Chapter 7

Big Data Management in the Context of Real-Time Data Warehousing

M. Asif Naeem
Auckland University of Technology, New Zealand

Gillian Dobbie
University of Auckland, New Zealand

Gerald Weber
University of Auckland, New Zealand

ABSTRACT

In order to make timely and effective decisions, businesses need the latest information from big data warehouse repositories. To keep these repositories up to date, real-time data integration is required. An important phase in real-time data integration is data transformation where a stream of updates, which is huge in volume and infinite, is joined with large disk-based master data. Stream processing is an important concept in Big Data, since large volumes of data are often best processed immediately. A well-known algorithm called Mesh Join (MESHJOIN) was proposed to process stream data with disk-based master data, which uses limited memory. MESHJOIN is a candidate for a resource-aware system setup. The problem that the authors consider in this chapter is that MESHJOIN is not very selective. In particular, the performance of the algorithm is always inversely proportional to the size of the master data table. As a consequence, the resource consumption is in some scenarios suboptimal. They present an algorithm called Cache Join (CACHEJOIN), which performs asymptotically at least as well as MESHJOIN but performs better in realistic scenarios, particularly if parts of the master data are used with different frequencies. In order to quantify the performance differences, the authors compare both algorithms with a synthetic dataset of a known skewed distribution as well as TPC-H and real-life datasets.

DOI: 10.4018/978-1-4666-4699-5.ch007

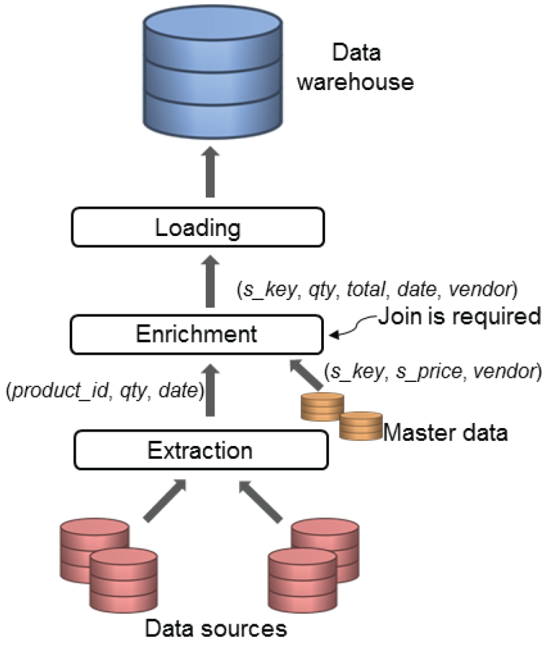
INTRODUCTION

Real-time data warehouse deployments are driving an evolution to more aggressive data freshness service levels. The tools and techniques for delivering against these new service levels are rapidly evolving (Karakasidis, Vassiliadis, & Pitoura, 2005)(Naeem, Dobbie, & Weber, 2008). In the beginning, most data warehouses fully refreshed all content during each load cycle. However, due to rapid growth in the size of warehouses and the increasing demand of information freshness, it became infeasible to meet the business needs. Thus the data acquisition mechanism in warehouses changed from full refresh to an incremental refresh strategy, in which new data is added to the warehouse without requiring a complete reload (Labio & Garcia-Molina, 1996)(Labio, Yang, Cui, Garcia-Molina, & Widom, 1999). Although this strategy is more efficient than the traditional one, it is still batch-oriented; a fraction of the data is propagated to the warehouse after a particular timestamp.

In order to overcome update delays, these batch-oriented and incremental refresh strategies are being replaced with a continuous refresh strategy (Golab, Johnson, Seidel, & Shkapenyuk, 2009)(Zhang & Rundensteiner, 2002) (Zhuge, García-Molina, Hammer, & Widom, 1995). In such strategies, end user data from data sources is being captured and propagated to the data warehouse in real-time in order to support high levels of data freshness. This leads to a stream processing approach. Stream processing is natural for todays Big Data, since no intermediate storage has to be considered. This can lead to an architecture that is even simpler than batch processing and at the same time delivers a distinctly new quality of service.

One important research area in the field of data warehousing is data transformation, since the updates coming from the data sources are often not in the format required for the data warehouse. In the field of real-time data warehousing where data arrives in the form of an infinite stream and a continuous transformation from a source to a target format is required, such tasks become more challenging. In the ETL (Extract-Transform-Load) layer, a number of transformations are performed such as the detection of duplicate tuples, identification of newly inserted tuples, and the enriching of some new attribute values from master data. One common transformation is the key transformation. The key used in the data source may be different from that in the data warehouse and therefore needs to be transformed into the required value for the warehouse key. To explain the transformation phase further we consider an example shown in Figure 1 that implements one of the above features called enrichment. In the example we consider the source updates with attributes *product_id*, *qty*, and *date* that are extracted from data sources. At the transformation layer in addition to key replacement (from source key *product_id* to warehouse key *s_key*) some information such as vendor information and sales price are needed to calculate the total amount. In the figure this information with attribute names *s_key*, *s_price*, and *vendor*

Figure 1. An example of content enrichment


Data warehouse
Loading
(s_key, qty, total, date, vendor)
Join is required
Enrichment
(s_key, s_price, vendor)
(product_id, qty, date)
Master data
Extraction
Data sources