An Event-Based Near Real-Time Data Integration Architecture

M. Asif Naeem, Gillian Dobbie, Gerald Weber
Department of Computer Science, The University of Auckland,
Private Bag 92019, 38 Princes Street, Auckland, New Zealand
mnae006@ec.auckland.ac.nz, {gill, gerald}@cs.auckland.ac.nz

Abstract

Extract-Transform-Load (ETL) tools feed data from operational databases into data warehouses. Traditionally, these ETL tools use batch processing and operate offline at regular time intervals, for example on a nightly or weekly basis. Naturally, users prefer to have up-to-date data to make their decisions, therefore there is a demand for real-time ETL tools. In this paper we investigate an event-based near real-time ETL layer for transferring and transforming data from the operational database to the data warehouse. One of our main concerns in this paper is master data management in the ETL layer. We present the architecture of a novel, general purpose, event-driven, and near real-time ETL layer that uses a Database Queue (DBQ), works on a push technology principle and directly supports content enrichment. We also observe that the system architecture is consistent with the information architecture of a classical Online Transaction Processing (OLTP) application, allowing us to distinguish between different kinds of data to increase the clarity of the design.

Keywords: event-based architecture, content enrichment, master data, extract-transform-load, enterprise service bus.

1. Introduction

The term data warehousing was defined by Bill Inmon in 1990 [1]: "A warehouse is a subject oriented, integrated, time variant and non volatile collection of data in support of management's decision making process". The basic purpose of data warehousing is to aggregate and analyze data in order to provide reliable information for making useful business related decisions. The traditional data warehouses are static data repositories implemented using batch driven Extract-Transform-Load (ETL) tools. These ETL tools work according to the pull technology principle as shown in Figure 1. The data loading from operational systems to data warehouses is usually performed on a nightly basis or even in some cases on a weekly basis; therefore typical data warehouses normally do not have the most current data [12][17]. Furthermore the operational systems have to go offline during the data extraction process, generating delays that are unacceptable in businesses that require instantaneous access to up-to-date information.

Businesses need to be prepared for growth and volatility. Access to accurate data at the right time to the right place in the right format has become more significant to achieve business success. Business organizations have a keen interest about sales trends based on latest information. The collection of such up-to-date information is a bottleneck in the current ETL system.

To make data integration near to real-time, different approaches [2][3][4][5][6] have been introduced in research that primarily support data translation during the transformation process but do not directly support data enrichment. In this paper our main target is to discuss support for content enrichment in loosely coupled and event-based approaches [14][15]. We use the term "near real-time" in order to indicate, that apart from event processing time the data is available as soon as possible. We prefer the prefix "near", since the service is not real-time in the sense of real-time systems. An event-based approach achieves near real-time data integration but also has to deal with many problems that are encountered with offline ETL tools.

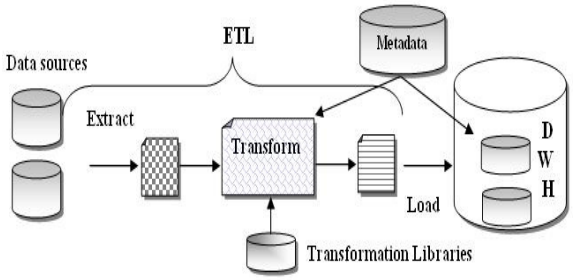

Figure 1: Traditional data warehouse architecture

The rest of the paper is structured as follows. Section 2 highlights the problems in proprietary approaches. Section 3 presents the proposed architecture with its important elements. Section 4 describes the information architecture and its relation to the system architecture. The related work is

explained in Section 5 and finally Section 6 concludes this paper.

2. Problems considered

In the field of near real-time data integration the following problems still need to be resolved and present an important area for further research.

1. Repeated extraction and transformation of master data for each data loading window is an overhead.
2. No plug-and-play support.

In this paper we discuss these problems and propose a new architecture for near real-time data integration directly supporting content enrichment and loose coupling. **Content enrichment** is a special form of data translation in which some additional information is injected into the current message [11]. Consider an example of an inventory system with two data sources, Shipping and Stock. The Shipping data source publishes messages with attributes *order_no, product_id, quantity* and *shipped_date* while the Stock data source publishes attributes *product_id, product_name, sale_price,* and *supplier_id* as shown in Figure 2. In the transform step, a natural join is necessary to create the fact table entries in the data warehouse.

Figure 2-a shows data integration using a proprietary ETL architecture. Assuming that the product table is small compared to the order data in every loading window, it is convenient to extract the product table afresh in each data loading window even if no change to the product table has occurred. Based on this, the join can be performed.

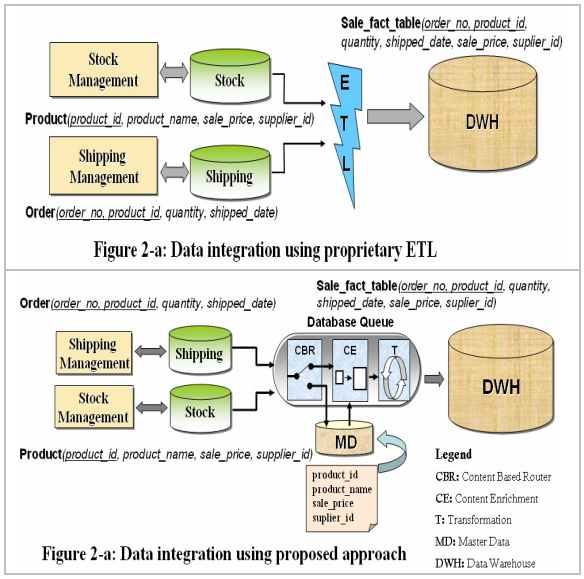

Figure 2: Data integration example using traditional and proposed approach

We will discuss in Section 4 that we can divide the data into two kinds, master data which does not change often and transactional data which is updated regularly. For example, we would consider the product table as master data and the order table as transactional data. For content enrichment all master data referenced by the transaction data in the current loading window has to be available. The extraction and transformation of master data for each data loading window might be acceptable in batch processing ETL tools. In near real-time data integration, where data is extracted and transferred at the transactional level, it is not feasible to extract the whole master data for each transaction. Therefore we have to use more sophisticated master-data management for the ETL process. The lack of plug-and-play support leads to a lack of flexibility in replacing components in the architecture and businesses want to avoid vendor lock-in. A standard event-based solution for data integration is shown in Figure 2-b which is explained later in Section 3.1.

3. Proposed system architecture

To overcome the problems, described in Section 2, a new architecture is proposed using event-based components as shown in Figure 3.

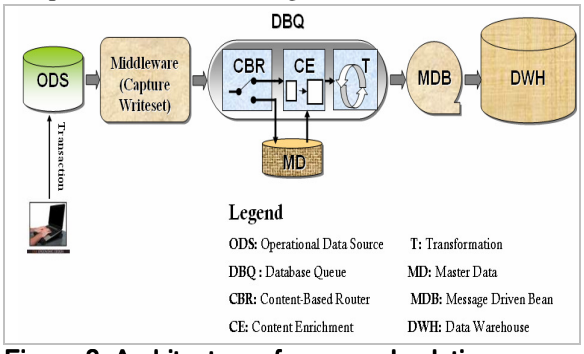

Figure 3: Architecture of proposed solution

Besides the master data repository and the content enrichment component the following three components of the architecture are important: middleware, database queue, and message driven bean.

The **middleware** is a software layer installed over each database to capture the changes in the form of a writeset. The basic purpose of using middleware in our proposed system is to capture the writeset at the middleware level without operational systems going offline or changing the internal structure of the database. The writeset is then propagated through the database queue as shown in figure 3.

The **database queue** is a message oriented queue that stores the messages in database tables. The advantages of using a database queue include:

1. Message will be processed once it is placed in the queue.
2. Multiple message driven beans can work on same queue.
3. All features of databases are available to use.
4. No need to write additional code for create(), enqueue(), dequeue(), poll() and destroy().

In our proposed architecture the database queue takes the role of typical enterprise service bus and performs the given operations.

The **content-based router** analyses the message on the basis of its contents and transmits it on an appropriate channel. The routing criterion is based upon the existence of fields and specific field values [11]. In our proposed architecture the original data sources forward master data and transaction data. Our approach is to distinguish the master data and store it in a separate repository. The content-based router is used to identify the master and the transactional data on the basis of table and field names stored in incoming messages. The master data is directed to the master data repository while the transaction data is propagated to the enrichment process.

A **message driven bean** continuously examines the database queue. As soon as the updates are transformed into the required format it extracts these updates and loads them into the data warehouse.

3.1. Content enrichment in the proposed architecture

Content enrichment, as described in Section 2, is a process of adding more information to the current message. We pass only the transaction data in the message and append the required attributes through the enrichment process before delivering the message at its destination.

To explain the role of content enrichment in our proposed system let us reconsider the example, described in Section 2. Figure 2-b explains data integration using the proposed architecture along with the support of content enrichment. According to Figure 2-b, both applications propagate their updates directly to the database queue.

In Figure 2-b the attributes *product_id*, *product_name*, *sale_price* and *supplier_id* belong to master data and are stored in the master data repository by generating a view of the original attributes. Now the attributes from the master data repository can be appended to the relevant message through the enrich process using *index nested loop join*. Content enrichment, if stated as a relational query for batch processing, involves joins of transaction data with the master data. Our system architecture executes this as on-the-fly *index nested loop join*, where the master data tables serve as inner loop tables. The tuples of the outer relation are presented by messages from the operational database. Then an index-lookup in the master data tables is performed using the *foreign keys* in the transaction data table.

The main advantage of enrichment in our proposed solution is that there is no need to replicate the master data for each transaction. The master data will be extracted from the data source and transferred only if it has changed.

In the data transformation process the data is prepared into required format similar to current approaches.

4. Information architecture

We distinguish the following kinds of data that together constitute an essential part of enterprise data.

Transaction data models events in the central operational business processes in the enterprise. A typical example of transaction data are shopping carts in the retail business. Transaction data will be consolidated in fact tables in the data warehouse. In the simplest case, every unit of transaction data will result in a row of the fact table. Transaction data refers to longer-lived master data.

Master data (dimension data) is referenced by transaction data. The fact tables in a data warehouse contain transaction data that refers to many different dimensions of master data such as customer, product, store etc. Master data is typically much smaller and slowly changing data, used as a reference for transaction data. Typically, master data is changed by idempotent overwrites (write-only transactions [16]).

Other data, such as metadata is not considered further and we focus on transaction and master data. The different kinds of data give structure to the overall data schema and constitute an information architecture.

4.1. Information architecture and system architecture

The distinction between the different kinds of data is originally motivated solely by the meaning of the data. From that alone we would not necessarily assume that these kinds of data are treated in a markedly different way. However, in our system architecture that is chiefly derived from distribution requirements we make the observation that the different kinds of data are naturally treated and routed differently, and this then contributes to the quality of the architecture.

Since the amount of master data is considerably smaller than the amount of transaction data, the master

data can be easily cached. In our real-time ETL architecture, master data is cached in the transport component.

5. Related Work

Most of the research related to data warehousing deals with managing proprietary warehouses [7][8][9] [10]. A major part of the research has focused on the front-end related tools and very limited literature is available related to backend tools.

ETL techniques are mostly working in offline fashion using manual applications [17]. At the same time offline techniques have disadvantages in terms of latency and reliability.

Enterprise Application Integration (EAI) [13][15] also seems to integrate business applications near to real-time basis. Two architectures, hub-spoke, and bus, have been proposed to improve traditional ETL tools. In the case of hub-spoke architecture scalability is affected as the number of messages increase and also a single point failure is introduced. In a bus architecture the message transformation engine is distributed among the applications, which improves the scalability but makes it more complex to manage.

Another effort is done to enhance ETL near to real-time using a queue network [5]. There is an Active Data Staging Area (ADSA) between the source system and data warehouse. ETL functions are performed with the help of queues in this area. Issues with this approach are, for instance, choosing the right topology and the right selection of communication methods.

6. Conclusion and future directions

In the traditional ETL approach the most current information is not available in the data warehouse. By treating all updates to data as events we minimize the latency of the data loading window and achieve a natural system architecture for near real-time ETL. The distinction between transaction data and master data is natural due to the different semantics of both kinds of data. By storing the master data in a repository, we need to extract it only when it has changed. We have shown how even complicated transform operations such as content enrichment can be performed with this architecture. As a proof of concept we have completed the writeset extraction using a trigger-based approach and also tested our database queue in a distributed database environment. In the future, we have a plan to propagate the writset into the database queue and implement the enrichment and transformation processes. Finally the performance evaluation of the proposed architecture is also a part of our future plan.

7. References

[1] Inmon, W.H., *Building the Data Warehouse.* John Wiley, 1992.

[2] Robert M. Bruckner, Beate List, and Josef Schiefer, *Striving towards Near Real-Time Data Integration for Data Warehouses*, Springer-Verlag Berlin Heidelberg, 2002, pp. 317-326.

[3] Tho, M. Nguyen, Tjoa, A. Min, *Zero-Latency Data Warehousing for heterogeneous data sources and continuous data streams*, The fifth International Conference of Information and Web-based Application and Services, 2003, pp. 55-64.

[4] Francisco Araque, *Real-time Data warehousing with temporal requirements*, June, 2003

[5] Alexandros Karakasidis, Panos Vassiliadis, Evaggelia Pitoura, *ETL Queues for Active Data Warehousing,* Baltimore, MD, USA, June2005.

[6] Neoklis Polyzotis, Spiros Skiadopoulos, Panos Vassiliadis, *Supporting Streaming updates in an Active Data Warehousing*, ICDE 2007, IEEE 23rd International Conference on Data Engineering, April 15-20, pp. 476-485.

[7] Galhardas, H., Florescu, D., Shasha, D., and Simon, E.. *Ajax: An Extensible Data Cleaning Tool*. In Proc. ACM SIGMOD, Dallas, Texas, May 2000, pp. 590.

[8] Wilburt Labio, Jun Yang, Yingwei Cui, Hector Garcia-Molina, Jennifer Widom: *Performance Issues in Incremental Warehouse Maintenance.* In Proc. VLDB, Cairo, Egypt, September 2000, 461-472.

[9] Wilburt Labio, Janet L. Wiener, Hector Garcia-Molina, Vlad Gorelik. *Efficient Resumption of Interrupted Warehouse Loads.* In Proc. of ACM SIGMOD, Dallas, Texas, USA, May 2000, 46-57.

[10] Vijayshankar Raman, Joseph M. Hellerstein: Potter's Wheel. *An Interactive Data Cleaning System.* In Proc. VLDB, Rome, Italy, September 2001, 381-390.

[11] Gregor Hohpe,Bobby Woolf, Kyle Brown, *Enterprise Integration Patterns,* Addison-Wesley,2003.

[12] Surajit Chaudhuri, Surajit Chaudhuri*, An Overview of Data Warehousing and OLAP Technology*, In ACM SIGMOD, 1997.

[13] J.E. Armendáriz, H. Decker, F.D.Muñoz-Escoí, L. Irún-Briz, and R. de Juan-Marín, *A middleware architecture for supporting adaptable replication of enterprise application data.* In: TEAA. Volume 3888 of LNCS., Springer (2006) 29.43.

[14] Zheng Yuan Luo, *Investigation of Real-time ETL layer,* Master Thesis, 2007, The University of Auckland, New Zealand.

[15] Ian Gorton, Anna Liu, *Architectures and Technologies for Enterprise Application Integration,* Proceedings of the 26th International Conference on Software Engineering ,ICSE'04.

[16] Agrawal, D. and Krishnaswamy, V. 1991. *Using multiversion data for non-interfering execution of write-only transactions.* SIGMOD Rec. 20, 2 (Apr. 1991), 98-107.

[17] Dave Chappell, *Enterprise Service Bus*. O'Reilly, USA, 2004.