A Bitemporal Storage Structure for a Corporate Data Warehouse*

Alberto Abelló
Universitat Politècnica de Catalunya
C/ Jordi Girona 1-3 E-08034 Barcelona
E-mail: aabello@lsi.upc.es

Carme Martín
Universitat Politècnica de Catalunya
C/ Jordi Girona 1-3 E-08034 Barcelona
E-mail: martin@lsi.upc.es

Key words: Data Warehouse, Temporal Databases, Temporal O-O Models

Abstract: This paper brings together two research areas, i.e. "Data Warehouses" and "Temporal Databases", involving representation of time. Looking at temporal aspects within a data warehouse, more similarities than differences between temporal databases and data warehouses have been found. The first closeness between these areas consists in the possibility of a data warehouse redefinition in terms of a bitemporal database. A bitemporal storage mechanism is proposed along this paper. In order to meet this goal, a temporal study of data sources is developed. Moreover, we will show how Object-Oriented temporal data models contribute to add the integration and subject-orientation that is required by a data warehouse.

1 Introduction

Data Warehousing has been an active research area in the last years. Roughly, it develops mechanisms to store and manage enterprise data to support the decision making processes. We can see basic concepts, as well as quality issues in (Jarke et al., 2000). A recent study of research issues in the field is in (Vassiliadis, 2000). Analyzing this study a clear practical business origin can be found. In contrast to this, temporal database area begins and evolves in an academic environment. Temporal database researchers, as it is shown in the bibliography of (Wu et al., 1998), have produced important results in this field. In addition to the common points we will explain further, the different approaches of these two areas are another interesting reason to relate them. Thus, the aim of this paper is to analize the "Data Warehouse" (DW) from the perspective of "Temporal Database" (TDB) community. Specifically, an storage mechanism and the usage of temporal "Object-Oriented" (O-O) concepts are proposed for the DW.

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. As defined in (Inmon et al., 1998), a DW is an architectural structure that supports the management of "Subject-oriented", "Integrated", "Time-variant", and "Non-volatile" data. A TDB is introduced in (Snodgrass and Ahn, 1986) 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.

In the literature, we find papers presenting the modeling of DWs as multidimensional databases. However, as argued in (Abelló et al., 2000), the structure of multidimensional "Star Schemas" is too rigid and absolutely query oriented, while the design of the central, corporate DW should be data driven.

The main advantages of "Star Schemas" are their simplicity and proximity to the business analysis concepts. It makes them quite easy to be understood by the final users. However, even more important than this is the fact that they imply a given kind of queries. This structure is quite concrete, and allows to propose specific optimizations, access paths, and storage methods. They are probably the best way to study a reduced set of facts with regard to the desired analysis dimensions. Nevertheless, they are not as good at keeping the data of the whole business, or just being

*Our work has been partially supported by the Spanish Research Program PRONTIC under project TIC2000-1723-C02-01.

accessed by data mining algorithms.

In our architecture, the DW is what (Kimball et al., 1998) calls the "Storage Structure", and it is only accessed to solve a small number of specific queries. As it is outlined in (Inmon et al., 1998), there is not an homogeneous access pattern in the DW. Multidimensional modeling and "Star Schemas" perfectly suit for small departmental DWs. Therefore, from the DW, some smaller, customized data structures, known as "Data Marts" are built. Thus, the non-multidimensional DW is mainly used to feed multidimensional "Data Marts" or data sets to be mined.

First of all, section 2 defines the requirements for the data model of the DW. Section 3 presents a bitemporal definition of the DW, and the different situations we can find for obtaining temporal information. Section 4 describes how temporal storage structures could be used in a DW environment, and an alternative technique is proposed, and compared with existing structures. Finally, section 5 concludes the paper.

2 Characteristics of the DW Model

It is well known that data models used in the operational world are not appropriate for analysis tasks. Lots of work have been devoted to model multidimensional "Data Marts". Let us analyze in this section some characteristics that a suitable DW data model should offer:

Time management Time is an omnipresent element in analysis tasks, and a difficult element to handle. As we will discuss in section 3, a DW is a bitemporal database with some specific characteristics. Thus, the data model should support VT and TT.

Expressiveness Going to the definition in (Inmon et al., 1998), we observe that a DW is "Integrated". In (Saltor et al., 1991), it is explained the importance of a semantically rich data model to overcome semantic heterogeneities in the data sources. The data model should allow to represent, at least, as much semantics as any of the data sources. Moreover, the importance of metadata is well known. The more metadata we could represent in the database schema, the better.

Internal identifiers Due to the long term nature of the DW, it is particularly difficult to find identifiers among the attributes of the objects. The data model should provide internal, real word independent identifiers.

(Saltor et al., 1991) argues that O-O models should be used for integration. In a DW, semantic richness of O-O models, not only would facilitate integration, but also would help analysts to understand the real meaning of data. Moreover, it could facilitate the achievement of "Subject-oriented", if we define each subject as a separate object. Each of this objects will encapsulate all data regarding the corresponding subject. In this sense, we propose the usage of a temporally enhanced O-O data model for the DW.

In a database containing historical data, where any attribute could change, the presence of system defined object identifiers (i.e. OID) seems mandatory. Besides these identifiers, an O-O model has objects, attributes and semantic relationships between the objects. To provide traceability (an essential for analysis), all these elements must always be timestamped with TT. Thus, this should be implicitly offered by the system. However, some objects, attributes and relationships could be related or not to VT depending on analysis needs, modeling decisions, or just availability in data sources. For objects, a special kind of VT (namely lifespan) should be distinguished here. This should be directly associated to OIDs, as in (Bertino et al., 1997) or (Steiner and Norrie, 1997).

TT is generally not a single time instant, it has duration. However, just being able to represent sets of instants as time intervals, in the sense of (Böhlen et al., 1998), should be enough. Sets of intervals could be used for TT as well as VT of the attributes and relationships. Nevertheless, lifespan should be considered a continuous set of instants (only one interval). Once an object has been destroyed, it cannot be recovered. (Steiner and Norrie, 1997) defines a "temporal object role model" timestamping instantiation relationships, which can be easily translated to this framework allowing dynamic classification. Thus, if an object is an instance of a given class for a given period of time, we timestamp the relationship between the class and the object with the interval. We can keep the same object and OID, while being able to change its classification. Given `Student` and `Employee` classes, we do not destroy the instance of a person and create a new one just because s/he finished her/his studies and found a job. The lifespan is continuous, while the VT of classification may be a set of intervals (later on, the person who abandoned the studies could return to her/his studies, generating two different VT intervals for her/his classification in `Student` class).

3 A Bitemporal Database

We consider the accepted definition of DW in (Inmon et al., 1998) 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 (Dyreson et al., 1994) states that it is the time when the fact is true in the modeled real-

ity. 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 (Dyreson et al., 1994) as the time when the fact is current in the database.

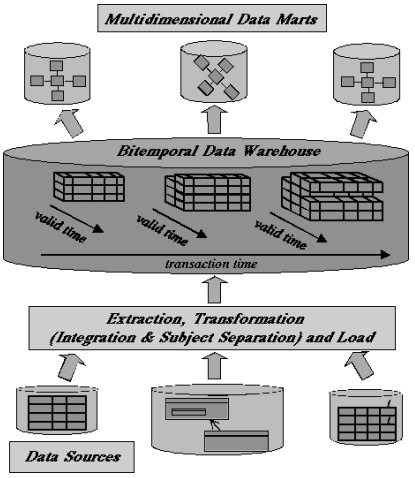

Figure 1: A Data Warehouse as a Bitemporal Database

Defining a "bitemporal database" as a database supporting VT and TT, a DW is *a bitemporal database containing integrated, 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 (Pedersen and Jensen, 1998), which asks DW systems for support of advanced temporal concepts.

3.1 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. Let us classify the different kinds of data sources in (Jarke et al., 2000) based on the temporal information we could obtain from them (see figure 2):

1. From "snapshot" and "queriable" sources that do not keep any kind of time, we can only store the TT in the DW.
2. From "logged" and "specific" (those able to write "delta files") sources, if they timestamp the entries, we can consider that the TT in the sources corresponds to VT for the DW. If no other information exists, the data is considered valid while it is current in the operational database.
3. From "cooperative" (for instance, those that implement triggers) sources, the TT in the sources corresponds again to the VT in the DW. Moreover, since both repositories are updated at the same time, TT in the DW also corresponds to TT in the sources.
4. From bitemporal data sources (not considered in (Jarke et al., 2000)), we could obtain VT and TT.

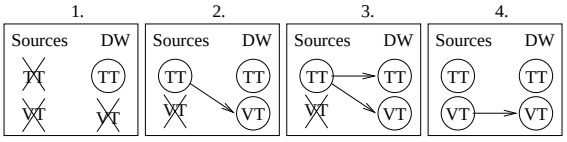

Figure 2: Transformations of time attributes

Out of this four situations, the most common is the second one. Therefore, from here on we will assume the usage of delta files to load the DW, and TT/VT will refer to those of the DW, if not said otherwise.

3.2 Description of Delta Operations

The delta files contain timestamped *insert*s, *update*s and *delete*s of values in the data sources. Let us analyze the effect that each of them has in the DW:

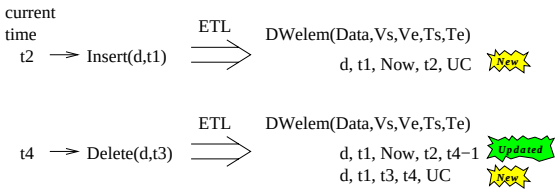

Figure 3: Effect of delta operations

Insert When an insertion (timestamped with the operational TT) is found in a delta file a new bitemporal element is always generated in the DW (as depicted in figure 3). A bitemporal event occurs at a "starting VT" (V_s) and is true until an "ending VT" (V_e). The V_s corresponds to the timestamp in the delta file (i.e. t1). However, V_e is not known at this moment, since data is currently valid in the sources. This is expressed with the special VT value "Now", whose semantics are explained in (Clifford et al., 1997). For example, if we hire an employee, the V_s will be the time when her/his data are introduced in the personnel database, and the V_e will be the value

"Now", until s/he is fired. Insertions initialize the "starting TT" (T_s) to the current time (i.e. t2) and the "ending TT" (T_e) to the value "Until_changed" (UC). As the current time inexorably advances, the value of UC always reflects the current time.

Delete A deletion (also timestamped with the operational TT) generates the logical removal of the existing bitemporal element in the DW. The value of T_e is changed to the current time, when the DW is loaded, minus one (i.e. t4-1). However, as it is shown in figure 3, this is not enough. A new bitemporal element is required, which expresses that from now on we know the V_e, i.e. the timestamp in the delta file (i.e. t3). Therefore, a deletion in the data sources implies an update and an insert in the DW.

Update Without loss of generality, the modification is defined by the deletion of old data immediately followed by the insertion of new values.

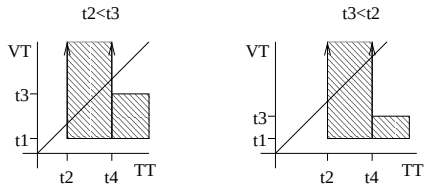

Figure 4: Valid area for VT and TT values

Figure 4 graphically shows the four temporal points of figure 3. Since we cannot delete data that was not previously inserted, it is true that t1<t3. Moreover, the *delete* will always be found strictly after the *insert* in the delta file. Therefore, t2<t4 and for every data there exist two bitemporal rectangles. One of them is open at top, because V_e is "Now". Notice that all time values in the delta files are previous to the load of the DW, which implies that t1<t2 and t3<t4. Since t1<t2, both rectangles have the bottom line below the diagonal of the graphic (which represents events that occur at the same time that they are recorded in the DW). The fact that t3<t4 implies that the closed rectangle also has the top line below the diagonal. However, nothing can be said about the relationship between t2 and t3, because the deletion could happen between the insertion and its load (t3<t2) or after the load of the insertion (t2<t3). Nevertheless, since t2 and t3 are never used in the same temporal element, it only has effect on the position of the rectangles along the TT axis.

4 Temporal Storage Structures

In (Devlin, 1997), the historical nature of the DW is explained, distinguishing two kinds of temporal data: "Transient Data" and "Periodic Data". These DW concepts of (Devlin, 1997) can be found in (Dyreson et al., 1994) for TDBs. In fact, "Transient Data" is a VT relation and "Periodic Data" is a bitemporal relation. Moreover, the representation structure for "Periodic Data" is the "backlog-based" representation of (Jensen et al., 1994). Let be a bitemporal relational schema R have the attributes $A_1, ..., A_n$, its "backlog-based" representation is as follows:

R = {A1,...,An,Vs,Ve,T,Op}

The attributes T and Op are respectively an atomic-valued timestamp attribute containing the TT chronon when the tuple was recorded, and the operation (*insert, update* or *delete*) performed.

In (Kimball et al., 1998) three different storage structures for DWs are described for warehouse inventory situations. The first inventory model is the "Transaction model", which can be also represented in TDBs with the "backlog-based" representation. The second inventory model is the "Delivery status model". In this case one record for each delivery of product to the warehouse is built, and then in this single record, the disposition of all the items in the delivery until they have left the warehouse are tracked. The third inventory model is the "Inventory snapshot model". In (Jensen et al., 1994), an appropriate representation structure for the "Inventory snapshot model" has been explained. This structure is the "tuple timestamped" representation. Let be a bitemporal relational schema R have the attributes $A_1, ..., A_n$, its representation is as follows:

R = {A1,...,An,Vs,Ve,Ts,Te}

The problem of the "tuple timestamped" representation is that the timestamp affects the whole tuple, while changes could affect only one attribute. This is a conceptual and a practical problem at the same time. Firstly, temporal information does not reflect reality (not all attributes change at a time). Moreover, space is wasted (values that do not change are replicated).

To solve this, a temporal structure can be associated with every attribute value, grouping all information about an object within a single tuple. As such, "attribute timestamped" representation is in NF^2. Let a bitemporal relation schema R have the attributes $A_1, ..., A_n$, its representation is as follows:

R = {{[A1,[Vs,Ve],[Ts,Te]],...},...,{[An,[Vs,Ve],[Ts,Te],...},...}

With this representation a tuple is composed of n sets. Each set element is a triple of an attribute value, a VT interval $[V_s, V_e]$, and a TT interval $[T_s, T_e]$.

If we try to physically store such structure, we must reallocate the record every time it is modified, because its size increases and it will not fit in the same place. In a DW, because of its special characteristics, this could and should be avoided. The load of the DW

is massive, and should be as fast as possible to obtain a small "update window". Time should not be wasted reallocating records.

4.1 A New Storage Proposal

As depicted in figure 5, we propose to store the values of every triple $[A_i, [V_s, V_e], [T_s, T_e]]$ (besides an OID that, as discussed in section 2, implicitly contains a lifespan) in a different table. Actually, it is not necessary to define a table for every attribute, but for every set of attributes that share exactly the same temporal behaviour (i.e. a change in one of them always occurs at the same time that the change of the others). If the VT of all attributes changes at once, all changes are due to the same event. Thus, TT will also change at the same time for all them, because the values will be available to be loaded in the DW at one stroke. However, to decide which attributes will be in the same structure a case study is required, based on the events that generate data in the sources.

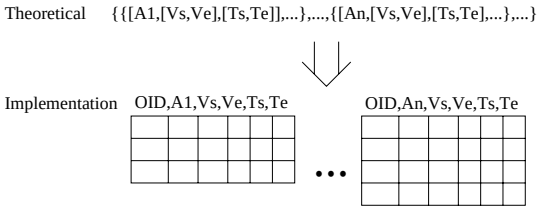

Figure 5: Temporal data representation

Actually, we are proposing a "tuple timestamped" representation where all attributes change at a time, which is a subject and time oriented representation, since each structure 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, we will assume that every table contains at most one attribute. The special case of a table without attribute column will be used to show dynamic classification of the objects. If there exists a record in the table, it means that the instance identified by the OID belongs to the corresponding class. In this way, we can show that a given object is an instance of a class during a VT and TT interval.

In the previous section we have seen that operations will generate three kinds of records in the tables. When a new value is assigned to an attribute (i.e. *insert*), a new record is added with V_e "Now" and T_e UC. This record shows the current value of the attribute in the operational databases. If that value is not valid any more (i.e. *delete* is found in the delta file), the record is updated (assigning a specific value to T_e) to reflect that the information is not current in the DW, and another record is added to indicate the current knowledge of historical information.

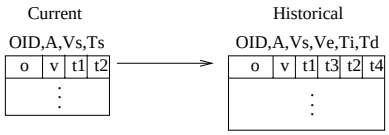

Figure 6: Two kinds of temporal data

As depicted in figure 6, we can use only two different tables to store all three kinds of record. The first kind of record, that reflects current data, only contain two different temporal values (V_s and T_s). Therefore, the other two columns are not necessary for these records, because always contain values "Now" for V_e and UC for T_e. Moreover, after the logical deletion of the value, only four different temporal values are necessary to show the whole historical evolution of the value. We do not actually need to store the two records in figure 3, but only one. Thus, we will have another table to keep the history of the attribute which has six columns: OID, attribute value, the time when the value was introduced in the operational database (the well known V_s), the time when this value was removed (the well known V_e), the time when the *insert* was processed (we will call it T_i), and the time when the *delete* was processed (we will call it T_d). From these four temporal values, we can easily reconstruct the whole history of the value (both records obtained by processing of the corresponding *delete*).

On loading the DW, all we need to do to process an *insert* is add a new record to the *Current* table of the attribute. 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 the *Historical* table. Notice that *Current* table does not contain historical data, so that its size should not be a problem for the search. Moreover, the huge *Historical* table does not need to be scanned in any of both operations. To process a *delete*, all we need to do in the *Historical* table is to append a record.

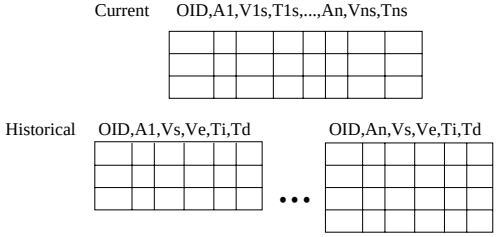

Figure 7: One table for current values, several for historical

Finally, it is important to notice that if an object belongs to a class, all its attributes will have one value in the corresponding *Current* table, despite of the number of historical values of each of them. Therefore, we can have only one *Current* table for the whole set of attributes of a class, independently of the temporal behaviour of each attribute (as depicted in figure 7).

Regarding the requirements enumerated in section 2, time management is clearly improved and internal identifiers empowered. Expressiveness does not actually depend on the storage structure, but on the ability to accurately capture reality. An O-O temporal storage structure should facilitate it. The representation of semantic relationships can be smoothly implemented with OID-OID pairs instead of OID-Attribute pairs.

4.2 Temporal Data Management

An "Integrity Constraint" (IC), also known as business rule, is a property that incorporates extra real-world semantics in a database schema. As it is explained in (Date, 2000), DWs are primarily considered read-only, so data integrity is guaranteed by the ETL module when the database is loaded. Thus, in DWs, it is often assumed that there is no point in declaring ICs. Such is not the case, however. While it is true that the ICs can never be violated, having them in the metadata repository provides a means of telling users what the data means, thereby helping them in their task of formulating queries. It is not a good idea to activate IC checking in the DW, but this should not mean we have to forget them. The DW stores and accesses time-related information, thus temporal ICs should place restrictions on the evolution of data.

EmployeeSalary

OID	Attribute	V_s	V_e
Jordi	1000	2002-7-1	2002-7-20
Jordi	3000	2002-7-10	2002-7-30

Figure 8: Erroneous state of `EmployeeSalary`

For example, "Primary Key"(PK) are no more than a restricted class of IC. Consider the personnel database example in figure 8. In this case, $\{OID, V_s\}$ is not the temporal PK of the relation. Including either V_s, V_e, or both in the PK does not prevent `Jordi` from having two different salaries at a given point in time (as shown in figure 8).

However, there is not such a problem in our structures. (Inmon et al., 1998) explains that an element of time should be added to the operational PK. In our case, *Current* table contains only one tuple per object, so that the only OID will be the PK. In the *Historical* tables, where more than one tuple per object could exist, the OID is not enough. Nevertheless, due to the special characteristics of VT in the DW, the management is not as complex as in the previous example. Since VT is actually TT in the data sources, it inexorably advances, so that we will never find overlapping VT intervals. Moreover, the existence of different VT intervals for the same attribute value of the same object, does not exclude the usage of TT in the PK, because each of this intervals will be registered in the DW at a different TT. Thus, the temporal attribute to be added to the PK can be chosen among the four temporal attributes in the *Historical* table.

Dynamic classification is another example of IC, which expresses the pertainance of an object to a class: the VT intervals of the attribute values of an object must be contained in the VT interval of the classification of the object in the corresponding class. Again, this IC must not be checked because of the nature of the data sources.

Since, as argued in section 1, the DW should not be multidimensional, "On-Line Analytical Processing" (OLAP) tools cannot be directly used to extract information. "Data Marts" are defined from the DW. The queries to feed "Data Marts" are massives, and refer to specific subjects. "Data Marts" always regard a set of measures of a kind of event. Every time one of these events happens, data sources take all measures at one stroke. Therefore in our storage structure, the measures will have the same temporal behaviour, and will be stored in the same structure, if they regard the same subject. Thus, to feed a "Data Mart", we will only need to access one *Historical* table. Furthermore, it is not only necessary to extract temporal data, but also to convert bitemporal data (in the DW) to data structures with only one temporal dimension (in the "Data Marts").

4.3 Comparison of Structures

In this section we are going to compare our storage structure with regard to those storage structures being used nowadays in DWs and TDBs (table 1 summarizes it). Query time, load time and storage space are the criteria used in the comparison:

Criteria	Back-log	Tuple timestamped	Attribute timestamped
Query time	?	?	?
Load time	×	√	√
Storage space	√	√	×

Table 1: Summary table

Query time Our structure is always better to query current values and historical queries regarding one attribute, because it is only necessary to access one table. For multi-attribute historical queries (which would require to access more than one table), all three structures could obtain better query response times in some cases. Nevertheless, we argue that our storage structure is better than the others, because queries involving a set of attributes that have

the same temporal behaviour are the most frequent on feeding the "Data Marts". Moreover, the separation of *Current* table, that facilitates the study of current data, would correspond to the "Operational Data Store", which is a well-known, essential, frequently accessed data repository in the information factory (as reflected in (Inmon et al., 1998)). Depending on the number of attributes, their sizes and update frequencies this storage structure will also be better than the others on solving multi-attribute historical queries. Finding the threshold for this case is out of the scope of this paper and will be tackled as future work.

Load time The "back-log" representation is better than ours, because it only needs to append values. However, it behaves better than "tuple timestamped" (due to the reduced size of the *Current* table, finding the record to update is much faster in our approach) and "attribute timestamp" (our approach neither needs to search historical data, nor reallocate the updated record in the table).

Storage space The "back-log" representation reiteratively replicates values, so that it will need much more storage space, and the "tuple timestamp" representation replicates values that do not change. Therefore, they are wasting space. Nevertheless, "attribute timestamped" representation is better than ours, because it does not replicates OIDs.

5 Conclusions

In this paper, a temporal point of view of a DW has been presented. Since a DW requires historic information, we identify the two existing orthogonals temporal dimensions in a DW: the valid time dimension and the transaction time dimension. In TDBs, temporal models and temporal management for time evolving data have been widely studied. Particularly, we concentrate in O-O temporal data models, which, besides temporal dimensions, facilitate integration and subject-orientation characteristics to the DW.

The correspondences between temporal attributes in the data sources and those in the DW have been analized. Moreover, a new subject and time oriented storage structure has been proposed and compared with the existing ones.

REFERENCES

Abelló, A., Oliva, M., Samos, J., and Saltor, F. (2000). Information System Architecture for Data Warehousing from a Federation. In *Proc. of the 3rd Int. Workshop on Engineering Federated Information Systems (EFIS'00)*, pages 33–40. IOS Press.

Bertino, E., Ferrari, E., and Guerrini, G. (1997). T_Chimera: A temporal object-oriented data model. *Theory and Practice of Object Sytems*, 3(2):103–125.

Böhlen, M. H., Busatto, R., and Jensen, C. S. (1998). Point-Versus Interval-Based Temporal Data Models. In *Proc. of the 14th Int. Conf. on Data Engineering (ICDE'98)*, pages 192–200. IEEE Computer Society.

Clifford, J., Dyreson, C., Isakowitz, T., Jensen, C. S., and Snodgrass, R. T. (1997). On the Semantics of "Now" in Databases. *ACM Transactions on Database Systems*, 22(2):171–214.

Date, C. J. (2000). *An Introduction to Database Systems*. Addison Wesley, seventh edition.

Devlin, B. (1997). Managing Time in the Data Warehouse. *InfoDB*, 11(1):7–11.

Dyreson, C. E., Grandi, F., Käfer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J. F., Sarda, N. L., Scalas, M. R., Segev, A., Snodgrass, R. T., Soo, M. D., Tansel, A., Tiberio, P., and Wiederhold, G. (1994). A Consensus Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 23(1):52–64. http://www.cs.auc.dk/˜ csj/Glossary.

Inmon, W. H., Imhoff, C., and Sousa, R. (1998). *Corporate Information Factory*. John Wiley & Sons.

Jarke, M., Lenzerini, M., Vassiliou, Y., and Vassiliadis, P., editors (2000). *Fundamentals of Data Warehousing*. Springer-Verlag.

Jensen, C. S., Soo, M. D., and Snodgrass, R. T. (1994). Unifying Temporal Data Models Via a Conceptual Model. *Information Systems*, 19(7):513–547.

Kimball, R., Reeves, L., M.Ross, and Thornthwaite, W. (1998). *The Data Warehouse lifecycle toolkit*. John Willey & Sons.

Pedersen, T. B. and Jensen, C. S. (1998). Research Issues in Clinical Data Warehousing. In *Proc. of the 10th Int. Conf. on Statistical and Scientific Database Management (SSDBM'98)*, pages 43–52. IEEE Computer Society.

Saltor, F., Castellanos, M., and García-Solaco, M. (1991). Suitability of Data Models as Canonical Models for Federated DBs. *ACM SIGMOD Record*, 20(4):44–48.

Snodgrass, R. T. and Ahn, I. (1986). Temporal Databases. *IEEE Computer*, 19(9):35–42.

Steiner, A. and Norrie, M. C. (1997). A Temporal Extension to a Generic Object Data Model. Technical Report TR-15, Time Center.

Vassiliadis, P. (2000). Gulliver in the land of data warehousing: practical experiences and observations of a researcher. In *Proc. of the 2nd Int. Workshop on Design and Management of Data Warehouses (DMDW'00)*. CEUR-WS (http://www.ceur-ws.org).

Wu, Y., Jajodia, S., and Wang, X. S. (1998). Temporal Database Bibliography Update. In *Temporal Databases: Research and Practice*, pages 338–366. Springer-Verlag.