

## **A Strategy for RUNSTATS in PeopleSoft environment**

RUNSTATS or Not to RUNSTATS will sure cause a pandemonium among the new and existing PeopleSoft customers alike. This article helps a DBA to focus the attention while trying to evolve a strategy. This article is a collection of items I have witnessed, designed and deployed to many of my customers. The following is a comprehensive list, gathered from a practical reality and **not from a pragmatic idealistic wisdom.**

### **Why RUNSTATS is important?**

A big 'Excuse Me' for those that think I am trying to tell you 'How to drink milk?' The importance of performing periodic RUNSTATS cannot be understated. Keeping current statistics helps DB2 to re-evaluate the correct access path resulting in the best query optimization. You can have an accurate cross validation for the assumptions pertaining to growth statistics. Perhaps you can collect statistics to decide timeline for performing maintenance. The most important reason amongst all such advantages is in the query optimization and a fair knowledge about the object's current statistics.

### **The Con Man in RUNSTATS**

A DBA's worst nightmare is the fact that the access path has the potential to change, for the better or worse. Well we will all accept it when it is always for better but problems get compounded when the access path worsens because of RUNSTATS. No one wants to get up at 2.00 AM to re-tune the GL journal allocation processing. Did I just negate myself about the advantages of RUNSTATS?

Well, the objective of this document is to find a balance and arrive at a strategy that fits everyone's needs and to clear some of the common doubts and address those concerns.

### **Biggest myth**

One of the biggest myths among the DBA community supporting PeopleSoft applications and customers is that the application is not conducive to periodic RUNSTATS. But contrary to this belief, in spite of being dynamic, most of the SQL statements can give a consistent access path. Exceptions do exist but those are just a few handful. One of the main reasons is probably due to the fact that PeopleSoft SQL statements do not tend to be very complex. Note that a SQL statement does not become complex because it involves a three or four of sub queries and it utilizes 10 tables in a join. An example of what I consider complex is given below.

### **SELECT COUNT (DISTINCT DEPTID)**

FROM

**PS\_DEPT\_TBL A,**

**PSTREELEAF C**

WHERE

**A. SETID =?**

**AND A.EFFDT = (SELECT MAX (B.EFFDT) FROM PS\_DEPT\_TBL B  
WHERE A.SETID = B.SETID  
AND A.DEPTID = B.DEPTID)**

**AND A.SETID = C.SETID**

**AND A.SETCNTRLREC = ' '**

**AND C.TREENAME =?**

**AND A.DEPTID BETWEEN C.RANGE\_FROM AND C.RANGE\_TO**

**AND C.RANGE\_FROM <> C.RANGE\_TO**

Take a look at it first. It is a fairly straightforward statement but does an I/O to 2 million pages. It is complex because it is hard to tune. No matter what you do DB2 tends to choose DEPT table as the first table. An ideal access is to start with TREELEAF as the first table, which will result in 10000 times less number of pages to probe. In order to devise a mechanism to switch the JOIN sequence, we need a very strong understanding of SQL optimization and tuning knowledge. I am not going to delve into the details about how to tune this SQL but I want everyone to understand what is meant by complex.

If a SQL follows an index access path with 100000 ROWS, it is bound to preserve the same index access path when you add 10 million ROWS to the table. On the other hand, if there were 10 unique values for a key column with 100000 ROWS and it becomes 10000 unique values after adding 10 million ROWS, the access path can change. In other words column cardinality plays a major role.

### **Evolving the Roadmap**

Performance tuning exercise is a significant part of any implementation. A crucial deliverable of this exercise is a strategy to tailor the installation requirements that would help the application achieve consistent performance.

The first step in doing so is to put together a plan for performance tuning if such a plan does not exist already. A complete implementation of financial suite involves a lot of effort from a performance tuning perspective. It is not uncommon to run into situations requiring upwards of 15 man months to complete the tuning effort. Having to tune both online and batch processes before going live requires dedicated personnel.

Special considerations are necessary for certain objects. As a thumb rule follow the strategy outlined below to arrive at a successful tailored methodology.

As a general rule favor running RUNSTATS on all the objects.

As you continue the tuning exercise, you would certainly capture situations that require specialized statistics or on very rare occasions even manual catalog updates.

### **Freakish FREQVAL**

A vast majority of the tables will rock with the normal RUNSTATS. Certain special queries may need FREQVAL / KEYCARD statistics to be collected. You have to carefully document the need and the SQL statements that require such specialized statistics. While using FREQVAL, the depth and the number of key-values will need a research and experimentation. Sans this research, you will be cluttering SYSIBM.SYSCOLDIST. In my experience I have found one or two SQL statements that benefited from FREQVAL in almost every installation I have visited. I also make it a point to save the access path after I am satisfied with the query performance. This helps me to get back the same access path without re-inventing the research wheels again. Statistics history collection is indeed a good practice and can come in handy when you desperately seek to get back the previous access path.

### **DSTATS**

My advise to customers is not to waste time in researching DSTATS. PeopleSoft applications do not normally require multi column cardinality on indexes. A few customers have tried this option and eventually saw no practical measurable improvements. DSTATS has not yielded substantial benefits in a typical PeopleSoft.

### **Dreaded Temporary tables**

Since more than 90% of the temporary tables are queried in such a way that the process will always work on all the rows, a better access path is likely to have the temporary table as the first table accessed in a multi table JOIN path evaluation. I have proved this in many places and those that understand SQL optimization would agree with me. In fact you can complete 60% of the SQL tuning for the Accounts Payable module by segregating all the temporary tables and performing a RUNSTATS with Zero Rows on those objects. Due to this advantage, plan to RUNSTATS the temporary tables with ZERO rows for the most part.

Exceptions do occur to this rule and I never said this rule would work for 100% of the temporary table objects. When there are two or three temporary tables participating in a JOIN operation, we certainly have a conflict. Which one should be the outermost table? In situations like these, the optimizer engine in DB2 is likely to be misled. In order to arrive at the best access path, we have to evaluate the query, determine the inner and outer tables and then provide DB2 with a good representation of statistics. Even here I have found that the thumb rules work for the most time. Instead of trying to capture an accurate statistics with the help of UPDATESTATS or RUNSTATS on the fly, first set the statistics to -1 (aka default statistics) to those objects that are candidates for being inner tables. If you are unsure which columns you may need to update, a drop and re-creation of the object will yield the same result. With this method, I have had a success rate of more than 97.5%. The JOIN sequence will certainly change. An example of a situation where it is unlikely to yield is during application upgrades where these temporary tables can potentially have more rows. A temporary table with million rows is not uncommon during upgrades. Those situations may benefit from the "UPDATESTATS" functionality or RUNSTATS on the fly.

As a general rule, you should refrain from manually zapping the catalog statistics with arbitrary values. Contrary to this rule, there are situations where this approach may be the quickest way to ensure proper performance even though it is the dirtiest way. Document these findings in a PDS so you can have a scheduled job that ensures proper updates before the application window begins.

#### **Just five rules**

Take a deep look at the set of five rules.

- 1.0 Plan to do periodic RUNSTATS on all objects.
- 2.0 Watch for SQLs that violate Rule 1. Design an exception list and document clearly.
- 3.0 Plan to do RUNSTATS with ZERO rows on temporary tables. (I challenge anyone who can disagree on this statement. There will be exceptions. But this will hold true for 90% of the time)
- 4.0 Isolate temporary table objects that violate Rule 3 and never run RUNSTATS on those objects. You should maintain an exclusion list in a PDS.
- 5.0 If there are batch processes that need specific statistics (zapped manually), then maintain a SQL UPDATE step to zap the catalog at the beginning of the job step and reset with the correct statistics at the end.

The set of five rules as described above will work for everyone. Depending upon the extent of proper SQL tuning conducted at the installation, you will be able to preserve the access path very easily. As described earlier, saving statistics history will help a lot. DB2 Real Time Statistics can provide accurate guidance to design an automatic schedule of RUNSTATS based on the extent of changes to the objects. There are a lot of issues associated with Real Time Statistics collection. Verify that you have all the pre-requisite APARs and PTFs and test the RTS implementation thoroughly before deciding to use in Production.

Above all of what I described above, proper documentation is crucial to maintain the access path. How many times you want to re-invent the wheel? Documenting the access path for future reference can help in many ways.

#### **Further Reading**

You may read the document describing the Updatestats functionality and its usage. The link to the document is [http://www.hewittandlarsen.com/documents/DB2\\_general/ps\\_updatestats.pdf](http://www.hewittandlarsen.com/documents/DB2_general/ps_updatestats.pdf).

If you want to discuss more on this topic, feel free to talk directly to our principal consultant by calling 866-DB2-PSADMIN for further information. He can also be reached at [venkat@hewittandlarsen.com](mailto:venkat@hewittandlarsen.com).