Real-time Data Warehousing with temporal requirements

Francisco Araque

E.T.S.I. Informática
University of Granada
18071 – Granada, Spain
faraque@ugr.es

Abstract. Flexibility to react on rapidly changing general conditions of the environment has become a key factor for economic success of any company. The competitiveness of an enterprise is therefore to a large extent determined by the ability of its executives to deliver exact and just-in-time decisions. These decisions must be based on high-quality, up-to-date and complete information. This paper focuses on feeding real-time data warehouses. We present our work related to minimize the delay between the time a web page changes on internet and the time this change is detected by our system. We extract data from semi-structured information sources maintaining the temporal consistency of the extracted data, and monitoring the web in accordance with the user temporal requirements.

1 Introduction

The extraordinary growth of the Internet and World Wide Web has been fueled by the ability it gives content providers to easily and cheaply publish and distribute electronic documents. Companies create web sites to make available their online catalogs, annual reports, marketing brochures, product specifications. Government agencies create web sites to publish new regulations, tax forms, and service information. Independent organizations create web sites to make available recent research results. Individuals create web sites dedicated to their professional interest and hobbies. The rapid evolution of Web pages requires making corresponding changes in the programs accessing them. In addition, most of the web information sources are created and maintained autonomously, and each others services independently. Interoperability of the web information sources remains the next big challenge.

One drawback to these sources is that there is no standard programming interface suitable for applications to submit queries. Second, the output (answer to a query) is not well structured. Structured objects have to be extracted from the HTML documents which contain irrelevant data and which may be volatile. Third, domain knowledge about the data source is also embedded in HTML documents and must be extracted.

Inmon [16] 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 data warehouse (DW) is a database that stores a copy of operational data whose structure is optimized for query and analysis. The scope is one of the data warehouse defining issues: it is the entire enterprise. Related to a more reduced scope, a new concept is defined: a data mart is a highly focused data warehouse whose scope is a single department or subject area. Related to the previous concepts, Inmon also defined the concept of operational data store (ODS) as "a subject-oriented, integrated, volatile (i.e., able to be updated), current or near-current collection of data in support of day to day operational decisions." DW and business intelligence applications are used for strategic planning and decision-making. As these applications have matured, 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 [8].

A Real Time Data Warehouse (RTDW) is an historical and analytic component of an enterprise level data stream. This data stream supports continuous, asynchronous, multi-point delivery of data. Data moves straight from the originating source to all uses that do not require some form of staging. This movement takes place soon after the original data is written. Any time delays are due solely to transport latency and (optionally) minimal processing times to dispatch or transform the instance of data being delivered.

Most of the magic in an RTDW comes from real-time data capture [9], [12], [15]. We record every meaningful event and every significant change to reference data as they occur (referred to web sites). This eliminates the most bothersome part of the whole business intelligence process – detecting and extracting changes that matter from the operational data sources. A real-time data warehouse eliminates the data availability gap. Continuous processing without delay opens up significant new opportunities for the practice of business intelligence.

This paper describes how we can use our system for capturing data from web sources and feed the real-time datawarehouse. Every data source is modeled according to its temporal characteristics that will be use to monitor it. We support monitoring at the page level as well as at the element level. *DETC* (Data Extraction with Temporal Constraints), is a system for extracting and integrating data from semi-structured web sources. Users are be able to rapidly create wrappers with temporal constraints for the Web. An application developer starts with a set of web sources (semi-structured pages), which may be located at multiple web sites, and use the output for feeding the real-time data warehouse. We think that, for web sources, it is better instead of continuous real-time data capture, use this approach to capture and monitoring the web sites that we want to use to load and refresh the RTDW according to the DW designer temporal requirements.

We think that in information extraction process it is necessary to maintain the temporal consistency [2], [3], [4], [5], [6] of the data that the user needs in the sense put forward in real-time applications. In real-time applications [17], [20] the state of the environment as perceived by the controlling system must be consistent with the

actual state of the environment being controlled. In this case we employ algorithms used to maintain coherency between a data source and cached copies of the data [26]. These algorithms try to minimize the number of times that the client has to connect to the server. Besides, we use this technique to maintain the temporal consistency between the data extracted from web information sources and the data loaded in the DW according to data warehouse designer temporal requirements [4], [26].

The metadata about information content that are implicit in the original web pages will be extracted and encoded explicitly as XML tags in the wrapped documents. The final task is to generate and construct a wrapper appropriate to each WebSource from the specification provided by the DW designer. This specification will include the temporal restrictions specified by the user (frequency of extraction, temporal constraints, adaptative algorithms, etc).

This paper is organized as follows. In section 2 we present the Real-Time Data Warehouse requirements. In section 3 we show how we can feed the RTDW with temporal constraints. Section 4 considers related work. Finally, we give a conclusion in Section 5.

2 Real-Time Data Warehouse requirements

In [9] we can find the requirements for transforming a DW to a RTDW (figure 1). Transforming a standard DW using batch loading during update windows (where analytical access is not allowed) to an analytical environment providing current data involves various issues to be addressed in order to enable (near) real-time dissemination of new information across an organization. We have to change three basic characteristics:

(i). Continuous data integration, which enables (near) real-time capturing and loading from different operational sources.

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

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

For [15] the components of a RT Data Distribution Environment are (simplified):

(a). Real-Time Capture. Data capture functionality is provided by the application or the DBMS on a transaction-by-transaction basis.

(b). Real-time Delivery. The prototypical pipeline of the real-time delivery mechanism is a cross-platform message queue.

(c). Transformation Engine. Data is transformed instance-by-instance by the continuous processing engine.

(d). The Real-time Data Warehouse. The magic of the RTDW comes from collecting data continuously in a real-time partition while sweeping a consis-

tent snapshot into the static partition at regular intervals. The real-time partition is also the host for incremental aggregation that allows key summary metrics to be built on-the-fly. The static partition functions as a traditional data warehouse. It maintains consistent history of atomic details with persistent storage of incremental net changes.

(e). Incremental Aggregator. The engine that aggregates on-the-fly.

(f). Preparation Engine. Preparation is a batch process of selecting, collecting, aggregating and projecting data to create specialized access-optimized tables. This engine reads data from the static partition of the data warehouse to populate consumer-specific data marts.

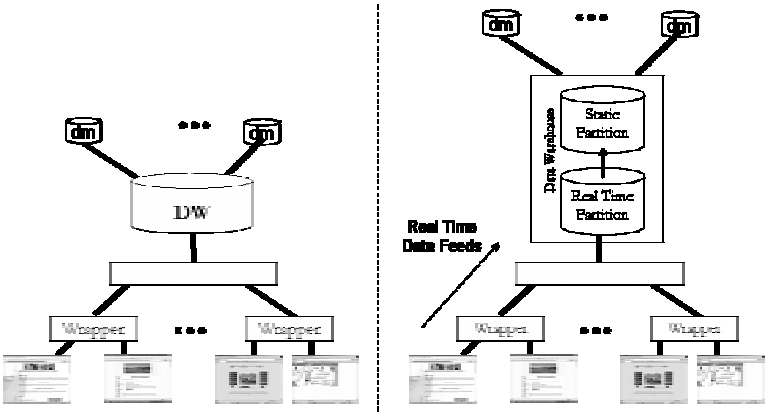

Figure 1. Data Warehouse and Real Time Data Warehouse

A real-time acquisition and delivery engine enables on-the-fly analysis while ratcheting up seamlessly to support incremental aggregation and trend-line confirmation [12]. The goal is to support immediate research of abnormal conditions in a manner not supported by the OLTP system. The enlightened perspective is that you cannot achieve point-in-time consistency without continuous, real-time data capture. The point-in-time that is relevant for one use may be radically wrong for another.

On the other hand, the closest existing decision-support systems get to real-time processing is in an operational data store, which comprises an integrated store of detailed near-current operational data. To maintain this near real-time view of operational data, ODS data-extract routines usually capture data continuously from source systems. But there are noteworthy differences between ODSs and RTDWHs [9]. An ODS serves the needs of an operational environment while real-time data warehouses serve the needs of the informational community. A real-time DWH is not a replacement for an ODS. An ODS is an architectural construct for a decision support system where collective integrated operational data is stored. It can complement a RTDWH with the information needs for knowledge workers. The ODS contains very detailed current data and is designed to work in an operational environment [9].

We think that, for web sources, it is better instead of (i) or (a), use DETC to capture and monitoring the web sites that we want to use to load the RTDW. It is more efficient to pull the web site when we need instead of continuous real-time data capture.

3 Feeding the real-time data warehouse with temporal constraints

Data warehouses are typically maintained by batch jobs that take periodic nonvolatile snapshots of operational data, and clean, transform and load the data into a warehouse database [12]. To support real-time data integration, the present batch snapshot approach to extracting operational data must be replaced by processes that continuously monitor source systems and capture and transform data changes as they occur, and then load those changes into a data warehouse in as close to real time as possible. By continuously collecting data, it is possible to analyze revenue and cost elements during any relevant time frame. Trends can be assembled with any periodicity you choose without delay.

A wrapper is a module used to translate information from the native format of the source into the format and data model used by our system. We use W4F [23], [24]. W4F is a toolkit that allows the fast generation of Web wrappers. Wrapper generation consists of Retrieval of an HTML page via GET or POST methods, followed by construction of HTML parse tree according to the HTML hierarchy. Information can then be Extracted declaratively using a set of rules applied on the parse tree. We have extended W4F with new functionalities. First, we have developed a WysiWig interface in which the HTML document is annotated in a way that the user can get the HTML path by selecting the appropriate text (select the text he/she wants to extract, as many times as necessary, and then copy it to the interface tool with ctrl+c). Second, we have added the algorithms presented in [26].

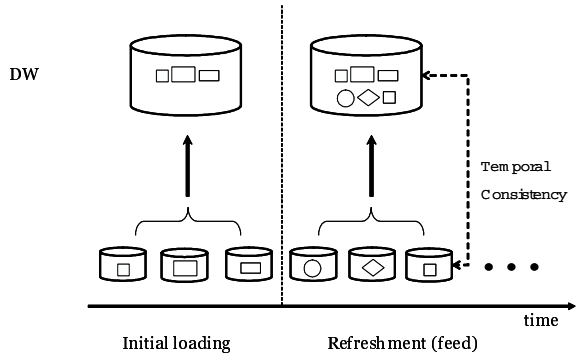

Figure 2. Temporal consistency

To deliver the data extracted we use a hybrid mode that combines the client-pull and server-push mechanisms. The transfer of information from servers to clients (RTDW administrator or designer) is first initiated by a client pull and the subsequent

transfers of updated information to clients are initiated by a server push. Clients receive the information that matches their profiles from servers.

In real-time applications [17], [20], [21] the state of the environment as perceived by the controlling system must be consistent with the actual state of the environment being controlled. Otherwise, the decisions of the controlling system may be wrong and their effects disastrous. Hence the timely monitoring of the environment, the timely processing of the sensed information, and the timely derivation of needed data are essential. Data maintained by the controlling systems and utilized by its actions must be up-to-date and temporally correlated [19]. This temporal consistency must be maintained through the timely scheduling of the actions that refresh the data. So, after the initial loading, it is important to maintain the temporal consistency in the refreshment process (figure 2).

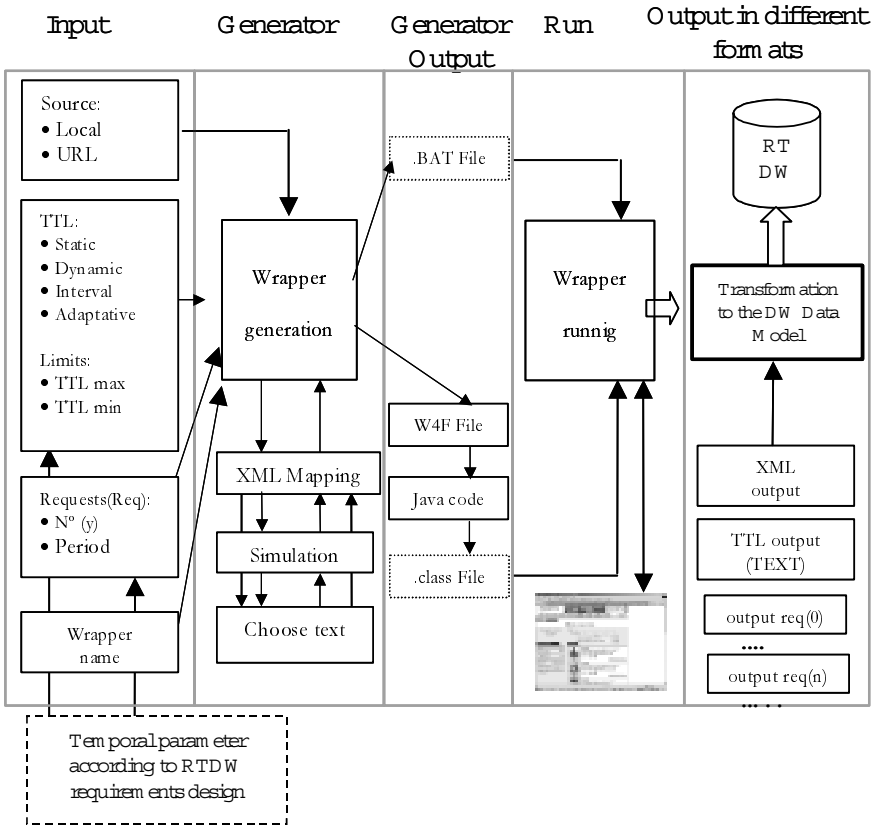

Figure 3. Functional architecture.

To achieve this goal we use a wrapper [7], [18], [19], [23], [24] and algorithms used to maintain coherency between a data source and cached copies of the data [26]. These algorithms try to minimize the number of times that the client has to connect to

the server. In the case of Web sources, the role of the wrapper is to convert information implicitly stored as an HTML document (or a set of documents), which consists of plain text with some tags, into information explicitly stored as part of a data-structure. HTML documents do not contain any type information. In the case of the Web, the wrapper also has to deal with the retrieving of information, via the HTTP protocol. Instead of having users tracking when to visit web pages of interest and identifying what and how the page of interest has been changed, the information change monitoring service is becoming increasingly popular, which enables information to be delivered while they are still fresh.

We use Time-To-Live (TTL) values, attached to cached objects (HTML pages). Upon its expiration, the source of the object can be contacted to update the page. For minimizing the incurred network overheads, the value of TTL must be high. But a low TTL value may compromise temporal consistency. Thus any TTL value must be judged depending on two factors - how well the cache consistency is maintained, and how often the remote servers are polled. Ideally, we must dynamically update this TTL value using an algorithm (adaptative) that decides the value depending on the present and past rates of source changes, with the goal of keeping remote requests to a minimum while maintaining the needed temporal accuracy of the data. We can approach how the data source evolves along time and estimating the frequency rate (for a concrete piece of text inside of a web page). We use the algorithms presented in [26] and a wrapper generation tool to maintain the temporal consistency.

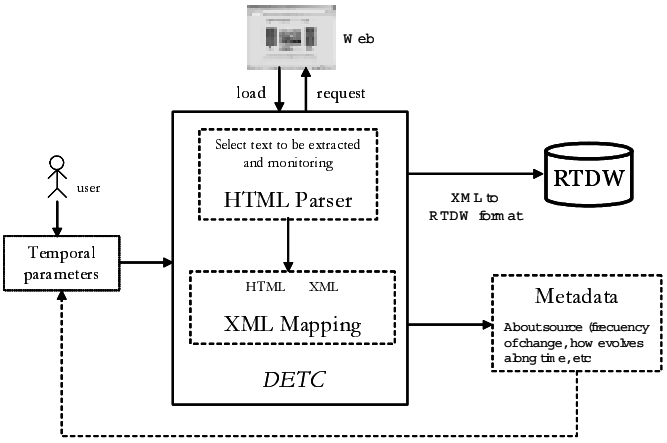

Figure 4. The steps.

The process of defining a new wrapper for a specific source is as follow (figure 4):

1. We choose the URL server that we want to extract data from. This can be done by a normal user (a client that it is using the tool) or by a specialized user (web admin, manager, ..) who wants to offer certain information to the client who connect to his server.
2. We start the tool and enter the URL choose en the first step. The tool downloads the page to the local server.

3. We can now surf the website (local copy) and locate the data that we want to ex-tract. We can select the data that want to wrap by clicking on the mouse (select operation) and copy it (or them) with CTLR+C.
4. We choose: the type of algorithm that we want to apply, the number of request and the initial time interval(t). For minimizing the incurred network overheads, the value of t must be high. But a low value of t may compromise temporal consistency. Thus any t value must be judged depending on two factors - how well the cache consistency is maintained, and how often the remote servers are polled. Ideally, we must dynamically update the value of t using an algorithm that decides the value depending on the present and past rates of source changes, with the goal of keeping remote requests to a minimum while maintaining the needed temporal accuracy of the data.
5. We have to define what is the connection between the data that we choose in the third step and the structure of the XML documents. This mapping will be used to generate the XML output file.
6. Generate the wrapper and running the wrapper.
7. Obtain the results to deliver to the user. These results (in XML), if necessary, are used to be integrated with others coming from others wrapper outputs.

If need be we can repeat the process (step 1-7) as many times as necessary. We can see the functional architecture in the figure 3. Of course, we can run as many wrappers as necessary depending on the number of sources that we want to poll in order to refresh the RTDW. Any wrapper can be parameterized with different values in accordance with the characteristics of the data source and the refreshment strategy chosen by the DW designer.

We can represent the temporal characteristics of the data source (metadata) with the temporal concepts exposed in [6]. In this way, we can know when the data source can offer the data and how this data evolves along time (temporal characteristics). After the data has been extracted from every source, and if necessary, you can integrate the XML files coming from different sources in a single XML unified view. To realize this integration you must: use the same XML mappings for the three sources, or realize a new mapping from every XML template coming from every source to the new XML global template.

4 Related work

Wrappers provide local views of data sources in a uniform data model. The local views can be queried in a limited way according to wrapper capabilities. Well-known research projects and prototypes based on this architecture include Garlic, Tsimmis [10]. A popular approach is to write wrappers to encapsulate the access to sources. For instance, the most recent generation of information mediator systems include a pre-wrapped set of web sources to be accessed via database-like queries. However, developing and maintaining wrappers by hand turned out to be labor intensive and error-prone. In addition, none of them deal with the problem of temporal consistency.

Once a DW is established, the problem of maintaining it consistent with the underlying ISs under updates remains a critical issue. It is popular to maintain the DW incrementally [1] instead of recomputing the whole extent of the DW after IS updates due to the large size of DWs and the enormous overhead associated with the DW loading process. Because the DW usually connects with ISs through the network and the network is generally not very stable, it's unacceptable to block the update transaction at an IS in order for the integrator to immediately accomplish the view maintenance process. Instead, recent work has focused on performing the maintenance in a separate process, then called deferred view maintenance [11]. The topic of DW refreshment so far has been investigated mainly in relation to techniques for maintaining materialized views [22].

Moreover, most of these approaches presume a "relational environment". This is not always true. Operational sources can be any kind of database systems. On the other hand, although the majority of projects make use of relational technology for implementing the data warehouse, there is an increasing demand for building data warehouses for "non-standard" applications like Web warehouses. In [14] can find the state of the art about DW refreshments. In [9] it is presented the characterization of RTDWHs, the identification of technologies that can support it, and the composition of these technologies in an overall architecture. They provide an in-depth discussion of using the J2EE platform for ETL environments of RTDWHs. The ETL tool presented in [27], namely Arktos, is capable of modelling and executing practical ETL scenarios by providing explicit primitives for the capturing of common tasks. Arktos provides three ways to describe an ETL scenario: a graphical point-and-click front end and two declarative languages. Xyleme is a system for monitoring XML data on the web [28]. It is a dynamic warehouse for XML data of the Web supporting query evaluation, change control and data integration. Related to poll the data sources, Xyleme tries to minimize the delay between the time an XML page changes on the web and the time this change is detected by the system. They estimate the frequency rate based on last date of change, but a page level. For us it is important to support monitoring at the page level as well as at the element level (a piece of text in a page). In this sense, a very interesting project is WebCQ [29]. With WebCQ it is possible monitoring and tracking various types of changes to static and dynamic web page changes and personalized delivery of page change notifications.

5 Conclusion and future work

We can use the output of the wrapper (this data have been extracted with the restriction of temporal consistency specialized for the user) for several purposes:

- To deliver to the user or client in real-time.
- To store in a database for future use. For example to query. This step requires a conversion from XML to the native format of the database.
- To integrate with other information to obtain a unificated vision of one or more sources.

- To load the RTDW for future analysis or data mining.
- To notify to the user when a specific value (or threshold) has been reached.

Besides, we use this technique to maintain the temporal consistency between the data extracted from web information sources and the data loaded in the data warehouse according to data warehouse designer temporal requirements.

We have exposed our current work in information retrieval and integration considering the temporal requirements of the user. Moreover, the observation of real-world events by computer systems is characterized by a delay and it is important to keep the temporal consistency between the data source that feed our system and the data extracted from the data source, according to temporal designer requirements.

As future work we are working to incorporate valid time and transaction time in the sense put forward in [13]. Besides we want to add the real time data warehouse characteristics to our current data warehouse architecture (figure 5) [4], [6], [25].

This work has been supported by the Spanish Research Program PRONTIC under project TIC2000-1723-C02-02.

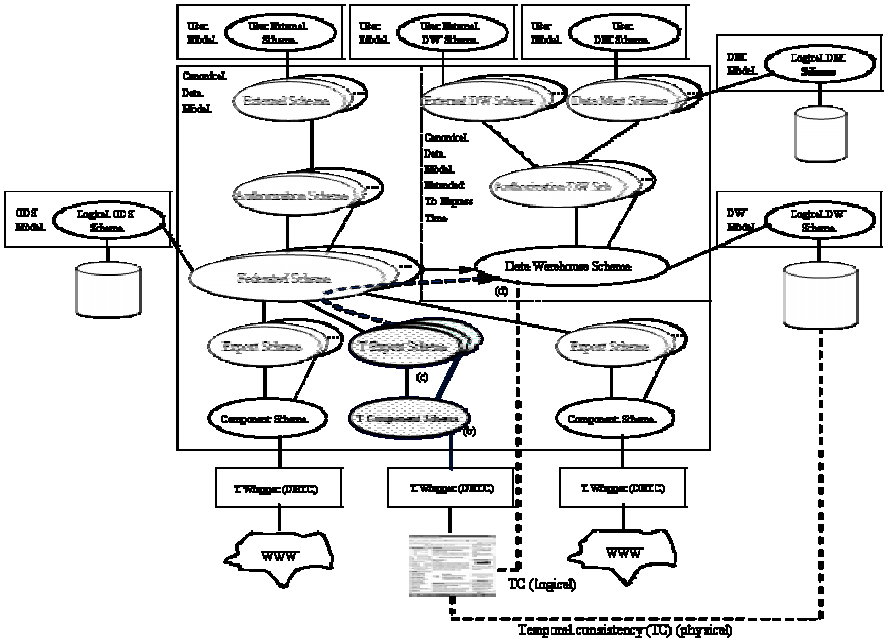

Figure 5. New architecture extended to express time

References

1. Agrawal, Abbadi, and Singh. Ecient View Maintenance at Data Warehouses. SIGMOD, pages 417-427, 1997.

2. Araque, F, Samos. J., 1999: External Schemas in Real-Time Object-Oriented Databases. 20th IEEE Real-Time Systems Symposium, WIP Proceedings, pp. 105-109 (Phoenix, AZ, 12/1999).
3. Araque, F. Integrating heterogeneous data sources with temporal constraints using wrappers. The 15th Conference On Advanced Information Systems Engineering, Caise Forum. Caise´03. Klagenfurt/Velden, Austria, 16 - 20 June, 2003.
4. Araque, F., 2002: Data Warehousing with regard to temporal characteristics of the data source. IADIS WWW/Internet Conference. 13-15 November, 2002. Lisboa, Portugal.
5. Araque, F., 2002: Personalized data extraction with temporal constraints. IADIS WWW/Internet Conference. 13-15 November, 2002. Lisboa, Portugal.
6. Araque, F., Samos, J. Data warehouse refreshment maintaining temporal consistency. 5th International Conference on Enterprise Information Systems, ICEIS´03. Angers, France, 23-26, April 2003.
7. Bhandari, D. Extraction Of Web Information Using W4F Wrapper Factory and XML-QL Query Language. Technical Report, Universitu of Pennsylvania, 1999.
8. Breitner, C. Data Warehousing and OLAP: Delivering Just-In-Time Information for Decision Support. Proceedings of the 6th Intl. Workshop for Oeconometrics. Juni 1997, Karlsruhe, Deutschland.
9. Bruckner, Beate List, and Josef Schiefer. Striving towards Near Real-Time Data Integration for Data Warehouses. Y. Kambayashi, W. Winiwarter, M. Arikawa (Eds.): DaWaK 2002, LNCS 2454, pp. 317-326, 2002. Springer-Verlag Berlin Heidelberg 2002.
10. Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., and Widom J.: "The TSIMMIS Project : Integration of Heterogeneous Information Sources", IPSJ Conference, pp. 7-18, Tokyo, Japan, October 1994.
11. Colby, L. S., Grin, T., Libkin, L., Mumick, I. S. and Trickey, H.. Algorithms for Deferred View Maintenance. SIGMOD, pages 469-480, 1996.
12. Colin White, Intelligent Business Strategies: Real-Time Data Warehousing Heats Up. DM Review, August 2002.
13. Fabio Grandi, Federica Mandreoli, The Valid Web: an XML/XSL Infrastructure for Temporal Management of Web Documents, Proc. ADVIS 2000 - Intl' Conf. on Advances in Information Systems , Izmir, Turkey, October 2000, LNCS 1909, © Springer-Verlag, Berlin, 2000, pp. 294-303.
14. Fabret, M. Matulovic, and E. Simon, Data Warehouse Refreshment: State of the Art, DWQ T Report. 1997.
15. Haisten, M. Real-Time Data Warehouse: The Next Stage in Data Warehouse Evolution. DM Review, August 1999.
16. Inmon, W., 1992: “What is a Data Warehouse?.” PRISM Tech Topic, Vol.1, No.1 (1992).
17. Kao, B., Garcia-Molina, H., 1995: “An Overview of Real-Time Database Systems.” In S. Son (Ed.), Advances in Real-Time Systems, chapter 19. Prentice Hall, 1995.
18. Liu, Calton Pu, Wei Han, David Buttler, Wei Tang, 1999: An XML-based Wrapper Generator for Web Information Extraction. Proceedings of the ACM SIGMOD International Conference, June 1-4, 1999, Philadelphia.
19. Liu, Calton Pu, Wei Tang, 2000: WebCQ: Detecting and Delivering Information Changes on the Web In the Proceesings of International Conference on Information and Knowledge Management (CIKM), Nov. 7-10, 2000 Washington D.C., ACM Press.
20. Ramamritham, K., 1993. “Time for Real-Time Temporal Databases?” In Proc. Int’l. Workshop on an Infrastructure for Temporal Databases (June, 1993).
21. Ramamritham, R. Sivasankaran, J. A. Stankovic, D. T. Towsley and M. Xiong, 1996: Integrating Temporal, Real-Time, and Active Databases, ACM Sigmod Record, Vol. 25. No. 1, March 1996, pp. 8-12.

22. Roussopoulos. Materialized Views and Data Warehouses. SIGMOD Record, 27(1), p21-26, March 1998.
23. Sahuguet A. and Azavant F, 1999: Web Ecology: Recycling HTML pages as XML documents using W4F. Informal proceedings of the ACM International Workshop on the Web and Databases (WebDB'99) Philadelphia, Pennsylvania, USA- June 3/4, 1999.
24. Sahuguet, A. and Azavant, F., 1998: W4F: a WysiWyg Web Wrapper Factory, Technical report from the Penn Database Research Group, 1998, University of Pennsylvania.
25. Samos, J., Saltor, F., Sistac, J., Bardés, A., 1998: "Database Architecture for Data Warehousing: An Evolutionary Approach." In G. Quirchmayr et al. (Eds.): Proc. Int'l Conf. on Database and Expert Systems Applications (Vienna, Aug. 1998), Springer-Verlag, pp. 746-756.
26. Srinivasan, Chao Liang, and K. Ramamritham, 1998: Maintaining Temporal Coherency of Virtual Warehouses (abstract). The 19th IEEE Real-Time Systems Symposium (RTSS98), Madrid, Spain, December 2-4, 1998.
27. Vassiliadis, Zografoula Vagena, Spiros Skiadopoulos, Nikos Karayannidis, Timos Sellis. Arktos: Towards the modeling, design, control and execution of ETL processes. Information Systems, vol. 26, no. 8, pp. 537-561, December 2001, Elsevier Science Ltd.
28. Benjamin Nguyen, Serge Abiteboul, Gregory Cobena, Mihai Preda. Monitoring XML Data on the Web. ACM SIGMOD 2001 May 2124, Santa Barbara, California, USA.
29. Ling Liu, Calton Pu, and Wei Tang. WebCQ: Detecting and Delivering Information Changes on the Web" In the Proceesings of International Conference on Information and Knowledge Management (CIKM), Nov. 7-10, 2000 Washington D.C., ACM Press.