The Data Warehouse: An Object-Oriented Temporal Database

Alberto Abelló and Carme Martín

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

Abstract. The aim of this paper is to bring together two research areas, i.e. "Data Warehouses" and "Temporal Databases", involving representation of time. In order to achieve this goal, data warehouse and temporal database research results have been surveyed. Looking at temporal aspects within a data warehouse, more similarities than differences between temporal databases and data warehouses have been found. Therefore, this paper is focussed on how contributions of the temporal database research could benefit data warehouses.

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 [JLVV00]. A recent study of research issues in the field is in [Vas00]. Analyzing this study a clear practical business origins 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 [WJW98], have produced important and consolidated results in this field. In addition to the common points we will explain further in detail, the different approaches of these two areas are another interesting reason to relate them. Thus, the aim of this paper is to present how contributions of the "Temporal Database" (TDB) research could benefit "Data Warehouse" (DW) area.

The affinity between both concepts (i.e. DW and TDB) may not be obvious. However, time references are essential in management business decisions, and the dissection of both definitions shows their closeness. As defined in [IIS98], 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 [SA86] as a database that supports "Valid time" (i.e. the time when the fact becomes effective in reality), or "Transaction time" (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. We consider the former

accepted definition of DW could be rewritten in terms of well-established latter temporal concepts. In [IIS98], we can see that "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 "Valid Time" (VT) in [DGK+94] 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 "Transaction Time" (TT), defined in [DGK+94] as the time when the fact is current in the 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 [PJ98], which asks DW systems for support of advanced temporal concepts. Besides, [YW00] explains how to define temporal views over non-temporal relations. We go further and show how TDB results could contribute to specific data warehousing research subjects. We do not show how to feed the DW, but outline the benefits of using a TDB for data warehousing.

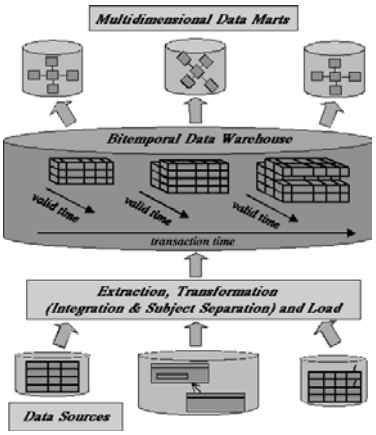

Fig. 1. A Data Warehouse as a Bitemporal Database

The bitemporal DW definition shows how the existence of the temporal dimension in a DW is inferred from its definition. Then, the next sections point out how this temporal nature of the DW leads to the usage of different aspects already studied in the field of TDBs. Next section elaborates on some general temporal issues that should be adapted to data warehousing. Section 3 explains how temporal languages could be used in the DW. Section 4 describes how temporal storage management could be used in a DW environment. Section 5

introduces "Object-Oriented" (O-O) temporal concepts in the data model of the DW. Finally, section 6 concludes the paper.

2 Temporal Issues Relevant to Data Warehouses

In the literature, we find papers proposing the modeling of DWs as multidimensional databases. However, as argued in [AOSS00], the structure of multidimensional star shape schemas is too rigid and absolutely query oriented, while the design of the central, corporate DW should be data driven. Since the DW should not be multidimensional, "On-Line Analytical Processing" (OLAP) tools cannot be directly used to extract information, and a flexible multi-purpose query language is still needed. This query language should facilitate the storage and retrieval of time-varying information.

Multidimensional modeling perfectly suits for small departamental DWs. Thus, from the DW, some smaller, customized data structures, known as "Data Marts" are built (as it is shown in figure 1). Since the DW is bearing them, for the sake of performance, the "Data Marts" could only contain a partial history of data, or data current at a given time. Therefore, 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"). In this sense, it can be very helpful to apply the VT and the TT snapshot operations of TDBs, explained in [JSS92]. The VT snapshot operation is applied to extract, from a bitemporal relation, the tuples valid at a given time and the TT snapshot operation is applied to extract the tuples current at a given time.

First of all, as stated in the previous definition, the DW system should manage both temporal dimensions, i.e. VT and TT. However, not all objects will have such bitemporal nature. It is important to notice that data in the DW comes from independent heterogeneous sources (depicted in figure 1), whose data are "integrated". Thus, some of these sources will probably provide only TT for their instances. Maybe, for some other instances, no temporal information needs to be kept at all (for example, 1 Euro equals 166.386 Pesetas forever). The DW must allow the possibility of defining different kinds of temporal objects and attributes as in [BFG97].

Moreover, independently of the temporal dimension, different time structures should also be allowed (i.e. instant -time point on an underlying time axis-, interval -the time between two instants-, set of instants, and set of intervals) as in any other TDBs, so that the different possibilities in reality can be represented. It could also offer the possibility of defining calendars (i.e. human interpretations of time), which would ascribe different meanings to temporal values. [Kim96] considers such calendars (represented there in terms of attributes of a relational table) an essential for analysis tasks, to be able to compare different periods of time based on their characteristics.

A timestamp is a time value associated with some object or attribute value. A non-decomposable time interval of some fixed, minimal duration is a chronon. The size of each chronon in a timestamp interpretation is called granularity.

Also due to the existence of different data sources, DW systems should allow the usage of multiple granularities. In general, during the integration process, we will not find that all sources are defined at the same temporal granularity (one could collect data daily, another one monthly, and so on). If we chose the coarser of the available granularities to integrate data, we would lose detail (which is obviously not desirable from the point of view of the analysts, who will demand information as detailed as possible).

In TDBs, a database which allows facts to be expressed in terms of different granularities is called a TDB with multiple granularities. For example, an employee is hired from a date to a date (year-month-day) but s/he will report her/his activity from an hour to another hour every day. Moreover, s/he will work only from Monday to Friday (a business-week), in spite of s/he has been hired for a complete week. The notion of granularity system is an excellent TDB tool to solve the DW problem of different granularities integration. Formally, in TDBs, a granule is a set of time instants perceived as a nondecomposable temporal entity when used to describe a phenomenon or when used to timestamp a set of data. A granule can be composed of a single instant, a set of contiguous instants (time intervals), or even a set of non-contiguous instants. In [BJW00], an algebra for symbolic manipulation of granularities has been explained. For example, the relationship: day "groups into" business-week, expresses that a day or a set of days are a subset of a business-week. The use of this algebra could be a well DW integration solution.

3 Temporal Data Manipulation

Data manipulation is a really problematic question in the DW area due to the huge volume of data to be managed. The updates are exclusively performed inside the "update window" (during which queries are forbidden to avoid interferences). This allows to consider the DW as a read-only database, as long as users are not allowed to query during the update window. Temporal update techniques, as it is shown in [JSS94], should be considered at this point, as well as temporal constraints (business rules). Another important element of temporal languages, with associated specific research, are temporal constraints. An integrity constraint is a property that incorporates extra real-world semantics in a database schema. As it is explained in [Dat00], DWs are primarily considered read-only, so data integrity is checked when the database is loaded (or refreshed). Thus, in DWs, it is often assumed that there is no point in declaring integrity constraints in the logical schema. Such is not the case, however. While it is true (if the database is genuinely read-only) that the constraints can never be violated, declaring them provide a means of telling users what the data means, thereby helping them in their task of formulating queries. Moreover, the DW is not really read-only, but data are added to it. TDBs store and access time-related information, thus temporal integrity constraints can place restrictions on the evolution of data in time. Moreover, TDB integrity constraint checking methods, as [Mar01], can be used.

Employee1

Name	Salary	VT_{start}	VT_{end}
Jordi	1000	2002-7-1	2002-7-20
Jordi	3000	2002-7-10	2002-7-30

Fig. 2. Erroneous state of `Employee1`

Dependencies are no more than a restricted class of integrity constraints. [IIS98] explains that adding an element of time to the operational key is a common thing to do. However, the simple addition of the timestamp attributes does not always convert a non-temporal key to a temporal key. "Temporal Functional Dependencies" (TFDs), as explained in [Wij99], would allow us to formally reason about whether it is necessary/effective or not to add that element. A "Functional Dependency" (FD), that is valid in the current data, may no longer be valid in the corresponding temporal data if the traditional definition of FDs is used without change. Consider the previous personnel database example without a TDB. In that case, we added a starting and ending valid time attributes. The original primary key is not, by itself, a primary key of the temporal relation. Including either $\texttt{VT}_{start}$, $\texttt{VT}_{end}$, or both in the primary key does not prevent `Jordi` from having two salaries at a given point in time (as shown in figure 2).

Furthermore, the DW problem of the use of null values for unknown end time has been widely studied in TDBs [CDI+97]. A temporal event -an instantaneous fact, something occurring at an instant- happens in a starting VT and is true until an ending VT. Sometimes, the ending VT is not a given value, is something that indicates the event is currently valid. This is expressed with the special VT value "Now". For example, if we hire an employee for a permanent job, the starting VT will be the initial date of the contract and the ending VT will be the value "Now". When an event is inserted or deleted, its VT, supplied by the user, is transformed into a bitemporal element, adding TT, supplied by the DBMS. Insertions initialize the starting TT to the current time and the ending TT to the value "Until_changed". As the current time inexorably advances, the value of "Until_changed" always reflects the current time. Deletions change the ending TT "Until_changed" to the current time when it is performed.

The volume of data is also a problem for the unlimited growing of the DW. Some aging policy must be defined. For example, data is usually stored at coarser granularities the older they are. This unearths another important temporal feature, known as coalescing, which means (in relational terms) obtaining one tuple from two tuples with exactly the same values for each and every attribute, when their timestamps are adjacent (or overlapping) in time. That is, coalescing is similar to duplicate elimination, which would save lots of space in a snapshot DW. Coalescing should not be confused with the multidimensional operation "Roll-up". The former eliminates redundant information, while the latter generates less detailed data from the more detailed (aggregating or summarizing).

Employee1

Name	Salary	VT_{start}	VT_{end}
Jordi	1000	2002-7-1	2002-7-10
Jordi	2000	2002-7-11	2002-7-20
Jordi	3000	2002-7-21	2002-7-30

Employee2

Name	Department	VT_{start}	VT_{end}
Jordi	LSI	2002-7-1	2002-7-15
Jordi	AC	2002-7-16	2002-7-30

Fig. 3. `Employee1` and `Employee2` relations

Consider, for example, a personnel TDB with two VT relations of employees, as in figure 3. The definition of VT relation, automatically, includes a starting and ending VT. The former (`Employee1`) has the attributes: name and salary. The latter (`Employee2`) has the attributes: name and department. We want to know what is the salary and the department history of our employees. In this case, the simple fact of having VT relations saves the usage of additional TSQL2 predicates for interval comparison (i.e. during, after, before, etc. defined in [All83]) to formulate the query. Thus, in a TDB, using TSQL2, this query is just a temporal join, like that in left hand side of figure 4. The simple formulation of this query is due to `Employee1` and `Employee2` are VT relations. The answer to this query, automatically, shows the VT history (right hand side figure 4).

```
SELECT Employee1.Name, Salary, Department
FROM Employee1, Employee2
WHERE Employee1.Name = Employee2.Name
```

Name	Salary	Dept	VT_{start}	VT_{end}
Jordi	1000	LSI	2002-7-1	2002-7-10
Jordi	2000	LSI	2002-7-11	2002-7-15
Jordi	2000	AC	2002-7-16	2002-7-20
Jordi	3000	AC	2002-7-21	2002-7-30

Fig. 4. Temporal join of `Employee1` and `Employee2`

Consider the previous personnel database example and the same earlier query. Without a TDB, we must add a starting and ending VT attributes. Using SQL, the formulation of the query is the union of the four queries (one for each intersection pattern of two intervals) that generate the four tuples in figure 4.

4 Temporal Storage Management

Let be a bitemporal relational schema R have the attributes $A_1, ..., A_n$, the most common three alternatives used in TDB for its representation are:

Backlog-based Attributes V_s, V_e, T and Op are atomic-valued timestamp attributes containing a starting and ending VT chronons, the TT chronon when the tuple was recorded, and the operation (insertion or deletion) performed. $R = (A_1, ..., A_n, V_s, V_e, T, Op)$

Tuple timestamped Attributes T_s, T_e, V_s and V_e are atomic-valued timestamp attributes containing a starting and ending TT chronons and a starting and ending VT chronons. $R = (A_1, ..., A_n, T_s, T_e, V_s, V_e)$

Attribute timestamped With this representation a tuple is composed of n sets. $R = (\{[A_1, [T_s, T_e], [V_s, V_e]], ..\}, .., \{[A_n, [T_s, T_e], [V_s, V_e]], ..\})$. Each set element is a triple of an attribute value, a TT interval $[T_s, T_e]$, and a VT interval $[V_s, V_e]$.

These general storage structures can be improved by considering the specific knowledge of temporal behaviour in DW. An specific bitemporal storage proposal for DW is found in [AM03]. The application of this storage structure for all different types of data sources has been explained in [MA03].

In [JS99], two basic storage management approaches for TDBs are explained. The "integrated" approach consists in modifying or extending the internal modules of a DBMS to support time-varying data. The "stratum" approach converts

temporal query language statements into conventional statements executed by an underlying DBMS. While the latter approach is more realistic now, the former approach ensures maximum efficiency and it should be the selected approach for DWs. TDBs go further than DWs in this issue. There has been a vast amount of work in storage structures and access methods for temporal data, as well as some temporal DBMS prototypes [Böh95]. Specifically, in the "integrated" approach for bitemporal databases many temporal indexing strategies are available, as it is shown in [ST99], a survey of this field.

5 Object-Oriented Temporal Data Models

Time is an omnipresent element in analysis tasks, and a difficult element to handle. We could find several temporal extensions of the E/R model that would ease the modeling of the huge, central DW of the company, in [GJ99]. However, going again to the definition in [IIS98], we observe that a DW is "Integrated" and "Subject-oriented". In [SCG91], it is explained the importance of a semantically rich data model to overcome semantic heterogeneities in the data sources. Thus, it is argued that O-O models should be used for integration. The DW schema would be equivalent to the "Federated Schema" (defined in [SL90]), where we integrate the data sources, and from where we define the multidimensional query oriented user views. Therefore, it should keep as much semantics as possible.

Semantic richness of O-O models, not only facilitates integration, but also helps analysts to understand the real meaning of data. Moreover, it facilitates 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.

First of all, 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 related to a TT. Thus, this could be implicitly offered by the system. However, as pointed out before, 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 [BFG97] or [SN97].

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 [BBJ98], 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. [SN97] 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). Finally, as already said, the data model of the DW should support multiple granularities, as in [MBFG99], for VT and lifespan.

An important characteristic of O-O models is the definition of methods for objects and classes. In this case, it could be used to facilitate the implementation of temporal associations and attributes. All we need is to define a method with temporal parameters that returns the value of the relationship at the given time. Thus, any attribute or relationship could be viewed as temporal methods some returning constants, others returning one object or another depending on the time parameters. This generalizes "mapping functions" in [EK01]. For example, we could implement the gross domestic product of Germany as follows:

```
class state {
  attribute string name;
  attribute Dictionary<int,real> GDP;

  real GDP(in int year) {
    if (year<1990 && name="Germany")
      return (SELECT s FROM state WHERE
        name="Western Germany").GDP(year)
        +(SELECT s FROM state WHERE
        name="Eastern Germany").GDP(year);
    else return GDP.lookup(year);
  }
```

6 Conclusions and Future Research

In this paper, we have brought together contributions from both research areas (i.e. Data Warehousing and Temporal Databases), and have shown how close they are. A TDB point of view of a DW has been presented. Since a DW requires historic information, it must have a temporal dimension. Specifically, we identify the two existing orthogonals temporal dimensions in a DW: the valid time dimension and the transaction time dimension. In TDBs, temporal models, temporal languages and access methods for time evolving data have been widely studied. We conclude that this knowledge can and should be used in DWs in two directions: firstly, offering tools to ease the design and query; secondly, providing special storage and access paths to improve the performance. Particularly, we concentrate in the Object-Oriented temporal data models, which, besides temporal dimensions, facilitate integration and subject-orientation to the DW.

Acknowledgements

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

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[AM03] Alberto Abelló and Carme Martín. A Bitemporal Storage Structure for a Corporate Data Warehouse. In *Proc. of the 5th Int. Conf. on Enterprise Information Systems (ICEIS 2003)*, pages 177–183. Instituto Politècnico de Setúbal, 2003.

[AOSS00] Alberto Abelló, Marta Oliva, José Samos, and Fèlix Saltor. 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, 2000.

[BBJ98] Michael H. Böhlen, Renato Busatto, and Christian S. Jensen. 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, 1998.

[BFG97] Elisa Bertino, Elena Ferrari, and Giovanna Guerrini. T_Chimera: A temporal object-oriented data model. *Theory and Practice of Object Sytems*, 3(2):103–125, 1997.

[Böh95] Michael H. Böhlen. Temporal Database Systems Implementations. *ACM SIGMOD Record*, 24(4):53–60, 1995.

[BJW00] Claudio Bettini, Sushil Jajodia, and X. Sean Wang. *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Springer-Verlag, 2000.

[CDI+97] James Clifford, Curtis Dyreson, Tomás Isakowitz, Christian S. Jensen, and Richard T. Snodgrass. On the Semantics of "Now" in Databases. *ACM Transactions on Database Systems*, 22(2):171–214, 1997.

[Dat00] C. J. Date. *An Introduction to Database Systems*. Addison Wesley, seventh edition, 2000.

[DGK+94] Curtis E. Dyreson, Fabio Grandi, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, John F. Roddick, Nandlal L. Sarda, Maria Rita Scalas, Arie Segev, Richard T. Snodgrass, Mike D. Soo, Abdullah Tansel, Paolo Tiberio, and Gio Wiederhold. A Consensus Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 23(1):52–64, 1994. http://www.cs.auc.dk/~ csj/Glossary.

[EK01] Johann Eder and Christian Koncilia. Changes of Dimension Data in Temporal Data Warehouses. In *Proc. of the 3rd Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2001)*, volume 2114 of *LNCS*, pages 284–293. Springer, 2001.

[GJ99] Heidi Gregersen and Christian S. Jensen. Temporal Entity-Relationship Models - A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):464–497, 1999.

[IIS98] William H. Inmon, Claudia Imhoff, and Ryan Sousa. *Corporate Information Factory*. John Wiley & Sons, 1998.

[JLVV00] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis, editors. *Fundamentals of Data Warehousing*. Springer-Verlag, 2000.

[JS99] Christian S. Jensen and Richard T. Snodgrass. Temporal Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):36–44, January/February 1999.

[JSS92] Christian S. Jensen, Michael D. Soo, and Richard T. Snodgrass. Extending Normal Forms to Temporal Relations. Technical Report TR-92-17, Computer Science Department. University of Arizona, 1992.

[JSS94] Christian S. Jensen, Michael D. Soo, and Richard T. Snodgrass. Unifying Temporal Data Models Via a Conceptual Model. *Information Systems*, 19(7):513–547, 1994.

[Kim96] Ralph Kimball. *The Data Warehouse toolkit*. John Wiley & Sons, 1996.

[MA03] Carme Martín and Alberto Abelló. A Temporal Study of Data Sources to Load a Corporate Data Warehouse. In *5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2003. To appear.

[Mar01] Carme Martín. Analyzing Temporal Integrity Constraints to Obtain the Minimum Number of Transition Rules. Technical Report LSI-01-52-R, Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 2001. http://www-lsi.upc.es/~ martin/papers/report01.ps.zip.

[MBFG99] Isabella Merlo, Elisa Bertino, Elena Ferrari, and Giovanna Guerrini. A Temporal Object-Oriented Data Model with Multiple Granularities. In *Proc. of the 6th Int. Workshop on Temporal Representation and Reasoning (TIME'99)*, pages 73–81. IEEE Computer Society, 1999.

[PJ98] Torben B. Pedersen and Christian S. Jensen. 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, 1998.

[SA86] Richard T. Snodgrass and Ilsoo Ahn. Temporal Databases. *IEEE Computer*, 19(9):35–42, 1986.

[SCG91] Fèlix Saltor, Malú Castellanos, and Manolo García-Solaco. Suitability of Data Models as Canonical Models for Federated DBs. *ACM SIGMOD Record*, 20(4):44–48, 1991.

[SL90] Amit P. Sheth and James A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SN97] Andreas Steiner and Moira C. Norrie. A Temporal Extension to a Generic Object Data Model. Technical Report TR-15, Time Center, 1997.

[ST99] Betty Salzberg and Vassilis J. Tsotras. A Comparison of Access Methods for Temporal Data. *ACM Computing Surveys*, 31(2):158–221, 1999.

[Vas00] Panos Vassiliadis. 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), 2000.

[Wij99] Jef Wijsen. Temporal FDs on Complex Objects. *ACM Transactions on Database Systems*, 24(1):126–176, 1999.

[WJW98] Yu Wu, Sushil Jajodia, and X. Sean Wang. Temporal Database Bibliography Update. In *Temporal Databases: Research and Practice*, pages 338–366. Springer-Verlag, 1998.

[YW00] Jun Yang and Jennifer Widom. Temporal View Self-Maintenance in a Warehousing Environment. In Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl, and Torsten Grust, editors, *Proc. of the 7th Int. Conf. on Extending Database Technology (EDBT'00)*, volume 1777 of *LNCS*, pages 395–412. Springer, 2000.