International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

An Efficient and Robust Tuple Timestamp Hybrid Historical Relational Data Model

Lalit Gandhi[1], Rahul Rishi[2], Sonia Sharma[3]
[1]University Institute of Engineering and Technology,
Maharshi Dayanand University, Rohtak - 124001, India
gandhi_lalit@yahoo.co.in
[2]University Institute of Engineering and Technology,
Maharshi Dayanand University, Rohtak - 124001, India
rahulrishi@mdu.ac.in
[3]University Institute of Engineering and Technology,
Maharshi Dayanand University, Rohtak - 124001, India
soniasharma.rp.uiet@mdu.ac.in

Abstract—This paper proposes a novel, efficient and robust tuple time stamped hybrid historical relational model for dealing with temporal data. The primary goal of developing this model is to make it easier to manage historical data robustly with minimal space requirements and retrieve it more quickly and efficiently. The model's efficiency and results were revealed when it was applied to an employee database. The proposed model's performance in terms of query execution time and space requirements is compared to a single relational data model. The obtained results show that the proposed model is approximately 20% faster than the conventional single relational data model. Memory consumption results also show that the proposed model's memory cost at different frequencies is significantly reduced, which is approximately 30% less than the single relational data model for a set of queries. Because net cost is strongly related to query execution time and memory cost, the suggested model's net cost is also significantly reduced. The proposed tuple timestamp hybrid historical model acts as generic, accurate and robust model. It provides the same functionality as previous versions, as well as hybrid functionality of previously proposed models, with a significant improvement in query execution speed and memory usage. This model is effective and reliable for the use in a wide range of temporal database fields, including insurance, geographic information systems, stocks and finance (e.g. Finacle in Banking), data warehousing, scientific databases, legal case histories, and medical records.

Keywords — tuple timestamp, valid time, relational data, query execution time, memory cost, net cost.

I. INTRODUCTION

The database that can record previous history data values along with current snapshot of data are temporal databases. Many computer applications rely on the ability of databases to represent this temporal aspect of the real world, including econometrics, finance, inventory control, accounting, legal, medical records, land and geographical information systems, and airline reservations. Temporal databases include regular time series data such as stock ticks, EEG, and event sequences such as sensor readings, packet traces, medical records, weblog data, and temporal data such as relations with timestamped tuples and databases with versioning [1]. The query execution time and storage space requirements are significant temporal database dimensions that need optimization. Temporal databases are considered efficient if the query execution time is kept as short as possible. Furthermore, there should be no redundancy, and the memory requirements for an effective temporal database should be minimal. As a result, it is critical to create a database that is efficient in terms of memory utilization, query execution, and thus overall net cost.

Because traditional databases lack the capacity to keep multiple versions of records, a variety of temporal data models have been proposed by many academicians and researchers. They preferred to layer temporal components on top of traditional databases and typically on relational database [2-3]. A temporal database management system called Roles Mode (TF-ORM) is built on top of a conventional database [4-5]. A temporal relational model for managing patient historical data [6], bi-temporal conceptual data model (BCDM) [7], Tera-data [8-11], tuple timestamped single relational model [12-13], tuple time stamp history relational model [14-15], and multiple history relational model [16] are among temporal data models that are developed using relational database as foundation [16-26]. The tuple timestamp single relational (TTSR) model stores all of its attributes, whether static or dynamic, in a single relation, resulting in increased relation redundancy and storage

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

requirements. Tuple timestamp historical relation (TTHR) uses two relations: one for storing current instances and another for storing past occurrences. The tuple timestamp multiple historical relation model (TTMHR) [16] was developed to deal with temporally diverse dynamic features. It made extensive use of relations to handle dynamic properties. Although these earlier models are easy to implement, they are not so efficient in terms of query execution speed. However, the redundancy caused by tuple time stamping resulted in excessive memory requirements. As a result, these models have a relatively high net cost. Despite the fact that several temporal data models have been proposed, none has received universal acceptance. As a result, a robust temporal data model with efficient query execution time, minimal memory cost, and enhanced net cost is still required.

The Tuple Timestamp Hybrid Historical Relational (TTHHR) model proposed here records temporally homogeneous and heterogeneous characteristics with valid time separately. This improved model employs valid time and tuple time stamping to account for the start and end time validity [26-27]. It keeps a single history relation for attributes with a consistent temporal pattern and multiple history relations for attributes with a variable temporal pattern when recording time entries. The model is robust and retains the behavior of previous versions; it acts as a tuple timestamp history relation model if all of the dynamic attributes of the relation are temporally homogeneous, and it does so with greater efficiency. If all of the relation's attributes are temporally heterogeneous, the model behaves similarly to a tuple timestamp multiple historical relational model. If, on the other hand, there is a mix of both homogeneous and heterogeneous attributes, the model exhibits hybrid behavior. Furthermore, the proposed model is hybrid and generic in nature, allowing it to handle all types of data while requiring less memory and requiring less query execution time, resulting in improved results and a lower net cost in comparison [29]. The model is deployed and evaluated for a hypothetical enterprise with ninety thousand employee records to calculate salary and other earnings such as reimbursements, arrears, and so on. The results show a significant decrease of approximately 30% reduction in memory storage space requirements, as well as a significant reduction in query execution time, i.e. 20% enhancement. As a result, the model produces better net cost results than the previous models.

Section II shows the architecture of the proposed tuple timestamp hybrid historical Relational data model. Section III explains how the proposed model is implemented, Section IV presents the results & discussion and Section V concludes the paper.

II. ARCHITECTURE OF PROPOSED MODEL

Figure 1 depicts the architectural design of the proposed model. The application interface allows the user to communicate with the temporal database. Attributes are filtered and classified into three categories: static attributes temporally homogeneous dynamic attributes and temporally heterogeneous dynamic attributes. Users can manage both historical and current data instances through the application interface. Only the current static single relation is required to save static attribute data values. A single current relation is required to store temporally homogeneous time-varying attributes, and a single History relation is required to preserve history values. To store data for temporally heterogeneous attributes, multiple (equivalent to the number of heterogeneous attributes) current and historical relations are required. Data insertion is recorded using the current relation only. When an attribute is changed, the trigger is activated, and previous values for that attribute are transferred into a historical relation, updating that specific current relation with the latest values. The tuple's past values are sent to the history table, and the higher bound of the time range is set to match the lower bound of the current table's time range. As a result, all current and historical values are kept in separate tables, each with its own time period. No records are erased because delete transactions are not permitted in temporal databases. The historical relations are also preserved using tuple time stamping. Valid-time is used as the time element in the proposed model's tuples. The primary key of the history table is the same as the primary key of the current table, as is the time range of the history table. B-tree indexing is used to tune query execution & improved performance

III. IMPLEMENTATION OF PROPOSED MODEL

To implement and evaluate the proposed TTHHR model, salary and other earnings dataset of around ninety thousand records for employees of a hypothetical organization is used. Figure 2 depicts the pre-processing of the dataset. The "tsrange" data type is used to timestamp the tuples. The original relations are segmented into static, temporally homogeneous, and temporally heterogeneous relations, allowing queries to be performed in accordance with the proposal. The attributes are separated so that relations and thus data can be recorded and preserved only according to their validity. The proposed model is found to be 1.2 times faster than the conventional tuple timestamp single relational model for a set of queries (TTSR). Furthermore, the proposed model has a lower memory cost by about 30%, resulting in a significant net cost reduction.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

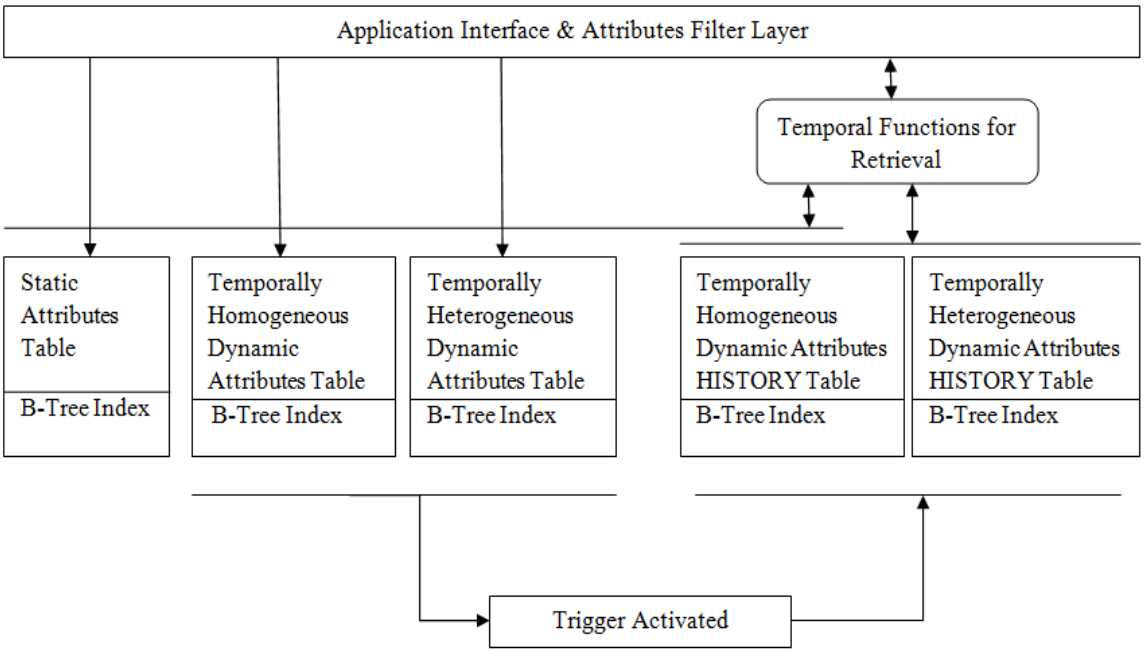

Figure 1 Architecture of proposed model

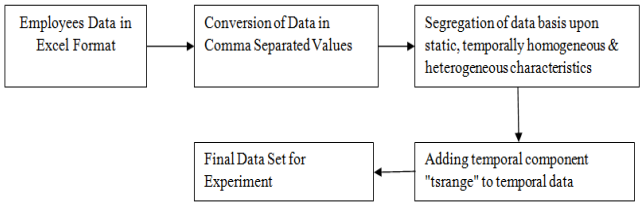

Figure 2 Dataset Processing

Figure 3 depicts the entity relationship diagram for the proposed tuple timestamp hybrid historical model's employee database. To group all static attributes of employees in the employee table, only one relation is required. Because the values for these attributes do not change over time, no history table is required. Dynamic attributes are classified according to their temporal nature. The attributes that change over the same time interval, such as BASIC SALARY, DA, HRA, MEDICAL ALLOWANCE, and TOTAL SALARY, are temporally homogeneous because DA, HRA, MEDICAL ALLOWANCE, and TOTAL SALARY are dependent on BASIC SALARY, and any change in BASIC SALARY has a direct impact on all other related attributes over the same time interval. Temporally heterogeneous attributes change at different time intervals, as shown in the diagram- REMUNERATION, ARREAR, and CHILD ALLOWANCE are examples of temporal heterogeneous attributes.

In the Entity Relationship Diagram, history tables are denoted by the suffix "hist," whereas current snapshots of the database are denoted by the suffix "fore." The employee table contains all static attributes. The earnings_fore table contains all temporally homogeneous attributes and a parallel history table, with EMPNO as the primary key and analogous time values. When the earnings_fore database is updated, the earnings_hist table is populated with previous values using a trigger. EMPNO in conjunction with valid-time serves as the earnings_hist table's primary key. There are also a number of other temporally diverse attributes, such as arrears, remuneration, and child allowance. Separate relations are designed for each attribute. EMPNO in conjunction with time values serve as the primary key for these heterogeneous relations such as ARREAR_FORE, REMUNERATION_FORE, CHILD_ALLOWANCE_FORE. When an update is performed on a “fore” relation, the corresponding history table ARREAR_HIST, REMUNERATION_HIST or CHILD_ALLOWANCE_HIST relation is populated with the previous values. When an update operation is performed on a fore table in the current database, the before update trigger is activated, and previous values in the relevant history table are transferred. The start valid time of the updated tuple is set in the fore table to coincide with the valid end time of the related history table.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
Article Received: 18 December 2022 Revised: 26 January 2023 Accepted: 15 February 2023

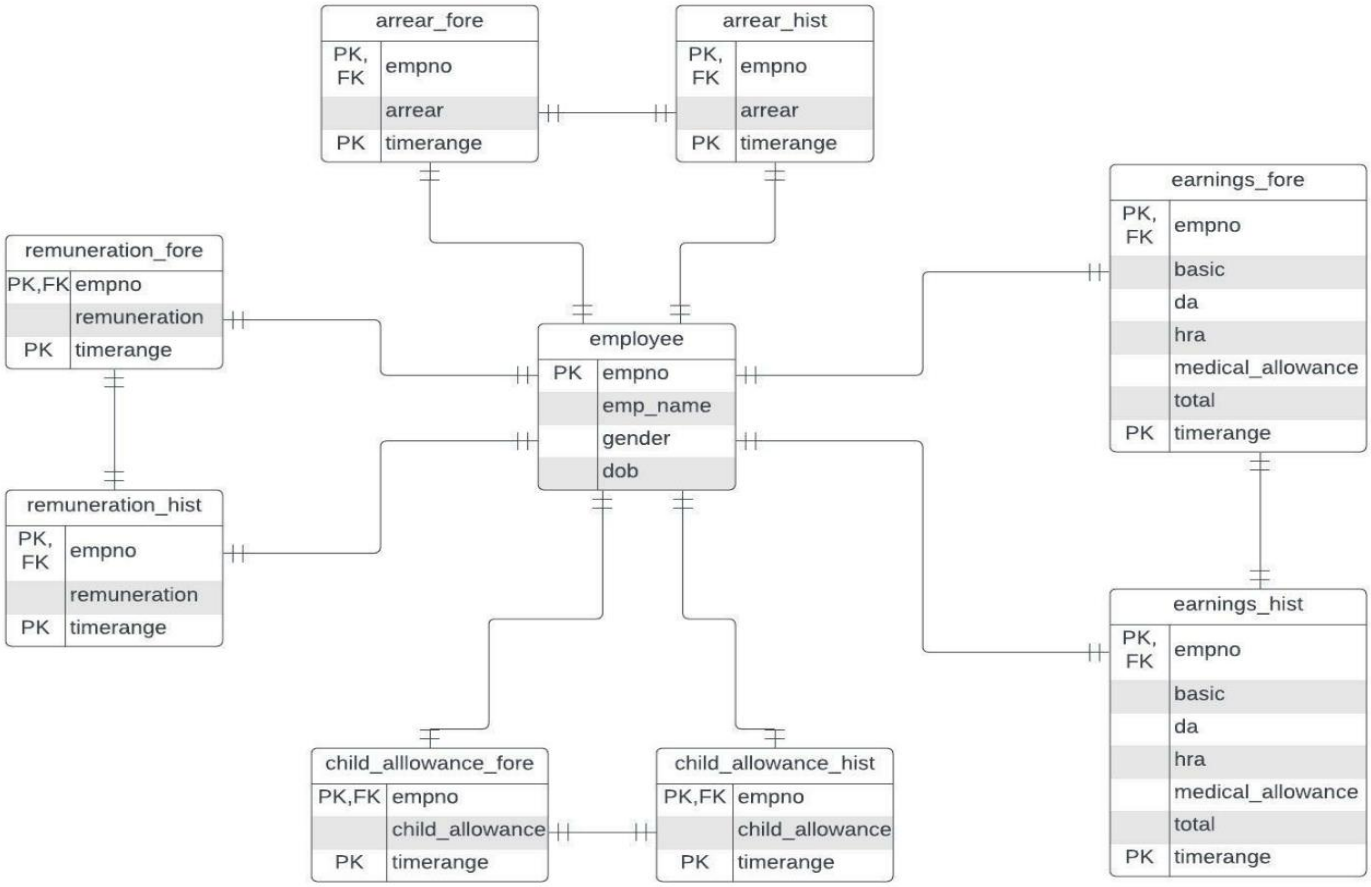

Figure 3 Entity Relationship Diagram

Because 'm' dynamic attributes are of temporally homogeneous type and there are 'n' total dynamic attributes. To record the data values for these m temporally homogeneous attributes appended with primary key and valid time, only one table and corresponding single history table are required. Rest n-m temporally heterogeneous attributes require n-m tables to record dynamic heterogeneous attribute values along with the primary key and valid time. To keep the history values for these relations, n-m history relations are also required. The proposed model supports the Insertion, Update and retrieval. Deletion is not permitted in temporal databases. Following are the algorithms for DML process.

a. INSERTION in temporal database

Only the current (fore) tables are exercised for insert transactions in temporal databases. Data values can be inserted by user for static and dynamic attributes. Valid start and end time are also inserted by user for the dynamic attributes with help of time range data types.

ALGORITHM TO INSERT NEW TUPLES IN TEMPORAL TABLES

```
INSERT INTO TABLE_NAME (COL1,COL2,….,COLN,TIME_RANGE )VALUES
(VALUE1,VALUE2,…….VALUEN,TSRANGE(DATE1,DATE2));
IF (PRIMARY KEY ALREADY EXISTS IN TABLE_NAME) THEN RETURN
PRIMARY_KEY_CONSTRAINT_VIOLATION
ELSE
RETURN INSERTION SUCCESSFUL
ENDIF
```

Example:

```
INSERTION OF DATA IN STATIC TABLE -
INSERT INTO EMPLOYEES (EMP_NO, EMP_NAME, GENDER, DOB) VALUES
(111,'ANSHUL','M', '1975/09/17');

INSERTION OF DATA IN DYNAMIC TABLE -
INSERT INTO EARNINGS_FORE (EMP_NO, BASIC, DA, HRA,
MEDICAL_ALLOWANCE, TOTAL, TIME_RANGE)
VALUES (111, 10200, 8000, 4000, 1000, 23200 TSRANGE
('2018/01/01 00:00:00',
'2018/01/31 23:59:00', '[]'));
```

b. RETRIEVAL from temporal database

Data retrieval from the proposed TTHHR model can be done via four different ways. In the first case if user requires only the static data, therefore user can directly access the data

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

from static table. Secondly, if the user intends to extract the dynamic data for current instance only then he can access from the 'fore' tables only. In case user needs the data that fall in history time period, then it can be accessed from 'hist' tables. Finally, if the data is required for time period that overlaps current and history time period, it can be accessed by joining the 'fore' and 'hist' tables for that particular time range.

```
ALGORITHM FOR RETRIEVAL OF DATA:
SELECT * FROM FUNC (PARAMETERS, TIME_RANGE)
        IF (TIME-RANGE IS NOT THERE AND USER NEED STATIC DATA)
        SELECT TUPLE/S FROM A STATIC TABLE WHEN THE CRITERIA
        MEETS THE PARAMETERS
STATIC TABLE TUPLE/S RETURNED
        ELSEIF (TIME_PERIOD &> (SELECT FORE_TABLE.TIME_RANGE
        FROM FORE_TABLE   WHERE (CRITERIA MEETS THE
        PARAMETERS)))
        FORE_TABLE TUPLE/S RETURNED
        ELSEIF (TIME_PERIOD << (SELECT HIST_ TABLE.TIME_ RANGE
        FROM HIST_TABLE WHERE (CRITERIA MEETS THE
        PARAMETERS)))
        HIST_TABLE TUPLE/S RETURNED
        ELSE
        BOTH FORE_TABLE AND HIST_TABLE TUPLE/S RETURNED WHERE
        FORE_TABLE.TIME_RANGE && TIME_PERIOD AND
        HIST_TABLE.TIME_RANGE&&TIME_PERIOD
        ENDIF
Example
       SELECT * FROM EARNINGS_FORE WHERE TIMERANGE && TSRANGE
       ('2020-06-01 00:00:00', '2020-06-30 23:59:59','[]')
```

c. UPDATE of temporal database

Update on the dynamic temporal table is different from static tables. Whenever update is applied on the 'fore' table, the previous values of the tuple are transferred to the 'hist' table. Before update trigger is activated as soon as there is any update is applied on the 'fore' table. The upper bound of the time-range attribute of hist table is set equal to the lower bound of the time-range attribute of the fore table, the older values of the present table are placed into the history table.

```
ALGORITHM FOR UPDATE IS AS FOLLOWS.
UPDATE FORE_TABLE SET
COLUMN1=VALUE1, COLUMN2=VALUE2 .... COLUMNN=VALUEN,
TIME_RANGE=TSRANGE (DATE1, DATE2)
WHERE CRITERIA MEETS PARAMETERS
BEFORE UPDATE TRIGGER <NAME-OF-TRIGGER>
IF (OLD.VALUES <> NEW.VALUES)
BEFORE UPDATE ON FORE_TABLE
EXECUTE PROCEDURE TRIGGER_PROC
INSERT INTO HIST_TABLE(TIME_RANGE_COLUMN,COLUMN2,....
COLUMNN) VALUES (TIME_RANGE_TYPE (LOWER (OLD.TIME_RANGE),
LOWER(NEW.TIME_RANGE)),OLD.COLUMN2,.... OLD.COLUMNN)
SET OLD.VALUES IN FORE_TABLE TO NEW.VALUES
SUCCESSFULL UPDATE IS RETURNED
ELSEIF(LOWER (NEW.TIME_RANGE) < LOWER (OLD.TIME_RANGE)) THEN
RETURN INPUT ERROR
ENDIF
Example:
UPDATE EARNINGS_FORE SET SALARY=100499, TIME_RANGE=TSRANGE
('2021/01/01
00:00:00', '2022/01/01 23:59:00', '[]') WHERE EMP_NO= 3067
```

IV. RESULTS & DISCUSSION

The employee database dataset (90,000 records) was used to run several queries on the proposed model (TTHHR) and the previous conventional tuple time stamped single relational model (TTSR). Unlike the TTHHR model, which keeps prior records in separate history relations and the current snapshot in "fore" tables, the TTSR model keeps both the current and past occurrences of data in a single current and history model. The results obtained revealed a significant reduction in memory space requirements and query execution time. As a result, the net cost is also lower.

(i) QUERY EXECUTION TIME

The term "query execution time" refers to how long it would take to execute the optimal query execution strategy [30]. A set of queries is created and executed in order to compare the results of the TTSR and TTHHR models. After seven iterations, the mean execution time for each query is used to compare results. The queries used to evaluate the performance of the TTSR and TTHHR models are listed in Table I. The mean time is recorded using the Postgres PgAdmin utility. PgAdmin provides an estimated optimal query execution plan for tracking query execution time.

Only history tables are utilized to run Q1 and Q4. Queries Q2, Q10, and Q11 can only be run with the current tables. For rest all queries Q3, Q5, Q6, Q7, Q8, and Q9, current and historical tables are exercised. Table II displays the mean execution times in milliseconds of both models TTSR and TTHHR for the queries listed in Table I. The proposed tuple time stamped hybrid historical relation (TTHHR) model has a faster execution time for the majority of queries. The accelerate ratio is calculated as the ratio of the mean query execution times for the two models.

Accelerate Ratio (A_R) =
Mean Query Execution Time (μTTSR)/ Mean Query Execution Time (μTTHHR)

In this formula, A_R stands for accelerate ratio, μTTSR is mean query execution time for tuple timestamp single relation model, and μTTHHR is mean execution time for proposed model mean query execution time. The performance in terms of query execution time of the proposed model is compared to single relational data model and results obtained shows that proposed (TTHHR) model is approximately 1.2 times faster than the conventional single relational (TTSR) data model.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

Hence, approximately 20% query execution time efficiency is observed. Figure 4 represents comparison of mean query execution time between single relational (TTSR) and proposed model (TTHHR).The results clearly show that the proposed model's query execution time is faster when the query only returns results from the current or history table. Whereas, the proposed model outperforms the single relation model for queries that require data values from both the present and history tables.

Table-I Queries used for Performance Evaluation of different Models

Query No.	Query Description
Q1	Select all attributes from the earnings table for employee 3067 where time period is in range ('2021/04/01/00:00:00','2022/01/31 23:59:00');
Q2	Select all attributes from the earnings table for employee 3067 where time period is in range ('2021/04/01/00:00:00','9999/01/01 23:59:00');
Q3	Select all attributes from the earnings table for employee 3067 where time period is in range ('2017/04/01/00:00:00','9999/01/01 23:59:00');
Q4	Select attributes from tables remuneration, child_allowance and arrear tables for employee 3067 where time period is in range ('2021/04/01/00:00:00','2022/01/31 23:59:00') for the period of one year.
Q5	Select attributes from tables earnings, remuneration, child_allowance and arrear for employee 3067 using the earnings.time_range, remuneration.time_range, child_allowance.time_range and arrear.time_range where time period is in range ('2021/04/01 00:00:00','9999/01/01 23:59:00') to find out total earnings during the period.
Q6	Select attributes from tables earnings, remuneration, child_allowance and arrear for employee 3067 using earnings.time_range, remuneration.time_range, child_allowance.time_range and arrear.time_range where time period is in range ('2017/04/01 00:00:00','9999/01/01 23:59:00') to find out total earnings during the period.
Q7	Select all attributes from earnings table for employee 125 where date is '2022/04/01 00:00:00'::timestamp;
Q8	Select all attributes from earnings table for employee number 2125 where date is '2017/04/01 00:00:00'::timestamp;
Q9	Select attributes from tables earnings, remuneration, child_allowance and arrear for employee 2125 using earnings.time_range, remuneration.time_range, child_allowance.time_range and arrear.time_range time period is '2017/04/01 00:00:00' :: timestamp to find out total earnings during the period.
Q10	Select attributes from tables earnings, remuneration, child_allowance and arrear for employee 2125 using earnings.time_range, remuneration.time_range, child_allowance.time_range and arrear.time_range time period is '2022/04/01 00:00:00' :: timestamp to find out total earnings during the period.

Table II: Mean Query Execution time and Accelerate ratio

Q No	Mean Query execution Time in Single Relation Mode µTTSR (milliseconds)	Mean Query execution Time in Proposed Model µTTHHR (milliseconds)	Accelerate Ratio = (µTTSR/ µTTHHR)
Q1	0.587	0.467	1.256959315
Q2	0.486	0.398	1.221105528
Q3	0.553	0.627	0.881977671
Q4	0.918	0.794	1.156171285
Q5	0.935	0.813	1.150061501
Q6	0.994	0.788	1.26142132
Q7	0.815	0.756	1.078042328

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

Q8	0.885	0.796	1.111809045
Q9	0.983	0.812	1.210591133
Q10	0.978	0.829	1.17973462

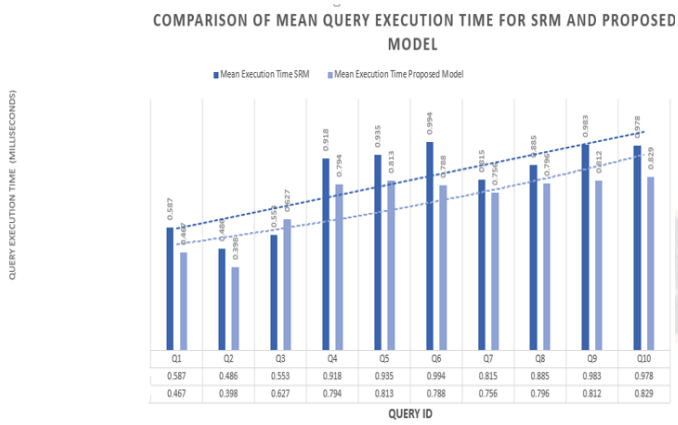

Figure 4 Comparison of Mean Query Execution Time by Single Relational and Proposed Model

(ii) MEMORY COST

The phrase "memory cost" refers to amount of memory storage required to store data. A number of queries are designed and run in order to compare the results of the TTSR and TTHHR models for required memory storage. Table- I lists the queries used to evaluate the memory cost of the TTSR and TTHHR models at different frequencies.

Total memory required for inserting 'f' (frequency) tuples because of update operation for tuple timestamp single relation (TTSR) will be given by

Memory Cost(R) = (Memory Cost (Key attributes) + Memory Cost (Static attributes) + Memory Cost (Dynamic attributes) + Memory Cost (Timestamps)) frequency*

$= (K+S+D+T)*f$

Total memory required for inserting 'f' tuples because of update operation for tuple timestamp hybrid history (TTHHR) relation will be given by

Memory Cost(R) = (Memory Cost (Key attributes) + Memory Cost (Dynamic attributes) + Memory Cost (Timestamps)) frequency*

$= (K+D+T)*f$

There will be two cases for calculating the memory cost, if the dynamic attributes are homogeneous in nature then, the cost relates to only homogeneous dynamic attributes(say *m* attributes out of total n attributes) and if the changes comes to heterogeneous dynamic attributes, then it will affect only the *(n-m)* attributes and cost accordingly.

Table III: Memory cost (M_C)

Q No	Frequency	M_C in (TTSR Bytes)	M_C of update for varying frequency in TTSR (Bytes)	M_C in proposed Model TTHHR (Bytes)	M_C of update for varying frequency in Proposed Data Model TTHHR (Bytes)
Q1	5	26	130	14	70
Q2	10	26	260	14	140
Q3	15	26	390	14	210
Q4	20	26	520	10	200
Q5	25	26	650	20	500
Q6	30	26	780	20	600
Q7	35	26	910	10	350
Q8	40	26	1040	10	400
Q9	45	26	1170	20	900
Q10	50	26	1300	20	1000

The comparative analysis of the proposed data model with respect to single relation data model in terms of memory cost:

*Memory Cost of TTSR with frequency (f) = (K+S+D+T) * f*

*Memory Cost of TTHHR with frequency (f) = (K +Di +T) * f;*

***Either** Case-I: when i= "m" for "m" homogeneous attributes*

OR

Case-II: when i= "n-m" for "n-m" heterogeneous attributes

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

Improvement in memory cost in TTHHR data model with respect to TTSR data model is given by equation

= *(Memory Cost (TTSR) – Memory Cost (TTHHR))/Memory Cost (TTSR).*

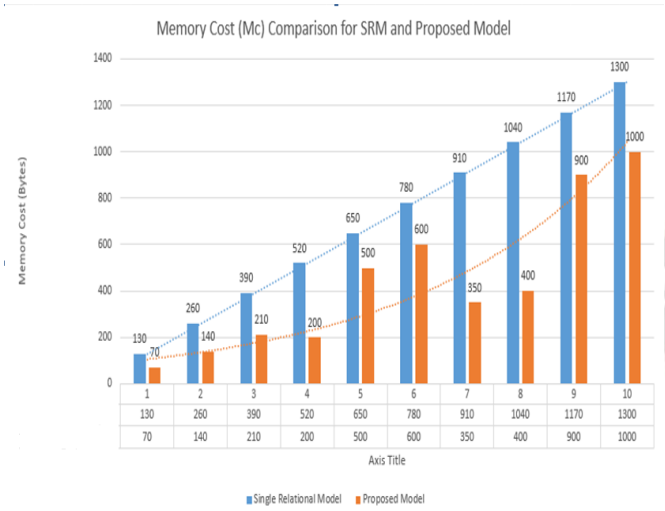

Figure 6 Memory Cost Comparison of TTSR and Proposed TTHHR Model

Figure 6 shows the Memory Cost Comparison of TTSR and Proposed TTHHR model, which clearly reveals lower memory requirements for the proposed model as compare to conventional single relational data model. Results obtained shows that proposed (TTHHR) model saves approximately 30% memory as compared to TTSR (Tuple Timestamp Single Relational Model).

Table IV Reduction in Net Cost of Proposed TTHHR w.r.t. TTSR

Query ID	Frequency	Mean Query Execution time TTSR	Memory Cost of TTSR	Net Cost of TTSR	Mean Query Execution time Proposed Model TTHHR	Memory Cost of Proposed Model TTHHR	Net Cost of Model TTHHR	Reduction in Net Cost of Proposed model TTHHR w.r.t. TTSR (%save)
Q1	5	0.587	26	76.31	0.467	14	32.69	57.161578
Q2	10	0.486	26	126.36	0.398	14	55.72	55.903767
Q3	15	0.553	26	215.67	0.627	14	131.67	38.948393
Q4	20	0.918	26	477.36	0.794	10	158.8	66.733702
Q5	25	0.935	26	607.75	0.813	20	406.5	33.113945
Q6	30	0.994	26	775.32	0.788	20	472.8	39.018728
Q7	35	0.815	26	741.65	0.756	10	264.6	64.322794
Q8	40	0.885	26	920.4	0.796	10	318.4	65.406345
Q9	45	0.983	26	1150.11	0.812	20	730.8	36.458252
Q10	50	0.978	26	1271.4	0.829	20	829	34.796288

(iii) NET COST

Table IV represents the Net Cost of both models. Net cost can be calculated using formula as

*Net Cost = (Mean Query Execution Time * Memory Cost M_C)*

The Net cost of TTSR and TTHHR estimated at various frequencies is shown in the table. The net cost of the TTSR model is derived by multiplying the mean query execution time at different frequencies by the memory cost. The proposed model's net cost is calculated using the same methodology. The reduction in net cost and percentage save is calculated using the formula as -

*= (Net cost (TTSR) - Net Cost (TTHHR)) /Net Cost (TTHHR) *100*

The findings indicate that the total net cost has been significantly reduced. The proposed tuple timestamp hybrid historical model acts as generic and robust model which offers the same functionality as older versions while significantly improving query execution speed and using less memory. The

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 3
DOI: https://doi.org/10.17762/ijritcc.v11i3.6193
***Article Received**: 18 December 2022 **Revised**: 26 January 2023 **Accepted**: 15 February 2023*

TTHHR behaves like the tuple timestamp history relation (TTHR) model if all of the dynamic attributes of the relation are temporally homogeneous and that too with more efficiency. If the attributes of the relation are temporally heterogeneous, the model behaves like a tuple timestamp multiple historical relational (TTMHR) model. Evidently, the filter layer separates the homogeneous and heterogeneous temporal features. When heterogeneous and homogeneous attributes are blended together in database, both types of attributes are handled through hybridization. TTHHR reduces the amount of memory needed effectively and improves query performance.

V. CONCLUSION

The tuple time-stamped hybrid historical relation (TTHHR) is introduced as an optimal data model capable of organizing temporal data efficiently. The proposed model is built on a tuple time-stamping method with valid time as the time dimension. All temporal database operations, such as record insertion, updating, and retrieval, are executed efficiently. To avoid data duplication, all static attributes are compiled into a single, unique relation. Separate temporal relations are established for all temporal attributes with varying valid times; for attributes with equal valid times, a single relation is built. temporal relations are established; for attributes with equal valid times, a single relation is built. This process enables the database to be cleared of superfluous redundancy. The query execution time and memory cost parameters are used to evaluate the performance of the proposed tuple time-stamped hybrid historical relation (TTHHR) data model. The proposed model's performance in terms of query execution time is compared to a single relational, and the results show that the proposed model is approx. 20% faster than the conventional single relational model . Memory consumption results show that memory cost at different frequencies for the proposed model is reduced significantly, which is approximately 30% lower when compared to the conventional single relational data model.

As a result, the suggested model's net cost is significantly reduced. The tuple timestamp hybrid historical model proposed here serves as a generic and robust model. It provides the same functionality as previous versions, as well as hybrid functionality of previously proposed models, with a significant improvement in query execution speed and memory usage. In the future, the model could be applied to a wide range of temporal database fields, including insurance, geographic information systems, stocks and finance, data warehousing, scientific databases, legal case histories, and medical records.

REFERENCES

[1] Bohlen, Michael H., Renato Busatto, and Christian S. Jensen. "Point-versus interval-based temporal data models." In Proceedings 14th international conference on data engineering, pp. 192-200. IEEE, 1998. doi: 10.1109/ICDE.1998.655777.

[2] Sevilla-Lara, L., Zha S., Yan Z., Goswami V., Feiszli M., Torresani L. “Only Time Can Tell: Discovering Temporal Data for Temporal Modeling” Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021, pp. 535-544

[3] Snodgrass, Richard T. "A Case Study of Temporal Data." Teradata Corporation, 2010.

[4] Kumar, Shailender, Rahul Rishi, and Rupender Duggal. "Implementation of temporal functionality in objects with role model (TF-ORM)." In 2016 1st India International Conference on Information Processing (IICIP), pp. 1-5. IEEE, 2016. 10.1109/IICIP.2016.7975368

[5] R. D. M. Galante, C. S. D. Santos, N. Edelweiss, A. F. Moreira, "Temporal and versioning model for schema evolution in object oriented databases," Data & Knowledge Engineering, pp 99-128, May, 2005. doi:10.1016/j.datak.2004.07.001

[6] Burney, Aqil, Nadeem Mahmood, and Kamran Ahsan. "Tempr-pdm: a conceptual temporal relational model for managing patient data." In Proc. Int. WSEAS conference AIKED, pp. 237-243. 2010.

[7] Yang, C., Wang, X., Zhang, M., Zheng, R., Wei, W., & Lou, Y., "Standardization on bitemporal information representation in BCDM," IEEE International Conference on Information and Automation, 2015, pp. 2052-2057. doi: 10.1109/ICInfA.2015.7279627

[8] Gandhi, Lalit. "Literature survey of temporal data models." International Journal of Latest Trends in Engineering and Technology 8, no. 4-1 (2017): 294-300.

[9] Kunzner, F., & Petkovic, D. “A Comparison of Different Forms of Temporal Data Management”. Springer International Publishing International Conference: Beyond Databases, Architectures and Structures, pp 92-106, 2015. doi:10.1007/978-3-319-18422-7-_8.

[10] Zemke, Fred. "What’s new in SQL: 2011," ACM SIGMOD Record 41.1, pp. 67-73, 2012.

[11] http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata

[12] Jensen, Christian S., Michael D. Soo, and Richard T. Snodgrass. "Unifying temporal data models via a conceptual model." Information Systems 19, no. 7, pp. 513-547, 1994. doi: 10.1016/0306-4379(94)90013-2.

[13] Gadia, Shashi K., and Chuen-Sing Yeung. "A generalized model for a relational temporal database." In ACM SIGMOD Record, vol. 17(3), pp. 251-259, ACM, 1988. doi: 10.1145/971701.50233.

[14] Edelweiss, Nina, Patrícia Nogueira Hubler, Mirella Moura Moro, and Giovani Demartini. "A temporal database management system implemented on top of a conventional database." In Proceedings 20th International Conference of

the Chilean Computer Science Society, pp. 58-67. IEEE, 2000. 10.1109/SCCC.2000.890392

[15] Alromema, Nashwan. "Retrieval optimization technique for tuple timestamp historical relation temporal data model." Journal of Computer Science 8, no. 2 (2012): 243-250.

[16] S.Kumar, R.Rishi "A New Optimized Model to Handle Temporal Data using Open Source Database", Advances in Electrical and Computer Engineering, Volume 17, Number 2, 2017

[17] C. S. Jensen, R. T. Snodgrass, "Temporal data management," IEEE Transactions on Knowledge and Data Engineering,11(1):36-44, 1999. doi:10.1109/69.755613.

[18] R. Elmasri and S.Navathe, "Fundamentals of Database Systems", Benjamin/Cummings 2012

[19] Kvet, Michal, Karol Matiaško, and Monika Vajsová. "Sensor based transaction temporal database architecture." In 2015 IEEE World Conference on Factory Communication Systems (WFCS), pp. 1-8. IEEE, 2015. DOI: 10.1109/WFCS.2015.7160547

[20] Ali, Noraida Haji, and Sumazly Sulaiman. "Managing News Archive Using Temporal Data Modeling." Journal of Applied Sciences 12, no. 3 (2012): 284-288.

[21] Krause, Josua, Adam Perer, and Harry Stavropoulos. "Supporting iterative cohort construction with visual temporal queries." IEEE Transactions on visualization and computer graphics 22(1), pp. 91-100, 2016. doi: 10.1109/TVCG.2015.2467622.

[22] Kumar, Shailender, and Rahul Rishi. "Retrieval of meteorological data using temporal data modeling." Indian Journal of Science and Technology 9, no. 37 (2016). DOI: 10.17485/ijst/2016/v9i37/99875.

[23] Christy, A., and G. Meera Gandhi. "Combining bitemporal conceptual data model with multiway join relations for forecasting." Procedia Computer Science 57 (2015): 1104-1114. https://doi.org/10.1016/j.procs.2015.07.396

[24] Wang W., Peng X., Qiao Y., Cheng J. "An empirical study on temporal modeling for online action detection" Complex & Intelligent Systems (2022) 8:1803–1817

[25] Petkovic, Dusan. "Temporal Data in Relational Database Systems: A Comparison." New Advances in Information Systems and Technologies. Springer International Publishing, pp. 13-23, 2016. doi: 10.1007/978-3-319-31232-3_2.

[26] Atay, Canan. "A comparison of attribute and tuple time stamped bitemporal relational data models." (2010). Int Conf on Applied Computer Science. 2010: 479-89.

[27] Anselma, L., Terenziani, P., & Snodgrass, R. T. "Valid-time indeterminacy in temporal relational databases: Semantics and representations". IEEE Transactions on Knowledge and Data Engineering, 25(12), pp. 2880-2894, 2013. doi: 10.1109/TKDE.2012.199.

[28] Terenziani, Paolo. "Irregular indeterminate repeated facts in temporal relational databases." IEEE Transactions on Knowledge and Data Engineering 28, no. 4 (2015): 1075-1079. doi: 10.1109/TKDE.2015.2509976.

[29] AL-romema, Nashwan. "Memory storage issues of temporal database applications on relational database management systems." Journal of Computer Science 6, no. 3 (2010).

[30] Murugan, K., and T. Ravichandran. "Intelligent query processing in temporal database using efficient context free grammar." Indian Journal of Science and Technology 5, no. 6 (2012): 1-6.

[31] Congdon P., "A spatio-temporal autoregressive model for monitoring and predicting COVID infection rates" Journal of Geographical Systems (2022) https://doi.org/10.1007/s10109-021-00366-2

[32] Gandhi, L., Rishi R., Sharma S., Wawale S.G., Chweya R. "Air quality data acquisition through temporal data modeling" Hindawi Mathematical Problems in Engineering.