Extending OLAP beyond Aggregates and Summaries

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

February 22, 2011

Abstract

Traditional OLAP data cubes only store data aggregates and existing designs are unable to efficiently maintain access to complete data records. In this paper we present a new type of data cube: the data value cube capable of storing both data record indices as well as data aggregates. This allows OLAP to drill down into data records and values, thus extending the capabilities of OLAP beyond simply presenting summaries and expanding the scope of applications for OLAP. This is particularly useful in applications that concentrate on detecting anomalies, a critical feature for applications in the increasingly critical security domain. We present our novel design and evaluate its performance, showing that the data value cube performs as well as existing data cubes while offering full data value access.

1 Introduction

OnLine Analytical Processing (OLAP) engines are commonly implemented as multidimensional data cubes - called hypercubes [2, 5, 7, 11, 12] 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 [8, 9].

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.

Current data cubes consist solely of data aggregates or summations. We are the first to propose extending the data cube to also maintain record values. This provides a more powerful OLAP to applications that are concerned with security and forensics.

2 Related Work

Currently, there are three kinds of OLAP data cube implementation [2, 5, 7, 11, 12].

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 [3, 4]. The star schema does not support OLAP - hence a data cube has to be created.

In Multi-dimensional OLAP (MOLAP), a multidimensional data cube (hypercube) is used as the data implementation. MOLAP does not support OnLine 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.

The MOLAP data cube was originally implemented as a multi-dimensional array [1,5,6,12]. Currently, most advanced OLAP applications implement the data cube as a tree forest [2, 5, 9, 10]. Each dimensional attribute is a separate and distinct tree. Each tree node points to another tree at the next dimension. In addition, each node contains a data aggregate for that attribute combination.

3 Solution

In this section we describe current MOLAP data cubes and their limitations. This motivates an extension to the data cube: the *data value cube*. We finish by showing how our solution extends the power of OLAP to a new class of problems.

Currently the best available data cube is a balanced b-tree forest [2, 5, 9, 10]. In general, each tree node (at a particular dimension) points to another tree at the next dimension. In addition, each tree node contains the aggregate for that combination of dimensions. Construction time for the tree forest is $O(\sum_{1 \leq i \leq d}(\log n_i))$ where d is the number of query dimensions and n_i is the number of attributes in the database. at level d.

One problem with the b-free forest is that it is limited to data aggregates. Data aggregates only identify the existence of a dimensional combination. They do not provide access to complete data records. With current OLAP implementations, examining data records requires issuing additional database queries, which is inefficient.

We solve this problem by extending a balanced b-tree forest to include references to data records. We call this new type of hypercube: the *data value cube*. Thus for our data cube, tree nodes not only contain data aggregates but a linked list of data record indices.

We claim that construction time for the data value cube is the same as a balanced b-tree forest. We prove this claim by noting that construction of our data cube is identical to the balanced b-tree forest, except for the addition of the linked list of data record indices. Adding an index to this linked list takes constant time, since the linked list is not sorted. And thus our proof is complete.

The data value cube increases the power of OLAP for applications that perform forensic analysis. For security applications, simple data aggregates are insufficient. Our solution enables OLAP to drill down past data aggregates and completely examine data records. Data aggregates identify data cube cells that are potentially of interest. Our data cube allows applications to verify data cube cell relevance and thus expand or contract the depth and breadth of queries.

4 Experimental Results

This section presents an application of the data value cube. 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 generic data value cube was implemented that used the same XSD to define the input data stream for the stream star schema. A star schema engine was implemented for the network data stream using MySQL. SQL GROUP BY queries were used to create data cubes from the star schema.

Both databases consisted of one million network data stream records (flows).

Network Data Stream queries were run on both databases. Query dimensions were from 1 to 16.

Figure 1 summarizes the performance results. Figure 1 shows that data value cube creation time was superior by a factor of at least 2. Figure 1 also shows that our data cube creation time was linear with respect to the number of dimensions and thus modular and predictable. This is highly beneficial to OLAP since the purpose of OLAP is to minimize query response time and improve the user experience.

The fact that data value cube creation time was so superior is an unexpected result. We were unable to completely determine why, since the MySQL source code was not available. Be that as it may, we now present some thoughts on the causes. First, MySQL has to support SQL in its entirety, while the data value cube is only concerned with GROUP BY queries. Second, the stream star schema was chunked with 1 million records. Perhaps the star schema chunking was not as efficient. And last, the star

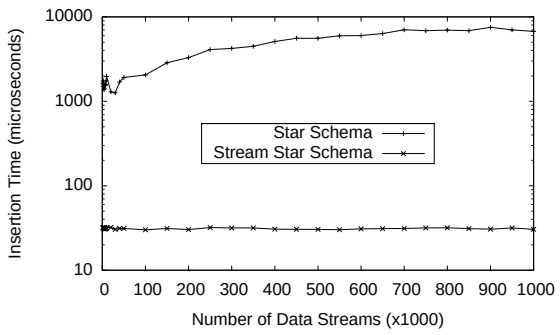

Figure 1: Query time (in seconds) for a stream star schema and star schema with million network data streams.

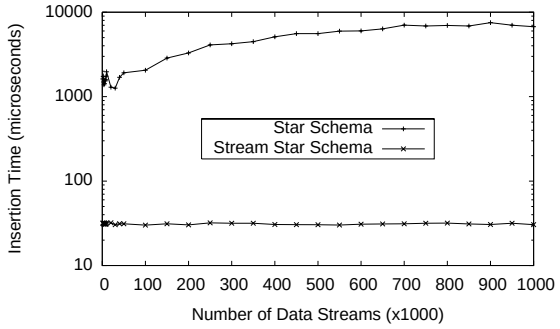

Figure 2: Memory usage (in Bytes) when querying a stream star schema with 1 million network data streams.

schema had 7 dimension tables for the network data stream. Our database had only 2 dimension tables.

Figure 2 shows the memory usage for the data value cube. Obtaining MySQL query memory usage was not feasible. Be that as it may, table 2 shows that memory usage for the data value cube was linear, scalable, and thus predictable.

5 Conclusion

This paper has presented a new type of OLAP data cube: the data value cube, capable of storing both data record indices as well as data aggregates. The time complexity of constructing such a data cube is $O(\sum_{1 \leq i \leq d}(\log n_i))$ where d is the number of query dimensions and n_i is the number of attributes in the database. at level d. This time complexity was verified through experimentation. So not only does the data value cube match the performance of existing data cubes, it allows OLAP to drill down into data records and values. This extends the capabilities of OLAP beyong simply presenting summaries and expands the scope of OLAP applications. This is of particular concern to applications that concentrate on detecting anomalies, a critical feature for applications in the increasingly critical security domain.

References

[1] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, *Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals*. 1997.

[2] 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.

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

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

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

[6] K. Ross and D. Srivastava, "Fast computation of sparse datacubes," in *23rd International Conference on Very Large Data Bases*, August 1997.

[7] 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.

[8] A. Shukla, P. M. Deshpande, J. F. Naughton, and K. Ramasamy, "Storage estimation for multidimensional aggregates in the presence of hi-

erarchies," in *22nd International Conference on Very Large Data Bases*, September 1996.

[9] 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.

[10] Y. Sismanis, N. Roussopoulos, A. Deligiannakis, and Y. Kotidis, "Dwarf: Shrinking the petacube," in *28th ACM SIGMOD/PODS International Conference on Management of Data/Principles of Data Systems*, June 2002.

[11] 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.

[12] 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.