Veri İşleme için Gerçek Zamanlı Veri Ambarı Yaklaşımı

A Real Time Data Warehouse Approach for Data Processing

Murat Obalı, Bünyamin Dursun, Zeki Erdem
Tübitak Bilgem
Information Technologies Institute
Kocaeli, Türkiye
{murat.obali,bunyamin.dursun,zeki.erdem}@tubitak.gov.tr

Abdül Kadir Görür
Department of Computer Engineering
Çankaya University
Ankara, Türkiye
agorur@cankaya.edu.tr

***Özetçe*— Farklı veri kaynaklarından elde edilen verilerin yönetimi, analizi ve izlenmesi çok fazla işlemeye ihtiyaç duyar. Ayrıca, bu verilerin füzyonunu sağlanmak ve etkili bir şekilde incelenmek zorundasınız ve sonuçları gösterebilmelisiniz. Veri ambarı farklı kaynaklardan gelen verileri konsolide eder. Bir gerçek zamanlı veri ambarı da aynı amaçla kullanılmaktadır ancak, ek olarak, veri gerçek zamanlı veri ambarına zamanında akar. Bu nedenle, gerçek zamanlı veri ambarları, sinyal işleme, veri analizi gibi bir çok farklı alanda kullanılabilir. Bu çalışmada, web servisleri kullanarak bir gerçek zamanlı veri ambarı önerilmektedir ve bir prototip geliştirilmiştir.**

Anahtar Kelimeler — veri konsolidasyonu; veri ambarı; gerçek zamanlı; gerçek zamanlı bölüm; sorgu yeniden yazma.

***Abstract*— Data that is acquired from different data sources needs too much processing for managing, analyzing and monitoring. Also, you have to fuse and exploit efficiently these data and display the results. Data warehouse consolidates data coming from different data sources. A real time data warehouse is used same purpose as data warehouse, in addition to these, data streams into real time data warehouse on time. Therefore, real time data warehouses can be used in many different areas, such as signal processing, data analysis. In this study, we proposed and implement a real time data warehouse using web services as a prototype.**

Keywords — data consolidation; data warehouse; real time; real time partition; query rewrite.

I. INTRODUCTION

Managing, analyzing and monitoring of raw data acquired from several kind of sensors or extracted from different data sources requires complex processing. Also, you have to fuse and exploit efficiently these data and display the results [*1*].

A real time data warehouse eliminates the data availability gap and enables organizations to concentrate on processing their valuable data. Furthermore, continuous data processing without delay opens up significant new opportunities [*2*].

The traditional data warehouses are implemented using batch driven approach and mainly according to the pull technology principle. The data loading from source systems to data warehouses is generally 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 [*3*]. Furthermore, the operational systems may have to be go offline during the data extraction process. It is an unacceptable situation which generates delays in businesses especially that require instantaneous access to up-to-date information [*4*].

The approach that we proposed allows users to access data acquired from different data sources in nearly real-time, to record data for future use, to analyze complex structure and relationships, and to process data. Thus the prediction and analysis efficiency can be improved with the use of old and new real-time data. The consolidation of data will correct inefficiencies, decrease costs and improve productivity. It can also be used as a basis for decision-making.

The term data warehouse (DW) is commonly used in industry and it denotes a kind of heterogeneous information system. We have to disclose firstly that a data warehouse is an environment, not a product. The need for building a data warehouse is that corporate data is often scattered in different databases and possibly in different formats. In order to view a complete picture of information, it is necessary to access these heterogeneous databases. Therefore, we have to obtain data and pieces of partial information from each, and then put them together to produce an overall picture. Attempting this process without a data warehouse is a cumbersome task, inefficient, ineffective and error-prone. Moreover, this task will usually need huge efforts of system analysts. All these difficulties discourage the effective use of complex, but valuable corporate data [*5*].

The definition of data warehouse has evolved since its origins in the early 1980s. Dyche [*6*] defines the data warehouse as a separate platform – a computer different from other computers in your IT environment. Bill Inmon [*7*] defines the term DW as: "*A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management's decisions*".

978-1-4673-5563-6/13/$31.00 ©2013 IEEE

II. Related Work

A major part of the research related to data warehousing deals with managing proprietary warehouses [*8*] [*9*] [*10*] [*11*]. Most of the research has focused on the front-end related tools and available literature related to backend tools is very limited [*12*].

Real Time Data Warehouse architecture consists of a lot of views, approaches. Before building RTDW, requirements must be clearly analyzed from a real time needs perspective. Some of these requirements are presented in [*13*].

ETL techniques are generally working in offline fashion [*3*], but offline techniques have several disadvantages in terms of latency and reliability [*12*].

A workable solution to put online is shortening the data warehouse loading cycles. This approach is referred to as near real time data warehousing or micro-batch ETL [*14*].

Another approach is Enterprise Application Integration (EAI) [*14*] that integrates business applications near to real-time basis. Hub-spoke and bus architectures have been proposed to improve traditional ETL tools.

III. Real Time Data Warehouse

In this study, we developed web services based real time data warehouse architecture, as shown in Fig.1. Components of this architecture as follows:

- Web Service Client
- Web Service Provider
- Metadata
- ETL
- Real Time Partition
- Data Warehouse
- Real Time Data Integration

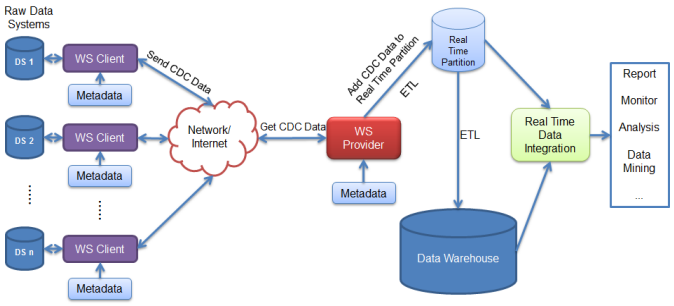

Fig. 1. Web Services Based Real Time Data Warehouse Architecture

Web Service (WS) Client is used for getting data changes (known as Change Data Capture - CDC) from raw data systems and send data to the Web Service Provider by calling related web service.

Data that is required to data warehouse is collected from the operational systems and other external sources. Data capturing techniques in use mainly include; source data extraction, log capture, triggered capture, application-assisted capture, timestamp-based capture, and file comparison capture.

Web Service Provider is basically a web service which gets data sent by RTDW Web Service Client and adds to Real Time Partition. This component gets Data Transfer Object which is sent by RTDW Web Service Client, and decomposed into two parts: data and metadata. RTDW Web Service uses metadata to generate SQL via SQL-Generator for inserting data to RTDW log tables then executes this generated SQL on RTDW database and inserts data.

A. Real Time Partition

Instant data changes (mostly daily) are put into this component first, then later merged into data warehouse. In this architecture, we used a novel structure that is different from traditional real time partition implementations. In this structure we used three steps:

Step 1 - Put CDC data into related warehouse log table: In the warehouse part of the architecture we create log tables for each OLTP table from which data is captured. RTDW Web Service puts all the data sent by client into this log tables.

Step 2 - Clean CDC log data on demand: This step firstly determines the latest state of the data in the log table and we use only latest data at this time for aggregations. We call this clean delta. For example, a data is first INSERTED and DELETED within the day, and at this moment, the latest status of data is DELETED and this step eliminates this data. After that, this step uses clean delta records to clean and format CDC data. Clean delta log type conversions are shown in table 1.

Table 1: Clean Delta Log Type Conversions

First Record Log Type	Last Record Log Type	Result
INSERT	DELETE	ELEMINATE
INSERT	UPDATE	INSERT LAST RECORD
UPDATE	INSERT	UPDATE LAST RECORD
UPDATE	DELETE	DELETE LAST RECORD
DELETE	INSERT	UPDATE LAST RECORD
DELETE	UPDATE	UPDATE LAST RECORD

Step 3 - Aggregate clean CDC data on demand: In this architecture, also, the real time aggregations are calculated on demand. If there is no request for real time data, the aggregation is not calculated throughout the day. When data is requested from real time partition, only the related aggregations are calculated on demand. We call this On Demand Aggregation.

An example implementation of Real Time Partition on Oracle can be seen on Source Code 1.

```
1. CREATE OR REPLACE VIEW v_sales_fact_rtp
2. AS
3.     SELECT od.product_id,
4.            o.customer_id,
5.           TO_NUMBER (TO_CHAR (o.order_date, 'YYYYMMDD')) AS time_id,
6.            SUM (od.quantity) AS sales_qty,
7.            SUM (od.quantity * od.unit_price) AS sales_amount,
8.            SUM (od.quantity * od.unit_price * od.discount / 100)
9.               AS discount_amount
10.       FROM (SELECT *
11.               FROM (SELECT t.*,
12.                            FIRST_VALUE (log_type)
```

```
                              OVER (PARTITION BY order_id, product_id
                                  ORDER BY log_id ASC) first_log_flag,
                              FIRST_VALUE (log_type)
                              OVER (PARTITION BY order_id, product_id
                                  ORDER BY log_id DESC) last_log_flag,
                              ROW_NUMBER ()
                              OVER (PARTITION BY order_id, product_id
                                   ORDER BY log_id DESC) seq
                         FROM t_order_details_log t)
               WHERE seq = 1 AND last_log_flag <> 'D') od,
            (SELECT *
               FROM (SELECT t.*,
                       FIRST_VALUE (log_type)
                     OVER (PARTITION BY order_id ORDER BY log_id ASC)
                       first_log_flag,
                       FIRST_VALUE (log_type)
                     OVER (PARTITION BY order_id ORDER BY log_id DESC)
                       last_log_flag,
                       ROW_NUMBER ()
                     OVER (PARTITION BY order_id ORDER BY log_id DESC)
                       seq
                      FROM t_orders_log t)
               WHERE seq = 1 AND last_log_flag <> 'D') o
      WHERE o.order_id = od.order_id
   GROUP BY od.product_id, o.customer_id,
            TO_CHAR(o.order_date, 'YYYYMMDD');
```

Source Code 1: An Example of Real Time Partition on Oracle

B. *Real Time Data Integration*

This component is used for integrating the data both in Real Time Partition and Data Warehouse. When a user sends a query to this component; if query only wants historical data, then this component sends the query to Data Warehouse; if query wants both historical and instant data, then this component rewrites the query to get and integrate data.

In our case study, query rewriting is done by using views. First, we get the SQL from user and determine the date predicate. Then if date predicate consists of today then we replace the fact table with our view which merges fact table and real time partition table as shown in Fig. 2.

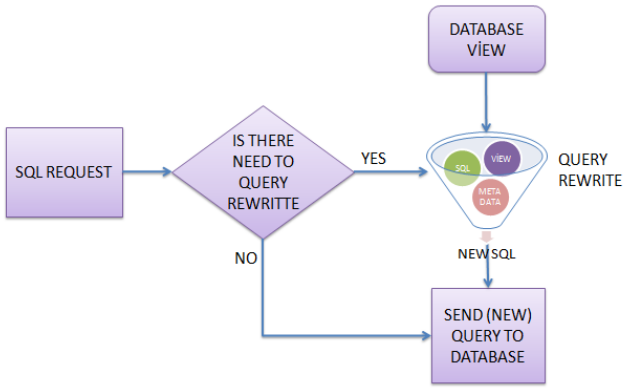

Fig. 2. Query Rewrite Flow

An example implementation of Data Integration Service on Oracle can be seen on Source Code 2.

```
1. CREATE OR REPLACE FUNCTION sales_fact_dis
2.                          (in_date1 IN NUMBER, in_date2 IN NUMBER)
3.   RETURN sales_fact_tab IS
4.   l_sales_fact_tab   sales_fact_tab
5.                        := sales_fact_tab (NULL,
6.                                           NULL,
7.                                           NULL,
8.                                           NULL,
9.                                           NULL,
10.                                          NULL);
11.   TYPE               type_data_cursor   IS REF CURSOR;
12.   c1                 type_data_cursor;
13.   r                  sales_fact%ROWTYPE;
14.   n                  NUMBER := 0;
15.   l_sql              VARCHAR2(32767);
16. BEGIN
17.   IF TO_NUMBER(TO_CHAR(SYSDATE, 'YYYYMMDD'))
18.         BETWEEN in_date1 AND in_date2 THEN
19.      l_sql := 'SELECT *
20.                  FROM (
21.                        SELECT *
22.                          FROM sales_fact
23.                        UNION ALL
24.                        SELECT *
25.                          FROM v_sales_fact_rtp
26.                          )';
27.   ELSE
28.      l_sql := 'SELECT * FROM sales_fact';
29.   END IF;
30.
31.   l_sql := l_sql || ' WHERE time_id BETWEEN '||in_date1||' AND
'||in_date2;
32.
33.   OPEN c1 FOR l_sql;
34.   LOOP
35.      FETCH c1 INTO r;
36.      EXIT WHEN c1%NOTFOUND;
37.
38.      l_sales_fact_tab.EXTEND;
39.      n := n + 1;
40.      l_sales_fact_tab (n) :=
41.         sales_fact_obj (r.product_id,
42.                         r.customer_id,
43.                         r.time_id,
44.                         r.sales_qty,
45.                         r.sales_amount,
46.                         r.discount_amount);
47.
48.   END LOOP;
49.   l_sales_fact_tab.TRIM(6);
50.   RETURN l_sales_fact_tab;
51. END;
```

Source Code 2: An Example of Data Integration Service on Oracle

IV. Conclusion and Future Work

Real time data warehouse is much more than a new feature. Moving to real time delivery of data challenges every aspect of the data warehouse.

Managing, analyzing and monitoring of raw data acquired from several kinds of data sources is a very difficult task and needs too much processing. The proposed approach allows users to access data acquired from different data sources in nearly real-time, to record data for future use, to analyze complex structure and relationships, and to process data.

We designed, developed and built a web services based on real time data warehouse. Web services are important component as they communicate easily to the servers which publish services. Because of the fact that they are easily adaptable and maintain many of the communication problems, we think that web services are preferable choice.

Capturing change data from source systems is also a major problem for data warehouse constructions. Log capture and triggering is the mostly used techniques in this area. We think that log capture may be preferred to triggering which gets come overhead to the source system. However, it's easy to implement and need nothing except source database. Since today, triggers are supported most of the databases, it may be a good choice.

Real time partition is another important issue for building a real time data warehouse. Because of the design purposed of the data warehouses, you could not load instant data to your data warehouse immediately. You need a staging table before aggregating data and load it to the data warehouse, because your data is completed later (for example at the end of the day). Therefore, we use real time partition. In addition to that, if user requests real time data for his analysis, then the data in the real time partition and data warehouse have to be merged. For this

purpose, we used query rewriting which decides to rewrite and replaces the fact table with a view joined by real time partition.

REFERENCES

[1] P.F. Biagi, C. Guaragnella, A. Guerriero, C.C. Pasquale, and F. Ragni, "A Data Warehouse for earthquakes signal precursors analysis," , 2009.

[2] Michael Haisten, "Real-Time Data Warehouse: What is Real Time about Real-Time Data?," *DM Review*, August 2000.

[3] David A. Chappell, *Enterprise Service Bus*. USA: O'Reilly, 2004.

[4] M. Asif Naeem, Gillian Dobbie, and Gerald Weber, "2008 12th Enterprise Distributed Object Computing Conference Workshops," in *An Event-Based Near Real-Time Data Integration Architecture*, Munich, Germany, 2008, pp. 401-404.

[5] Patricia Ward and George Dafoulas, *Database Management Systems*.: Thomson Learning, 2006.

[6] Jill Dyche, *e-Data: Turning Data into Information with Data Warehousing*.: Addison-Wesley, 2000.

[7] William H. Inmon, *Building the Data Warehouse*.: Wiley, 2005.

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

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

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

[11] Praveen Sharma, "Advanced Applications of Data Warehousing Using 3-tier Architecture," *DESIDOC Journal of Library & Information Technology*, vol. 29, pp. 61-66, March 2009.

[12] M. Asif Naeem, Gillian Dobbie, and G. Webber, "An Event-Based Near Real-Time Data Integration Architecture," in *Enterprise Distributed Object Computing Conference Workshops*, 2008, pp. 401-404.

[13] R., Cacerta, J. Kimball, *he Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*.: Wiley, 2004.

[14] Ralph Kimball and Joe Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*.: John Wiley & Sons, Inc., 2004.

[15] J.E. Armendáriz, H. Decker, F.D. Muñoz-Escoí, L. Irún-Briz, and R. Juan-Marín, "A middleware architecture for supporting adaptable replication of enterprise application data," vol. 3888 of LNCS, p. 29.43., Springer (2006).