study in data mining has resulted in the discovery of numerous complex patterns. It utilises a unique technique to effectively handle these patterns for future study. The Pattern Storage System functions as a centralised database for storing the information derived from various algorithms. These technologies enable the efficient comparison, querying, and storage of the Pattern in order to retrieve information or patterns as required" [14]. Patterns should be represented, stored, manipulated, and retrieved in a manner that is analogous to how data is handled in conventional database management systems (DBMS) [13], [42]. Due to the limitations of DBMS in handling data with complex meaning, it is necessary to utilise a distinct database management system to efficiently manage such data.

In response to the unpredictable nature of patterns, the authors have suggested a pattern warehouse model with a specific architectural design. Nevertheless, the problem remains if patterns continue to vary in response to changes in input datasets. The proposed storage system presents a paradigm that guarantees the durability of these patterns in a NoSQL database, so resolving the challenge at hand.

The paper is segmented into significant subsections. 1. The Literature Survey entails a comprehensive examination of association rules mining algorithms such as Apriori, FP Growth, and Eclat. It also includes a comparison analysis of these association rule mining algorithms. The background of the study includes essential aspects about the production and storage architecture of patterns. 3. The suggested design provides a detailed description of the pattern storage architecture. 4. The Result and Discussion section provides details regarding the duration of pattern generation and the time necessary for pattern storage in the pattern database.

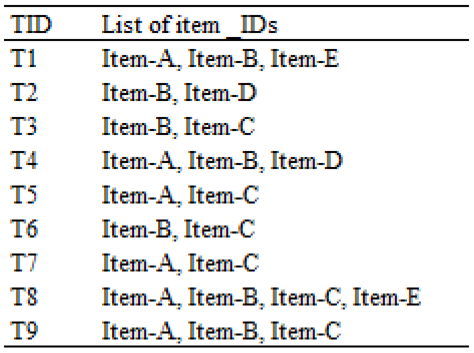
II. Literature Survey

Currently, there exists a variety of techniques that can be used to extract significant insights, sometimes known as patterns, from large datasets. The subpoints A, B, C, and D provide further clarification on the previous study undertaken in this topic. Subpoint D includes a table that provides a comparison of several FPM algorithms developed by different scholars.

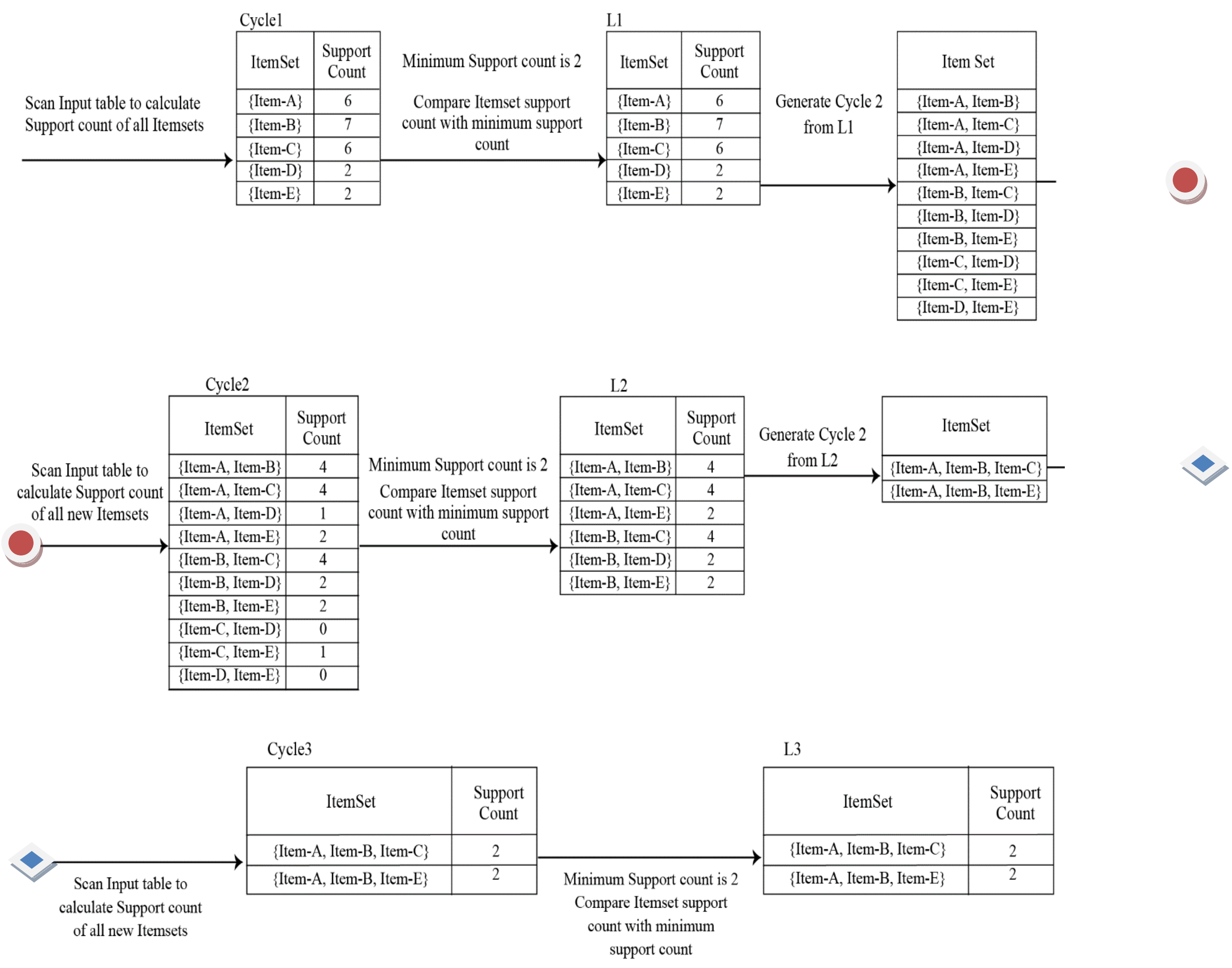
A. The Apriori Algorithm

The Apriori algorithm [16] generates item sets that recur frequently in order to establish association rules. The method utilises a recursive strategy to detect sets of (K + 1) items from all sets of k items. Transactional data refers to the information related to the acquisition of different products in various transactions, as shown in Table I. In order to identify frequent item sets, it is necessary to scan the complete database and compute the support of each item set. Only those item sets that above the minimal support level are selected. This method continues until the entire database has been examined and no further frequent item-sets can be recognised.

Table 1. **Sample Transactional Data**



TID	List of item _IDs
T1	Item-A, Item-B, Item-E
T2	Item-B, Item-D
T3	Item-B, Item-C
T4	Item-A, Item-B, Item-D
T5	Item-A, Item-C
T6	Item-B, Item-C
T7	Item-A, Item-C
T8	Item-A, Item-B, Item-C, Item-E
T9	Item-A, Item-B, Item-C


Scan Input table to calculate Support count of all Itemsets
Cycle1
Minimum Support count is 2
Compare Itemset support count with minimum support count
L1
Generate Cycle 2 from L1
Scan Input table to calculate Support count of all new Itemsets
Cycle2
Minimum Support count is 2
Compare Itemset support count with minimum support count
L2
Generate Cycle 2 from L2
Scan Input table to calculate Support count of all new Itemsets
Cycle3
Minimum Support count is 2
Compare Itemset support count with minimum support count
L3

Cycle1

ItemSet	Support Count
{Item-A}	6
{Item-B}	7
{Item-C}	6
{Item-D}	2
{Item-E}	2

L1

ItemSet	Support Count
{Item-A}	6
{Item-B}	7
{Item-C}	6
{Item-D}	2
{Item-E}	2

Item Set
{Item-A, Item-B}
{Item-A, Item-C}
{Item-A, Item-D}
{Item-A, Item-E}
{Item-B, Item-C}
{Item-B, Item-D}
{Item-B, Item-E}
{Item-C, Item-D}
{Item-C, Item-E}
{Item-D, Item-E}

Cycle2

ItemSet	Support Count
{Item-A, Item-B}	4
{Item-A, Item-C}	4
{Item-A, Item-D}	1
{Item-A, Item-E}	2
{Item-B, Item-C}	4
{Item-B, Item-D}	2
{Item-B, Item-E}	2
{Item-C, Item-D}	0
{Item-C, Item-E}	1
{Item-D, Item-E}	0

L2

ItemSet	Support Count
{Item-A, Item-B}	4
{Item-A, Item-C}	4
{Item-A, Item-E}	2
{Item-B, Item-C}	4
{Item-B, Item-D}	2
{Item-B, Item-E}	2

ItemSet
{Item-A, Item-B, Item-C}
{Item-A, Item-B, Item-E}

Cycle3

ItemSet	Support Count
{Item-A, Item-B, Item-C}	2
{Item-A, Item-B, Item-E}	2

L3

ItemSet	Support Count
{Item-A, Item-B, Item-C}	2
{Item-A, Item-B, Item-E}	2

Figure 1. Testing data- load current (amperes)

Although the Apriori algorithm demonstrates commendable performance, it is important to recognise its considerable limits. The approach requires the repeated examination of the full database for each iteration [26]. In addition, processing large input datasets requires a significant quantity of internal memory [1] [27]. In order to tackle these difficulties, scientists have devised innovative algorithms such as AprioriTID, AprioriHybrid, MR-Apriori, and HP-Apriori with the aim of minimising the duration of execution. Nevertheless, the processing time needed to analyse the existing patterns remains constant.

An enduring resolution is necessary to eradicate the necessity for iterative database scanning and to minimise the data kept in internal memory. The Pattern Storage System (PSS) provides a practical answer to this problem. PSS stores previously formed patterns in the NoSQL database, which reduces the need for repetitive scanning of the database and minimises the internal memory needed for creating new patterns.

B. Eclat Algorithm

The Eclat technique, short for Equivalence class transformation, is utilised to detect common itemsets by employing a vertical data format, as demonstrated in Table III. This algorithm employs a grouping mechanism to categorise transactions involving related items, hence minimising the need for repetitive scanning of the full database. The Eclat method converts the input data format from horizontal to vertical, as shown in Table III. After the transactions from all k-item sets are intersected, (k+1)-item sets (Frequent) are formed.

Table 2 **Item sets-1 in Vertical Data Format [27]**

ItemSet	TIDSet
{Item-A}	{T1, T4, T5, T7, T8, T9}
{Item-B}	{T1, T2, T3, T4, T6, T8, T9}
{Item-C}	{T3, T5, T6, T7, T8, T9}
{Item-D}	{T2, T4}
{Item-E}	{T1, T8}

Table 3 Item sets-2 in Vertical Data Format [27]

ItemSet	TIDSet
{Item-A, Item-B}	{T1, T4, T8, T9}
{Item-A, Item-C}	{T5, T7, T8, T9}
{Item-A, Item-D}	{T4}
{Item-A, Item-E}	{T1, T8}
{Item-B, Item-C}	{T3, T6, T8, T9}
{Item-B, Item-D}	{T2, T4}
{Item-B, Item-E}	{T1, T8}
{Item-C, Item-E}	{T8}

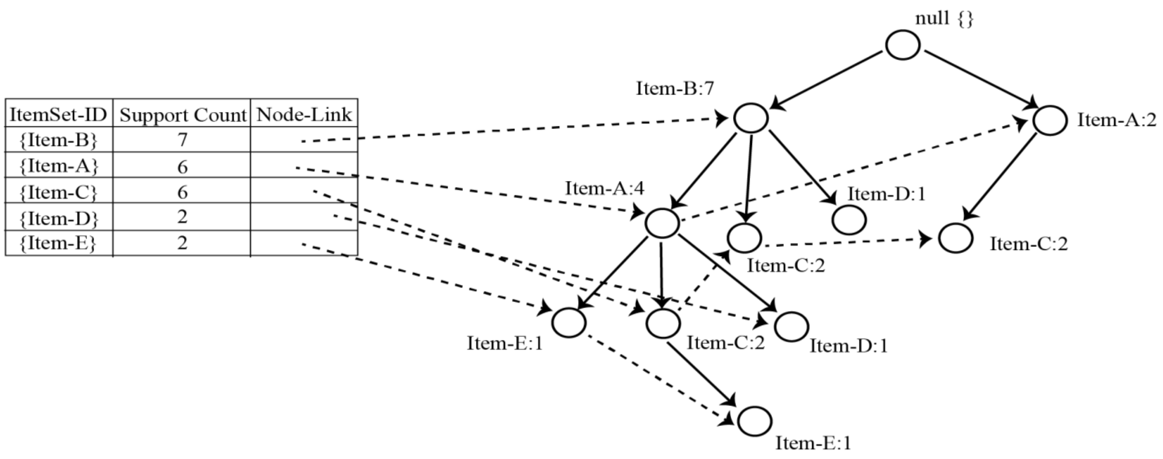
Table 4 Item sets-3 in Vertical Data Format [27]

ItemSet	TIDSet
{Item-A, Item-B, Item-C}	{T8, T9}
{Item-A, Item-B, Item-E}	{T1, T8}

In this approach, the first iteration entails a comprehensive examination of the entire database. Subsequent scans to find frequent item groupings are unnecessary. Due to the vertical data format, support counts per itemset are calculated only in the initial iteration, reducing the necessity of recalculating the support count of an item.

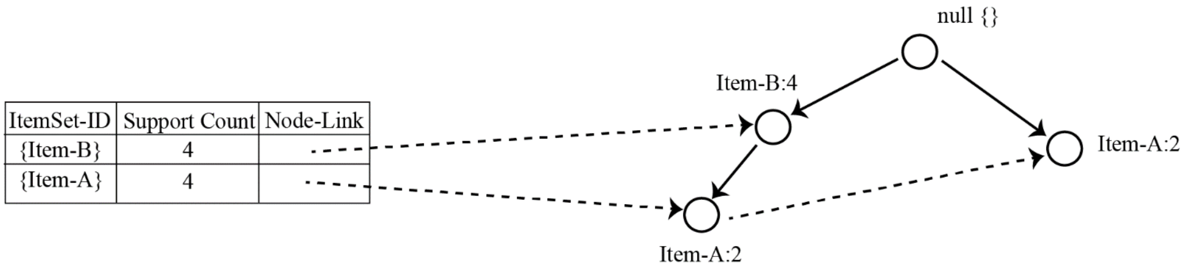
C. FP Growth Algorithm

FP-Growth is an association rule mining technique that discovers item sets without incurring fees for generating regular candidates [28]. This strategy use a two-pass methodology to discover recurring patterns. During the first iteration, the FP Growth Algorithm assesses the frequency of numerous instances of an item in the transactional datasets and keeps track of this frequency in a data structure called the 'Header Table.' In the second iteration, the method creates an FP-tree structure by placing transactions one by one into a tree [36]. By utilising a Division and Conquer strategy, this method preserves data regarding the correlation between regularly compressed elements within a pattern tree. Utilising the hierarchical arrangement of the tree effectively addresses the difficulty of regularly recognising patterns. Figure 4 displays the Conditional FP-tree specifically linked to node I3, serving as an illustrative example. Figure 3 provides comprehensive details about all conditional FP-Trees. [27].



ItemSet-ID	Support Count	Node-Link
{Item-B}	7	
{Item-A}	6	
{Item-C}	6	
{Item-D}	2	
{Item-E}	2	

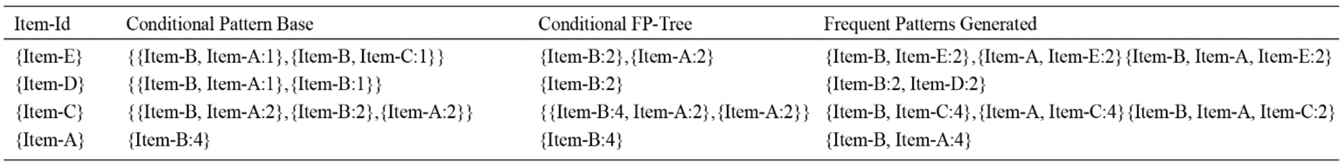
Figure 2. Frequent pattern Tree (FP-Tree)



ItemSet-ID	Support Count	Node-Link
{Item-B}	4	
{Item-A}	4	

Figure 3. Conditional FP-Tree associated with Node 13

Table 5 Conditional Pattern Base and Conditional Fp-Tree [27]



Item-Id	Conditional Pattern Base	Conditional FP-Tree	Frequent Patterns Generated
{Item-E}	{{Item-B, Item-A:1},{Item-B, Item-C:1}}	{Item-B:2},{Item-A:2}	{Item-B, Item-E:2},{Item-A, Item-E:2}{Item-B, Item-A, Item-E:2}
{Item-D}	{{Item-B, Item-A:1},{Item-B:1}}	{Item-B:2}	{Item-B:2, Item-D:2}
{Item-C}	{{Item-B, Item-A:2},{Item-B:2},{Item-A:2}}	{{Item-B:4, Item-A:2},{Item-A:2}}	{Item-B, Item-C:4},{Item-A, Item-C:4}{Item-B, Item-A, Item-C:2}
{Item-A}	{Item-B:4}	{Item-B:4}	{Item-B, Item-A:4}

As a result, this approach results in a decrease in the cost of identifying common patterns. Nevertheless, the FP-Growth algorithm exhibits a significant time overhead when employed on large input datasets. There are many techniques available for detecting repeating patterns in raw input datasets. Below is a compilation of many algorithms, including the TreeProjection algorithm, COFI algorithm, TM algorithm, P-Mine algorithm, LP-Growth method, Can-Mining algorithm, EXTRACT algorithm, and others.

D. *Comparisons of Frequent Pattern Mining Algorithms*

Table 6 **Comparison of Frequent Pattern Mining Algorithms [29]**

FPM Algorithm	Technique Used	Advantages	Disadvantages
Apriori	An iterative level-based search mechanism	k-item sets uses a repeated level-based search technique to find (k + 1) –item sets	If datasets are more than k-item sets, it creates multiple sets of candidates and scans the database frequently to generate the support count for the item sets.
FP-Growth	Divide-and-conquer technique	Keeps all association information of item sets, which reduces the searching time of data.	The processing time to create FP-Tree becomes more if the input data set size is large.
EClaT	Vertical data format technique	Repetitive scanning of the entire database is not required.	In the case of a large input dataset, it needs more internal memory and processing time.
TreeProjection	Different searching techniques such as breadth-first, depth-first, or a mixture of the two.	It requires lesser time because the algorithm searches the frequent item sets, which are a part of a subset for all transactions.	Various representations of the lexicographic tree become a limitation in the form of efficiency in memory consumption.
COFI	Pruning method used to constructs short trees from the Frequent Pattern(FP)-Tree	The pruning method is used to minimize the use of memory-space by creating small COFl-Tree	The performance of algorithms reduces in sparse databases when the threshold value is low for minimum support. Its performance depends on the threshold value.
TM	representation of vertical data like the EClaT.	It saves the intersection time for finding frequent item sets by compressing the item sets into a list of transactions.	The processing speed of this algorithm is slower than the FP-Growth algorithm.
P-Mine	A parallel disc-based approach to multi-core processors.	Optimizes scalability and performance by executing frequent	Algorithms can only be optimized at the maximum level when there are multiple cores available in the

		mining of items in parallel with multiple processors	processors.
LP-Growth	Linear Prefix Tree (LP-Tree)	Quickly generates LP-Tree in the manner as a group of array operations are used to create various nodes concurrently	Memory needs to be released continuously as the items from the transaction saved in various LPNs.
Can-Mining	Incremental manner of Canonical-Order Tree (Can-Tree)	When the threshold value for the minimum support is high, then it performs better than the FP-Growth algorithm.	If the minimum support for the threshold value is too low, then mining time is high.
EXTRACT	Galois lattice a mathematical concept	More than 300 items and mines 10 attributes with an execution time	If the data set changes, then the algorithm must be repeated to mine a new set of items.

Table 7 **Runtime of Different Horizontal Layout Algorithms [29]**

Algorithm	Transaction size (MB)	Threshold	Execution time (in Sec)
Apriori	30	1.5	15.9
SETM	30	1	114
AprioriTID	30	1.5	150
AprioriHybrid	30	0.75	22.5
FPGROWTH	30	3	31.404
PP-Mine	30	1.18	34.311
COFI	30	3.11	18.8445
DynGrowth	30	5	8.23
PRICES	30	5	450
TFP	30	3	4.1955
SSR	30	1	5.298

The results produced by data mining algorithms, which encompass significant insights or patterns, are ephemeral and immediately deleted after utilisation. A pattern warehouse provides the essential structure for ensuring the constant maintenance of all patterns [30]. The current system provides a solution for pattern generation, classification, and manipulation. "To reveal hidden knowledge and increase the value of these patterns, it is imperative to conduct further investigation into sophisticated and advanced techniques." [13, 31].

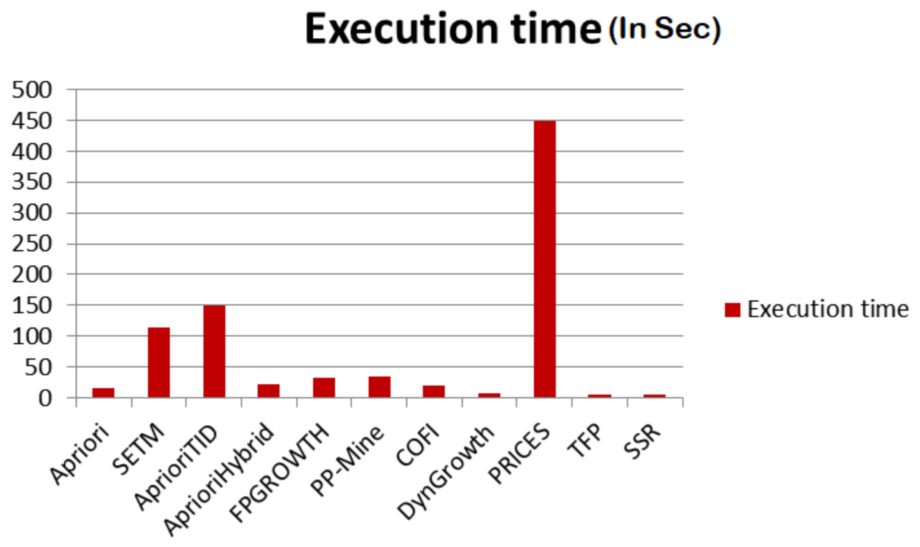


Figure 4. Runtime of different horizontal layout algorithms [29]

The results obtained via data mining algorithms, such as useful insights or patterns, are ephemeral. Once the patterns are employed, they are instantly erased. An architecture for pattern warehousing is implemented to ensure consistent maintenance of all patterns [30]. The existing system deals with the generation, classification, and manipulation of patterns. "Further investigation into more intricate and advanced methodologies is necessary to unveil concealed knowledge and augment the significance of these patterns" [13], [31].

The existing approach is limited by its inability to retain the transient outcomes of data mining techniques. Data mining outcomes are evident in the form of association rules, decision trees, clusters, and other patterns that reveal the distinctive features of the input data [3], [33]. The input data structure for the generated patterns is both semantically meaningful and concise [4], [38]. When patterns become unstable, other programmes require access to them, which necessitates the regeneration of the same pattern. This exacerbates superfluous and duplicative processing expenses.

III. Pattern Storage Architecture

The proposed system provides a solution for this situation. It enables the storage of previously created patterns, either in the pattern database or in the pattern warehouse, depending on the type of pattern. When there are repeated patterns for the same data items, the stored patterns are directly given to the application, avoiding the need for additional processing on raw data sets.

Presently, traditional databases lack the capability to manage and store the vast amount of data, hence requiring the utilisation of data warehouses for storing and subsequent processing. Data mining techniques heavily depend on databases and data warehouses as their main sources of information. The process of extracting latent knowledge from data warehouses leads to the creation of distinctive forms of information, generally referred to as patterns [6]. These patterns are obtained using several data mining algorithms [7], such as association rules, classification, clustering, decision trees, and other approaches employed to extract concealed knowledge from large datasets. Valuable knowledge becomes ephemeral if not stored [8], [32]. Currently, apps that generate patterns operate on a disposable basis. Patterns that are generated can fall into two categories: 'simple' or 'complex' [9]. It is important to note that the nature of these patterns is not permanent [10]. The suggested system presents two approaches to guarantee the long-term storage of all patterns: storing patterns in a pattern database and storing patterns in the pattern warehouse.

A. Architecture

Figure 5 illustrates the operational mechanism of the Pattern Storage System (PSS) using a layered design. This figure illustrates the methodology for guaranteeing the enduring existence of all patterns. The architecture is divided into four separate layers: The system consists of four layers: the Input Data

Layer, the Processing/Data Mining Layer, the PSS Layer, and the Application Layer.

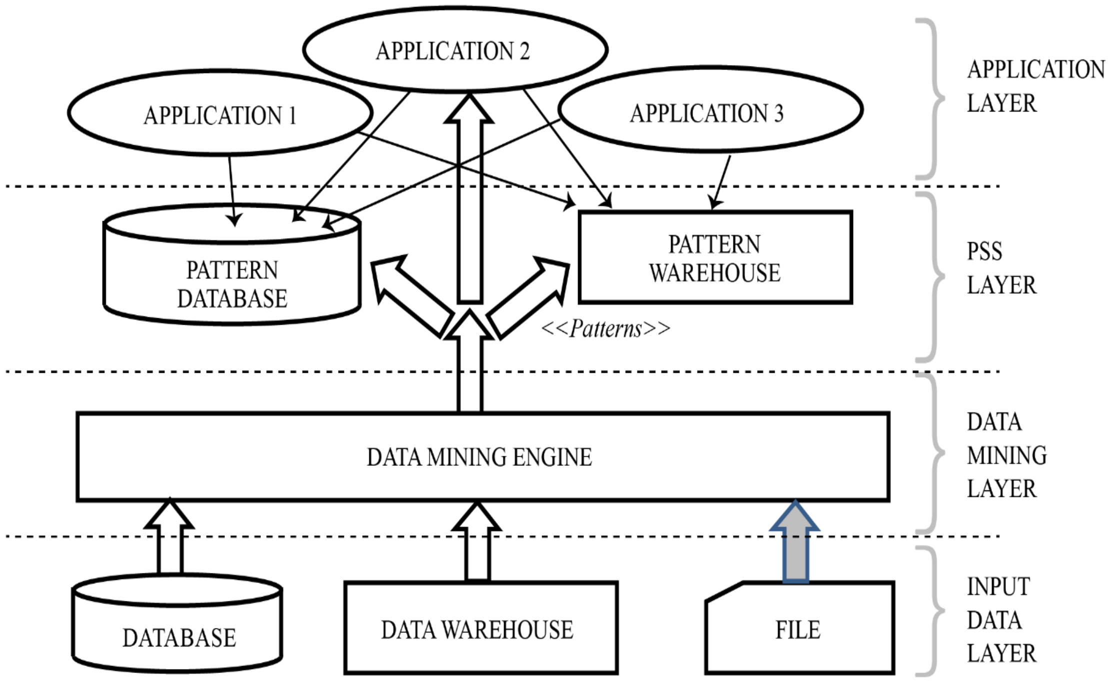


Figure 5. Layered Architecture for Pattern Storage System

1. Input Data Layer:

This layer provides all necessary info to the layer above it. Raw or unprocessed data is used as an input for different data mining techniques and can be found in databases, data warehouses, and files. The main purpose of this layer is to guarantee the availability of all essential data for the next stage. The accuracy of the outcomes relies on the quality of the dataset and the correctness of the data mining algorithm.

2. Processing / Data Mining Layer

Data mining is the process of obtaining valuable information from large databases, data warehouses, and files. Popular data mining techniques encompass classification, association rule mining, clustering, and decision trees. Every technique produces a variety of patterns depending on the algorithms employed and the specifications of the customer. The Processing Layer receives extensive datasets from the Input Data Layer, applies several data mining algorithms to analyse them, and produces patterns according to customer requirements.

3. Pattern Storage System (PSS) Layer

After the production of patterns and the fulfilment of client requirements, the inherently unstable nature of patterns results in their automatic obliteration. Therefore, if another customer has the same request for the same dataset, there is no provision to use patterns that were developed before. The entire process of generating patterns must be repeated, which places an unnecessary demand on the processing unit. To resolve this issue, it is necessary to ensure that all created patterns are stored permanently. The created patterns can be classified as either static or dynamic. The Pattern Storage System (PSS) offers two kinds, specifically Storing Patterns in NoSQL Databases and Pattern Warehouses, to guarantee the long-term existence of patterns.

a) Store Patterns in NoSQL Database:

Variable patterns experience modifications as the input dataset is refreshed with the most recent values. The variable patterns are stored in databases for the convenience of updating them whenever updates are made. NoSQL databases are a suitable choice for assuring the survival of all patterns in data mining findings, as they can accommodate the lack of a defined structure.

b) *Store Patterns in Pattern Warehouse:*

The pattern warehouse functions as a storage facility for established patterns. The patterns produced by different data mining techniques are kept in the pattern warehouse for later examination. The inclusion of a particular pattern in the pattern warehouse is contingent upon its similarity to existing patterns, and the patterns are categorised and stored according to their respective types.

4. Application layer:

Like the view component in the Model-View-Controller (MVC) architecture, this layer is linked to one or more applications. It has the capability to immediately obtain suitable outcomes from the Data Mining layer, bypassing the PSS System, as well as from the Pattern Database and the Pattern Warehouse.

B. Pattern Warehouse

The pattern warehouse facilitates the long-term storing of patterns. Invariant patterns, derived from past data, remain unaltered under particular circumstances. The pattern warehouse provides the ability to store comparable patterns in close proximity. Furthermore, it offers segregated compartments for the outcomes of different data mining methodologies.

P = { P1, P2, P3, …. Pn}   // P is pattern
D = {D1,D2,D3,….,Dn}   // D is a data input
P = PSS (D)

If *D* is Historical / not change time to time then Patterns are fixed then pattern *P* will be stored in the Pattern Warehouse.

C. Pattern Database

The pattern database serves to temporarily store patterns, which may undergo changes over time due to alterations or updates in the source datasets. The proposed system allows for the storage of variable (frequently changing) patterns in the pattern database using a NoSQL database such as MongoDB. The choice of the specific NoSQL database is contingent upon the type of patterns being handled. For instance, in the case of association rules generating associated items and quantities, such as "Milk-Butter-Toast->15," the pattern name "Milk-Butter-Toast" and the resulting quantity of 15 are stored. Here, the pattern name functions as the "Key," and the quantity serves as the "Value." Therefore, a Key-value pair NoSQL Database or a Document Oriented NoSQL database would be suitable for storing the aforementioned patterns.

P = { p1, p2, p2, …. Pn}   nth number of patterns
D = {D1,D2,D3,….,Dn}   // D is a data input
P = Algorithm (D)

If *D* is not historical / change from time to time then Patterns P are variable then store in the Pattern Database.

D. Advantages of PSS

PPS offers several advantages, such as a significant decrease of over 80% in processing time, which depends on the number of patterns stored in the database. It also requires minimal internal space for generating new frequent patterns, reduces algorithmic complexity, ensures secure storage for all generated patterns, and enables rapid computation of results.

E. Database Selection for Storage System

The association rule algorithm produces diverse frequent patterns, necessitating the use of a NoSQL database for storing due to the lack of a predefined pattern structure. Unlike relational databases, NoSQL databases are the preferable choice for managing input in any format and sequence due to their ability to accommodate data input without requiring a predefined format.

F. Disadvantages / Drawbacks of SQL Databases

SQL databases have restrictions in terms of data type support, necessitating the use of predetermined data input formats and sequences. Moreover, their ability to scale is lower compared to NoSQL databases, particularly when dealing with huge input datasets. SQL database performance decreases as the volume of input data grows, while NoSQL databases are not impacted by the size of the input dataset, ensuring efficient database operations.

G. MongoDB NoSQL Database

The objective of the proposed system is to reduce algorithmic complexity, decrease processing time for large input datasets, and minimise needless internal memory consumption. It accomplishes this by employing two primary approaches: firstly, by retaining previously generated patterns in a durable storage system (NoSQL Database), and secondly, by updating previously stored patterns using the MAP-REDUCE technique as needed. The table below presents a comparative analysis of MySQL, MongoDB, and other NoSQL databases.

***Table 8* Comparison of MySQL, MongoDB and Other NoSQL databases [25]**

Parameters	**MySQL**	**Other NoSQL DB**	**MongoDB**
ACID Property	Yes	No	Yes
Rich and flexible data model	No	Partial- Only simple data structures support the flexibility of schema.	**Yes**
Schema governance	Yes	No	Yes
Powerful aggregations, expressive joins, graphs queries, faceted search	Yes	No	Yes
Native, Idiomatic language drivers	No	No	**Yes**
Horizontal scale-out	No	Partial: Controls on data locality not available	**Yes**
BI and Analytics ready	Yes	No	Yes
Enterprise-grade security, reliable management tools	Yes	No	Yes
Cloud Service (Database as a service)	Yes	No	Yes

IV. Pattern Storage in NoSQL Database

A. Architecture

The following architecture shows a building blogs of the proposed model. The proposed model accepts input data from various data sources as mentioned in the diagram, minimum support and minimum confidence.

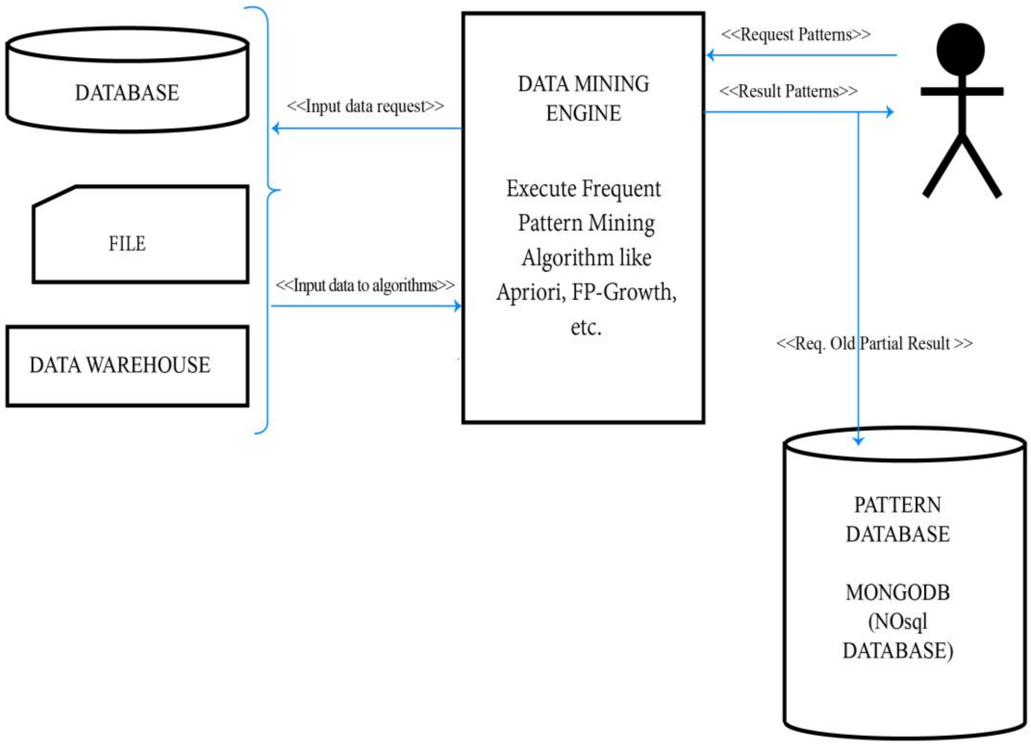


Figure 6. Proposed model for Pattern Storage System

The database, file and data warehouse are a storage media for all row datasets. Data mining Engine is the brain of the pattern storage system which accepts input data and generates patterns. Pattern database is a storage unit of the proposed model. Initially, data is fed into the FPM algorithm, which generates patterns based on the input parameters. The input parameters are the Name of the file, minimum support, minimum confidence, etc. The generated patterns will be preserved or store in a MongoDB NoSQL Database using a pattern storage system with a unique pattern id. Section IV. B shows how pattern id will be generated?

B. Mathematical Representation of PID

PID = Index for patterns
FN = Name of the file
MS = Minimum Support
MC = Minimum Confidence
FD = From Date
TD = To Date

$PID = FN + MS + MC + FD + TD$

Name of the file, minimum support, and minimum confidence, From-Date, To-Date are the parameters to understand the existence of already generated patterns.

C. Algorithm of the proposed model

Algorithm 1: Pattern Storage in Pattern Database

Input: Supermarket data which consist of date, time, customer id, and all grocery items with quantity and price
Output: Patterns (association rules) generated and preserve in pattern database.

```
1:  Start
2:  $Pid = Fn + Ms + Mc + Fd + Td$   // Pid required to store result in NoSQL Database
3:  $I = \{ i_1,i_2,i_3,....,i_n\}$   //Input data items
4:  $T= \{t_1,t_2,t_3,....,t_n\}$   //Transactions
5:  $P=\{p_1,p_2,p_3,....,p_n\}$   //Resultant Patterns
6:  tPid = find(Pid);
7:  IF $tPid \neq Pid$ THEN
8:  P = Algorithm(I,Ms,Mc);  //Generate patterns by Apriori, Eclat, FP-Growth etc
9:  D = store(P,Pid);   //Store patterns in Database (D) where Key=Pid, value=Patterns P
10: ELSE
11: Fetch patterns from the database where id = Pid
12: End
```

As shown in Algorithm 1, the pattern ID functions as a unique identifier for all freshly generated patterns. The construction of this ID involves the amalgamation of the file name, minimum support, minimum confidence, and the from date and to date. In Algorithm 1, 'I' represents the input datasets, 'T' represents all transactions, and 'P' defines the set of patterns. The purpose of the 'find(Pid)' function is to validate the existence of the provided 'Pid' in the NoSQL database by taking it as an input parameter. If the pattern ID is not present in the NoSQL database, it signifies that the pattern corresponding to the given parameters has not been created. Here, the system invokes a suitable algorithm to create the patterns and then saves them in the NoSQL database, along with their respective pattern ID. Conversely, if the pattern ID exists in the NoSQL database, the system directly obtains all patterns linked to that specific pattern ID.

V. Results and Discussion

The input dataset for this research consists of supermarket data that includes information such as the date, time, customer ID, and the amounts and prices of all grocery goods. This dataset is quite valuable for extracting correlations between two or more components. The suggested architecture aims to save association rules (patterns) in the pattern database, reducing duplicate processing on datasets where the patterns have already been developed and are accessible in the pattern storage system.

A. Time required for storage

The following table shows the execution time needed for an algorithm to generate patterns. It depends on the size of input data given to an algorithm. Various researchers have performed multiple experiments to evaluate the efficiency of

Frequent Pattern Mining (FPM) algorithms in terms of execution time and memory usage when mining frequent item sets from a data set.

Table 9 to 15 display time required to execute an algorithm and store the pattern database patterns and total execution time. Table value shows that the execution time for an algorithm is different for different support and confidence and time required to store patterns in pattern database is also different for different support and confidence.

Table 9 **Time require for execution and storage for 10 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	0.006913	0.0065011	0.0134141
0.1	0.5	0.046891928	0.036593199	0.083503962
0.3	0.5	0.006407976	0.001573801	0.007990837
0.2	0.6	0.006868839	0.005994081	0.012897015
0.1	0.6	0.010102034	0.036376953	0.046500921
0.3	0.6	0.008606911	0.001837969	0.010458946

Table 10 **Time require for execution and storage for 100 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	0.010266	0.007652	0.017918
0.1	0.5	0.01058507	0.033909798	0.044523001
0.3	0.5	0.007591963	0.001481056	0.0090909
0.2	0.6	0.009322166	0.005419016	0.014793158
0.1	0.6	0.032276869	0.032276869	0.042598009
0.3	0.6	0.007725	0.001298189	0.009031057

Table 11 **Time require for execution and storage for 1000 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	0.015124	0.005512	0.020636
0.1	0.5	0.026736975	0.031322956	0.05811286
0.3	0.5	0.01424408	0.001650095	0.015913963
0.2	0.6	0.015249014	0.004982948	0.020246983
0.1	0.6	0.026834965	0.036405087	0.063281059
0.3	0.6	0.014594078	0.001335859	0.015939951

Table 12 **Time require for execution and storage for 10000 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	0.22868	0.004312	0.232992
0.1	0.5	0.30761385	0.036018133	0.343665838
0.3	0.5	0.083166122	0.001574993	0.084755182
0.2	0.6	0.095443964	0.00454998	0.100008965
0.1	0.6	0.198750019	0.034356117	0.233121157

0.3	0.6	0.085319996	0.001767874	0.087103844

Table 13 **Time require for execution and storage for 100000 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	1.656647	0.005782	1.662429
0.1	0.5	2.071568966	0.035224915	2.106809855
0.3	0.5	0.803340197	0.001611948	0.804961205
0.2	0.6	0.943425179	0.005573988	0.949010134
0.1	0.6	2.000555038	0.030066013	2.030660868
0.3	0.6	0.772572041	0.001523018	0.774102926

Table 14 **Time require for execution and storage for 500000 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	8.074649	0.006062	8.080711
0.1	0.5	10.57745409	0.034883022	10.61238408
0.3	0.5	4.233985901	0.001554966	4.235555887
0.2	0.6	5.031282902	0.00455904	5.035853863
0.1	0.6	10.62795019	0.030773878	10.65874505
0.3	0.6	0.010827065	0.001610994	4.051609039

Table 15 **Time require for execution and storage for 1000000 transactions**

Support	**Confidence**	**Time in sec. for rule generation**	**Time in sec. to store in database**	**Total time in sec. for execution**
0.2	0.5	14.688235	0.004801	14.693036
0.1	0.5	20.68601894	0.03431201	20.72034597
0.3	0.5	8.75031805	0.001667023	8.751995087
0.2	0.6	9.780344963	0.005496025	9.785851955
0.1	0.6	22.16622782	0.036239147	22.20249105
0.3	0.6	8.419794083	0.00159502	8.42140007

Table 9 displays the time required for execution and storage for ten transactions. The limited transactions were purposefully chosen to test the proposed model's performance on a small dataset. It was discovered that the time required to compute patterns is less and comparatively the same with different parameters, and the time necessary to save generated patterns in pattern database is almost the same with varied support and confidence. For various levels of assistance and confidence, the total time necessary to complete the process ranges from 0.007990837 to 0.083503962.

Table 10 displays the time required for execution and storage for 100 transactions. The number of transactions was raised to test the proposed model's performance on different size datasets. It was discovered that the time required to compute patterns is less and comparatively the same with different parameters, and the time necessary to save generated patterns in pattern database is almost the same with varied support and confidence. For various levels of support and confidence, the total time required to complete the process ranges from 0.009031057 to 0.044523001.

Table 11 displays the time required for execution and storage for 1000 transactions. The number of transactions was raised to test the proposed model's performance on different size datasets. It has been discovered that the time required to compute patterns is less and comparatively the same with different parameters, and that the time necessary to save generated patterns in the pattern database is virtually the same with varied support and confidence. For various levels of support and confidence, the total time necessary to complete the process ranges from 0.015913963 to 0.063281059.

Table – 12 shows the time required for execution and storage for 10000 transactions, The limit of transactions increased to check the working of the proposed model on different size datasets. It has been found that time required to calculate patterns has slightly increased and comparatively the same with different parameters and time required to save generated patterns in pattern database is also nearly equal to same with different support and confidence. Total time required to perform entire process is between 0.774102926 to 2.106809855 for different support and confidence. Table – 13 shows the time required for execution and storage for 100000 transactions, The limit of transactions increased to check the working of the proposed model on different size datasets. It has been found that time required to calculate patterns has slightly increased and is comparatively same with different parameters and time required to save generated patterns in pattern database is also nearly equal to same with different support and confidence. Total time required to perform entire process is between 0.803340197 to 2.071568966 for different support and confidence. Table – 14 shows the time required for execution and storage for 500000 transactions, The limit of transactions increased to check working of proposed model on different size datasets. It has found that time required to calculate patterns has majorly increased and comparatively same with different parameters and time required to save generated patterns in pattern database is also nearly equal to same with different support and confidence.

The Table – 12 shows the time required for execution and storage for 1000000 transactions, The limit of transactions increased to check working of proposed model on different size datasets. It has been found that the time required to calculate patterns has majorly increased and is comparatively the same with different parameters and the time required to save generated patterns in pattern database is also nearly equal to the same with different support and confidence. The total time required to perform entire process is between 4.235555887 to 22.20249105 for different support and confidence.

After various experiments on different data sets and different support, confidence value, we found that whenever the input data size is less, then the execution time for pattern generation is similar to the time required to store in the database. But as soon as input data size increases, the time required to execute an algorithm also increases and the time required to store patterns in the database remains the same or slightly changes. This demonstrates that whenever amount of the data increases, the time required to mine the frequent item sets will inevitably increase. Hence it has experimentally proved that "Whenever input data size increases then the pattern generation time also increases. However, if pattern storage system is used to preserve patterns in pattern database then it will reduce unnecessary processing time and pattern retrieval can be possible within minimum time as compared to traditional way.

VI. Conclusion

The goal of this research is to evaluate the strengths and shortcomings of algorithms in Frequent Pattern Mining (FPM) in order to create a more efficient model that addresses at least one of the stated issues. The investigation uncovered two major issues within FPM. To begin, the extraction of latent patterns in a dataset gets increasingly time-consuming as data amount increases, resulting in high memory usage. To address this issue quickly, a comprehensive solution for all created patterns was designed and implemented, lowering search times for patterns that are already accessible in the NoSQL database. The suggested pattern storage system enables users to accelerate decision-making processes and improve overall system performance, potentially increasing business income.

Several research issues and research fields remain unanswered. The first is concerned with retrieving patterns from the pattern database. Second, it entails the finding of complicated patterns, such as patterns inside patterns. Third, it aims to discover new strategies for creating pattern recognition. Finally, it intends to build query processing and optimisation approaches that include access methods and algebraic features, as well as to establish appropriate structures for linking patterns and databases. Addressing these issues indicates a viable direction for future study in this field.

VII. References

[1] www.insidebigdata.com, "The Exponential Growth of Data", www.insidebigdata.com, 2019.

[2] C.-H. Chee, W. Yeoh, H.-K. Tan, and M.-S. Ee, “Supporting Business Intelligence Usage: An Integrated Framework with Automatic Weighting”, J. Comput. Inf. Syst., Volume- 56, Issue- 4, PP- 301–312, 2016.

[3] J. P. McGlothlin and L. Khan, “Managing evolving code sets and integration of multiple data sources in health care analytics”, in Proceedings of the 2013 international workshop on Data management & analytics for healthcare DARE '13, PP- 9–14, 2013.

[4] F. Rebón, G. Ocariz, J. K. Gerrikagoitia, and A. Alzua-Sorzabal, “Discovering Insights within a Blue Ocean Based on Business Intelligence”, Procedia - Soc. Behav. Sci., Volume- 175, PP- 66–74, 2015.

[5] Qun Qiu, J. A. Fleeman, D. R. Ball, G. Rackliffe, J. Hou, and L. Cheim, "Managing critical transmission infrastructure with advanced analytics and smart sensors", IEEE Power & Energy Society General Meeting, PP- 1–6, 2013.

[6] V. Chang, "The Business Intelligence as a Service in the Cloud", Futur. Gener. Comput. Syst., Volume- 37, PP- 512–534, 2014.

[7] W. P. Lira et al., "A Visual-Analytics System for Railway Safety Management", IEEE Comput. Graph. Appl., Volume- 34, Issue- 5, PP- 52–57, 2014.

[8] R. Haupt, B. Scholtz, and A. Calitz, "Using Business Intelligence to Support Strategic Sustainability Information Management", Proceedings of the Annual Research Conference on South African Institute of Computer Scientists and Information Technologists - SAICSIT, PP- 1–11, 2015.

[9] S. Qaiyum, I. A. Aziz, and J. Bin Jaafar, "Analysis of Big Data and Quality-of-Experience in High-Density Wireless Network", International Conference on Computer and Information Sciences (ICCOINS) , PP- 287–292, 2016.

[10] M. H. Hasan, J. Jaafar, and M. F. Hassan, "Monitoring web services quality of service: a literature review", Artif. Intell. Rev., Volume- 42, Issue- 4, PP- 835–850, 2014.

[11] R. L. Oakley, L. Iyer, and A. F. Salam, "Examining the Role of Business Intelligence in Non-profit Organizations to Support Strategic Social Goals", Hawaii International Conference on System Sciences, PP- 4641–4650, 2015.

[12] I. Bartolini et al., "PAtterns for Next-generation DAtabase systems: preliminary results of the PANDA project.", PP- 293–300, 2003.

[13] M. Terrovitis et al., "Modeling and language support for the management of pattern-bases", Data Knowl. Eng., Volume- 62, Issue- 2, PP- 368–397, 2007.

[14] J. Wang, "Encyclopedia of Business Analytics and Optimization", J. Clean. Prod., Volume- 16, Issue- 15, PP- 1799–1808, 2014.

[15] C.-H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining: a literature review", Artif. Intell. Rev., 2018.

[16] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases", International Conference on Very Large Data Bases, PP- 487–499, 1994.

[17] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00, PP- 1–12, 2000.

[18] M. J. Zaki, "Scalable algorithms for association mining", IEEE Trans. Knowl. Data Eng., Volume- 12, Issue- 3, PP- 372–390, 2000.

[19] R. C. Agarwal, C. C. Aggarwal, and V. V. Prasad, "A Tree Projection Algorithm for Generation of Frequent Item Sets", J. Parallel Distrib.

Comput., Volume- 61, Issue- 3, PP- 350–371, 2001.

[20] M. El-hajj and O. R. Zaiane, "COFI-tree Mining: A New Approach to Pattern Growth with Reduced Candidacy Generation", Workshop on Frequent Itemset Mining Implementations (FIMI2003) in conjunction with IEEE-ICDM, 2003.

[21] E. Baralis, T. Cerquitelli, S. Chiusano, and A. Grand, "P-Mine: Parallel itemset mining on large datasets", IEEE 29th International Conference on Data Engineering Workshops (ICDEW), PP- 266–271, 2013.

[22] G. Pyun, U. Yun, and K. H. Ryu, "Efficient frequent pattern mining based on Linear Prefix tree", Knowledge-Based Syst., Volume- 55, PP- 125–139, 2014.

[23] M. S. Hoseini, M. N. Shahraki, and B. S. Neysiani, "A new algorithm for frequent mining patterns in Can Tree", International Conference on Knowledge-Based Engineering and Innovation (KBEI), PP- 843–846, 2015.

[24] I. Feddaoui, F. Felhi, and J. Akaichi, "EXTRACT: New extraction algorithm of association rules from frequent item sets", IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), PP- 752–756, 2016.

[25] E. Kotsifakos, I. Ntoutsi, and Y. Theodoridis, "Database support for data mining patterns," 10th Panhellenic Conference on Informatics, PCI 2005, Volas, Greece, PP- 14-24, 2005.

[26] M. Han, Jiawei and Kamber, "Data Mining: Concepts and Techniques", Volume- 12, 2011.

[27] J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques", 2012.

[28] C.-H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining: a literature review", Artif. Intell. Rev., Volume- 52, Issue- 4, PP- 2603–2621, 2019, doi: 10.1007/s10462-018-9629-z.

[29] C. H. Chee, J. Jaafar, I. A. Aziz, M. H. Hasan, and W. Yeoh, "Algorithms for frequent itemset mining: a literature review", Artif. Intell. Rev., PP- 1–19, 2018.

[30] J. Dean and S. Ghemawat, "Map-Reduce: Simplified Data Processing on Large Clusters", D Osdi Ieee, Volume- 51, Issue- 1, PP- 107–113, 2004.

[31] S. Mallik, T. Bhadra, and A. Mukherji, "DTFP-Growth: Dynamic Threshold-Based FP-Growth Rule Mining Algorithm Through Integrating Gene Expression, Methylation, and Protein–Protein Interaction Profiles", *IEEE Trans. Nanobioscience*, Volume- 17, Issue- 2, PP- 117–125, 2018, doi: 10.1109/TNB.2018.2803021.

[32] B. Huynh, B. Vo, and V. Snasel, "An Efficient Parallel Method for Mining Frequent Closed Sequential Patterns", *IEEE Access*, Volume- 5, PP- 17392–17402, 2017, doi: 10.1109/ACCESS.2017.2739749.

[33] Md Shamsur Rahim, et al., "A clustering solution for analyzing residential water consumption", Knowledge-Based Systems, Volume 233, 107522, ISSN 0950-7051, 2021, https://doi.org/10.1016/j.knosys.2021.107522
.

[34] Cristina Nica, Victor-Petru Almăşan, Adrian Groza, "FastRCA-Seq: An efficient approach for extracting hierarchies of multilevel closed partially-ordered patterns", Knowledge-Based Systems, Volume 210, 2020, 106533, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2020.106533
.

[35] Dakshi T. Kapugama Geeganage, Yue Xu, Yuefeng Li, "Semantic-based topic representation using frequent semantic patterns", Knowledge-Based Systems, Volume 216, 2021, 106808, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2021.106808
.

[36] Youxi Wu, et al.,"HANP-Miner: High average utility nonoverlapping sequential pattern mining", Knowledge-Based Systems, Volume 229, 2021, 107361, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2021.107361
.

[37] Huan Liu, et. al.,"Memory transformation networks for weakly supervised visual classification", Knowledge-Based Systems, Volume 210, 2020, 106432, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2020.106432
.

[38] D. S. Rajput, R.S. Thakur, G.S. Thakur, "A Computational Model for Knowledge Extraction in Uncertain Textual Data using

Karnaugh Map Technique ”, International Journal of Computing Science and Mathematics, Inderscience. ISSN: 1752-5063 Vol. 7 No 2, pp. 166-176, 2016.

[39] D. S. Rajput, Neelu Khare, “FSS Decision Making Model for Social Networking Sites”, published in International Journal of Social Network Mining, Inderscience, Vol 2(3) pp. 256-266, 2016.

[40] D. S. Rajput, “Review on recent developments in frequent itemset based document clustering, its research trends and applications”, accepted in International journal of data analysis techniques and strategies, InderScience ISSN: 1755-8069, Vol. 11(2), pp. 176-195, 2019.

[41] Arpita Jadhav, Saurabh Rajput, Khedekar Vilas Baburao, Dr. Dharmendra Singh Rajput, "Smart Helmet Using Natural Language Processing, Head Mounted Display And Solar Panel", International Journal Of Scientific & Technology Research Volume 8, Issue 10, 2019, ISSN 2277-8616.

[42] K.V, B. & Rajput, D., A Pattern Storage System using Pattern Warehouse along with Sources of Pattern Generation and Applications. International Journal of Innovative Technology and Exploring Engineering Special Issue, 8(10S):357–362, 2019, doi: 10.35940/ijitee.J1063.08810S19

Author Biographies

First Author Khedekar Vilas Baburao, from Pune, India, Date of Birth: 16th May, 1985, pursuing Ph.D. from SCOPE, VIT, Vellore, India. Completed M.Tech (July 2014) in Computer Science from JNTU, Hyderabad, India. Completed B.E. (May 2009) in Information Technology from Pune University, India. Area of research work is Data mining, Pattern Mining, Pattern Database.

Second Author The Dr. Dharmendra Singh Rajput working as Professor in the School of Computer Science Engineering and Information Systems, VIT, Vellore since June 2014. He has Completed Ph.D. (2014) from NIT, Bhopal, India. His research areas are Data Mining, machine learning, and big data. He has published 35+ reputed Journal Papers, 5 edited books published under reputed publishers, and 17 papers presented in the reputed international conference. He is also a guest editor of various reputed journals. He has received various awards from the Indian Government like DST-SERB, CSIR Travel Grant, and MPCST Young Scientist Fellowship. He is doing the funded project of 80 lakhs which is received from Erasmus + Programme of the European Union with the partner the University of Nottingham UK. He has visited various countries UK, France, Singapore, UAE, China, and Malaysia for academic purposes.