ISSN No. 0976-5697

Volume 6, No. 4, June 2015 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Partitioning Retail Database for Performance Improvement

Anil Rajput
Professor Computer Science & Mathematics
Govt.C.S.A.P.G.College, Sehore, Madhya Pradesh
e-mail:dranilrajput@hotmail.com

Kshmasheel Mishra
Former Reader Computer Science
Vikram University,Ujjain,Madhya Pradesh
e-mail:mishra_ks@hotmail.com

Vaibhav Khanna
Assistant Professor Dezyne E'cole College,
Civil lines, Ajmer, Rajasthan
e-mail: vaibhavajmer@yahoo.com

Abstract: Retail data analysts have a challenging task to analyze large datasets relating to retail stores. With the advent of materialized views, a database administrator creates one or more materialized views to expedite query results. This requires a proactive decision making on the right partitioning strategy during the logical design phase and the same should be implemented in the physical database schema. This in turn requires experimentation with various alternative strategies for measured performance improvements before suggesting these strategies as standard data models. This paper outlines the performance improvement techniques relating to table partitioning experiments conducted on retail sample data set. The partitioning strategies were initially applied on a representative subset of the real data (10K and 100K records). Then the experiments were conducted to measure performance improvements on the entire model data (10 lakh records**). S**ignificant performance benefits were achieved in the experiments conducted during this study using right partitioning strategy. These strategies can positively contribute towards improving the overall user satisfaction.

Keywords: Multidimensional databases, performance, partitioning, interval partitioning, range partitioning, hash partitioning, database performance experiments.

I. INTRODUCTION: BUSINESS INTELLIGENCE ORIENTED PARTITIONING IN RETAIL DATABASES

Business analysts, information and data analysts, market analysts and sales analysts have a challenging task to analyze large datasets relating to retail stores and periodically generate the Business Intelligence Reports. This includes all retailers - including department stores, specialty store chains, mass merchants, convenience stores, restaurants, grocery stores, and multi-channel retailers. All of these retail businesses need data analysis capabilities to develop retail-specific insights that are relevant, actionable, and can improve revenues as well as profits.

The retail stores frequently requires summarized, aggregated information or may need to quickly navigate through the data (drill-down transaction details) to better understand business issues. Often these are made available as 'Materialized Views in the Retail Data Model'. Materialized views are query results that have been stored or "materialized" in advance as schema objects[3]. From a physical design point of view, materialized views resemble tables or partitioned tables to improve performance. In the past, organizations using summaries spent a significant amount of time and effort creating summaries in a customized fashion, identifying which summaries to create, indexing the summaries, updating them, and advising their users on which ones to use. With the advent of materialized views, a database administrator creates one or more materialized views, which are the equivalent of a summary.[3] This requires a proactive decision making on the right partitioning strategy during the logical design phase and implemented in the physical database schema. This in turn requires experimentation with various alternative strategies for measured performance improvements, before one could suggest these strategies as standard data models[13].

This paper outlines the performance improvement techniques relating to table partitioning experiments conducted on retail sample data set. The data in the core sample data set contained more than 10 lakh rows of data. The experiment was carried out with a random subset of data containing 10 K records and then with 1 lakh records.

Dimensional modeling is the core of any Business Intelligence or Business Analytics activity. This may require the redesigning of existing normalized schema to dimensional (star) schema: a central fact table surrounded by dimension tables. The fact table is typically the business process for which measurement is needed. The dimension tables contain the business dimensions and their attributes. These attributes are to be used to constrain the records in the fact table, and to provide the values used for aggregating grouping and sorting the results.

II. DIMENSIONAL ANALYSIS AND STAR QUERY PERFORMANCE IMPROVEMENT IN RETAIL DATABASES

In dimension tables the dimensions are stored, usually in a de-normalized form. The redundancy is intentionally created in the dimensional table to eliminate the need for multiple joins on dimension tables. In a star schema, only one join is needed to establish the relationship between the fact table and any one of the dimension tables.[13] A star query is a join between a fact table and a number of dimension tables. Each lookup table is joined to the fact table using a primary-key to foreign-key join, but the lookup tables are not joined to each other.[9] A classical star-query from a data-mining application may have to generate scheduled analytical reports such as:

CONFERENCE PAPER
National Conference on
"Internet Technology and Cyber Crime"
On 11 January 2015
Organized byPraful Narooka and
Sponsored by Agarwal College, Merta City (Nagpur)

- Customer Analytics
- Promotion Analytics
- Category Analytics
- Merchandise Analytics
- Workforce Analytics
- Point of Service Analytics
- Loss Prevention Analytics
- Inventory Analytics
- Order Management Analytics
- Store Operation Analytics

The customer analytics includes the demography analysis or frequent shopper analysis and cluster analysis for STP decisions. Often retail chains require Customer Loyalty Profile, Customer Loyalty Factor Rank, Customer Loyalty Prediction indicators for business growth predictions.[16]

The promotion analytics includes the Actual and Plan Analysis, Performance analysis, and Response Analysis. For instance Promotion Comparison Analysis provides response rate information for customers in the target group, all of whom received promotion, by frequency segment. This is used for understanding the performance of any typical promotional effort and its contribution by Promotion Event Type. These may have to be published as Promotion Scorecard, Promotional Performance Reports, and Promotion Impact Analysis.

The category analysis is the most complex of these tasks in retail chain networks and may include the following areas:

- Contribution Analysis (Item Scorecard, Item Profit on Cost, Pack Sales, Profit on Net Cost, Cost Trend, Class Item Level Profit) providing sales contribution information based on department.[16]
- Performance analysis includes following reports:
  - Bottom Performance by Sales,
  - Space Performance,
  - Bottom Performance % Profit,
  - Top Performance, and
  - Top by Store
- Pricing analysis enables comparison of product pricing for stores and items[2]
- Product Analysis area includes Sales by Banner, Listing (Bottom), Sales by Channel, and Listing by Category.[2]
- Sales Analysis area includes the Sales by Channel, Cross Sell Analysis, and Vendor Sales by Channel Analysis
- Sales and Profit area includes the following reports: Gross Profit, Sales Profit, Return by Channel (with store for local values), Sales Profit & Return by Channel, Sales Profit by Product Cluster, Product Cluster Sales and Profit, Sales & Profit by Customer Cluster, and Sales Profit & Return by Location Trait etc
- Product Price Elasticity analysis is very important for sales prediction in a dynamic setting of changing variables.[10]
- Product Category Mix Analysis is used to map the product range.

These and many other similar analysis reports requires proactive performance tuning and partitioning of databases for efficient star query response. Materialized views improve query performance by pre-calculating expensive join and aggregation operations on the database before executing and storing the results in the database. Aggregate objects are implemented using either materialized views or table partitioning. As there is a large volume of data partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt.[1]

III. SELECTING A PARTITIONING STRATEGY FOR RETAIL DATA MODEL

Partitioning is a method of physically storing the contents of a data model. It improves the performance of multidimensional data in the following ways:

- Improves scalability by keeping data structures small.
- Each partition functions like a smaller measure.
- Keeps the working set of data smaller both for queries and maintenance.
- Data that is used together - - is stored together.
- Enables parallel aggregation during data maintenance.
- Each partition can be aggregated by a separate process.
- Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.
- The number of partitions affects the database resources that can be allocated to loading and aggregating the data.
- Partitions can be aggregated simultaneously when sufficient resources have been allocated.

As the partitioning strategy is driven primarily by life-cycle management considerations, so the partitions in this study were made on the Time dimension. One obvious advantage of the same is that the old time periods can then be dropped as a unit, and new time periods added as a new partition. [17]

The level on which to partition the physical data model was determined based on a tradeoff between load performance and query performance. Typically, we cannot afford a partition on too low a level (for example, on the DAY level) because then too many partitions must be defined at load time which slows down an initial or historical load.[4] Also, a large number of partitions can result in unusually long Analytic Workspace Attach Times and slows down the Time Series-based calculations. Also, a Quarterly Cumulative measure (Quarter to Date Measure) needs to access 90 or 91 partitions to calculate a value for one Customer. Usually a good compromise between the differing load and query performance requirements is to use an intermediate level like MONTH as the partition level. The usage will depend on the individual acumen of the DBA and there can be no standard solution that works best in all situations.[6] The partitioning strategy used in the current experiment was:

Table 1 : Interval Partitioning on week key

Physical Table Name	Partition Type	Partition Key Column
CRTFCT_ACTVTY_DAY	Interval	Day Key
INV_POSN_DEPT _DAY	Interval	Day Key
PCHSE_ORDR_LI_DAY	Interval	Day Key
PCHSE_ORDR_SBC_DAY	Interval	Day Key

RTL_SL_RETRN_DEPT_DAY	Interval	Day Key

Table 2 : Interval Partitioning on week key

Physical Table Name	Partition Type	Partition Key Column
CARRIER_CMPLNC_WK	Interval	Week Key
CUST_TYP_ORDR_SBC_WK	Interval	Week Key
INV_RCPT_SBC_WK	Interval	Week Key

Table 3 : Interval Partitioning on Month key

Physical Table Name	Partition Type	Partition Key Column
ACCT_PAYBL_MO	Interval	Month Key
ACCT_RCVBL_MO	Interval	Month Key
ACTVTY_RQST_MO	Interval	Month Key
ASSTS_MO	Interval	Month Key
COST_MO	Interval	Month Key
CUST_EMP_RLTNSHP_MO	Interval	Month Key
CUST_EMP_SL_RETRN_MO	Interval	Month Key
CUST_TYP_ORDR_DEPT_MO	Interval	Month Key
INV_POSN_SBC_MO	Interval	Month Key
PCHSE_ORDR_DEPT_MO	Interval	Month Key

For a few other cases the partition was conducted on a different on a different dimension. This is suitable for the cases where the decision support queries are organized into any other dimension e.g. region. This speeds up the queries for a particular region by not having to scan an extra dimension (region). This was more applicable to the situations where there is no definable active time period.

Table 4 : Hash Partitioning on Business Unit Key

Physical Table Name	Partition Type	Partition Key Column
ACCT_PAYBL_MO	Hash	Business Unit Key
ACCT_RCVBL_MO	Hash	Business Unit Key
CUST_TYP_ORDR_SBC_WK	Hash	Business Unit Key
CARRIER_CMPLNC_WK	Hash	Business Unit Key
CRTFCT_ACTVTY_DAY	Hash	Business Unit Key
INV_POSN_DEPT_DAY	Hash	Business Unit Key

Partition by a pre determined size of the table was done for the cases where there is no clear basis of partitioning the fact table on any dimension. A new partition of the fact table was created when it is about to exceed a pre-determined size. This requires metadata to identify the content of each partition.

IV. RESEARCH METHODOLOGY

Multiple problem queries were run on the sample data in its raw state (10 lakh + records in fact table) and many of them could not complete even after several hours. Application of the partitioning strategies resulted in completing the query within less than 30 minutes. The partitioning strategies were initially applied on a representative subset of the real data. Then the research experiments were carried out on the entire model data (10 lakh records) after getting substantial improvement (40X) in performance on the representative data set (1 lakh records).

Materialized views were used to pre-compute and store joins and aggregated data there by eliminating the overhead associated with expensive joins and aggregations queries.[2] One of the major benefits of using materialized views is the query rewrite capability. It is a query optimization technique that transforms a user query written in terms of tables and views, to execute faster by fetching data from materialized views. It is completely transparent to the end user, requiring no intervention or hints in the SQL application because the database server will automatically rewrite any appropriate SQL application to use the materialized views.

The important init.ora parameters that are relevant for the current experiment set are:

CHECKPOINT_Process to turns the CKPT background process as the retail data analysis queries are mostly read only, so it will improve the overall performance by effective utilization of the CPU cycles.

STAR_TRANSFORMATION_ENABLED to a TRUE value will enable star transformation of the queries. This also improves the query performance timing as it uses star transformation algorithm when optimizing star queries and works well with a large number of dimensions and sparse fact tables.[8]

ENABLE_HASH_JOIN will enable the hash joins, which can be much faster than sort merge joins, especially when one input is much smaller than the other.

V. EXPERIMENT QUERY PLAN

Table 5: Experiment Query Plan

ID	PARENT PROCESS	OPERATION	OBJECT_NAME
0		SELECT STATEMENT (S)	
1	0	SORT ORDER BY	
2	1	CUSTOMERS NESTED LOOPS OUTER	
3	2	SESSIONS NESTED LOOPS	
4	3	ORDER_HEADERS NESTED LOOPS OUTER	
5	4	PRODUCTS NESTED LOOPS OUTER	
6	5	ASSORTMENTS NESTED LOOPS OUTER	

7	6	PROMOTIONS NESTED LOOPS OUTER	
8	7	CONTENT NESTED LOOPS OUTER	
9	8	CAMPAIGN NESTED LOOPS OUTER	
10	9	TABLE ACCESS FULL	ANALYSE_ORDERS_ CLICKS_GIFTS
11	9	TABLE ACCESS BY INDEX ROW	ANALYSE_CAMPAIG N
12	11	INDEX UNIQUE SCAN	XPK ANALYSE_CAMPAIG N
13	8	TABLE ACCESS BY INDEX ROWI	ANALYSE_CONTENT
14	13	INDEX UNIQUE SCAN	XPK ANALYSE_CONTENT
15	7	TABLE ACCESS BY INDEX ROWID	ANALYSE_PROMOTI ONS
16	15	INDEX UNIQUE SCAN	XPK ANALYSE_PROMOTI ONS
17	6	TABLE ACCESS BY INDEX ROWID	ANALYSE_ASSORTM ENTS
18	17	INDEX UNIQUE SCAN	XPK ANALYSE_ASSORTM ENTS
19	5	TABLE ACCESS BY INDEX ROWID	ANALYSE_PRODUCT S
20	19	INDEX UNIQUE SCAN	XPK ANALYSE_PRODUCT S
21	4	TABLE ACCESS BY INDEX ROWID	ANALYSE_ORDER_H EADERS
22	21	INDEX UNIQUE SCAN	XPK ANALYSE_ORDER_H EADERS
23	3	TABLE ACCESS BY INDEX ROWID	ANALYSE_SESSIONS
24	23	INDEX UNIQUE SCAN	XPK ANALYSE_SESSIONS
25	2	TABLE ACCESS BY INDEX ROWID	ANALYSE_CUSTOME RS
26	25	INDEX UNIQUE SCAN	XPK ANALYSE_CUSTOME RS

VI. Original Query Performance and Query Performance Bounds

The experiment set was initiated with zero intervention stage and no partitioning

Original Query timings with no partitioning

Table 6 : Query timings with no partitioning

Fact table size (#rows)	Query response time (sec)
10,000	108
1,00,000	1701
10,00,000	36900

Query Performance Bounds

To conduct experiments and measure performance we established three bounds on query performance in order to understand the range of performance improvement for the query:

- The original plan with no tuning is indicative of the worst-case performance for the query. This plan is shown in table 7.
- The lower bound is the hypothetical best case in which all SQL operations other than reading/writing data take zero time. The query result was materialized into a new table. The Materialized-Query plan simply reads this new table and writes it to another table rather than execute the original SQL.
- The Best execution time shows the maximum reduction achieved in query execution time.
- The output of a query was always written to a temporary table so as to reduce the time taken for output.
- Candidate techniques for performance improvements were tested with reduced fact table sizes of 10,000 and 100,000 rows to keep the turn-around time of experiments within practical limits.

VII. Findings / Results Achieved After Interventions and Partitioning

The representative results for a typical star query were:

Table 7: Experiment Results after partitioning

Fact table size (#rows)	Original Query timings with no partitioning (sec)	Lower-bound using Materialized-Query plan (sec)	Best execution time (sec)	Improvement (with respect to original plan)
10,000	108	7	18	6x
1,00,000	1701	40	196	8.6x
10,00,000	36900	220	1220	30.2x

VIII. Conclusion

Partitioning is a very strong functionality to logically partition objects into smaller pieces. Partitioning is generally conducted for Performance Improvement, Manageability, and Availability and is often driven by business requirements. This paper demonstrates how significant performance benefit can be achieved using right partitioning strategy. These performance improvements optimize computing abilities, storage usage and improve overall user satisfaction. Records used together are grouped together and so it generally is much faster to query a multiple small tables than a single large table.[18] Also each partition can be optimized for performance, security and recovery. Retail Business Intelligence Models are often needed for Employee Analysis, Customer Analysis, Store Analysis, Item Analysis, and Product Analysis etc. This paper touches upon the important aspects of Enterprise Data Management, Business Intelligence Modeling and Information Lifecycle Management. A well designed table partitioning strategy has a big role in ensuring user efficiency and satisfaction and can be useful in multiple forms.

IX. REFERENCES

[1] Avnur R, Hellerstein JM. Eddies: Continuously adaptive query processing. Proceedings of ACM SIGMOD Conference, Dallas, TX, 2000. ACM Press, 2000.

[2] Baralis E, Paraboschi S, Teniente E. Materialized view selection in a multidimensional database. Proceedings of the Data Partitioning Conference, Athens, 1997. Morgan Kaufmann: San Francisco, CA, 1997

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

[4] DeWitt DJ, Gray J. Parallel database systems: The future of high performance database systems. Communications of the ACM 1992; 35(6):85–98.

[5] Gray J et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining and Knowledge Discovery, vol. 1, Fayyad U, Mannila H, Piatetsky-Shapiro G (eds.), 1997; 29–53.

[6] Gupta A, Mumick IS. Maintenance of materialized view: Problems, techniques, and applications. Data Engineering Bulletin 1995; 18(2):3–18.

[7] Bouganim L, Florescu D, Valduriez P. Dynamic load balancing in hierarchical parallel database systems. Proceedings of the 22nd VLDB Conference, Bombay, 1996. Morgan Kaufmann: San Francisco, CA, 1996.

[8] Lu H, Tan KL. Dynamic and load-balanced task-oriented database query processing in parallel systems. Proceedings of the 3rd EDBT Conference, Vienna, 1992 (Lecture Notes in Computer Science, vol. 580). Springer: Berlin, 1992.

[9] Manegold S, Obermaier JK, Waas F. Load balanced query evaluation in shared-everything environments. Proceedings of the 3rd Euro-Par Conference, Passau, 1997 (Lecture Notes in Computer Science, vol. 1300). Springer: Berlin, 1997.

[10] Nodine MH, Vitter JS. Deterministic distribution sort in shared and distributed memory multiprocessors. Proceedings of the 5th ACM SPAA, Velen, 1993. ACM Press, 1993.

[11] O'Neil P, Graefe G. Multi-table joins through bitmapped join indices. ACM SIGMOD Record 1995; 24(3):8–11.

[12] O'Neil P, Quass D. Improved query performance with variant indexes. Proceedings of the ACM SIGMOD Conference, Tucson, AZ, 1997. ACM Press, 1997.

[13] Qureshi W. A performance analysis of alternative multi-attribute declustering strategies. Proceedings of the ACM SIGMOD Conference, San Diego, CA, 1992. ACM Press, 1992.

[14] Rahm E, St¨ohr T. Analysis of parallel scan processing in parallel shared disk database systems. Proceedings of the 1st Euro-Par Conference, Stockholm, 1995 (Lecture Notes in Computer Science, vol. 966). Springer: Berlin, 1995.

[15] Rahm E, M¨artens H, St¨ohr T. On flexible allocation of index and temporary data in parallel database systems. Proceedings of the 8th HPTS Workshop, Asilomar, 1999. http://research.microsoft.com/∼gray/hpts99/

[16] Spiliopoulou M, Freytag JC. Modelling resource utilization in pipelined query execution. Proceedings of the 2nd Euro-Par Conference, Lyon, 1996 (Lecture Notes in Computer Science, vol. 1123/1124). Springer: Berlin, 1996.

[17] Stohr T, M¨artens H, Rahm E. Multi-dimensional database allocation for parallel data warehouses. Proceedings of the 26th VLDB Conference, Cairo, 2000. Morgan Kaufmann: San Francisco, CA, 2000.

[18] Zhou X, Orlowska ME. Handling data skew in parallel hash join computation using two-phase scheduling. Proceedings ICA3PP Conference, Brisbane, 1995. IEEE Computer Society Press: Washington, DC, 1995