Turkish Journal of Electrical Engineering and Computer Sciences

Volume 24 | Number 5 Article 73

1-1-2016

An attribute or tuple timestamping in bitemporal relational databases

CANAN ATAY

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4305 – 4321

doi:10.3906/elk-1403-39

An attribute or tuple timestamping in bitemporal relational databases

Canan ATAY*
Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey

Received: 05.03.2014 • **Accepted/Published Online:** 04.08.2015 • **Final Version:** 20.06.2016

Abstract: Much of the research on bitemporal databases has focused on the modeling of time-related data with either attribute or tuple timestamping. While the attribute-timestamping approach attaches bitemporal data to attributes, the tuple-timestamping approach splits the object's history into several tuples. Although there have been numerous studies on bitemporal data models, there is no work contrasting these two common approaches in terms of system performance and ease of use. In this paper, we compared interval-based attribute- and tuple-timestamped bitemporal data models by running sample queries to measure processing time, and then we evaluated their usability by using the same data. Our tests indicate that the attribute-timestamping model with one-level nested approach required less time and used less disk space; therefore, it is more appropriate for modeling bitemporal data.

Key words: Bitemporal database, attribute timestamping, bitemporal query, model comparison, object relational database systems

1. Introduction

Temporal database researchers have mostly focused on the description of data models to support temporal implementation strategies, efficient temporal query evaluation, temporal data presentation, and data storage. Although a temporal database may store the history of objects, bitemporal database systems support both the history of objects (valid time) and database activity (transaction time). A temporal database system that captures only the history of an object does not save the retroactive and postactive changes. The systems that store only database activities are not capable of preserving the validity period of data values or future data. However, bitemporal databases provide a complete history of both data values and of when those values were changed, by associating data values with facts and by specifying when those facts were valid. In order to model the real world accurately and completely, both time dimensions need to be considered. Auditing is exceptionally important for certain application areas such as insurance, tax, and finance, which can profit from the support of transactions and valid times.

The proposed temporal relational data models are categorized according to the timestamps attached and the time dimension they support. The data that have time-related information are known as temporal data. A timestamp is a time value associated with a data value that might be a temporal element, time interval, or time point. Timestamps may be attached to attributes (attribute-value timestamping) or tuples (tuple timestamping). Whereas the first approach requires nonfirst normal-form (N1NF) relations, the latter uses first normal-form (1NF) relations.

Temporal database researchers have worked extensively for more than three decades; nevertheless, there

*Correspondence: canan@cs.deu.edu.tr

is no standardized temporal database model by any organization. To the best of our knowledge, in spite of extensive research on bitemporal data models, such as [1–6], no work has been conducted that compares these bitemporal data models in terms of system performance and ease of use. In this paper, we will compare and contrast interval-based attribute- and tuple-timestamped bitemporal data models by running sample queries to measure the processing time and we will evaluate their usability using the same data for both approaches.

In the evaluation process, we consider the nested bitemporal relational data model (NBRDM) [6], with one and two levels of nesting, and Snodgrass's tuple-timestamping model [2]. We compare them with a set of queries to study the performance of the three models. This paper intends to facilitate the implementation and evaluation of bitemporal database models by presenting queries with a time component. The major factors that have the most significant impact on system performance will be identified, and the results of the queries will be analyzed.

The following tasks were conducted in order to resolve which approach is more appropriate for bitemporal databases: 1) implementing three approaches in an object-relational database with the same bitemporal data; 2) introducing six plain English queries and writing them for three approaches; and 3) measuring system performances for all three models. We note that the first approach is more appropriate for bitemporal databases based on the following observations. Attribute timestamping with two levels of nested structure takes longer for queries; nevertheless, one level of nested structure outperforms the tuple-timestamped method. Besides, it is important to note that the nested structure is easier to follow, and therefore it can be preferred as a temporal solution in real-life projects. Moreover, an attribute-timestamping method requires much less space than the tuple-timestamping method.

The paper is organized as follows. Section 2 briefly reviews related work on bitemporal databases. Section 3 introduces interval-based attribute- and tuple-timestamped bitemporal relational data models. Section 4 presents an overview of the system implementations. Section 5 compares three approaches based on the same queries and discusses the system performances of the data models. Section 6 concludes the paper and provides detailed plans for future research.

2. Related work

Temporal database research has been undertaken extensively since the early 1980s. During this time, numerous temporal extensions to the relational data model and query languages have been proposed. There are two common approaches for extending the relational data model: tuple timestamping and attribute timestamping. The distinction between the two lies in where the timestamps are attached. Whereas tuple timestamping uses the 1NF relations [1,2,4,5], attribute timestamping requires N1NF relations [3,6–13].

There are three different timestamping representation methods: time point, time interval, and temporal element. The time point references the time domain as a discrete, infinite, countable, linearly ordered set without end points [14]. Clifford made the discreteness assumption in [12]. The time-interval schema represents a domain as the continuous maximum time interval. TSQL2 [15] and IXQL [16] introduced interval-based temporal data models. The temporal element is the finite unions of time intervals; examples of temporal element-based temporal data models are ParaSQL [17] and nested relational tuple calculus [18].

A taxonomy for classifying databases in terms of valid time and transaction time was developed in [19]. According to this taxonomy, [8,12,13,20,21] are valid time (historical), [3] is a transaction time (rollback), and [1,2,5,6,11] are bitemporal databases. Ben-Zvi proposed the first data model for bitemporal databases. He proposed a temporal query language, indexing, synchronization, concurrency, storage architecture, and

its implementation [1]. Snodgrass's temporal model supports transaction and valid times, where tuples are timestamped with either time instants or time intervals [2]. Bhargava and Gadia attached transaction and valid timestamps to attribute values, and relational algebra was defined for their model in [3]. Their proposed model was one of the first to be used as an auditing database system, which also allowed queries and updates to be restructured. The bitemporal conceptual data model (BCDM) forms the basis for the temporal structured query language (TSQL) proposed by Jensen et al. [4]. BCDM is based on a tuple-timestamping approach in which tuples include an implicit attribute value with an ordered pair of integers. Relations in BCDM are in N1NF, since the timestamps associated with the tuples are time units. Therefore, only uniform tuples are supported in BCDM. Atay and Tansel attached bitemporal data to attributes and defined a bitemporal relational algebra as well as a bitemporal relational calculus language for bitemporal data support [6]. They also implemented a string and abstract data-type representation of bitemporal data and developed a preprocessor for translating a bitemporal SQL statement into standard SQL statements in [11]. Most of these implementations use tuple timestamping; however, [20] presented a model that builds valid time support directly into an extensible commercial object-relational database system. Chau and Chittayasothorn's [21] model is another example of the implementation of temporal databases on top of object-relational databases in which attribute timestamping is used. Their model supports four different types of users, which is similar to the concept of context in [6], except it only supports valid time.

T4SQL was proposed based on the tuple timestamping approach as a new query language in [22], which operates in multidimensional temporal relations. It allows one to query temporal relations provided with (a subset of) the temporal dimensions of validity, transaction, availability, and event time, according to di?erent semantics. Although any T4SQL query can be translated into an equivalent SQL query, the corresponding SQL queries are more complex, their size is bigger, and their execution is often quite ine?cient.

XML is also a new database model serving as a powerful tool for approaching semistructured data. The hierarchical structure of XML provides a natural environment for the use of temporally grouped [7] or attribute timestamping approaches. The authors in [23] showed that transaction-time, valid-time, and bitemporal database histories can be represented in XML and queried using XQuery without requiring any extensions of the current standards. The study in [24] presented the ArchIS system, which uses XML to support the attribute time-stamping approach; XQuery to express powerful temporal queries, temporal clustering, and indexing techniques for managing the actual historical data in a RDBMS; and SQL/XML for executing the queries on the XML views as equivalent queries on the relational database. The study in [25] performed a comparison of the various temporal XML data models that occur in the literature.

3. Bitemporal relational data models

In this section, the general concept of attribute- and tuple-timestamped models will be discussed. The NBRDM [6] will be used for the attribute-timestamped approach, whereas Snodgrass's proposed bitemporal model [2] will be used for the tuple-timestamped approach.

3.1. Attribute-timestamped model

N1NF relations are used for attribute-timestamped bitemporal data models; timestamps are attached to the attribute values. Each tuple has the object's entire history, and each attribute stores the history of values. Only values in a tuple that are updated have to be changed; others remain the same. All time-related attributes can be modeled in one relation.

A bitemporal atom is defined as the triplet <transaction time, valid time, value> where time components can be applied as a temporal element, a time interval, or a time point. A bitemporal atom (< [TT_l, TT_u), [VT_l, VT_u), V>) represents:

TT_l: Transaction time lower bound,

TT_u: Transaction time upper bound,

VT_l: Valid time lower bound,

VT_u: Valid time upper bound,

V: Data value.

Because the value of *now* changes as time progresses, the [TT_l, *now*] or [VT_l, *now*] interval is closed and expanding.

An atom's and a bitemporal atom's order are equal to zero. The order of a nested bitemporal relation scheme is one more than the order of its tuple scheme. The inductive definitions of bitemporal tuple and nested bitemporal relation schemes were given in [26].

The nested bitemporal relation schema EMPLOYEE depicted in Table 1 is shown below. DNAME and MANAGER attributes are defined in the DEPARTMENT relation, which makes the EMPLOYEE relation's nesting level two.

Table 1. A nested bitemporal relation, EMPLOYEE. (Note that DEPARTMENT and SALARY-B are columns of EMPLOYEE and displaced on the next line to save space.)

EMP#	ENAME-B	ADDRESS-B	BIRTH DATE
	NAME	ADDRESS	
E_1	<[1, *now*], [1, *now*], Robin Willie>	<[1, *now*], [1, *now*], A_1 >	1980
E_2	<[1, *now*], [1, *now*], Kevin Alan>	<[1, *now*], [1, *now*], A_2 >	1975
E_3	{<[17,32), [20, 32), Amy Irene> <[33, 45), [33, 45), Amy Willie> <[46, 50), [46, 50), Amy Irene> <[58, *now*], [60, *now*], Amy Irene>}	{<[17,32), [20, 32), A_3 >, <[33, 45), [33, 45), A_1 >, <[46, 50), [46, 50), A_3 >, <[58, *now*], [60, *now*], A_3 >}	1990
E_4	<[20, *now*], [23, *now*], Nina Brown>	{<[20,61), [23,61), A_4 >, <[62, *now*], [62, *now*], A_6 >}	1982
E_5	<[15, *now*], [15, *now*], Kyla Hazel>	<[15, *now*], [15, *now*], A_5 >	1985

E_3 has disjoint time intervals, from valid time 50 to 60 and transaction time 50 to 58, when she left the company and rejoined. E_3's return to the company was recorded at transaction time 58 and joined at valid time 60. In addition, 'Kevin Alan' was assigned as employee E_4's manager at transaction time 20, valid from time 23. However, it was noticed at time 30 that actually 'Kyla Hazel' should have been E_4's manager. Bitemporal database models are capable of listing any error correction.

3.2. Tuple-timestamped model

Bitemporal data models with a tuple-timestamping approach stay within 1NF relations. A bitemporal relation has four additional attributes for each time variant attribute: *VT_LB* and *VT_UB* attribute pairs for the valid time, and *TT_LB* and *TT_UB* attribute pairs for the transaction time. For each update a new tuple is inserted into a bitemporal relation. As a result, data redundancy occurs. If there is more than one time-related attribute, each update increases redundancy extensively. A solution might be to decompose each time-related attributes into separate tables. Nonetheless, the model results in many small relations in the tuple-timestamping approach.

Example 1 After converting the EMPLOYEE table in Table 1 into tuple-timestamped relations with valid and transaction times, six relations are obtained. Three are listed in Tables 2–4, and the other three tables are omitted to save space.

Table 2. EMPBIRTH.

DEPARTMENT		SALARY-B
DNAME-B	MANAGER-B	SALARY
DNAME	MANAGER	
{<[1, 13), [1, 15), Marketing>, <[14, *now*], [16, *now*], Planning>}	<[1, *now*], [1, *now*], Kevin Alan>	{<[1, 13), [1, 15), 25K>, <[14, 25), [16, 27), 30K>, <[26,37), [28, 39), 32K>, <[38, *now*], [40, *now*], 40K>}
{<[1, 13), [1, 15), Sales>, <[14, *now*], [16, *now*], Planning>}	<[1, *now*], [1, *now*], Robin Willie>	{<[1, 18), [1, 20), 25K>, <[19, 37), [21, 39), 32K> <[38, *now*], [40, *now*), 40K>}
{<[17, 50), [20, 50), TechSup>, <[58, *now*], [60, *now*], TechSup >}	{<[17, 50), [20, 50), Kyla Hazel>, <[58, *now*], [60, *now*], Kevin Alan>}	{<[17, 26), [20, 30), 20K>, <[26, 50), [31, 50), 22K>, <[58, *now*], [60, *now*], 25K>}
<[20, *now*], [23, *now*], Sales>	{<[20, 30), [23, *now*], Kevin Alan>, <[31, *now*], [23, *now*], Kyla Hazel>}	{<[20, 32), [23, 35), 22K>, <[33, 44), [36, 47), 24K>, <[45, *now*], [48, *now*], 26K>}
<[15, *now*], [15, *now*], Sales>	<[15, *now*], [15, *now*], Robin Willie>	{<[15, 35), [15, 39), 25K>, <[36, 51), [44, 53), 28K>, <[52, *now*], [54, *now*], 30K>}

Table 3. EMP-ADDRESS.

EMP#	BIRTH DATE
E1	1980
E2	1975
E3	1990
E4	1982
E5	1985

Table 4. EMP-MANAGER.

EMP#	ADR.	TT_LB	TT_UB	VT_LB	VT_UB
E1	A1	1	*now*	1	*now*
E2	A2	1	*now*	1	*now*
E3	A3	17	32	20	32
E3	A1	33	45	33	45
E3	A3	46	50	46	50
E3	A3	58	*now*	60	*now*
E4	A4	20	61	23	61
E4	A6	62	*now*	62	*now*
E5	A5	15	*now*	15	*now*

4. Implementation

This section outlines how the attribute- and tuple-timestamped approaches presented in the previous section can be implemented. In order to measure the performance of the aforementioned three bitemporal models, query tests are performed. For each bitemporal model, a hypothetical company database of more than 15 years

of past data is utilized. The first implementation database constitutes the EMPLOYEE relation, whose schema is listed in Table 5 (referred to as EMPLOYEE_2 hereafter). The second implementation database is similar to the EMPLOYEE relation, except the level of nesting is equal to 1 (EMPLOYEE_1), as seen in Table 6. Lastly, the third implementation database is made up of six 1NF tables (EMPLOYEE_T), and their schema is depicted in Table 7.

Table 5. Attribute-timestamped database schema EMPLOYEE_2, two-level nested bitemporal relation.

EMP#	MANAGER	TT_LB	TT_UB	VT_LB	VT_UB
E1	Kevin Alan	1	*Now*	1	*now*
E2	Robin Willie	1	*Now*	1	*now*
E3	Kyla Hazel	17	50	20	50
E3	Kevin Alan	58	*Now*	60	*now*
E4	Kevin Alan	20	*25*	23	*now*
E4	Kyla Hazel	31	*Now*	23	*now*
E5	Robin Willie	15	*Now*	15	*now*

Table 6. Attribute-timestamped database schema EMPLOYEE _1, one-level nested bitemporal relation.

EMP#	ENAME-B	ADDRESS-B	BIRTH DATE	DEPARTMENT		SALARY-B
	NAME	ADDRESS		DNAME-B	MANAGER-B	SALARY
				DNAME	MANAGER	

Table 7. Tuple-timestamped database schema, EMPLOYEE_T.

EMP#	ENAME-B	ADDRESS-B	BIRTH DATE	DEPT-B	MANAGER-B	SALARY-B
	NAME	ADDRESS		DNAME	MANAGER	SALARY

4.1. Attribute-timestamped bitemporal data model NBRDM

4.1.1. Bitemporal atom type

We defined a bitemporal atom in Section 3 as having the form <[TTl, TTu), [VTl, VTu), V>. The bitemporal atom contains the DATE data type for the lower and upper bounds of valid and transaction times. Depending on the application, time intervals might be TIMESTAMP or DATE. The value part may be INTEGER, BIGINT, SMALLINT, CHARACTER, CHARACTER LARGE, CHARACTER VARYING, OBJECT, DECIMAL, NUMERIC, or BOOLEAN. Figure 1 depicts the bitemporal atom as an abstract data type, BTA.

```
CREATE TYPE BTA AS (
        TT_LB DATE,
        TT_UB DATE,
        VT_LB DATE,
        VT_UB DATE,
        VALUE CHARACTER VARYING(50));
```

Figure 1. Representation of a bitemporal atom as an abstract data type.

The object-relational database's type system allows us to define a BTA as a structured abstract data type. Retrieving or removing a component is allowed in the query expressions. Defined BTA can be used, similarly to other built-in types, in SQL statements.

4.1.2. Nested bitemporal relation

A tuple in a nested bitemporal relation is an instance of the structured type on which the table is defined. Having a set of identical abstract data types in a single tuple actually simulates the attribute timestamping approach with a single-attribute table for each object's time related attributes [6]. Although Table 5 summarizes the nested relation schema EMPLOYEE_2, which has a nesting level equal to two, Table 6 depicts the nested relation schema EMPLOYEE_1 with a nesting level of one. The schema trees of EMPLOYEE_1 and EMPLOYEE_2 are shown in Figures 2 and 3, respectively.

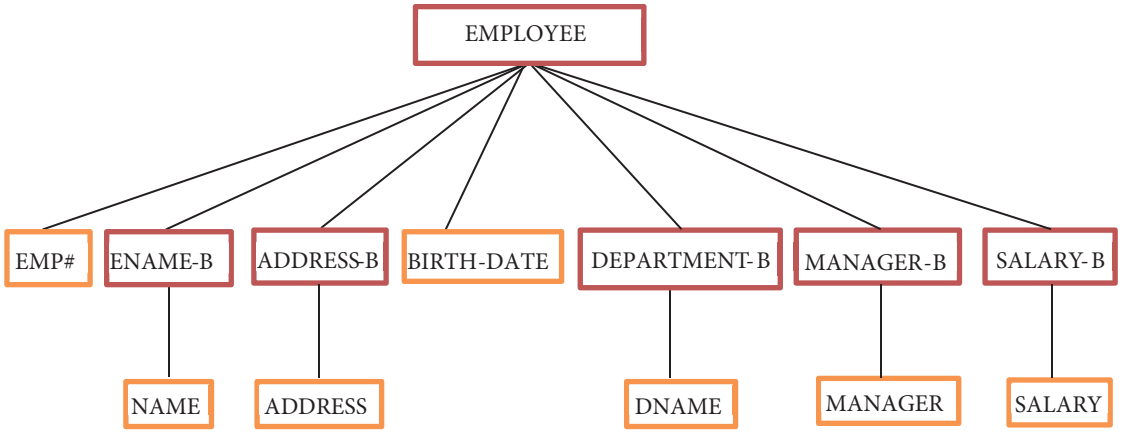

Figure 2. Schema tree for the one-level nested bitemporal relation EMPLOYEE_1.

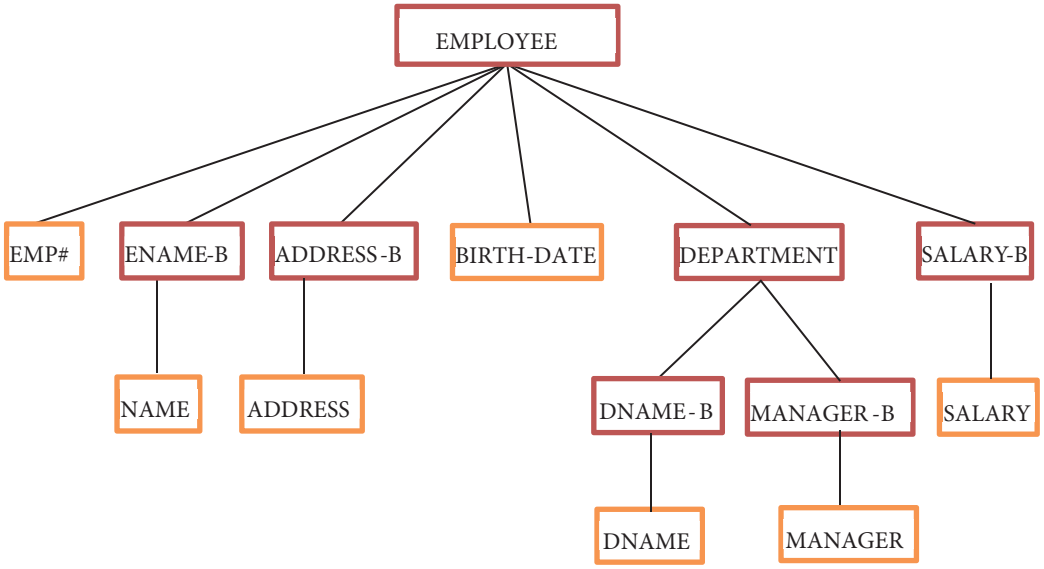

Figure 3. Schema tree for the two-level nested bitemporal relation EMPLOYEE_2.

4.2. Tuple timestamped bitemporal data model

An implementation of a hypothetical company database with the tuple-timestamped approach results in six tables, namely EMP-NAME, EMP-BIRTH, EMP-ADDRESS, EMP-DEPARTMENT, EMP-MANAGER, and EMP-SALARY. Besides EMP-BIRTH, each table has six attributes, which are EMP#, time-related attribute's name, TT_LB, TT_UB, VT_LB, and VT_UB. EMP# is employee id, TT_LB is the start of transaction time,

TT_UB shows the end of transaction time, VT_LB is the start of valid time, and VT_UB is the end of valid time. Because EMP-BIRTH is not time-related, this table has two attributes: employee id and employee birth date in date format. A UML diagram for Snodgrass's model is depicted in Figure 4.

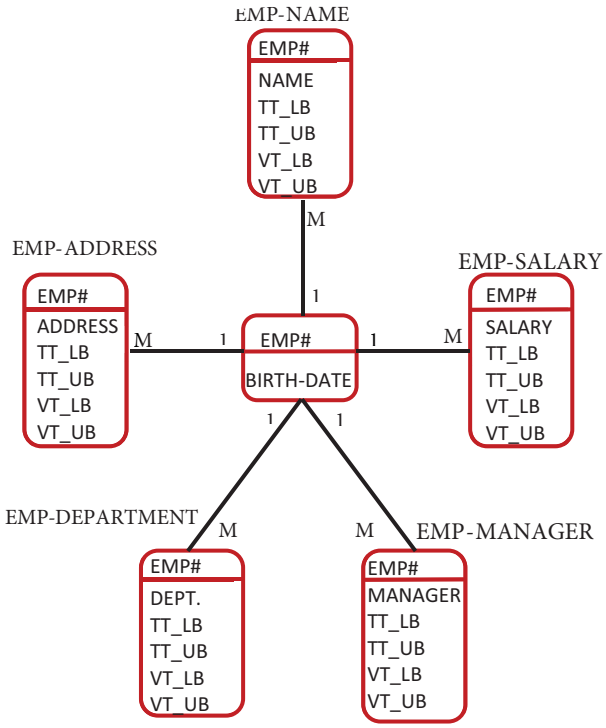

Figure 4. UML diagram for the tuple timestamped database schema EMPLOYEE_T.

5. Query comparisons

Initially, 10,000 initial tuples and 10 years of data are inserted into bitemporal nested tables and six tuple-timestamped 1NF tables. These data reflect typical uses in the existing company's application domain. Generating the initial set of 10,000 data was clearly explained in [11]. Although bitemporal nested tables contain six attributes and 10,000 tuples, the corresponding six tuple-timestamped tables contain a total of 32 attributes and 310,000 tuples. An additional 100,000 and 1,000,000 tuples are inserted into all three tables to compare different approaches, and then the same queries are run.

Queries

This section compares attribute-timestamped (based on EMPLOYEE_2 and EMPLOYEE _1 tables) and tuple-timestamped (based on EMPLOYEE_T schema; EMP-BIRTH, EMP-NAME, EMP-ADDRESS, EMP-MANAGER, EMP-DEPARTMENT, and EMP-SALARY tables) approaches for seven English queries. These queries are written in SQL for Oracle 9i. Each query is run five times and the average value is selected. '12.31.9999' is a special constant for representing the infinite upper limit and/or 'now'. This is common practice in other implementations as well.

Query 1: List employee numbers and names that currently work in Department 22 and earn more than 100K.

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.SALARY) SAL, TABLE(E.DEPARTMENT) DM, TABLE(DM.DNAME) DEP, WHERE DEP.VALUE = 'DEP_ID22' AND SAL.VALUE > 100000 AND NAM.VT_UB ='12.31.9999' AND SAL.VT_UB ='12.31.9999' AND DEP.VT_UB ='12.31.9999'
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.SALARY) SAL TABLE(E.DNAME) DEP WHERE DEP.VALUE = 'DEP_ID22' AND SAL.VALUE > 100000 AND NAM.VT_UB ='12.31.9999' AND SAL.VT_UB ='12.31.9999' AND DEP.VT_UB ='12.31.9999'
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT NAM.EMP#, NAM.NAME AS NAME, FROM EMP-NAME NAM, EMP-DEPARTMENT DEP, EMP-SALARY SAL WHERE DEP.DEPARTMENT ='DEP_ID22' AND SAL.SALARY > 100000 AND DEP.EMP#=SAL.EMP# AND SAL.EMP#= NAM.EMP# AND NAM.VT_UB = '12.31.9999'

Query 2: List the salary values stored in the database between the times 01.01.2008 and 01.01.2012.

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, SAL.VALUE AS SALARY, SAL.TT_LB AS TT_LOWERBOUND, SAL.TT_UB AS TT_UPPERBOUND, SAL.VT_LB AS VT_LOWERBOUND, SAL.VT_UB AS VT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.SALARY) SAL WHERE SAL.VT_LB BETWEEN '01.01.2008' AND '01.01.2012'
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
Same as two-level nested bitemporal relation
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT EMP#, SALARY AS SALARY, SAL.TT_LB AS TT_LOWERBOUND, SAL.TT_UB AS TT_UPPERBOUND, SAL.VT_LB AS VT_LOWERBOUND, SAL.VT_UB AS VT_UPPERBOUND FROM EMPSALARY SAL WHERE VT_LB BETWEEN '01.01.2008' AND '01.01.2012'

Query 3: Obtain all records of the departments in which the employee CANAN EREN ATAY has worked in the database during the date range ['01.01.2010', '12.12.2012'].

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, DEP.VALUE AS DEPARTMENT, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.DEPARTMENT) DEP_MAN, TABLE(DEP_MAN.DNAME) DEP WHERE NAM.VALUE = 'CANAN EREN ATAY' AND '01.01.2010' >= DEP.VT_LB AND '12.12.2012' <= DEP.VT_UB
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, DEP.VALUE AS DEPARTMENT, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.DNAME) DEP, WHERE NAM.VALUE = 'CANAN EREN ATAY' AND '01.01.2010' >= DEP.VT_LB AND '12.12.2012' <= DEP.VT_UB
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT EMP#, EN.ENAME AS NAME, DEP.DNAME AS DEPARTMENT, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND FROM EMP-DEPARTMENT DEP, EMP-NAME EN WHERE EN.ENAME = 'CANAN EREN ATAY' AND DEP.EMP#=E.EMP# AND '01.01.2010' >= DEP.VT_LB AND '12.12.2012' <= DEP.VT_UB

Query 4: As of time '11.11.2011', who was working in the DEP_ID11 department?

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND, DEP.TT_LB AS TT_LOWERBOUND, DEP.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.DEPARTMENT) DEP_MAN, TABLE(DEP_MAN.DEPARTMENT) DEP WHERE DEP.VALUE = 'DEP_ID11' AND '11.11.2011' BETWEEN DEP. TT_LB AND DEP. TT_UB
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
SELECT E.EMP#, NAM.VALUE AS NAME, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND, DEP.TT_LB AS TT_LOWERBOUND, DEP.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.DNAME) DEP WHERE DEP.VALUE = 'DEP_ID11' AND '11.11.2011' BETWEEN DEP.TT_LB AND DEP.TT_UB
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT EMP#, EN.ENAME AS NAME, DEP.VT_LB AS VT_LOWERBOUND, DEP.VT_UB AS VT_UPPERBOUND, DEP.TT_LB AS TT_LOWERBOUND, DEP.TT_UB AS TT_UPPERBOUND FROM EMP-DEPARTMENT DEP, EMP-NAME EN WHERE EN.EMP# = DEP.EMP# AND DEP.DNAME = 'DEP_ID11' AND '11.11.2011' BETWEEN DEP.TT_LB AND DEP.TT_UB

Query 5: Who was CANAN EREN ATAY's manager between the times '03.03.2008' and '06.06.2008', as known to the database system within the time range ['03.01.2008', '06.08.2008']?

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, MAN.VALUE AS MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND, MAN.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.DEPARTMENT) DEP_MAN, TABLE(DEP_MAN.MANAGER) MAN WHERE NAM.VALUE = 'CANAN EREN ATAY' AND '03.03.2008'>= MAN. TT_LB AND '06.08.2008'<= MAN. TT_UB AND '03.01.2008'>= MAN.VT_LB AND '06.08.2008'<= MAN.VT_UB
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
SELECT E.EMP#, MAN.VALUE AS MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND, MAN.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.NAME) NAM, TABLE(E.MANAGER) MAN WHERE NAM.VALUE = 'CANAN EREN ATAY' AND '03.03.2008'>= MAN. TT_LB AND '06.08.2008'<= MAN. TT_UB AND '03.01.2008'>= MAN.VT_LB AND '06.08.2008'<= MAN.VT_UB
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT EN.EMP#, MAN.MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND, MAN.TT_UB AS TT_UPPERBOUND FROM EMP-NAME EN, EMP-MANAGER MAN WHERE EN.NAME = 'CANAN EREN ATAY' AND MAN.EMP#=EN.EMP# AND '03.03.2008'>= MAN.TT_LB AND '06.08.2008'<= MAN.TT_UB AND '03.03.2008'>= MAN.VT_LB AND '06.08.2008'<= MAN.VT_UB

Query 6: What are the EMP# and managers of employees who were employed by the company between '01.01.2012' and '03.03.2012', as known by the database on '06.06.2012'?

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, MAN.VALUE AS MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND, MAN.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.DEPARTMENT) DEP_MAN, TABLE(DEP_MAN.MANAGER) MAN WHERE '06.06.2012' BETWEEN MAN. TT_LB AND MAN. TT_UB AND MAN. VT_LB >='01.01.2012' AND MAN.VT_UB <= '03.03.2012'
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
SELECT E.EMP#, MAN.VALUE AS MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND, MAN.TT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.MANAGER) MAN WHERE '06.06.2012' BETWEEN MAN. TT_LB AND MAN. TT_UB AND MAN.VT_LB >='01.01.2012' AND MAN.VT_UB <= '03.03.2012'
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT MAN.EMP#, MAN.MANAGER AS MANAGER, MAN.VT_LB AS VT_LOWERBOUND, MAN.VT_UB AS VT_UPPERBOUND, MAN.TT_LB AS TT_LOWERBOUND,MAN.TT_UB AS TT_UPPERBOUND FROM EMP-MANAGER MAN WHERE '06.06.2012' BETWEEN MAN. TT_LB AND MAN. TT_UB AND MAN.VT_LB>='01.01.2012' AND MAN.VT_UB<'03.03.2012'

Query 7: What is the employee number of the employees who shared the same addresses at the same time? When was it?

Attribute timestamped table EMPLOYEE_2, two-level nested bitemporal relation.
SELECT E.EMP#, A.ADDRESS.VALUE, E1.EMP#, B.ADDRESS.VALUE, A.VT_LB AS VT_LOWERBOUND, A.VT_UB AS VT_UPPERBOUND, B.VT_LB AS VT_LOWERBOUND, B.VT_UB AS TT_UPPERBOUND FROM EMPLOYEE E, TABLE(E.ADDRESS) A, EMPLOYEE E1, TABLE(E1.ADDRESS) B WHERE A.VALUE = B.VALUE AND E.EMP# > E1.EMP# AND TIME_SLICE(E.EMP#, E1.EMP#, A, B)
Attribute timestamped table EMPLOYEE_1, one-level nested bitemporal relation.
Same as two-level nested bitemporal relation.
Tuple timestamped bitemporal tables EMPLOYEE_T.
SELECT A.EMP#, A.ADDRESS.VALUE, B.EMP#, B.ADDRESS.VALUE, A.VT_LB AS VT_LOWERBOUND, A.VT_UB AS VT_UPPERBOUND, B.VT_LB AS VT_LOWERBOUND, B.VT_UB AS TT_UPPERBOUND FROM EMP-ADDRESS A, EMP-ADDRESS B WHERE A.ADRESS = B. ADRESS AND A.EMP# > B.EMP# AND ((A.VT_LB > B.VT_LB AND A.VT_LB < B.VT_UB) OR (A.VT_UB > B.VT_LB AND A.VT_UB < B.VT_UB) OR (B.VT_UB > A.VT_UB AND B.VT_LB < A.VT_LB)))

According to the taxonomy in [6], the queries can be classified as follows: Query 1 is a current context query, Query 2 is a bitemporal context query with a valid time interval, and Queries 3–6 are historical context queries. Query 3 has a valid time interval, Query 4 a transaction time point, Query 5 valid and transaction time intervals, Query 6 a valid time interval and transaction time point, and, finally, Query 7 is a bitemporal context query with join operation.

Query 1 selects tuples where DEPARTMENT = 'DEP_ID22' and SALARY > 100,000. If the mentioned time-related attributes' valid time upper bounds are equal to '12.31.9999', then it projects the EMP# and NAME attributes. Although all these data are stored in one table in the attribute-timestamping approach, the tuple-timestamping approach requires three tables. Figure 5 shows that the one-level nested approach (EMPLOYEE_1) performs almost the same as the tuple-timestamped approach (EMPLOYEE_T) in terms of run time for Query 1.

Query 2 returns the salary values of employees whose timestamps are between the given valid time ranges, i.e. between 01.01.2008 and 01.01.2012. The one-level nested bitemporal relation, EMPLOYEE_1, returned the selected tuples faster than EMPLOYEE_2 and EMPLOYEE_T, as illustrated in Figure 6.

Query 3 selects the DEPARTMENT attribute of a particular employee with a valid time interval between '01.01.2010' and '12.12.2012'. The DEPARTMENT bitemporal attribute is on two levels of nesting for the EMPLOYEE table. Because decomposing the DEPARTMENT attribute to one-level requires more time for the two-level nested approach (EMPLOYEE_2), and the tuple-timestamped approach (EMPLOYEE_T) joins two tables, the one-level nested approach (EMPLOYEE_1) outperforms the others, as shown in Figure 5.

Query 4 retrieves the state of a table as of the given time point in the past. The DEPARTMENT attributes' value component equal to 'DEP_ID11' is selected. The DEPARTMENT bitemporal attribute is on two levels of nesting for the EMPLOYEE_2 table. Because decomposing DEPARTMENT to one-level requires

more time than the EMPLOYEE_2 table, and the tuple-timestamped approach joins two different tables, the one-level nested approach, EMPLOYEE_1, outperforms the others for Query 4, as depicted in Figure 5.

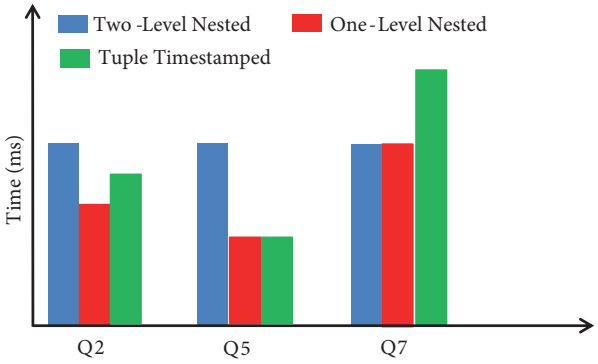

Figure 5. Query 1, Query 3, Query 4, and Query 6 times for bitemporal data models.

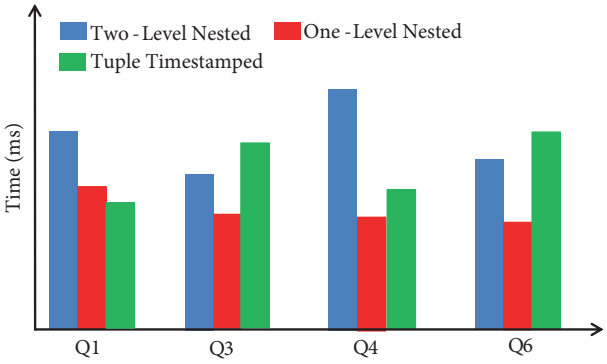

Figure 6. Query 2, Query 5, and Query 7 times for bitemporal data models.

Query 5 is a typical bitemporal query when the valid-time and transaction-time qualifiers are both interval values. The query first decomposes the two-level nested DEP attribute to reach the 'MANAGER' bitemporal attribute, then retrieves the state of a table between ['06.06.2008', '08.08.2008'] for the EMPLOYEE table. This query shows if any error correction was made. Because the MANAGER bitemporal attribute is on two levels of nesting for the EMPLOYEE table, it requires more time than the other two approaches. Tuple-timestamped and one-level nested approaches perform with the same time for Query 5, as shown in Figure 6.

Query 6 retrieves the state of a table as of a time point in the past, namely '06.06.2012' in this example. Because this query deals with one bitemporal tuple for EMPLOYEE_2 and EMPLOYEE_1 tables, and tuple timestamping for the EMPLOYEE_T MANAGER table, all approaches take almost the same time, as illustrated in Figure 5.

Query 7 is a bitemporal context query. Join and slice operations are used. The selection operation picks tuples where the ADDRESS attributes' value components are equal. The time slice operation synchronizes the valid time component of the ADDRESS with respect to the ADDRESS_A valid time component, and hence implements 'when'. The projection operation retains EMP# and ADDRESS bitemporal attributes' value components along with common valid time lower and upper bounds. Because the ADDRESS bitemporal attribute is on the same level for both EMPLOYEE_2 and EMPLOYEE_1 tables, it requires the same time for attribute timestamped approaches. However, the tuple-timestamped approach joins the ADDRESS table by itself. Due to excessive redundancy, as seen in Figure 6, this approach performs weaker.

After these statistics were collected, 100,000 and then 1,000,000 new tuples were inserted into all tables. Figures 7 and 8 illustrate the run times for Query 1 to Query 6 for 100,000 and Figures 9 and 10 for 1,000,000, respectively.

5.1. Query results

Attribute-timestamped nested bitemporal tables, both two-level (EMPLOYEE_2) and one-level (EMPLOYEE_1), acquired 2,097,152 bytes in segments, and the tuple-timestamped table (EMPLOYEE_T) captured 17,039,360 bytes in segments. The latter approach requires 8 times more memory than the other approaches for 10,000 initial data. After inserting an additional 100,000 and 1,000,000 tuples, memory requirements were measured and the following values were retained, as depicted in Table 8. As can be seen, tuple-timestamped tables ac-

quired 15 times more for 100,000 and 30 times more for 1,000,000 memory spaces than the other nested tables, respectively.

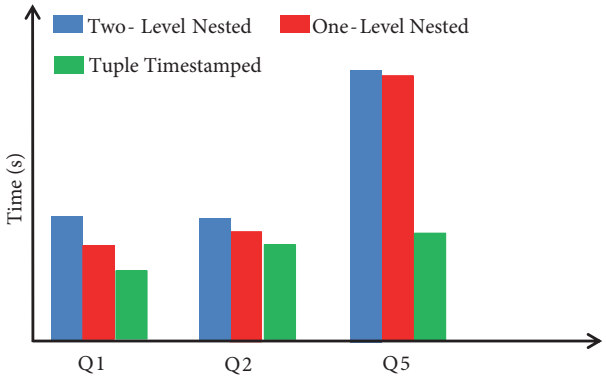

Figure 7. Query 1, Query 2, and Query 5 run times for 100,000 tuples.

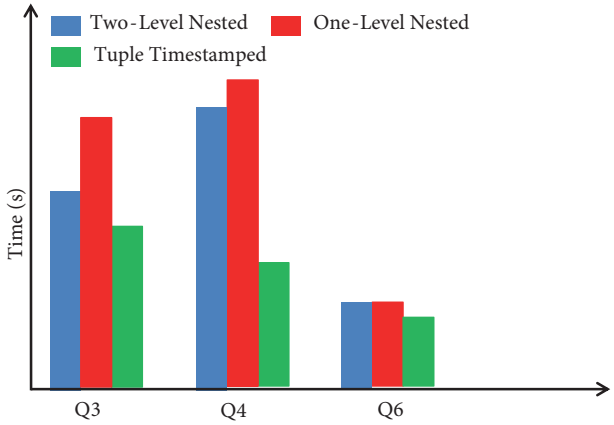

Figure 8. Query 3, Query 4, and Query 6 run times for 100,000 tuples.

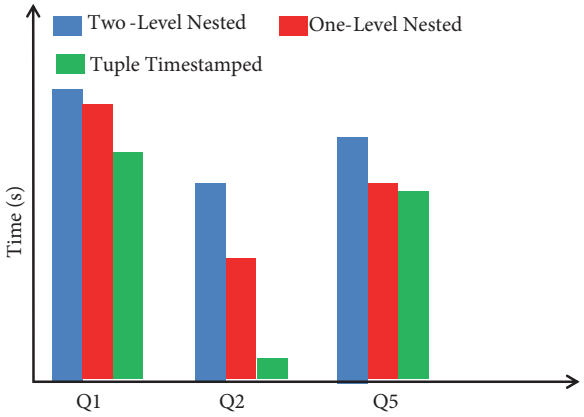

Figure 9. Query 1, Query 2, and Query 5 run times for 1,000,000 tuples.

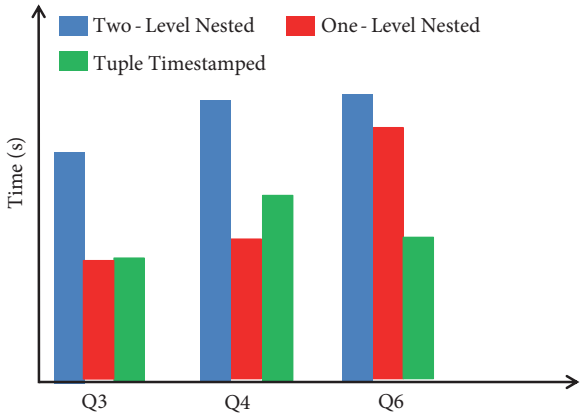

Figure 10. Query 3, Query 4, and Query 6 run times for 1,000,000 tuples.

Statistics are also obtained from Oracle 9i's AUTOTRACE utility and listed in Table A1 in the Appendix. The meaning of these statistical explanations can be found at www.oracle.com. According to these statistics, for the initial 10,000 data in Table A2 in the Appendix, the number of times a buffer was free when visited is lower for Queries 2, 5, and 6 for EMPLOYEE_1, and Queries 1 and 3 for EMPLOYEE_T. This means that the buffer is freer in the attribute-timestamping approach. The total number of bytes received was less for EMPLOYEE_T tables over Oracle Net Services. The total number of bytes sent was less for the EMPLOYEE_2 table from the foreground processes. The number of times a consistent read was requested for a block was almost the same for EMPLOYEE_1 and EMPLOYEE_T tables. Logical reads of database blocks from either the buffer cache or process private memory were higher in the attribute-timestamping approach tables, EMPLOYEE_2 and EMPLOYEE_1. The number of blocks encountered from the buffer cache for scanning was higher in attribute-timestamping approach tables, EMPLOYEE_2 and EMPLOYEE_1. The number of rows that were processed during a scan operation was higher in EMPLOYEE_2. The amount of CPU time used was higher for

EMPLOYEE_2, because decomposing from nesting level two to level one takes more time. The amount of CPU time used by a session from the time a user's call starts until it ends is higher for the attribute-timestamping approach tables, EMPLOYEE_2 and EMPLOYEE_1. Database time was higher for EMPLOYEE_2 for Queries 1, 3, 4, and 5. Queries 2 and 6 took the same time for EMPLOYEE_2 and EMPLOYEE_T. The nonidle wait count was higher for Queries 1 to 3 in EMPLOYEE_2, and for Queries 4 to 6 in EMPLOYEE_T. Again, the attribute-timestamping approach, EMPLOYEE_1, with one-level nesting, outperforms the others. The total number of parse calls is lower with EMPLOYEE_1, whereas it is almost the same with EMPLOYEE_2 and EMPLOYEE_T. The number of user calls is the same for all three approaches. Among all three approaches, work area execution is always less with EMPLOYEE_1. Statistics for 100,000 and 1,000,000 are listed in Tables A3 and A4, respectively, in the Appendix.

Table 8. Memory space requirements for all tables.

EMP-NAME

EMP#	NAME	TT_LB	TT_UB	VT_LB	VT_UB

EMP-ADDRESS

EMP#	ADDRESS	TT_LB	TT_UB	VT_LB	VT_UB

EMP-BIRTH

EMP#	BIRTH_DATE

EMP-DEPARTMENT

EMP#	DEPT.	TT_LB	TT_UB	VT_LB	VT_UB

EMP-MANAGER

EMP#	MANAGER	TT_LB	TT_UB	VT_LB	VT_UB

EMP-SALARY

EMP#	SALARY	TT_LB	TT_UB	VT_LB	VT_UB

Table 9. Oracle 9i AUTOTRACE utility.

Approach type	Bytes in segments for 10,000 objects	Bytes in segments for 100,000 objects	Bytes in segments for 1,000,000 objects
Attribute timestamped, two-level nested, EMPLOYEE_2	2,097,152	10,485,760	49,283,072
Attribute timestamped, one-level nested, EMPLOYEE_1	2,097,152	12,582,912	58,720,256
Tuple timestamped, EMPLOYEE_T	17,039,360	158,334,952	1,471,296,869

On the basis of these tests, we have come to the following conclusions. Bitemporal/temporal queries have the same complexity for both attribute- and tuple-timestamped approaches. If the query must join tables (as it must in most cases), the one-level nested attribute-timestamped approach outperforms the tuple-timestamped approach. In addition, when the priority is to keep all data together rather than split them into many tables and use memory sparingly, the one-level nested attribute-timestamped approach should be preferred. Nevertheless, if the number of objects is large and a fast response time is the highest priority, then the tuple-timestamped approach might be chosen. Finally, the attribute-timestamped approach, regardless of whether it is two-level or one-level nested, requires less memory than the tuple-timestamped approach.

6. Conclusion

In this paper, it has been shown that attribute- and tuple-timestamped interval-based bitemporal databases can be implemented in an object-relational database. Both models have been tested to determine which performs better in terms of time and space. In order to demonstrate the implementation of the aforementioned models in this paper, we performed experiments to observe the performance of the models using the same data. The experiment was intended to compare the performance of the two different bitemporal schemas. For instance, if the designer is interested in determining the most cost-effective approach for a given table size, then this study can use the approach that is superior for certain table sizes.

Comparison items included query performance and space requirement. In order to gain insight into each implementation's relative performance, different tasks were evaluated. Each approach has its own advantages and disadvantages. Our test conclusion indicated that the attribute timestamped with more than one level of the nesting method might require slightly more time, because the nested bitemporal relational database creates a separate table for each time-related attribute. In addition, decomposing the nested level requires time. However, the attribute timestamped with the one-level nested approach required less time than the tuple-timestamped approach when the query joined more than one tuple-timestamped table. The attribute-timestamped method used less disk space because the tuple-timestamping method splits the object's history into several tables. If every time-related attribute is separated into different tables and queries need to join these tables, then the tuple-timestamping run time is higher. On the other hand, tuple timestamping results in a significant amount of redundancy if every time-related attribute is in the same table.

Database management query languages and systems will eventually include bitemporal data management as an integral part. Some commercial DBMS software products have included limited bitemporal support. We believe that this work will influence the design and implementation decisions of bitemporal relational databases.

Future work includes the study of generating benchmark data and queries for bitemporal databases.

Acknowledgment

I would like to express gratitude to the anonymous reviewers for their valuable comments, which improved the manuscript significantly.

References

[1] Gadia S. Ben-Zvi's pioneering work in relational temporal databases. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R, editors. Temporal Databases. Redwood City, CA, USA: Benjamin Cummings, 1993. pp. 202-207.

[2] Snodgrass R. An overview of Tquel. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R, editors. Temporal Databases. Redwood City, CA, USA: Benjamin Cummings, pp. 141-182.

[3] Bhargava G, Gadia S. Relational database systems with zero information loss. IEEE T Knowl Data En 1993; 5: 1-20.

[4] Jensen CS, Soo MD, Snodgrass R. Unifying temporal data models via a conceptual model. Inf Syst 1994; 19: 513-547.

[5] Shasha D, Zhu Y. SpyTime: a performance benchmark for bitemporal database. Available online at http://www.cs.nyu.edu/shasha/spytime/spytime.html. 2001.

[6] Atay CE, Tansel A. Bitemporal Databases: Modeling and Implementation. Saarbrücken, Germany: VDM Verlag, 2009.

[7] Clifford J, Croker A. The historical relational data model (HRDM) and algebra based on lifespans. In: IEEE 1987 International Conference on Data Engineering; 3–5 February 1987; Los Angeles, CA, USA. New York, NY, USA: IEEE. pp. 1-26.

[8] Gadia S. A homogeneous relational model and query languages for temporal databases. ACM T Database Syst 1988; 13: 418-448.

[9] Gadia S, Bhargava G. A Formal Treatment of Updates and Errors in a Relational Database. Ames, IA, USA: Iowa State University, 1997.

[10] Tansel AU. Temporal relational data model. IEEE T Knowl Data En 1997; 9: 464-479.

[11] Atay CE, Tansel AU. BtSQL: nested bitemporal relational database query language. Turk J Elec Eng & Comp Sci 2014; 1: 1-20.

[12] Clifford J, Tansel AU. On an algebra for historical relational databases: two views. In: ACM SIGMOD 1985 International Conference on Management of Data; 15–18 May 1985; Dallas, TX, USA. New York, NY, USA: ACM. pp. 247-265.

[13] Tansel AU. Adding time dimension to relational model and extending relational algebra. Inf Syst 1986; 11: 343-355.

[14] Toman D. Point-based temporal extension of temporal SQL. Lect Notes Comp Sci 1997; 1341: 437-452.

[15] Snodgrass R. The TSQL2 Temporal Query Language. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1995.

[16] Lorentzos NA, Mitsopoulos YG. SQL extension for interval data. IEEE T Knowl Data En 1997; 9: 480-499.

[17] Gadia S, Nair SS. Temporal databases: a prelude to parametric data. In: Tansel AU, Clifford J, Gadia S, Jajodia S, Segev A, Snodgrass R, editors. Temporal Databases. Redwood City, CA, USA: Benjamin Cummings, 1993. pp. 28-66.

[18] Tansel AU, Tin E. The expressive power of temporal relational query languages. IEEE T Knowl Data En 1997; 9: 120-134.

[19] Snodgrass R, Ahn I. Performance evaluation of a temporal database management system. Commun ACM 1986; 15: 96-107.

[20] Yang J, Ying H, Widom J. TIP: A temporal extension to informix. In: ACM SIGMOD 2000 International Conference on Management of Data; 15–18 May 2000; Dallas, TX, USA. New York, NY, USA: ACM. pp. 596-671.

[21] Chau VT, Chittayasothorn S. A temporal compatible object relational database system. In: IEEE 2007 Southeast Conference; 22–25 March 2007; Richmond, VA, USA. New York, NY, USA: IEEE. pp. 93-98.

[22] Combi C, Montanari A, Pozzi G. The T4SQL temporal query language. In: ACM 2007 International Conference on Information and Knowledge Management; 6–10 November 2007; Lisbon, Portugal. New York, NY, USA: ACM. pp. 193-202.

[23] Wang F, Zaniolo C. XBiT: An XML-based bitemporal data model. Lect Notes Comp Sci 2004; 3288: 810-824.

[24] Wang F, Zhou X, Zaniolo C. Using XML to build efficient transaction-time temporal database systems on relational databases. In: IEEE 2006 International Conference on Data Engineering; 3–8 April 2006; Atlanta, GA, USA. New York, NY, USA: IEEE. pp: 131-135.

[25] Ali KA, Pokorny J. A comparison of XML-based temporal models. In: IEEE 2006 International Conference on Signal-Image Technology and Internet-Based Systems; 17–21 December 2006; Hammamet, Tunisia. New York, NY, USA: IEEE. pp. 339-350.

[26] Tansel AU, Atay CE. Nested bitemporal relational algebra. In: International Symposium on Computer and Information Sciences; 1–3 November 2006; İstanbul, Turkey. pp. 622-633.

Appendix

Table A1. Oracle 9i AUTOTRACE utility.

1	Buffer is not pinned count
2	Bytes received via SQL*net from client
3	Bytes sent via SQL*net to client
4	Consistent gets
5	Session logical reads
6	Table scan blocks obtained
7	Table scan rows obtained
8	CPU used by this session
9	CPU used when call started
10	Db time
11	Nonidle wait count
12	Parse count (total)
13	User calls
14	Work area executions

Table A2. Statistics values for 10,000 data.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Q1														
	80	1818	4404	1843	1843	1785	195,047	8	8	9	7	9	5	7
	3193	1714	4742	4504	4503	1142	129,125	6	6	5	3	4	5	2
	66	1684	5009	1277	1277	1253	174,774	6	6	7	4	5	3	2
Q2														
	96	3363	762,276	1408	1417	1262	144,671	8	8	12	223	18	54	7
	3	2190	2,133,727	1309	3932	1303	144,794	11	10	8	298	3	54	2
	116	1998	1,149,644	1022	1022	890	138,189	7	7	12	170	15	54	9
Q3														
	40	1779	4621	622	622	564	51,718	7	7	8	9	5	5	2
	48	1660	4868	327	256	8167	24,978	4	4	6	3	4	4	1
	29	1531	5221	326	326	285	32,037	5	5	6	7	5	5	3
Q4														
	24	1759	15,729	709	571	656	59,887	8	8	9.2	14	7.2	5	5
	521	1634	16,221	963	771	436	39,461	4	4	7	4	4.6	5	2
	74	1517	24,611	441	441	381	41,512	7	7	8	110	9	5	6
Q5														
	159	2130	4106	851	851	626	58,234	8	8	19	53	30	5	9
	31	2008	4155	342	342	297	29,276	5	5	8	46	9	5	2
	115	1759	3927	422	428	313	33,154	6	6	9	75	14	5	6
Q6														
	102	2077	3958	345	256	214	21,334	7	7	8	8	13	5	6
	45	1930	4055	191	154	186	21,331	5	5	6	3	4	5	1
	97	1560	3838	222	657	9831	14,222	6	5	8	19	14	4	8

Table A3. Statistics values for 100,000 data.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Q1														
	111	1818	5079	6873	6873	6006	637,506	20	20	54	27	55	5	30
	163	1767	5022	6115	6111	5194	533,552	15	15	51	25	56	5	30
	121	1656	5053	3518	3518	3362	424,898	9	9	14	34	11	5	5
Q2														
	33	2195	2,089,949	3425	1567	1627	161,211	14	14	18	309	7	33	3
	222	2180	2,090,224	3950	2060	3346	320,184	16	16	237	306	5	53	14
	5	2000	1,149,622	1464	1456	1453	225,888	10	10	12	151	4	54	1
Q3														
	336	1778	4920	5998	3853	2116	209,578	20	20	42	70	43	4	20
	284	1661	4921	4011	4011	1808	215,918	10	10	29	31	35	5	17
	129	1531	5227	2109	2109	1958	206,558	10	10	16	31	12	5	8
Q4														
	526	1758	16,109	3514	3514	2982	319,119	10	10	15	8	7	5	3
	523	1635	16,454	2622	2622	2154	219,061	11	11	14	12	6	5	2
	0	1525	24,591	1968	1968	1958	206,558	7	7	9	3	3	5	2
Q5														
	102	2130	4573	4706	4706	4202	417,487	18	18	111	23	27	5	11
	53	3483	2362	4139	4139	3616	317,407	11	11	98	13	28	5	11
	92	1759	4262	2157	2157	2053	217,539	9	9	13	17	15	5	9
Q6														
	9	2061	4685	1095	1095	1078	111,166	7	7	6	3	6	5	1
	6	1931	4749	1096	1096	1084	111,120	6	6	6	3	5	5	1
	0	1561	4354	1010	1010	1005	111,122	5	5	5	3	3	5	1

Table A4. Statistics values for 1,000,000 data.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Q1														
	30	1613	5320	26,837	26,837	26,750	2,860,274	47	41	110	90	7	4	6
	45	1579	5352	13,775	13,775	9304	1,390,886	28	28	264	732	6	4	2
	50	1503	5941	18,863	18,863	18,787	2,274,684	27	27	101	105	7	4	5
Q2														
	49	1918	1,317,631	12,038	12,038	5977	627,187	28	28	85	266	12	29	3
	27	1917	1,317,806	13,180	13,180	13,082	1,598,957	35	35	102	308	14	52	6
	63	1987	1,149,888	7070	7070	6995	1,095,676	15	14	60	197	10	53	5
Q3														
	38	1599	4152	19,331	19,331	19,243	1,989,410	34	34	106	9	5	4	3
	46	1553	4237	10,416	14,955	14,851	1,997,065	36	36	142	46	4	4	2
	8	1455	4077	4990	4990	4973	496,419	27	6	29	34	2	4	2
Q4														
	1314	1510	16,065	12,781	12,781	10,994	1,131,866	30	30	134	256	27	4	13
	1118	1459	16,631	5807	7690	6046	991,987	13	13	48	1706	25	4	11
	689	1371	24,378	7750	5815	6868	690,137	12	12	53	63	10	4	7
Q5														
	26,691	1849	4482	51,817	51,817	9563	1,034,135	43	43	397	701	15	4	6
	13,334	1804	4402	13,245	19,163	9916	1,682,624	17	17	63	3021	15	4	6
	75,289	1580	4093	5012	5087	4973	496,419	11	11	27	29	13	4	8
Q6														
	137	1542	4138	6818	6818	6640	715,416	15	15	37	39	20	4	9
	115	1499	4153	4887	6517	6504	1,062,732	11	10	26	164	2	4	1
	30	1365	3782	6630	6630	6592	715,190	12	12	50	63	5	4	3