Chapter 13
HYBRIDJOIN for Near–Real–Time Data Warehousing

M. Asif Naeem
The University of Auckland, New Zealand

Gillian Dobbie
The University of Auckland, New Zealand

Gerald Weber
The University of Auckland, New Zealand

ABSTRACT
An important component of near-real-time data warehouses is the near-real-time integration layer. One important element in near-real-time data integration is the join of a continuous input data stream with a disk-based relation. For high-throughput streams, stream-based algorithms, such as Mesh Join (MESHJOIN), can be used. However, in MESHJOIN the performance of the algorithm is inversely proportional to the size of disk-based relation. The Index Nested Loop Join (INLJ) can be set up so that it processes stream input, and can deal with intermittences in the update stream but it has low throughput.

This paper introduces a robust stream-based join algorithm called Hybrid Join (HYBRIDJOIN), which combines the two approaches. A theoretical result shows that HYBRIDJOIN is asymptotically as fast as the fastest of both algorithms. The authors present performance measurements of the implementation. In experiments using synthetic data based on a Zipfian distribution, HYBRIDJOIN performs significantly better for typical parameters of the Zipfian distribution, and in general performs in accordance with the theoretical model while the other two algorithms are unacceptably slow under different settings.

INTRODUCTION
Near-real-time data warehousing exploits the concepts of data freshness in traditional static data repositories in order to meet the required decision support capabilities. The tools and techniques for promoting these concepts are rapidly evolving (Thomsen & Pedersen, 2009; Golfarelli & Rizzi, 2009a, 2009b; Vassiliadis, 2009). Most data warehouses have already switched from a full refresh (Gupta & Mumick, 1999; Zhang & Rundensteiner, 2002; Zhuge, Garcia-Molina,
HYBRIDJOIN for Near-Real-Time Data Warehousing

Hammer, & Widom, 1995) to an incremental refresh policy (Labio & Garcia-Molina, 1996; Labio, Wiener, Garcia-Molina, & Gorelik, 2000; Labio, Yang, Cui, Garcia-Molina, & Widom, 2000). Furthermore, the batch-oriented, incremental refresh approach is moving towards a continuous, incremental refresh approach (Thiele, Fischer, & Lehner, 2007; Karakasidis, Vassiliadis, & Pitora, 2005; Nguyen, 2003).

With regards to terminology, data warehousing approaches that follow such a best-effort data freshness approach have various names. Frequently used terms are zero-latency, active, real-time or near-real-time data warehouses. The term near-real-time is the most descriptive in a context where there could be confusion with real-time control systems, but for the sake of brevity, we will mostly use the term real-time in this paper where no such confusion is possible.

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. For real-time data warehousing a continuous transformation from a source to target format is required, so the task becomes 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 updates with values from the master data. Enrichment in particular can often be expressed as a join between the update stream and the master data (Naeem, Dobbie, & Weber, 2008). One important example of enrichment is a key transformation. Normally the key used in the data source is different from that in the data warehouse and therefore needs to be replaced. This transformation can be obtained by implementing a join operation between the update tuples and a lookup table. The lookup table contains the mapping between the source keys and the warehouse keys. Figure 1 shows a graphical interpretation of such a transformation.

The attributes with column name $id$ in both data sources $DS_i$ and $DS_j$ contain the source data keys and the attribute with name warehouse key in the lookup table contains the warehouse key value corresponding to these data source keys. Before loading each transaction into the data warehouse each source key is replaced by the warehouse key with the help of a join operator.

In traditional data warehousing the update tuples are buffered in memory and joined when resources become available (Wilschut & Apers, 1991; Shapiro, 1986). Whereas, in real-time data warehousing these update tuples are joined immediately when they are generated in the data sources. One important factor related to the join is that both inputs of the join come from different sources with different arrival rates. The input from the data sources is in the form of an update stream which is fast, while the access rate of the lookup table is comparatively slow due to disk I/O cost.

A novel stream-based equijoin algorithm, MESHJOIN (Polyzotis, Skiadopoulos, Vassiliadis, Simitsis, & Frantzell, 2007; Neoklis Polyzotis, Skiadopoulos, Vassiliadis, Simitsis, & Frantzell, 2008) is in principle a hash join, where the stream serves as the build input and the disk-based relation serves as the probe input. The main contribution is a staggered execution of the hash table build and an optimization of the disk buffer for the disk-based relation.

The algorithm successfully joins the continuous data stream of updates with the slow access rate disk-based relation. However, we have identified two issues that have to be addressed. Firstly, the throughput of MESHJOIN is inversely proportional to the size of the disk-based master data table. Secondly, the algorithm cannot deal with an intermittent update stream efficiently. An intermittent stream is a stream that is dropping to a rate close to zero tuples per unit of time for periods of time. A detailed explanation of these issues is provided in Section 3.

The Index Nested Loop Join (INLJ) (Ramakrishnan, 1999) is traditionally considered for non-stream data, but it can easily be set up so
Related Content

Data Discovery Approaches for Vague Spatial Data
Frederick E. Petry (2013). Data Mining: Concepts, Methodologies, Tools, and Applications  (pp. 50-65).
www.igi-global.com/chapter/data-discovery-approaches-vague-spatial/73433?camid=4v1a

Classification of Web Pages Using Machine Learning Techniques
www.igi-global.com/chapter/classification-web-pages-using-machine/29148?camid=4v1a

Mining Frequent Generalized Patterns for Web Personalization in the Presence of Taxonomies
www.igi-global.com/article/mining-frequent-generalized-patterns-web/38954?camid=4v1a

Introduction to Ranking Models
www.igi-global.com/chapter/introduction-ranking-models/36632?camid=4v1a