Ilkogretim Online - Elementary Education Online, Year; Vol 20 (Issue 5): pp. 431-442
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.05.45

Performance Enhancement of Valid-time Query Retrieval through Application of Access Method

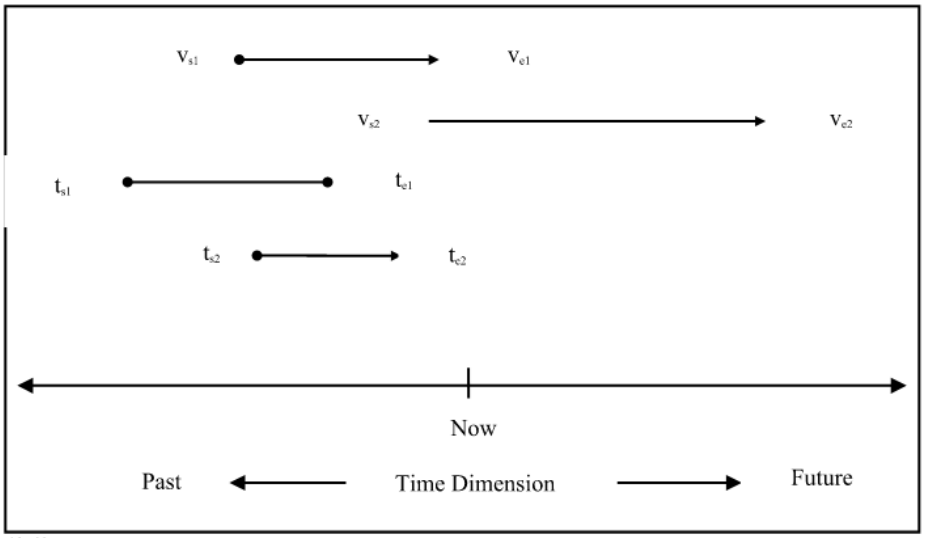
Shailender Kumar, Department of Computer Science and Engineering, Delhi Technological University, Delhi, shailenderkumar@dce.ac.in
Pankaj Lathar, P.G. Dept of Computer Science and Applications, Bhai Parmanand Institute of Business Studies, Delhi, pankajlathar@gmail.com

Abstract- Valid-time database depicts a certain error-free effigy of database which besteads agile delegation of the set of interval objects that may be historical or condign to the current environment and even pertinent for the future aspects that can be ameliorated randomly at any point in valid time domain. As data is emerging endlessly with time in a vast magnitude so to retrieve it efficiently is a key idea of research. The access method is exigent for time-varying databases than conventional one due to their timely and eternally growing traits. In this paper, we have implemented data model and wielded the binary-tree approach to implement query retrieval acquisition of time-oriented database, which sustains the preferential scenario as collated to the extant approach supported by database environment. We also adumbrated the various proposed access approaches that support dynamic interval management. In the end of the paper, we have summarized them based on the time and capacity requirement.

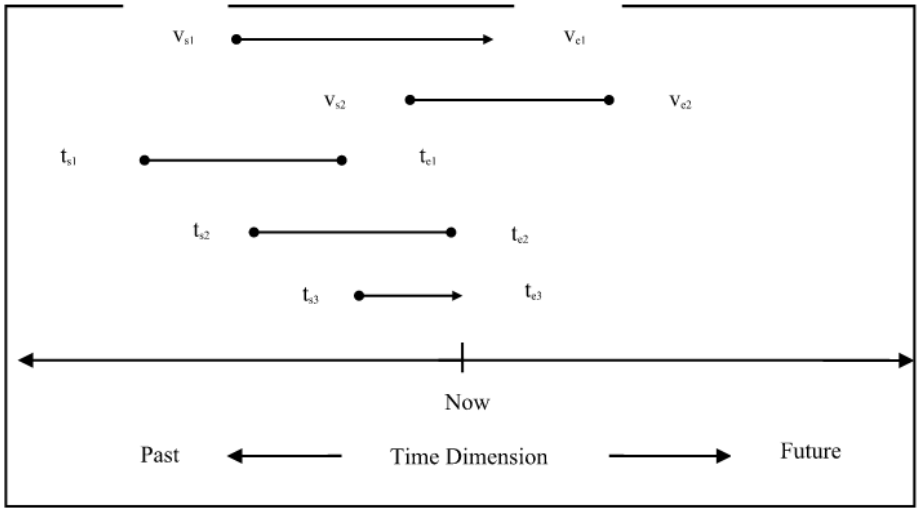
Keywords: Valid-time interval, temporal database, indexing.

I. INTRODUCTION

In database applications, intricacy of information is accruing perpetually and handling time-varying nature of data is a crucial aspect. In recent times, the size of the databases keep continuously growing in the competition-oriented industry and to cope up with this endlessly evolving scenario an effective access approach is vital to support the time-varying query retrieval dextrously and swiftly with minimum space exploitation. The traditional database maintains the only current state of an entity or objects i.e. whenever place mutation happens, the past state is overwritten with the new one. In these database applications, transaction history is maintained in the forms of logs (i.e., Backups, checkpoints) which are severe and costly to bring about. Time-varying databases should accompany facts or data with time which represents the more accurate perspective of a database that is expedient for analysing and monitoring information. Time may be represented by either time instants or durations by associating with the facts or events to represent information more precisely. Transaction time (or machine time) database also called as rollback databases [1][2] stores historical as well as the current state of aeon objects that pullulate in accruing order. Machine time refers to the time at which facts are registered in the database and evolves in an append-only manner, also called machine activity. Once the record is entered into the system, the amendment to the data is not possible and commutation of data is done logically without affecting the previous state physically. These traits help in maintaining the complete history by recording the changing state of objects and can be used in numerous applications such as auditing and monitoring.
Valid time refers to the time at which actual facts or events occurred in the real-world and that represents the clearer and legitimate perspective of data (also called as historical database as defined in ([1][3]). The indicated database stores the actual time-varying aspect of the database, as it reflects the real-world history instead of database history. It renders the agile establishment of epoch-objects and stores the rigorous image of database which maintains preceding, current as well as succeeding time of data. In numerous applications, it can be adopted such as banking, project planning, and weather forecasting. In Fig.1 we have attempted to represent the concepts for both valid and machine time intervals management and sustain how they differ from each other with the help of an example. The interval $<v_s, v_e>$ denotes time duration in which v_s represents beginning time and v_e represents ending time for a valid-time object and similarly, the interval $<t_s, t_e>$ denotes time duration in which ts represents begin time and te represents end time for a machine-time object.



(1.1)



(1.2)

Fig.1. Changes in the State of interval object with previous state represented in (1.1) and after modification in intervals is shown in (1.2).

The <vs1, ve1> , denotes time duration for valid-time object1 and <vs2, ve2> represents the time duration for object 2 and in similar way the machine time objects are interpreted. In machine time databases, time pullulates in perpetually increasing order without affecting the previous state of an object in reality but in valid-time databases, time-durations can be commutated from any point in intervals domain which does not follow any specific order. That means intervals can be modified or deleted randomly whenever errors/changes happened. In Fig. 1 the agile nature of valid-time intervals have been illustrated, which describes the changes in the states of the objects collection. The combination of both machine time and valid-time is recorded in Bi-temporal databases. Bi-temporal databases depict the more realistic representation for time-varying information because the information is represented by machine time along with the absolute incidence time[4].

As data is emerging endlessly with time in a vast magnitude so to retrieve them efficiently is a key idea of research. An approach for retrieving and managing the huge amount of time-varying data is a vital obligation. Dynamic management of time-durations is difficult because changes occur in an unordered manner and effective access method is essential for fast retrieval of queries with effective utilization of space[5]. In this paper, we have implemented the temporal data model and utilized the binary-tree approach on temporal attributes to optimize the query retrieval process and analysed the performance of the valid-time database by applying indexing on it with existing database query approach and the results are encouraging. There is a significant reduction in the time required to answer the time-varying query problem. We also adumbrate the various access approaches that support the agile set of valid-time duration management proposed in previous works of literature. We have also summarised the time and capacity required

for various indexing approaches. The previous work section presents the details about related work in previous literature. Preliminaries section provides the concept and various parameters needed to evaluate the access method cost. Access structure section presents a different categorization of access structures. In the results segment, the enhancement of time-varying query retrieval is presented. Finally, in the last section, the conclusion and future work is summarized.

II. PREVIOUS WORK

A variety of approaches has been introduced in previous works of literature for indexing valid-time intervals. The popular R-trees in [6], is the height-balanced search tree mainly introduced for spatial indexing and was further utilized for valid-time indexing in which aeons were represented into a one-dimensional range search problem. Time Index in [7] has been accustomed for indexing valid-time durations using B+ tree with the assumption that time pullulates in increasing order and changes to data happen rarely. The problem with this method was the requirement of more space which takes Oc2 time, where c is a number of interval object collections. Kolovson & Stonebraker in [8][7] proposed another access approach Segment R Tree, which combines Segment-Tree with R-Tree to reduce the overlapping problem among nodes as resulted in R-trees. In [9] Meta block tree has been introduced which divides the area above the diagonal in two-dimensional planes into the number of metablocks. It is semi-dynamic in nature and supports the only insertion efficiently. The author in [10] has proposed a new technique called as Path caching which externalizes the existing data structure by augmenting the various main memory access frameworks such as interval tree, priority search tree, and other.
Time index+ in [11] has been proposed to triumph the space and update requirement issue in time-index but still, it does not deal with it efficiently. External Interval tree method in [12]contends I/O optimal solution for one-dimensional interval query. Ramaswamy in [13] introduced a window-based method to solve the point-based query. MAP21 in [14] maps the valid-time intervals into a single point and indexes them using regular b-tree. Relational Interval (RI) tree access structure rendered by [15] has been implemented on the top of the relational storage system and can be integrated with Relational database management system (RDBMS) as well as with Object RDBMS. The RI-tree was further extended by [16]in which the basic interval-relationship based query has been represented. The authors in [17] were introduced Triangular Decomposition method, the indexing approach for the temporal database that evolves in an append-only manner.

A compilation on similar concepts has been stated where evaluation on temporal constraints for migration of legacy applications is being done from non-temporal data systems to the temporal environment[30][31]. For a successful migration, Temporal compatibility should be kept in check. The Temporal compatibility contains the aspects and features of temporal languages that are mapped in accordance with the nontemporal languages that they change. This ensures smooth conversion and migration of legacy applications that have a non-temporal touch[32].

Mate, S. et al[33]provided with the graphical model to implement temporal queries. Stating the example of retrieving of patient cohorts in electronic patient data, which often demands the definition of temporal constraints between the selection criteria. They argued that form-based methodology may be limited when modelling such constraints beyond a certain degree of temporal complexity. Based on Allen's time interval algebra, a time query can be modelled by placing a simple horizontal bar that represents a symbolic time interval. They have also applied two extensions to allow the application to handle the complex time models. Periodic intervals allow inferences about the relative time distance between patient events and time interval modifiers. It supports counting and excluding patient events as well as constraining numerical value. They also demonstrated the formation of a database query from this notation and also provided a typical implementation consisting of a temporary front end for query modelling and an experimental back end that connects to an i2b2 system. These modelling techniques are evaluated in the MIMIC-III database and proven that they can be used to model typical temporal phenotypic queries.

Anselma, L et al.[34] showed that Temporal information has a vital role in medicine. The relational approaches in the present scenario have certain drawbacks in treating “now-relative” data (i.e., sticking at the present moment). The proposed methodology handles now-relative relational data which can be paired with different decision support systems. They proposed a new data model and temporal algebra to support Allen's possible necessary temporal relationships. The case study was also done to measure the impact of their methodology and its theoretical and computational properties.

Eleftheriou et al.[35] addressed one of the biggest challenges of Managers from complex organisations, which is to choose whether it is worthwhile to migrate an application and data to new software or not. These choices are difficult for a massive organization. not only costs but risks are underseen. In this paper, they proposed a new method aiming to predict the locations of high cost and risk when existing non-temporal data needs to move to new temporal development. This new low-cost method, which uses inexpensive social and technical information to achieve a lightweight model called the Data Path Model that sets the path of datasets from their original location in the knowledge infrastructure to the new one. via a complex network, People and organizations.

Kvet, M. At el.[36] proposed a new methodology of effective data retrieval by identifying effective data block location even if no suitable index for the query is available in the system, learning to optimized performance of the entire system in terms of processing time and costs. They introduced the concept of the master index, which projects only the relevant blocks due to which there is no need for sequential block scanning which leads resources conservation and can have a significant impact on sensor oriented data, as well.

III. PRELIMINARIES

Real-time applications in database management system are quite large in size to fit in main memory, so external memory is vital to handle these and also needed for efficient exploitation of space and I/O (Input/output) activities. As data relocate from external storage disc to internal storage RAM (Random Access Memory) in the form of segments (or blocks), so the I/O time is the time elapsed while transferring one segment from disc to main memory and it is considered as one I/O, a concept used in [18]. The accomplishment of rendered approaches is evaluated on the basis of requisite capacity to store the data structure and time required to answer the time-varying query. Traditional databases were small in size so data structure was developed specifically to internal memory as they are not adequate for secondary storage database. Due to the growing size of data, the secondary memory data structure is proposed which have more capacity than the main memory and can be further optimized in the context of the query and space-time complexity.
The suitable features of access frameworks are Index pagination- It represents the relocation of the number of segments per change from disk to main memory and vice versa. And Query Clumping- refers to the bunching of the data segments based on the retrieval of answered problems/queries that are rationally (or closely) similar which meliorate the query implementation time faster. Clumping is very difficult to maintain in valid-time durations as changes happen frequently in the object collections which can be updated at any point in their expertise and as a result, it requires frequent restructuring of clump objects that is not an efficient way.
The touchstones used in this paper are: c represents the number of interval-objects in the collection, s is the size of the time-varying query to be accessed, and p is the block (or segment) size which consists of the number of records registered per segment. The approaches should be estimated in the context of job loads that can be considered as a set of segments.

3.1 Internal Memory Access Structures

Access approaches can be classified based on their access framework.
Internal Storage access Framework (ISF)-The most popular approaches in ([19], [20] and [21]) defined are Segment tree (ST), Interval tree (IT), Priority search tree (PST).

Segment tree is a primary storage access framework to register end-points of time duration in a non-internal node of B-tree.

Interval Tree uses the concept of augmented self-balanced binary search tree like red/black tree, AVL tree and renders the intervals into the number of range queries.

The priority search tree is a combination of a balanced binary search tree (BST) and heap (or priority queue). It has been utilised to register the intervals into a two-dimensional point (x, y) and answers semi-infinite range queries.

The above structures support the one-dimensional range query (or stabbing query) which takes $O\log_2 c$+squery time and $O(\log_2 c)$update time for maintaining the index structure. Interval tree and prior-

ity search tree occupies linear space that is O(c) and segment tree required O($clog_2c$)space. PST can answer more severe problem which can be more than one magnitude in range. The time and capacity estimated can be summarised in Table 1.

Table 1.Represents internal storage access approaches.

Method	Space	Update time	Query time
PST	$O(c)$	$O(log_2 c)$	$O(log_2 c + s)$
ST	$O(clog_2 c)$	$O(log_2 c)$	$O(log_2 c + s)$
IT	$O(c)$	$O(log_2 c)$	$O(log_2 c + s)$

The time and space utilized by internal memory data structures can further be optimized by extending them in the external storage structure. Space can be meliorated from O(c) to O(c/p), query time can be reduced up to O(logpc+s/p) and update time to maintain data framework optimized by Ologpc time. This represents the secondary storage lower bound for one-dimensional range query which is the worst-case behaviour for the introduced approach; it is hard to estimate average-case behaviour for the access structures. In Fig. 2 the result depicts the internal storage structures that can be further optimized by adopting the external storage structure on disk with an accruing size of segments in terms of numbers of records occupied. Since the value of p increases, the query time (QT) and updates time (UT) has been meliorated.

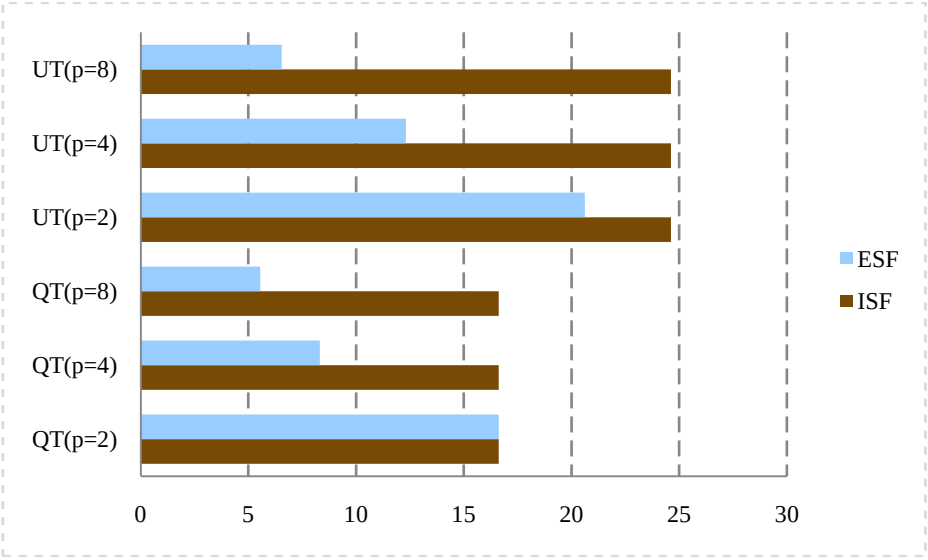


Fig.2. Comparison of the query and update time in ISF and ESF

External Storage Access Framework (ESF)-In previous research numerous of data structure has been proposed for valid-time durations which extend the primary structure. Salzberg and Tsotras present the comprehensive study of numerous access methods which were developed for efficient query retrieval in temporal databases such as valid time, machine time and Bi-temporal database but primarily focused on machine time access structure and provided the detailed analysis of time and space complexity for time-evolving nature of data. Some of them are external segment R-tree [12], path caching, Meta block tree, external memory interval tree, binary blocked interval tree, and time-polygon index [22]. These methods aggrandized the existing internal storage access structure and attempted to render the more optimized approaches for the vast volume of complex time-varying database applications by utilising external storage framework. These frameworks are utilized for improving the performance in terms of answering one-dimensional interval queries and for the time required for restructuring the data structure while single update takes place in the database. The indexing approaches require extra capacity (or disk overhead) on disk along with database itself to meliorate the query performance, so the secondary access structures are more suitable for large datasets.

3.2 External Memory Access approaches

As valid-time access approaches require maintenance of the time-duration of events which are mercurial in nature. Consequently, traditional R-tree has been used to index valid-time durations in which data has been partitioned into the rectangular area which minimizes the overlapping and coverage of set of rectangles corresponding to leaves and internal nodes. Multidimensional R-tree method supports more complex queries than one-dimensional range-based query. In the R-tree approach, records which are rationally closed can be clumped based on the length of a set of epoch objects. Due to the mixture of short and large time-durations, it reflects the problems of overlapping which affects the query and update performance. Kolovson and Stonebraker introduced another variation named as Segment R tree (SR tree) which combines the R-tree with internal storage access structure and segment tree to store line segment. This method translates the time-durations into single point in two-dimensional spaces and Point Access Method was adopted to index them. For example, suppose u and v are the two time dimensions denoting start time point in u direction and end time point in v direction and the interval I=(u1,v1) is represented in two dimensional planes.

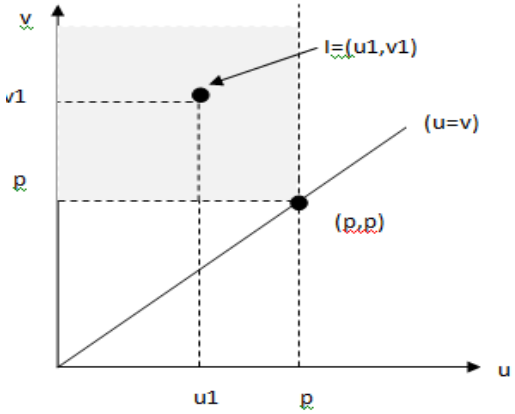


Fig. 3. Transformation of valid-time duration into a single point $<u_1, v_1>$.

The interval I consists of point p iff it satisfies the condition u≤p≤v. Fig. 3 shows the mapping of interval into a point <u1,v1> in two dimensional spaces. The query p lies in the box [0,p] and [p,∞]. Since u is always less than v, denoted by the diagonal where u=v and all interval points are represented in the region which are above the diagonal. This kind of mapping is also supported in PST which externalize in [9]. The access framework for indexing coercions and indexing classes were introduced for the concept of coercion programming and object-oriented programming. The constraint query language (CQL) is merged with two-dimensional time duration search which retrenched the indexing coercion into maintenance of active duration on external storage problem (special case of two-dimensional range search). The time durations were represented as indexing checks and objects which consist of forest of class hierarchy and a new external storage access framework called Metablock-Tree was introduced to solve the diagonal-query problem. The introduced access structure was semi-dynamic in nature which splited the area above the diagonal into blocks/segment and assigned the records to them horizontally and vertically. The method supports insertion and retrieval operations but not the commutation of time-duration events.

Path caching technique has been introduced by [10] in which various internal storage access approaches such as PST, and ST has been augmented into secondary storage access framework to provide the solution for special cases of two dimensional range retrieval query and attempted to provide optimized or nearly-optimized query performance with some extra space overhead as compared to primary access approaches. External segment-tree [12] extends the segment tree primary memory access structure attempting to provide a solution for point-enclosure problem (also called cover-equilibrating problem). Let v be a universal set, consists of all possible endpoint duration which represents valid-time interval then the time required to answer the stabbing queries search problem is O(log2v+s), and update per changes requires Olog2v time with requirement of O((c/p)log2v) space. EST is augmented to solve the query problem with key predicate by engrafting b-tree data structure.

Arge and Vitter in [23] rendered a new approach called External Interval-Tree which externalizes the primary storage Interval tree data structure proposed by Edelsbrunner which contended an I/O optimal solution for the problem of mercurial maintenance of time duration object and overcomes the limitation in the previous approaches such as meta-block tree, and path caching. Basically, it utilized two secondary access structures: binary tree (or b-tree) and the corner framework and provides the solution for one-dimensional stabbing queries. The main concept was to replace the fan-out from 2 to p , where p is the size

of segment/page, which was beneficial for successfully externalization of the internal storage access structure into secondary storage structure. The time durations were divided into p sub-durations and these durations were called as slabs that can be effectuated using binary tree and corner framework. Interval B-tree of Ang & Tan [24] externalizes the interval tree primary structure to resolve the problem recognized in time-index method.

Bozkaya and Ozsoyoglu in [25] developed an approach, Interval B+-tree to overcome the limitation of time index method in which interval tree structure was transformed into secondary access structure for increasing time-varying query efficiency. It defines lower endpoint of intervals acting as a key and queries which consist of upper endpoints were not considered. It was developed for valid-time dimensions in which the concept of now (current-time data) and related to future (that tends to infinite) were also considered and the main focus was to answer the pure time-slice queries. The method was suitable for the retrieval of long time-duration because it splits the leaves data into parts which improves the query performance by preventing large fruitless scans with the help of some extra space consumption.

Nascimento and Dunham introduced an approach MAP21 (expand as map two one) which maps valid-time bounded periods of historical databases into single point and indexed these points using B+- tree. It requires that time-duration endpoints should be known in advance for mapping i.e. the database should be stable and does not have to change frequently. The approach was developed for answering the range-based queries efficiently and also to ensure effective space exploitation as compared to Time-Indexed method. Let the interval be I=(b,e) where b and e denotes the beginning and end time instants and d is the maximum number of digits to represent the time-duration endpoints then mapping can be done by using function

$$F(b,e) = b * 10^d + e$$

The above function maps the time-durations into single point representation and inverse of function is also specified to get back the actual values of the time-durations. This approach is static in nature and does not support the dynamic interval management problem efficiently. Relational Interval tree is proposed to exploit the existing relational storage access structure which uses the original interval-tree and is implemented on existing relational database management system (RDBMS). According to author, an access method can be created by augmenting existing one or by creating a new approach from scratch but the integration of these structures were hardly supported by existing RDBMS. So a new approach has been proposed which was effectuated on existing database using in-built access framework and can be easily integrated with existing RDBMS and ORDBMS (Object RDBMS). The approach was developed for answering the intersection based queries in which two relational-indexes were created for starting and ending point in time duration. The time required for resolving the query was O(hlogpc+s/p), where h is the height of primary virtual framework.

The triangular decomposition tree (TD tree) is another access approach which leverages the existing in-built relational approach for managing time-varying objects using virtual index framework and space-division method. Virtual index depends on the delegation of time-durations in two dimensional planes which transform the query problem into spatial representation. The method divides the basic triangular region into sub-triangles and represents them in the leaves of resultant unbalanced binary tree where actually data is stored. Main idea was proposed to answer the multitudinous time-varying query forms such as point queries, intersection based queries and time-duration association queries using the same algorithm. The time-slice or point-based queries were solved in O(m/p+logpc+s/p), where d represents the size of directory-table which consists of the details such as identifiers and number of records per leaf, about all leaves in a tree.

Finally, in table 2 we have summarized the various access approaches proposed for valid time-durations along with the time and capacity requisite based on the parameters that are defined above in preliminaries section.

Table 2.Performance features of valid-time interval access approaches.

Method	Space	Update time	Query time
Time Index	$O(c^2/p)$	$O(c/p)$	$O\left(log_p c + s/p\right)$
Segment R-Tree	$O\left((c/p)log_p c\right)$	$O(c/p)$	$O(c/p)$
Metablock Tree	$O(c/p)$	$O\left(log_p c + \left(log_p c\right)^2/p\right)$	$O\left(log_p c + s/p\right)$

Path Caching	$O((c/p)log_2 log_2 p)$	$O(log_p c)$	$O(log_p c + s/p)$
External Segment trees	$O((c/p)log_2 v)$	$O(log_2 v)$	$O(log_2 v + s)$
External Interval Tree	$O(c/p)$	$O(log_p c)$	$O(log_p c + s/p)$
MAP 21	$O(c)$	$O(log_2 c)$	$O(log_2 c + s)$
RI Tree	$O(c/p)$	$O(log_p c)$	$O(log_p c + s/p)$
TD-tree	$O(c/p)$	-	$O(m/p + log_p c + s/p)$

3.3 Temporal Model Used

Time-oriented system can be implemented either by creating a new system from scratch or by augmenting the existing structure from the conventional database management system (DBMS). Developing a new system is more costly and time consuming as it requires too much time and manpower and incompetent environment new system deployment becomes quite challenging task. In another approach, the existing system is not affected and the implemented data-model can be easily integrated with commercial database application. We have implemented valid-time database using second approach in which row-based timestamping model is used and data is represented in first-normal form (1NF) as in [26].The schema adopted for representing valid time relation $R^{VT}$is a collection of time-varying and non-time varying attributes is defined as

R^{VT}=<C1, C2,..,CN|V_S,V_E>, here V_S<V_E

It consists of non-temporal finite set of attributes C1, C2,..,CN which encode the states of an event and temporal attributes V_S,V_E represent valid-time durations for that where V_S is start time and V_E is end time for the interval object. The relation R^{VT} represents the row-based timestamping model which is utilized for performing experiment and optimizing the query results. The schema is defined using PostgreSQL environment and consists of the following fields given as.

Loan_id	Country	Loan_status	Interest_rate	Project	Amount	Start_date	End_date

The schema that is used for evaluation of our approach consists of the following attributes:
Key Identifier - A unique constraint is assigned to the relation. In our relation R^{VT} loan_id and start_date as a composite field represents the key constraint.
Field – It comprises of time-varying or non-temporal attributes.
Start time – It is the left part of the time duration in a time range that is start_date.
End time - The right part of the time duration in a time range that is end_date.

3.4 Algorithm

Binary tree (B-tree) is a self-equi poised multiple-way search tree which unlike other search trees (like AVL search tree, Binary search tree, and red/black tree) must have the property of storing more than two children in every node and searching can be possible in dissonant ways depending on the number of children it points. It stores keys in ordered manner corresponding to the leaves of the tree and the leaf nodes are linked with each other which support range queries efficiently in logarithmic time as compared to sequential method. We have exploited Binary tree approach for valid-time duration to meliorate the time varying query retrieval process. The access method is applied on defined relation R^{VT} which optimizes the retrieval time for larger dataset.

3.5 Query Problem

The main focus is on to answer the time-varying queries using temporal operators. The problem statement that should be answered in efficient way can be defined by the following scenario. Suppose a universal set U_s comprises of a set of valid-time intervals which can be represented as:

$$U_s = V_{S_i}, V_{E_i}, where S_i < E_i \wedge 1 \leq i \leq c>$$

Here V_{S_i} represents the valid start time and V_{E_i} represents valid end time for an interval object and the interval $Q = <q_1, q_2>$ should be searched in interval collection which intersects the intervals in the set U_s. This problem is called as one dimensional time duration management problem. If a single instant q_1 is queried in set of valid time interval collection then it is said to be the stabbing query problem as defined in [27][29]. These are the most frequently used problems which are defined and solved by various access structure approaches.

IV. Performance Evaluation

The experiments are actualised on the well established hardware such as 4 GB RAM and Intel(R) Pentium(R) CPU N3700 @ 1.60GHz. PostgreSQL is an open- source database server which is used for deployment of database application and psql/pgAdmin for querying the time-varying data. The machine was kept in the state of isolation while empirical evaluation of temporal queries retrieval process was accomplished.

4.1 Dataset

We have compared and analysed our indexing approach with the existing method using real-time banking application scenario. The database handles the valid-time data i.e. past records, records suitable for the current state and even the future aspects dynamically which means that it can be altered randomly without following any order. The evaluation of both approaches is done by partitioning the data set into subsets in increasing size of records which reflects the performance boost when compared to the existing approach after applying indexing. The database comprises of a large number of fields as represented in preliminaries section to store the records for bank loan management system and reflects the legal time that is historical, current and related to future also.

4.2 Results

A performance evaluation of access method approach with respect to existing sequential method is performed over row based timestamping model. Experiments are performed over multiple or large number of fields. Our goal is to test both the approaches under different conditions. Performance of indexing approach and existing sequential method is evaluated on the basis two parameters: *query retrieval time* (in milliseconds) and number of records accessed that is the *size of answered query*. The following results show the Performance comparison of existing method with our approach that is using B-Tree and it is evident that our approach requires less access time. The performance improves when the query size increases. In Fig. 4, Fig. 5, and Fig. 6, results are evaluated with respect to the temporal query predicates which are specified using 'greater than' or 'less than' or 'equal to' operators on temporal attribute. Below the Fig. 4 depicts the result for query predicate that is greater than the single time instant.

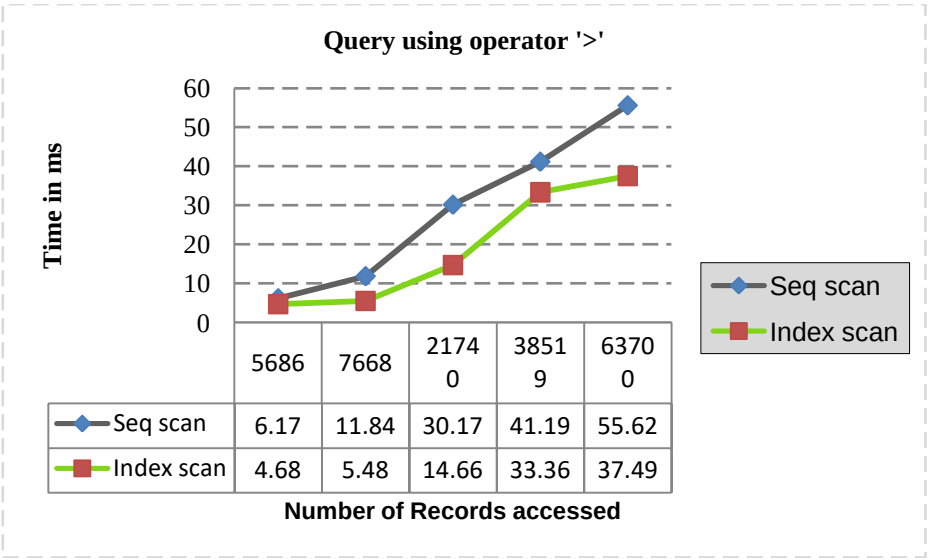


Fig.4. Greater than operator query result for indexing vs. sequential method.

Below the Fig. 5 depicts the result for query predicate that is greater than the time instant.

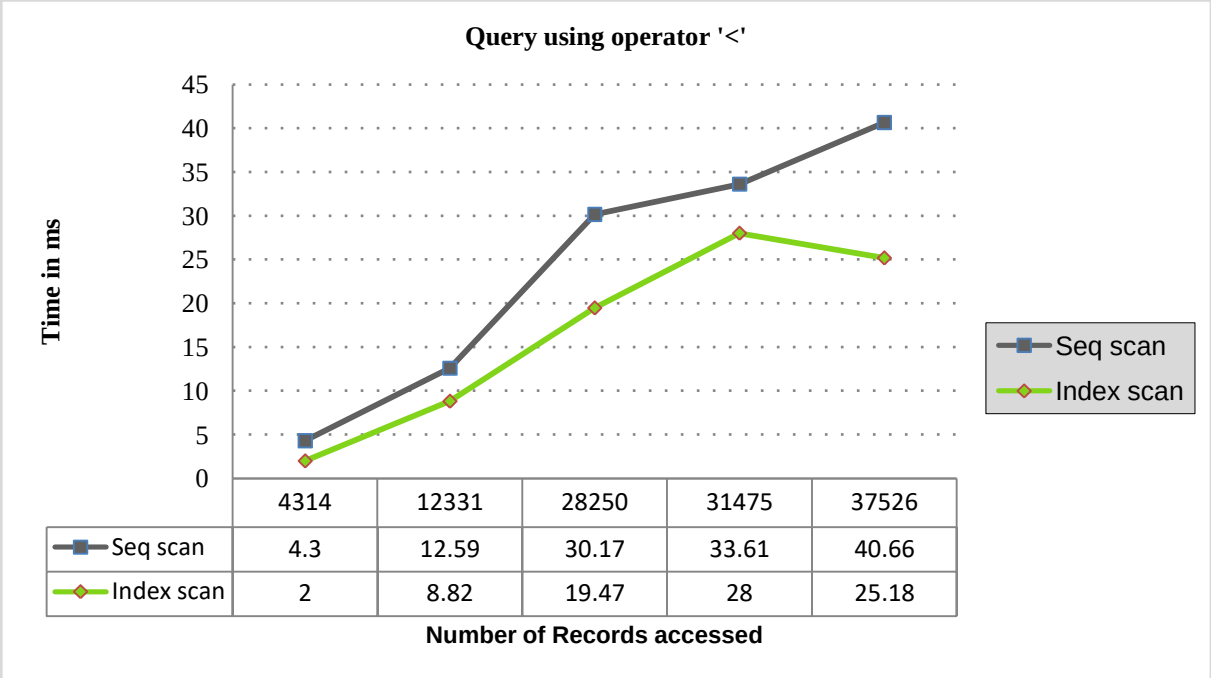


Fig.5. Less than operator query result for indexing vs. sequential method.

Below the Fig. 6 shows the result for query predicate that is equal to time instant which shows a huge difference between indexed and sequential scan.

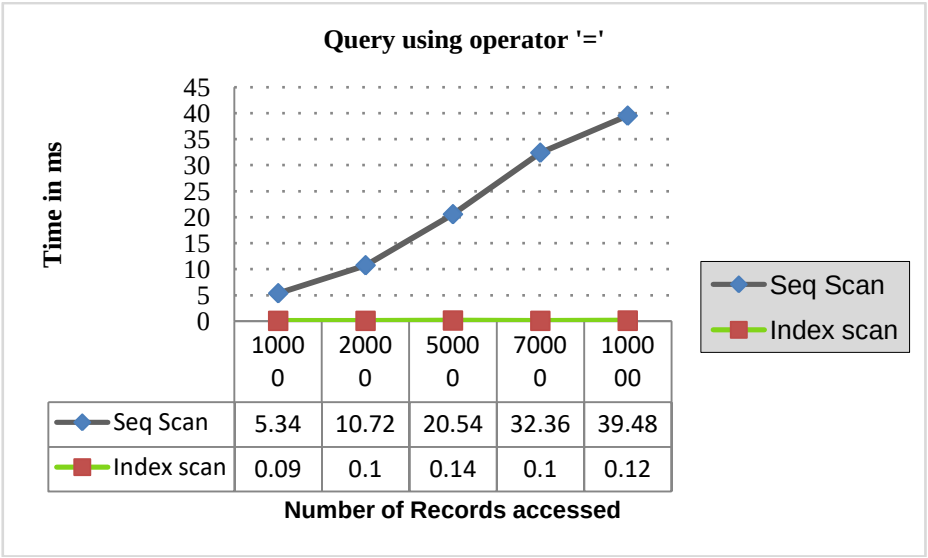


Fig. 6: Equal operator query result for indexing vs. sequential method.

And finally in Fig. 7 results shows the performance improvement for one dimensional range queries that are implemented using GiST (Generalized Search Tree) index structure as described in [28].

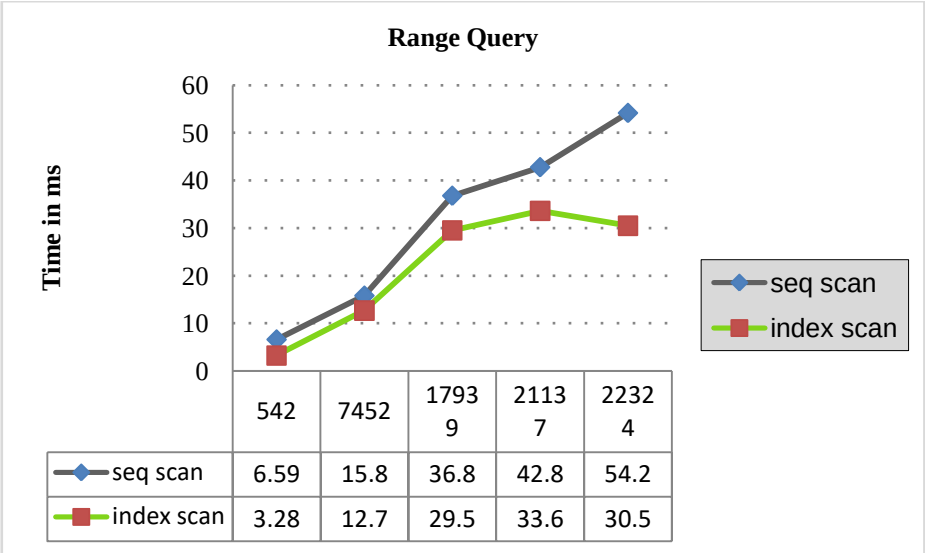


Fig.7. Range query results for existing vs. indexing method

V. CONCLUSION AND FUTURE WORK

Time-varying database applications are huge in size for its ever growing traits. These databases maintain the state of interval objects which is called as the collection of snapshot databases. Whenever changes take place data is not overwrote but it records every state in timely fashion. The valid-time database usually handles the dynamic nature of the objects. Managing valid-time intervals is completely different from that of machine time intervals. An approach for retrieving and handling the huge volume for time-varying data is a vital obligation. We have implemented the row based timestamping data model for valid-time relation and exploited the binary balanced search tree approach which results in significant improvement in query execution time as compared to existing method supported by database environment. In this paper, the various access structures for dynamic collection of intervals are classified and represented with their performance features in summarized form. Most of the approaches render the solution for one dimensional range search problem but for the multidimensional range searching is still an issue and requires further research on development of efficient access structure. Indexing, the uncertain (that is not exactly known) data in valid time domain is also an important aspect in future research.

REFERENCES

1.Snodgrass, R. T. & Ahn, I. Temporal databases. IEEE Computer 19. 9, 35–41 (Sept. 1986).

2.Kumar, S.,& Rishi, R. Retrieval of Meteorological Data using Temporal Data Modeling. Indian Journal of Science and Technology. 9. 10.17485/ijst/2016/v9i37/99875. (2016).

3.Kumar, S.,& Rishi, R. A New Optimized Model to Handle Temporal Data using Open Source Database. Advances in Electrical and Computer Engineering. 17. 55-60. 10.4316/AECE.2017.02008. (2017)

4.Kumar, S.,& Jolly, A. "A Novel Study of Attribute time stamping models in Temporal Database." International Journal of Advanced Research in Computer Science (2017)

5.Kumar, Shailender . Performance Evaluation of Tuple Timestamp Multiple Historical Relation Data Model. Journal of Engineering and Applied Sciences, 13: 222-230. DOI:10.36478/jeasci.2018.222.230URL:https://medwelljournals.com/abstract/?doi=jeasci.2018.222.230(2018)

6.Guttman, A. R-trees: A dynamic index structure for spatial searching (Vol. 14, No. 2, pp. 47-57). ACM (1984).

7.Elmasri, R., Wuu, G. T., & Kim, Y. J. The time index: An access structure for temporal data. In Proceedings of the 16th International Conference on Very Large Data Bases (pp. 1-12). Morgan Kaufmann Publishers Inc. (1990, August).

8.Kolovson, C. P., & Stonebraker, M. Segment indexes: Dynamic indexing techniques for multi-dimensional interval data (Vol. 20, No. 2, pp. 138-147). ACM(1991).

9.Kanellakis, P. C., Ramaswamy, S., Vengroff, D. E., & Vitter, J. S. Indexing for data models with constraints and classes. In Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (pp. 233-243). ACM(1993, August).

10.Ramaswamy, S., & Subramanian, S. Path caching (extended abstract): a technique for optimal external searching. In Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (pp. 25-35). ACM (1994, May).
11.Kouramajian, V., Kamel, I., Elmasri, R., & Waheed, S. The time index+: an incremental access structure for temporal databases. In Proceedings of the third international conference on Information and knowledge management (pp. 296-303). ACM (1994, November).
12.Blankenagel, G., & Güting, R. H. External segment trees. Algorithmica, 12(6), 498-532 (1994).
13.Ramaswamy, S. Efficient indexing for constraint and temporal databases. In International Conference on Database Theory (pp. 419-431). Springer, Berlin, Heidelberg (1997, January).
14.Nascimento, M. A., & Dunham, M. H. Indexing valid time databases via B/sup+/-trees. IEEE Transactions on Knowledge and Data Engineering, 11(6), 929-947(1999).
15.Kriegel, H. P., Pötke, M., & Seidl, T. Managing intervals efficiently in object-relational databases. In VLDB (pp. 407-418) (2000, September).
16.Kriegel, H. P., Pötke, M., & Seidl, T. Object-relational indexing for general interval relationships. In SSTD (pp. 522-542) (2001, July).
17.Stantic, B., Topor, R., Terry, J., & Sattar, A. Advanced indexing technique for temporal data. Computer Science and Information Systems, (16), 679-703(2010).
18.Salzberg, B., & Tsotras, V. J. Comparison of access methods for time-evolving data. ACM Computing Surveys (CSUR), 31(2), 158-221 (1999).
19.Bentley, J. L. Solutions to Klee's rectangle problems. Technical report, Carnegie-Mellon Univ., Pittsburgh, PA (1977).
20.Edelsbrunner, H. A new approach to rectangle intersections part I. International Journal of Computer Mathematics, 13(3-4), 209-219 (1983).
21.McCreight, E. M. Priority search trees. SIAM Journal on Computing, 14(2), 257-276 (1985).
22.Shen, H., Ooi, B. C., & Lu, H. The TP-Index: A dynamic and efficient indexing mechanism for temporal databases. In Data Engineering, 1994. Proceedings. 10th International Conference (pp. 274-281). IEEE (1994, February).
23.Arge, L., & Vitter, J. S. Optimal dynamic interval management in external memory. In Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on (pp. 560-569). IEEE (1996, October).
24.Ang, C. H., & Tan, K. P. The interval B-tree. Information Processing Letters, 53(2), 85-89(1995).
25.Bozkaya, T., & Ozsoyoglu, M. Indexing valid time intervals. In Database and Expert Systems Applications (pp. 541-550). Springer Berlin/Heidelberg (1998).
26.Halawani, S. M., AlBidewi, I., Ahmad, A. R., & Al-Romema, N. A.: Retrieval optimization technique for tuple timestamp historical relation temporal data model. Journal of Computer Science, 8(2), 243(2012).
27.Moro, M. M., & Tsotras, V. J. Valid-Time Indexing. In Encyclopedia of Database Systems (pp. 3254-3258). Springer US (2009).
28.Hellerstein, J. M., Naughton, J. F., & Pfeffer, A. Generalized search trees for database systems (pp. 562-573). September (1995).
29.Böhlen, M. H. Temporal database system implementations. ACM Sigmod Record, 24(4), 53-60 (1995).
30.Revesz P. Temporal Constraints. In: Liu L., Özsu M.T. (eds) Encyclopedia of Database Systems. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-8265-9_391(2018)
31.Chomicki J., Toman D. Temporal Logic in Database Query Languages. In: Liu L., Özsu M.T. (eds) Encyclopedia of Database Systems. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-8265-9_402(2018)
32.Anselma, L., Piovesan, L., Stantic, B., & Terenziani, P. Representing and querying now-relative relational medical data. Artificial Intelligence in Medicine, 86, 33–52. https://doi.org/10.1016/j.artmed.2018.01.004(2018)
33.Eleftheriou, I., Embury, S. M., Moden, R., Dobinson, P., & Brass, A. Data journeys: Identifying social and technical barriers to data movement in large, complex organisations. Journal of Biomedical Informatics ,78,102–122. https://doi.org/10.1016/j.jbi.2017.12.001(2018)
34.Mate, S., Bürkle, T., Kapsner, L. A., Toddenroth, D., Kampf, M. O., Sedlmayr, M., Castellanos, I., Prokosch, H.-U., & Kraus, S. . A method for the graphical modeling of relative temporal constraints. Journal of Biomedical Informatics, 100, 103314. https://doi.org/10.1016/j.jbi.2019.103314 (2019)
35.Eleftheriou, I., Embury, S. M., Moden, R., Dobinson, P., & Brass, A. Data journeys: Identifying social and technical barriers to data movement in large, complex organisations. Journal of Biomedical Informatics ,78,102–122. https://doi.org/10.1016/j.jbi.2017.12.001(2020)
36.Kvet, M., & Matiasko, K. . Data Block and Tuple Identification Using Master Index. Sensors, 20(7), 1848. https://doi.org/10.3390/s20071848(2020)