Curio: A Novel Solution for Efficient Storage and Indexing in Data Warehouses

Anindya Datta
College of Computing.
Georgia Institute of Tech.
adatta@cc.gatech.edu

Krithi Ramamritham
Computer Science Dept.
Indian Institute of Technology, Bombay
krithi@cse.iitb.ernet.in

Helen Thomas
College of Computing.
Georgia Institute of Tech.
adatta@cc.gatech.edu

1 Introduction

Data warehousing and *On-Line Analytical Processing* (OLAP) are becoming critical components of decision support as advances in technology are improving the ability to manage and retrieve large volumes of data. *Data warehousing* refers to "a collection of decision support technologies aimed at enabling the knowledge worker (executive, manager, analyst) to make better and faster decisions" [1]. OLAP refers to the technique of performing complex analysis over the information stored in a data warehouse. It is often used by management analysts and decision makers in a variety of functional areas such as sales and marketing planning. Typically, OLAP queries look for specific trends and anomalies in the base information by aggregating, ranging, filtering and grouping data in many different ways [8]. Efficient query processing is a critical requirement for OLAP because the underlying data warehouse is very large, queries are often quite complex, and decision support applications typically require interactive response times.

Two main approaches for fast OLAP query processing have emerged:

1. *Precomputation Strategies.* This approach relies on *summary tables*, derived tables that house precomputed or "ready-made" answers to queries [1]. This has been, by far, the most explored area in the context of data warehouses [3].

2. *Ad-hoc Strategies.* This approach to fast OLAP query processing supports ad-hoc querying by using fast access structures on the base data such as *B^+-tree indexes*, *bitmapped indexes* [7], *bit-sliced indexes* [8] and *projection indexes* [8].

However, in these and other index structures proposed for OLAP, a separate set of indices or access structures is typically maintained in addition to the base data. Given the large size of data warehouses, storage is a non-trivial cost, and so is the additional storage requirement due to the index structures. This is especially true given that data and storage maintenance costs are often up to seven times as high per year as the original purchase cost [11]. Hence, a terabyte-sized system, with an initial media cost of $100,000, could cost an additional $700,000 for every year it is operational.

Curio, a data repository and OLAP query server, provides a potential solution to this problem. Curio is based on a novel design technique that allows fast access to data, yet *does not require indexes* [6]. Hence, Curio provides drastically im-

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

proved performance for ad-hoc queries, while *simultaneously* reducing the storage costs associated with warehousing. We briefly describe the underlying positional indexing techniques behind Curio and report results from a performance comparison with several leading commercial relational warehousing products in terms of query performance. We demonstrate the improved query performance of Curio over a specific product reported here, namely Oracle [9].

2 The Positional Indexing Approach

We now present an example to motivate the need for efficient storage and retrieval in data warehousing environments. Figure 1 contains a simple warehouse *star schema* [5], which models the 10-year history of the sales of an automobile manufacturer with dealerships located around the globe and is intended to support marketing and strategic decision-making.

In conventional database design, one envisions a set of relations or table structures, and a separate set of indices or access structures. In a relational system, tables would be stored "as-is": each table would be represented as a series of records partitioned over a chain of data blocks linked by *block pointers*. In addition to the base tables, for retrieval efficiency, index structures would typically be defined. More specifically, the `SALES` table will likely be indexed on each of its four dimensional attributes. A large number of indexing schemes have been proposed in the literature. Among these, four index types have been shown in [8] to be particularly appropriate for data warehousing/OLAP and they are indeed used in both commercial and research systems. These structures are *"standard" B^+-tree indexes*, *bitmapped B^+-tree indexes*, *projection indexes*, and *bit-sliced indexes*. While these structures have indeed improved query performance, they have imposed a significant space penalty. Certain schemes, such as the standard B^+-tree and bitmapped B^+-tree, incur much more storage overhead than others due to the nature of their structure. It is not uncommon for a standard or bitmapped B^+-tree index on a single column of a table to be 50 to 100% of the size of the table itself.

Curio is based on the notion of vertically partitioning a table into sets of attributes. By applying this idea, we can divide the `SALES` table in figure 1, into five smaller tables, as shown in figure 2. The new schema is then composed of 5 vertical partitions: one for each of the dimensional attributes (i.e., `SALES.TimeStamp`, `SALES.PoS ID`, `SALES.Profile ID` and `SALES.Product ID`), and one for the remaining columns from the original, i.e, the business metrics of the `SALES` table (i.e., `Tax`, `Discount` and `Status`). Each of these partitions can be considered a positional index. The resulting database size is essentially the same as the size of the raw data in the original database configuration. However, we can now utilize the separate dimensional columns of the partitioned fact table as *both elements of and indexes onto* that table. In terms of storage cost, the indexing is free. This indexing is made possible by a mapping that exists between the positional index and the original or reduced table such that one can easily associate the elements of a record in the positional index and the reduced table.

The ideas presented here have been extended to develop several additional proprietary data structures and algorithms that allow efficient join and aggregation operations.

3 Performance Evaluation of Curio

We now present selected results from a performance study, which compares the query processing speeds of Curio with those of several existing RDWMS products. We report the results for 3 specific products: Oracle (version 8.0.5) [9], Red Brick Warehouse (version 5.1.5) [10], and DB2 Universal Database (version 5.0) [4]. All tests were performed on a Windows NT machine having a single 300 MHz Intel Pentium processor and 64 MB of RAM. The queries used in this experiment are based on a star schema similar to the one shown in Figure 1. For each RDWMS product, indexes were built *only on those columns used in the test queries*, thus providing a significant advantage for the above mentioned products in terms of storage requirements. Typically, in a warehouse environment, indexes would be built on most (if not all) attributes, incurring significant additional storage overhead. Where possible, we used specialized indexes to give the other products as much of an advantage as possible. For example, bitmapped indexes were used in Oracle and star indexes in Red Brick.

The data for the schema were randomly gener-

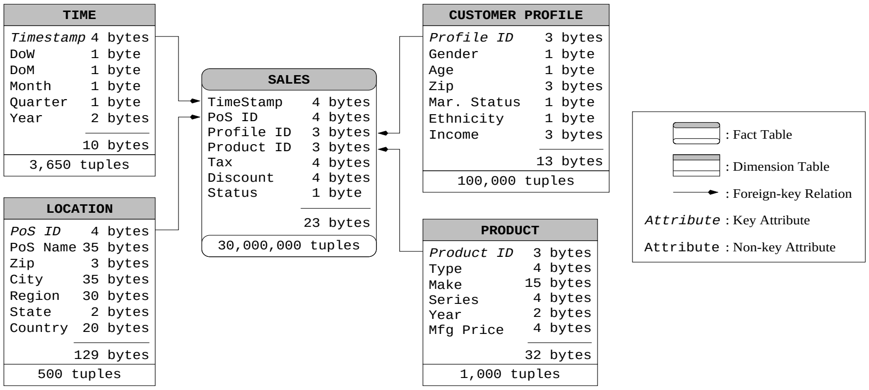

Figure 1: A Simple Warehouse Star Schema

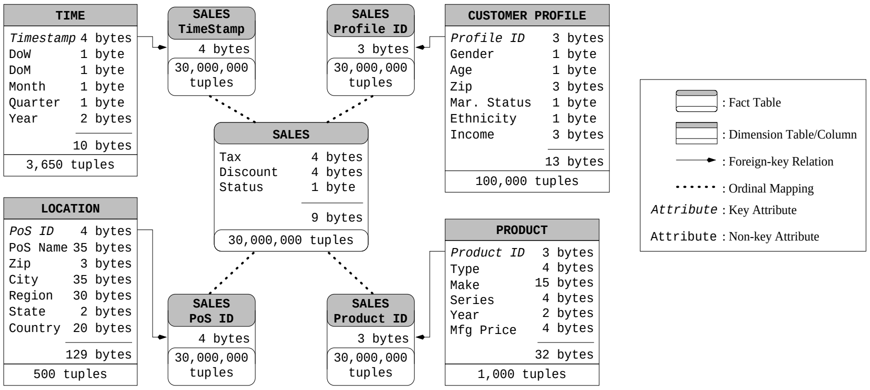

Figure 2: Example Warehouse Schema with Vertical Partitioning Index Scheme

ated and loaded into each RDWMS using their respective bulk load utilities. Three database sizes were considered in this experiment: 0.25 GB, 0.5 GB, and 1 GB. Database size here refers to the size of the raw data, and thus does not include any overhead that may be added once the data is loaded.

For each of the queries, we provide a plot of the response times for each RWDMS at each of the 3 data size levels. The results for all tests are presented using bar charts. The numbers above each bar for Red Brick, DB2, and Oracle are ratios of the performance compared to that of Curio. Two queries are shown in Figure 3, a 2-way join (Figure 3A) and a 4-way join (Figure 3B). Curio displays a remarkable peformance advantage over all other products, and this advantage increases with the query complexity. Two aggregation queries are shown in Figure 4, a 2-way join with 1 `GROUP BY` column (Figure 4A) and a 4-way join with 3 `GROUP BY` columns (Figure 4B). The performance advantage of Curio is similar, but on a greater scale.

4 Conclusion

We have presented Curio, an efficient OLAP query engine and data repository based on a simple data storage and indexing scheme. Clearly, one of the main advantages of this approach lies in the fact that essentially indexing is provided "for free". Our demonstration at VLDB shows that Curio is highly effective in reducing storage requirements while also delivering excellent query performance. To this end we demonstrate its superior performance with a set of typical OLAP queries and compare the run-times under Curio against a leading commercial relational warehousing product.

References

[1] S. Chauduri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, March 1997.

[2] International Data Corp. Data warehousing tools: 1998 worldwide markets and trends, report 17622, 1998.

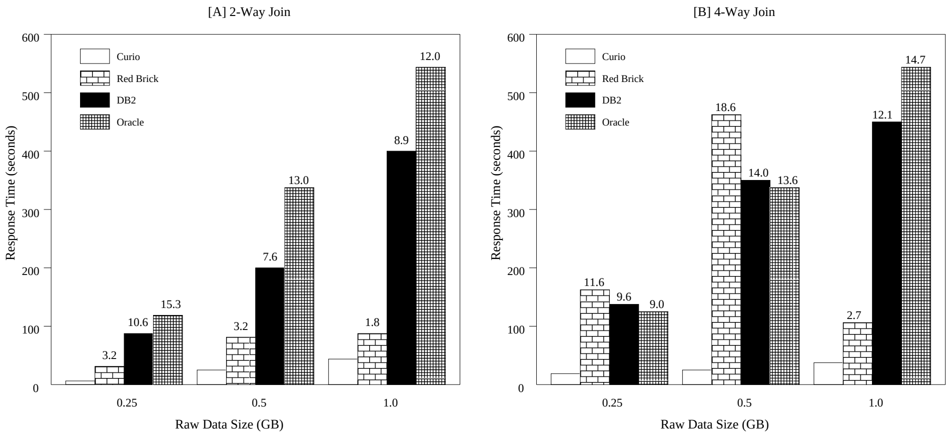

Figure 3: Response Times for Non-Aggregation Queries

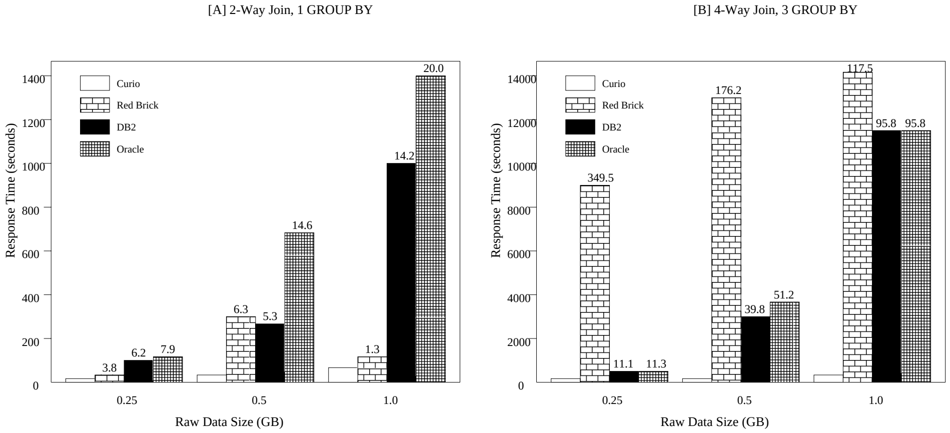

Figure 4: Response Times for Aggregation Queries

[3] V. Harinarayan, A. Rajaraman, and J.D. Ullman. Implementing data cubes efficiently. In *Proc. ACM SIGMOD*, pages 205–216, Montreal, Canada, June 4-6 1996.

[4] IBM Corp. Db2 universal database version 5.0 for windows nt, 1997.

[5] R. Kimball. *The Data Warehouse Toolkit*. J. Wiley & Sons, Inc., first edition, 1996.

[6] Muninn Technologies, LLC. Curio: A novel solution for efficient storage and indexing in data warehouses. White Paper, URL: `http://www.muninn.com`, 1999.

[7] P. O'Neil. Model 204 architecture and performance. In *2nd Intl. Workshop on High Performance Transaction Systems (HPTS)*, volume 359 of *Springer-Verlag Lecture Notes on Computer Science*, pages 40–59. Springer-Verlag, Asilomar, CA, 1987.

[8] P. O'Neil and D. Quass. Improved query performance with variant indexes. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 38–49, Tucson, AZ, May 13-15 1997.

[9] Oracle Corp. Oracle8 enterprise edition release 8.0.5 for windows nt, 1998.

[10] Red Brick Systems. Red brick warehouse version 5.1 for windows nt, 1998.

[11] D. Simpson. Corral your storage management costs. *Datamation*, pages 88–93, April 1997.