



Capturing Delays and Valid Times in Data Warehouses—Towards Timely Consistent Analyses

ROBERT M. BRUCKNER

bruckner@ifs.tuwien.ac.at

A MIN TJOA

tjoa@ifs.tuwien.ac.at

Institute of Software Technology & Interactive Systems, Department of Information & Software Engineering, Vienna University of Technology, Favoritenstr. 9/188, A-1040 Wien, Austria

Abstract. Real-world changes are generally discovered delayed by computer systems. The typical update patterns for traditional data warehouses on an overnight or even weekly basis increase this propagation delay until the information is available to knowledge workers. Typically, traditional data warehouses focus on summarized data (at some level) rather than detailed data.

For active data warehouse environments, detailed data about entities is required for checking the data conditions and triggering actions to automatize routine decision tasks. Hence, keeping data current (by minimizing the latency from when data is captured until it is available to knowledge workers) and consistent in that context is a difficult task.

In this paper we present an approach for modeling conceptual time consistency problems and introduce a data model that deals with timely delays. It supports knowledge workers in finding out, why (or why not) an active system responded to a certain state of the data. Therefore, the model enables analytical processing of detailed data (enhanced by valid time) based on a knowledge state at a specific time. All states that were not yet known by the system at that point in time are consistently ignored. This enables timely consistent analyses by considering that the validity of detailed data and aggregates can be restricted to time intervals only, due to frequent updates and late-arriving information.

Keywords: data modeling, temporal data models, propagation delays, time semantics, data warehouse

1. Introduction

The amount of information available to large-scale enterprises is growing rapidly. New information is being generated continuously by operational sources such as order processing, inventory control, and customer service systems. In order to support efficient analysis and mining of such diverse, distributed information, a *data warehouse* (DWH) collects data from multiple, heterogeneous sources and stores integrated information in a central repository. The DWH needs to be updated periodically to reflect source data updates. The operational source systems collect data from real-world events captured by computer systems. The observation of these real-world events is characterized by a delay. This so-called *propagation delay* is the time interval it takes for a monitoring (operational) system to realize an occurred state change. The update patterns (daily, weekly, etc.) for data warehouses and the data integration process (extract-transform-load) result in increased propagation delays.

While operational systems are among other things designed to meet well-specified (short) response time requirements, the focus of data warehouses is generally the strategic analysis of data integrated from heterogeneous systems (Inmon, 1996). The data integration process can be very complex and covers the acquisition, extraction, transformation (staging area),

and loading of the data into the DWH. Traditionally, there is no real-time connection between a DWH and its data sources, because the write-once read-many decision support characteristics would conflict with the continuous update workload of operational systems and result in poor response time. Data loading is done during frequent update windows (e.g. once a week, every night), when the analysis capabilities of DWHs are not affected.

Consequently, up till recently, *timeliness* requirements (the relative availability of data to support a given process within the timetable required to perform the process) were restricted to mid-term or long-term (English, 1999). W.H. Inmon, known as the founder of data warehousing, cites time variance (Inmon, 1996) as one of four silent characteristics of a DWH. Timeliness can be viewed as an aspect of data quality, which has a strong influence on the delay until a system realizes a certain state of the data. While a complete, real-time enterprise DWH might still be the panacea, there are some approaches to enable DWHs to react “just-in-time” (Westerman, 2000) to changing customer needs, supply chain demands, and financial concerns.

Data warehouse and business intelligence applications are separate standalone decision-support systems used for strategic planning and decision-making. As these applications have matured, however, it has become apparent that the information and analyses they provide are vital to tactical day-to-day decision-making, and many organizations can no longer operate their businesses effectively without them. Consequently, there is a trend towards integrating decision processing into the overall business process. The advent of e-business is also encouraging this integration because organizations need to react much faster to changing business conditions in the e-business world.

New concepts, like active data warehousing (Schrefl and Thalhammer, 2000), try to combine active mechanisms based on ECA (event-condition-action) rules known from active databases (Dayal, 1988; ACT-NET, 1996) with the integrated analysis capabilities of DWH environments to extend (passive) systems with reactive capabilities: An *active data warehouse (ADWH)* is event driven, reacts in a timeframe appropriate to the business needs, and makes tactical decisions or causes operational actions rather than waiting to produce periodic reports (Brobst and Ballinger, 2000). Therefore, the design of an ADWH has to consider technical aspects (scalability, high availability, frequent (just-in-time or continuously) data loading, mixed workload, etc.) as well as the integration of active mechanisms, which have to deal with two sorts of *propagation delays* in DWH environments mentioned earlier: (a) delays in capturing real-world events by operational systems, and (b) delays in loading and integrating data into the DWH.

Figure 1 demonstrates that the typical update pattern for traditional DWHs on a weekly or even monthly basis enlarges propagation delays until information is available to knowledge workers. However, in the case of traditional DWHs any significant delay in the recognition of real-world events may result in a number of additional considerations needing to be taken into account, such as the following two problems:

- *data integration*. Aggregates have to be updated, because the new records will change counts and totals of the prior history.
- *analytical processing*. Historical analysis results are not repeatable any longer, if additional information regarding that time period is integrated delayed (numbers and summaries will change unexpectedly from the user’s perspective).

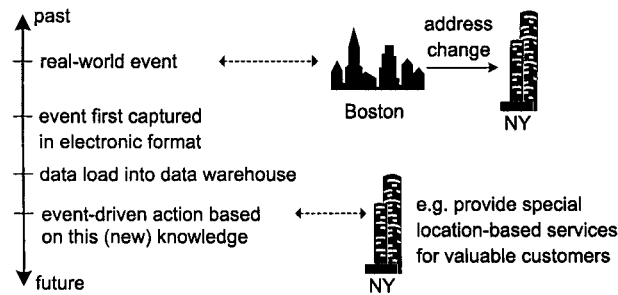


Figure 1. Delayed discovery of real-world changes.

Traditional DWHs providing on-line analytical processing (OLAP) typically focus on summarized data (at some level) rather than detailed data. These considerations do not entirely apply to ADWH environments, where it is important to have enough detailed data about an individual entity to form an accurate picture of its relationship to an organization. The complexity of analysis and data access required, places the typical ADWH query outside the scope of most transaction-processing systems and addresses a type of automated decision-making that has rarely been accomplished by passive DWHs (Thalhammer et al., 2001).

Consequently, keeping data current and consistent in that context is difficult. Although higher refresh frequencies shrink propagation delays, storing data timely consistent becomes even more important, because business rules can be formulated according to the ECA structure of triggers in ADWHs. However, delays in the execution of such rules can cause unexpected and undesired side-effects (Roddick and Schrefl, 2000).

We will concentrate on the development and examination of a schema model that comes along with delays and allows a timely consistent representation of information. This enables consistent views for analysis purposes considering that *information validity* of detailed data (or even aggregates) is typically restricted to time periods, because of frequent updates. It is important to note that the aim of our approach is not necessarily to shrink delays. It is a pragmatic approach which ensures that the side-effects of propagation delays and late-arriving data are reduced in ADWHs and enables meaningful analyses across time even when data change (traceability). Existing models lack built-in mechanisms for handling change and time. We will introduce overlapping valid time intervals for information stored in data warehouses. This supports knowledge workers in finding out, why (or why not) an active system responded to a certain state of the data at a particular point in time. Therefore, the model enables analytical processing of detailed data (enhanced by valid time) based on a knowledge state at a specified instant of time. All states that were not yet comprehensible to the system at that point in time are consistently ignored.

The remainder of this paper is organized as follows. Section 2 considers research issues and related work. The motivation (minimizing latency for analytical environments) is given in Section 3, followed by the explanation of basic temporal data structures (Section 4). Time consistency is introduced in Section 5 and described from different viewpoints (conceptual, logical, and physical). Section 6 describes application fields (in particular ADWH), followed

by a short case study and an in-depth evaluation in Section 7. Finally, we give a conclusion in Section 8.

2. Research issues and related work

The notion of time is fundamental to our existence and an important aspect of real-world phenomena. We can reflect on past events and on possible future events, and thus reason about events in the domain of time. In many models, time is an independent variable that determines the sequence of states of a system. There are two different temporal characterizations of the information appearing in a DWH: one is the classical description of the time instant when a given fact occurred; the other represents the instant when the information is actually intelligible to the system. This distinction, implicit and usually not critical in on-line transactional processing (OLTP) applications, is of particular importance for data warehouses, where it can be very useful (or even necessary) in determining and analyzing what was the situation in the past, knowing only the information available at a given point in time.

Since the approach of modeling time consistency is related to data quality and timeliness, we give a brief overview concerning important issues. Quality management in information processing is not nearly as established as it is in manufacturing (Ballou and Tayi, 1999; English, 1999). Since DWHs store gigabytes up to terabytes of data, a manual quality control is often not feasible. Instead, human interaction should be reduced to a minimum by considering data quality issues during the design of data warehouse environments. In data quality research, there are two well-known projects, namely MIT's *Total Data Quality Management (TDQM)* (Wang, 1998) and the ESPRIT project *Foundations of Data Warehouse Quality (DWQ)* (Jarke et al., 1998).

- TDQM claims to establish a theoretical foundation of data quality. Based on the enterprise philosophy of Total Quality Management, a so-called TDQM cycle is defined consisting of four phases: quality planning, measurement, analysis, and improvement. *Timeliness* is considered as a so-called quality dimension of the contextual data quality. Examples for other dimensions include accuracy, relevancy and completeness.
- DWQ is tailored to the data-warehousing world and considers not only the quality of the warehouse data, but also the warehouse architecture itself. Although DWQ follows a holistic approach, it disregards the need for a formal foundation of data quality concepts.

L. English defines *timeliness* as the time from when a fact is first captured in an electronic format and when it is actually available to a knowledge worker who needs it (English, 1999). For data captured in a source system and uploaded to a central DWH over the weekend, the propagation delay can be remarkable. This is unacceptable for implementations tightly integrated into business processes (like ADWHs). Late-arriving fact and dimension records can complicate this situation (Kimball, 2000), because they change the counts and totals of the prior history. Some industries, like health care, have to deal with huge numbers of late-arriving records (Bruckner and Tjoa, 2001).

Conventional database systems capture only a single logical state of the modeled reality (usually the most current). Although the contents of the database continues to change as

new information is added, these changes are viewed as modifications to the state, with the old, out-of-date data being discarded after the transaction commits.

In contrast a *temporal database* provides support for past, current, or even future data and allows sophisticated queries over time to be stated (Zaniolo et al., 1997). Research in temporal databases has grown immensely in recent years. Numerous temporal data models and query languages have been proposed and surveyed, e.g. in Gregersen and Jensen (1999). In particular, *transaction time* and *valid time* have been proposed (Jensen and Dyreson, 1998) and investigated (Torp et al., 2000; Jensen and Snodgrass, 1999).

Only a few DWH research projects paid attention to those results and concentrated for instance on the shortcomings of star schemas in the context of slowly changing dimensions (Bliujute et al., 1998). They conclude that DWHs based on state-oriented data allow easier analytical processing and even better query performance than observed in DWHs based on event data. Our formal approach to managing time consistency also follows this state-oriented approach.

Although data warehousing has been studied extensively by the database research community (e.g. Widom, 1995; Chaudhuri and Dayal, 1997), most of the previous research ignores the *temporal aspects* of DWHs. Users of a data warehouse may not only be interested in the most up-to-date information, but also in the history of how source data has evolved. Therefore, real-world DWHs are often temporal, but their temporal support is implemented in an ad hoc manner that is difficult to automate. In practice, many operational source systems are nontemporal, i.e., they store only the current state of their data, not the complete history. *Temporal data warehouses* address the issue of supporting temporal information efficiently in data warehousing systems. Keeping them up-to-date is complex, because temporal views may need to be updated not only when source data is updated but also as time progresses, and these two dimensions of change interact in subtle ways. Yang and Widom present efficient techniques (e.g. temporal view self-maintenance) for maintaining temporal DWHs without disturbing source operations (Yang, 2001; Yang and Widom, 2000). A related challenge is supporting large-scale temporal aggregation operations in DWHs (Yang and Widom, 2001). However, most of the research to date has concentrated on performance issues, rather than higher-level issues, such as conceptual modeling (Pedersen et al., 2001).

Recent research in *active databases* (Widom and Ceri, 1996) strives to extending the power of active rules (for responding to changes to the database and external events) to trigger on complex patterns of temporal events (Dittrich and Gatzju, 2000). An important issue is the accommodation of delays, which has been investigated for (business) transactions in temporal active databases (Roddick and Schrefl, 2000; Obermair and Schrefl, 2000). The conclusion is that *temporal faithfulness* for transactions has to be provided, which preserves the serialization order of a set of business transactions. Commit-time timestamping guarantees transaction-consistent pictures of past states of a database. However, the timestamps assigned to these transactions are generated by the database system and do not bear application semantics.

This paper concentrates on delays in the young discipline of *active data warehouses* (Schrefl and Thalhammer, 2000; Bruckner and Tjoa, 2001) and presents a temporal model that allows even *overlapping* validity time periods for information in a data warehouse

systems. Our valid-time based approach allows the integration of both, real-world state information as well as application semantics. Existing models lack built-in mechanisms for handling change and time. These shortcomings are revealed by a survey of 14 multidimensional data models (Pedersen et al., 2001).

Although data changes over time, it should be possible to perform meaningful analyses across times when data change (Eder and Koncilia, 2001). The problem typically referred to as handling *slowly changing dimensions* (SCD) (Kimball, 1996) is only part of this problem, because we also need to effectively handle the change of state-oriented data (discussed in detail in Section 7).

Related research (Yang and Widom, 1998, 2000) considers issues on temporal view maintenance in the context of the relational model with focus on implementation techniques, but not on query semantics.

TOLAP (*temporal* on-line analytical processing) fully supports OLAP analysis across temporal changes in the dimensions (Mendelzon and Vaisman, 2000; Vaisman and Mendelzon, 2001). However, the model does not explicitly support handling of validity time periods associated with facts. A new, extended multidimensional data model, which addresses eleven requirements for capturing and querying complex data is proposed in Pedersen et al. (2001). The developed algebra for multidimensional objects handles change and time, but does not address late-arriving data associated with valid time.

3. The big picture: Zero-latency data warehousing

Transforming a standard DWH using batch loading during update windows (where analytical access is not allowed) to a zero-latency analytical environment involves various issues to be addressed in order to enable (near) real-time dissemination of new information across an organization. The goal of a *zero-latency* data warehouse is to allow organizations to deliver *relevant* information as fast as possible to knowledge workers or decision systems who need it to react in near real-time to new information captured by an organization's computer system. Therefore, it supports an immediate discovery of abnormal data conditions in a manner not supported by an OLTP system.

The business requirements for a zero-latency analytical environment introduce a set of service level agreements that go beyond what is typical of a traditional DWH. These service levels focus on three basic characteristics, which will be explained in detail in the forthcoming subsections:

Continuous data integration, which enables (near) real-time capturing and loading from different operational sources. This sort of data integration results in an increasing number of *late-arriving data* (e.g. due to propagation delays). Therefore, we need a systematic temporal approach to deal with those situations.

Active decision engines that can make recommendations or even (rule-driven) tactical decisions for routine decision tasks encountered in an analytical environment.

Highly available analytical environments based on an analysis engine that can consistently generate and provide access to current business analyses at any time not restricted by loading windows typical for the common batch approach.

3.1. Continuous data integration

Permanently integrating data from different operational sources addresses the time lines issue discussed in the previous section. Feeding data continuously into a data warehouse minimizes the average latency from when a fact is first captured in an electronic format somewhere in an organization until it is available for a knowledge worker who needs it. Besides technical challenges (mixed workload caused by concurrent updates and analytical queries, scalability, performance, minimized scheduled downtimes, etc.) there are issues which directly affect the analytical environment:

- *Analysis results may change unexpectedly* from the analyst's perspective during the repetition of an identical analytical query if the result set was affected by newly integrated data in the meantime. This is a really critical situation, because it confuses analysts that are accustomed to the stable snapshot paradigm for data warehouses. It is very difficult for them to determine the cause for such an unexpected change: is it newly integrated data, or is there a data quality problem. Thus, there is a need for providing a data warehouse model that copes with continuously integrated data and nonetheless is able to reflect a stable view of the data as it exists at some point in time.
- *Keeping aggregates current.* Aggregates are intended to provide better performance for analytical queries which return results at some aggregated level, rather than all the detailed data. This is a common situation in analytical environments using OLAP. In a traditional data warehouse all the aggregates are updated at the end of every update window. However, in a continuous loading environment this is not feasible. We need a model that is able to reflect multiple versions of aggregates regarding the same dimension hierarchy levels.
- *Active analytical environments.* Active data warehouses need a powerful scheduling mechanism for the mixed workload of queries and continuous data updates. The response time requirement for tactical queries is very critical (and therefore, they should have highest priority) compared with complex (long-running) strategic analyses, where a decrease in priority will only have minor effects on the response time.

3.2. Active data warehouses

An *active* data warehouse is event driven, reacts in a timeframe appropriate to the business needs, and makes tactical decisions or causes operational actions rather than waiting to produce periodic reports (Brobst and Ballinger, 2000). It provides an integrated information repository to drive both *strategic and tactical* decision support within an organization. Therefore, it extends the traditional DWH's functionality into the realm of tactical decision making. This leverages the integrated source of data in the warehouse by deploying timely, cleansed information to decision makers throughout an organization.

Furthermore, rule-driven (active) decision engines can use this information in order to make recommendations or initiate operational actions in near real time responding to pre-defined data conditions in the warehouse. This exploits the high data freshness achieved by continuous data integration. For example, determine the best offer for a single customer ("one-to-one" customer relationship management) in a particular situation (e.g. if the

customer complains on a product at a call-center agent) using up-to-date information (if it is continuously integrated) combined across all available customer touchpoints (accounting, marketing, customer care, etc.). E-business is a major catalyst in the demand for such active data warehouses. Since humans are not (directly) involved in managing the customer relationship experience at a web site, access to advanced analytics and information from a DWH becomes critical for personalizing this sort of interaction. Active data warehouses address specifically the requirements for tactical decision support and represent the central component of a zero-latency data warehousing environment.

3.3. *Late-arriving data*

Late-arriving datasets are defined as data, which is available for loading and is logically related to DWH datasets already integrated during previous update periods (e.g. weeks or even months ago). Data warehouses (in particular zero-latency DWHs) try to represent the history as accurately as possible (to enable tactical decisions). Therefore, late-arriving records are welcome because they make the information more complete.

However, those facts and dimension records are bothersome because they are difficult to integrate (e.g. when using surrogate keys in order to cope with slowly changing dimensions; discussed in more detail in Section 7). The newly integrated datasets change the counts and totals for prior history. There are several reasons why we have to systematically handle late-arriving data:

- *Analysis results may change retrospectively.* Late-arriving data can possibly change analysis results unexpectedly from the analyst's perspective. This situation is similar to that encountered in continuous load environments. It is very difficult to determine the cause of an unexpected change. It can be one of the following two possibilities:
 - Late-arriving data is integrated into the data warehouse and affects historical analysis results by changing counts and totals for prior history. If the data warehouse has to "tie to the books" it is impossible e.g. to change an old monthly sales total, even if the old sales total was incorrect.
 - Loading errors or data quality problems during data staging (e.g. duplicate rows after restarting a loading process) affect detailed data and aggregates.
- *Active analytical environments.* Late-arriving data may invalidate the triggering event for a rule in an active analytical environment, which already triggered an action based on the old (incomplete) view of the data. It is necessary to analyze and compare the situations before and after the insertion of the late-arriving piece of data in order to determine compensating actions for those triggered actions in the past.

3.4. *High availability for data warehouses*

Availability service levels for a zero-latency analytical environment are typically more stringent than for strategic decision support implementations. The active mechanisms of the proposed architecture expand the scope of a traditional DWH to include tactical decision support queries that are critical for the operational aspects of an organization's business. As

a result, $24 \times 7 \times 52$ (24 hours a day, 7 days a week, 52 weeks a year) availability becomes an important requirement, because without access to the DWH, the business cannot operate in an optimal way. These availability requirements apply to both planned (system upgrades, etc.) and unplanned (disaster, etc.) downtimes.

4. Basic temporal data structures

Data warehouses describe the evolving history of an organization. In the case of financial, insurance, and medical applications, to name but a few, it is frequently very important to enable *accountability* or *traceability* of data changes, i.e. to retain a complete record of past states. For example, this need mirrors the practice in finance where accountants correct errors, not by replacing values, but by posting compensating transactions to the books. In medical applications, documenting the situations (“diagnosis”) on which decisions were made by such a record may guard against wrongful malpractice lawsuits.

Timestamps allow the maintenance of temporal data. When considering temporal data for DWHs we need to understand how time is reflected in a database, how this relates to the structure of the data, and how a state change affects existing data. There are a number of approaches:

- *Transient data*. The key characteristic of transient data is that alterations and deletions of existing records physically destroy the previous data content. This type of data is typically found in operational environments (e.g. order-entry systems).
- *Periodic data*. Once a record is added to a database, it is never physically deleted, nor is its content ever modified. Rather, new records are always added to reflect updates or even deletions. Periodic data thus contain a complete record of the changes that have occurred in the data. DWHs are periodic in nature.
- *Semi-periodic data*. This kind of data is typically found in the real-time data of operational systems where previous states are important (bank account systems, insurance premiums systems, etc.). However, almost all operational systems keep only a small history of the data changes due to performance and/or storage constraints. Therefore, this kind of data may be termed semi-periodic data.
- *Snapshot data*. A data snapshot is a stable view of data as it exists at some point in time. It is a special kind of periodic data. Snapshots usually represent the data at some time in the past, and a series of snapshots can provide a view of the history of an organization.

The standard approach for storing periodic data (typically found in DWHs) is to use time stamped status and event records. There are a variety of schemes to maximize the efficiency of timestamps (Boehlen et al., 1998; Jensen and Snodgrass, 1999; Dyreson et al., 2000; Torp et al., 2000; Zhang et al., 2001). Recently, a performance benchmark has been proposed for comparing bitemporal database management systems (Shasha and Zhu, 2001).

The *single* timestamp approach storing only a start time when a record became valid, is applicable to event data, but faces serious deficiencies in the context of DWHs, where in general state information is stored. There are two common types of queries used in DWH environments, which explain the problem:

- A query that needs to access current data. In a single timestamp scheme, the only way to identify current records is to find the latest timestamp of the periodic set, which can be a computationally expensive process.
- A query that builds a view of the data at a particular time in the past. Supporting this kind of query, the period of validity of each record must be known in order to compare it with the required time period. With a single timestamp approach, the end of the period of validity can only be found from the next record in the periodic sequence. In general this is also an expensive process.

In order to address the first problem, a second timestamp (called *end time*) can be added to each fact. It identifies the end of the period of validity. This improves the performance in retrieving data. However, it is not sufficient to fully solve the second problem, because the period of validity can change over time due to new information integrated into the DWH later. A temporally consistent view (similar to snapshot data) during analytical processing requires the storage of both, the old and the new (possibly overlapping) validity period. Therefore, we enhance DWH data models with the following time dimensions (Bruckner et al., 2001):

- *valid time dimension* (validity of knowledge) as discussed above.
- *revelation time dimension* (transaction time). It describes the point in time, when a piece of information was realized by at least one source system.
- *load timestamp*. This represents the point in time when the new piece of information was integrated.

5. Managing temporal consistency

This section presents the temporal model used in this paper. We utilize the terminology proposed by the temporal database community (Etzion et al., 1998). First, we describe the model of time and explain why delays in the gathering of changes can lead to overlapping periods of information validity. Then, we explain how to manage temporal consistency from a conceptual perspective, followed by a discussion how to implement these concepts in a logical and physical data warehouse data model. Finally, we explain how to integrate data from temporal and non-temporal operational sources and how to aggregate data in our model.

5.1. Model for time

The continuum of real time can be modeled by a directed timeline consisting of an infinite set of instants T_i (time points on an underlying time axis). It is a totally ordered set of time points with the ordering relation ' \leq '. A section of the timeline is called a time interval (determined by the duration between two corresponding start-end instants). An event takes place at a particular point in time (T), and therefore does not have a duration. Based on this model we can reflect on past events and possible future events, and thus reason about the validity periods of state information in the domain of time.

Figure 2 describes a situation, where a computer system observes at T_1 that a specified person (Mr. Smith) lives in Boston (state S_1). It gathers that Mr. Smith stays there from T_{1S}

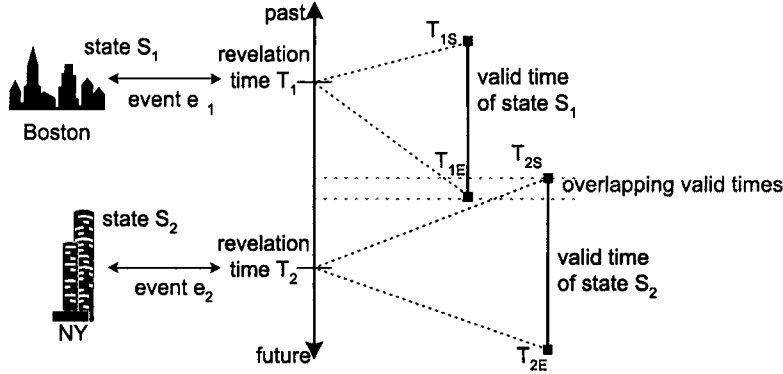


Figure 2. Sequencing the changes to capture the history regarding an entity of interest.

to T_{1E} . The next state (S_2) observed by the computer system regarding Mr. Smith is the new address in New York at T_2 . Furthermore, additional information is captured revealing Mr. Smith already lived in New York since T_{2S} . By modeling this situation time consistently, it will always be possible to find out, that the system did not know where Mr. Smith lived after T_{1E} until the instant T_2 , when the new piece of information was integrated. Based on this knowledge an ADWH can determine the “right” strategy for customer relationship management (CRM): In this example an “information package for new citizens of NY” is appropriate at T_{2S} , but may be unsuitable at the instant T_2 , when the real-world event was actually available to the DWH.

5.2. Conceptual model

A timely consistent representation of information requires a reliable view on historical data at any point in time independent from propagation delays during data integration.

A *knowledge state* (KS) is determined by a specified instant T . It considers all information (knowledge) that was observed, captured, and integrated until T . An ordered relation of two instants $T_1 < T_2$ implies that $KS(T_1) \leq KS(T_2)$. In other words, moving forward in time causes the knowledge state to grow. In general an analysis focuses on a time interval containing at least one *instant of interest* (II): $II_{interest} = [II_{start}, II_{end}]$ (e.g. December 1st, 2001). The KS and II are two *orthogonal* time dimensions and therefore independent from each other regarding analysis capabilities.

A data model enables time consistency if a set of nine conditions listed in Table 1 is satisfied for any combination of two stored datasets (S_1, S_2) regarding the same subject (permuted selections of two datasets if n datasets are involved). In general a stored state S_X is determined by an instant T_X (*revelation time*) and a corresponding *valid time* indicated by a time interval $[T_{XS}, T_{XE}]$.

Any combination of II with KS during analytical processing has to retrieve a correct state, which is actually available to the system at the specified KS . The retrieved state regarding an entity of interest will be S_1 , or S_2 , or “not defined” (if it is neither S_1 nor S_2). Table 1

Table 1. Conceptual model for time consistency.

Knowledge state (KS)	Instants of interest (II)	Retrieved state	Case
$KS < T_1$	any II	–	(1)
$T_1 \leq KS < T_2$	$II < T_{1S}$	–	(2a)
	$T_{1S} \leq II < T_{1S}$	S_1	(2b)
	$II \geq T_{1E}$	–	(2c)
$KS \geq T_2$	$II < T_{1S}$	–	(3a)
	$T_{1S} \leq II < \min(T_{1E}, T_{2S})$	S_1	(3b)
	$T_{1E} < II < T_{2S}$	–	(3c)
	$\min(T_{1E}, T_{2S}) \leq II < T_{2E}$	S_2	(3d)
	$II \geq T_{2E}$	–	(3e)

describes the timely correct state that should be retrieved during analytical processing. The retrieved state can be viewed as the output of a function considering the applied KS and the specified II .

Based on the conceptual model (Table 1, figure 2) we give a more detailed description below and point out the differences of this model compared to conventional temporal models restricted to non-overlapping valid times.

- *Case 1:* An analysis with an applied KS before the timepoint T_1 , reveals that neither state S_1 nor S_2 were already gathered by the organization.
- *Case 2:* At any point in time (i.e. independent from the time when the analysis is conducted), an analysis based on a KS between T_1 and T_2 ($T_1 \leq KS < T_2$) is able to figure out:
 - (2a) the state S_1 was not valid before timepoint T_{1S} in real-world and the organization gathered this state change at timepoint T_1 , which is in general later than T_{1S} due to propagation delays.
 - (2b) the state S_1 is valid from T_{1S} till T_{1E} (based on the applied KS).
 - (2c) the state S_2 is not available to the system (based on the applied KS).
- *Case 3:* Moving forward in time (by applying a *knowledge state* of $KS \geq T_2$) will reveal that:
 - (3a) the state S_1 was not valid before timepoint T_{1S} .
 - (3b) the state S_1 was only valid till T_{2S} (and not till T_{1E}), under the assumption of overlapping valid times for S_1 and S_2 (figure 2).
 - (3c) this case is only relevant, if the corresponding (closed-open) valid time intervals $[T_{1S}, T_{1E})$ and $[T_{2S}, T_{2E})$ do *not* overlap. It reveals that neither state S_1 nor S_2 were valid during time interval $[T_{1E}, T_{2S})$.
 - (3d) the state S_2 is valid from T_{2S} till T_{2E} and replaces state S_1 (which is only valid till T_{2S} and not T_{1E}), under the assumption of overlapping valid times for S_1 and S_2 .
 - (3e) neither state S_1 nor S_2 are valid after the timepoint T_{2E} .

The conceptual model is applicable to n parallel states associated with possibly overlapping valid times. However, at a particular point in time (fixed as KS in the analysis), there is always only one state that is valid for a given subject in a DWH. In order to guarantee this, we establish a temporal order during data staging, which is explained in Section 5.6. The conceptual model is able to handle past and future validity periods (e.g. $T_1 > T_{1E}$, $T_1 < T_{1S}$), which is important (1) to deal with propagation delays regarding timeliness, and (2) to support planning.

5.3. Modeling temporal consistency for data warehouses

The issue of modeling temporal consistency for centralized DWH environments is challenging due to the following reasons. First, DWHs have to deal with propagation delays determined by the time interval it takes for an operational system to realize an occurred state change (discussed in Section 1). Second, the data integration from distributed source systems (e.g. different time zones, delays in data processing) is necessary during data staging.

In our opinion, the most efficient approach is to separate these two issues as described above by extending the logical and physical data model to handle the first one. The latter step, which is complicated due to temporal order issues, is done during data staging and therefore does not affect analytical processing.

We propose an enhanced time dimension model for DWHs to accomplish time consistency, because simply adding timestamps during data loading on an overnight or even weekly basis is not precise enough to model the instant, when a new fact was first captured electronically and therefore available to an organization. The identified propagation delays in a DWH implementation are represented as follows:

- *Valid time (validity of knowledge)*. This property is always related to a base event at an instant T (stored as dataset), and describes the time interval of validity (available at instant T) of that piece of information.
- *Revelation time (transaction time)*. The revelation time describes the point in time, when a piece of information was captured by at least one source system. This concept is tightly related to the notion of transaction time (Jensen and Dyreson, 1998; Torp et al., 2000) in temporal databases. However, in the context of ADWHs we call this property “revelation time” to clarify that (1) transaction results were already recorded at the operational sources and (2) DWH datasets can be assembled together from multiple sources during the data staging (hence, revealing interesting data conditions).
- *Load timestamp*. The loading timestamp is a time value associated with some integrated datasets in a DWH environment and determines the instant when a fact is actually available to active mechanisms. Thus, in the presence of delays, it is the load timestamp that represents the point in time to which automatic decision making or compensating actions should refer in ADWH environments.

A temporal item (corresponding to an identified state S at a source system) in our model has assigned at least three system-maintained temporal attributes, denoted by $S.T_{RS}$ (start revelation time), $S.T_{VS}$ (start valid time), $S.T_{VE}$ (end valid time). The first value describes

the time when the state was captured by an operational source system. Alternatively, this can be modeled using two time instants describing a time interval $[S.T_{RS}, S.T_{RE})$. The other two values define a closed-open time interval $[S.T_{VS}, S.T_{VE})$ when the data item S is valid in the real-world. Updates at source systems cause updates to these time attributes during data staging in order to obtain these semantics. They are discussed in detail in Section 5.6.

The conceptual model is associated with the mentioned (orthogonal) types of time information as follows: The valid time is related to the instants of interest (II) during analytical processing. Typically the knowledge state (KS) relates to the revelation time, to get a timely accurate picture of the entity of interest. However, it is possible to relate to the load timestamp, to enable knowledge workers to observe and control active mechanisms in an ADWH.

5.4. Logical model

When designing a temporal data model, an important and central aspect is the choice of appropriate timestamps of the database facts. A *single timestamp* approach storing only a start time when a record became valid is applicable to event data, but faces serious deficiencies in the context of DWHs, where in general state information is stored. Simply returning such temporary values to users during analytical processing without integration into the data model may lead to confusion.

In order to integrate the load timestamps of facts (as described above) we will use the time dimension commonly available in data warehouse models to effectively support analytical processing known from traditional DWHs. Both, the revelation and valid time of facts will be integrated as separated logical time dimensions modeled by period timestamps. The implementation can be done either by *bitemporal* tables (where information about time is directly stored in separate data fields for valid and revelation timestamps) or by introducing separate logical time dimensions (shown in figure 3). The first approach is appropriate for complex ad-hoc reporting (by decreasing the number of necessary joins). The other approach (logical dimensions) is better suitable for OLAP analysis, because it follows the dimension structure used for modeling (hyper-)cubes, e.g. analyzing the average propagation delay (difference between valid and revelation time) by drilling down to the distributed operational systems located in different geographical regions.

However, independent from the physical representation, both models have the properties of a bitemporal database that combines the features of a transaction-time and a valid-time database (Salzberg and Tsotras, 1999). Thus, it more accurately represents reality and allows for retroactive as well as postactive changes. In order to model past and future instants two symbolic time instants are introduced: $-\infty$ (“*since ever*”) and $+\infty$ (“*until changed*”).

5.5. Physical model

The temporal logical model presented above permits the designer to mix temporal and nontemporal aspects. In particular it is possible to apply different strategies to every fact star in a DWH. Actually implementing a logical data model supporting time consistency is influenced by following issues:

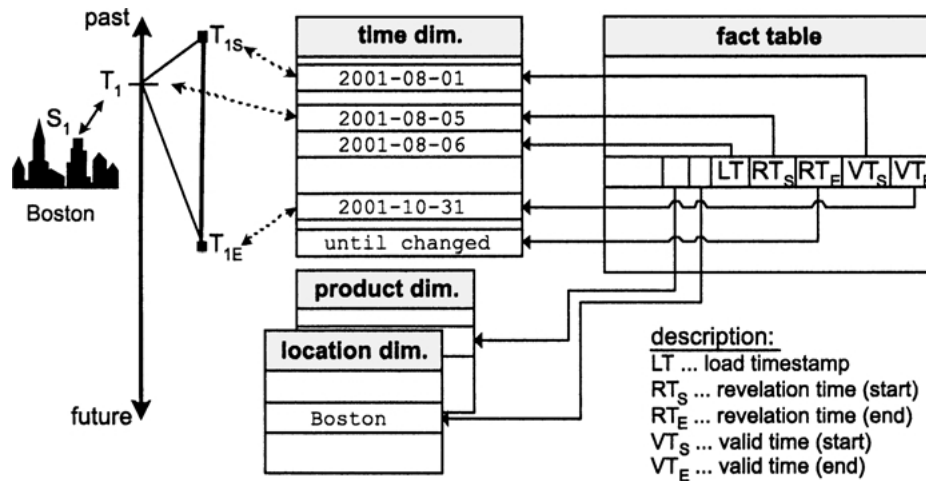


Figure 3. Representation of state S_1 . This piece of information was revealed on August 5th with a valid time from August 1st to October 31st and loaded into the ADWH on August 6th.

- It is necessary to provide a physical representation for symbolic instants ($+\infty$, $-\infty$), e.g. by pre-defined date constants.
- Indexes are very important in temporal relations due to the increased size of the DWH (storing multiple temporal versions to provide an accurate history). Furthermore, the focus of temporal analytical processing on coalescing (merging value-equivalent tuples with intervals that overlap or match) and temporal joins requires appropriate indexes (Salzberg and Tsotras, 1999).
- In temporal queries, conjunctions of inequality predicates (which are harder to optimize) appear more frequently. Consequently, improved cost models for the optimization of temporal operators have to be applied.

5.6. Data extraction and integration from operational sources

While some source systems may store a partial history, many nontemporal sources store only the current state of their data. It is helpful to identify these sources where delays possibly occur which have side-effects on automatic decision making in ADWHs and thus should be modeled in a time consistent manner. Whereas data staging in traditional DWHs establishes a delivery order for new information, managing time consistency requires the (stronger) temporal order, which takes into account already stored datasets.

During data staging we have to consider three issues in order to get a single version of truth regarding time during data staging: (1) different time zones of distributed sources, (2) differences in observing the same event by different operational systems; e.g. a CRM system records an order cancellation on May 2nd, whereas the billing system observes it on May 3rd, and (3) propagation delays (discussed in Section 1).

The previous two issues are addressed by establishing a global time base known from distributed or real-time systems. The latter one is managed by the proposed approach to

Table 2. Representation before the integration of a late-arriving dataset.

Location	Valid time	Revelation time	Load timestamp
Boston	2001-08-01–2001-10-31	2001-08-05– <i>until changed</i>	2001-08-06

Table 3. Temporal order for overlapping valid times.

Location	Valid time	Revelation time	Load timestamp
Boston	2001-08-01–2001-10-31	2001-08-05– 2001-10-14	2001-08-06
NY	2001-09-25–2001-12-31	2001-10-15 – <i>until changed</i>	2001-10-16

time consistency modeling: The integration of new data enhancing the knowledge state regarding a particular subject is done by retroactive updates (see Table 3). Existing data structures (in particular the end date for revelation time) of the ADWH only have to be modified if valid times overlap. It is important to note that both the valid times of the old state provided by the ADWH and the new one from data staging (provided by temporal source systems) are *not* changed.

Tables 2 and 3 show the physical datasets according to the situation exemplified in figure 1 and figure 2. This kind of temporal integration method slows down data staging in the case of overlapping valid times, but simplifies analytical processing considerably.

5.7. Aggregates

The concept of temporal consistency, in particular the introduction of the valid time concept, is also appropriate for aggregates in DWHs. The validity of an aggregate can refer to a specified time period (e.g. two successive data loads). Current aggregates will be characterized by a revelation time of $+\infty$ (“*until changed*”). The integration of new facts concerning information about previous data loads (termed as *late-arriving data*), results in (1) the adjustment of the associated revelation time from “*until changed*” to the load timestamp for all affected aggregates, and (2) the re-computation of aggregates based on current data.

6. Application fields and case study

Managing time consistency is essential in order to store the evolving history of an organization accurately. Our approach enhances analytical processing by the introduction of the knowledge state viewpoints utilizing additionally stored temporal information to deal with propagation delays and to enable traceability and accountability of data changes. Furthermore, planning and monitoring including what-if scenarios (i.e. different versions of plans) are supported at various aggregation levels, because validity periods are suitable to past, present and future states.

Active data warehousing is an emerging discipline supporting automatic decision making. The focus on managing time consistency and storing various versions of states regarding

the same subject enables knowledge workers to control the active behavior and to initiate appropriate compensation actions regarding significantly delayed information.

We investigated our approach in a *case study* (Bruckner and Tjoa, 2001). The PLUS project was an industry project in the domain of accident insurance implemented at a large Austrian insurance authority starting in 1998 and continued till 2000. One focus of the project was the management of time consistency and the introduction of active behavior to automate routine decision tasks, e.g. initiate and control the stopping of accident pension payments due to pre-specified conditions (successful rehabilitation, death, etc.). From a technical viewpoint, the project implemented a data mart (~5 GB) on an Oracle database tightly integrated with a workflow management system and the organization's invoice system. The data model contained fact stars using the proposed temporal model, as well as non-temporal fact stars. The source systems were partially enhanced to provide additional temporal information to better deal with delays. Legacy data was integrated using valid times reconstructed from the sequence of states available regarding the same subject. The active mechanisms checking for predefined data conditions were implemented by using database triggers and temporal events (based on a scheduling system). The ETL process consists of batch loading for data available on a daily or weekly basis, and continuously loading particular transactions from the workflow management system.

A very important analytical application for public insurance authorities is the periodic generation of many timely consistent statistical reports classifying and investigating changes e.g. of accident pensions from various viewpoints. In particular, generating official statistics for the European Statistical System (ESS) and maintaining statistics for local authorities is complex, because of different classification criterias (e.g. *Eurostat-classification*). Therefore, a timely consistent DWH handling late-arriving data systematically to enable *traceability* of data changes is essential. Our approach to manage time consistency reduced the necessary effort to generate and maintain statistics considerably. Generating such statistics without temporal support is difficult, because every report has to contain consistent aggregated figures concerning previous months or years.

Another driving force to establish this kind of temporal data marts is the limited support of traditional DWHs for delays of cash flow transactions. For example, a person has an accident on June 29th and thus will receive a pension. Due to typical delays (pension request, data processing, weekends, etc.) the decision is made on July 5th, when both the billing and statistics for June are already completed. Traditional DWHs ignoring the revelation time and thus storing June 29th, will invalidate cash flows and statistics. The proposed approach supports accountability by identifying real-world states accurately (knowledge state viewpoint) and managing late-arriving data by maintaining valid and revelation times.

7. Evaluation and discussion

The hypercube-based multidimensional model, and the star-schema based extended relational model have emerged as candidate data models for DWHs. However, these models do not adequately address issues related to history data and temporal consistency, which are certainly core issues in data warehousing. In order to model the history of an organization

as accurately as possible, DWHs have to cope with propagation delays. If a DWH has to “*tie to the books*” (cash flows, statistics, strategic planning, etc.), it is not possible to change aggregates (e.g. an old monthly sales total), even if the new information received later indicates that the old sales total were incorrect.

Many multidimensional models implicitly assume that all dimensions are independent of one another. However, most dimensions in a DWH are dependent on the omnipresent *time dimension*. For example, the prices of products in the product dimension may change over time. We term such time-dependent dimensions *slowly changing*. Their presence leads to potential consistency and correctness problems, which we addressed by our temporal model. Three methods were proposed to handle *slowly changing dimensions (SCD)* (Kimball, 1996). We will give a brief overview and identify why they are not sufficient to handle change and time.

The first method suggests to *overwrite* the old value in the dimension row if the attribute description (e.g. product price) changes. This method does not satisfy the DWH goal of tracking history accurately. The problem is that existing fact-table rows that pointed to the old values then incorrectly reference the new values.

The second method proposes to create an additional dimension row at the time of the change with the new attribute value, but *without retaining the actual date* of the change. This makes it impossible to handle facts that may overlap (as proposed by our model) and to deal with late-arriving data that completes the historical view of a particular subject. It also requires the creation of a surrogate key (i.e. artificial, generalized) in order to describe the old and the new relationship between the facts and the dimension. This method is often used in real world DWHs to track history. However, the surrogate key logic affects the extract-transform-load (ETL) performance. Furthermore, late-arriving data which should have been loaded into the DWH weeks or months ago, violate this surrogate key logic (Kimball, 2000).

The third method suggests creating an *additional field* in the slowly changing dimension table and to place the new attribute value in this field while retaining the original attribute value in its place. A problem occurs when the facts for both of these values overlap. Furthermore, this method only retains the original and current values of the changed attribute (intermediate values are lost).

Our approach for modeling temporal consistency is more powerful than the three methods to handle the SCD issue, because it is based on separate time dimensions rather than enhancements for each particular dimension. It enables accurately tracking the validity of facts and dimension levels and allows multiple hierarchies (Pedersen and Jensen, 1999) in the time dimension based on the load timestamp, e.g. days could roll up into weeks or months. To our knowledge it is the first DWH model that handles the change in data over time systematically by adding validity periods and allowing overlapping valid times regarding the same subject.

The applicability of the proposed approach was evaluated by the application of an industry project in the domain of accident insurance. The introduction of the knowledge state viewpoint proved to be an enhancement to the analytical processing. Our approach for managing temporal consistency enables knowledge workers to establish and control active behavior for DWH environments (traceability).

Table 4. Evaluation of the proposed approach.

+/-	Description
+	It is based on a formal model with strong (but simple) guidelines. The implementation uses well-understood concepts, e.g. bitemporal tables (Bliujute et al., 1998).
+	It enhances the time dimension and thus is suitable to various application domains.
+	It improves data quality during data staging by considering stored data structures.
+	It can be introduced at any point in time. If temporal information is not available for legacy data, regular valid and revelation time intervals will be used to upgrade the old data structures (“ <i>until changed</i> ” concept).
+	It enhances analytical capabilities of DWHs (knowledge state, instants of interest).
+	It does not restrict the entire data warehouse to be temporal, but rather permits the designer to mix temporal and nontemporal aspects.
-	It covers only time relevant aspects of data quality management in DWHs.
-	It needs to consider performance issues: Updates to time attributes of existing fact-table rows are necessary when bulk loading the ADWH.

Managing time consistency according to the presented state-oriented approach allows us to achieve consistent analysis results at any point in time for a specified knowledge state. This does not guarantee a timely correct view of the modeled real world. However, it ensures that every piece of information captured by an organization’s operational source system is integrated in a timely correct manner into an entire DWH (evaluated in detail in Table 4).

In contrast, an event-oriented DWH (each row in the fact table represents the change occurred according to the previous event) is easier to feed with new data. However, analytical processing of event-oriented data is slowed down, because queries about states of objects in a certain period of time will result in the comparison of timestamps from different rows. Due to the ability of ADWHs to make decisions on their own (e.g. based on a rule system), it is necessary to allow *overlapping* validity intervals for states (figure 2) to enable compensating actions for automatically triggered decisions in the past.

In general, a temporal data warehouse requires a database system with adequate temporal mechanisms. Most commercial database systems do not provide such support (Yang, 2001). Temporal database prototypes have been built, but often they are unable to deliver the performance and scalability required by a data warehouse. Therefore, a critical issue is to find an approach to implementing temporal database systems that can leverage the power of commercial database systems. By now, many of them are based on the SQL-92 terminology (Melton, 1992) for basic temporal concepts.

Our presented approach proposes extensions to the DWH data model using available, nontemporal database technology and does not necessarily require a temporal database system. The data model and query evaluation techniques discussed in this paper were implemented using standard relational database technology. Therefore, the approach is also applicable to existing data warehouses and operational sources that provide *ad hoc* temporal support using relational database technology. These systems store temporal information as normal attributes in the relational model, and query it using SQL or multidimensional query languages.

8. Conclusion

The advantages provided by built-in temporal consistency support in data warehouses include higher-fidelity data modeling, more efficient gathering of an organization's history, as well as analyzing the sequence of changes to that history.

We have shown how to provide knowledge workers with simple, consistent, and efficient support for active data warehouse environments. The presented approach enables a time consistent view for analysis purposes considering that the validity of detailed data (or aggregates) is typically restricted to time periods because of frequent updates and late-arriving data. Ignoring these temporal issues leads to diminished expressive flexibility and questionable query semantics in many real-life scenarios. The side-effects of propagation delays and late-arriving data can result in overlapping valid time intervals for information stored in data warehouses. Existing models lack built-in mechanisms for handling change and time (Pedersen et al., 2001). The proposed temporal model systematically copes with these issues and enables meaningful analyses across time even when (dimension) data change.

Important future research directions in this field will be the maintenance of data warehouses fed from dynamically changing information systems (data updates, schema changes, dynamic operational sources) (Rundensteiner et al., 2000), and enhancements to the active behavior (e.g. automatic compensation actions for significant changes) in the field of active data warehouses.

Acknowledgments

The authors would like to thank J. Schiefer (IBM Watson Research Center, NY) and A. Rauber (Istituto di Elaborazione dell'Informazione, Pisa, Italy) for their valuable comments and suggestions on previous drafts of this work.

References

- ACT-NET Consortium (1996). The Active Database Management System Manifesto: A Rule-Base of ADBMS Features. *ACM SIGMOD Record*, 25(3), 414–471.
- Ballou, D.P. and Tayi, G.K. (1999). Enhancing Data Quality in Data Warehouse Environments. *Communications of the ACM*, 42(1), 73–78.
- Bliujute, R., Saltenis, S., Slivinskas, G., and Jensen, C.S. (1998). Systematic Change Management in Dimensional Data Warehousing. In *Proc. of the 3rd Intl. Baltic Workshop on Databases and Information Systems*, Riga, Latvia, (pp. 27–41).
- Boehlen, M.H., Busatto, R., and Jensen, C.S. (1998). Point-Versus Interval-Based Temporal Data Models. In *Proc. of 14th Intl. Conf. on Data Engineering (ICDE)*, Orlando, FL, (pp. 192–201). IEEE CS Press.
- Brobst, S. and Ballinger, C. (2000). Active Data Warehousing. Whitepaper EB-1327, NCR Corporation.
- Bruckner, R.M., List, B., Schiefer, J., and Tjoa, A.M. (2001): Modeling Temporal Consistency in Data Warehouses. In *Proc. of the 12th Intl. Workshop on Database and Expert Systems Applications (DEXA)*, Munich, Germany (pp. 467–471). IEEE CS Press.
- Bruckner, R.M. and Tjoa, A.M. (2001). Managing Time Consistency for Active Data Warehouse Environments. In *Proc. of the 3rd Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK)*, Munich, Germany, Springer LNCS, Vol. 2114 (pp. 254–263).
- Chaudhuri, S. and Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1), 65–74.

- Dayal, U. (1988). Active Database Management Systems. In *Proc. of 3rd Intl. Conf. on Data and Knowledge Bases*, Jerusalem (pp. 150–169).
- Dittrich, K.R. and Gatzju, St. (2000). *Aktive Datenbanksysteme*, Heidelberg, Germany: Dpunkt. Verlag (in German).
- Dyreson, C.E., Evans, W.S., Lin, H., and Snodgrass, R.T. (2000). Efficiently Supporting Time Granularities. *IEEE Trans. on Knowledge and Data Engineering*, 12(4), 568–587.
- Eder, J. and Koncilia, C. (2001). Changes of Dimension Data in Temporal Data Warehouses. In *Proc. of the 3rd Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK)*, Munich, Germany, Springer LNCS, Vol. 2114 (pp. 284–293).
- English, L.P. (1999). *Improving Data Warehouse and Business Information Quality*. New York: J. Wiley.
- Etzion, O., Jajodia, S., and Sripada, S. (Eds.). (1998). *Temporal Databases: Research and Practice*, Heidelberg, Germany, Springer LNCS, Vol. 1399.
- Gregersen, H. and Jensen, C.S. (1999). Temporal Entity-Relationship Models—A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3), 464–497.
- Inmon, W.H. (1996). *Building the Data Warehouse*. New York: J. Wiley.
- Jarke, M., Jeusfeld, M.A., Quix, C., and Vassiliadis, P. (1998). Architecture and Quality in Data Warehouses. In *Proc. of the 10th Intl. Conf. on Advanced Information Systems Engineering (CAiSE'98)*, Pisa, Italy, Springer LNCS, Vol. 1413 (pp. 93–114).
- Jensen, C.S. and Dyreson, C.E. (1998). The Consensus Glossary of Temporal Database Concepts. In O. Etzion, S. Jajodia, and S. Sripada (Eds.), *Temporal Databases: Research and Practice*, Springer LNCS, Vol. 1399 (pp. 367–405).
- Jensen, C.S. and Snodgrass, R.T. (1999). Temporal Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), 36–44.
- Kimball, R. (1996). *The Data Warehouse Toolkit*. New York: J. Wiley.
- Kimball, R. (2000). Backward in Time. *Intelligent Enterprise Magazine*, 3(15).
- Melton, J. (Ed.). (1992). *Database Language—SQL*. ANSI X 3.135.
- Mendelzon, A.O. and Vaisman, A.A. (2000). Temporal Queries in OLAP. In *Proc. of the 26th Intl. Conf. on Very Large Databases (VLDB)*, Cairo, Egypt (pp. 242–253). Morgan Kaufmann Publishing.
- Obermair, W. and Schrefl, M. (2000). Temporally Faithful Execution of Business Transactions. In *Proc. of 12th Intl. Conf. on Advanced Information Systems Engineering (CAiSE)*, Stockholm, Sweden, Springer LNCS, Vol. 1789 (pp. 462–481).
- Pedersen, T.B. and Jensen, C.S. (1999). Multidimensional Data Modeling for Complex Data. In *Proc. of the 15th Intl. Conf. on Data Engineering (ICDE)*, Sydney, Australia (pp. 336–345). IEEE CS Press.
- Pedersen, T.B., Jensen, C.S., and Dyreson, C.E. (2001). A Foundation for Capturing and Querying Complex Multidimensional Data. *Information Systems*, 26(5), 383–423.
- Roddick, J.F. and Schrefl, M. (2000). Towards an Accommodation of Delay in Temporal Active Databases. In *Proc. of the 11th Australasian Database Conference (ADC)*, Canberra, Australia (pp. 115–119). IEEE CS Press.
- Rundensteiner, E.A., Koeller, A., and Zhang, X. (2000). Maintaining Data Warehouses over Changing Information Sources. *Communications of the ACM*, 43(6), 57–62.
- Salzberg, B. and Tsostras, V.J. (1999). Comparison of Access Methods for Time-Evolving Data. *ACM Computing Surveys*, 31(2), 158–221.
- Schrefl, M. and Thalhammer, T. (2000). On Making Data Warehouses Active. In *Proc. of the 2nd Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK)*, London, UK. Springer LNCS, Vol. 1874 (pp. 34–46).
- Shasha, D. and Zhu, Y. (2001). SpyTime—A Performance Benchmark for Bitemporal Databases. Available at <http://cs.nyu.edu/cs/faculty/shasha/spytime/spytime.html>.
- Thalhammer, T., Schrefl, M., and Mohania, M. (2001). Active Data Warehouses: Complementing OLAP with Analysis Rules. *Data & Knowledge Engineering*, 39(3), 241–269.
- Torp, K., Jensen, C.S., and Snodgrass, R.T. (2000). Effective Timestamping in Databases. *The VLDB Journal*, 8(3/4), 267–288.
- Vaisman, A.A. and Mendelzon, A.O. (2001). A Temporal Query Language for OLAP: Implementation and a Case Study. In *Proc. of the 8th Biennial Workshop on Data Bases and Programming Languages (DBPL)*, Rome, Italy.
- Wang, R.Y. (1998). Total Data Quality Management. *Communications of the ACM*, 41(2), 58–65.

- Widom, J. (1995). Research Problems in Data Warehousing. In *Proc. of the 5th Intl. Conference on Information and Knowledge Management (CIKM)*, Baltimore, Maryland (pp. 25–30). ACM-Press.
- Widom, J. and Ceri, S. (1996). *Active Database Systems: Triggers and Rules For Advanced Database Processing*. San Francisco: Morgan Kaufmann Publishers.
- Westerman, P. (2000). *Data Warehousing: Using the Wal-Mart Model*. San Francisco: Morgan Kaufmann Publishers.
- Yang, J. (2001). *Temporal Data Warehousing*. Ph.D. Thesis, Department of Computer Science, Stanford University.
- Yang, J. and Widom, J. (1998). Maintaining Temporal Views over Nontemporal Information Sources for Data Warehousing. In *Proc. of the 6th Intl. Conf. On Extending Database Technology (EDBT)*, Valencia, Spain, Springer LNCS Vol. 1377 (pp. 389–403).
- Yang, J. and Widom, J. (2000). Temporal View Self-Maintenance. In *Proc. of the 7th Intl. Conf. On Extending Database Technology (EDBT)*, Konstanz, Germany, Springer LNCS, Vol. 1777 (pp. 395–412).
- Yang, J. and Widom, J. (2001). Incremental Computation and Maintenance of Temporal Aggregates. In *Proc. of the 17th Intl. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany (pp. 51–60). IEEE CS Press.
- Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R.T., Subrahmanian, V.S., and Zicari, R. (1997). *Advanced Database Systems*. San Francisco: Morgan Kaufmann Publishers.
- Zhang, D., Markowetz, A., Tsotras, V.J., Gunopulos, D., and Seeger, B. (2001). Efficient Computation of Temporal Aggregates with Range Predicates. In *Proc. of the 2001 ACM Symposium on Principles of Database Systems (PODS)*, Santa Barbara, CA.