ALGORITHMS FOR INTEGRATING TEMPORAL PROPERTIES OF DATA IN DATA WAREHOUSING

Francisco Araque
Department of Software Engineering, University of Jaén, Jaén, Spain

Alberto Salguero[a], Cecilia Delgado[b], Eladio Garví[b], José Samos[b]
[a] *E.T.S.I.I., University of Granada, Granada, Spain*

[b] *Department of Software Engineering, University of Granada, Granada, Spain*

Keywords: Data Warehouse, Temporal integration, Middleware Integration, Organisational Issues on Systems Integration.

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. While there are many facets to the large number of variables that are needed in the integration phase, what we are interested in is the temporal problem. It is necessary to solve problems such as what happens when data from data source A is available but data from data source B is not. This paper presents our work into data integration in the Data Warehouse on the basis of the temporal properties of the data sources. Depending on the extraction method and data source, we can determine whether it will be possible to incorporate the data into the Data Warehouse. We shall also examine the temporal features of the data extraction methods and propose algorithms for data integration depending on the temporal characteristics of the data sources and on the data extraction method.

1 INTRODUCTION

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.

For example, there is a data element *a* in the enterprise DW with a source data element b from legacy application B and a data element *c* from legacy application C. To make things more complicated, legacy application B is in IMS and legacy application C is in Adabas (Inmon, 2002).

The first complication arises when more than one condition is satisfied. In this case, the designer must decide how to make tie breakers. The second complication lies in accessing all of the legacy data in order to determine the proper value of *a*. One problem here are the resources required to examine all the data. Although in some cases, the data needed to satisfy the logic is readily available, in other cases, the data is anything but available, and lengthy calculations are needed in order to satisfy even the simplest logical condition. The third issue is one of timing. What happens when data from data source B is available but data from data source C is not?. The fourth complication is that of documenting the audit trail for the calculation of *a*. At a later point in time, an analyst might want to know the origins of *a*. In such a case, the origins are anything but straightforward. How will the DSS analyst be able to determine exactly which path of logic was used to form the basis of *a*?. The fifth complication is that of specifying temporarily available data as part of the condition set required to calculate the value of *a*.

The ability to integrate date from a wide range of data sources is an important field of research in data engineering (Haas et al., 1998). Data integration is a prominent theme in many areas and enables widely

Araque F., Salguero A., Delgado C., Garví E. and Samos J. (2006).
ALGORITHMS FOR INTEGRATING TEMPORAL PROPERTIES OF DATA IN DATA WAREHOUSING.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - DISI*, pages 193-199
DOI: 10.5220/0002493801930199

distributed, heterogeneous, dynamic collections of information sources to be accessed and handled. The use DW and Data Integration has been proposed previously in many fields. In (Haller et al. 2000) the Integrating Heterogeneous Tourism Information data sources is addressed using three-tier architecture. In (Moura et al. 2004) a Real-Time Decision Support System for space missions control is put forward using Data Warehousing technology. And in (Oliva & Saltor, 2001) a multilevel security policies integration methodology to endow tightly coupled federated database systems with a multilevel security system is presented.

It would therefore be extremely useful to have algorithms which determine 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 characteristics of the data sources and their extraction methods.

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 DW Administrator) can be integrated on the basis of the temporal characteristics of the sources (not in how this integration is carried out).

In other words, the means by which two or more data from different sources (for example, changing the formats of the data to adapt them to the DW data model) may be integrated is beyond the scope of this paper. We are at a previous stage where it is decided whether it is possible to integrate data from different sources with different temporal properties. For example, if we can obtain a data element with a 24-hour granularity (once a day) and another data element (which is to be integrated with the previous one at a subsequent stage) with a 1-hour granularity from another source, the result of the temporal integration is: the two data elements can be temporally integrated every 24 hours when both are available.

This work presents algorithms for processing information integration requirements using temporal properties of data sources. Section 2 reviews the concepts of the DW and Online Analysis Processing (OLAP). Section 3 introduces temporal conceps used in this work. Section 4 presents whether data from two data sources with their data extraction methods can be integrated. Section 5 describes the algorithms proposed. And we finish with the conclusions.

2 BASIC CONCEPTS

2.1 Data Warehouse

Inmon (Inmon, 2002) defined a Data Warehouse (DW) 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 is a highly focused DW covering a single department or subject area. The DW and data marts are usually implemented using relational databases (Hammer et al. 1995), (Harinarayan et al. 1996) which define multidimensional structures.

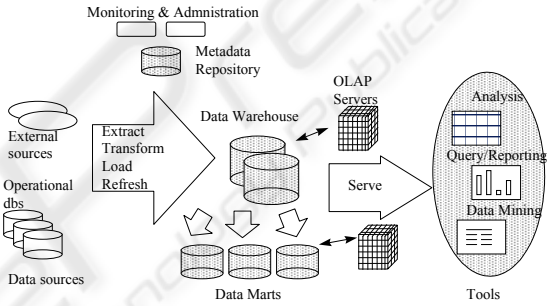

Figure 1: A generic DW architecture.

The generic architecture of a DW is illustrated in Figure 1 (Chaudhuri & Dayal, 1997), which shows that data sources include existing operational databases and flat files (i.e. spreadsheets or text files) combined with external databases. Data is extracted from the sources and then loaded into the DW using various data loaders (Araque, 2002), (Araque, 2003a). The warehouse is then used to populate the various subject/process-oriented data marts and OLAP servers. Data marts are subsets of a DW which has been categorized according to functional areas depending on the domain (addressing the problem area) and OLAP servers are software tools that help a user to prepare data for analysis, query processing, reporting and data mining. The entire DW then forms an integrated system that can support various reporting and analysis requirements of the decision-making function (Chaudhuri & Dayal, 1997).

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 & Samos, 2003), (Araque, 2003b).

2.2 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) 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.

2.3 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 data.

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 (Devlin, 1997). 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.

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 (shown below), 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
- 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.

3 TEMPORAL CONCEPTS

In order to represent the data discussed above, 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). We can represent the temporal characteristics of the data source with the temporal concepts presented in (Araque, 2002), (Araque, 2003a). 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.

3.1 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 (Bruckner & Tjoa, 2002). 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 (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

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.

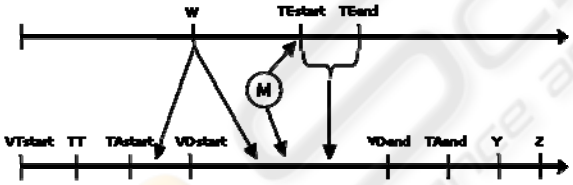

Figure 2: Temporal properties of data.

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 available 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:
  - VDstart, time instant when the availability window is initiated
  - 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:
  - TEstart, time instant when the data extraction is initiated.
  - 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, VDstart< TEstart < TEend < 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, TAstart <= VDstart and VDend <= 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).

4 TEMPORAL PROPERTIES INTEGRATION

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 (see 3.1) except TT.

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 (see 3.1).

5 ALGORITHMS

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 (fig. 3). 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. The pre-integration algorithm is out the scope of this paper (Castellanos, 1993).

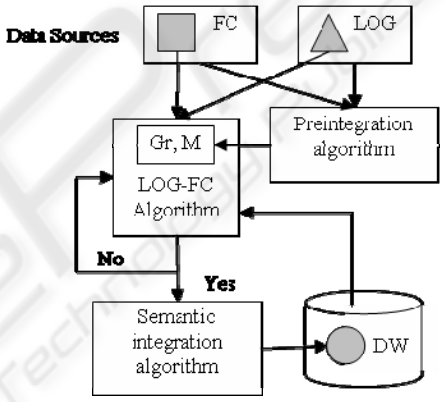

Figure 3: Integration Procress.

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 (Araque et al, 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 data sources are used to obtain this type of information:

- The US National Weather Service Website. We can access weather measurements. 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. 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.

5.1 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.

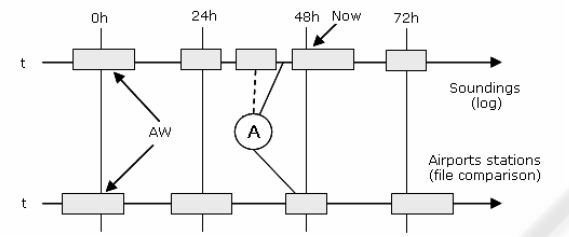

Figure 4: LOG – FC.

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 4 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:

```
available = true
If any source is not periodical
  available = CheckAvailabilityW(Log)
  available = CheckAvailabilityW & available
If availabile = true
  newValue^Fc = readValue(FC)
  If newValue^Fc <> oldValue^Fc
    newValue^Log = last log value
    If TT(newValue^Log) < M_i-1
      ;Imposible 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
```

5.2 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 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.

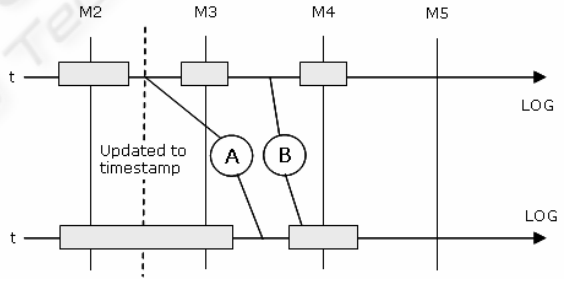

Figure 5: LOG – LOG.

Figure 5 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:

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

6 CONCLUSIONS

In this paper we have presented our work related to algorithms for processing information integration requirements using temporal properties of data sources. Data from different data sources are integrated according to the requirements of the DW Administrator. It is very useful to have algorithms which stated whether it is possible to integrate data from two data sources (with their respective data extraction methods associated), being based on the temporal properties of the data in the data sources and their extraction methods. How two or more data from different sources (for example, changing the formats of the data to adapt them to the DW data model) may be integrated is a later stage. What it is decided whether it is possible to integrate data from different sources with different temporal properties. The next step in our research will be to apply it not only to the data temporal properties, but also to include spatial properties. In sum, to integrate spatio-temporal properties of data in DW.

This work has been supported by the Spanish Research Program PRONTIN under project TIN2005-09098-C05-03.

REFERENCES

Araque, F. & Samos, J. (2003). Data warehouse refreshment maintaining temporal consistency. *5th Intern. Conference on Enterprise Information Systems, ICEIS´03.Angers.* France.

Araque, F. (2002). Data Warehousing with regard to temporal characteristics of the data source. *IADIS WWW/Internet Conference*. Lisbon, Portugal.

Araque, F. (2003a). Real-time Data Warehousing with Temporal Requirements. *Decision Systems Engineering, DSE'03 (in conjunction with the CAISE'03 conference)*. Klagenfurt/Velden, Austria.

Araque, F. (2003b). Integrating heterogeneous data sources with temporal constraints using wrappers. *The 15th Conference On Advanced Information Systems Engineering.* Caise Forum. Klagenfurt, Austria.

Araque, F., Salguero, A., and Abad, M.M. 2006. Application of data warehouse and Decision Support System in Soaring site recommendation. *Proc. Information and Communication Technologies in Tourism, ENTER 2006. Springer Verlag*, 18-20 January. Lausanne, Switzerland.

Bruckner, Robert M. and Tjoa, A M. Capturing Delays and Valid Times in Data Warehouses - Towards Timely Consistent Analyses. *Journal of Intelligent Information Systems (JIIS)*, Vol. 19(2), pp. 169-190, Kluwer Academic Publishers, September 2002.

Castellanos, M. Semiautomatic Semantic Enrichment for the Integrated Access in Interoperable Databases. *PhD thesis*, Dept. Lenguajes y Sistemas Informáticos, Universidad Politécnica de Cataluña, Barcelona (SPAIN), June 1993.

Chaudhuri, S., & Dayal, U. (1997). OLAP technology and data warehousing, *ACM SIGMOD Records*, Hewlett-Packard .

Devlin, Barry. (1997). *Data warehouse: from architecture to implementation*. Addison Wesley , c1997.

Haller, M., Pröll, B. , Retschitzegger, W., Tjoa, A. M., & Wagner, R. R. (2000). Integrating Heterogeneous Tourism Information in TIScover - The MIRO-Web Approach. *Proceedings Information and Communication Technologies in Tourism, Springer Verlag, ENTER 2000*. Barcelona.

Hammer, J., García-Molina, H., Widom, J., Labio, W., & Zhuge, Y. (1995). The Stanford Data Warehousing Project. *IEEE Data Engineering Bulletin*.

Harinarayan, V., Rajaraman, A., & Ullman, J. (1996). Implementing Data Cubes Efficiently. *Proc. of ACM SIGMOD Conference*. Montreal.

Haas L., Kossman D., Wimmers E., Yang J.: "Optimizing Queries across Diverse Data Sources", *23rd Very Large Data Bases*, August 1998, Athens, Greece.

Inmon W.H. (2002). *Building the Data Warehouse*. John Wiley.

Oliva, M., and Saltor, F. Integrating Multilevel Security Policies in Multilevel Federated Database Systems. In B. Thuraisingham, R. van de Riet, K.R. Dittrich, and Z. Tari, editors, *Data and Applications Security: Developments and Directions, pages 135–147. Kluwer Academic Publishers*, Boston, 2001.

Moura Pires, J., Pantoquilho, M., & Viana, N. (2004). Real-Time Decision Support System for Space Missions Control. *Int. Conference on Information and Knowledge Engineering*, Las Vegas, USA.