Chapter XI
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

Francisco Araque
University of Granada, Spain

Alberto Salguero
University of Granada, Spain

Cecilia Delgado
University of Granada, Spain

ABSTRACT

One of the most complex issues of the integration and transformation interface is the case where there are multiple sources for a single data element in the enterprise Data Warehouse (DW). There are many facets due to the number of variables that are needed in the integration phase. This chapter presents our DW architecture for temporal integration on the basis of the temporal properties of the data and temporal characteristics of the data sources. If we use the data arrival properties of such underlying information sources, the Data Warehouse Administrator (DWA) can derive more appropriate rules and check the consistency of user requirements more accurately. The problem now facing the user is not the fact that the information being sought is unavailable, but rather that it is difficult to extract exactly what is needed from what is available. It would therefore be extremely useful to have an approach which determines whether it would be possible to integrate data from two data sources (with their respective data extraction methods associated). In order to make this decision, we use the temporal properties of the data, the temporal characteristics of the data sources, and their extraction methods. In this chapter, a solution to this problem is proposed.
INTRODUCTION

The ability to integrate data from a wide range of data sources is an important field of research in data engineering. Data integration is a prominent theme in many areas and enables widely distributed, heterogeneous, dynamic collections of information sources to be accessed and handled.

Many information sources have their own information delivery schedules, whereby the data arrival time is either predetermined or predictable. If we use the data arrival properties of such underlying information sources, the Data Warehouse Administrator (DWA) can derive more appropriate rules and check the consistency of user requirements more accurately. The problem now facing the user is not the fact that the information being sought is unavailable, but rather that it is difficult to extract exactly what is needed from what is available.

It would therefore be extremely useful to have an approach which determines whether it would be possible to integrate data from two data sources (with their respective data extraction methods associated). In order to make this decision, we use the temporal properties of the data, the temporal characteristics of the data sources and their extraction methods. Notice that we are not suggesting a methodology, but an architecture. Defining a methodology is absolutely out of the scope of this paper, and the architecture does not impose it.

It should be pointed out that we are not interested in how semantically equivalent data from different data sources will be integrated. Our interest lies in knowing whether the data from different sources (specified by the DWA) can be integrated on the basis of the temporal characteristics (not in how this integration is carried out).

The use of DW and Data Integration has been proposed previously in many fields. In (Haller, Proll, Rettschitzger, Tjoa, & Wagner, 2000) the Integrating Heterogeneous Tourism Information data sources is addressed using three-tier architecture. In (Moura, Pantoquillo, & Viana, 2004) a Real-Time Decision Support System for space missions control is put forward using Data Warehousing technology. In (Oliva & Saltor, A Negotiation Process Approach for Building Federated Databases, 1996) a multi-level security policies integration methodology to endow tightly coupled federated database systems with a multi-level security system is presented. In (Vassiliadis, Quix, Vassiliou, & Jarke, 2001) a framework for quality-oriented DW management is exposed, where special attention is paid to the treatment of metadata. The problem of the little support for automatized tasks in DW is considered in (Thalhammer, Schrefl, & Mohania, 2001), where the DW is used in combination with event/condition/action (ECA) rules to get an active DW. Finally, in (March & Hevner, 2005) an integrated decision support system from the perspective of a DW is exposed. Their authors state that the essence of the data warehousing concept is the integration of data from disparate sources into one coherent repository of information. Nevertheless, none of the previous works encompass the aspects of the integration of the temporal parameters of data.

In this chapter a solution to this problem is proposed. Its main contributions are: a DW architecture for temporal integration on the basis of the temporal properties of the data and temporal characteristics of the sources, a Temporal Integration Processor and a Refreshment Metadata Generator, that will be both used to integrate temporal properties of data and to generate the necessary data for the later DW refreshment.

Firstly, the concept of DW and the temporal concepts used in this work and our previous related works are revised; following our architecture is presented; following section presents whether data from two data sources with their data extraction methods can be integrated. Then we describe the proposed methodology with its corresponding algorithms. Finally, we illustrate the proposed methodology with a working example.
FEDERATED DATABASES AND DATA WAREHOUSES

Inmon (Inmon, 2002) defined a Data Warehouse as “a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management’s decision-making process.” A DW is a database that stores a copy of operational data with an optimized structure for query and analysis. The scope is one of the issues which defines the DW: it is the entire enterprise. In terms of a more limited scope, a new concept is defined: a Data Mart (DM) is a highly focused DW covering a single department or subject area. The DW and data marts are usually implemented using relational databases, (Harinarayan, Rajaraman, & Ullman, 1996) which define multidimensional structures. A federated database system (FDBS) is formed by different component database systems; it provides integrated access to them: they co-operate (inter-operate) with each other to produce consolidated answers to the queries defined over the FDBS. Generally, the FDBS has no data of its own, queries are answered by accessing the component database systems.

We have extended the Sheth & Larson five-level architecture (Sheth & Larson, 1990), (Samos, Saltor, Sistac, & Bardés, 1998), which is very general and encompasses most of the previously existing architectures. In this architecture three types of data models are used: first, each component database can have its own native model; second, a canonical data model (CDM) which is adopted in the FDBS; and third, external schema can be defined in different user models.

One of the fundamental characteristics of a DW is its temporal dimension, so the scheme of the warehouse has to be able to reflect the temporal properties of the data. The extracting mechanisms of this kind of data from operational system will be also important. In order to carry out the integration process, it will be necessary to transfer the data of the data sources, probably specified in different data models, to a common data model, that will be the used as the model to design the scheme of the warehouse. In our case, we have decided to use an OO model as canonical data model, in particular, the object model proposed in the standard ODMG 3.0.

ODMG has been extended with temporal elements. We call this new ODMG extension as ODMGT. This is also our proposal: to use for the definition of the data warehouse and data mart schema an object-oriented model as CDM, enhanced with temporal features to define loading of the data warehouse and data marts.

ARCHITECTURE EXTENSION WITH TEMPORAL ELEMENTS

Taking paper (Samos, Saltor, Sistac, & Bardés, 1998) as point of departure, we propose the following reference architecture (see Figure 1):

Native Schema. Initially we have the different data source schemes expressed in its native schemes. Each data source will have, a scheme, the data inherent to the source and the metadata of its scheme. In the metadata we will have huge temporal information about the source: temporal data on the scheme, metadata on availability of the source, availability of the log file or delta if it had them, etc.

Some of the temporal parameters that we consider of interest for the integration process are (Araque, Salguero, & Delgado, Information System Architecture for Data Warehousing):

- **Availability Window (AW):** Period of time in which the data source can be accessed by the monitoring programs responsible for data source extraction.
- **Extraction Time (ET):** Period of time taken by the monitoring program to extract significant data from the source.
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

- **Granularity (Gr):** It is the extent to which a system contains discrete components of ever-smaller size. In our case, because we are dealing with time, it is common to work with granules like minute, day, month...

- **Transaction time (TT):** Time instant when the data element is recorded in the data source computer system. This would be the data source TT.

- **Storage time (ST):** Maximum time interval for the delta file, log file, or a source image to be stored.

**Preintegration.** In the Preintegration phase, the semantic enrichment of the data source native schemes is made by the conversion processor. In addition, the data source temporal metadata are used to enrich the data source scheme with temporal properties. We obtain the component scheme (CST) expressed in the CDM, in our case, ODMGT (ODMG enriched with temporal elements).

**Component and Export Schemas.** Apart from the five-scheme levels mentioned (Sheth & Larson, 1990), three more different levels should be considered:

- **Component Scheme T (CST):** the conversion of a Native Scheme to our CDM, enriched so that temporal concepts could be expressed.
- **Exportation Scheme T (EST):** it represents the part of a component scheme which is available for the DW designer. It is expressed in the same CDM as the Component Scheme.
- **Data Warehouse Scheme:** it corresponds to the integration of multiple Exportation Schemes T according to the design needs expressed in an enriched CDM so that temporal concepts could be expressed.

From the CST expressed in ODMGT, the negotiation processor generates the export schemes (EST) expressed in ODMGT. These EST are the part of the CST that is considered necessary for its integration in the DW.

**Integration.** From many data sources EST schemes, the DW scheme is constructed (expressed in ODMGT). This process is made by the Integration Processor that suggests how to integrate the Export Schemes helping to solve semantic heterogeneities (out of the scope of this paper). In the definition of the DW scheme, the DW Processor participates in order to contemplate the characteristics of structuring and storage of the data in the DW.

Two modules have been added to the reference architecture in order to carry out the integration of the temporal properties of data, considering the extraction method used: the Temporal Integration Processor and the Metadata Refreshment Generator.

The Temporal Integration Processor uses the set of semantic relations and the conformed schemes obtained during the detection phase of similarities (Oliva & Saltor, A Negotiation Process Approach for Building Federated Databases, 1996). This phase is part of the integration methodology of data schemes. As a result, we obtain data in form of rules about the integration possibilities existing between the originating data from the data sources (minimum granularity, if the period of refreshment must be annotated between some concrete values). This information is kept in the Temporal Metadata Warehouse. In addition, as a result of the Temporal Integration process, a set of mapping functions is obtained. It identifies the attributes of the schemes of the data sources that are self-integrated to obtain an attribute of the DW scheme.

The Metadata Refreshment Generator determines the most suitable parameters to carry...
out the refreshment of data in the DW scheme (Araque & Samos, Data warehouse refreshment maintaining temporal consistency, 2003). The DW scheme is generated in the resolution phase of the methodology of integration of schemes of data. It is in this second phase where, from the minimum requirements generated by the temporal integration and stored in the Temporal Metadata warehouse, the DW designer fixes the refreshment parameters. As result, the DW scheme is obtained along with the Refreshment Metadata necessary to update the former according to the data extraction method and other temporal properties of a concrete data source.

Obtaining of the DW scheme and the Export schemes is not a linear process. We need the Inte-
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

Data Warehouse Refreshment. After temporal integration and once the DW scheme is obtained, its maintenance and update will be necessary. This function is carried out by the DW Refreshment Processor. Taking both the minimum requirements that are due to fulfill the requirements to carry out integration between two data of different data sources (obtained by means of the Temporal Integration module) and the integrated scheme (obtained by the resolution module) the refreshment parameters of the data stored in the DW will be adjusted.

TEMPORAL PROPERTIES INTEGRATION

After the initial loading, warehouse data must be regularly refreshed, and modifications of operational data since the last DW refreshment must be propagated into the warehouse so that the warehouse data reflects the state of the underlying operational systems (Araque, Data Warehousing with regard to temporal characteristics of the data source, 2002), (Araque, Real-time Data Warehousing with Temporal Requirements, 2003), (Araque, Integrating heterogeneous data sources with temporal constraints using wrappers, 2003).

Data Sources

Data sources can be operational databases, historical data (usually archived on tapes), external data (for example, from market research companies or from the Internet), or information from the already existing data warehouse environment. They can also be relational databases from the line of business applications. In addition, they can reside on many different platforms and can contain structured information (such as tables or spreadsheets) or unstructured information (such as plain text files or pictures and other multimedia information).

Extraction, transformation and loading (ETL) (Araque, Salguero, & Delgado, Monitoring web data sources using temporal properties as an external resources of a Data Warehouse, 2007) are data warehousing processes which involve extracting data from external sources, adapting it to business needs, and ultimately loading it into the data warehouse. ETL is important as this is the way data actually gets loaded into the warehouse.

The first part of an ETL process is to extract the data from the source systems. Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization/format. Common data source formats are relational databases and flat files, but there are other source formats. Extraction converts the data into records and columns.

The transformation phase applies a series of rules or functions to the extracted data in order to derive the data to be loaded.

During the load phase, data is loaded into the data warehouse. Depending on the organization’s requirements, this process can vary greatly: some data warehouses merely overwrite old information with new data; more complex systems can maintain a history and audit trail of all the changes to the data.

Data Capture

DWs describe the evolving history of an organization, and timestamps allow temporal data to be maintained. When considering temporal data for DWs, we need to understand how time is reflected in a data source, how this relates to the structure of the data, and how a state change affects existing
A number of approaches have been explored (Bruckner & Tjoa, 2002):

- **Transient data**: Alterations and deletions of existing records physically destroy the previous data content.
- **Semi-periodic data**: Typically found in the real-time data of operational systems where previous states are important. However, almost all operational systems only retain a small history of the data changes due to performance and/or storage constraints.
- **Periodic data**: Once a record has been added to a database, it is never physically deleted, nor is its content ever modified. Instead, new records are always added to reflect updates or even deletions. Periodic data thus contain a complete record of any data changes.
- **Snapshot data**: A stable view of data as it exists at some point in time.

Capture is a component of data replication that interacts with source data in order to obtain a copy of some or all of the data contained therein or a record of any changes (Castellanos, 1993). In general, not all the data contained in the source is required. Although all the data could be captured and unwanted data then discarded, it is more efficient to capture only the required subset. The capture of such a subset, with no reference to any time dependency of the source, is called static capture. In addition, where data sources change with time, we may need to capture the history of these changes. In some cases, performing a static capture on a repeated basis is sufficient. However, in many cases we must capture the actual changes that have occurred in the source. Both performance considerations and the need to transform transient or semi-periodic data into periodic data are the driving force behind this requirement. This type is called incremental capture.

Static capture essentially takes a snapshot of the source data at a point in time. This snapshot may contain all the data found in the source, but it usually only contains a subset of the data. Static capture occurs from the first time a set of data from a particular operational system is to be added to the data warehouse, where the operational system maintains a complete history of the data and the volume of data is small.

Incremental capture is the method of capturing a record of changes occurring in a source data set. Incremental capture recognizes that most data has a time dependency, and thus requires an approach to efficiently handle this. As the volume of changes in a set of data is almost always smaller than the total volume, an incremental capture of the changes in the data rather than a static capture of the full resulting data set is more efficient.

Delayed capture occurs at predefined times, rather than when each change occurs. In periodic data, this behaviour produces a complete record of the changes in the source. In transient and semi-periodic data, however, the result in certain circumstances may be an incomplete record of changes that have occurred. These problems arise in the case of deletions and multiple updates in transient and semi-periodic data.

There are several data capture techniques, and static capture is the simplest of these. Incremental capture, however, is not a single topic. It can be divided into five different techniques, each of which has its own strengths and weaknesses. The first three types are immediate capture, whereby changes in the source data are captured immediately after the event causing the change to occur. Immediate capture guarantees the capture of all changes made to the operational system irrespective of whether the operational data is transient, semi-periodic, or periodic. The first three types are:

- Application-assisted capture, which depends on the application changing the operational data so that the changed data may be stored in a more permanent way
- Triggered capture, which depends on the database manager to store the changed data in a more permanent way
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

- Log/journal capture, which depends on the database manager’s log/journal to store the changed data

Because of their ability to capture a complete record of the changes in the source data, these three techniques are usually used with incremental data capture. In some environments, however, technical limitations prevent their use, and in such cases, either of the following two delayed capture strategies can be used if business requirements allow:

- Timestamp-based capture, which selects changed data based on timestamps provided by the application that maintains the data.
- File comparison, which compares versions of the data in order to detect changes.

Temporal Concepts

In order to represent the data discussed previously, we use a time model consisting of an infinite set of instants Ti (time points on an underlying time axis). This is a completely ordered set of time points with the ordering relation ‘≤’ (Bruckner & Tjoa, 2002). Other temporal concepts may also be necessary:

- An instant is a time point on an underlying time axis.
- A timestamp is a time value associated with some object, e.g. an attribute value or a tuple.
- An event is an instantaneous fact, i.e. something occurring at an instant.
- The lifespan of an object is the time over which it is defined. The valid-time lifespan of an object refers to the time when the corresponding object exists in the modelled reality. Analogously, the transaction-time lifespan refers to the time when the database object is current in the database.

- A temporal element is a finite union of n-dimensional time intervals. These are finite unions of valid time intervals, transaction-time intervals, and bitemporal intervals, respectively.
- A time interval is the time between two instants.
- The transaction time (TT) of a database fact is the time when the fact is current in the database and may be retrieved.
- The valid time (VT) of a fact is the time when the fact is true in the modelled reality. A fact may have any number of associated instants and intervals, with single instants and intervals being important in special cases. Valid times are usually supplied by the user.

We can represent the temporal characteristics of the data source with the temporal concepts presented previously. It is therefore possible to determine when the data source can offer the data and how this data changes over time (temporal characteristics). This can be represented in the temporal component schema and used by the DW administrator to decide how to schedule the refreshment activity. It depends on the temporal properties of the data source.

Temporal Properties of Data

The DW must be updated periodically in order 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 DWs and the data integration process (ETL) result in increased propagation delays.

Having the necessary information available on time means that we can tolerate some delay
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

(be it seconds, minutes, or even hours) between the time of the origin transaction (or event) and the time when the changes are reflected in the warehouse environment. This delay (or latency) is the overall time between the initial creation of the data and its population into the DW, and is the sum of the latencies for the individual steps in the process flow:

- Time to capture the changes after their initial creation.
- Time to transport (or propagate) the changes from the source system to the DW system.
- Time to have everything ready for further ETL processing, e.g. waiting for dependent source changes to arrive.
- Time to transform and apply the detail changes.
- Time to maintain additional structures, e.g. refreshing materialized views.

It is necessary to indicate that we take the following conditions as a starting point:

- We consider that we are at the E of the ETL component (Extraction, Transformation and Loading). This means we are treating times in the data source and in the data extraction component. This is necessary before the data is transformed in order to determine whether it is possible (in terms of temporal questions) to integrate data from one or more data sources.
- Transforming the data (with formatting changes, etc.) and loading them into the DW will entail other times which are not considered in the previous “temporal characteristic integration” of the different data sources.
- We suppose that we are going to integrate data which has previously passed through the semantic integration phase.

We consider the following temporal parameters to be of interest on the basis of the characteristics of the data extraction methods and the data sources (Figure 2):

- VTstart: time instant when the data element changes in the real world (event). At this moment, its Valid Time begins. The end of the VT can be approximated in different ways which will depend on the source type and the data extraction method. The time interval from VTstart to VTend is the lifespan.
- TT: time instant when the data element is recorded in the data source computer system. This would be the transaction time.
- W: time instant when the data is available to be consulted. We suppose that a time interval can elapse between the instant when the data element is really stored in the data source computer system and the instant when the data element is available to be queried. There are two possibilities:
  - that W <VDstart (in this case, the data element would only be available on the local source level or for certain users)
  - that VDstart <= W <VDend (in this case, the data element would be avail-

**Figure 2. Temporal properties of data**
able for monitoring by the extraction programs responsible for data source queries)

- **VD**: Availability Window (Time interval). Period of time in which the data source can be accessed by the monitoring programs responsible for data source extraction. There may be more than one daily availability window. Then:
  - \( \text{VDstart} \): time instant when the availability window is initiated
  - \( \text{VDend} \): time instant when the availability window ends

- **TE**: Extraction Time (Time interval). Period of time taken by the monitoring program to extract significant data from the source. Then:
  - \( \text{TEstart} \): time instant when the data extraction is initiated.
  - \( \text{TEend} \): time instant when the data extraction ends.
  - We suppose that the TE is within the VD in case it were necessary to consult the source to extract some data. In other words, \( \text{VDstart} < \text{TEstart} < \text{TEend} < \text{VDend} \).

- **M**: time instant when the data source monitoring process is initiated. Depending on the extraction methods, M may coincide with TEstart.

- **TA**: maximum time interval storing the delta file, log file, or a source image. We suppose that during the VD, these files are available. This means that the TA interval can have any beginning and any end, but we suppose that it at least coincides with the source availability window. Therefore, \( \text{TAstart} \leq \text{VDstart} \) and \( \text{VDend} \leq \text{TAend} \).

- **Y**: time instant from when the data is recorded in the DW.

- **Z**: time instant from when certain data from the DW are summarized, passed from one type of storage to another because they are considered unnecessary.

From VTstart to Z represents the real life of a data element from when it changes in the real world until this data element moves into secondary storage. Y and Z parameters it is not considered to be of immediate usefulness in this research.

By considering the previous temporal parameters and two data sources with their specific extraction methods (this can be the same method for both), we can determine whether it will be possible to integrate data from two sources (according to DWA requirements).

### DATA INTEGRATION PROCESS

Prior to integration, it is necessary to determine under what parameters it is possible and suitable to access the sources in search of changes, according to their availability and granularity, obtained automatically by the tool of the previous section. This process is carried out by the pre-integration algorithm. It is only possible to determine these parameters previously if there is some pattern related to the source availability. The parameters obtained as a result shall be used in the specific integration algorithms whenever the data sources are refreshed (M).

One of the most complex issues of the integration and transformation interface is the case where there are multiple sources for a single element of data in the DW. For example, in the DW there is a data element that has as its source data elements \( a1 \) from legacy application A and a data element \( b1 \) from legacy application B. If it is possible to temporally integrate the data from both sources (on the basis of their temporal properties), semantic integration is undertaken and the result is stored in the DW.

The integration methodology, shown in Figure 2, consists of a set of processes that define the rules for capturing a parameter from a single source as well as integrate a set of values semantically equivalent coming from different data sources. It has two phases, shown in Figure 3: Temporal
Integration (A) and Generation of Refresh metadata (B). The elements of the architecture that are of interest in this paper have been shadowed in Figure 3.

The temporary process of integration can also be divided into two different tasks: the analysis of the accessibility of both sources and the analysis of temporal requirements. The first of the previous tasks, which this article is focused on, verifies that certain temporary parameters common to any type of extraction method are satisfied, so the integration can be carried out, whereas the second one, which would be carried out only in the case of surpassing the first task, is focused on determining whether the integration of specific sources of data is possible. We obtain as a result data in form of rules about the integration possibilities existing between the data of the sources (the minimum granularity that can be obtained, the intervals in which refreshment should be performed, etc). The second task will be explained in temporal requirements algorithm section.

In the second phase the most suitable parameters are selected to carry out the refreshment process of the data. It is in this second phase where, from the minimum requirements selected by the temporary first stage of integration, the DW designer sets the refreshment parameters. These parameters can be set automatically by the system taking care of different criteria (like the maximum level of detail, the no-saturation of the communication resources, etc). As a result, the necessary metadata are obtained so that the DW can be refreshed coherently depending on the type of extraction method and other temporary characteristics of the data sources.

This process does not guarantee that the integration of all of the changes detected in the sources can be carried out satisfactorily. Instead, what it guarantees is that the process of integration of a change can be carried out only and exclusively the times that are necessary to obtain the objectives proposed by the DW designer, attending to aspects related to the refreshment and the availability of the data.

### Accessibility Algorithm

Given two data sources, the first task to do is to determine the smallest sequence in the intersection of the set of the availability window values of both data sources that is repeated periodically. We will denominate this concept “common pattern of availability”. For example, if the availability window of a data source is repeated every thirty six hours and the window of another is repeated...
every twenty four hour, the “common pattern of
availability” will be an interval of duration equal
to seventy two hours (see Figure 4).

The algorithm, shown in Figure 5, first
determines the maximum level of detail which
both data sources can provide. For example, if
a source provides data with a level of detail of
day, whereas another one provides them at an
hour level, it is not possible to integrate them to
obtain a level of detail better than a day (hours,
minutes, seconds ...).

It can occur that the unit (the granule) of the
level of detail that can be obtained after the inte-
gration of both data sources has a length greater
than the “common pattern of availability”. For
example, that a granularity at day level can be
obtained and the length of the common pattern
is of several hours. In this case, querying the
data sources once a day would be enough (it
does not make sense to check a data source more
often than it is going to be stored). Therefore, the
maximum interval width of refreshment in the
algorithm is adjusted to the length of the unit
of the level of detail, obtained by means of the
function “interval” in the algorithm. The value
of the period of sampling could be, in the case
of the previous example, multiple of a day (two
days, three days, one week ...). Within the com-

Figure 5. Accessibility algorithm

In:
source[] : list of sources that contains the semantically equivalent parameter to
integrate
commonAW : common Availability Window pattern.

Out:
M[] : list of instants to query the sources

If commonAW is periodical then
GrMax = MinDetail(Granularity(source[1]), Granularity(source[2]), ...)
// Example: day - MinDetail(hour, day)
If interval(GrMax) >= interval(commonAW) then
LongestAW = LongestAWInterval(commonAW)
M[0] = LongestAW.Start
RefreshmentInterval = interval(GrMax)
Else
i = 0, j = 0
While interval(GrMax) * j < Interval(commonAW).end
If all sources are accessible at interval(GrMax) * j
M[i] = interval(GrMax) * j
i++
j++
Else
"It is not possible to determine the integration process previously"

"It is not possible to determine the integration process previously"
mon pattern the moment in which the interval of maximum length begins is chosen to make the refreshment in which both sources are available, so that there is more probability to satisfy the restrictions imposed in the second phase, the Analysis of Temporal Requirements (out of scope of this paper). This interval is determined by the “LongestAWInterval” function.

In case that the unit (the granule) of the level of detail that can be obtained after the integration of both data sources has a length smaller than the common pattern of availability, it is necessary to determine in what moments within the common pattern both data sources are going to be available to refresh their data. Since it does not make sense to refresh a data more often than is going to be stored, only values that distant the length of the integrated granularity unit are chosen.

For example, if the granularity with which the data are going to be integrated correspond to “seconds”, the instants will be temporarily distanced one second. Then it is verified that, for all those instants of the common pattern, both data sources are accessible. If it is successful it will be added to the set of instants (M) in which the refreshment can be made.

Some of the instants included in the M set will be discarded in the following phase because they do not fulfil some of the specific requirements that depend on the precise kind of sources. In this case, due to the fact that we are integrating web data sources which usually are simply HTML flat files, we will use a File Comparison-based method to do the integration process. This method consists on compare versions of the data in order to detect changes (Araque, Salguero, Delgado, & Samos, Algorithms for integrating temporal properties of data in Data Warehousing, 2006).

Every extracting method has its own requirements. If we are using a File Comparison-based method we need to ensure that the following sentence is valid:

$$(ET(DS1) \cup ET(DS2)) \subseteq (AW(DS1) \cap AW(DS2))$$

where ET(X) is the time needed to extract a change from the source X (Extraction Time), AW(X) is the Availability Window of the source X and DS1 and DS2 are both data sources. In other words, we cannot carry out the integration process of both data sources more often than the time we need to extract the changes. Obviously, if we need thirty seconds to extract the changes from a source and forty seconds to extract them from another source, it is not possible to integrate them every minute because we are not able to get the changes from both data sources so quickly.

Temporal Requirements Algorithms

In the following paragraphs, we shall explain how verification would be performed in order to determine whether data from data sources can be integrated. It is necessary to indicate that if we rely on 5 different extraction methods, and the combination of these two at a time, we would have 15 possible combinations. In this article, we shall focus on only two cases: firstly, the combination of two sources, one with the File Comparison method (FC) and the other with the Log method (LOG); secondly, the combination of two sources both with the same log method (LOG).

We suppose that the data recorded in the delta and log files have a timestamp which indicates the moment when the change in the source occurred (source TT). The following paragraphs describe the crosses between extraction methods on an abstract level, without going into low level details which shall be examined in subsequent sections.

LOG – FC. In this case, the LOG method extracts the data from the data source and provides us with all the changes of interest produced in the source, since these are recorded in the LOG file. The
FC method, on the other hand, only provides us with some of the changes produced in the source (depending on how the source is monitored). We will therefore be able to temporally integrate only some of the changes produced in both sources. Integration of the TT parameter would not be possible as the FC method does not have this parameter. On an abstract level, we can say that temporal integration may be carried out during all of the previously mentioned temporal parameters or characteristics except TT.

The granularity is a parameter that is inherent to the data source, while the refreshment period depends on the DW designer. This is true in all cases except for the case of data sources with File Comparison extracting method, in which the level of detail of the changes is determined by the time elapsed between two consecutives images of the data source.

Let suppose the data sources in Figure 6. The maximum level of detail we can obtain for the parameter level once integrated (and the rest of attributes) is a day, i.e. the highest level of detail available in both data sources. In the temporal warehouse metadata repository is generated a rule that states this fact. This rule implies, in addition, a restriction in the process of refreshment. It does not make sense to query the data sources more

*Figure 6. LOG and FC integration process example*
frequently than the level of detail used to store the parameter in the warehouse. Thus, in this case, there is reason to query the data sources more than once a day. Moreover, since these sources can be accessed simultaneously only on Mondays, the period of refreshment should be multiple of seven days (once a week, once every two weeks, once every three weeks, once a month, ...), and should be twenty-three hours and fifty-nine minutes length.

**LOG—LOG.** In this case, we carry out the temporal integration of data (from the same or different sources) extracted with the same method. From the source where the data are extracted with the LOG method, all the produced changes are available. We will therefore be able to temporally integrate all the changes produced in both sources. On an abstract level, we can say that temporal integration may be carried out during all of the previously mentioned temporal properties.

Prior to integration, it is necessary to determine under what parameters it is possible and suitable to access the sources in search of changes, according to their availability and granularity (Gr). This process is carried out by the pre-integration algorithm. It is only possible to determine these parameters previously if there is some pattern related to the source availability (Figure 7). The parameters obtained as a result shall be used in the specific integration algorithms whenever the data sources are refreshed (M). If it is possible to temporally integrate the data from both sources (on the basis of their temporal properties), semantic integration is undertaken and the result is stored in the DW.

**Data sources.** By way of example to show the usefulness of these algorithms, an application is used which has been developed to maximize the flight experience of soaring pilots. These pilots depend to a large extent on meteorological conditions to carry out their activity and an important part of the system is responsible for handling this information. Two data sources are used to obtain this type of information:

- The US National Weather Service Web site. We can access weather measurements (temperature, pressure, humidity, general conditions and wind speed and direction) every hour in every airport in the world. It is a FC data source.
- In order to obtain a more detailed analysis and to select the best zone to fly, pilots use another tool: the SkewT diagram. The SkewT, or sounding chart, is a vertical snapshot of temperature, dew point and winds above a point on the earth. These soundings are carried out in some airports every twelve hours by launching a balloon sounding. It is a LOG data source.

The information provided by both data sources is semantically equivalent in certain cases. Given an airport where soundings are carried out, the lower layer meteorological information obtained in the sounding and that obtained from a normal meteorological station must be identical if relating to the same instant. In order to integrate these data, it is necessary to use the algorithms described in the following section.
Algorithm for FC – LOG

Every time the data source with the FC method is accessed, the value of the parameter to be integrated is extracted and this is compared with its last known value. If there has been a change, it is necessary to search for the associated change in the LOG source in order for integration to be performed. Since the LOG source might have collected more than one change in the period which has elapsed since the last refreshment, only the last change occurring in this period is taken into account. This is verified by consulting the TT value of the change in question.

If integration was possible, the value of the variable which stores the previous value of the FC-type source is updated. If integration was not possible, the value of this variable is not updated, so that if the change is detected in the LOG source in subsequent refreshments, integration can be carried out even if there has been no further change in the value of the parameter in the FC source.

Figure 8 represents the evolution of the meteorological data sources from the example which we are following (one source with a LOG extraction method and another with an FC method). If the designer wants to obtain this information with a daily level of detail, the integration process of the change “A” detected in the temperature would be carried out in the following way: every twenty-four hours, both sources are consulted; if the temperature value on the airport website has changed in relation to our last stored one, the two changes of the same parameter which have occurred in the source corresponding to the soundings in the last twenty-four hours are recovered (as they are carried out every twelve hours and all the changes are recorded). The value from the website is then semantically integrated with the latest one of these. The algorithm for FC – LOG is as follows:

Algorithm for FC – LOG

available = true
If any source is not periodical
    available = CheckAvailabilityW(Log)
available = CheckAvailabilityW & available
If available = true
    newValue^FC = readValue(FC)
If newValue^FC <> oldValue^FC
    newValue^LOG = last log value
If TT(newValue^LOG) < M_{i-1} ; Impossible to integrate the change
    ; because it still has not been
    ; detected in the Log source.
If-not
    result=Integrate(newValue^FC, newValue^LOG)
    oldValue^FC = newValue^FC

Algorithm for LOG – LOG

This algorithm maintains a record of the changes which still remain to be detected in both sources. Every so often, the algorithm is executed and the two data sources from this temporal record are consulted and the pairs of changes are integrated. The first change is obtained in the source 1 of the parameter to be integrated. This change must take place after the record which indicated the first change which could not be integrated.

If either of these two changes has occurred since the last refreshment, this means that this

Figure 8. LOG – FC
is the first time that a change in some source has been recorded and so integration may be carried out. Since this is a log, all the changes repeated in both sources must appear and must also be ordered temporally.

Figure 9 shows an integration example of two log-type data sources. The third time that the data sources are consulted (instant M3), it is not possible to integrate change “A” because it is still unavailable in one of the sources. The instant corresponding to the change detected is saved and no action is taken until the following refreshment. The fourth time that the sources are consulted, the temporal record is read first. In this case, change “A” is recorded in the second data source, and we therefore know that this change has not been integrated previously. It is then integrated semantically and the main loop of the algorithm is reiterated. When change “B” is detected in both sources, integration may be carried out directly.

The algorithm is as follows:

```plaintext
available = true
allChanges = true
If any source is not periodical
    available = CheckAvailabilityW(Log)
    available = CheckAvailabilityW & available
If Now – LastTimeRefreshed < ST
    allChanges = false
If available = true & allChanges = true
    Repeat
        v1 = firstChangeAfter(updatedTo, Log1)
        v2 = firstChangeAfter(updatedTo, Log2)
        If TT(v1) > M_{i-1} || TT(v2) > M_{i-1}
            result = integrate(v1, v2)
            updatedTo = min(TT(v1), TT(v2))
        while v1 <> null && v2 <> null
```

**EXAMPLE**

A Decision Support System (DSS) being based on a DW (March & Hevner, 2005) is presented as an example. This can be offered by Small and Medium-Sized Enterprises (SMEs) as a plus for adventure tourism. Here, a DSS (Figure 10) is used to assist novel and expert pilots in the decision-making process for a soaring trip (Araque, Salguero, & Abad, Application of data warehouse and Decision Support System in Soaring site recommendation, 2006).

These pilots depend to a large extent on meteorological conditions to carry out their activity and an important part of the system is responsible for handling this information. Two web data sources are mainly used to obtain this kind of information:

- The US National Weather Service Website. We can access weather measurements (temperature, pressure, humidity, etc) in every airport in the world.
- In order to obtain a more detailed analysis and to select the best zone to fly, pilots use another tool: the SkewT diagram. The SkewT, or sounding chart, is a vertical snapshot of temperature, dew point and winds above a point on the earth.

The information provided by both data sources is semantically equivalent in certain cases. In order to efficiently integrate these data, it is necessary to use the algorithm described in the previous
section. It is needed to use an efficient approach because these kinds of services are offered by SMEs which often have limited resources. The continuous integration of Web data sources may result in a collapse of the resources they use to communicate with their clients, which are not designed to support the laborious task of maintaining a DW up to date.

In our approach, the DW Administrator (DWA) introduces the data sources temporal properties in DECT tool (Araque, Real-time Data Warehousing with Temporal Requirements, 2003), (Araque, Integrating heterogeneous data sources with temporal constraints using wrappers, 2003) and selects the parameters to integrate, for example the temperature. This tool is able to determine the maximum level of detail (granularity) provided by each data source after a period of time. It uses an algorithm to determine the frequency of the changes produced at the data source. We approximate the granularity of the source by selecting the smallest interval that take place between two consecutive changes.

In the first source, the information about the temperature can be precise with a detail of “minute” (for example, that at 14 hours and 27 minutes there was a temperature of 15°C), whereas in the second case it talks about the temperature with a detail of “hour” (for example, that at 14 hours there were 15°C). The reason is that in the first source has been detected more than one change within an hour at least once, whereas in the second source all the changes has been detected at least one hour distanced.

It can also determine the time intervals in which this information is available to be queried. Let us suppose that the first data source is always available, but the second one is only accessible from 23:10 to 00:10 and from 12:00 to 15:59 (availability window). Common pattern of availability would include, therefore, a whole day. Applying the accessibility algorithm we would obtain all possible instants of querying in which both sources are accessible and are distanced an interval equal to the maximum integrated granularity unit each other (hourly in the example we are using). Using the values of this example we would obtain {00:00, 12:00, 13:00, 14:00, 15:00}.

For each one of the previous set of instants is necessary to verify that the extraction and integration of the data sources would be possible. For this purpose we use the second algorithm mentioned in the previous section (out of the scope of this paper).

To help DWA in this process we have developed a tool that is able of performing both algorithm described in this paper: Accessibility Algorithm and Analysis of Temporal Requirements. A capture of this tool can be seen in Figure 11.

Using the data extracted from Web data sources a DSS for adventure practice recommendation can be offered as a post-consumption value-added service by travel agencies to their customers. Therefore, once a customer makes an on-line

Figure 10. Motivation example applied to tourism area
reservation, the travel agency can offer advice about adventure practices available in the area that customer may be interested in. Due to the high risk factor accompanying most adventure sports, a regular information system is far from being accurate. A more sophisticated ICT system is required in order to extract and process quality information from different sources. In this way, the customer can be provided with true helpful assistance to be aided in the decision-making process.

While logging reservation systems do not need supplementary information as weather forecast, other products in the tourist industry, such as eco-tourism can take a tremendous advantage of last-minute DW. The system allows to query a last-minute DW and use the output report to filter the on line availability of outdoor activities offered by the on line reservation system.

ACKNOWLEDGMENT

This work has been supported by the Research Program under project GR2007/07-2 and by the Spanish Research Program under projects EA-2007-0228 and TIN2005-09098-C05-03.

CONCLUSION AND FUTURE WORK

We have presented our work related to Data Warehouse design using data sources temporal metadata. The main contributions are: DW architecture for temporal integration on the basis of the temporal properties of the data and temporal characteristics of the sources, a Temporal Integration Processor and a Refreshment Metadata Generator, that will be both used to integrate temporal properties of data and to generate the necessary data for the later DW refreshment. In addition, we proposed a methodology with its corresponding algorithms.

Actually we are working about using a parallel fuzzy algorithm for integration process in order to obtain more precise data in the DW. The result is more precise because several refreshments of data sources are semantically integrated in a unique DW fact (Carrasco, Araque, Salguero, & Vila, 2008), (Salguero A. , Araque, Carrasco, Vila, & Martinez, 2007), (Araque, Carrasco, Salguero, Delgado, & Vila, 2007).

On the other hand, our work is now centred on use of a canonical data model based on ontologies to deal with the data sources schemes integration. Although it is not the first time the ontology model has been proposed for this purpose, in this case the work has been focused on the integration of spatio-temporal data. Moreover, to our knowledge this is the first time the metadata storage capabilities of some ontology definition languages has been used in order to improve the DW data refreshment process design (Salguero, Araque, & Delgado, Using ontology metadata for data warehousing, 2008), (Salguero, Araque, & Delgado, Data integration algorithm for data warehousing based on ontologies metadata, 2008), (Salguero & Araque, Ontology based data warehousing for improving touristic Web Sites, 2008).
REFERENCES


Moura, J., Pantoquillo, M., & Viana, N. (2004). Real-time decision support system for space missions control. *International Conference on*
Methodology for Improving Data Warehouse Design using Data Sources Temporal Metadata

Information and Knowledge Engineering. Las Vegas.


