CHAPTER 5
AN ENHANCED QUERY MANIPULATIONS IN TEMPORAL DATABASE USING TSQL3MINE APPROACH

5.1 INTRODUCTION

The database that offers an effective storage capacity to the information in various time samples is referred to as temporal database. The primary and most common categorical data type and associated numeric values are termed as temporal data sequences. The event sequences that specify the packet traces and sensor readings, time series and temporal databases (i.e. databases with versioning, relation with time stamped tuples) are some examples. It is used in following fields: geographical data processing, biomedical, internet site usage for financial data forecasting and monitoring. Temporal databases consist of time-stamping information. Moreover, it contains built-in support for manipulating the data involving time. Initially, Temporal-Standard Query Language (TSQL) extracts the data time stamp information from the database in temporal domain that allows the various distinction database states.

The temporal database permits entity time stamping with time periods and properties. Specifically, the major temporal aspects are time attributes for transaction and validation. Valid time is time required for validation of results obtained from query processing on a real world basis. Alternatively, the duration for an effective transaction of user query and the response refers to transaction time. Hence, two types are available with time aspects basis. They are rollback and historical database and each of which stores the data corresponding to timing attributes.
Both the timing attributes (valid and transaction) collectively create the bi-temporal data. The management of temporal database and data access are governed by following features.

- Non-overlapping period constraints-based temporal primary keys
- Transaction time maintenance by system
- Temporal constraints (referential integrity/non-overlapping uniqueness)
- Current / past / over time temporal queries
- Definition of valid / transaction time attributes.

A significant extension to data mining is Temporal Data Mining (TDM) that refers to the non-trivial extraction of implicit, useful information from temporal database. The determination of hidden relations between sequences and sub-sequences of events is the major objective of TDM that include three steps:

1. Suitable data sequence modelling,
2. Definition of sequence similarity measures, and
3. Effective problem handling by the representations and models.

Temporal data mining is a significant extension of the data mining process. It deals with the gathering of useful information from temporal data. 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 relationship between sequences and sub-sequences of events. The determination of relationship 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 that includes nominal symbols corresponding to a specific alphabet is normally referred to as a temporal sequence. If the sequences are continuous and real valued elements, then they called as time series. The temporal and time series occur in following domains: Engineering, Finance, Scientific research and Medicine. Usually, the temporal classification of data models into two domains namely, log-based system monitoring or sensor-based monitoring. The TDM involves following necessary operations i.e., association, classification, prediction, trend analysis, characterization, clustering and pattern discovery.

Structured Query Language (SQL) is normally used for developing the applications that use the information in these temporal databases. But standard SQL3 does not provide adequate support for the temporal applications and query processing. In order to overcome these drawbacks, new keywords are proposed in this phase of the research work to provide temporal extension to SQL3. In the Temporal Semantic Mining process, new keywords are used to define different versions of temporal operators to SQL. The objects are defined and a dictionary is created to provide new keywords. The object oriented operations are performed in the proposed method. The new functions and methodologies are used to perform the query operations. The query processing is simplified by the proposed Temporal Semantic Mining approach. 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 Database (DB). The Semantic method allows for easy processing and navigation. The proposed semantic enhances query manipulations in the temporal database and achieves improved time efficiency, search efficiency, processing speed and accuracy when
compared with other existing techniques such as Standard Query Language (SQL) data model and Object Role Modeling (ORM) approach.

In this research work, a temporal SQL3 mining (TSQL3MINE) technique is proposed to enhance query manipulations. The generation of ASCII based keywords governed the TSQL3MINE. These keywords define the diverse temporal operator versions. The generation of new keywords enhances the operator expressions used in SQL3. The generated objects perform the operations of natural join, table creation etc. The utilization of new string formulation and ASCII based keyword generation simplified the query operations. The technique reduced the complexity and improved the accuracy. Easy navigation and processing are allowed by the TSQL3MINE approach.

5.2 TEMPORAL DATA MODELS

The important step is the database structure extension by a Database Management System (DBMS). The mechanisms provided for capturing the timing attributes are verified by the relations. The capture and verification process are important to identify the temporal relations. Consequent steps are provided to support temporal data modeling and design of database and temporal query languages. Temporal data models are considered as attribute time stamped or tuple stamped, First Normal Form (FNF) or Non-First Normal Form (NFNF), timing and important attributes.

5.2.1 Temporal Relational Data Models

Temporal Relational Data Model (TRDM) refers to the models that use tuple time stamping. Non-First Normal Form (NFNF) relational data models using attribute time stamping are discussed here. A TRDM that combines tuple and attribute time stamping concludes this section. Figure 5.1 shows the temporal relations. Figure 5.2 shows the snapshot relation, Figure 5.3 shows the valid-time
relation for sequence query and Figure 5.4 shows the valid-time relation for non-sequence query.

The different types of relations are

- Snapshot
- Valid-time
- Transaction-time
- Bi-temporal relation

![Figure 5.1 Temporal Relations](image-url)
Figure 5.2 Snapshot Relation

Figure 5.3 Valid-Time Relation for Sequence Query

Figure 5.4 Valid-Time Relation for Non-Sequence Query
5.3 QUERYING TEMPORAL DATABASE

The objectives of utilization of temporal databases in TSQL3MINE are the manipulation and modelling of diverse temporal facts regarding timing attributes (valid and transaction time). In SQL standard, a major development is DBMS, which proposes a transaction-time databases. A transaction-time database characterizes the historical database, designed to preserve a reality that was valid in the past. Additionally, transaction-time plays a similar role to valid time in databases for web documents. For example, Wikipedia, the page valid-time is parallel to the time when the page became accessible on the web. Therefore, it refers to the start transaction-time in DBMS. In general, there are three temporal facts,

- Valid-time
- Transaction time
- Bi-temporal facts

The proposed TSQL3MINE focus on transaction-time databases, though related queries could be used on valid time-based bi-temporal databases. The instants of temporal elements in a given calendar and granularity refers to valid-time. Therefore, maintain that the valid facts of history are real-time updated with post active effect. However, active effect cannot record deletions and modifications of errors. Transaction time facts can record the DBMS facts manipulations, timestamp and it activates the fact. Therefore, this track covers insert operations, update operation, development function updates the error corrections. The unit of transaction time facts according to the Operating System (OS) schedule and granularity is the second. Consequently, maintain the facts history, valid, current and but not future. Furthermore, only the past facts are updated. Figure 5.5 shows the flow of the proposed approach.
Figure 5.5 Flow of the Temporal SQL3 Mine Approach
These valid or transaction histories have inadequacies. A whole history can be confident only if timestamp facts are validated by both valid-time and transaction time, achieving bi-temporal facts. Then, it becomes potential to update the facts with post active effects which records valid and invalid facts. Although, the organization of these facts become difficult and troublesome, the temporal dimensions utilization must be acceptable by the needs of users. The database in temporal domain (TDB) must contains non-temporal facts, historical and valid-time facts. Hence for a valid history of facts in reality, transaction-time facts were needed.

5.4 TEMPORAL SQL3 MINING (TSQL3MINE)

The conventional SQL3 does not allow manipulating and querying of temporal database. A temporal extension of conventional SQL3 by new keywords is proposed to allow temporal query manipulation. The new keywords incorporation defines the numerous temporal operator versions to SQL and to achieve many forms of TDBs query and update. The proposed work simplifies the expression of temporal queries and temporal manipulations. The formal semantics is proposed to explore the new techniques for indexing and presentation.

Hence, the proposed mechanism guaranteed to make the legacy queries results remain the same when the other queries run on DBs. By using this, the temporal dimension incorporation brings the current data. The database language SQL3 includes various object concepts. The temporal extensions of object relational model provided the object relational DBMS-based valid-time extension of SQL. The TDB supports the relations identified in the valid-time and defines the SQL extension. The language incorporates only the attribute-level valid-time dimension with SQL basis and temporal.
Although, the two time stamped columns must make the timestamp intersection, the joins and temporal versions are not treated according to the SQL3 syntax. A model permits columns and tables time stamping and supporting the time dimensions such as valid-time and transaction-time. Therefore, this model can be implemented using object-relational technology. SQL3 extension can be coordinated with all the operators of this language. The extension elements targeted to the dimension introduction without considering the specific aspect relations. This describes the proposed object oriented SQL3 for temporal mining. Most TDBs store time-varying information. Normally, SQL is used for developing the applications with the information stored in these databases. But the SQL was not responsible for provision of an adequate support to temporal applications. The SQL3 does not allow manipulation and querying of temporal database.

5.5 PRE-PROCESSING THE TEMPORAL DATASET

In the TSQL3MINE approach, a temporal extension of SQL3 is provided by using new keywords. The keywords utilization in proposed TSQL3MINE defines the diverse SQL temporal operator versions. Many new operators are proposed for retrieving results with given keyword basis. New keywords such as ON, ANY, BETWEEN, AS and JOIN are proposed with an object oriented SQL queries basis. Many query and update forms available in the TDB. The temporal SQL3 mining involves object creation which performs the following operations i.e., table creation, natural join, etc. The objects are defined and the dictionary creation operation is performed. The generated dictionary provides keywords and objects used to simplify temporal database query operations. In the TSQL3MINE technique, the object oriented query performed the join operation and sub-queries generation. Some of the operators used are IN, BETWEEN and ANY.
The TSQL3MINE simplifies the expression of temporal manipulations and temporal queries. Initially, the TSQL3MINE gathers the data and it is stored in a DB. The data storage in TDB is different compared to the data storage in the non-temporal database. In TDB, the time attribute attachment with the data useful for the storage of database states stores. The data loading into the TDB is continued to preprocess stage with the following steps.

- Data cleaning – Fill in missing values, noise removal, outlier removal
- Data Integration – Multiple databases integration
- Data Transformation-Normalization and aggregation
- Data Reduction – Reduced volume representation
- Data Discretization – data reduction with the specific importance

The non-availability of the whole data (no recorded values of several attributes) or the missing values in the database caused the limitation in the mining process. Hence, the noise removal and the missing values prediction are the necessary tasks in the preprocessing stage. The inaccurate results are produced due to the outliers in the TDB, it detects and removes the outliers following which TSQL3MINE extracts the features and some existing keywords are analyzed. Initially, the features from the data set are given as the input to the preprocessing stage. The basic steps such as sorting, removal of an outlier and noisy features and smoothing are performed on the features. The normalization applied on outlier-free features converts the data format into a unified form.

The algorithm for preprocessing stage is as follows:
Preprocessing

**Input:** Features ($F_d$)

**Output:** Preprocessed form

- Sort features based on equal-depth level
- Detect and remove the outliers
- Detect the suspicious values
- Smooth the feature based on regression function
- Integrate the feature from the multiple datasets (that avoids the redundancies)
- Remove the noise in data
- Construct the data cube and normalize them
- Obtain the reduced representation of features

5.6 TEMPORAL VERSIONS OF RELATIONAL OPERATORS

The temporal information modelling required subjective/object-oriented approaches, which led to the classification and temporal models design. The parallel functionality is re-engineered every time a temporal model is generated for an application. It submits for combining the various features of time under a single infrastructure, permitting design reprocess. This work proposed an object-oriented framework that affords such a unified infrastructure. An object oriented method permits capture of the complex semantics of time by demonstrating a basic entity.

The querying process in a temporal database is based on the temporal terms expressed by new keywords such as AS, ON etc. These keywords
distinguish temporal versions of the relational operators that are specific to SQL. These terms are utilized in various clauses of a query. The proposed SQL operators and functions are applied with temporal tables and temporal 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. Such a relationship is realized by a temporal column and a referential constraint.

The timestamps of temporal columns are verified by an associated timestamps with the concerned objects. The temporal joins relate the data from the temporal context relations. The frequent context is that the facts are required to be valid. The intervals should be same in the equality condition and one interval must occur within the other in the ‘during’ condition. Normally, the tuples satisfy the join condition are concatenated. As the timestamp for the tuple is defined, the concatenation is not essential in temporal joins. It is also assumed that both the resultant and individual timestamp intersections are equal. The temporal join types are equal join, after the join, overlap join, and intersection joins etc. The equal join uses same timestamps and their intersection is performed at the intersection join. The timestamps overlap but do not start or end at the same point in the overlap join due to the dependency of temporal join operator with link timestamps.
5.7 SIMILARITY CALCULATION

An important computation in data mining and time series analysis is the similarity estimation. The time series data model includes following categories continuous nature and numerical. Then, the approximate similarity measurement is frequently performed for time series model since, it has the characteristics: numerical and continuous nature.

The predictable query results afford useful information for different stock analysis. Time series domain planning an appropriate similarity function is by no means trivial. In addition, the whole length of all-time series is considered during the similarity search in whole sequence matching. It involves the comparison of query sequences which corresponds to all candidate series with the distance formulation and record the smallest distance sequence. In subsequence matching, a query and a longer sequence $Q$ and $P$ are assumed, then the task find the subsequences in $P$ that matches $Q$. Subsequence matching requires the placement of query $Q$ within longer sequence.

The similarity estimation computes the semantic keywords. The similarity existence between the generated keywords and stored TDB, i.e. the ASCII-based keyword generation. The similarity results are obtained from string matching. The measurement of similarity described as follows,

$$\text{Sim}(A, B) = \frac{\sum_{i=0}^{n} A_i * B_i}{\sqrt{\sum_{i=0}^{n} A_i^2} * \sqrt{\sum_{i=0}^{n} B_i^2}}$$  \hspace{1cm} (4.1)

where, $i = 1$ to $n$. $A$ and $B$ are objects and n is the feature count. The data similarity is high if the ASCII values are equal. The algorithmic steps for the temporal SQL3 mining are shown below:
Algorithm 5.1 Temporal SQL3 Mining and Data Similarity

**Input:** Dataset D, Query Q

**Output:** Retrieved result (RR) Using TSQLMINE

**Process:**

Copy G: D

Features $F_d$

$F_d = \text{Parsing G}$

$F_d = \text{Apply Preprocessing (F_d) // Detecting & removal of Outliers}$

//Compute Semantic keyword using similarity calculation

for int $i = 1$ to $n$ then

    for j = 1 to m then

        $\text{sim}_{ij} = \text{Calculate\_Similarity (Q,f_{ij})}$

        if (m!=$n$)

            $\text{RR}= \text{sim}[0][m - 1] / [m + n]$;

            Return RR; //Results retrieved with less relevancy

        else if (m==$n$)

            $\text{RR} = \text{sim}[0][m - 1]$;

            Return RR; //Results retrieved with high relevancy

        end if

e nd for

e nd for
Here, $n$ denotes features count and $m$ denotes the instance count. The dataset $D$ is the input to temporal SQL3 mining (TSQLMINE). The dataset $D$ contains following fields Query, QueryTime, ItemRakn and ClickURL. QueryTime specifies the time at which the query occurs. ItemRakn denotes the frequency of a particular query, i.e. the occurrence count of a specific query. The website associated with a query is denoted by ClickURL. The pseudocode for similarity calculation is shown below

\textbf{Algorithm 5.2: Similarity Calculation}

\begin{verbatim}
Calculate_Similarity (String str1, String str2)

ASCII = Calculate_ASCII_Difference(str1, str2);

N =str1.length ();

M = str2.length ();

\textbf{for} i=1 to N \textbf{then}

\textbf{for} j=1 to M \textbf{then}

\hspace{1cm} Sim$_{ij}$ = Sim($ASCII_i, ASCII_j$) \text{ // refer eqn(5.1)}

\textbf{end for}

\textbf{end for}

\textbf{return} Sim$_{ij}$
\end{verbatim}

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

**Algorithm 4.3 ASCII difference between two strings**

Calculate_ASCII_Difference (String str1, String str2)

N = str1.length ();
M = str2.length ();
ASCII_{NM} = \emptyset \quad //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 \( M \) represent the length of the strings and \( init \) represents the array declaration. The ASCII diversity in two strings is calculated and stored in ASCII variable. If both temporal data and the ASCII keyword are equal, then the results with high relevancy are retrieved. The result with less relevancy is acquired if the ASCII keyword and temporal dataset and are not equal. The semantics explore the new techniques for indexing and presenting. The time estimation for object oriented SQL3 operations is performed. The time for retrieving results is minimized by an object oriented SQL and operators. New string formulations and the ASCII based keyword generation are introduced by the TSQL3MINE approach. The generated 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 two tables (a and b) are joined and QueryTime are selected from two tables if the ID of table ‘a’ is same as ID of table ‘b’. The imposing of relationship of timestamps specifies the join condition is temporal. Multiple temporal relations are combined by using a condition. The time stamps decided the interval relationship. A single interval related with the other intervals regarding interval timestamps. There may be overlap, start and end occur in the intervals in the same time. The interval relationship categorized the temporal joins with the basis of join condition. Table 5.1 illustrates 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.

**Table 5.1 Temporal Data $D$**

<table>
<thead>
<tr>
<th>AID</th>
<th>Query</th>
<th>QueryTime</th>
<th>Item Rank</th>
<th>Click URL</th>
</tr>
</thead>
</table>
The simple expressions that are used in the temporal join operation describe the relationship of start / end points. The temporal joins compose complex types. If the join is processed by partitioning over the interval timestamp attribute, then the replication operation is essentially performed. The impact of replication is reduced based on join conditions in the intersection process. But the replication operation is essential for other temporal joins such as right-overlap, left-overlap and overlap joins. The example for ANY operator is as follows,

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

Here, ANY operator select ID from dataset by specifying the QueryTime, i.e. startdate and enddate. With these enhancements, the suggested temporal versions become much simpler. The temporal SQL3 mining (TSQL3MINE) assures that the legacy query results remain the same when the other queries run on the DB. Finally, the performance of the proposed temporal Semantic mining approach TSQL3MINE is evaluated by comparing it with the existing techniques such as Standard Query Language (SQL) data model and Object Role Modeling (ORM) approach. The performance analysis shows that the proposed TSQL3MINE approach achieves improved time efficiency, resource efficiency, processing speed when compared with the other existing approaches.

5.8 SUMMARY

In this research work, a temporal SQL3 mining (TSQL3MINE) approach was proposed to provide a temporal extension of the conventional SQL3. The generated new keywords defined diverse temporal operator versions to SQL3 and performed many query operations. The utilization of object oriented
programming in TSQL3MINE simplified query manipulations in the TDB. The operations of table creation, natural etc. were performed in the TDB by the generated objects. Accurate results were produced by the TSQL3MINE approach. The Temporal Semantic mining approach TSQL3MINE was compared with the existing techniques such as Standard Query Language (SQL) and Object Role Modeling (ORM) approach. The performance analysis showed that the proposed TSQL3MINE approach achieved improved time efficiency, resource efficiency, processing speed when compared with the other existing approaches.