



Z17

New and Improved Statistics in DB2 for z/OS Version 8

Jim Ruddy

IBM DB2 Information Management
Technical Conference

Sept. 20-24, 2004

Las Vegas, NV

© IBM Corporation 2004



Disclaimer



The information contained in this presentation has not been submitted to any formal IBM review and is distributed on an "As Is" basis without any warranty either express or implied. The use of this information is a customer responsibility.

The measurement results presented here were run in a controlled laboratory environment using specific workloads. While the information here has been reviewed by IBM personnel for accuracy, there is no guarantee that the same or similar results will be obtained elsewhere. Performance results depend upon the workload and environment. Customers attempting to adapt this data to their own environments do so at their own risk.

In addition, the material in this presentation may be subject to enhancements or Programming Temporary Fixes (PTFs) subsequent to general availability of the code.

© IBM Corporation 2004



Major points covered



- **New/changed data statistics**
- **New/changed index statistics**
- **Handling part level statistics for DPSIs**
- **Distribution Statistics Enhanced**
- **HISTORY statistics without updating main statistics**
- **Flushing the dynamic statement cache**
- **What statistics should I gather?**

© IBM Corporation 2004



Notes



We are going to cover:

New and changed catalog statistics columns for data and indexes.

How part level information is handled for DPSIs.

More non-uniform statistics and on non-indexed columns - RUNSTATS is enhanced to also collect least frequent column values and to provide most and least frequent column values on non-indexed columns.

HISTORY statistics can now be stored without updating the current statistic values in the catalog.

A way to cause the dynamic statement cache to get flushed.

A peek at some experimental work to help specify RUNSTATS

© IBM Corporation 2004



New/Changed Data Statistics



- **SPACEF at the tablespace level**
 - 4096 partitions can hold a lot of data!
- **AVGROWLEN at the tablespace/partition level**
 - V7 only collected at the table level
 - Useful for estimating current cardinality of tablespace from file size without having to run RUNSTATS
 - Conversely, can calculate tablespace size allocation more accurately
- **HIGHKEY/HIGH2KEY/LOWKEY/LOW2KEY expanded**
 - From CHAR(8) to VARCHAR(2000)
 - 8 bytes not adequate for multi-byte character representations especially with Unicode
 - Optimizer has better information to estimate filter factors and determine access paths

© IBM Corporation 2004



Notes



Total space in a tablespace is stored in SYSTABLESPACE.SPACEF, a floating point necessary to accumulate the space of 4096 partitions.

AVGROWLEN is now stored at the tablespace/partition level whereas Version 7 only collected AVGROWLEN at the table level. One use of this is estimating current cardinality of tablespace from file size without having to run RUNSTATS again. Conversely, it can be used to calculate tablespace size allocation more accurately.

Because 8 bytes is NOT adequate for the optimizer to estimate filter factors when data uses multi-byte character representations (especially with Unicode data) the HIGHKEY/HIGH2KEY/LOWKEY/LOW2KEY columns are expanded from CHAR(8) to VARCHAR(2000). This will give the optimizer to better determine access paths in Version 8.

© IBM Corporation 2004



Tablespace size



- For example:
Number of records = 100000
Maximum record size = 130 bytes
Average record size = 80 bytes
Page size = 4 KB
PCTFREE = 5
FREEPAGE = 20
MAXROWS = 255

© IBM Corporation 2004



Notes



Lets look at an example of how AVGWLEN can lead to a more accurate estimate of the space you need to allow for a tablespace.

Taking the example from the DB2 for z/OS Version 8 Administration Guide, we have specifications for a moderate size tablespace.

© IBM Corporation 2004



Tablespace size



- Using the maximum row size, you get the following results:
Usable page size = $4074 \times 0.95 = 3870$ bytes
Records per page = $\text{MIN}(\text{MAXROWS}, \text{FLOOR}(3870 / 130)) = 29$
Pages used = $2 + \text{CEILING}(100000 / 29) = 3451$
Total pages = $\text{FLOOR}(3451 \times 21 / 20) = 3624$
Estimated number of kilobytes = $3624 \times 4 = 14496$ KB
- Or using the AVGWLEN
Usable page size = $4074 \times 0.95 = 3870$ bytes
Records per page = $\text{MIN}(\text{MAXROWS}, \text{FLOOR}(3870 / 80)) = 48$
Pages used = $2 + \text{CEILING}(100000 / 48) = 2085$
Total pages = $\text{FLOOR}(2085 \times 21 / 20) = 2189$
Estimated number of kilobytes = $2189 \times 4 = 8756$ KB

You save 5740 KB

© IBM Corporation 2004



Notes



Without knowing the average row length in a table, a cautious DBA might use the worst case where all the rows are at the maximum length. From this example we see this calculation results in an estimate of 14496 KB.

However, using the AVGWLEN value collected by RUNSTATS, we see our calculation results in a much different estimate of 8756 KB.

Using the first calculation 5740 KB of space would have been wasted.

© IBM Corporation 2004



New/Changed Index Statistics



- **AVGKEYLEN at the index/partition level**
 - Needed for non-padded indexes
 - Useful for estimating current cardinality of indexspace without having to run RUNSTATS
 - Conversely, can calculate indexspace size allocation more accurately

© IBM Corporation 2004



Notes



DB2 Version 8 allows you to define indexes with varying length keys as NOT PADDED, that is, the varying length character fields are not padded to their full size. One advantage of this is allow index only queries against these indexes and return the columns with the correct length.

Version 8 has a new statistic to record the average length of these index keys - AVGKEYLEN. One use of this is estimating current cardinality of indexspace from file size without having to run RUNSTATS again. Conversely, it can be used to calculate indexspace size allocation more accurately. Stay tuned, the following example will show you another reason to use NOT PADDED indexes.

© IBM Corporation 2004



Indexspace size



- For example:
Unique Index NOT PADDED
Number of records in table = 100000
Key is a single column defined as VARCHAR(100) NOT NULL
Max Key Len = 102
AVGKEYLEN = 62
PCTFREE = 5.
FREEPAGE = 4.

© IBM Corporation 2004



Notes



Lets look at an example of how AVGKEYLEN can lead to a more accurate estimate of the space you need to allow for a indexspace.

Modifying the example from the DB2 for z/OS Version 8 Administration Guide, we have specifications for a moderate size indexspace.

© IBM Corporation 2004



Indexspace size using max



- Calculate total leaf pages using max key size
Space per key = $102 + 7 = 109$
Usable space per page = $\text{FLOOR}((100 - 5) \times 4038/100) = 3844$
Entries per page = $\text{FLOOR}(3844 / 109) = 35$
Total leaf pages = $\text{CEILING}(100000 / 35) = 2858$
- Calculate total nonleaf pages using max key size
Space per key = $102 + 7 = 109$
Usable space per page = $\text{FLOOR}(\text{MAX}(90, (100 - 5)) \times (4046/100)) = 3836$
Entries per page = $\text{FLOOR}(3836 / 109) = 35$
Minimum child pages = $\text{MAX}(2, (35 + 1)) = 36$
Level 2 pages = $\text{CEILING}(2858 / 36) = 80$
Level 3 pages = $\text{CEILING}(80 / 36) = 3$
Level 4 pages = $\text{CEILING}(3 / 36) = 1$
Total nonleaf pages = $(80 + 3 + 1) = 84$
- Calculate total space required using max key size
Free pages = $\text{FLOOR}(2585 / 4) = 646$
Tree pages = $\text{MAX}(2, (2585 + 84)) = 2669$
Space map pages = $\text{CEILING}((2669 + 646)/8131) = 1$
Total index pages = $\text{MAX}(4, (1 + 2669 + 646 + 1)) = 3317$
- **TOTAL SPACE REQUIRED, in KB = $4 \times (3317 + 2) = 13276$ KB**

© IBM Corporation 2004



Indexspace size using max



- Calculate total leaf pages using max key size
Space per key = $102 + 7 = 109$
Usable space per page = $\text{FLOOR}((100 - 5) \times 4038/100) = 3844$
Entries per page = $\text{FLOOR}(3844 / 109) = 35$
Total leaf pages = $\text{CEILING}(100000 / 35) = 2858$
- Calculate total nonleaf pages using max key size
Space per key = $102 + 7 = 109$
Usable space per page = $\text{FLOOR}(\text{MAX}(90, (100 - 5)) \times (4046/100)) = 3836$
Entries per page = $\text{FLOOR}(3836 / 109) = 35$
Minimum child pages = $\text{MAX}(2, (35 + 1)) = 36$
Level 2 pages = $\text{CEILING}(2858 / 36) = 80$
Level 3 pages = $\text{CEILING}(80 / 36) = 3$
Level 4 pages = $\text{CEILING}(3 / 36) = 1$
Total nonleaf pages = $(80 + 3 + 1) = 84$

© IBM Corporation 2004



Indexspace size using max



- Calculate total space required using max key size
Free pages = $\text{FLOOR}(2585 / 4) = 646$
Tree pages = $\text{MAX}(2, (2585 + 84)) = 2669$
Space map pages = $\text{CEILING}((2669 + 646)/8131) = 1$
Total index pages = $\text{MAX}(4, (1 + 2669 + 646 + 1)) = 3317$
- TOTAL SPACE REQUIRED, in KB = $4 \times (3317 + 2) = 13276$ KB

© IBM Corporation 2004



Notes



Without knowing the average key length in an index, a cautious DBA might use the worst case where all the keys are at the maximum length. (The only case possible in Version 7). From this example we see this calculation results in an estimate of 13276 KB.

© IBM Corporation 2004



Indexspace size using avg



- Calculate total leaf pages using AVGKEYLEN
Space per key = $102 + 7 = 69$
Usable space per page = $\text{FLOOR}((100 - 5) \times 4038/100) = 3844$
Entries per page = $\text{FLOOR}(3844 / 69) = 55$
Total leaf pages = $\text{CEILING}(100000 / 55) = 1819$
- Calculate total nonleaf pages using AVGKEYLEN
Space per key = $62 + 7 = 69$
Usable space per page = $\text{FLOOR}(\text{MAX}(90, (100 - 5)) \times (4046/100)) = 3836$
Entries per page = $\text{FLOOR}(3836 / 69) = 55$
Minimum child pages = $\text{MAX}(2, (55 + 1)) = 56$
Level 2 pages = $\text{CEILING}(2858 / 56) = 56$
Level 3 pages = $\text{CEILING}(80 / 56) = 1$
Total nonleaf pages = $(56 + 1) = 57$
- Calculate total space required using AVGKEYLEN
Free pages = $\text{FLOOR}(1819 / 4) = 454$
Tree pages = $\text{MAX}(2, (1819 + 57)) = 1876$
Space map pages = $\text{CEILING}((1819 + 57)/8131) = 1$
Total index pages = $\text{MAX}(4, (1 + 1819 + 57 + 1)) = 2332$
- **TOTAL SPACE REQUIRED, in KB = $4 \times (2332 + 2) = 9336$ KB**

3 instead of 4!
You save 3940 KB

© IBM Corporation 2004



Indexspace size using avg



- Calculate total leaf pages using AVGKEYLEN
Space per key = $102 + 7 = 69$
Usable space per page = $\text{FLOOR}((100 - 5) \times 4038/100) = 3844$
Entries per page = $\text{FLOOR}(3844 / 69) = 55$
Total leaf pages = $\text{CEILING}(100000 / 55) = 1819$
- Calculate total nonleaf pages using AVGKEYLEN
Space per key = $62 + 7 = 69$
Usable space per page = $\text{FLOOR}(\text{MAX}(90, (100 - 5)) \times (4046/100)) = 3836$
Entries per page = $\text{FLOOR}(3836 / 69) = 55$
Minimum child pages = $\text{MAX}(2, (55 + 1)) = 56$
Level 2 pages = $\text{CEILING}(2858 / 56) = 56$
Level 3 pages = $\text{CEILING}(80 / 56) = 1$
Total nonleaf pages = $(56 + 1) = 57$

© IBM Corporation 2004



Indexspace size using avg



- Calculate total space required using AVGKEYLEN
Free pages = $\text{FLOOR}(1819 / 4) = 454$
Tree pages = $\text{MAX}(2, (1819 + 57)) = 1876$
Space map pages = $\text{CEILING}((1819 + 57)/8131) = 1$
Total index pages = $\text{MAX}(4, (1 + 1819 + 57 + 1)) = 2332$

You save 3940 KB

- **TOTAL SPACE REQUIRED, in KB = $4 \times (2332 + 2) = 9336$ KB**

© IBM Corporation 2004



Notes



However, using the AVGROWLEN value collected by RUNSTATS, we see our calculation results in a much different estimate of 9336 KB.

Using the first calculation 3940 KB of space would have been wasted.

Did you notice that if the index was PADDED it would require more space than the data?

This also points out the I/O savings which can be achieved by defining an index with varying length columns as NOT PADDED; there is 1 fewer levels to probe and there are fewer total pages to scan.

© IBM Corporation 2004



Part level statistics for DPSIs



- Statistics are not kept at the partition level for logical partitions of NPIs
- Data Partitioned Secondary Indexes need to have the same partition independence and capabilities (from a statistics gathering perspective) as classic partitioning indexes.
- Partition level statistics for DPSIs are stored in SYSCOLDISTSTATS with rollup to SYSCOLDIST
- Rollup requires SYSCOLDISTSTATS rows to be sorted requiring two new parameters
 - SORTDEVT
 - SORTNUM
 - If not specified then SORT will use sort product defaults
- Can also use FORCEROLLUP to aggregate partition level statistics when not all partitions have statistics

© IBM Corporation 2004



Notes



RUNSTATS has historically only kept partition level statistics for the partitioning index - the only partitioned index allowed on a partitioned table. With Version 8 and Data Partitioned Secondary Indexes (DPSIs) and with the shift from index controlled partitioning to table controlled partitioning, it seemed obvious to keep partition level statistics for all partitioned indexes. As before, the part level starts are kept in SYSCOLDISTSTATS and the aggregated rollup of these starts are stored in SYSCOLDIST. To perform the rollup, we use DFSORT and this leads to new parameters for specification of the number of sort work datasets as well as the device type where the sort work datasets should be allocated.

Many users have partitions which are empty and normally RUNSTATS will not perform the rollup until there is statistics for all partitions. This behavior can be altered by specifying the FORCEROLLUP parameter to aggregate the statistics anyway.

© IBM Corporation 2004



Distribution Statistics Enhanced



- **As queries become**
 - more complex
 - less predictable
- **Data skew becomes more important.**
- **Problem with skewed data and regular statistics**
 - Optimizer assumes inaccurate distribution of values
 - Less efficient join sequence could be chosen
 - Less efficient method of accessing individual tables
- **DSTATS program could be downloaded to collect statistical data for non-indexed columns**
 - Great improvement in access path selection, however
 - Run separate from RUNSTATS
 - Slow with big impact to RDS workfile database

© IBM Corporation 2004



Notes



The problem of data skew was described in the IDUG Solutions Journal March 1999, Volume 6, Number article "Improving DB2 for OS/390 Query Performance with DSTATS" by Steve Bower.

© IBM Corporation 2004



Filter factors and catalog statistics



- **SYSCOLDIST** contains frequency (or distribution)
- If frequency statistics do not exist, DB2 assumes that the data is uniformly distributed
- For example:

AGE_CATEGROY	FREQUENCY
INFANT	5%
CHILD	15%
ADOLESCENT	25%
ADULT	40%
SENIOR	15%

- Default filter factor is $1/5$ ($1/\text{COLCARD}$), or 20%, to estimate the number of rows that qualify for any predicate value.
- For `AGE_CATEGORY='ADULT'` underestimate by 50%.
- For `AGE_CATEGORY='INFANT'` overestimate by 400%.

© IBM Corporation 2004



Notes



Just as a reminder of why non-uniform statistics are valuable...

This example shows how simple statistics can lead the optimizer to make inaccurate estimates based on an assumption of even distribution of data values. You also see that this goes both ways - in some cases the number of values may be underestimated and some cases over estimated. Perhaps the underestimate led to an inefficient join order (I am making this up) and perhaps the overestimated caused an index to not be used as an access path.

© IBM Corporation 2004



Distribution Statistics Enhanced



- **Non-uniform distribution statistics on non-index columns**
 - Now part of RUNSTATS
 - Significant performance improvement - no impact on RDS workfile and data only has to be scanned once
 - Uses external sort requiring two new parameters
 - SORTDEVT
 - SORTNUM
 - If not specified then SORT will use sort product defaults
- **Extend non-uniform to collect on index or non-index**
 - most frequent values
 - least frequent values
 - both
- **As part of this, the previous limit of 10 names in the COLUMN parameter has been removed.**

© IBM Corporation 2004



Notes



Skewed data distributions are responsible for a high proportion of performance problems with DB2 queries, especially in ad hoc queries. Symptoms can be less than optimal join sequences, too much synchronous I/O, and long response times. When there is asymmetrical distribution of data, not having distribution statistics on non-leading indexed columns and/or non-indexed columns can cause DB2 to make sub-optimal table join order and table join method decisions.

Collecting distribution statistics for non-leading indexed columns and/or non-indexed columns allows DB2 to use these statistics for better access path selection. Better index selections can be made, when there are screening predicates or there are matching in-list / in-subq predicates which break up matching equals predicates.

© IBM Corporation 2004



Average Jim



- New "reality" show
- Average Jim producers are looking for contestants meeting the following characteristics:
 - Gender = Female
 - Status = Single
 - Age = 39

© IBM Corporation 2004



Notes



To better illustrate non-uniform distribution statistics and review the way the DB2 optimizer computes access paths, lets look at this in the paradigm of the now ubiquitous reality show

© IBM Corporation 2004



Average Jim Data Statistics



	Pred	Dflt	Col	Calc	NUD
	=	FF	card	FF	FF
Gender	=	1/25	2	1/2	2/5
Status	=	1/25	2	1/2	2/15
Age	=	1/25	39	16/40	1/4
Est * Card		<1		9	20

*Database cardinality = 1500

© IBM Corporation 2004



Notes



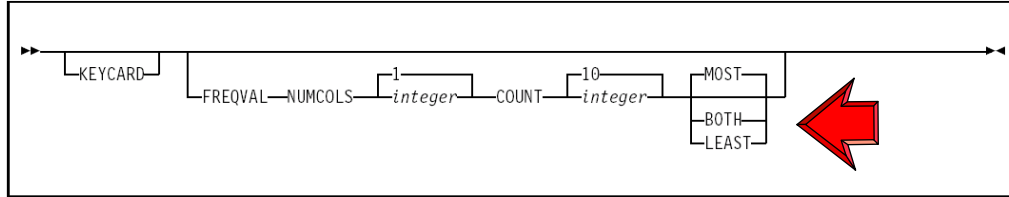
In a paper published by IBM Research in 1979 title "Access Path Selection in a Relational Database Management System" the basic algorithms for access path filter factor and join orders were described. In this paper there is a very interesting statement that for a column equal value predicate "This assumes an even distribution of tuples among the index key values". Experience has taught us that this only occurs with test data.

© IBM Corporation 2004

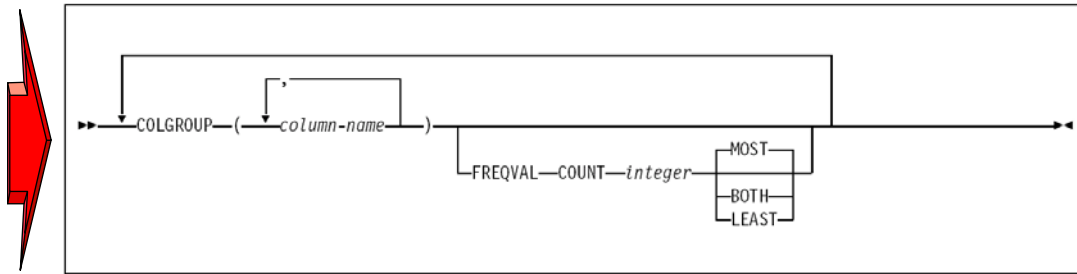


■ Changed/new syntax

correlation-stats-spec:



colgroup-spec:



COLGROUP (column-name, ...)

Indicates that the set of columns specified with the COLGROUP keyword should be treated as a group. This option allows RUNSTATS to collect a cardinality value on the column group. To collect distribution statistics for non-indexed or non-leading indexed columns, specify the COLGROUP keyword. To collect frequency statistics, specify FREQVAL with COLGROUP.

MOST

Indicates that the utility collect the most frequently occurring values for the set of specified columns when COLGROUP is specified. RUNSTATS collects the correlation statistics that occur most frequently on the column groups and stores the information in the catalog.

LEAST

Indicates that the utility collect the least frequently occurring values for the set of specified columns when COLGROUP is specified. RUNSTATS collects the correlation statistics that occur least frequently on the column groups and stores the information in the catalog.

BOTH

Indicates that the utility collect the most and the least frequently occurring values for the set of specified columns when COLGROUP is specified. RUNSTATS collects the correlation statistics that occur most frequently on the column groups and the correlation statistics that occur least frequently on the column groups and stores the information in the catalog.



Distribution Statistics Enhanced



- **Example: Collect distribution statistics for specific columns in a table space and retrieve the most and least frequently occurring values. Collect statistics for the columns EMPLEVEL, EMPGRADE, and EMPSALARY and use the FREQVAL and COUNT keywords to collect the 10 most frequently occurring values for each column and the 10 least frequently occurring values for each column.**
- **RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
TABLE(DSN8810.DEPT)
COLGROUP(EMPLEVEL,EMPGRADE,EMPSALARY)
FREQVAL COUNT 10 BOTH**

© IBM Corporation 2004



Notes



Here we see an example from the DB2 Version 8 Utility Guide and Reference specifying the new parameters to cause distribution statistics to be collect on a group of columns which may not have any indexes defined on them.

© IBM Corporation 2004



HISTORY statistics without updating main statistics



- V7 required update of main catalog statistics if history statistics were wanted
- V8 relaxes this and history statistics can now be kept without updating current statistics.
 - Monitor statistics such as SYSTABLES.CARDF
 - No surprises for dynamic SQL access paths
 - **CAUTION:** If you use this you have to be remember that your static packages bound in that time frame may not have used the statistics in the history tables.
- For example,
 - in V7 UPDATE NONE HISTORY OPTIMIZER was prohibited.
 - in V8 UPDATE NONE HISTORY OPTIMIZER is allowed and you can monitor statistics changes over time without concern that access paths may change.

© IBM Corporation 2004



Notes



Version 8 relaxes the Version 7 requirement that statistics history would only be collected if the main catalog statistics were also updated. This greater flexibility allows the user to keep track of statistics changes overtime without the concern that main statistics changes could result in access path changes, especially for dynamic SQL.

© IBM Corporation 2004



Flushing the dynamic statement cache



- **RUNSTATS with UPDATE NONE REPORT NO**
- Any statement in the Dynamic Statement Cache which is dependent on the effected table space or index space will be removed from the cache.
- **Why?** If users manually update the statistics in the catalog tables, the related dynamic SQL in the cache needs to be invalidated and the next prepare of the statements will cause the access paths to be reevaluated.

© IBM Corporation 2004



Notes



RUNSTATS with keywords REPORT NO and UPDATE NONE allows users to invalidate dynamic SQL caching for the table space and/or index space.

© IBM Corporation 2004



What statistics should I gather?



- **No simple answer**
 - **Some collect no or insufficient statistics**
 - **Prime reason for poor performing access paths**
 - **Do you want to collect statistics on every column and permutations of combination of columns?**
 - **No way!**
- **Requires similar analysis of SQL as for index design**
 - **Have to include columns which you may not benefit from adding to an index**
 - **Analysis of queries labor intensive**
 - **Iterative process analyzing explain data (as always)**

© IBM Corporation 2004



Notes



What statistics should I gather? This question is as old as DB2 itself. We have seen some users collect no statistics and others gather insufficient statistics. As the optimizer performs more analysis each release, you may see applications which had adequate access paths in a prior release now get an access path which performs poorly because of the lack of statistics. If you overreact and try to collect all possible statistics on your data, you will be unhappy with the performance of RUNSTATS.

The analysis leading to RUNSTATS recommendations is similar to index design but must take into account columns which you would not consider indexing. As we have seen with index design, analysis of a large application can be overwhelming.

The next few pages give you some insight into research in this area.

© IBM Corporation 2004



Input SQL, Click start



```
Query No: 1 SQLID: ADMFO01

SQL Text

T1.X_NEW_POSTN_ID,
T1.PR_PER_ADDR_ID,
T1.CREATED,
T14.FST_NAME
FROM
  TSIEBEL.S_CONTACT T1
  INNER JOIN TSIEBEL.S_POSTN T2 ON
T1.PR_POSTN_ID = T2.ROW_ID
  INNER JOIN TSIEBEL.S_POSTN_COM T3 ON
T1.PR_POSTN_ID = T3.POSTN_ID AND T1.ROW_ID = T3.COM_ID
  LEFT OUTER JOIN TSIEBEL.S_ORG_EXT T4 ON
T1.PR_DEPT_OU_ID = T4.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_POSTN T5 ON
T4.PR_POSTN_ID = T5.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_POSTN T6 ON
T1.PR_POSTN_ID = T6.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_EMPLOYEE T7 ON
T1.EMP_ID = T7.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_EMPLOYEE T8 ON
T5.PR_EMP_ID = T8.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_EMPLOYEE T9 ON
T6.PR_EMP_ID = T9.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_ADDR_PER T10 ON
T1.PR_PER_ADDR_ID = T10.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_POSTN T11 ON
T1.X_NEW_POSTN_ID = T11.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_ASSET T12 ON
T1.X_NEW_CUSTOMER_ID = T12.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_POSTN T13 ON
T1.PR_POSTN_ID = T13.ROW_ID
  LEFT OUTER JOIN TSIEBEL.S_EMPLOYEE T14 ON
T2.PR_EMP_ID = T14.ROW_ID
WHERE
  ((T1.BU_ID = ?) AND
  (T1.X_CATEGORY_FLG = ?)) AND
  (T1.X_POS_CHG_SIGN = ?)
```



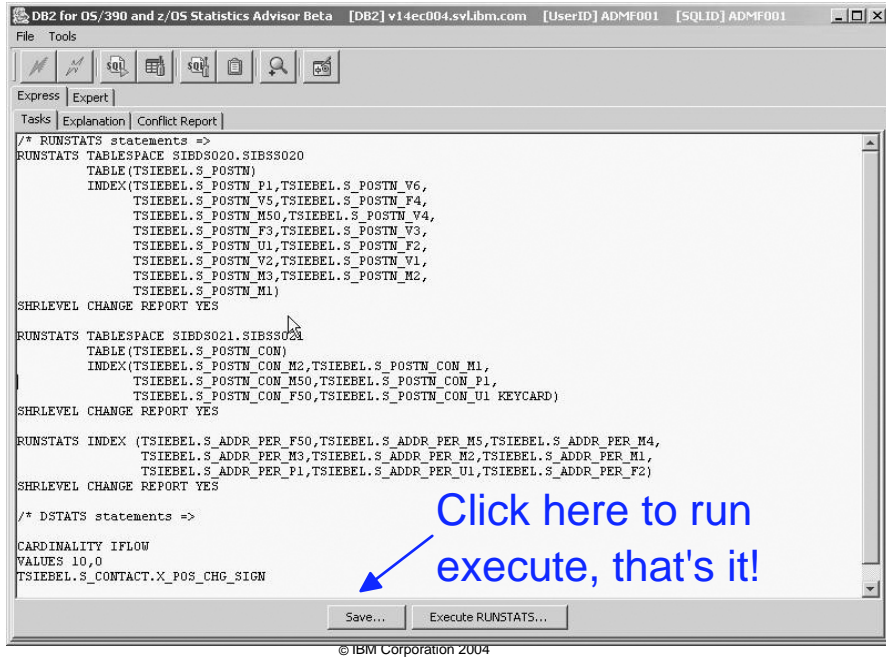
Notes



Here we see a query which would be truly challenging to determine what kind of statistics should be gathered to give the optimizer the best chance of picking the optimal access path.



Suggestions for one Siebel query



```
DB2 for OS/390 and z/OS Statistics Advisor Beta [DB2] v14ec004.svl.ibm.com [UserID] ADMF001 [SQLID] ADMF001
File Tools
Express Expert
Tasks Explanation Conflict Report
/* RUNSTATS statements =>
RUNSTATS TABLESPACE SIBDS020.SIBSS020
TABLE(TSIEBEL.S_POSTM)
INDEX(TSIEBEL.S_POSTM_P1,TSIEBEL.S_POSTM_V6,
TSIEBEL.S_POSTM_V5,TSIEBEL.S_POSTM_F4,
TSIEBEL.S_POSTM_M50,TSIEBEL.S_POSTM_V4,
TSIEBEL.S_POSTM_F3,TSIEBEL.S_POSTM_V3,
TSIEBEL.S_POSTM_U1,TSIEBEL.S_POSTM_F2,
TSIEBEL.S_POSTM_V2,TSIEBEL.S_POSTM_V1,
TSIEBEL.S_POSTM_M3,TSIEBEL.S_POSTM_M2,
TSIEBEL.S_POSTM_M1)
SHRLEVEL CHANGE REPORT YES

RUNSTATS TABLESPACE SIBDS021.SIBSS021
TABLE(TSIEBEL.S_POSTM_COM)
INDEX(TSIEBEL.S_POSTM_COM_M2,TSIEBEL.S_POSTM_COM_M1,
TSIEBEL.S_POSTM_COM_M50,TSIEBEL.S_POSTM_COM_P1,
TSIEBEL.S_POSTM_COM_F50,TSIEBEL.S_POSTM_COM_U1 KEYCARD)
SHRLEVEL CHANGE REPORT YES

RUNSTATS INDEX (TSIEBEL.S_ADDR_PER_F50,TSIEBEL.S_ADDR_PER_M5,TSIEBEL.S_ADDR_PER_M4,
TSIEBEL.S_ADDR_PER_M3,TSIEBEL.S_ADDR_PER_M2,TSIEBEL.S_ADDR_PER_M1,
TSIEBEL.S_ADDR_PER_P1,TSIEBEL.S_ADDR_PER_U1,TSIEBEL.S_ADDR_PER_F2)
SHRLEVEL CHANGE REPORT YES

/* DSTATS statements =>
CARDINALITY IFLOW
VALUES 10,0
TSIEBEL.S_CONTACT.X_POS_CHG_SIGN
```

© IBM Corporation 2004



Notes

Here we see some sample output from the Statistics Advisor for that previous query. Only one RUNSTATS is specifying KEYCARD and although this example shows the use of the DSTATS program, we can see that the Statistics Advisor is recommending non-uniform statistics be gathered one column of one table.



Statistics Advisor Current Status



- **Not quite ready for prime time yet**
- **Used as a serviceability tool**
 - Service team use prototype on real problems
 - Demonstrates research of automation of query analysis
- **Identifying, addressing areas of improvement**
 - Move forward from prototype status

- **For more information see**
 - Z32 Don't miss the overhauled DB2 for z/OS Visual Explain V8 by Patrick D Bossman
 - Z34 DB2 for z/OS Exploiting the V7 & V8 Optimization Enhancements by Terry Purcell

© IBM Corporation 2004



Notes



The Statistics Advisor is still pretty much a prototype but it is being used by the optimizer service team on real customer problems. It is a good example of the ongoing research into automating query analysis. So far it has been very useful in identifying areas which need to be improved.

© IBM Corporation 2004



References



- DB2 UDB for z/OS home page
<http://www.software.ibm.com/data/db2/os390/>
- utilities@work
http://www.ibm.com/software/data/db2imstools/details/html/us_text.html
- The IDUG Solutions Journal March 1999 - Volume 6, Number 1
Improving DB2 for OS/390 Query Performance with DSTATS By Steve Bower
http://www.idug.org/neo_apps/cfmfiles/mainnavbar.cfm?body=/journal/index.html
- DB2 UDB for z/OS and OS/390 Version 7 Performance Topics, SG24-6129
- DB2 UDB for z/OS and OS/390 Version 7: Using the Utilities Suite, SG24-6289
- DB2 UDB for z/OS Version 8 What's New
<http://www-3.ibm.com/software/data/db2/os390/v8/dsnwnj1.pdf>
- DB2 UDB for z/OS Version 8 Administration Guide
- DB2 UDB for z/OS Version 8 Utilities Guide and Reference

© IBM Corporation 2004



Notes



To learn more about IBM DB2 RUNSTATS, check out the information in these publications

© IBM Corporation 2004