Chapter 2

Online Processing of End-User Data in Real-Time Data Warehousing

M. Asif Naeem
Auckland University of Technology, New Zealand.

Noreen Jamil
University of Auckland, New Zealand.

ABSTRACT

Stream-based join algorithms are a promising technology for modern real-time data warehouses. A particular category of stream-based joins is a semi-stream join where a single stream is joined with a disk based master data. The join operator typically works under limited main memory and this memory is generally not large enough to hold the whole disk-based master data. Recently, a seminal join algorithm called MESHJOIN (Mesh Join) has been proposed in the literature to process semi-stream data. MESHJOIN is a candidate for a resource-aware system setup. However, MESHJOIN is not very selective. In particular, MESHJOIN does not consider the characteristics of stream data and its performance is suboptimal for skewed stream data. This chapter presents a novel Cached-based Semi-Stream Join (CSSJ) using a cache module. The algorithm is more appropriate for skewed distributions, and we present results for Zipfian distributions of the type that appear in many applications. We conduct a rigorous experimental study to test our algorithm. Our experiments show that CSSJ outperforms MESHJOIN significantly. We also present the cost model for our CSSJ and validate it with experiments.

INTRODUCTION

Real-time data warehousing plays a prominent role in supporting overall business strategy. By extending data warehouses from static data repositories to active data repositories enables business organizations to better inform their users and to take effective timely decisions. In real-time data warehousing the changes occurring at source level are reflected in data warehouses without any delay. Extraction, Transformation, and Loading (ETL) tools are used to access and manipulate transactional data and then load them into
the data warehouse. An important phase in the ETL process is a transformation where the source level changes are mapped into the data warehouse format. Common examples of transformations are units conversion, removal of duplicate tuples, information enrichment, filtering of unnecessary data, sorting of tuples, and translation of source data key.

To explain the transformation phase further we consider an example shown in Figure 1 that implements one of 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`) there is some information added, sales price to calculate the total amount, and the vendor information. In the figure these information with attributes name `s_key`, `s_price`, and `vendor` are extracted at run time from master data and are used to enrich the source updates using a join operator.

In traditional data warehousing the source updates are buffered and join is performed off-line. On the other hand, in real-time data warehousing this operation needs to be performed as the updates are received from the data sources. In implementing the online execution of join, it is observed that due to different arrival rate of both inputs, the transactional or stream input is fast and huge in volume while the master or disk input is slow; the algorithm faces some performance issues due to a bottleneck in the stream of updates.

With the availability of large main memory and powerful cloud computing platforms, considerable computing resources can be utilized when executing stream-based joins. However, there are several

Figure 1. An example of content enrichment
Related Content

Discovery of Latent Patterns with Hierarchical Bayesian Mixed-Membership Models and the Issue of Model Choice
www.igi-global.com/chapter/discovery-latent-patterns-hierarchical-bayesian/7568?camid=4v1a

Current Issues and Future Analysis in Text Mining for Information Security Applications
www.igi-global.com/chapter/current-issues-future-analysis-text/21752?camid=4v1a

A Query Beehive Algorithm for Data Warehouse Buffer Management and Query Scheduling
www.igi-global.com/article/a-query-beehive-algorithm-for-data-warehouse-buffer-management-and-query-scheduling/116892?camid=4v1a

Efficient and Effective Aggregate Keyword Search on Relational Databases
Luping Li, Stephen Petschulat, Guanting Tang, Jian Pei and Wo-Shun Luk (2012). *International Journal of Data Warehousing and Mining* (pp. 41-81).
www.igi-global.com/article/efficient-effective-aggregate-keyword-search/74755?camid=4v1a