International Journal of Computer Applications Technology and Research
Volume 6–Issue 1, 32-37, 2017, ISSN:-2319–8656

# Performance Tuning of Data Warehouse

Srikanth Kumar Tippabhotla
CTS

**Abstract**: There are many data warehouse performance aspects. and knowing all the different performance aspects. issues and considerations for building your data warehouse can be a tremendous task. Learn how to build your data warehouse for maximum SOL access performance while maintaining the key values for partitioning and parallelism considerations. Through this article you will learn many design options and alternatives to maximize performance for your high performance data warehouse design.

**Keywords**: Performance, Data warehouse

## 1. INTRODUCTION

There are many data warehouse performance aspects, and knowing all the different performance aspects, issues and considerations for building your data warehouse can be a tremendous task. DB2 offers many performance aspects missing from other DBMSs that can make a huge performance difference. This article will highlight the performance advantages of DB2 and how to handle all the design issues, alternatives and considerations experienced while building a large high performance data warehouse.

This article also explores how to build your data warehouse for maximum SOL access performance while building or maintaining the key values for partitioning and parallelism considerations. Since no data warehouse should be design as an island of information, OLAP tool performance aspects for large numbers of concurrent users are also critical design points that will be discussed. Through this article you will

be exposed to many design options and alternatives to maximizing performance for your data warehouse system.

## 2. DATA SOURCES

After analyzing and choosing the best data sources, the first task is to design and build the extract, transformation, load (ETL) and data maintenance processes. Parallelism needs to be designed into these processes to reduce the time windows for ETL, maintenance and maximize end-user access. Parallel processing should be considered and weighed against each

design alternative to ensure that the best overall design is achieved.

Many corporations only focus on the data sources that can be gathered internally and not information that is available for

purchase from outside vendors. This can be a major mistake as the needed information or extra value may only exist

from an external source. Outside data should be considered but it should have analyzed to determine the efforts and processes for data standardization and possible complex data conversions.

Any data brought into the data warehouse should be as clean, consistent and carefree as possible. Data transformations

processes can be very expensive by adding a large number of additional l/O's to the overall processing. This additional l/O's can be expensive for large amounts of daily or weekly ETL processes. These ETL cleansing or transformation

processes also need to be consistent in business rules and context across multiple business rules and context across multiple diverse source systems. Data consistency and standard domains and ranges are vital for developing clean and usable data that can easily be grouped, rolled up, and

associated appropriately for useful business decisions.

Different types of business decision points and business intelligence analysis require different types of data to make useful and insightful decisions from the warehouse data. The input data used and its transformation into the warehouse needs to be reflective of the type of decisions and the overall project objectives set by management for the data warehouse.

Evaluating different input source systems available for your data warehouse, a comparison criterion of how much standardization and transformation each source requires needs to be analyzed. This analysis should determine the additional

number of I/O's for standardization and transformation required by each source Considering these additional amounts of processing can quickly add up especially when additional validation and dimensional key translation IIO activity is included also.

## 3. DATA STRUCTURES AND PARTITIONING

Common data warehouse designs define fact and dimension data table objects. Properly designing these tables to minimize

their size, the number of required indexes, size and the clustering method of the data is very Important for overall performance. DB2 partitioning scheme is used to minimize object size and separate out data locking Issues.

**Split Key Partitioning**

Zip Code = 55103

Key = 551

Key = 03

One of the ways to partition DB2 database tablespaces on the z/ OS environment is to split a primary key data into two or more columns shown in the above figure. Any of these partial keys can be used to achieve different partitioning clustering orders and grouping opportunities to leverage a desired data warehouse analysis point; This redefinition must be done carefully to make sure end-user access or ETL process performance does not suffer and is enhanced by the splitting the key and potentially the processing. Also by splitting the data along the separated column values. the data can be grouped to a particular machine, partition or department. This method can also be very effective for segregating the data to different physical devices that can provide an extra performance factor by separating the it's to different physical devices. Keys that are randomly generated from a formula or

algorithm can also be used for partitioning. This method can also be very effective for randomizing the data across the database structures. Using a secondary column as the data partitioning limit key the data might be able to be spread evenly across 100 segments of the database tablespace. These 100 different partitions could then be placed across several different machines or DASD volumes to help parallelism, processing performance or backup and recovery issues. Care must be taken when splitting a key into multiple columns because of Implications of end-user SQL or OLAP tools but in most cases analysis shows these negative or out weighted by the tremendous performance improvements offered by the clustering and grouping opportunities.

Another way to partition database tablespace is through the use of results keys as shown in the below picture

**Result Key Partitioning**

ID = 96808755, Divide by 40

Result = 2420218

Remainder = 35

Sometimes a key is developed from a formula or business practice that uses a derived key. This can be done for a variety of reasons such as security, new business practice or for randomizing the key values. Result keys or hashing keys are usually a secondary key, calculated and used for a particular analysis type or grouping purpose. A simple remainder formula or a complicated calculation can be embedded in a source system that can be leveraged into the design of the partitioning of the warehouse. Documenting this type of key generation is very important because a transformation process will probably need to duplicate the formula or hashing process to generate new key values. Generating keys through formulas or hashing algorithms is good for precisely distributing the data across a range of partitions or machines as desired. The distribution can be done evenly or through a processing that maximizes data density. Manipulating the formulas or hashing routines can also be used to target the data to certain locations while maintenance is done on other portions of the database. The difficulty with using formulas or hashing routines for partitioning is that the end-user can rarely use this type of key for querying the database unless properly defined and accessible through a user-defined function (UDF}. So using this type of index should be researched and justified carefully but its flexibility can be tremendous performance asset.

In some DBMS limit keys are defined in an index structure provides the mechanism for separating the data across partitions or nodes. These limit keys can be customized to reflect the data value distributions to provide an even distribution. The distribution can be controlled via e single column value or multiple column values. Composite keys are much more common than partial keys for indexing (Below Figure). Composite keys are built from many different columns to make the entire key unique or to include numerous columns in the index structure.

**Composite Key Partitioning**

Region Code = 01

Zip Code = 10015

Complete key = 0110099

CIT Sub-partition = 0110050

Sub-partition2 = 0110099

Sub-partltion3 = 0120050

Sub-partition x.= 0540099

For example, a product department will not distinctly identify an item but add SKU number, color and size and it will uniquely identify an item. Each of the columns used to form the composite index provide a partitioning option along with the options of combining columns to form additional partitioning alternatives. During dimension definition sometimes it is best to minimize the number of objects. if the dimension's keys have common domains sometimes it is convenient to combine the entities and combine their index into a composite key. Combining objects should be studied deeply to make sure relationships and integrity are maintained but it can sometimes be a great performance enhancer by eliminating the extra IO going to the extra dimension table. Partitioning with composite keys can also help spread the data appropriately when using secondary column as partitioning limit keys.

Index structures, configuration parameter and partitioning definitions can all be used to separate indexes away from data information across the DASD configuration the physical nodes or the database partitions. Based on a key or a rule the data is stored in a particular allocation of DASD. Separating the data away from its index structure makes it easier to reorganize, maintain, backup or recover in the event of a hardware failure. The data separation is also very important because of the potential to evenly distribute the IOs of any parallel processing. Distributing the lO and data allows multiple processing entry points so that no single process or device becomes saturated with I10 requests. Eliminating and minimizing lO can make or break performance objectives. A prototype design should be put together and typical queries estimated. SQL Traces on the system during the user testing can help point out what tables are most popular or which ones might become bottlenecks or system issues. Determining the optimum number of partitions or nodes depends on a variety of factors. The physical CPU configuration and the HO hardware can be a big performance factor. By matching the hardware to the design, the designer can determine how many partitions and parallel processing streams your CPU, IO and network can support.

## 4. MULTI-DIMENSIONAL CLUSTERING

Another new feature in DB2 Ver8 is new patent pending clustering technique MDC- Multi dimensional clustering. This feature does exactly as the name Implies; it clusters the data against multiple dimension keys. This unique MDC clustering is achieved by managing data row placement into a brand new page extent blocks space management scheme based on their

dimensional key values. The new space management, placement and access are facilitated through a new Version 8 Block Index object type. This new Block Index object type is created for each of the dimensions, is similar in structure to a normal index but cross references rows to a larger dimensional data block instead of an individual row. The new extent data page block sizes are chosen at Multi-Dimensional Clustering definition time and if additional space is needed consecutive block extents are defined. Since the rows are managed to a data block, the cross-referencing Block index information needed is smaller, resulting in a smaller index structure. With consecutive pages and the Block index only referencing data blocks, Block index reorganization will rarely or not is needed as often as a regular indexes referencing individual rows. Taking data placement management one- step further than partitioning, Multi-Dimensional Clustering groups the rows to the various dimensional key blocks ideally organizing the rows for data warehousing and OLAP application access. End-user SOL can reference the Multi-Dimensional Clustering Block indexes individually, and or combined with regular indexes and utilized in all the intra and inter parallelism optimizer access methods to quickly retrieve large amounts of data. Since the Multi-Dimensional Clustering blocks are defined and extended in consecutive page blocks, similar data is contained in consecutive pages making caching, pre- fetching, RID lists and accessing data that much quicker. Clustering along multi-dimensional keys also has tremendous value for regular insert, update activities also. With a regular table, the data is placed via a single lustering value and becomes un-clustered with more insert and update activity. Multi-Dimensional Clustering tables maintain their clustering continuously over time because the clustering is based on multiple clustering keys that point to large data blocks instead of individual rows. Multi-Dimensional Clustering —MDC tables are the next step in database table design evolution. With all of its advantages this patent pending unique DB2 clustering technique gives new performance advantages and flexibility to data warehouse, OLAP and even OITP applications database designs.

## 4.1 Specialized Tables

DB2's Materialized Query Tables (MQTS) formerly known as Automatic Summary Tables and summary tables that total departments or product lines also called horizontal aggregation can greatly enhance end-user access performance by minimizing the amount of data accessed. For example, by using a MOT or summary table of monthly totals, sales-to-date figures could be developed with fewer I/O than referencing all the detail sales data. Tracking query activity can sometimes point to special data requirements or queries that happen on a frequent basis. These queries may be totaling particular products or regions that could be developed and optimized through a MQT or horizontal aggregate. Analysis must be done to justify the definition of an MOT to make sure it is used enough. Like all aggregates, MQTs and horizontal aggregates if used enough can eliminate IO and conserve CPU resources. MQTs and Horizontal aggregates work from a particular dimension key that is can be easily separated from the rest of the data.

Another method for creating specialized tables is through the use of Global Temporary Tables. Care needs to be taken to include the GTTs information in all OLAP or end-user tool information so it can be evaluated and possibly utilized for end user query result sets. Sometimes GTTs can also be used to limit the data accessed and can provide extra security against departments looking at other department's data. The GTTs or horizontal aggregate security technique is very effective and also maximizes query performance by minimizing access to only the data needed for the analysis. Another method of speeding analysis is through the use of Materialized Views as aggregate data stores that specialize in a limited dimensional data. These are good for taking complicated join predicates and stabilizing the access path for end-users. Data warehouse data can also be summarized into MVs to provide standard comparisons for standard accounting periods or management reports. Aggregate functions work best when defined to existing end-user comparison points, for example a department, product code or time data. These aggregates can be used extensively for functions and formulas because of their totaled data and definite criteria for gathering the information.

Documentation and meta-data about aggregates must be well published and completely understood by all end-users and their tools. Any end-user or OLAP tools should be aggregate aware and be able to include the different appropriate aggregates in their optimization costing and end-user processing. These data aggregates can save a tremendous amount of |l'Os and CPU. Make sure the aggregates and summaries are monitored to demonstrate and justify their creation.

# 5. UNIQUE DB2 SQL OLAP FEATURES

The SQL OLAP functions performed inside DB2 provide the answers much more efficiently then manipulating the data in a program. Like join activities and other data manipulation that DB2 can do directly, the SQL OLAF' functions can greatly reduce overall IEO and CPU utilization. These OLAP functions are particularly good for getting the top number of data rows that match a criterion.

## 5.1 OLAP Rank Function Example 01

This function can be used to order and prioritize your data according to your specific criteria. RANK orders your SQL query data by your criteria and assigns successive sequential numbers to the rows returned. The SQL query ranked rows can be individual data rows or groups of data rows.

**RANK Example**

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
RANK() OVER
(ORDER BY AVG(SALARY+BONUS) DESC)
AS RANK_AVG_SAL
FROM DAVEBEULKE.EMPLOYEE
GROUP BY WORKDEPT
ORDER BY RANK_AVG_SAL
```

| WDEPT | AVGTOTSAL | RANK_AVG_SAL |
|---|---|---|
| A00 | 43666.66 | 1 |
| B01 | 42050.00 | 2 |
| E01 | 40975.00 | 3 |
| C01 | 30790.00 | 4 |
| D21 | 25636.66 | 5 |
| D11 | 25166.66 | 6 |
| E21 | 24302.50 | 7 |
| E11 | 21418.00 | 8 |

Example 1

## 5.2 DENSERANK Function Example 02

This function can also be used to order and prioritize your data according to your specific criteria. DENSE_RANK orders your data and assigns successive sequential numbers based on the Over Partition data values found. DENSE_RANK differs from RANK because common values or ties are assigned the same number.

**DENSE RANK Example**

```
SELECT WORKDEPT, EMPNO,LASTNAME, FIRSTNME, EDLEVEL,
DENSE_RANK()
OVER (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC)
AS RANK_EDLEVEL
FROM DAVEBEULKE.EMPLOYEE
ORDER BY WORKDEPT, RANK_EDLEVEL
```

| WORKDEPT | LASTNAME | FIRSTNAME | EDLEVEL | RANK_EDLEVEL |
|---|---|---|---|---|
| A00 | LUCCHESSI | VINCENZO | 19 | 1 |
| A00 | HAAS | CHRISTINE | 18 | 2 |
| A00 | O'CONNELL | SEAN | 14 | 3 |
| B01 | THOMPSON | MICHAEL | 18 | 1 |
| C01 | KWAN | SALLY | 20 | 1 |
| C01 | NICHOLLS | HEATHER | 18 | 2 |
| C01 | QUINTANA | DOLORES | 16 | 3 |
| D11 | LUTZ | JENNIFER | 18 | 1 |
| D11 | PIANKA | ELIZABETH | 17 | 2 |
| D11 | SCOUTTEN | MARILYN | 17 | 2 |
| D11 | JONES | WILLIAM | 17 | 2 |
| D11 | STERN | IRVING | 16 | 3 |
| D11 | ADAMSON | BRUCE | 16 | 3 |

Example 2

There are many more considerations that effect data ware house system performance. Nut taking advantage of the tips and techniques discussed with in the ETL processes, parallelism, partitioning and OLAP functions can greatly Improve overall performance.

## 6. REFERENCES

[1] Vijay Datla, “Software Performance Tuning”, IJARCST, vol. 4, issue 4, 2016.

[2] "DB2 UDB for OSISQO V6 Performance Topics" IBM Redbook SG24-5351

[3] "DB2 UDB for AIX V3.1 Administration Guide" IBM

[4] "DB2 UDB for AIX V8.1 SOL Reference" IBM

[5] Vijay Datla "Performance of Ecommerce Implementation", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, issue 12, 2016.

[6] "Approaches and Methodologies for Capacity Planning for Business Intelligence Applications" IBM Redbook SG24 -5689

[7] Vijay Datla “Software Performance Workload Modelling”, International Journal of Computer Applications Technology and Research (IJCATR), Volume 6-Issue 1, 2017. doi:10.7753/IJCATR0601.1003

[8] "DB2 UDB for ZIOS V8 Administration Guide“IBM

[9] "DB2 UDB for Unix, Linux and Windows V8 Administration Guide“ IBM

[10] Vijay Datla “Performance Lifecycle in Banking Domain”, International Journal of Computer Applications Technology and Research (IJCATR), Volume 6-Issue 1, 2017. doi:10.7753/IJCATR0601.1004

[11] Bob Lyle "DB2 OLAP Functions" International DB2 Users Group European Conference (2001)