Concise Papers

Schema Vacuuming in Temporal Databases

John F. Roddick, *Member*, *IEEE Computer Society*

Abstract—Temporal databases facilitate the support of historical information by providing functions for indicating the intervals during which a tuple was applicable (along one or more temporal dimensions). Because data are never deleted, only superceded, temporal databases are inherently append-only, resulting, over time, in a large historical sequence of database states. Data vacuuming in temporal databases allows for this sequence to be shortened by strategically, and irrevocably, deleting obsolete data. Schema versioning allows users to maintain a history of database schemata without compromising the semantics of the data or the ability to view data through historical schemata. While the techniques required for data vacuuming in temporal databases have been relatively well covered, the associated area of vacuuming schemata has received less attention. This paper discusses this issue and proposes a mechanism that fits well with existing methods for data vacuuming and schema versioning.

Index Terms—Schema versioning, temporal databases, vacuuming.

1 Introduction

The development of temporal databases allows users a systematic and sound mechanism for storing and retaining information with due regard to both the history of an object and a system's recording of that history. In practice, complete histories are not required and the removal of *obsolete* (as opposed to merely old) information needs to occur. There may be strong pragmatic and business reasons for both the retention of data and its subsequent deletion after a period of time. For example, on one hand, statutory requirements may require data to be held for a given number of years, and on the other hand, exposure to Freedom of Information (FOI) requests may encourage organizations to dispose of unnecessary data.

The term *vacuuming* was first used in the Postgres database system as a mechanism for moving old data to archival storage [1]. It was later refined by Jensen and Mark in the context of temporal databases to refer to the removal of *obsolete* information [2] and subsequently developed into a comprehensive and useable adjunct to temporal databases [3], [4], [5]. Data expiry has also been investigated in the context of data warehouses by Garcia-Molina et al. [6] and others [7].

Schema versioning facilitates the provision of multiple schemata that together form the history of the structure of a database. Historical schemata can be used not only to query data according to the *shape* of the database at some point in the past but also to eliminate (or at least reduce) the need to regenerate application systems when minor (in theory bijective) schema changes are made. They can also allow for prospective data entry, where the structure of future data does not match the current schema.

The development of the TSQL2 language [8] discussed *inter alia* both data vacuuming [9] and schema versioning [10]. In the latter (and in other work [11]), it was noted that metadata also becomes obsolete and, like data, needs a vacuuming process. While schema expiration has not been explicitly discussed, the self-maintenance of temporal views in data warehouses has been investigated by Yang and Widom [12] who use the BCDM (which was also the TSQL2) temporal data model [13] as a base.

This short paper discusses the removal of obsolete schema in temporal databases providing a solution that is largely compatible with existing frameworks.

• *The author is with the School of Computer Science, Engineering and Mathematics, Flinders University, P.O. Box 2100, Adelaide, SA 5001, South Australia. E-mail: roddick@csem.flinders.edu.au.*

Manuscript received 5 July 2007; revised 22 Mar. 2008; accepted 8 Aug. 2008; published online 17 Sept. 2008.
Recommended by acceptance by S. Wang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-07-0331.
Digital Object Identifier no. 10.1109/TKDE.2008.201.

2 Related Research

Over the past decade, substantial research has been undertaken to extend conventional static DBMS to accommodate time [14], [15], [16], [17], [18], [19]. As a result, temporal databases now provide significant semantic and functional advantages. However, their append-only nature results in the need to handle, in a systematic manner, issues that have previously been either inapplicable or can otherwise be trivially dealt with. Some of these issues include the following:

- The management of obsolete data and the need to faithfully manage out-of-date data, as well as queries over that data.
- The management of changing schema (and subschema) definitions, including the deletion of obsolete schema definitions.

We outline below a short background to some associated concepts. Given the space available, there is insufficient space to provide full details, and the reader is directed to the cited work for more information. The BCDM model is given to illustrate the ideas; however, the ideas are more widely applicable to other temporal models.

2.1 The BCDM Temporal Data Model

The bitemporal conceptual data model (BCDM) [13], [20] was developed to bring together a variety of existing models by presenting a single bitemporal *conceptual* model independent of the data model on which the physical system might be built. Using the idea of bitemporal chronons (as the smallest element in the two-dimensional space defined by transaction time and valid time), a relation $\mathcal{R}$ consists of a number of attributes, $A_1, A_2, \ldots A_n$, plus a timestamp attribute T. A tuple can therefore be given as $(a_1, a_2, \ldots, a_n|t)$, where t is a collection of chronons.

The BCDM has been used widely, including in data warehouses, temporal OODBS, schema versioning, and XML support [12], [21], [22], [23], [24], [25].

2.2 Data Vacuuming

Temporal databases are append only, which, as discussed earlier, requires an additional procedure to remove obsolete data. Consider the relation outlined in Fig. 1, which shows the histories of five staff, one of which had an incorrect salary recorded for almost a year (t_2, t_3), one of whom was left in 2004 (t_6), and a third who received a salary increase on 1 April 2007 (t_7, t_8, t_9). Fig. 2 shows the same relation vacuumed according to the specification:

$$v_1 \quad \rho(Empl) : \sigma_{TT_{end} \leq NOW-1yr}(Empl)$$
$$v_2 \quad \rho(Empl) : \sigma_{VT_{end} \leq NOW-3yrs}(Empl).$$

That is, all corrected errors are removed (ρ) after one year, and all other old data are deleted after three years. Note that the deletion of data violates the principle of *faithful history encoding* [3] or *temporal succession* [26]. In essence, these two similar concepts stipulate that the history of an object of interest should be preserved. The deletion of data clearly does not allow this.

Empl	EmplId	Dept	Salary	TT_{start}	TT_{end}	VT_{start}	VT_{end}
t_1	8872	Sales	45K	1 Mar 2004	∞	1 Mar 2004	∞
t_2	9982	Ops	31K	1 Mar 2004	18 Feb 2005	14 Mar 2004	∞
t_3	9982	Ops	32K	18 Feb 2005	∞	14 Mar 2004	∞
t_4	7176	IT	29K	1 Aug 2006	∞	27 Jul 2006	∞
t_5	3376	Ops	49K	23 Oct 2003	26 Jan 2004	25 Oct 2003	∞
t_6	3376	Ops	49K	26 Jan 2004	∞	25 Oct 2003	27 Jan 2004
t_7	7224	Mgmt	67K	1 Dec 2002	7 Apr 2007	11 Nov 2002	∞
t_8	7224	Mgmt	67K	7 Apr 2007	∞	11 Nov 2002	1 Apr 2007
t_9	7224	Mgmt	76K	7 Apr 2007	∞	1 Apr 2007	∞

Fig. 1. Example bitemporal relation before vacuuming.

Empl	EmplId	Dept	Salary	TT_{start}	TT_{end}	VT_{start}	VT_{end}
t_1	8872	Sales	45K	1 Mar 2004	∞	1 Mar 2004	∞
t_3	9982	Ops	32K	18 Feb 2005	∞	14 Mar 2004	∞
t_4	7176	IT	29K	1 Aug 2006	∞	27 Jul 2006	∞
t_7	7224	Mgmt	67K	1 Dec 2002	7 Apr 2007	11 Nov 2002	∞
t_8	7224	Mgmt	67K	7 Apr 2007	∞	11 Nov 2002	1 Apr 2007
t_9	7224	Mgmt	76K	7 Apr 2007	∞	1 Apr 2007	∞

Fig. 2. Example temporal relation after vacuuming (assuming $\mathrm{NOW} = 1\text{-Jun-}2007$).

Empl	EmplId	Dept	Salary	PhD?	TT_{start}	TT_{end}	VT_{start}	VT_{end}
t_1	8872	Sales	45K	υ	1 Mar 2004	∞	1 Mar 2004	∞
t_2	9982	Ops	31K	υ	1 Mar 2004	18 Feb 2005	14 Mar 2004	∞
t_3	9982	Ops	32K	ϕ	18 Feb 2005	∞	14 Mar 2004	∞
t_4	7176	IT	29K	No	1 Aug 2006	∞	27 Jul 2006	∞
t_5	3376	Ops	49K	υ	23 Oct 2003	26 Jan 2004	25 Oct 2003	∞
t_6	3376	Ops	49K	υ	26 Jan 2004	∞	25 Oct 2003	27 Jan 2004
t_7	7224	Mgmt	67K	υ	1 Dec 2002	7 Apr 2007	11 Nov 2002	∞
t_8	7224	Mgmt	67K	Yes	7 Apr 2007	∞	11 Nov 2002	1 Apr 2007
t_9	7224	Mgmt	76K	Yes	7 Apr 2007	∞	1 Apr 2007	∞

Fig. 3. Example temporal relation with new attribute defined on 1 January 2005.

Skyt proposed a *Persistent Views* (P-Views) method to provide data independence for queries in temporal databases with vacuuming capability [4]. Her P-Views mechanism provides a rigorous method whereby subschemas may be defined that is resilient to vacuuming.

2.3 View Maintenance

View maintenance, both for databases [27] and for data warehouses [28], [29] deals with the maintenance of views as the structure (and sources) of the underlying information changes. Since views are decoupled from the schema definition, little of the research deals with the deletion of views and none with the consequences for any dependent data.

2.4 Schema Versioning

While historical schema do not take up a lot of space, they restrict the possible evolution of a schema. *Full schema versioning* is commonly not possible, and in previous work, *partial schema versioning* was identified as a pragmatic compromise. Specifically, partial schema versioning is supported when ... *a database system allows the viewing of all data, both retrospectively and prospectively, through user definable version interfaces. Data updates are allowable through reference to one designated (normally the current) schema definition only* [11].

Schema vacuuming is important because even partial schema versioning is only possible in those cases where the current schema is able to show the effect of updates through historical schemata.

3 Vacuuming Schemata

In the work of Skyt et al. [3] two concepts are defined:

1. **Faithful history encoding** discussed earlier, which is violated when all previously current states are not retained. Data vacuuming violates this principle.
2. **Faithful history querying** that states that only queries able to provide a faithful answer (i.e., an answer that would produce identical results over both vacuumed and unvacuumed databases) are to be allowed.

Since these concepts have been regarded as sensible by a number of researchers, we discuss this work as it relates to these concepts. Note that the extensions proposed here do not restrict the allowable data vacuuming specifications.

3.1 Faithful History Encoding

Unlike for data vacuuming, schema vacuuming does not necessarily violate faithful history encoding. Given two schemata[1] S_{t_1} and S_{t_2}, where $t_1 < t_2$, if $S_{t_1} \preceq S_{t_2}$ (i.e., S_{t_2} dominates S_{t_1} through an information capacity preserving mapping [q.v. 30]), then from the perspective of faithful history encoding, we would not lose access to any data if S_{t_1} was vacuumed, and S_{t_2} was backdated to t_1. However, a new issue now arises in that there are now attributes defined at t_2 that have no data defined for the interval $(t_1, t_2]$. Merely placing nulls in those attributes runs the risk of these nulls being misinterpreted. Thus, a new null value, that of *attribute undefined*, υ, needs to be introduced.

Consider the relation from the example earlier in Fig. 1 in which a schema change was performed to add an attribute on 1 January 2005. Since schema versioning is tied to transaction time, all tuples entered before 1 January 2005 have no value for the new attribute, and if viewed through the revised schema, it must show *attribute undefined*, υ, in the new PhD? attribute, as shown in Fig. 3. Those tuples entered after the schema change can contain either a value

1. We use the notation that schema S_{t_n} is the schema valid during the interval $(t_n, t_{n+1}]$ when schema $S_{t_{n+1}}$ takes over as the current schema.

or the null ϕ value. Note that the v can persist—applying the data vacuuming specification earlier will still have tuples (t_1 and t_7) with an attribute undefined value.

Consider now the case where there are two schemata S_{t_1} and S_{t_2} such that $S_{t_1} \succeq S_{t_2}$. For example, consider that the PhD? attribute existed in S_{t_1} and was deleted in S_{t_2}. In that case, S_{t_1} could not be vacuumed without violating faithful history encoding. However, S_{t_2} could be vacuumed if all tuples added after January 1, 2005 had a value of v in PhD?

In the case where S_{t_1} and S_{t_2} are disjoint, neither can be vacuumed; however, a completed schema (first discussed in [31] following on from the completed relation concept of Clifford and Warren [32]) could be used that would contain the minimal union of all attributes that have been defined during the interval in question.

Formally, given a set of schema $\mathcal{S} = \{S_{t_1}, \ldots, S_{t_i}\}$, consisting of combinations of relations $R_1(A_{1_1} \ldots A_{1_m}) \ldots R_i(A_{i_1} \ldots A_{i_n})$, a completed schema $\mathbf{C}(R'_1 \ldots R'_i)$ over $\mathcal{S}$ is defined such that $\forall S_{t_k} \in \mathcal{S}$, $\forall R_j \in S_{t_k}$, $\exists R'_j \in \mathbf{C} : \forall S_{t_k} \forall A_{j_i} \in R_j : A_{j_i} \in R'_j$.

Note that two caveats are imposed:

- It is assumed that where a version of a schema is created through the merging of two relations (or the splitting of a relation into two), the resulting relation(s) are considered new with the old relation(s) deleted.
- It is assumed that when the primary keys of a relation are amended, a new relation is created.

In all cases, a completed schema can be created but the user will be required to facilitate historical data viewing. For example, consider the merging of two employee relations containing, say, the details of employees of organizations being combined. Consider now a new employee's data—to which original relation is this data to be attributed?

In the general case, if $\mathbf{C}(R'_1 \ldots R'_i)$ is a completed schema defined over a sequence of schemata $S_{t_1} \ldots S_{t_n}$, then $S_{t_1} \ldots S_{t_n}$ can be vacuumed subject to $\mathbf{C}$ being used as a replacement and undefined values being populated with a value of v. In the simple case of two temporally adjacent schema such that $S_{t_h} \preceq S_{t_{h+1}}$, S_{t_h} can be removed as long as $S_{t_{h+1}}$ is applied over the complete interval and undefined attributes are populated with v.

In terms of update, since data updates are allowable through reference to one designated schema, there are no issues with v.

3.2 Faithful History Querying

In respect of schema vacuuming, faithful history querying can be violated in two ways:

1. When a query implicitly or explicitly specified a schema that has been vacuumed.
2. When a query is *unsafe*.[2] For schema vacuuming, this means it would either

   a. provide in the result, or
   b. evaluate over

   an attribute value of *attribute undefined*, v. For example,

$$\pi_{(EmplId,Salary)}\sigma_{(Dept='Sales')}(Empl)$$
$$\pi_{(EmplId,PhD?)}\sigma_{(Dept='IT')}(Empl)$$

   are safe, whereas

$$\pi_{(EmplId,Salary)}\sigma_{(PhD?='Yes')}(Empl)$$
$$\pi_{(EmplId,PhD?)}\sigma_{(Dept='Mgmt')}(Empl)$$

   are not.

2. Unsafe in this context implies the ability for the user to misinterpret the results as a result of the overloading of the semantics of *null*.

There are two strategies that can be adopted. First, we can adopt the same approach as that of Skyt et al. [3] in that the query is rejected. This is the only option for Cases 1 and 2b above and can also be applied in Case 2a. Secondly, for Case 2a, we can also, optionally, provide a result as long as the *attribute undefined*, v, values are clearly distinguished and understood.

3.3 Other Issues

3.3.1 Single versus Multi-Pool Extensional Data

De Castro et al. [33] discuss multi-pool data storage. In this architecture, each version has its own part of the extensional data store corresponding to the schema active at that point in history. That is, a query about data stored at t_n references the data pool for the schema active at t_n.

The multipool solution has little need for the ideas outlined in this paper until multi-schema queries are issued. At that point, because hybrid schema are then required, the strategies outlined here can be usefully employed.

3.3.2 Consistency with Previous Proposals

In the TSQL2 proposals [10] two suggestions are made regarding schema vacuuming:

1. **Automatic vacuuming.** *All schema definitions that predate all data (both in format and in transaction-time values) are to be considered obsolete and should be deleted.*

   In practice, since it is unlikely that data vacuuming statements that use TT_{Start} as a clause will have been specified, we will need to inspect the database following any data vacuuming to ascertain if any schema are now obsolete. That is, are there any schema S_{t_n} valid between t_n and t_{n+1} such that $\forall t : t_n \leq t < t_{n+1}$, $\mathbf{C}(S_{t_n}, S_{t_{n+1}}) = S_{t_n}$?
2. **Selective vacuuming.** *Old schema definitions are considered valuable independent of whether data exists and may only be deleted through an explicit request to vacuum.*

   In this case, the inverse criterion to that above would be used to warn users of a potential loss of data.

4 Conclusions

Schema vacuuming is an important consideration, but it must be done with a view to maintaining, as far as possible, concepts such as faithful history querying. This short paper has outlined a mechanism centred on the use of completed schema and a new *attribute undefined*, v value, which is compatible to preexisting research such as that proposed for TSQL2 and in work such as that discussed by Skyt et al. [3].

Acknowledgments

The author would like to thank his colleagues with the School of Computer Science, Engineering and Mathematics, Flinders University, for discussions on this topic over a long period of time. He would also like to thank the anonymous referees whose insightful comments helped to improve this paper.

References

[1] M. Stonebraker and L. Rowe, "The Design of Postgres," *Proc. ACM SIGMOD '86*, C. Zaniolo, ed., pp. 340-355, 1986.
[2] C.S. Jensen and L. Mark, "A Framework for Vacuuming Temporal Databases," Technical Report CS-TR-2516, Univ. of Maryland, College Park, 1990.
[3] J. Skyt, C.S. Jensen, and L. Mark, "A Foundation for Vacuuming Temporal Databases," *Data and Knowledge Eng.*, vol. 44, no. 1, pp. 1-29, 2003.
[4] J. Skyt, "Specification-Based Techniques for the Reduction of Temporal and Multidimensional Data," PhD dissertation, Aalborg Univ., 2001.

[5] D. Toman, "Expiration of Historical Databases," *Proc. Eighth Int'l Symp. Temporal Representation and Reasoning, (TIME '01),* pp. 128-135, 2001.
[6] H. Garcia-Molina, W. Labio, and J. Yang, "Expiring Data in a Warehouse," *Proc. 24th Int'l Conf. Very Large Data Bases (VLDB '98),* A. Gupta, O. Shmueli, and J. Widom, eds., pp. 500-511, 1998.
[7] J. Skyt, C.S. Jensen, and T.B. Pedersen, "Specification-Based Data Reduction in Dimensional Data Warehouses," *Proc. 18th Int'l Conf. Data Eng. (ICDE '02),* p. 278, 2002.
[8] *The TSQL2 Temporal Query Language,* R. Snodgrass, ed. Kluwer Academic Publishing, 1995.
[9] C. Jensen, "Vacuuming," *The TSQL2 Temporal Query Language,* R. Snodgrass, ed., chapter 23, pp. 451-462, Kluwer Academic Publishing, 1995.
[10] J.F. Roddick and R. Snodgrass, "Schema Versioning Support," *The TSQL2 Temporal Query Language,* R. Snodgrass, ed., chapter 22, pp. 427-449, Kluwer Academic Publishing, 1995.
[11] J.F. Roddick, "A Model for Schema Versioning in Temporal Database Systems," *Proc. 19th Australian Computer Science Conf.,* K. Ramamohanarao, ed., pp. 446-452, 1996.
[12] J. Yang and J. Widom, "Temporal View Self-Maintenance," *Proc. Seventh Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT '00),* C. Zaniolo, P. Lockemann, M. Scholl, and T. Grust, eds., pp. 395-412, 2000.
[13] C.S. Jensen, M.D. Soo, and R.T. Snodgrass, "Unifying Temporal Data Models via a Conceptual Model," *Information Systems,* vol. 19, no. 7, pp. 513-547, 1994.
[14] N. Kline, "An Update of the Temporal Database Bibliography," *SIGMOD Record,* vol. 22, no. 4, pp. 66-80, 1993.
[15] L. McKenzie, "Bibliography: Temporal Databases," *SIGMOD Record,* vol. 15, no. 4, pp. 40-52, 1986.
[16] G. Özsoyoğlu and R.T. Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Trans. Knowledge and Data Eng.,* vol. 7, no. 4, pp. 513-532, Aug. 1995.
[17] M. Soo, "Bibliography on Temporal Databases," *SIGMOD Record,* vol. 20, no. 1, pp. 14-23, 1991.
[18] R. Stam and R. Snodgrass, "A Bibliography on Temporal Databases," *Data Eng.,* vol. 7, no. 4, pp. 53-61, 1988.
[19] Y. Wu, S. Jajodia, and X. Wang, "Temporal Database Bibliography Update," *Temporal Databases—Research and Practice,* O. Etzion, S. Jajodia, and S. Sripada, eds., pp. 338-366, Springer, 1998.
[20] C. Jensen, R. Snodgrass, and M. Soo, "The TSQL2 Data Model," *The TSQL2 Temporal Query Language,* R. Snodgrass, ed., pp. 157-240, Kluwer Academic Publishers, 1995.
[21] F. Grandi and F. Mandreoli, "The Valid Web: An XML/XSL Infrastructure for Temporal Management of Web Documents," *Proc. First Int'l Conf. Advances in Information Systems (ADVIS),* 2000.
[22] J. Yang, "Temporal Data Warehousing," PhD dissertation, Stanford Univ., 2001.
[23] J.F. Roddick, F. Grandi, F. Mandreoli, and M.R. Scalas, "Beyond Schema Versioning: A Flexible Model for Spatio-Temporal Schema Selection," *Geoinformatica,* vol. 5, no. 1, pp. 33-50, 2001.
[24] F. Grandi and F. Mandreoli, "A Formal Model for Temporal Schema Versioning in Object-Oriented Databases," *Data and Knowledge Eng.,* vol. 46, no. 2, pp. 123-167, 2003.
[25] P. Terenziani and R.T. Snodgrass, "Reconciling Point-Based and Interval-Based Semantics in Temporal Relational Databases: A Treatment of the Telic/Atelic Distinction," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 5, pp. 540-551, May 2004.
[26] J.F. Roddick and J.D. Patrick, "Temporal Semantics in Information Systems—A Survey," *Information Systems,* vol. 17, no. 3, pp. 249-267, 1992.
[27] A. Gupta and I.S. Mumick, "Maintenance of Materialized Views: Problems, Techniques and Applications," *IEEE Data Eng. Bull.,* special issue on materialized views and warehousing, vol. 18, no. 2, pp. 3-18, 1995.
[28] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment," *Proc. ACM SIGMOD '95,* pp. 316-327, 1995.
[29] D. Agrawal, A.E. Abbadi, A. Singh, and T. Yurek, "Efficient View Maintenance in Data Warehouses," *Proc. ACM SIGMOD '97,* J. Peckham, ed., pp. 417-427, 1997.
[30] R. Miller, Y. Ioannidis, and R. Ramakrishnan, "The Use of Information Capacity in Schema Integration and Translation," *Proc. 19th Int'l Conf. Very Large Data Bases (VLDB '93),* R. Agrawal, S. Baker, and D. Bell, eds., pp. 120-133, 1993.
[31] J.F. Roddick, "SQL/SE—A Query Language Extension for Databases Supporting Schema Evolution," *SIGMOD Record,* vol. 21, no. 3, pp. 10-16, 1992.
[32] J. Clifford and D. Warren, "Formal Semantics for Time in Databases," *ACM Trans. Database Systems,* vol. 8, no. 2, pp. 214-254, 1983.
[33] C. De Castro, F. Grandi, and M. Scalas, "Schema Versioning for Multitemporal Relational Databases," *Information Systems,* vol. 22, no. 5, pp. 249-290, 1997.