STAR SCHEMA BENCHMARK FOR DATA WAREHOUSING AND ADJOINED DIMENSION COLUMN CLUSTERING METHOD FOR IMPROVED STAR SCHEMA QUERY PERFORMANCE

A Dissertation Proposal Presented

by

XUEDONG CHEN

June 2, 2008

DOCTOR OF PHILOSOPHY

Computer Science Department
University of Massachusetts Boston

STAR SCHEMA BENCHMARK FOR DATA WAREHOUSING AND ADJOINED DIMENSION COLUMN CLUSTERING METHOD FOR IMPROVED STAR SCHEMA QUERY PERFORMANCE

A Dissertation Proposal Presented

by

XUEDONG CHEN

Directed by Professor Patrick O'Neil

Approved as to style and content by:

____
Dan Simovici, Professor, Graduate Program Director
Committee Member

____
Patrick O'Neil, Professor
Committee Member

____
Elizabeth O'Neil, Professor
Committee Member

____
Donghui Zhang, Assistant Professor, Northeastern University
Committee Member

1. Introduction

The Star Schema design is commonly used in a query-mostly data warehouse. A Star Schema consists of a Fact table and various Dimension tables as shown for example in Figure 2. Star Schema queries typically retrieve aggregates of Fact table measurements with restrictions on Dimension table columns [1]. TPC-H is a standard benchmark (schema in Figure 1) that many database venders use to claim superior performance for data warehousing [2]. However, TPC-H isn't in Star Schema design. It is unnecessarily complicated for practical data warehousing. We designed a Star Schema Benchmark (SSB, Figure 2) based on the standard TPC-H benchmark, and used it as a basis for data warehouse performance measurement.

Hard Disk technology progress over the past 15 years has greatly improved sequential scan speed. DBMSs on data warehouses tend to apply sequential scan for Star Schema Queries. As we will see in section 3, clustering of physical data is quite important for data warehouse performance with modern disk technology. DB2's Multi-Dimensional Clustering (MDC) introduced in 2003 [3], provides methods to "dice" the fact table along a number of orthogonal "dimensions", which must however be columns in the fact table. MDC is an excellent new approach to clustering, but none of the current clustering methods can leverage the information of the dimension tables in the Star Schema database.

We present Adjoined Dimension Column (ADC) clustering, a methodology to bring the dimension table columns into the fact table and create clustering based on those adjoined dimension columns.

The rest of this proposal is organized as follows. In Section 2 we present the design of the SSB. We explore in Section 3 the DB2 MDC clustering method and the design of ADC clustering. Section 4 presents some experiment results for the significant query performance improvement with ADC clustering. Finally, Section 5 discusses future work for ADC clustering.

2. Star Schema Benchmark (SSB)

The schema of the SSB (Figure 2) provides a Star Schema based on the schema of the TPC-H benchmark. The queries are also based on a few of the TPC-H queries, and they are all Star Schema Queries. The transformations from TPC-H to SSB are guided by principles explained in [1].

- Create SSB LINEORDER table. We combine that LINEITEM and ORDER table in TPC-H to make a LINEORDER table. This denormalization is standard in data warehousing, and makes many joins unnecessary in common queries. Some columns of LINEITEM and ORDER are dropped and altered.
- Drop PARTSUPP table. We drop the PARTSUPP table of TPC-H because of a grain mismatch. The PARTSUPP table has what is called a Periodic Snapshot grain which does not match the finest transaction level grain in the LINEORDER table. Thus queries which combine PARTSUPP with LINEORDER will be meaningless.
- Drop tables NATION and REGION. We drop the tables NATION and REGION, which would turn the SSB star schema into a snowflake schema.

Such tables, which add joins to TPC-H queries, may be appropriate in an OLTP system to enforce integrity, but not in a warehouse.

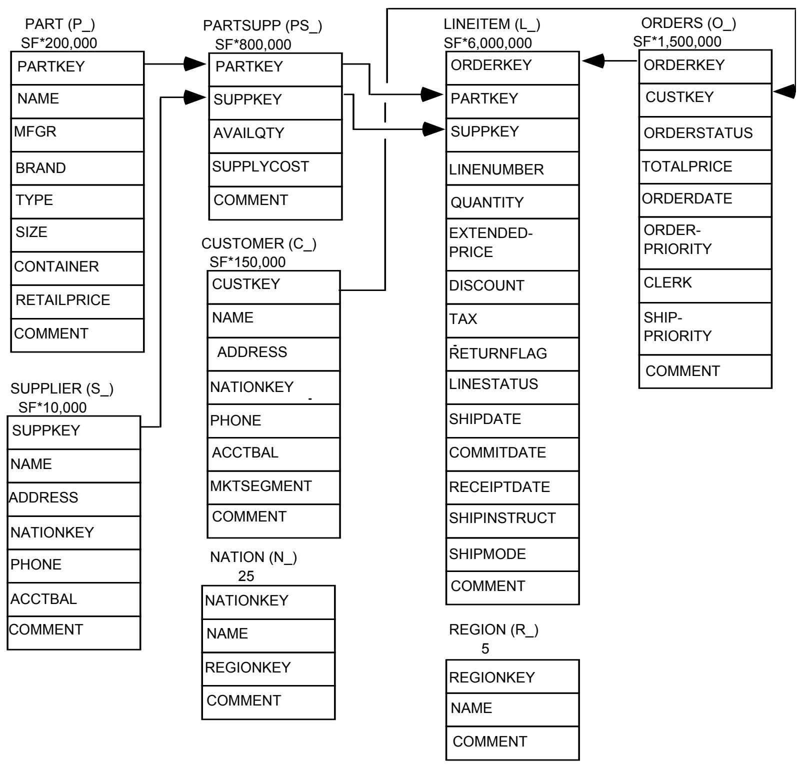

Figure 1 TPC-H Schema

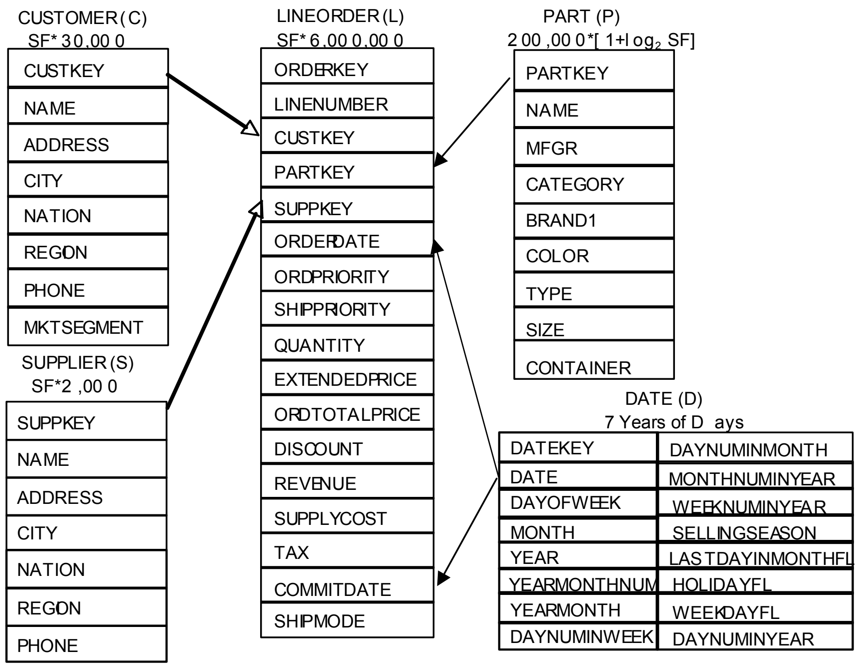

Figure 2 SSB Schema

The classic data warehouse queries appearing in SSB select from the LINEORDER table exactly once, with no self-join, and have predicate restrictions on dimension table attributes. We make an attempt to provide Functional Coverage and Selectivity Coverage features by defining four Query Flights with restriction on various numbers of dimension tables.

- Query Flight Q1, based on TPC-H query TPCQ6, has a restriction on one dimension and two LINEORDER columns. The query measures the revenue increase from eliminating various ranges of discounts in given product order quantity intervals shipped in a given year. Query Flight Q1 has three queries with different select Filter Factors.
- Query Flight Q2, has restrictions on two dimensions. The query compares revenues for certain product classes and suppliers in a certain region, grouped by more restrictive product classes and all years of orders. Query Flight Q2 also has three queries in it.
- Query Flight Q3, based on TPC-H query TPCQ5, has restrictions on three dimensions. The query retrieves total revenue for LINEORDER transactions within a given region in a certain time period, grouped by customer nation, supplier nation and year. There are four queries in this Query Flight
- Query Flight Q4, provides a "What-If" sequence of queries that might be generated in an OLAP style of exploration. Starting with a query with rather weak constraints on three-dimensional columns, we retrieve aggregate profit grouped by year and customer nation. Three queries in this Query Flight modify the predicate constraints by drilling down to find the source of an anormaly

Figure 3 shows the Filter Factor of all Query Flight queries. The dissertation will present the design for SSB tables and queries in great detail.

Query	**FF on lineorder**	**FFs of indexable predicates on dimension columns**				**Combined FF Effect on lineorder**
	FF on discount & quantity	FF on Date	FF on part Brand1 roll-up	FF on supplier city roll-up	FF on customer city roll-up	
Q1.1	.47*3/11	1/7				.019
Q1.2	.2*3/11	1/84				.00065
Q1.3	.1*3/11	1/364				.000075
Q2.1			1/25	1/5		1/125 = .0080
Q2.2			1/125	1/5		1/625 = .0016
Q2.3			1/1000	1/5		1/5000 = .00020
Q3.1		6/7		1/5	1/5	6/175 = .034
Q3.2		6/7		1/25	1/25	6/4375 = .0014
Q3.3		6/7		1/125	1/125	6/109375 =.000055
Q3.4		1/84		1/125	1/125	1/1312500= .000000762
Q4.1			2/5	1/5	1/5	2/125 = .016
Q4.2		2/7	2/5	1/5	1/5	4/875 = .0046
Q4.3		2/7	1/25	1/25	1/5	2/21875 = .000091

Figure 3 SSB Queries Filter Factors

3. Adjoined Dimension Column (ADC) Clustering

Over the past twenty years, the performance of indexed retrieval with a moderate sized filter factor has lost its competitive edge compare to sequential scan of a table. We re-measured the performance for Set Query Benchmark (SQB) [4] on DB2 UDB, and discovered that for other than "needle-in-haystack" queries, DB2 UDB (the modern UNIX DB2) prefers to use series of sequential scans, rather than using secondary index access to rows as was the case 18 years ago with MVS DB2. Because of this the only way to speed up sequential scan is to restrict certain query ranges to a smaller range of disk, an approach known as "clustering". Because of this, clustering has become more crucial than ever with modern disk accesses.

Many databases provide simple index clustering on single columns of a table. Such clustering does very well when there is one standout among columns to sort the data that will speed up most queries of interest. But what if there is not? DB2 was the first database product to provide an ability to cluster by more than one column at a time. DB2's *Multi-Dimensional Clustering* partitions table data into cells (physically organized as *Blocks* in MDC), by treating some columns within the table as orthogonal axes of a cube, each cell corresponding to a different combination of individual values of these cube axis columns. A *Block* in MDC is a contiguous sequence of pages on disk identified with a table extent, and a block index is created for each "dimension" axis. Every value in one such block index is followed by a list of Block Identifiers (BIDs), forming what is called a *Slice* of the multi-dimensional cube corresponding to a value of one dimension. The set of *BIDs* in the intersections of slices for values on each axis is a *Cell.*

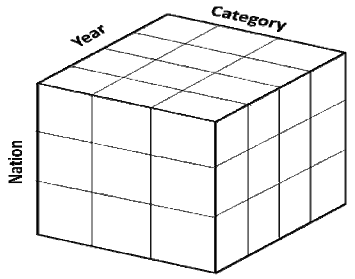

Figure 4 MDC Cube

The "dimensions" of a table in MDC are columns within the table, and all equal-match combinations of dimension column values are used to define cells. For a data warehouse, the star schema query's Where clause typically has restrictions on the columns of dimension tables. We need a way to cluster the fact table based on the columns in the dimension tables.

Our ADC approach adjoins physical copies of a several dimension columns to the fact table. We choose columns at a rather high level in some hierarchy commonly restricted in queries, e.g., Customer_nation or Part_department. The point of using only a few high-level hierarchy columns is to generate a relatively small number of conjoint values of the columns making up cells in the cube. Thus we ensure that the cells contain enough data that sequential access within the cell can outweigh disk inter-cell access. The right number of cells depends on the size of the fact table and disk performance. A formal mathematical formula for calculating the right number of cells will be presented in the dissertation.

ADC is a clustering method that can be implemented differently in various DBMS products. With DB2, we can create a materialized view for the new Fact table with adjoined dimension columns. Then we apply the native MDC to the materialized view. With Oracle, we can also create a materialized view for the new Fact table with adjoined dimension columns. However we have to leverage the Partition feature [5] to achieve the dimensional cubing into cells. The detailed implementation of ADC clustering for both of these DBMS products will be presented in the dissertation.

4. ADC Experimental Results on SSB

We measured three commercial database products, anonymized with names A, B and C, using SSB tables at Scale Factor 10 (SF10). Figure 4.2 shows the significant star schema query performance improvement with ADC clustering. The full experiment results and analysis will be presented in the dissertation

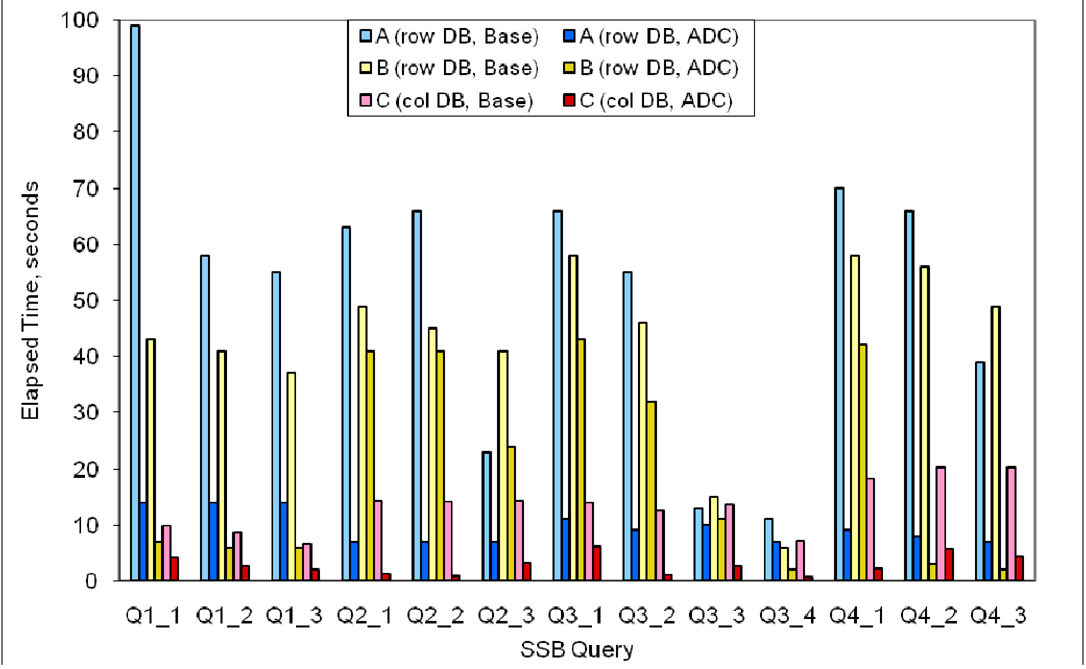

Figure 5 SSB Query time with/without ADC

5. ADC Future work

We have published a poster paper “Adjoined Dimension Column Index (ADC Index) to Improve Star Schema Query Performance” in ICDE 2008, which gives an introduction to this work [6]. We will publish an extended ADC clustering paper with more experimental results and analysis.

To build an ADC clustering for a large data warehouse, a database practitioner needs to analyze varieties of ADC column combination in order to find out the optimal ADC column combination. The entire point of a star schema design is to support a reasonably thin fact table, which means keeping most columns in the dimension tables. Only the ADC clustering dimension columns that can greatly accelerate the Star Schema queries earn their place in the fact table. Looking beyond the dissertation, it will be a great challenge for us to implement an ADC advisor application. With the help of this application, database practitioners can import the statistics of an existing Data Warehouse, set of queries as well as some other parameters. The ADC advisor application will use the generic query optimization scheme to analyze the queries and their pattern. The ADC advisor estimates the best ADC plan and provides the estimated query performance improvement results.

References

[1] Kimball, R. and Ross, M, The Data Warehouse Toolkit, Second Edition, Wiley, 2002.

[2] TPC-H Version 2.4.0 in PDF Form from; http://www.tpc.org/tpch/default.asp

[3] IBM Research, DB2's Multi-Dimensional Clustering.
http://www.research.ibm.com/mdc/

[4] O'Neil, P. “The Set Query Benchmark.” Chapter 6 in The Benchmark Handbook for Database and Transaction Processing Systems, Jim Gray, Ed., Morgan Kauffmann, 1993, pp. 209-245. Download: http://www.cs.umb.edu/~poneil/SetQBM.pdf

[5] Partitioning in Oracle Database 10g Release 2, May 2005
http://www.oracle.com/solutions/business_intelligence/partitioning.html

[6]Chen, X O’Neil, P and O’Neil, E, “Adjoined Dimension Column Clustering to Impove Data Warehouse Query Performance.” ICDE 2008, pp. 1409-1411

[7]Mike Stonebraker, Daniel Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran and Stan Zdonik. “C-Store: A Column Oriented DBMS.” VLDB 2005, pp. 553-564