A Temporal Study of Data Sources to Load a Corporate Data Warehouse

Carme Martín and Alberto Abelló

Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya.
Jordi Girona Salgado 1-3. E-08034 Barcelona, Catalunya.
{martin@lsi.upc.es | aabello@lsi.upc.es }

Abstract. The input data of the corporate data warehouse is provided by the data sources, that are integrated. In the temporal database research area, a bitemporal database is a database supporting valid time and transaction time. Valid time is the time when the fact is true in the modeled reality, while transaction time is the time when the fact is stored in the database. Defining a data warehouse as a bitemporal database containing integrated and subject-oriented data in support of the decision making process, transaction time in the data warehouse can always be obtained, because it is internal to a given storage system. When an event is loaded into the data warehouse, its valid time is transformed into a bitemporal element, adding transaction time, generated by the database management system of the data warehouse. However, depending on whether the data sources manage transaction time and valid time or not, we could obtain the valid time for the data warehouse or not. The aim of this paper is to present a temporal study of the different kinds of data sources to load a corporate data warehouse, using a bitemporal storage structure.

1 Introduction

As defined in [4], a data warehouse (DW) is an architectural structure that supports the management of "Subject-oriented", "Integrated", "Time-variant", and "Non-volatile" data. A temporal database (TDB) is introduced in [9] as a database that supports "Valid time" (VT) (i.e. the time when the fact becomes effective in reality), or "Transaction time" (TT) (i.e. the time when the fact is stored in the database), or both times. Note that this definition excludes "User-defined time", which is an uninterpreted attribute domain of time directly managed by the user and not by the database system. A "bitemporal database" is a database that supports VT and TT. The affinity between both concepts (i.e. DW and TDB) may not be obvious. However, time references are essential in business decisions, and the dissection of both definitions shows their closeness.

We consider the accepted definition of DW in [4] could be rewritten in terms of TDB concepts. Firstly, "Time-variance" simply specifies that every record in the DW is accurate relative to some moment in time. On the other hand, the definition of VT in [6] states that it is the time when the fact is true in the modeled reality. Therefore,

both outline the importance of showing when data is correct and exactly corresponds to reality. Moreover, "Non-volatility" refers to the fact that changes in the DW are captured in the form of a "time-variant snapshot". Instead of true updates, a new "snapshot" is added to the DW in order to reflect changes. This concept can be clearly identified with that of TT, defined in [6] as the time when the fact is current in the database.

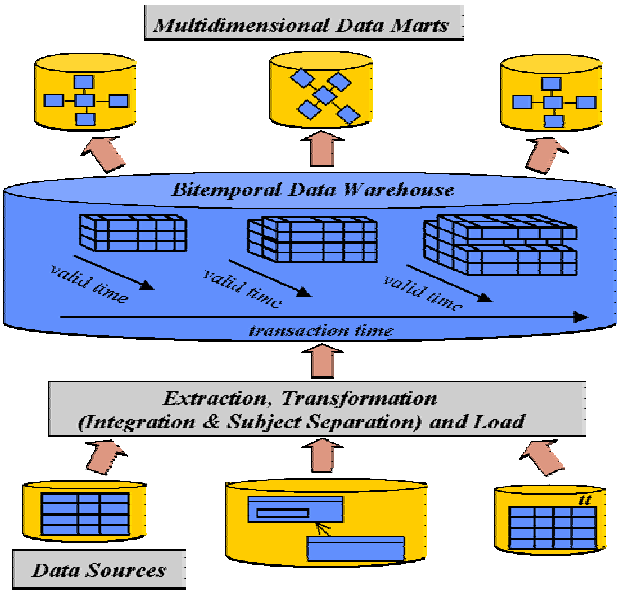

Fig. 1. A Data Warehouse as a Bitemporal Database

In [1], we define a DW as *a bitemporal database containing integrated and subject-oriented data in support of the decision making process*, as it is sketched in figure 1. The first implication of this definition is that TT is entirely maintained by the system, and no user is allowed to change it. Moreover, the system should also provide specific management mechanisms for VT. The importance of this temporal conception is also outlined in [8], which asks DW systems for support of advanced temporal concepts.

Data in the DW comes from independent heterogeneous sources. Therefore, TT and, specially, VT, introduced in the DW depend on which times are provided by the data sources. In [1], we propose a storage structure to implement a bitemporal DW. We concentrate our previous work is the most common case of data sources, i.e. specific and logged sources. In this paper, we are going to explain the behaviour of our bitemporal storage structure for all the different kinds of data sources to show that it can also be used with all possible data sources.

The paper is organized as follows. Next section describes the bitemporal storage structure used throughout the paper. Section 3 explains the temporal study of the different kinds of data sources and the implementation of these in the bitemporal storage structure. Finally, section 4 provides some conclusions.

2 A Bitemporal Storage Structure

In [1], we present a bitemporal storage structure, named Current/Historical, consisting of a *Current* table that reflects current data and a set of *Historical* tables that show the historical evolution of the data, as depicted in figure 2. The set of *Current* tables in the DW give rise to the *Operational Data Store*, defined in [4].

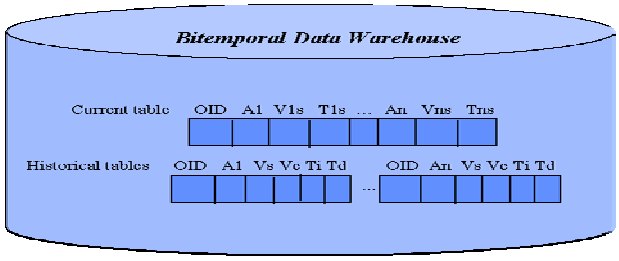

Fig. 2. *Current* and *Historical* tables in the bitemporal DW

A bitemporal event occurs at a starting VT and is true until an ending VT. However, if the ending VT is not known at this moment, since data is currently valid in the sources, we use the special VT value "Now", whose semantics are explained in [3]. For example, if we hire an employee, the starting VT will be her/his starting hiring data, and the ending VT will be the value "Now", until s/he is fired. DW insertions initialize the starting TT to the current TT and the ending TT to the value "Until_changed" (UC). As the current time inexorably advances, the value of UC always reflects the current time. Moreover, the current TT is denoted by “Current”.

The *Current* table only contains two different temporal values for each attribute (A_i), i.e. the starting VT (V_{is}) and the starting TT (T_{is}). The other time values for current data always contain “Now” for ending VT and “UC” for ending TT, so they do not need to be stored. Moreover, we have one *Historical* table for each set of attributes with the same temporal behaviour. In this way, each *Historical* table contains only one semantic concept, and all attributes change at a time. For example, if we store employee information (only one subject), we should use a different structure for home address and telephone, and another one for work address and telephone (two different temporal behaviours). If the company reallocates somebody in a new room, only her/his job data will change. If s/he moves to a bigger house, only home data will change. Even though all data regards the same subject, it shows a different time behaviour. Without loss of generality, in this paper, we will assume that every table contains at most one attribute. Therefore, *Historical* tables contain the history for each attribute, with the starting VT (V_s), the ending VT (V_e), the TT when the insert was processed (T_i) and the TT when the delete was processed (T_d). From these four temporal values, we can easily reconstruct the whole history of the value.

In TDB research area, insertion, deletion, and modification operations, are defined in [7]. In addition, in [10], temporal insertion, deletion, and modification operations are explained for relational databases with time support. However, in our bitemporal storage structure, we require a redefinition of TDB insertion, deletion and

modification operations. On loading the DW, all we need to do to process an *insert* is add a new record to the *Current* table. The processing of a *delete* is not much more difficult: the corresponding record is removed from the *Current* table, and a new one is added to each one of the *Historical* tables. The *modification* changes the old values of the *Current* table with the new values and adds a new record in the corresponding *Historical* table of the modified attribute. This behaviour can be easily inferred from *deletion* and *insertion* behaviours.

3 Temporal Study of Data Sources

The input data of the DW is provided by the data sources, that are integrated. Depending on whether the data sources manage TT and VT or not, we could obtain the VT for the DW or not. TT in the DW can always be obtained, because it is internal to a given storage system. When an event is loaded into the DW, its VT, supplied by the "Extraction, Transformation and Load" (ETL) module, is transformed into a bitemporal element, adding TT, generated by the DW DBMS. In the next sections, let us study the different kinds of data sources proposed in [5] to be implemented in our bitemporal storage structure.

We consider throughout the paper, as example, a bitemporal DW with a *courses* relation with an *OID*, a *Name* and a *Cost* attributes.

3.1 Snapshot and Queryable Sources

Snapshot sources are sources that the only way to access to the data source content is through a dump of its data. A queryable source is a data source that offers a query interface. From snapshot and queryable sources that do not keep any kind of time, we can only store the TT in the DW. From snapshot and queryable sources, if they do not have any temporal information, we can only consider timestamping the data while we extract them. In the absence of true VT, all we can do is approximate it by the DW TT.

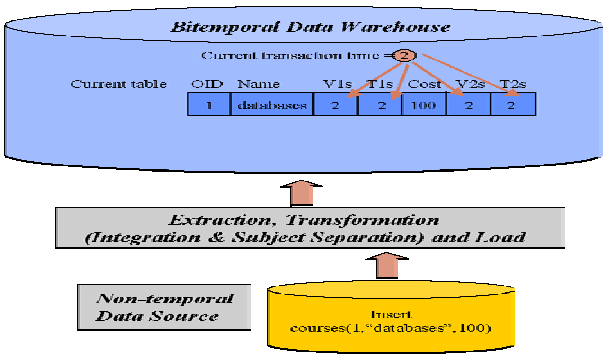

Fig. 3. The *databases* course insertion in snapshot and queryable sources

In figure 3, we show the insertion of a *databases* course with a *100* euros cost, for the *course* relation. Inserting this course into the bitemporal DW, the starting TT is recorded. Given the bitemporal nature of the DW, all we can do in this case is approximate the starting VT by means of TT information. The best approximation we could obtain for the starting VT is the starting point of the "update window", because all we know is that the event occurred before the beginning of the DW load. Moreover, since during the "update window" queries are not allowed, we can consider that the load is atomic, in the sense that there is no temporal order among the operations. Therefore, the TT can also be fixed at the beginning of the "update window". Thus, TT and VT will have the same value.

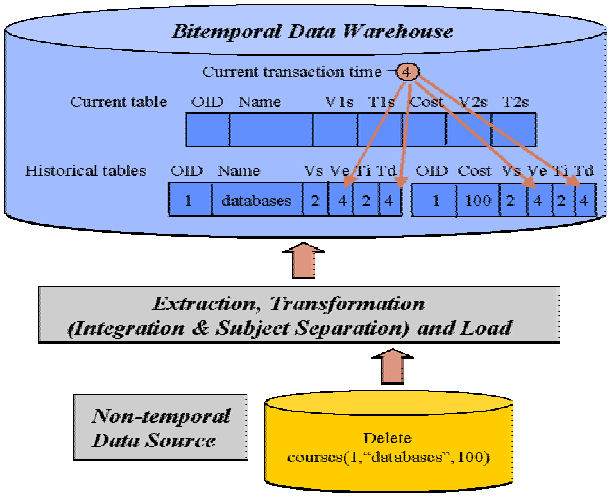

Fig. 4. The *databases* course deletion in snapshot and queryable sources

In figure 4, the deletion process of the *databases* course is described. The deletion operation eliminates the bitemporal element of the *Current* table and adds a new one in each *Historical* table. Similar to the insertion operation, we approximate now the ending VT by means of TT.

3.2 Specific and Logged Sources

Specific sources are able to write "delta files" that describe the system actions. Logged sources have a "log file" where all their actions are registered. From specific and logged (those able to keep track of the performed operations) sources, if they timestamp the entries with the source TT, we can approximate the VT by means of it. If no other information exists, the data can be considered valid while it is current in the operational database. This is the most usual environment for a DW.

In figure 5, we show the insertion of a *databases* course with a *100* euros cost and with a source TT value of *1*. We can see that source TT is converted into VT, and we also have the TT of the DW.

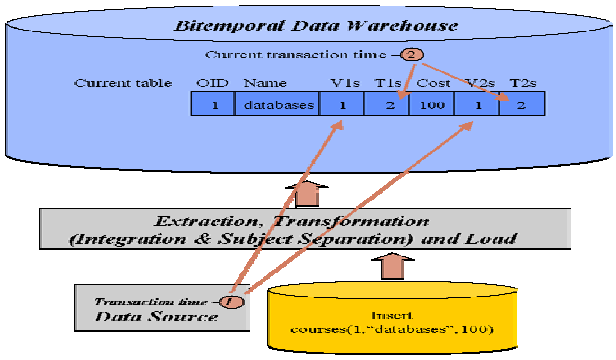

Fig. 5. The *databases* course insertion in specific and logged sources

In figure 6, the deletion process of the *databases* course is described. When a deletion operation comes from the data source, it has an ending TT value to be converted in an ending VT value in the DW. As it is shown, a logical deletion generates the physical removal of the existing bitemporal element in the *Current* table. The ending TT of the *Historical* table is the TT of the DW. However, this is not enough and a new bitemporal element is added to the *Historical* tables, which expresses that from now on we know the ending VT value, i.e. the timestamp of the data source.

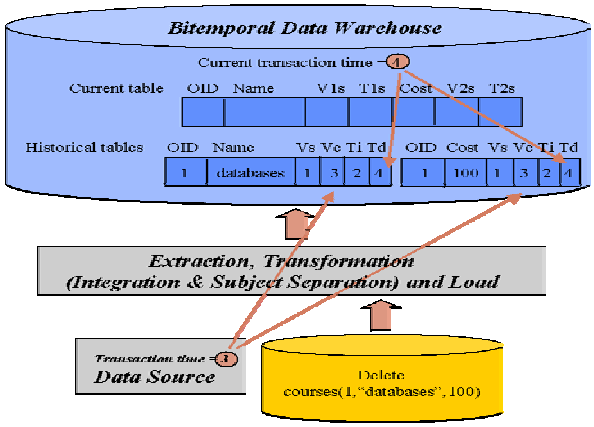

Fig. 6. The *databases* course deletion in specific and logged sources

3.3 Callback and Internal Action Sources

Callback sources are sources that provide triggers or other active capabilities. Internal action sources are similar to callback sources, except that to define triggers in the data

source requires to create auxiliary relations called "delta tables". In these "cooperative" sources, the TT in the sources needs to be used again to approximate the VT. However, this kind of data sources offer two different kinds of DW load:

Deferred load. The most common possibility is to use the triggers to generate "delta files", and later on use these files to load the DW. Thus, the temporal information in the "delta files" will be the source TT. We have already described this case in section 3.2.

Real-time load. Another possibility would be that both repositories (i.e. the data source and the DW) are updated at the same time. Then, the TT of the DW corresponds to source TT. Notice that, it is the same assumption that we have considered for snapshot and queryable sources. Therefore, this case has the same temporal behaviour than snapshot and queryable sources, explained in section 3.1. Nevertheless, they do not have the same temporal knowledge. In callback and internal action sources we really have a source TT to approximate the VT, while in snapshot and queryable sources we have no temporal information in the source, so this is a better approximation.

3.4 Bitemporal Data Sources

Bitemporal data sources are sources whose data are stored in a bitemporal database. From bitemporal data sources (not considered in [5]), we could obtain true VT besides TT. Moreover, we also know the TT of the DW. Therefore, we should choice one temporal attribute out of those two of the sources to be used as VT in the DW (the other one will be managed as a user-defined time attribute):

Source TT used as VT. If the VT of the DW is obtained from the source TT, the source VT could be an additional user-defined time attribute to be considered. In this case, bitemporal data sources will have the same temporal behaviour than specific and logged sources (explained in section 3.2) with an additional user-defined time attribute to express true valid time information. The temporal information provided to the ETL process will be: source TT (that will give rise to the VT of the DW), and source VT (that will be treated as a user-defined time attribute). Considering temporal attributes in this way allows to keep using effectively the Current/Historical storage structure.

Source VT used as VT. If the VT of the DW is obtained from the source VT, the source TT could be an additional user-defined time attribute to be considered. These bitemporal sources provide the following temporal information: source VT (that will give rise to the VT of the DW), and source TT (that will be treated as a user-defined time attribute). Since VT in the data source is an interval, we need to add another temporal attribute to the *Current* table to record the ending VT.

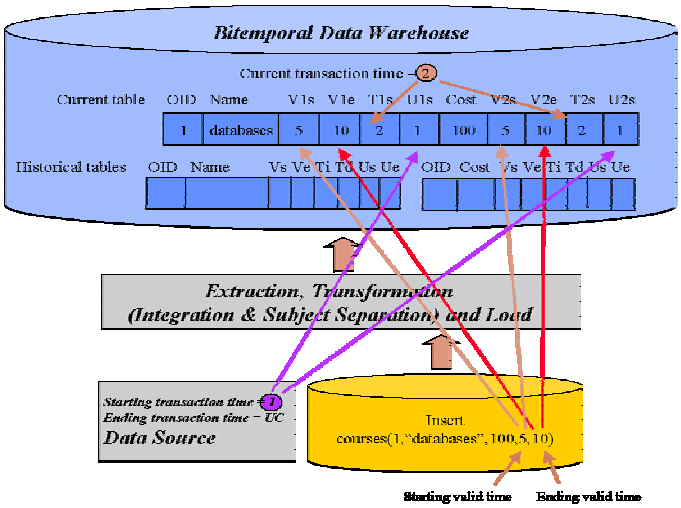

Fig. 7. The *databases* course insertion in bitemporal sources

Figure 7 presents this new possibility having bitemporal sources. Two new attributes need to be added to the *Current* table for every attribute: One for ending VT and another one for the user-defined time representing source TT. Notice that, the *Current* table represents current data in the source. Therefore, the ending source TT will always be UC so that it is not necessary to record it. In this example, starting and ending VT are later than the load of the DW.

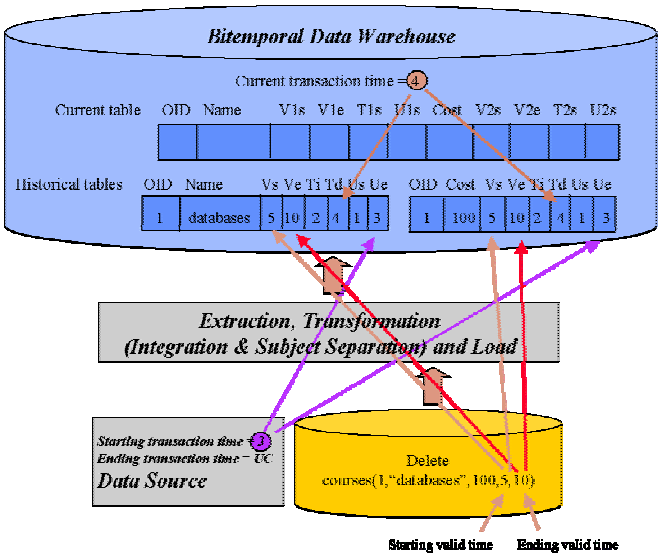

Fig. 8. The *databases* course deletion in bitemporal sources

Figure 8 shows the deletion of the *databases* course inserted in figure 7. Regarding the *Historical* tables, also two new user-defined time attributes need to be added to each table: one for starting source TT and another one for ending TT. We already had two attributes for starting and ending VT.

The main problem in this case (not taking into account the increase in the number of attributes of the different tables) is that the OID is no more an identifier by itself for the *Current* table. Even more, we cannot guarantee that there are the same number of current values for each attribute, so that we should divide the *Current* table into independent tables for every attribute. Thus, we would have the same number of *Current* and *Historical* tables, which would worsen the performance of this storage technique.

In this kind of sources, in order to reduce storage space, when two time intervals overlap or are adjacent, the coalescing operation of TDB [2] could be applied either to *Current* or *Historical* tables. When either VT or TT intervals have identical non-temporal attribute values in two different tuples, then the coalescing operation can obtain only one tuple from both if the other time interval that do not coincide overlap or the ending point of one of them is the starting point of the other. Even if the bitemporal source uses the coalescing operation, we can still find in the DW tuples to be coalesced, if they come from different sources.

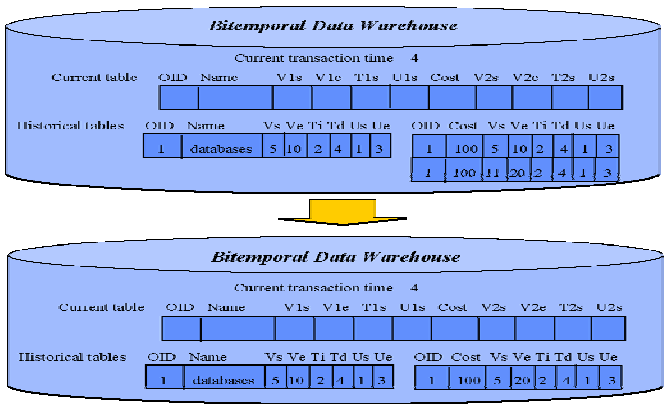

Fig. 9. A coalescing example

As example, in figure 9, we see that the *databases* course information could be coalesced in only one tuple.

4 Conclusions

In this paper, we have presented a temporal study of the different kinds of data sources to load a DW. The correspondences between temporal attributes in the data sources and those in the DW have been analyzed. In the corporate DW, we have identified the two existing orthogonal temporal dimensions: valid time dimension and

transaction time dimension. In this bitemporal DW environment, we have used our bitemporal storage structure [1] to represent all the temporal data source knowledge obtained by the different kinds of data sources. Analyzing these different data sources, presented in [5] with an additional type, i.e. bitemporal source, we found that in some cases snapshot and queryable sources could have the same temporal behaviour than callback and internal action sources (just the latter being more precise). In general, these "cooperative" sources would behave like specific and logged sources. Regarding bitemporal data sources, they are more difficult to manage and, in general, would need an ad hoc storage structure. However, using an appropriate interpretation, they can also be treated as specific and logged sources. Thus, we can use the Current/Historical bitemporal storage structure to warehouse any kind of data sources.

Acknowledgments

The authors would like to thank Núria Castell for the support she has given to this work. This work has been partially supported by the Spanish Research Program PRONTIC under project TIC2000-1723-C02-01.

References

1. Abelló, A.; Martín, C. "A Bitemporal Storage Structure for a Corporate Data Warehouse". Proc. of the 5th. Int. Conf. on Enterprise Information Systems (ICEIS). pages 177-183, 2003.
2. Böhlen, M.; Snodgrass, R.T.; Soo, M.D. "Coalescing in Temporal Databases". In Proc. of the 22nd. Int. Conf. on Very Large Data Bases (VLDB). pages 180-191, 1996.
3. Clifford, J.; Dyreson, C.; Isakowitz, T.; Jensen, C.S., Snodgrass, R.T. "On the Semantics of 'Now' in Databases". ACM Transactions on Database Systems. 22(2):171-214, 1997.
4. Inmon, W. H., Imhoff, C.; Sousa, R. "Corporate Information Factory". John Wiley & Sons, second edition, 1998.
5. Jarke, M.; Lenzerini, M.; Vassilious, Y.; Vassiliadis, P., editors, "Fundamentals of Data Warehousing". Springer-Verlag, 2000.
6. Jensen, C.S.; Clifford, J.; Gadia, S.K.; Segev, A.; Snodgrass, R.T. "A Glossary of Temporal Database Concepts". ACM SIGMOD Record. 21(3):35-43, 1992.
7. Jensen, C.S.; Soo, M.D.; Snodgrass R.T. "Unifying Temporal Data Models via a Conceptual Model". Information Systems. 19(7):513-547, 1994.
8. Pedersen, T.B.; Jensen, C.S. "Research Issues in Clinical Data Warehousing". In Proc. of the 10th. Int. Conf. on Statistical and Scientific Database Management (SSDBM). pages 43-52, 1998.
9. Snodgrass, R.T.; Ahn, I. "Temporal Databases". IEEE Computer. 19(9):35-42, 1986.
10. Snodgrass, R.T. "Developing Time-Oriented Database Applications in SQL". Morgan Kaufmann Publishers, 2000.