Efficient Lossless Real-Time Stream Processing

Stephen A. Broeker *steve_broeker@yahoo.com*
supervised by Ahmed Amer *aamer@scu.edu* and Weijia Shang *wshang@scu.edu*

February 22, 2011

Abstract

Streaming OLAP is relatively new (traditional OLAP focuses on data warehousing - data at rest). This is the first solution that handles the full data path, not through sampling (a serious compromise necessitated by existing approaches). As well as improving performance for existing applications, this is particularly critical for security applications and forensic processing, which are applications that are now enabled. My experiments show that the effective observed input rate for a one gigabit Ethernet Network Interface Card (NIC) is up to 3 million flows per hour. This input rate is too much for existing databases. The proposed database can process over 113 million network data flows per hour, a near 40 fold improvement in performance.

1 Introduction

OnLine Analytical Processing (OLAP) engines are commonly implemented as multidimensional data cubes - called hypercubes [4, 7, 9, 12, 13] to handle SQL GROUP BY or aggregate queries. The hypercube is constructed so that each cell corresponds to a unique combination of database attribute values.

Given a database with l number of attributes, the number of cells in the corresponding fully populated data cube is $\prod_{1 \leq i \leq l}(a_i + 1)$ where each attribute i has a_i values [10, 11].

The complete creation of the data cube is essentially impossible because the number of hypercube cells is prohibitive. The complete creation of the data cube is unnecessary because some combinations of attributes may not be used by the database.

In a streaming database, data streams come in at a high rate (gigabits per second) and the database is dynamic where data records are constantly added. Examples of such data streams are: network traffic, web sit hits, credit card transactions, road traffic, video, power supplies, phone calls, and financial markets. Applying OLAP to streaming databases is not only a relatively new problem but a much more difficult problem than traditional OLAP (data warehousing).

Current data cubes are commonly supported by star schema databases. A star schema is optimized to minimize string space - all string attributes are stored in separate dimension tables. Each dimension table is sorted to optimize query performance. Dimension table insertion time thus depends on the table size and is $O(\log n)$ where n is the number of records in a table. Star schema insertion time then, is the sum of all dimension table insert times $O(\sum_{1 \leq i \leq l}(\log n_i))$ where l is the number of attributes in the database and n_i is the number of values for attribute i.

A common problem with a star schema is that it cannot keep up with input data stream rates, which results in database filtering, which in turn limits OLAP effectiveness. This paper presents a new type of star schema: the *stream star schema* that is much better at keeping up with data stream rates. This eliminates the need for database filtering and thus empowers streaming OLAP.

The stream star schema is optimized to minimize insertion time. This star schema uses a global string table to store all data strings. Multiple copies of strings are allowed in the string table. The string

table does not need to be sorted - thus strings are appended, resulting in a constant insert time. String attributes are only represented by dimension tables when duplication is significant. This results in a constant insertion time for dimension tables. Thus stream star schema insertion time is constant and does not depend on the size of the database.

2 Related Work

Currently, there are three kinds of OLAP data cube implementation [4, 7, 9, 12, 13].

In Relational OLAP (ROLAP), a relational database is used as the data implementation. This typically is a star schema [10]. The star schema was invented by Ralph Kimball [5, 6]. The star schema does not support OLAP - hence a data cube has to be created.

In Multi-dimensional OLAP (MOLAP) - a multi-dimensional data cube (hypercube) is used as the data implementation. MOLAP does not support On-Line Transactional Processing (OLTP).

In Hybrid OLAP (HOLAP), a combination of ROLAP and MOLAP is used. A relational database is used to store data values and a data cube is used to store data aggregates.

Streaming OLAP exacerbates the data cube creation problem since input rates can be high. For a one gigabit Network Interface Card (NIC), the effective input stream rate can be as high as 3 million flows per hour. Some authors use filtering which is restrictive [1–3, 8]. A better (complete) method is to input the entire data stream into the database. Thus database insertion time has to improve.

3 Solution

In this section, the star schema is described. The challenges presented by streaming data are discussed which motivates the proposed new database: the it stream star schema. The stream star schema is then defined and illustrated by an example. Advantages and disadvantages are discussed.

For this paper, the network data stream is used as a reference example. A network data stream record consists of the 16 tuple: {content, time stamp, destination ip, destination location, destination port, mail bcc, mail cc, mail file name, mail recipient, mail sender, mail subject, protocol, size, source ip, source location, source port}. String attributes are {content, destination location, mail bcc, mail cc, mail file name, mail recipient, mail sender, mail subject, protocol, source location}. Numeric attributes are {time stamp, destination ip, destination port, size, source ip, source port}.

A star schema is usually constructed to minimize string space [5, 6]. A star schema consists of fact tables and dimension tables. Fact table string attributes point to dimension tables to minimize disk space. Numeric attributes are of fixed size and thus are not coalesced into dimension tables. Dimension tables are usually sorted to support OLTP.

Insertion time into a dimension table is $O(\log n)$ where n is the number of rows in the dimension table. Thus star schema insertion time is $O(\sum_{1 \leq i \leq l}(\log n_i))$ where l is the number of dimension tables and n_i is the number of records in dimension table i.

The star schema for the network data stream contains a separate dimension table for each string attribute. This star schema thus contains a single fact table {flow} and 7 dimension tables {content, location, mail file name, mail recipient, mail sender, mail subject, protocol} for the string attributes.

A stream star schema is optimized to minimize insertion time. Thus, this star schema has a single string table which contains all database strings. The string table does not need to be sorted. So string table insertion simply consists of appending a string to the table, resulting in constant insertion time. Duplicate strings are thus allowed in the string table. All string attributes are indices into the global string table. Dimension tables are created for string attributes that have significant duplication. Thus dimension table insertion time is constant.

The stream star schema for the network data stream contains a single fact table {flow} and 2 dimension tables {content, protocol}. There are 59 possible protocol types and 201 possible content types. Thus both of these dimension tables are small.

The proposed stream star schema has two main

differences from a star schema. The first difference is that all strings are stored in a global string table. The second difference is data insertion. In the star schema, strings are inserted into sorted dimension tables. This operation takes $O(\sum_{1 \leq i \leq l}(\log n_i))$ time. In the stream star schema, all strings are appended to a global string table that is unsorted. This reduces the insertion time to a constant. Clearly the disadvantage of this solution is that the global string table may become large.

A star schema focuses on minimizing disk space. Dimension tables are sorted without duplication. The down side is that insertion time for each data record is $O(\sum_{1 \leq i \leq l}(\log n_i))$ and becomes unacceptable when the database size becomes large and the data rate is high. The stream star schema is proposed to keep up with the incoming data stream rate. The focus is thus not on minimizing disk space but rather on minimizing insertion time. There is a tradeoff between disk space and insertion time. In the stream star schema, more dimension tables can be created for string attributes as long as the database can keep up with the input data stream rate. The fastest database is one without any dimension tables. Table 1 summarizes the advantages and disadvantages of this database and the star schema.

4 Experimental Results

This section presents an application of the stream star schema. A generic stream star schema engine was implemented that used an XSD to define the input data stream. This XSD was defined for the network data stream. A star schema engine was implemented for the network data stream using MySQL.

The performance metrics for inserting one million network data streams into the star schema and the stream star schema are presented in Table 2 and Figure 1.

Table 2 and Figure 1 show that the star schema could insert less than 700,000 network data streams per hour. The stream star schema could insert more than 133,000,000 network data streams per hour. The difference is a factor of 177, which is significant.

My experiments show the maximum effective network data stream input rate for a 1 gigabit NIC is 3 million per hour. Clearly, the star schema cannot handle this input rate. My database can easily handle this input rate. So much so, that it could handle a 37 gigabit NIC.

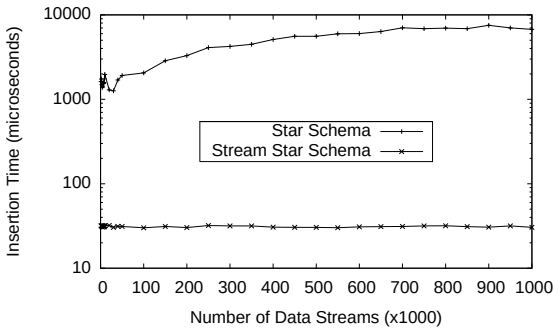

Figure 1: Database insertion time per data stream (in usecs) when inserting 1 million network data streams into the stream star schema and star schema.

The flexibility of the stream star schema allows for some tuning when applied to the network data stream. Since this database can easily handle the input rate, spare cpu cycles could be used to convert other string attributes to dimension tables - thus optimizing disk space without a loss of input.

Table 3 compares the memory usage for the two databases when inserting one million network data streams. This table shows that the stream star schema database was superior to the star schema by a factor of 2. The reason is that star schema dimension tables are sorted - thus requiring index tables.

5 Conclusion

This paper has presented a new approach to processing data streams: the *stream star schema*. This new type of star schema is proposed to accommodate high data stream rates: giga bits per second, by reducing insertion time to a constant. An experimental implementation of both star schema types on the network data stream showed that stream star schema insertion performance is constant and superior to star schema insertion performance by a fac-

Stream Star Schema	Star Schema
Three kinds of tables - *fact, dimension, string.*	Two kinds of tables - *fact, dimension.*
Few dimension tables.	All string dimensions have dimension tables.
Minimize insertion time.	Minimize disk space.
Dimension tables are small.	Dimension tables can be large.
Insert time is constant.	Insert time = $O(\sum_{i=1}^{k} logn_i)$
Allow string duplication.	Do not allow string duplication.

Table 1: Comparison of the stream star schema and the star schema.

Database type.	Stream Star Schema	Star Schema
Time to insert 1 data stream in usecs.	31.77	1,293 to 7,517
Network data stream input rate per second.	31,476	177
Network data stream input rate per minute.	1,888,574	10,619
Network data stream input rate per hour.	113,314,447	637,135
Time to input 1 million network data streams.	31.77 seconds	1 hour, 34 minutes, 10 seconds

Table 2: Performance metrics for inserting 1 million network data streams into the stream star schema and star schema.

Database type.	Stream Star Schema	Star Schema
Input bytes.	182,211,249	182,211,249
Memory bytes.	167,005,635	333,048,780
Raio of memory bytes to input bytes.	0.92	1.83
Memory bytes per stream.	167	133

Table 3: Memory usage when inserting 1 million network data streams into the stream star schema and star schema.

tor of 177. In addition, star schema memory usage was greater than stream star schema memory usage by a factor of 2. My experiments thus show that the stream star schema improves performance for existing OLAP applications. In addition, the stream star schema enables security applications and forensic processing.

References

[1] Y. Chen, G. Dong, J. Han, B. Wah, and J. Wang, "Multi-dimensional regression analysis of time-series data streams," in *28th International Conference on Very Large Data Bases*, August 2002.

[2] C. Giannella, J. Han, J. Pei, X. Yan, and P. Yu, *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*, ch. 3, pp. 191–210. Chapman and Hall, 2003.

[3] J. Han, "Mining unusual patterns by multi-dimensional analysis of data streams," tech. rep., University of Illinois at Urbana-Champaign, 2002.

[4] T. Johnson and D. Sasha, "Some approaches to index design for cube forests," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 1999.

[5] R. Kimball, "www.ralphkimball.com."

[6] R. Kimball and J.Caserta, *The Data Warehouse ETL Toolkit*. Wiley, 2004.

[7] Y. Kudryavcev, "Efficient algorithms for MOLAP data storage and query processing," in *Spring Young Researcher's Colloquium on Database and Information Systems*, June 2006.

[8] X. Li, J. Han, Z. Yin, J. Lee, and Y. Sun, "Sampling cube: A framework for statistical OLAP over sampling data," in *28th ACM SIGMOD/PODS International Conference on Management of Data/Principles of Data Systems*, June 2008.

[9] M. Sabhnani, A. Moore, and A. Dubrawski, "T-cube: A data structure for fast extraction of time series from large datasets," Tech. Rep. CMU-ML-07-114, Carnegie Mellon, April 2007.

[10] A. Shukla, P. M. Deshpande, J. F. Naughton, and K. Ramasamy, "Storage estimation for multidimensional aggregates in the presence of hierarchies," in *22nd International Conference on Very Large Data Bases*, September 1996.

[11] Y. Sismanis, Y. Kotidis, A. Deligiannakis, and N. Roussopoulos, "Hierarchical dwarfs for the rollup cube," in *ACM Sixth International Workshop on Data Warehousing and OLAP*, November 2003.

[12] N. Stefanovic, J. Han, and K. Koperski, "Object-based selective materialization for efficient implementation of spatial data cubes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 6, 2002.

[13] Y. Zhao, P. Deshpande, and J. F. Naughton, "An array-based algorithm for simultaneous multidimensional aggregates," in *ACM SIGMOD International Conference on Management of Data*, May 1997.