International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 29733-29747

http://www.ripublication.com

An Enhanced Query Manipulations In Temporal Database Using TSQL3MINE Approach

N. Duraimutharasan
Assistant Professor, *Periyar Maniammai University, Vallam, Thanjavur duraibose@pmu.edu*,
Dr.K.Sarukesi
Dean, Kamaraj College of Engineering & Technology, Virudhunagar, *profsaru@gmail.com*

Abstract

The Temporal databases stores historical information's and it also maintains time varying data. SQL3 is normally used to retrieve the temporal data from the database. But adequate support for the temporal applications and query processing are not provided by the standard SQL3. In order to overcome these drawbacks, we incorporate new keywords and syntax to define different versions of temporal operators and functions applied in SQL. After the extension of Temporal SQL3 Mining (TSQL3MINE), it become suitable for performing various query operation related to temporal facts. In this paper, we proposed TSQL3MINE approaches which reduce the query processing with the help of new function and methodologies. The Semantic concepts are used to improve the relevancy calculation in query processing and also to compute the similarity measures. The proposed technique assures that the legacy query results remain the same when other queries run on the DB. The TSQL3MINE method allows easy processing and navigation. The experimental result shows that the proposed TSQL3MINE enhances query manipulations in the temporal database. The proposed method also achieves improved time efficiency, search efficiency, processing speed and accuracy when compared with other existing techniques such as Moving Objects Spatio-Temporal (MOST) data model and Object Role Modeling (ORM) approach.

Index Terms: Relational operators, Semantic keyword, Similarity calculation, Temporal database, Temporal mining, Object oriented SQL.

Introduction

Temporal data are sequences of a primary data type, most commonly categorical and numerical values. Examples of temporal data are event sequences (packet traces,

sensor readings), time series, and temporal databases (i.e. databases with versioning, relations with timestamped tuples). Temporal data are used in variety of fields such as geographical data processing, biomedicine, internet site usage monitoring and financial data forecasting. Temporal databases are the databases that consist of time-stamping information. It contains built-in support for manipulating the data involving time. For eg. a temporal version of SQL. The first step in temporal database is to timestamp the data that allows the distinction of various database states. The temporal database allows timestamping of entities with time periods and property values of the entities. Specifically, the temporal aspects include transaction time and valid time.

Valid time is referred as the time period at which a fact is true with concern to the real world. The transaction time is referred as period at which a fact is kept in the database. There are different forms of temporal databases based on the valid time and transaction time. A rollback database keeps data based on transaction time and a historical database keeps data based on valid time. The bitemporal data are created by combining these attributes. Temporal databases manage and access temporal data by proving some specific features. Some of these features are

- Temporal primary keys, such as non-overlapping period constraints
- System-maintained transaction time
- Temporal constraints, including referential integrity and non-overlapping uniqueness
- Temporal queries at present time, time points in the past or over durations
- Ability to define a valid and transaction time period attributes.

Temporal database is used in many applications where time is needed to maintain the data in the database. The various application where temporal database are efficiently used are heathcare, insurance, finance, personal management and banking. The collection of helpful information from the temporary data is known as Temporal data mining and it is the significant extension for data mining. It is the non-trivial extraction of implicit, useful and previously unrecorded data with an explicit or implicit temporal content from large database. The main aim of temporal data mining is to determine hidden relations between sequences and sub-sequences of events. The determination of relations between sequences of events include three steps: the modeling of data sequence in a suitable form, the definition of similarity measures between sequences, and the application of representations and models to the actual mining problems. A sequence containing a series of nominal symbols from a specific alphabet is normally known as temporal sequence. The time series is a sequence of continuous, real-valued elements. The temporal sequences and time series occur in various domains such as engineering, finance, scientific research and medicine. Usually, the temporal sequences appear with either log-based system monitoring or sensor-based monitoring such as telecommunication. The task of temporal data mining involve association, classification, prediction, characterization, clustering, trend analysis and pattern discovery.

The manipulation and querying operations in the temporal database are not supported by the standard SQL. In this paper, a temporal SQL3 mining (TSQL3MINE) technique is proposed to increase the query manipulations in the temporal database. A temporal extension to the standard SQL3 is provided by using

the new keywords. These keywords define different versions of temporal operators in SQL3. The new keywords enhance the expressions of operators that are used in standard SQL3. The generated objects perform the operations such as natural join, table creation, set, negation and division. The query operations are simplified by the use of new functions and methodologies. The technique reduces the complexity and improves the accuracy. Easy navigation and processing are allowed by the TSQL3MINE approach.

The remaining part of the paper is referred as follows: Section II involves the works related to temporal mining approaches. Section III involves the description of the TSQLMINE3 approach. Section IV involves performance analysis of the TSQL3MINE approach. Section V includes conclusion.

Related Work

A new similarity search interface [1] allowed user to specify a query by placing events on a blank timeline. The similarity-ranked list of results were retrieved. The users could also customize a new similarity measure for event sequences by using four decision criteria. The requirement for the development of fuzzy-temporal relational database model (FTRDM) [2] was investigated. The guideline for representing vague and imprecise information concepts in a conceptual data model were also provided. A new technique for fuzzy itemsets [3] was based on the temporal database manipulations in terms of fuzzy clustering. The main aim of the fuzzy cluster analysis was to segement the given set of data or objects into cluster. The semantics and patterns using fuzzy clustering were also explained. When compared with the other approaches, fuzzy method extracted more hidden information. Temporal data update methodologies [4] considered the coexistence of load and query. The key areas of temporal data warehouse based on the practical experience in database implementation were explained. Many database applications retained previous and current states of data from traceability and accountability perspectives. The performance of load and report queries and the maintenance of views on top of the tables with temporal data were also discussed.

The operational data were retrieved from heterogeneous operational sources and the join operations were performed with inequality predicates. The operations were then loaded in analytical subject area tables while maintaining the transaction lineage. The captured transaction lineage provided subset of data required by different applications based on different time slices. The development of temporal based multimedia data management model [5] was presented. The easy definition, manipulation and querying of temporal database were not allowed by the standard SQL3. The manipulation and querying of temporal facts in SQL3 [6] were simplified by using a model that integrated time in an inherent manner. New keywords and syntax were suggested to define different temporal versions for relational operators and functions utilized in SQL.The model associated the two elements of time in order to satisfy the requirement. The valid time and transaction time monitored the changes (event) and transaction in the database. A new schema to query multimedia data was also provided by using the time elements. The moving objects contained different

features [7] that led to a variety of queries. It is essential to understand the full spectrum of moving object queries before creating an index structure for such objects. The capabilities of moving object queries were explained. The approach defined a taxonomy of moving object queries consisting of perspectives such as location, object perspective, temporal perspective, motion perspective and patterns perspective. The uncertainty modeling in spatial object-relational databases [8] was performed by using the structured query language (SQL). The basic principles of uncertainty modeling by fuzzy sets were applied in the area of spatial databases and geographic information systems (GIS). A spatial database system included types of spatial data and implemented the spatial extension of SQL. The efficient processing of uncertain data required the implementation of fuzzy logic priniples in spatial databases. The challenging issues of spatio-temporal data mining [9] were presented. The STDB extended the conventional spatial database that dealt with only stationary data and inapplicable to moving objects. The ability to analyze the spatial temporal information remained incomplete. The appropriate data types and query languages for time-evolving spatial objects must be provided by STDB.

Some of the spatio-temporal database (STDB) applications included weather forecast, flight control systems, mobile computing etc. These applications recorded object's geographical locations at different timestamps and supported the queries that explore their historical and predictive behaviors. The temporal data model based on probability (BPTM) [10] was used to manage the indeterminacy temporal semantics of indeterminacy data. The tuple-timestamp method stored the temporal data including indeterminacy and determinacy data. The potential information about the indeterminacy data were retrieved by using a new probability method. A general framework for the management of temporal trends [11] involving different granularities considered specific temporal features. The temporal extension of the relational calculus was used to map the relational expressions into plain SQL queries. An enhanced approach was used to analyze frequent temporal itemsets in the database [12]. Initially, the database was partitioned into sub-databases based on the common ending time or common starting time. The number of occurrences of candidate itemsets was then accumulated for each partition. This approach was also used to determine the temporal association rules among transactions within relational databases.

A new load shedding algorithm [13] were suitable for temporal query processing. The accuracy of attribute values are measured based on jaccard coefficient. The PQI interval orders were used to calculate the exactness of the valid time intervals. The challenges in the temporal information retrieval [14] were discussed. The temporal information about the documents could be utilized to develop time-specific information retrieval and exploration applications. The creation time and date of last modification were the most general type of temporal data associated with a document. The cascading spatiotemporal pattern (CSTP) discovery process [15] determined partially ordered subsets of spatiotemporal (ST) event types whose instances were located together and occured serially. The determination of CSTPs from ST data sets was essential for application domains such as natural disaster planning and public safety. A data model [16] was used to track the history of spatial data in the

geographic information systems. The model involved timestamping of spatial objects in each layer. The strategies such as timestamp-based and snapshot-based representations were used. The approach also introduced a first-order spatio-temporal query language. A generalized representation of temporal events [17] was provided. An algorithm MLTPM discovered multi-label temporal patterns from temporal databases. The approach produced highly efficient and scalable results.

A geographical pictorial query language (GeoPQL) [18] was used to express the spatio-temporal queries. The approach was based on the idea of temporal layer that allowed the specification of spatial configuration of changing objects in a time peroid. The spatio-temporal query language was user-friendly and ease-to-use when compared with the other textual query languages. The SQL commands [19] were used to solve the ramification problem in temporal databases. A data model was formulated [20] based on an occupancy relation with a real-valued probability attribute. Some of the spatio-temporal queries in SQL were also described. The precision and recall concepts quantified the performance of the model based on its ability to solve various spatio-temporal queries.

Proposed Method: Temporal Sql3 Mining (Tsql3mine)

This section describes the proposed object oriented SQL3 for temporal mining. Most temporal databases store time-varying information. Normally, SQL is used for developing the applications that use the information stored in these databases. But, the adequate support for temporal applications is not provided by the plain SQL. The standard SQL3 does not allow manipulation and querying of temporal database. The flow of the SQL3MINE is shown in Fig.1.

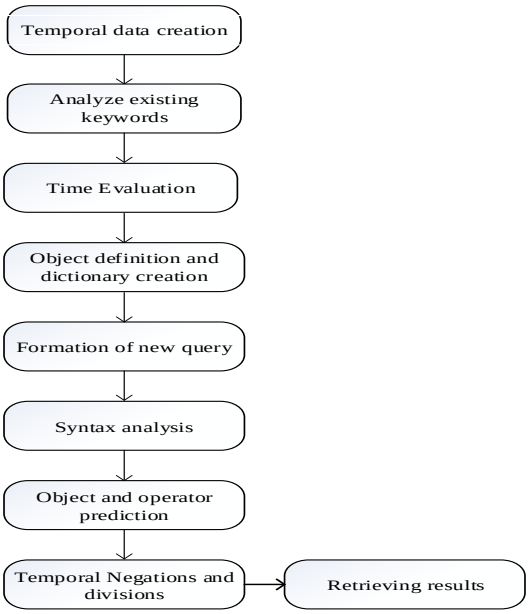

Figure 1: Flow of the proposed temporal SQL3 mine (TSQL3MINE)

A. Preprocessing the temporal dataset

In the TSQL3MINE approach, a temporal extension to the standard SQL3 is provided by using new keywords. These keywords are utilized to define different versions of temporal operators to SQL. Many new operators are proposed for retrieving results based on the given keyword. New keywords such as ON, ANY, BETWEEN, AS and JOIN are proposed depending on the object oriented SQL queries. Many forms of querying and updations are performed in the temporal database. The temporal SQL3 mining involves the creation of objects that are used to perform operations such as table creation, natural join etc. The objects are defined and the dictionary creation operation is performed. The generated dictionary provides new keywords and objects that are used to simplify the query operations in temporal database. In the TSQL3MINE technique, join operation is performed by using the object oriented query. The subqueries are also generated by using the objects. Some of the operators used are InBetween and ANY. So, the proposed approach simplifies the expression of temporal manipulations and temporal queries. Initially, the technique involves the creation of temporal data. The temporal data stored in a temporal database is different from the data that is stored in non-temporal database.

In a temporal database, a time period is attached to the data and so it is possible to store various database states. The temporal data are loaded into the database and then preprocessed. The inaccurate results are produced due to the presence of outliers in the temporal data. So, preprocessing of the temporal data is performed to detect and remove the outliers. The features are then extracted from the preprocessed temporal data and some existing keywords are analyzed.

B. Temporal versions of relational operators

In temporal database, the querying process is based on the temporal terms carried out with new keywords namely AS, ON etc. By adding these keywords the temporal version will be looking different from traditional SQL. These terms are applied in various clauses of a query. The proposed SQL operators and functions related to the valid time dimension are applied with temporal tables and columns. A single temporal dimension, i.e. the transaction-time or the valid-time dimension is considered. Some terms are defined to incorporate temporal specifications to the relational operators such as natural join, cross product, set operators, restriction, negation etc. These terms are expressed by the keywords that are used to set the operator by a date or time period. For example, temporal natural join is applied when there is a temporal relationship between two tables that can be temporal or non-temporal.

These relationships are recognized as temporal column and referential constraints. The timestamps of these columns are verified by using the possible timestamps associated with the concerned objects. The relations in a temporal context are combined using temporal joins. The most frequent temporal context is that the facts are required to be valid at the same time. The intervals should be same in the equality condition and one interval must occur within the other in the ‘during’ condition. Normally, the tuples that satisfy the operation join are concatenated. The resulting tuple is defined with the value of the timestamp, the concatenation is not essential in temporal joins. The resulting tuple is defined based on the type of the temporal join. It

is also assumed that the temporal intersection of the resulting timestamp is equal to the intersection of the active tuples for the individual timestamps. Some of the temporal join types are equal join, after join, overlap join, and intersection join etc. The equal join uses same timestamps and the intersection of timestamps is performed in the intersection join. The timestamps in the overlap join do not start or end at the same point. The temporal join operator is dependent on the timestamps of the links.

C. Similarity Calculation

Semantic keywords are computed by using the similarity calculation. The similarity calculation is performed between the generated keywords and the temporal data stored in the database. i.e the ASCII values of the generated keyword and the temporal data are computed. The determination of similarity results depend on the string matching. The equation for similarity calculation is as follows

$$Sim(A,B) = \frac{\sum_{i=0}^{n} A_i * B_i}{\sqrt{\sum_{i=0}^{n} A_i^2} * \sqrt{\sum_{i=0}^{n} B_i^2}} \quad (1)$$

where i= 1 to n. A and B are the similarity between the objects and n is the number of features. The data similarity is high if the ASCII values are equal. The steps involved in the temporal SQL3 mining are shown below

Algorithm 1: Temporal SQL3 mining and Data Similarity

```
Input: Dataset D, Query Q
Output: Retrieved result (RR) Using TSQLMINE
Process:
Copy G: D
Features Fd
Fd = Parsing G
Fd = Apply Preprocessing (Fd)
                        // Detecting & removal of Outliers
//Compute Semantic keyword using similarity calculation
For int i = 1 to n  then
 For j = 1 to m then
simij  = Calculate_Similarity (Q,fij)
If (m != N)
                    RR= sim [0][m - 1] / [m + n];
Return RR;    //Results retrieved with less
              relevancy
Else If (m==N)
                    RR = sim[0][m - 1];
                  Return RR;    //Results retrieved with
                                  high relevancy
End if
End for
End for
```

Here, n denotes the number of features and m denotes the number of instances. The dataset D is the input to temporal SQL3 mining (TSQLMINE). The dataset D contains various fileds such as Query, QueryTime, ItemRank and ClickURL. QueryTime specifies the time at which the query occurs. ItemRank denotes the frequency of a particular query. i.e. the number of occurrences of a specific query. The website associated with a query is denoted by ClickURL. Table 1 shows the dataset with various fields. The temporal dataset D is parsed and then stored in F_d. F_d is preprocessed to remove outliers and features are extracted. The pseudocode for similarity calculation is shown below

Algorithm 2: Similarity calculation

```
Calculate_Similarity (String str1, String str2)
ASCII = Calculate_ASCII_Difference(str1, str2);
N =str1.length ();
M = str2.length ();
For  i=1 to N then
For  j=1 to M then
Sim_ij = Sim(ASCII_i , ASCII_j)// refer eqn(1)
        End for
End for
Return Sim_ij
```

Here N and M are represents the length of the strings.

The similarity calculation is performed between the generated new keyword and the temporal dataset. The str1 and str2 denote the strings that are compared for computing similarity values. N and M are the length of strings. The ASCII difference is calculated between the two strings by using the Calculate_ASCII_Difference(str1, str2) function. The pseudocode for ASCII calculation is shown below

Algorithm 3: ASCII difference between two strings

```
Calculate_ASCII_Difference (String str1, String str2)
N = str1.length ();
M = str2.length ();
ASCII_NM = Ø          //where init is an array declaration
For  i=1 to N then
For  j=1 to M then
              ASCII_ij = |ASCII(str1_i) –ASCII(str2_j)|
        End for
End for
 return ASCII_NM
```

Here *N* and *M* are represents the length of the strings and *init* represents the array declaration. The ASCII difference between the two strings is calculated and stored in ASCII variable. If the ASCII values of keyword and the temoral data are equal, then

the results with high relevancy are retrieved. The result with less relevancy is acquired if the ASCII values of the temporal dataset and the keyword are not equal. The semantics are used to explore the new techniques for indexing and presentation. The time estimation for object oriented SQL3 operations is performed. The time for retrieving results is minimized by using the object oriented sql and operators. New functionalities and methodologies are introduced by the TSQL3MINE approach. The new keywords simplify the expressions of operators used in standard SQL3. For example, the temporal join operation can be expressed in the variant of TSQL2 for inclusion into SQL3.

```
Select a.AID, a.Query, a.QueryTime, d.AID, d.QueryTime
from preprocess a, detail d
Where a.AID=d.AID
```

Here, the join operation is performed between two tables (a and b). The ID and QueryTime is selected from two tables if the ID of table 'a' is similar to the ID of table 'b'. A join condition is temporal if it imposes a certain relationship between the timestamps of the participating tuples. In the temporal join operation, two or more temporal relations are combined by using a temporal join condition. The intervals are related to each other in terms of interval timestamps. There are many possible relationships between the two intervals. Both intervals can overlap each other or start and end at the same time etc. Temporal joins can be categorized according to the type of relationship on which the join condition is based.

Table 1: Temporal Dataset D

AID	Query	QueryTime	Item Rank	Click URL
124	basquet artwork	24/4/2006 22:44	1	http://www.cool..
125	funny aim icons	24/4/2006 22:48	3	http://www.studi..
126	custom myspace	24/4/2006 22:53	2	http://www.killer..
127	unusual artwork	24/4/2006 22:55	1	http://www.satg..
128	basquiats artwork	27/4/2006 22:57	7	http://www.xang..
129	peace and hate	30/4/2006 22:57	9	http://www.picg..

The simple expressions that are used in temporal join operation describe the relationships between start and endpoints. The temoporal joins are also used to compose complex types. If the join is processed by partitioning over the interval timestamp attribute, it is essential to perform the replication operation. The impact of replication is reduced in the intersection based join conditions. But the replication operation is essential for other temoporal joins such as right-overlap, left-overlap and overlap joins. The example for ANY operator is as follows

```
SELECT * FROM `detail`
Where ItemRank<any (select AID from dataset
Where Query Time between Startdate and enddate)
```

Here, ANY operator is used to select ID from dataset by specifying the QueryTime. i.e. startdate and enddate. With these enhancements, the suggested temporal versions of the operators become much simpler. The proposed temporal

SQL3 mining assures that the legacy query results remain the same when the other queries run on DB.

Performance Analysis

This section describes the performance evaluation of the proposed temporal SQL3 mining (TSQL3MINE) that is used to improve the query manipulations in temporal database. The TSQL3MINE approach is compared with the existing techniques such as Moving Objects Spatio-Temporal (MOST) data model and Object Role Modeling (ORM) approach. The performance analysis shows that the TSQL3MINE approach achieves improved time efficiency, resource efficiency, processing speed when compared with the other existing approaches.

A. Execution time

The execution time of the TSQL3MINE technique is compared with the existing approaches and the result is shown in Fig.2. The analysis shows that the proposed approach consumes minimum time for query processing than the other existing techniques.

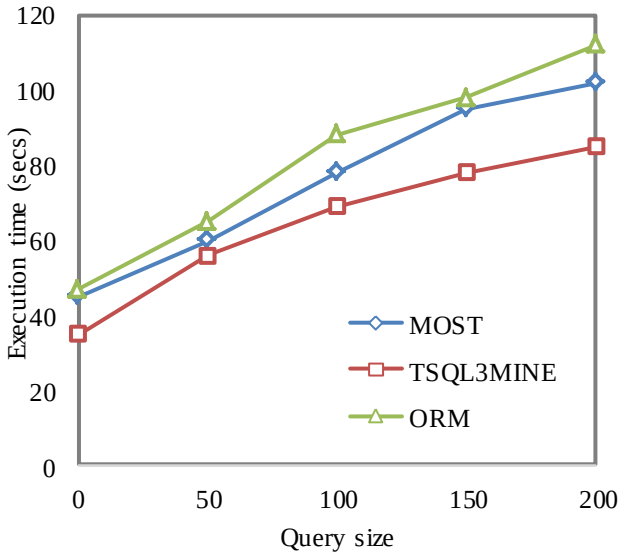

Figure 2: Comparison of Execution Time of TSQL3MINE And Existing Methods

B. Resource efficiency

Fig.3 shows the comparison of resource efficiency of the TSQL3MINE and the existing techniques. The results show that the resource efficiency is high in the TSQL3MINE method. The TSQL3MINE method consumes minimum resource for query processing and hence resource efficiency is enhanced.

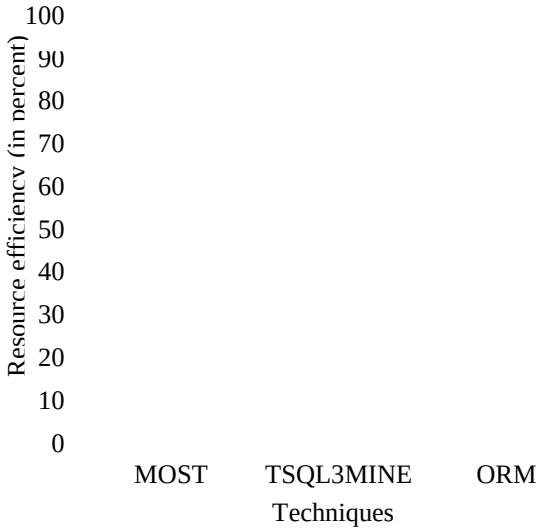

Figure 3: Comparison of resource efficiency of TSQL3MINE, MOST and ORM

C. Processing speed

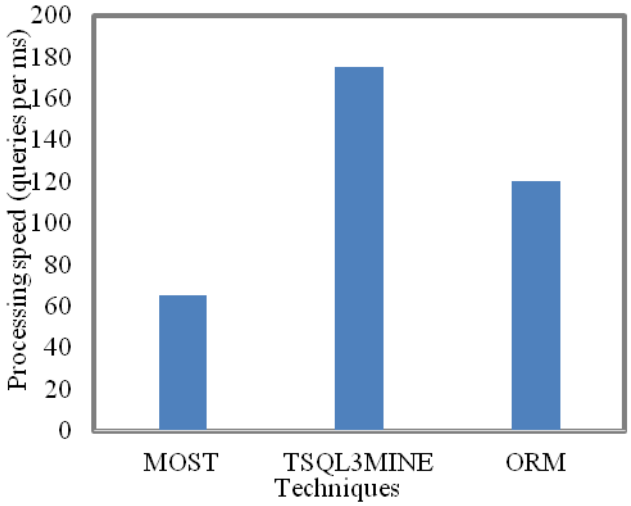

Figure 4: Processing Speed of TSQL3MINE and Existing Techniques

The processing speed of the TSQL3MINE approach is compared with the existing techniques and the result is shown in Fig.4. The analysis shows that the proposed method achieves high processing speed and computing power than the existing methods.

D. Latency

The latency (response time) of the TSQL3MINE technique is compared with the existing approaches and the result is shown in Fig.5. The analysis shows that the TSQL3MINE approach achieves minimum time delay for different number of queries when compared with other existing techniques

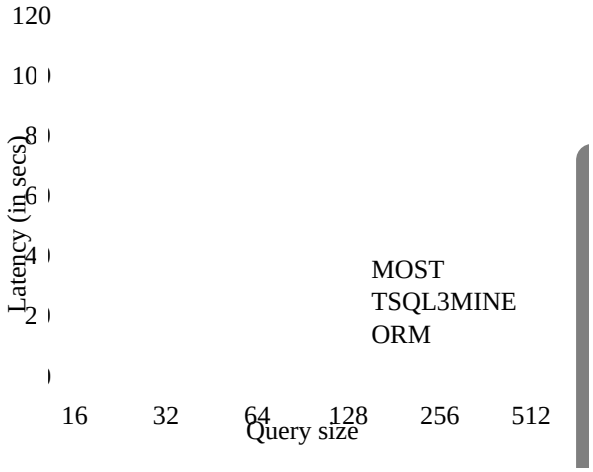

Figure 5: Time Delay of TSQL3MINE and Existing Methods

E. Accuracy

The accuracy of the TSQL3MINE method is compared with other existing techniques and the result is shown in Fig.6. The analysis shows that the proposed approach generates high accurate results for different number of queries when compared with other methods.

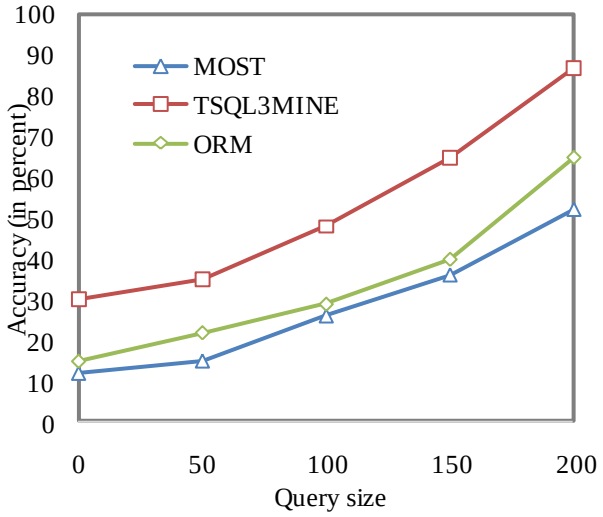

Figure 6: Accuracy of the TSQL3MINE and standard SQL

F. Time Complexity

The time complexity of the TSQL3MINE is compared with the standard SQL approach and the result is shown in Fig.7. The analysis shows that the time complexity is reduced in the TSQL3MINE when compared with SQL.

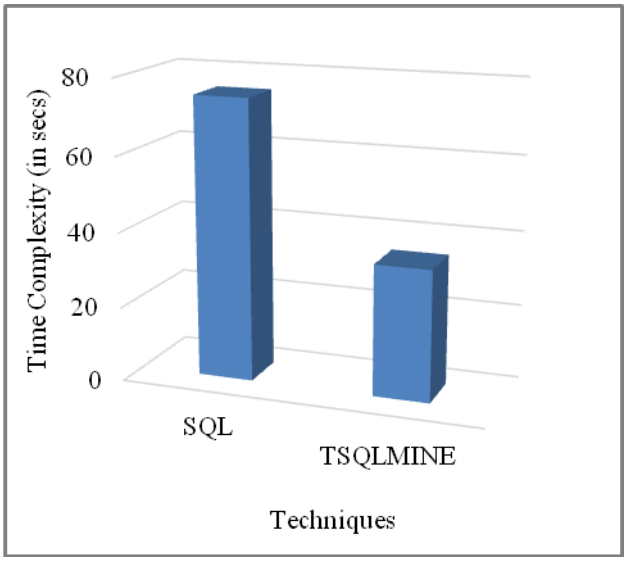

Figure 7: Time Complexity of The TSQL3MINE and Standard SQL

Conclusion

In this paper, a temporal SQL3 mining (TSQL3MINE) approach is proposed to provide a temporal extension to the standard SQL. We introduced new keyword to define different versions of temporal operators to SQL3 and perform many query operations. The query manipulations are simplified in the temporal database by using the object oriented processing. The operations such as table creation, natural join are performed in the temporal database by using the generated objects. The TSQL3MINE approach produces the accurate results and the performance of the TSQL3MINE is compared with the existing MOST and ORM techniques.

The experimental results show that the TSQL3MINE method achieves improved search efficiency, resource efficiency, processing speed and time efficiency when compared with the other existing methods.

REFERENCES

[1] K. Wongsuphasawat, C. Plaisant, M. Taieb-Maimon, and B. Shneiderman, "Querying event sequences by exact match or similarity search: Design and empirical evaluation," *Interacting with computers*, vol. 24, pp. 55-68, 2012.

[2] N. Mahmood and A. Burney, "Advances in fuzzy temporal relational databases: a review," *WSEAS Transactions on Information Science and Applications,* vol. 8, pp. 171-184, 2011.

[3] K. Kamaraj and C. Chandrasekar, "Spatio-Temporal Database Using Fuzzy Clustering."

[4] N. Rahman, "Temporal Data Update Methodologies for Data Warehousing," *Journal of the Southern Association for Information Systems,* vol. 2, 2014.

[5] F. M. Farham Mohamed, M. N. A. R. M Nordin A Rahman, Y. M. L. Yuzarimi M Lazim, and S. B. M. Saiful Bahri Mohamed, "Managing Multimedia Data: A Temporal-Based Approach," *International Journal of Multimedia and Ubiquitous Engineering,* vol. 7, pp. 73-86, 2012.

[6] M. Mkaouar, R. Bouaziz, and M. Moalla, "Querying and manipulating temporal databases," *International Journal of Database Management Systems* vol. 3, pp. 1-17, 2011.

[7] S. Alamri, D. Taniar, and M. Safar, "A taxonomy for moving object queries in spatial databases," *Future Generation Computer Systems,* vol. 37, pp. 232-242, 2014.

[8] R. Ďuračiová, "Querying Uncertain Data in Geospatial Object-relational Databases Using SQL and Fuzzy Sets," *Slovak Journal of Civil Engineering,* vol. 21, pp. 1-12, 2013.

[9] A. B. Rashid and M. A. Hossain, "Challenging issues of spatio-temporal data mining," *Computer Engineering and Intelligent Systems,* vol. 3, pp. 55-63, 2012.

[10] R. Shuxia, Z. Zheng, and Z. Xiaojian, "An Indeterminacy Temporal Data Model Based on Probability," *TELKOMNIKA Indonesian Journal of Electrical Engineering,* vol. 11, pp. 6686-6692, 2013.

[11] C. Combi, G. Pozzi, and R. Rossato, "Querying temporal clinical databases on granular trends," *Journal of biomedical informatics,* vol. 45, pp. 273-291, 2012.

[12] A. S. Kumar, A. Alrabea, and P. C. Sekhar, "Temporal Association Rule Mining in Large Databases," *Knowledge Discovery Practices and Emerging Applications of Data Mining: Trends and New Domains,* p. 48, 2011.

[13] M. Al-Kateb and B. S. Lee, "Load Shedding for Temporal Queries over Data Streams," *JCSE,* vol. 5, pp. 294-304, 2011.

[14] O. Alonso, J. Strötgen, R. A. Baeza-Yates, and M. Gertz, "Temporal Information Retrieval: Challenges and Opportunities," *TWAW,* vol. 11, pp. 1-8, 2011.

[15] P. Mohan, S. Shekhar, J. A. Shine, and J. P. Rogers, "Cascading spatio-temporal pattern discovery," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 24, pp. 1977-1992, 2012.

[16] L. Gómez, B. Kuijpers, and A. Vaisman, "A data model and query language for spatio-temporal decision support," *Geoinformatica,* vol. 15, pp. 455-496, 2011.

[17] Y.-L. Chen, S.-Y. Wu, and Y.-C. Wang, "Discovering multi-label temporal patterns in sequence databases," *Information Sciences,* vol. 181, pp. 398-418, 2011.

[18] A. D'Ulizia, F. Ferri, and P. Grifoni, "Moving GeoPQL: a pictorial language towards spatio-temporal queries," *Geoinformatica,* vol. 16, pp. 357-389, 2012.

[19] N. Papadakis, D. Plexousakis, and Y. Christodolou, "The ramification problem in temporal databases: a solution implemented in SQL," *Applied Intelligence,* vol. 36, pp. 749-767, 2012.

[20] V. Menon, B. Jayaraman, and V. Govindaraju, "Spatio-temporal querying in smart spaces," *Procedia Computer Science,* vol. 10, pp. 366-373, 2012.