ZERO-LATENCY DATA WAREHOUSING FOR HETEROGENEOUS DATA SOURCES AND CONTINUOUS DATA STREAMS

Tho, M. Nguyen; Tjoa, A. Min[1]

Abstract.

In this paper, a framework for building an overall Zero-Latency Data Warehouse system (ZLDWH) is provided. Such a ZLDWH requires tasks such as data changes detection, continuous loading and updating of new data, autonomous analysis and the execution of actions to be completed in the Data Warehouse (DWH). For this purpose, we combine (1) an existing solution for the continuous data integration and (2) the known approach of Active Data Warehousing (ADWH) by introducing protocols that enable the correct collaboration between the two. The continuous data integration is realized on the one hand by asynchronous communication between multiple information sources and the DWH through a message queuing system, and on the other hand by the continuous loading and updating. To obtain a low-latency time for this cycle, an "active rule evaluation" is provided between the data feeding stage and its loading into the DWH. This is realized by applying ECA-Techniques (Event-Condition-Action) to obtain a more "active" characteristic of the DWH.

A second goal of this paper is to take into consideration a special class of information source namely continuous data streams. Continuous data streams are information sources in which data arrives in high-volume, rapidly, bursty, and needs to be processed continuously. Therefore novel protocols and techniques are needed for capturing, storing and evaluate analysis decision based on new coming data from such continuous data stream sources.

1. Introduction

Data warehouses (DWHs) [9,10] are special-purpose database systems which are dedicated to collecting and storing data from on-line transaction processing (OLTP) systems to provide a profound database for statistical analysis and managerial decision making. While operational systems are designed to meet well-specified response time requirements, the focus of DWHs is generally the strategic analysis of data from heterogeneous systems. The data integration process is very complex and covers the acquisition, extraction, transformation (staging area), and loading of the data into the DWH. Traditionally there is no real-time connection between a DWH and its data sources, because the "write-once read-many" decision support characteristics would conflict with the continuous update workload of operational systems. Data loading is performed during frequent update time-windows (e.g. once a day, once a week, or every night), when the analysis capabilities of DWHs are not affected. Users, knowledge workers, or sometimes, other decision systems will then access the DWH with the help of

[1] Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstr. 9-11 /188, A-1040 Vienna, Austria, {tho, tjoa}@ifs.tuwien.ac.at

OLAP tools [9,10,26] to gather information for business decision tasks. Obviously, such a traditional approach cannot afford the requirement of real-time decision support systems because of the latency from the time point the events occur to the time point the decisions are made.

The aim of a zero-latency or near-real time Data warehouse (ZLDWH) [5,6,17] is to allow organizations to deliver relevant information as fast as possible to applications which need a near real-time action to new information captured by an organization's information system. Some systems can even react automatically to the events without user intervention. Traditional DWHs focus on strategic decision support and usually do not update data frequently enough to react with the data changes in the operational systems. The ZLDWH therefore needs some extensions for tactical decision support. In such a ZLDWH, not only the *continuous data integration* is required, but also the assembling of *active rules*, a technique derived from Active Database [25], into the DWH. In general, the most requirements of the ZLDWH are:

• *Data freshness*: data must be updated more frequently in order to improve the freshness of data.

• *Continuous data integration*: (Near) real-time capturing and loading of data from heterogeneous, distributed data sources. Here data conversion and propagation between different formats must also be taken into consideration.

• *Time consistency for stored data*: the effects for some late-arriving data in the continuous data integration environment could be so more critical that we need a systematic temporal approach to deal with this kind of data.

• *Active action/notification*: Active rules are defined to analyze the data collected by the DWH and to issue suitable actions (invoking stored procedures in OLTP systems) or to generate notifications to the knowledge workers as fast as possible.

• *High availability*: A highly available analytical environment, which operates 7x24, can consistently generate and provide access to current business analysis at any time.

• *Performance and scalability*: the number of users and performance requirements for the zero-latency DWH will increase with the deployment of analytic applications enabling tactical decision supports.

This paper outlines the framework of developing an overall ZLDWH that integrates the *continuous data integration* and the *active rules technique* for making the DWH more "active". The system therefore can deal with real-time data capturing, continuous loading and updating new data, autonomous analysis and executing the actions to be completed in the DWH. We will also introduce novel protocols and techniques for capturing, storing and performing analysis for decisions based on new arriving data from a special class of information source namely *continuous data streams*. Other requirements such as high availability, performance and scalability are out of the scope of this paper. The remainder of this paper is organized as follows. Section 2 considers related work. The global system architecture will be introduced in Section 3. Section 4 describes the Continuous Data Integration module. The Autonomous Active Decision Engine is described in Session 5. Protocols and techniques dealing with continuous data streams will be introduced in Section 6. Section 7 deals with time consistency issues. Finally, we give a conclusion and future works in Section 8.

2. Related Work

There are several approaches from both academia and the industrial community for realizing zero-latency information systems. *Compaq's Zero Latency Enterprise (ZLE) framework* [12] is centered around an operational data store (ODS) as the primary data repository instead of a DWH. NCR's Teradata division has developed the concept of the *@ctive Data warehouse* [27] which essentially marries the operational data store (ODS) and DWH concepts. Teradata also offers a variety of continuous data integration utilities (e.g. FastLoad, MultiLoad, Tpump). *Active rules* [1,3,4,11,25] have been widely accepted to achieve the goal of auto-decision-making. It minimizes (or eliminate, after all) users' intervention in processing a series of complex tasks and managing data automatically with the verification of integrity constraint, transaction, security, etc. Typically, active information management has rule processing features based on an Event-Condition-Action (ECA) construct by which complicated business strategies are modeled and processed without looking into application programs and the underlying database. Researches in *active databases* (ADBs) [25] extend the power of active rules to react to events and conditions in the database. ADBs offer a wide range of events from different sources, support variant action execution, but only accept condition as a simple Boolean predicate or a query. ARML [11] provides an XML-based uniform rule description for the sharing of business rules among heterogeneous active rule processing systems .Bailey et al. [3] suppose the ECA language for XML repositories to incorporate reactive functionality in XML setting. Active View for Electronic Commerce [1] also applies active rules in declarative view specification language. In [4] the authors present the use of ECA rules in Object-Oriented Database system. Applying triggers for active rules evolution and implementation was described in [28].

Active Data Warehouses (ADWHs) [5,28] are systems which use event-condition-action rules (ECA) or other event-driven mechanisms in order to carry out routine decision tasks automatically within a DWH environment. Thalhammer [28,29] adopted ECA rules to mimic the work of analyst, so he calls them *analysis rules*. His approach combines ADBs, DWHs, and OLAP to automate decision processes for which well-established decision criteria exist. However, the data integration process is based on traditional batch load, and concerns that data warehouse state constant during a cycle. Obviously, it does not cope with near-real time data integration as well causes some major problems in case of late-arriving data and decision-making based on incomplete information. Qtool [5], Bruckner's solution for zero-latency data warehousing is based on the continuously data integration using Microsoft Message Queuing (MSMQ) and Teradata Tpump utilities [16]. Although data were loaded and integrated continuously, QTool does not deal with other problems such as feeding data from heterogeneous data sources, detecting data changes, or implementing the active decision engine. *Message queuing* provides reliable, secure, time-dependent, cross-platform application messaging services. Communication protocols are hidden from the applications. Message queues work asynchronously, and enable a decoupling of message-streams and temporarily operations. Examples of message queuing products are Oracle Advanced Queuing (AQ)[24], MSMQ [21,22], IBM's MQ Series.

In many applications such as network monitoring, sensor processing, telecommunications, financial analysis, Web tracking and others, data may take the form of *continuous data streams* (rather than finite stored data sets) and queries are long-running, continuous and real-time. There has been a surge of interest recently in the area of query processing over continuous data streams [8,14,20], and other related problems (resource management, approximately computation, architectures) [7,15,23,30,31,32].

3. Architecture of the ZLDW

A ZDLWH aims to significantly decrease the time to react to business events. It enables analysis across corporate data sources and notifies the handling of actionable recommendations. Such a ZDLWH requires an *Integrated Analysis Environment,* which minimizes the latency between the causes (data changes, transactions, system down, etc) and the effects (notifications, recommendations, operational actions, etc) of a business decision. For this purpose, the traditional "passive" DWH approach must be adapted to obtain the freshness of data sources and the "active" characteristic of the DWH.

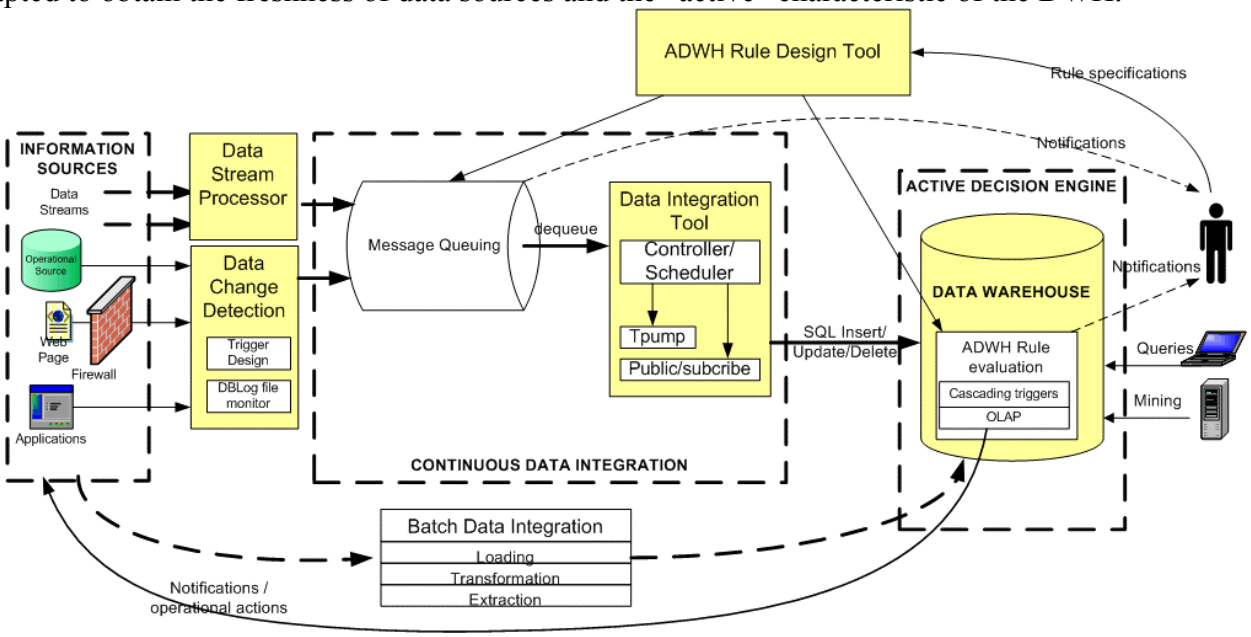

Fig. 1. The overall structure of the Zero-Latency Data warehouse System.

Our proposed system integrates the continuous data integration approach [5] and the ADWH concept [28] by providing protocols that enable them to correctly collaborate. Figure 1 depicts the overall structure of the ZLDWH. Data arrives from heterogeneous information sources. The Data Stream Processor taps the continuous data streams, and after some processing put them into the message queuing system. The Data Change Detection module detects changing data in other information sources and also feeds this new data into the message queuing system. *The first main component* of the system is the Continuous Data Integration module that feeds the data asynchronously, continuously from the heterogeneous information sources into the DWH. This module will be described in more detail in Section 4. Because not all data needs to be continuously updated and integrated, the traditional Batch Data Integration is still necessary in the system. The ADWH Rule Design tool allows the knowledge workers to specify active rules to deal with some kinds of events and cause some actions or notification at the stage of data feeding. More importantly, the knowledge workers can specify the analysis rules in the DWH to deal with decision tasks that needs incremental multidimensional analysis. The evaluation of the analysis rule causes cascading triggers in the DWH leading to suitable notifications and actions (i.e. *operational actions* in the operational data source or *notifications/ recommendations* sent back to the analysis workers). These tasks are accomplished by *the second main component*, Active Decision Engine, which deals with autonomous active rules evaluation and reaction. Section 5 will discuss the technique used in this component in more detail. It is obvious that other services in traditional, passive DWH such as OLAP queries or request from data mining systems are still supported in the ZLDWH.

4. Continuous Data Integration

Traditional DWHs are populated by performing an initial data load, which is then followed by periodical update processes. During the update processes (the so called update windows), analytical queries are not processed. A ZLDWH therefore must support the continuous and near real-time query processing. There are several approaches to realize this purpose. To minimize the update window, Labio et al in [19] try to use bulk loading tools to achieve high performance for data integration. Recently published scientific approaches try to identify optimal strategies for updating single materialized views of DWHs by minimizing the whole update work. Other approaches focus on Data/Table Swapping [18] in which the ETL tools take for granted that they have free reign to drop tables, re-load table and conduct other major database system without disturbing simultaneous end-user queries. The downside of this type is that the data in the warehouse is not truly real-time. For applications where true real-time is required, the best approach is to continuously trickle-feed the changing data from the source system directly into the DWH. *Continuous Data Integration* processes real-time data flows from transactional systems and integrates them continuously into the DWH. The data integration process is performed in parallel with complex analytical processing in the DWH. Parallel processing capabilities become a very important agent in such processes.

There are several styles of data delivery mechanisms that can be compared by considering three main characteristics: (1) push vs. pull, (2) periodic vs. aperiodic, (3) unicast vs. 1-to-N. Push technology stems from the very straight forward idea that rather than requiring systems to explicitly request (i.e. "pull") the information needed, data can be sent to a system without being requested. The request-response style is pull-based, because the transfer of information from servers to clients (e.g. from source system to ETL tool) is initiated by a client pull. Both push and pull can be performed in either an aperiodic or a periodic fashion. Aperiodic delivery is event-driven, i.e. a data request (for pull) or transmission (for push) is triggered by an event such as a user action (for pull) or data update (for push). In contrast, periodic data delivery is performed according to some pre-arrange schedule. With unicast communication, data items are sent from a data source to one other machine, while 1-to-N communication allows multiple machines to receive the data sent by the source. A data integration process for traditional DWH works with unicast communication, because there is only one receiver. Two types of 1-to-N data delivery can be distinguished: multicast and broadcast. With multicast, data is sent to a specific subset of clients who have indicated their interest in receiving data so the data recipients are known. In contrast, broadcasting sends information over a medium on which an unidentified and possibly unbound set of clients can listen. The classification of Data Delivery Options is shown in Table 1 below.

Pull vs. Push	Aperiodic vs. Periodic	Unicast vs. 1-to-N	Data Delivery Option
Pull	Aperiodic	Unicast	Request/Response
		1-to-N	Request/Response with snooping
	Periodic	Unicast	Polling
		1-to-N	Polling with snooping
Push	Aperiodic	Unicast	-
		1-to-N	Publish/Subscribe
	Periodic	Unicast	e.g. E-mail sending
		1-to-N	e.g. E-mail list digest

Table 1. Classification of Data Delivery Options

Pull mechanisms are quite common in traditional data integration approaches for DWH implementations. Realizing that both, aperiodic pull (request/response) and periodic pull (polling) can put additional load on operational sources, these problems get even worse when we want to integrate data continuously in near-real time. Push mechanism are ideal for operational systems, because they can control themselves when and where to push the data. Since every event is significant, the loss of a single event can lead to loss of synchronization in state between a source system and the DWH. Messages containing event information must be queued and removed after reading to ensure that every event is processed exactly once. With the discussions above and addressing the two identified problem areas, we propose the architecture given in Fig. 2 for near real-time data integration.

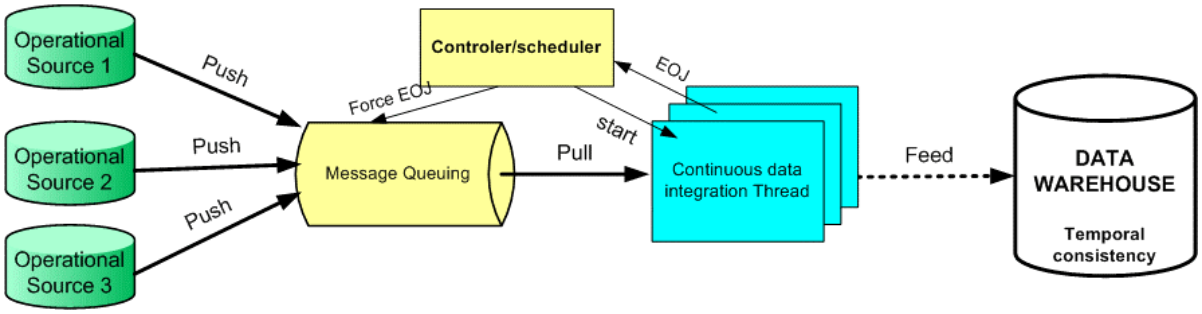

Fig. 2. Near Real-time Data Integration Architecture.

The operational data source systems push their events (data changes) locally to an asynchronous message system. By this way, the recipients do not interfere operational systems. The messaging infrastructure provides reliable data transport from source system and support basic publish/subscribe mechanism for data delivery to recipients. Furthermore, it works asynchronously, which enables a decoupling of message-streams and temporarily disconnected operation. Transaction data can be placed into the message queue in real-time. The actual parallel data integration threads controlled by controller/scheduler module will use a pull (request/response) approach when accessing the message transport system. The controller/scheduler is responsible for starting and stopping these threads, and managing the post-job processing followed by each thread. The control is then transferred back to the data integration controller rather than to the data providers. Therefore, ETL tools are able to work with their most efficient mode (aperiodic pull) processing event-driven data continuously. We are able to establish temporal order during data integration by using an underlying data model that provides temporal consistency. Therefore, we can (1) systematically deal with late-arriving data and (2) even support priority-based message transport and delivery (see more detail in Section 7).

5. Autonomous Active Decision Engine

In the traditional approach, a DWH extracts, transforms, and loads (ETL) data from operational source systems autonomously without user interaction. After the data is loaded, analysts perform interactive analyzes using OLAP tools to find solutions for different kinds of decision tasks. The decision tasks can be classified into three styles [29] (1) “non-routine decision tasks”, (2) “routine decision tasks” and (3) “semi-routine decision tasks”. In the case of “non-routine decision tasks”, analysts query the DWH to create different scenarios and then make the decision. Such tasks do not occur regularly and there are no generally accepted decision criteria. Some decisions in strategic business management such as setting up the new product or changing the business policy are examples of “non-routine decision

tasks". In the case of "routine decision tasks", analysts extract information from the DWH to solve well-structured problems by applying generally accepted procedures in decision making. Such decision 4tasks occur frequently at predictive time points. Examples can be found in areas of customer management (e.g. grant the discount to special customers) product management (e.g. change price or withdraw products). The "semi-routine decision tasks" typically occur when routine problems require non-routine treatment (e.g. if there are contradictory in decision before withdraw a product need to be discussed more before making the decision).

The main function of *Active Decision Engine* is automatizing decision-support activities of "routine decision tasks" and the routine elements of "semi-routine decision tasks". The autonomous reaction processes are based on complex incremental multi-dimensional analysis of the collected data in the DWH, which was usually implemented manually by the analyst using OLAP tools. The Active Decision Engine combines DWH, ADB, and OLAP to automatize decision processes that occur on a regular schedule and for which well-established decision criteria exist. The "glue" among these technologies are analysis rules. They realize the basic ECA rule structure from ADB, but carry out complex OLAP analyzes on warehouse data instead of evaluating simple conditions as in case with ECA rules in OLTP system. Like ECA rules, the *event part* of an analysis rule represents the time points at which decision processes should be initiated. However, compared to ECA rules in ADB which offers a variety of events [25] (e.g. structure operation, behavior invocation, transaction, abstract or user-defined, exception, clock, external), the scope of the event part of analysis rules is limited (e.g. OLTP method event, absolute and relative temporal event). The *condition part* represents the incremental multidimensional analysis that are usually carried out manually by the analyst using some OLAP tools. In contrast to conventional ECA rules, the condition part in analysis rules is not a simple Boolean predicate but consists of hierarchically organized predicates that require the incrementally performing of multi dimensional analysis to evaluate. The *action part* of an analysis rule is the directive to execute a particular transaction in the OLTP system, representing the decision for which analyzes have been carried out. The action could also be a recommendation or notification to the users for further analysis in the case it can not make the decision positively. Compare to the action part of ECA rules in ADB [25], which maybe an arbitrary sequence of DML statements update the database or rule set, transaction statements, behavior invocation, external calls, do-instead, etc., we can stay that the actions which analysis rules can cope are more simple. Fig. 3 depicts the logical structure of the active decision engine and the various components communicate with it.

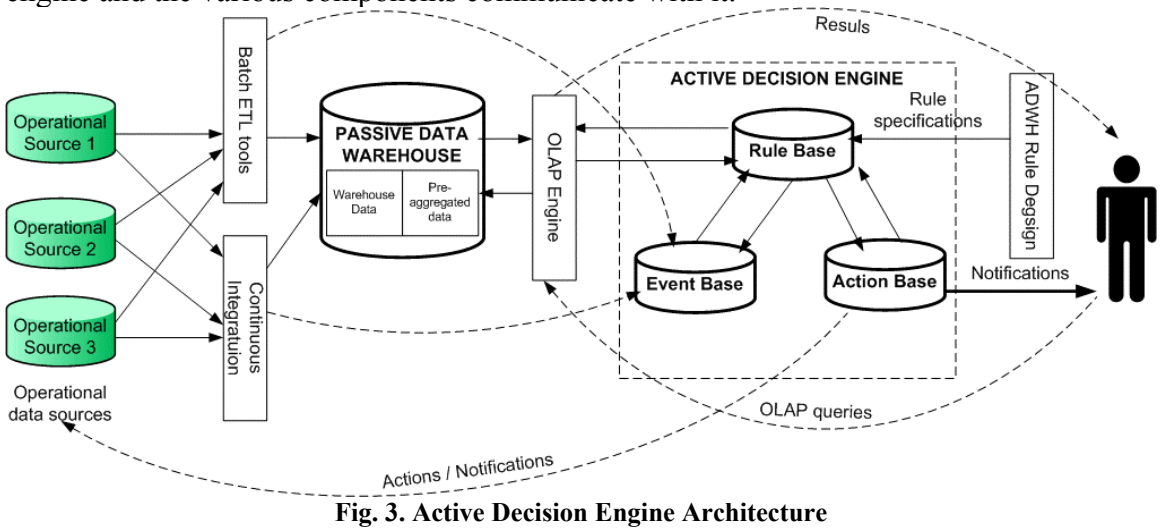

Fig. 3. Active Decision Engine Architecture

Data is extracted, loaded, transformed and integrated in the DWH through the batch ETL or the continuous data integration (described in section 4). Besides loading conventional passive warehouse data, changes in the operational data sources (OLTP events) are also loaded into the *data event repository* (the event base in figure 3). These events are used to trigger of other complex types of events (e.g. temporal events). Within the *active decision engine*, the temporal events (e.g. end of month) will be used to determine when analysis rules should be fired. The passive DWH, which maintains integrated data and pre-aggregated data (e.g. materialized views) is responsible to answer two types of requests: (1) The ad-hoc OLAP request issued by analysts using conventional OLAP tool, (2) Pre-defined OLAP requests issued by analysis rules when these rules are executed. Active decision engine includes three basic components. The *Rule base* stores executable rule definition and meta data about these rules. The *Event base* maintains occurrences of events that fire the analysis rules. The *Action base* maintains all information about the actions can be happened (e.g. specifications of the action, parameter binding, etc.). The ADWH rules-design-tool provides a graphical user interface with which the users can define new rules, update existing rules, inspect generated decisions and modify the event base. Analysis rules are modeled from the perspective of the analyst who specifies: (1) the time points at which the analysis rule has to be fired (*the event*); (2) the data cubes that have to be inspected (*the analysis graph*) by using certain criteria for decision making (*the decision steps*); and (3) the transaction that has to be executed for a particular entity in an OLTP system if one of the decision criteria is satisfied (*the action*). The data cubes in the analysis graph will be analyzed to evaluate some criteria defined by the Rule Base, and if certain decision criteria are satisfied, the action (in the Action Base) will be realized without user interaction. As concerned in [29], analysis rules can be realized by cascading triggers, which are available in most of commercial database system nowadays.

6. Continuous data streams processing

Data may take the form of continuous data streams, rather than finite stored data sets in several applications such as network monitoring, sensor processing, Web tracking, telecommunications, and financial analysis. A stored dataset is appropriate when significant portions of the data are required again and again, and updates are small and/or relative infrequent. In contrast, a data stream is appropriate when the data is constantly changing and it is either unnecessary or impractical to operate on large portions of data multiple times. Stored datasets relate to traditional DWH, whereas the data streams relate to the continuous data integration and require minimized latency for DWH. However, the research area of continuous queries over data streams is quite young, and mainly focuses on DB architectures [7,30,32] and query processing techniques [8,14,31]. We now describe a framework that allows the ZDLWH to process and feed the continuous data streams into the DWH while still supports the continuous and ad-hoc queries. This purpose can be obtained by storing data stream (with suitable timestamps) into the message queuing system, and then using continuous updating tools to add new data to DWH with time consistency. All the data needed to evaluate the queries will be stored in some views that stored both individual data and pre-aggregation data (with consistent timestamp) in DWH. Whenever new data streams arrive, the views will be re-calculated, new data will be added into the DWH with suitable timestamp and the queries will be evaluated again based on these updated views.

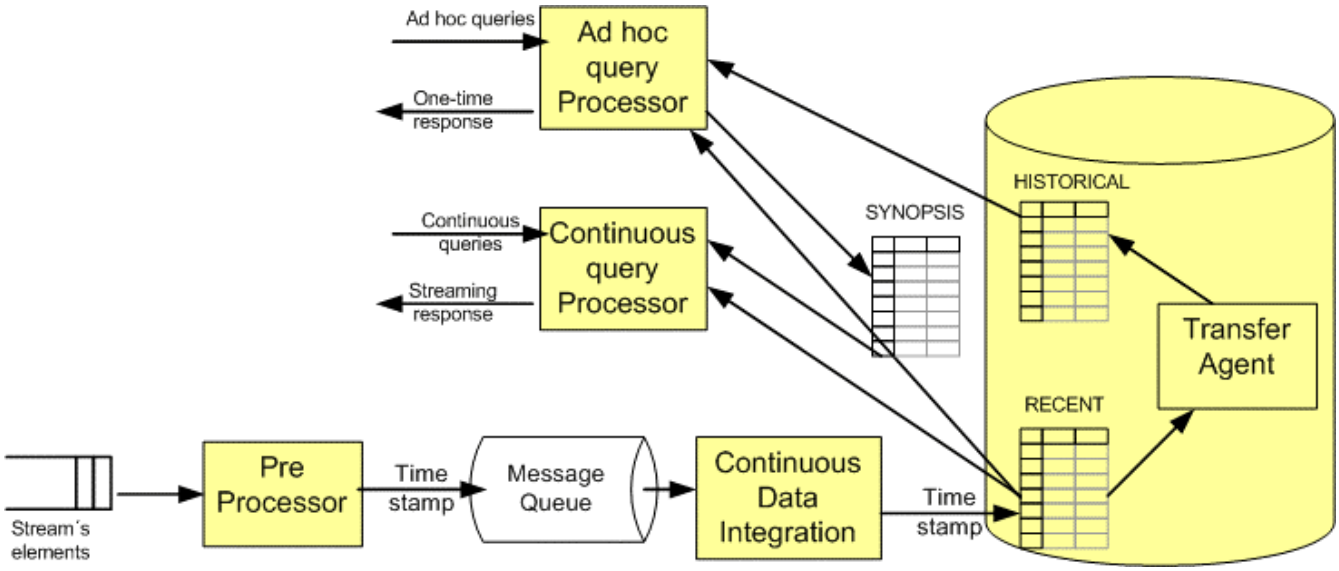

Fig. 4. Continuous data streams processing

Figure 4 describes the framework of continuous data stream processing. Rows arrives as a set of streams. The Pre-Processor taps into those streams and extracts new data elements. It will then add the timestamp to those data elements and put them into the message queue system. The continuous data integration feeds the new data into the DWH with suitable timestamps. To make the queries processing efficiently, recent data will be stored in RECENT views, and historical data will be put into HISTORICAL views. These views will be organized in a well structured form organized by security and timestamps to support rapid querying. Therefore, new data elements from message queue system will be fed into RECENT Views, and they will also update some aggregation data available in RECENT views. Eventually, data in RECENT views become aged. Whenever that happens, or at strategic moments (e.g. low system load), the Transfer Agent is responsible for purging the data from RECENT, and inserting data into HISTORICAL views. Ad hoc queries are handled by the Ad hoc query processor. This module is responsible for breaking down an ad hoc query into component queries that can be executed against RECENT and HISTORICAL data. It then creates a synopsis data structure that stores a synopsis of older information for many possible queries and passes the queries to the continuous query processor. The continuous query processor accesses to RECENT views and synopsis data structure to adapt with recent data, so the answer to the query can be maintained as updates arrives.

7. Time consistency issues

The presence of time and the dependence upon it is one of the properties that sets DWH applications apart from traditional operational system. In a DWH, time affects the structure of the system. In continuous data streams, a timestamp or a sequence number attribute represents a tuple´s arrival time that will affect the processing windows. Therefore, how to grant the correct timestamp to data tuples and how to store data into the DWH in time consistency become essential issues in our ZDLWH. For this purpose, we need a time consistency model that suits well for both DWH data and data streams tuples. The authors in [6] suggest an approach for modeling conceptual time consistency problems and introduce a data model that deals with timely delays in the active DWH environment. Their approach

bases on an enhanced time dimension model for DWHs to accomplish time consistency. This time dimension includes following time intervals and timestamp:

• *Valid time (validity of knowledge)*. This property is always related to a base event at an instant T (stored as dataset), and describes the time interval of validity (knowable at instant T) of that piece of information.

• *Revelation time (transaction time)***.** The revelation time describes the point in time, when a piece of information was realized by at least one source system.

• *Load timestamp*. The load timestamp is a time value associated with some integrated dataset in a DWH environment and determines the instant, when a fact is actually knowable to active mechanisms.

In Models and Issues in Data Stream Systems [2], the authors introduce *implicit timestamps*, in which the system adds a special field to each incoming tuple, and *explicit timestamps*, in which a data attribute is designated as the timestamp. Implicit timestamps are used when the exact moment in time associated with a tuple is not important. Explicit timestamps are used when each tuple corresponds to a real-world event at a particular time that is of importance to the meaning of the tuple. Obviously, explicit timestamps are more appropriate to DWH because they provide more precise time information. Explicit timestamps have a draw back that tuples may not arrive in the same order as their timestamps, but this problem can be solved by the above time dimension model. Therefore, in our ZDLWH, we will use the model in [6] for granting the timestamps to data tuples and storing data into DWH in time consistency.

8. Conclusion and future work

In this paper, we specify a framework to implement the ZLDWH by combining the concepts of continuous data integration and analysis rules in ADWH. Furthermore, we take into consideration to a special class of information source namely continuous data streams. Therefore, we suggest novel protocols and techniques for capturing, processing and storing data arrives from continuous data stream sources. The data model that provides time consistency and deals with timely delays in the active DWH environment was used to stored data into the DWH. Our team is considering to realize parts of the concepts presented in this paper as a prototype implementation. Special future efforts will also be taken on the use of gird computing [13] for data streams, further clarification of analysis rules for ZLDWH and its meta data.

Acknowledgement

The authors are grateful for the discussion and suggestion from Dr. Robert Bruckner, IFS, Vienna University of Technology. This work has been partly supported by a Technology Grant South East Asia (No. 322/1/EZA/2002) of the Austrian Council Research and Technology and the ASEA-UNINET.

References

[1] ABITEBOUL, S.; CLUET, S.; MIGNET, L.; AMANN, B.; MILO, T.; EYAL, A., Active Views for Electronic Commerce, in: Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, 138-149.

[2] BABCOCK, B.; BABU, S.; DATAR, M.; MOTWANI, R; WIDOM, J., Models and Issues in Data Stream Systems, in: Proc. of the 2002 ACM Symp. on Principles of Database Systems, pp. 1–16, June 2002.
[3] BAILEY, J.; POULOVASSILIS, A.; WOOD, P., An Event-Condition-Action Language for XML, in: Proc. WWW2002, Hawaii, May 2002, pp 486-495.
[4] BERTINO, E.; GUERRINI, G. ; MERLO, I., Trigger Inheritance and Overriding in an Active Object Database System, in : IEEE Transaction on Knowledge and Data Engineering, 2000. Vol.12 (4), pp. 588-608.
[5] BRUCKNER, R., Zero-Latency Data Warehousing: Toward an Integrated Analysis Environment with Minimized Latency for Data Propagations. Ph.D. Thesis, Vienna University of Technology, Nov. 2002.
[6] BRUCKNER, R.; TJOA, A M., Managing Time Consistency for Active Data Warehouse Environments, in: Y. Kambayashi, W. Winiwarter, and M. Arikawa (Eds.): DaWaK 2001, LNCS 2114, pp. 254–263, 2001. Springer.
[7] CHANDRASEKARAN, S.; FRANKLIN, M. et al., TelegraphCQ: Continuous Dataflow Processing for an Uncertain World, in: First Biennial Conf. on Innovative Data Systems Research (CIDR), Jan. 2003.
[8] CHANDRASEKARAN, S.; FRANKLIN, M., Streaming queries over streaming data, in: Proc. 28th Intl. Conf. on Very Large Data Bases, Aug. 2002.
[9] CHAUDHURI, S.; DAYAL, U. ; GANTI, V., Database Technology for Decision Support System, in: IEEE Computer. December 2001. Vol. 34 (12), pp. 48– 55.
[10] CHAUDHURI, S.; DAYAL, U., An overview of data warehousing and OLAP technology, in: ACM SIGMOD Record 1997; Vol. 26(1) pp. 65– 74.
[11] CHO, E.; PARK, I.; HUYN, S. ; KIM, M., ARML: an Active Rule Markup Language for Sharing Rules among Active. Information Management System, http://www.soi.city.ac.uk/~msch/conf/ruleml/park.pdf
[12] COMPAQ, Compaq Global Services - Zero Latency Enterprise, http://clac.compaq.com/globalservices/zle/
[13] DIFFUSE, Diffuse Final Conference 12th December 2002 http://www.diffuse.org/conference3.html
[14] DOBRA, A.; GAROFALAKIS, M.; GEHRKE, J.; RASTOGI, R., Processing complex aggregate queries over data streams, in: Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, pp. 61–72, 2002.
[15] GUHA, S.; KOUDAS, N., Approximating a data stream for querying and estimation: Algorithms and performance valuation, in: Proc. of the 2002 Intl. Conf. on Data Engineering, pp. 567–576, 2002.
[16] HAHN, B.; BALLINGER, C., Tpump in a Continuous Environment, Assembling the Teradata Active Data Warehouse Series, Active Data Warehouse Center of Expertise, April, 2001.
[17] HAISTEN, M., The Real-Time Data warehousing Defined, white paper, Daman Consulting, 2001.
[18] KIMBALL, R. Real time Partitions, in: Intelligent Enterprise Magazine Vol. 5 (10), June 2002.
[19] LABIO, W J.; YERNENI, R.; GARCIA-MOLINA, H., Shrinking the Warehouse Update Window, in: ACM SIGMOD Record, Vol. 28 (2), pp. 383-394, June 1999.
[20] LERNER, A.; SHASHA, D. The Virtues and Challenges of Ad Hoc + Streams Querying in Finance, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003, pp. 49-56.
[21] MICROSOFT CORP., Microsoft Message Queuing Service, 1997.
[22] MICROSOFT CORP., Message Queuing in Windows XP: New features, white paper, 2001.
[23] MOTWANI, R. ; WIDOM, J. et al., Query processing, approximation, and resource management in a data stream management system, in: Proc. First Biennial Conf. on InnovativeData Systems Research (CIDR), Jan. 2003.
[24] ORACLE CORP. Oracle 9i Application Developer's Guide Advance Queuing, 2002.
[25] PATON, W.; DIAZ, O., Active Database Systems. In: ACM Computing Surveys, Vol. 31, (1), March 1999.
[26] PEDERSEN, T.; JENSEN, C., Multidimensional Database Technology. IEEE Computer. December 2001, pp. 40– 47.
[27] TERADATA CORP., Teradata Online Document, 2002.
[28] THALHAMMER,T.; SCHREFL, M., Realizing active data warehouses with off-the-shelf database technology, in: Softw. Pract. Exper. 2002; Vol. 32, pp. 1193–1222.
[29] THALHAMMER,T.; SCHREFL, M.; MOHANIA, M., Active rules in data warehouses, in: Data and Knowledge Engineering 2001; Vol. 39(3), pp. 241–269.
[30] THE STREAM GROUP, Stanford University, STREAM: The Stanford Stream Data Manager, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003, pp. 19-26.
[31] TUCKER, P.; MAIER, D.; SHEARD, T., Applying Punctuation Schemes to Queries Over Continuous Data Streams, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003, pp. 33-40.
[32] ZDONIK, S.; STONEBRAKER, M. et al., The Aurora and Medusa Projects, in: Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 2003, pp. 3-10.