SHARE
Technology · Connections · Results

IBM DB2 RUNSTATS Utility and Real-Time Statistics

Bryan F. Smith
IBM

Tuesday, August 14, 2007
Session 1316

Abstract

- This presentation reviews the basics of the RUNSTATS utility (What it does; Why you need to run it; How DB2 uses the information), and explores new statistics collected on data and indexes, including: partition level information on Data Partitioned Secondary Indexes; non-uniform distributon statistics on non-indexed columns; and historical statistics. The real-time statistics are also reviewed. Upon completion of this session, the attendee, whose skill level may range from low to high, will be able to understand how to get the most out of DB2's statistics and operate at optimal efficiency.

Topics

- Why RUNSTATS?
- Invoking RUNSTATS
- Commonly asked questions (about the stats)
- Real-time Statistics
- Rebinding considerations
- Reorg recommendations
- When is RUNSTATS needed?
- New/changed data statistics
- New/changed index statistics
- Handling part level statistics for DPSIs
- Distribution Statistics Enhanced
- HISTORY statistics changes
- Flushing the dynamic statement cache
- What statistics should I gather?

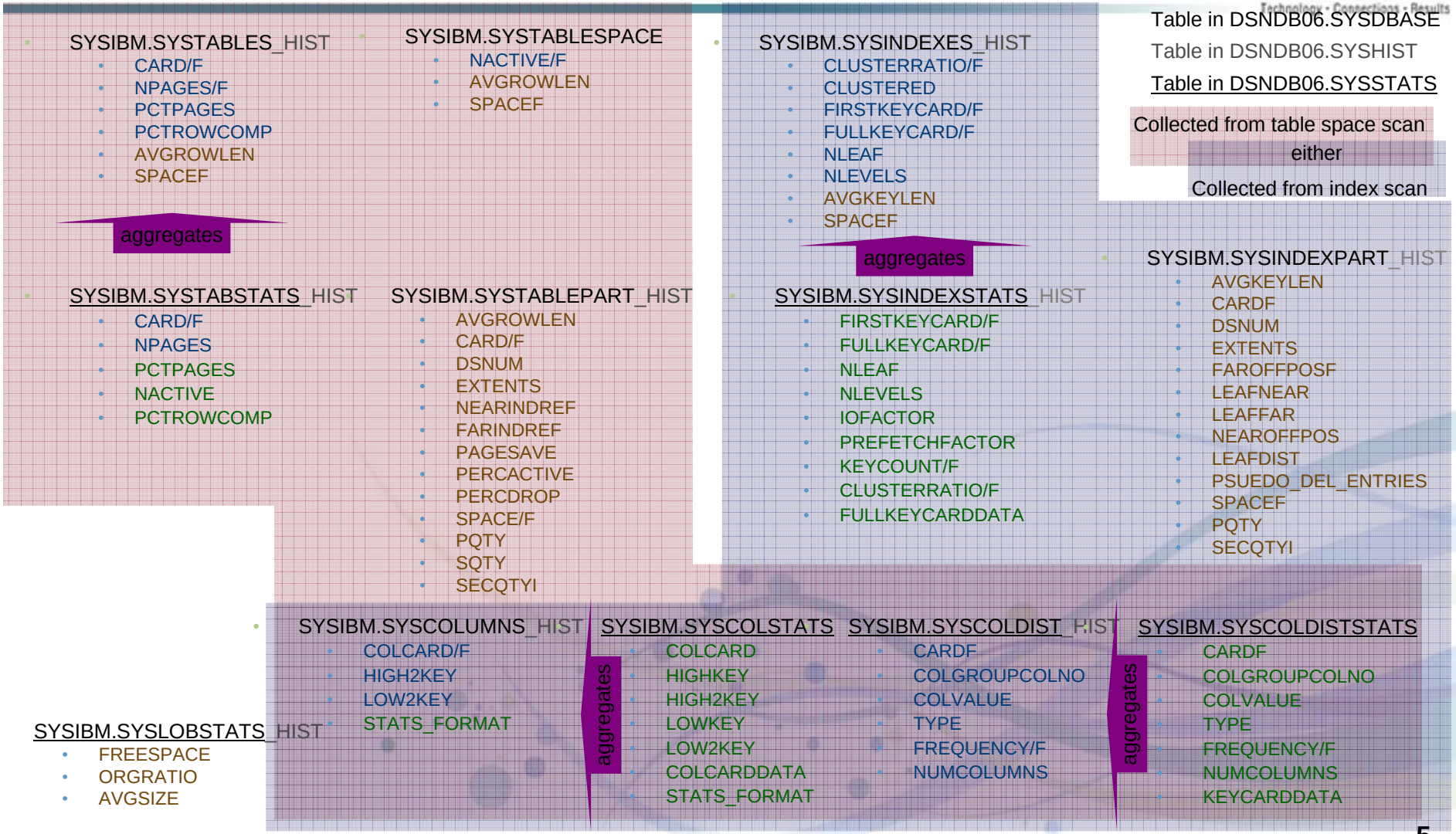
Why RUNSTATS?

- The RUNSTATS utility computes statistics on a specified table space or index and updates the DB2 catalog
- Two types of statistics
  - Access path statistics

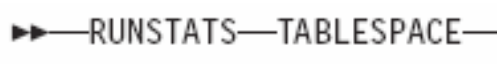
    Those used by BIND/PREPARE in its process of optimization to determine access path (some can also be used to help determine when to reorg)
  - Space

    Those used by the DBA to monitor space usage; to assist in capacity planning; to help determine when to reorg; etc.

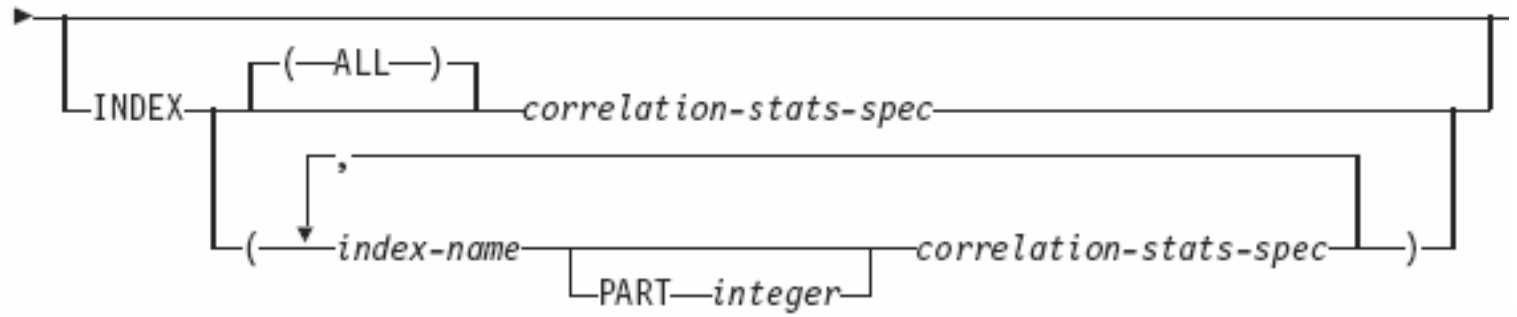
Statistics gathered by RUNSTATS



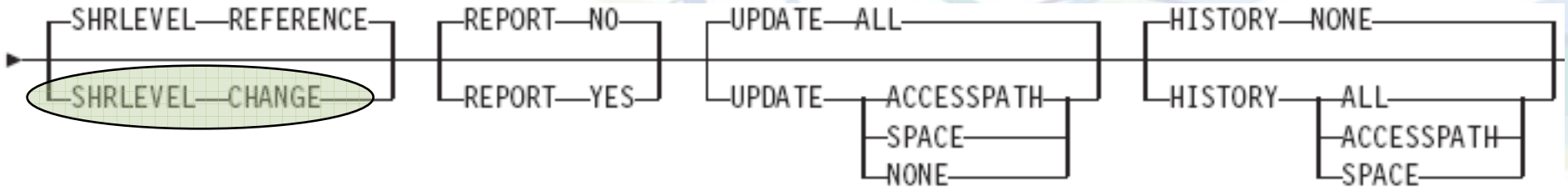
Invoking RUNSTATS



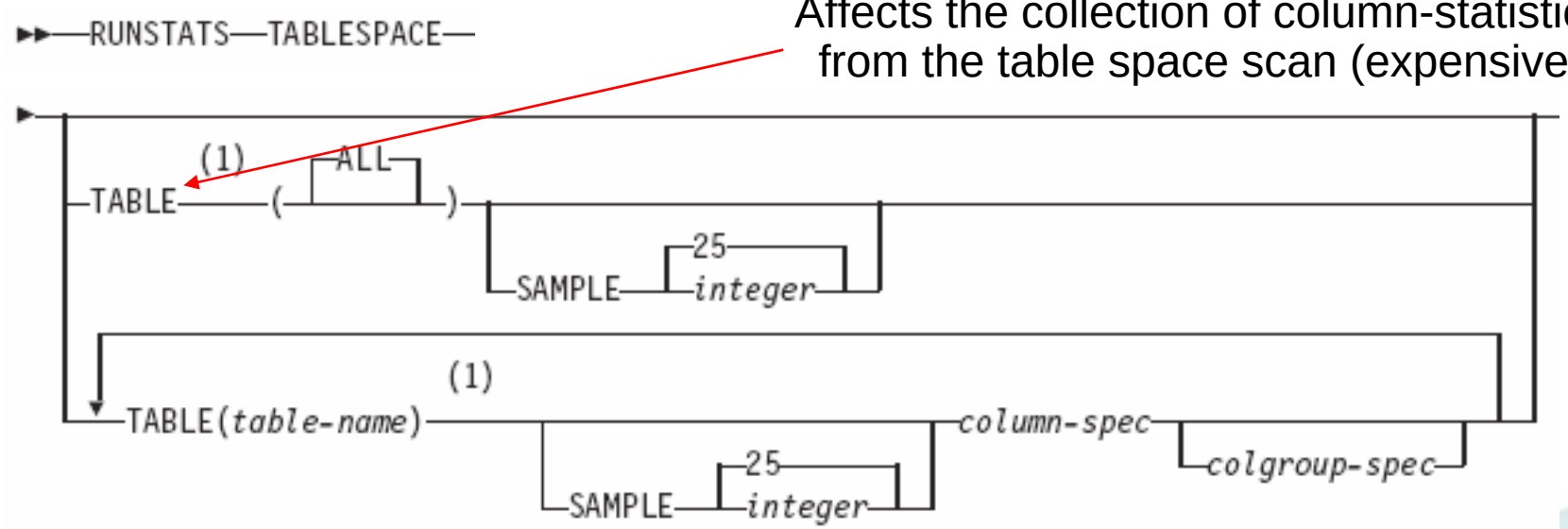
Scans the tablespace



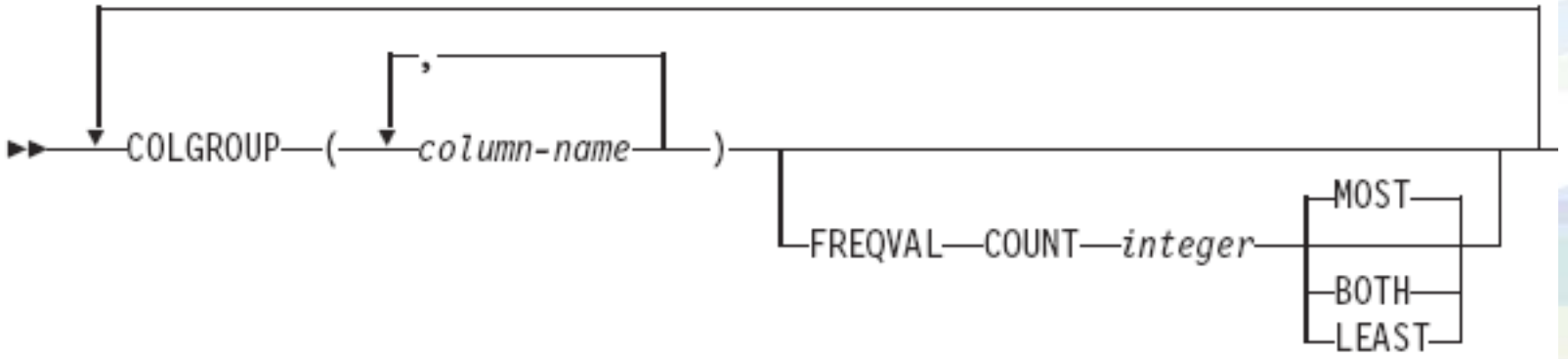
Scans the index



Invoking RUNSTATS



colgroup-spec



KEYCARD (Recommended)

- Collects all of the distinct values in all of the 1 to *n* key column combinations for the specified indexes. *n* is the number of columns in the index. For example, suppose that you have an index defined on three columns: A, B, and C. If you specify KEYCARD, RUNSTATS collects cardinality statistics for column A, column set A and B, and column set A, B, and C.
- So these are cardinality statisics across column sets... if we had a 3-column index that had these values:

Col1	Col2	Col3
A	B	C
A	B	D
A	B	E
A	B	E
A	C	A
A	C	A
A	D	A
B	B	B

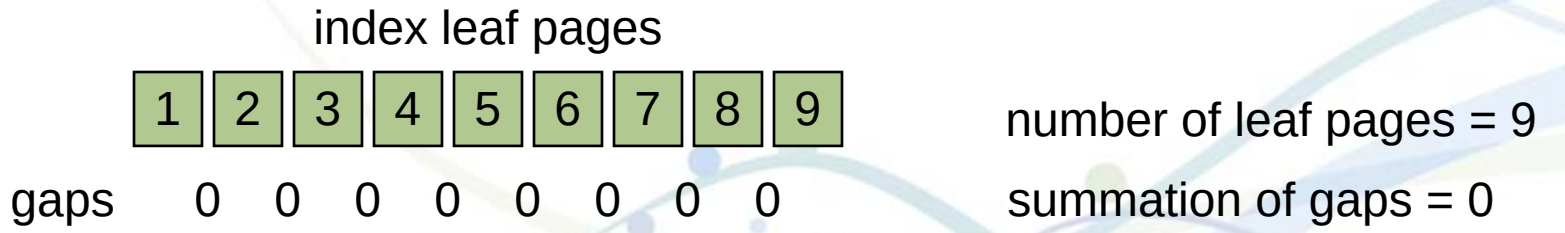
then these stats would be collected:

- Col1 cardinality = 2
- Col1 and Col2 cardinality = 4
- Col 1, Col2, and Col3 cardinality = 6

Commonly asked questions about the stats

- What is SYSIBM.SYSINDEXPART.LEAFDIST?
  - LEAFDIST is 100 times the average number of pages between successive leaf pages of the index

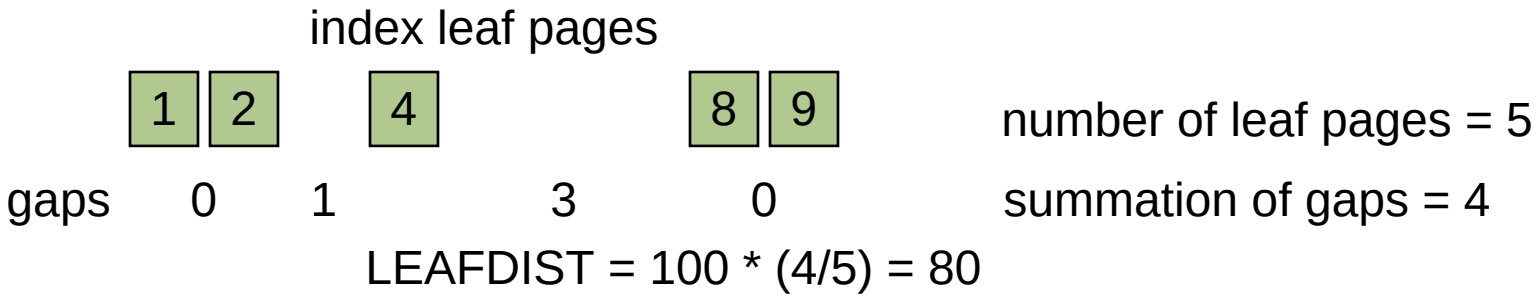
$$\text{LEAFDIST} = 100 \times \frac{\text{summation of distance between pages}}{\text{Number of leaf pages}}$$



LEAFDIST = 100 * (0/9) = 0 (%)

Commonly asked questions about the stats

- Another example of LEAFDIST

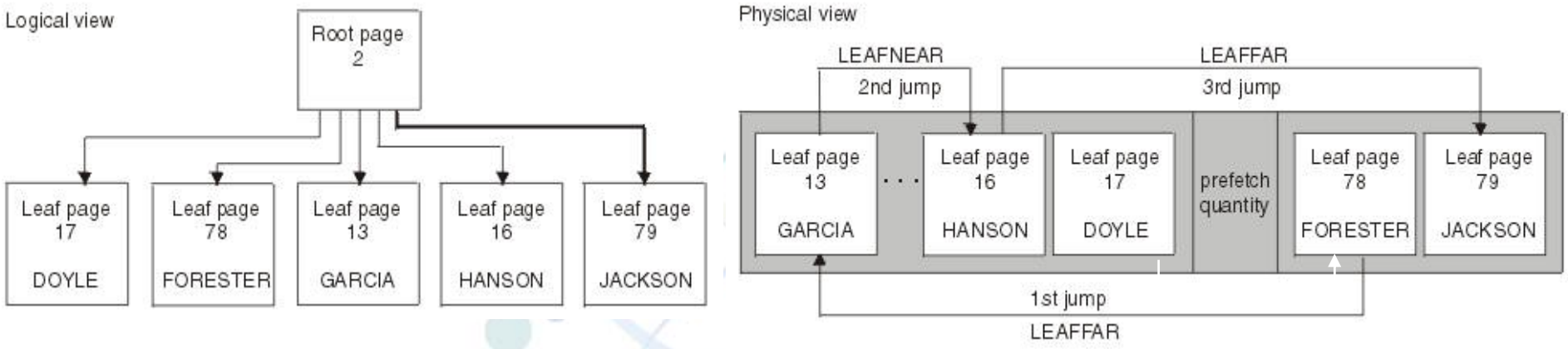


- If there were more gaps than active pages, LEAFDIST would be larger
- FREEPAGE on an index can certainly affect the calculation of LEAFDIST
- We used to use this value to determine when to reorg an index, but now we have better stats to determine this (LEAFFAR/NEAR)

Commonly asked questions about the stats

- What is SYSIBM.SYSINDEXPART.LEAFNEAR and LEAFFAR?
  - LEAFNEAR/FAR measure the disorganization of physical leaf pages
    - Number of pages that are not in an optimal position due to
      - *index pages being deleted or*
      - *index leaf page splits caused by an insert that cannot fit onto a full page*

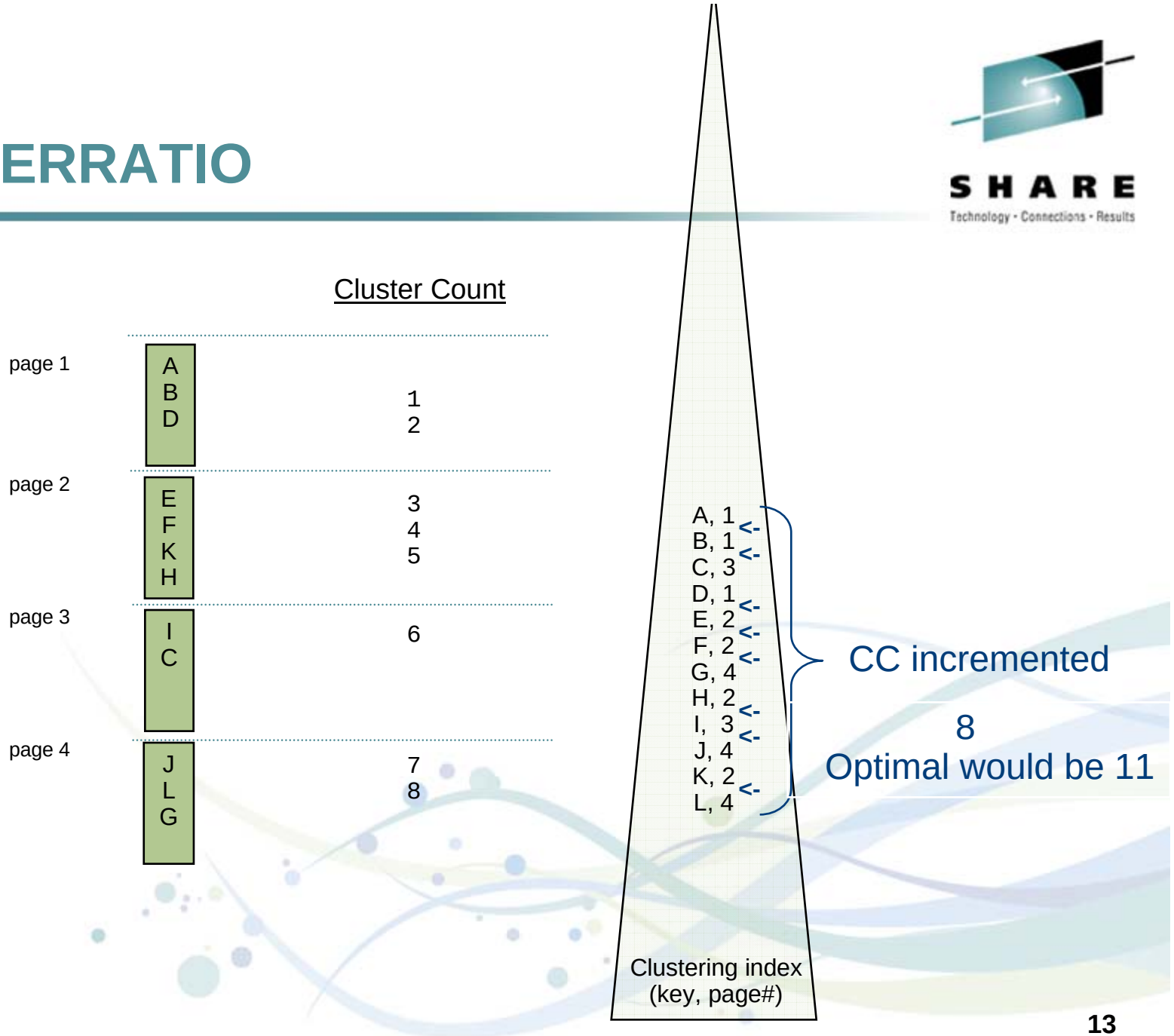
Logical and physical views of an index in which LEAFNEAR=1 and LEAFFAR=3



Commonly asked questions about the stats

- SYSIBM.SYSINDEXES.CLUSTERRATIO
  - An access path statistic that can also helps in determining when to reorg
  - % of the rows that are in cluster order
  - Rows are counted as being “clustered” if they are in a greater or equal page number of the previous row
  - This is a statistic that describes the data in the table(space), even though it is reported in SYSINDEXES – REORG INDEX will never affect this statistic

CLUSTERRATIO

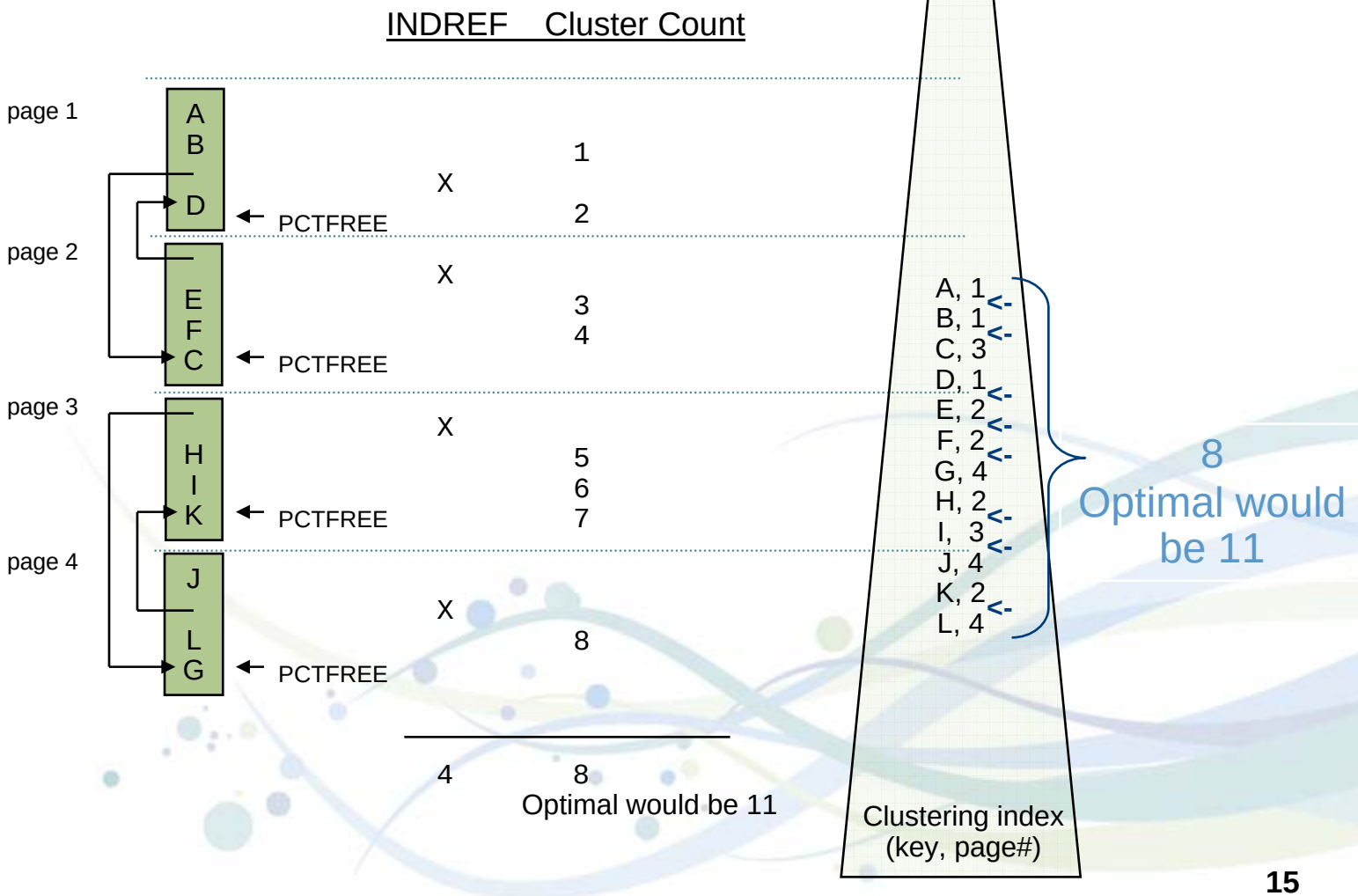


Commonly asked questions about the stats

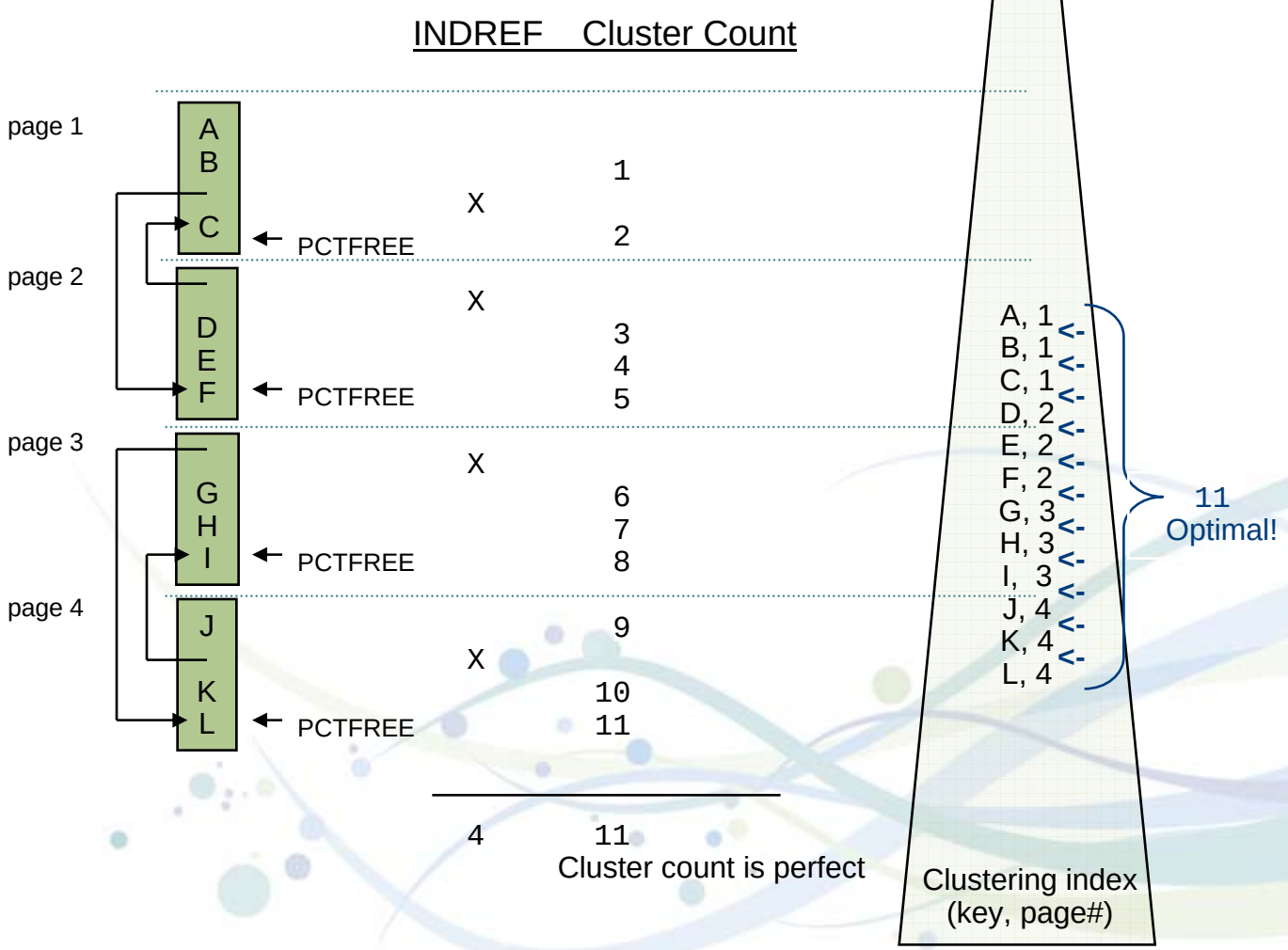
- How does NEAR|FAR INDREF and NEAR|FAR OFFPOS contribute to CLUSTERRATIO?
- *INDREF correlates closely with the cluster count if the keys are in cluster order and then rows are relocated to another page, but we can create cases where these stats are correlated and cases where they are not correlated
- *OFFPOS directly affects the cluster count. A single “jump” counts as two OFFPOS’, so almost always, the cluster count is ½ of the sum of the *OFFPOS’.

- SYSIBM.SYSTABLEPART_HIST
  - NEARINDREF
  - FARINDREF
- SYSIBM.SYSINDEXES_HIST
  - CLUSTERRATIO/F
- SYSIBM.SYSINDEXPART_HIST
  - FAROFFPOSF
  - NEAROFFPOS

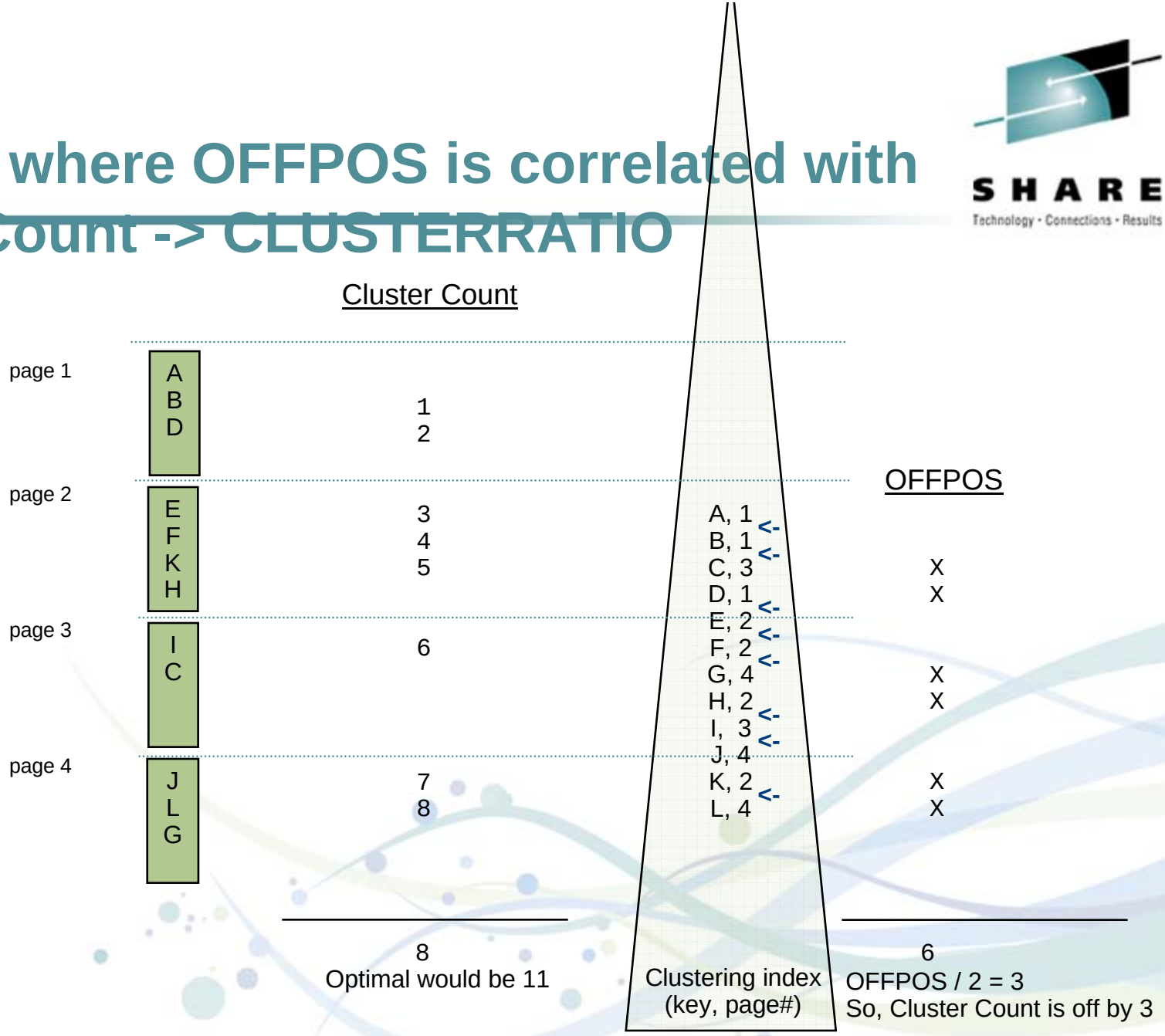
Example where INDREF is correlated with Cluster Count -> CLUSTERRATIO



Example where INDREF is not correlated with Cluster Count -> CLUSTERRATIO



Example where OFFPOS is correlated with Cluster Count -> CLUSTERRATIO



Exercise for the reader…

We just saw an example where *OFFPOS is correlated to the cluster count (which is used to compute CLUSTERATIO). Can an example be created showing non-correlation between these two metrics?

Commonly asked questions

- Can you collect stats and have them stored in the catalog without affecting any binds/prepares?
  - Yes (by specifying REPORT YES UPDATE NONE or UPDATE NONE HISTORY ALL)
- Should you collect statistics on the DB2 Catalog?
  - Yes. Will it benefit DB2 processing like BIND or PREPARE?
    - No, …but SQL against the catalog can benefit
- Is there any difference between running

  RUNSTATS TABLESPACE DB1.TS1 INDEX (ALL)

  vs.

  RUNSTATS TABLESPACE DB1.TS1

  RUNSTATS INDEX(ALL) TABLESPACE DB1.TS1 -- ??
  - No, they are semantically equivalent, but you could run these two utility statements in parallel to reduce overall elapsed time
- Can/should you update the statistics in the DB2 Catalog?
  - It depends
- What is the semantic difference between RUNSTATS TABLESPACE and RUNSTATS TABLESPACE TABLE (ALL)?
  - The TABLE keyword triggers collection of column statistics

Extra credit

- Is there any difference between running

  RUNSTATS TABLESPACE DB1.TS1 TABLE (ALL) INDEX (ALL)

  vs.

  RUNSTATS TABLESPACE DB1.TS1 TABLE (ALL)

  RUNSTATS INDEX(ALL) TABLESPACE DB1.TS1 -- ??

There is a difference – what is it?

Real-time Statistics

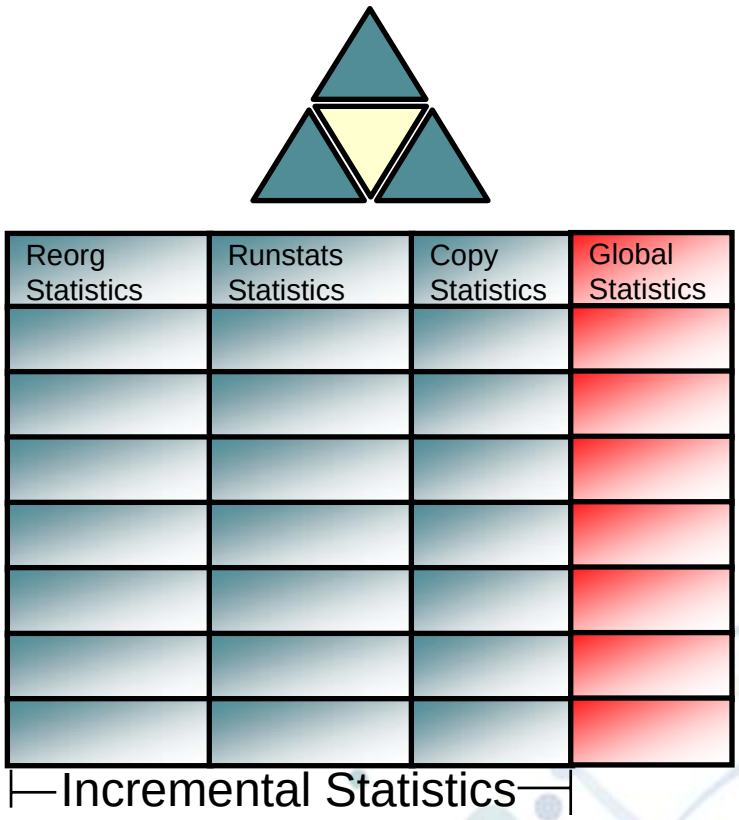
- Introduced in V7
- Contain “space” and some “accesspath” statistics in user-defined tables:
  - SYSIBM.TABLESPACESTATS (one row per partition)
  - SYSIBM.INDEXSPACESTATS (one row per partition)
  - In DB2 9, these are moved into the DB2 Catalog (DSNDB06.SYSRTSTS) as
    - SYSIBM.SYSTABLESPACESTATS
    - SYSIBM.SYSINDEXSPACESTATS
- Intended to eliminate running RUNSTATS for reasons of running utilities by exception
- Access path selection doesn’t use RTS in V7, V8 or V9

Real-time statistics tables in DSNRTSDB.DSNRTSTS

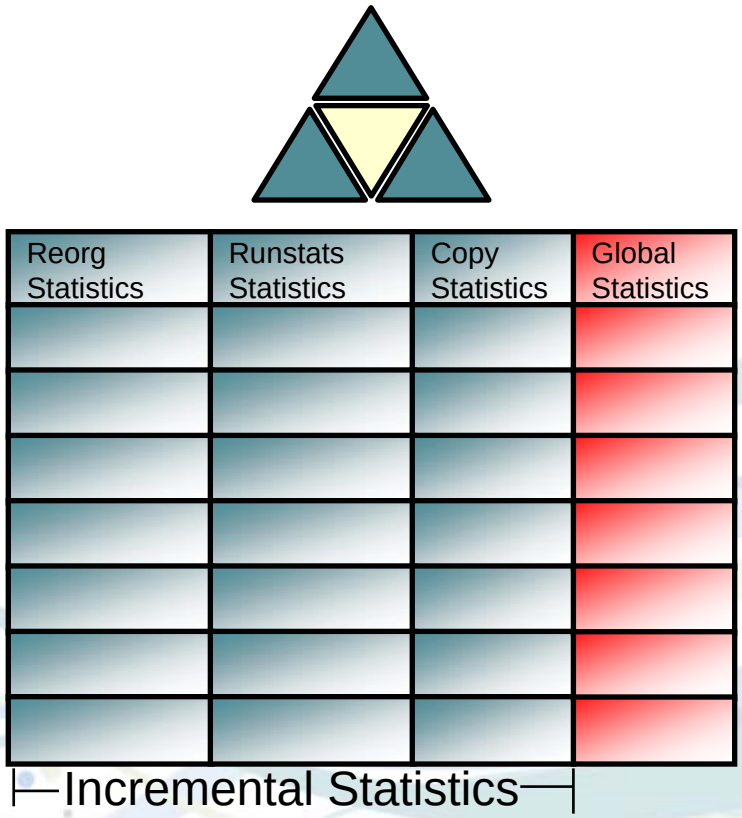
DSNDB06.SYSRTSTS

Index SYSIBM.DSNRTX01 ← New in V9 → **Index SYSIBM.DSNRTX02**

(dbid, psid, partition.instance) ← New in V9 → **(dbid, isobid, partition.instance)**



SYSIBM.SYSTABLESPACESTATS



SYSIBM.SYSINDEXSPACESTATS

RTS

- SYSTTABLESPACESTATS
  - Global
    - NACTIVE
    - NPAGES
    - EXTENTS
    - SPACE
    - TOTALROWS
    - DATASIZE
    - UNCOMPRESSEDDATASIZE
    - UPDATESTATSTIME
  - Incremental
    - REORG Statistics
      - LASTTIME
      - INSERTS
      - UPDATES
      - DELETES
      - DISORGLOB
      - UNCLUSTINS
      - MASSDELETE
      - NEARINDREF
      - FARINDREF
    - COPY Statistics
      - LASTTIME
      - UPDATEDPAGES
      - CHANGES
      - UPDATELRSN
      - UPDATETIME
    - RUNSTATS Statistics
      - LASTTIME
      - INSERTS
      - UPDATES
      - DELETES
      - MASSDELETE
- SYSINDEXSPACESTATS
  - Global
    - NACTIVE
    - NLEVELS
    - NPAGES
    - NLEAF
    - EXTENTS
    - SPACE
    - TOTALENTRIES
    - LASTUSED
    - UPDATESTATSTIME
  - Incremental Statistics
    - REORG Statistics
      - REBUILDLASTTIME
      - LASTTIME
      - INSERTS
      - UPDATES
      - DELETES
      - APPENDINSERT
      - PSEUDODELETES
      - MASSDELETE
      - LEAFNEAR
      - LEAFFAR
      - NUMLEVELS
    - COPY Statistics
      - LASTTIME
      - UPDATEDPAGES
      - CHANGES
      - UPDATELRSN
      - UPDATETIME
    - RUNSTATS Statistics
      - LASTTIME
      - INSERTS
      - DELETES
      - MASSDELETE

Enable/Disable Real Time Statistics in V7/V8

- START DATABASE (DSNRTSDB)
  - Validate table space, table and index definitions
  - Enable real time statistics collection
  - Issue this command to enable RTS after the statistics tables and indexes are first created
  - Data may not be accurate until a new REORG/RUNSTATS/COPY is done
- START DB2
  - Implicitly to enable real time statistics if
    - DSNRTSDB is not STOPPED and
    - DB2 Catalog is accessible
- STOP DATABASE(DSNRTSDB)
  - Flush all in-memory statistics
- In V9, RTS are a part of the catalog and are always enabled

Collect Real Time Statistics in Memory

- Allocate RTS blocks
  - At first update for table spaces since the pageset/partition is opened
  - At open time for indexes since we collect SYSINDEXSPACESTATS.LASTUSED
  - In DBM1 Address Space (~140 bytes per pageset/partition – moved above bar in V9)
  - 0/32KB per pageset/partition – above the bar in V9)
- Free RTS blocks when
  - Pagesets/Partitions are closed
  - After statistics are written to RTS tables
- In a data sharing system, statistics are collected by each member
- In-memory statistics are always collected even if RTS is not enabled

When to externalize in-memory statistics?

- On a timer interval
  - STATSINT in ZPARM - default 30 minutes
  - REAL TIME STATS in DSNTIPO install panel
    - Range: 1 to 1,440 minutes
- STOP/START DATABASE SPACENAM command
  - Flush in-memory statistics for all target objects
- STOP/START DATABASE(DSNRTSDB) in V7/8
  - Flush all in-memory statistics
- STOP DB2 MODE(QUIESCE)
- A utility operation (e.g. LOAD, REORG, RUNSTATS, COPY, REBUILD, RECOVER)

Process to externalize in-memory statistics

- RTS manager externalizes in-memory statistics to the RTS Tables
- RTS manager runs under a system task in DBM1 address space
  - CPU time is included in DBM1's SRB time
  - The system task is created during START DB2
- RTS manager is triggered on a timer interval
  - Default is 30 minutes
  - Scan in-memory statistics blocks
    - Free dormant statistics blocks that belong to closing data sets
  - Order active statistics blocks in clustering order
  - Insert/update rows in the RTS tables via the clustering index
- Each data sharing member externalizes its own statistics

When to collect statistics for DB2 Objects?

- Newly created table spaces and indexes
  - Rows are inserted into RTS tables at CREATE
    - Loadrlasttime and Reorglasttime is set to CREATE timestamp
    - Stats/Copylasttime are set to NULL
    - Totalrows/Totalentries are set to zero, all other global counters are set to null or a known value, incremental counters are set to zero
- Table spaces and Indexes existed before RTS is enabled
  - Rows are inserted when the objects are first updated
    - At the next STATSINT timer interval
    - All statistics values are set to NULL (except for Nactive, Space, Extent)
    - Reorg/Stats/Copy/Loadr-lasttime are set to NULL
    - Statistics values will be set after the first REORG, RUNSTATS, or COPY
  - No RTS rows for read only table space accessed objects (LASTUSED will be updated for read only indexes)

How SQL affects table space statistics?

- CREATE/DROP TABLESPACE
  - Insert/delete a row in SYSIBM.TABLESPACESTATS
- Insert
  - Increment Inserts,Totalrows, Copy Changes counters
  - May update Nactive, Space, Extents, Uncluster_Inserts, Distinct Updated Pages, Update LRSN, Update Timestamp, Datasize
- Update
  - Increment Updates, Copy Changes counters
  - May update NearIndRef/FarIndRef, Nactive, Space, Extents for VARCHAR Tables, Distinct Updated Pages, Update LRSN, Update Timestamp, Datasize
- Delete
  - Increment Deletes, Copy Changes counters, Datasize

How SQL affects table space statistics? ...

- Delete without the WHERE clause or DROP TABLE for Segmented Table Spaces
  - Increments the Mass Deletes/Drops counter
- Rollback
  - Insert
    - Increment Deletes counter
  - Delete
    - Increment Inserts counter
  - Update
    - Increment Update counter
  - Mass Delete/Drop Table
    - Will not decrement the Mass Deletes/Drops counter
- Statistics counters will not be updated during DB2 Restart
- Triggers may cause statistics updated for other tables

How SQL affects index space statistics?

- CREATE/DROP INDEX
  - Insert/delete a row in/from SYSIBM.INDEXSPACESTATS
- Insert
  - Increment Inserts,TotalEntries counters
  - May update Append_Inserts, LeafNear, LeafFar, ReorgNumLevels, Nactive, Space, Extents, Nleaf
- Delete
  - Increment Deletes counter
  - May update Pseudo Deletes, ReorgNumLevels
- COPY YES indexes (Insert/Delete)
  - Maintain Copy Changes, Distinct Updated Pages, Update LRSN, Update Timestamp
- Delete without a WHERE clause or DROP TABLE
  - Increment Mass Deletes counter
- Rollbacks/Restart - same as for table space statistics

How Utility affects real-time statistics?

- REORG
  - Set Last_REORG_Timestamp
  - Reset REORG related statistics
  - Log apply changes for online REORG will be treated as Inserts/Deletes/Updates
- RUNSTATS
  - Set Last_RUNSTATS_Timestamp
  - Reset RUNSTATS related statistics
- COPY
  - Set Last_COPY_Timestamp
  - Reset COPY related statistics
- LOAD REPLACE
  - Set Last_Load_Replace Timestamp
  - Reset REORG related statistics

How Utility affects real-time statistics? ...

- REORG/LOAD REPLACE PART
  - Will not reset REORG statistics for non-partitioned indexes
  - Statistics for NPIs will be updated as INSERT and DELETE
- COPY with the DSNUM option
  - Will not reset Last_Copy_Timestamp
  - Will not reset COPY related statistics
  - We maintain statistics if DSNUM <> 0 refers to partitioned object
  - If DSNUM references a data set, statistics are NOT maintained for the data set
- RECOVER TORBA/TOCOPY
  - Set Last_REORG, Last_RUNSTATS, Last_COPY, Last_Load_Replace, Last_Rebuild_Index to NULL
  - Reset REORG, RUNSTATS, COPY statistics to NULL
- REBUILD INDEX
  - Set Last_Rebuild_Index_Timestamp
  - Reset REORG related statistics
- Online LOAD Resume
  - Treated as Inserts

Accuracy of the statistics

- Always delayed by the timer interval
  - Controlled by ZPARM STATSINT (default 30 minutes)
- Loss all in-memory statistics when DB2 is crashed or STOP DB2 MODE(FORCE)
- Unable to externalize statistics when DSNRTSDB is stopped or statistics tables are unavailable
- Need to run REORG, RUNSTATS, COPY to establish a reference point
- Statistics could be inaccurate if running vendor utilities without flushing the in-memory statistics
- Only physical space statistics (i.e. Nactives, Space, Extents) are maintained for DSNDB07 and the TEMP databases

Guideline for SQL/Utility to access RTS objects

- Avoid Timeouts or Deadlocks with RTS manager
  - Use Uncommitted Read lock isolation when accessing RTS tables
  - Use SHRLEVEL CHANGE when running REORG, RUNSTATS, COPY on the RTS objects
- Don't mix RTS objects with other user objects in a utility list operation
  - If mixed, RTS statistics will not be reset for all objects in the list
- For Disaster Recovery
  - Recover RTS objects after DB2 catalog and directory objects are recovered
  - Explicitly issue START DATABASE(DSNRTSDB) after RTS objects are recovered

What is DSNACCOR?

- A DB2 stored procedure that accesses the RTS tables
- And makes IFI calls
  - to gain -DISPLAY status on DB2 objects
- Primary purpose -
  - To recommend any DB2 object that requires a:
    - REORG
    - RUNSTATS
    - IMAGE COPY
- New version of DSNACCOR in DB2 9 is named DSNACCOX

Historical RTS

- There is no historical capability in RTS
- This can easily be built manually
  - Create SYSIBM.TABLE/INDEXSPSTATS_HIST LIKE SYSIBM.SYSTABLE/INDEXSPACESTATS and add CAPTURE_TIME AS TIMESTAMP NOT NULL WITH DEFAULT cols
  - Periodically (daily?) insert into history tables with a subselect from the RTS tables those rows that aren't already in the history tables; and delete old information. Code this up in a stored proc?

Rebinding considerations

- Consider the following guidelines regarding when to rebind
  - CLUSTERRATIOF changes to less or more than 80% (a value of 0.80)
  - NLEAF changes more than 20% from the previous value
  - NLEVELS changes
  - NPAGES changes more than 20% from the previous value
  - NACTIVEF changes more than 20% from the previous value
  - The range of HIGH2KEY to LOW2KEY range changes more than 20% from the range previously recorded
  - Cardinality changes more than 20% from previous range
  - Distribution statistics change the majority of the frequent column values

Reorg recommendations

- These are generic and do not apply in all cases – there is no absolutely reliable statistic as to when reorganization of table spaces or indexes should occur; however, understanding the rules of thumb will help in understanding data disorganization
- If reorg for performance, then track performance over time
- DSNACCOR (V7/8) /DSNACCOX (V9) usage

Reorg table space (incl. LOBs in V9) recommendations

- Consider running REORG TABLESPACE in the following situations:
  - Real-time statistics (TABLESPACESTATS)
    - REORGUNCLUSTINS (number of records inserted since the last Reorg that are not well-clustered)/TOTALROWS > 10%
      - *Irrelevant if predominantly random access*
      - *REORGUNCLUSTINS is only an indication of the insert behavior and is correlated to the cluster ratio only if there are no updates or deletes. To prevent DSNACCOR/X from triggering on these, identify such objects and put them in exception list*
    - (REORGNEARINDREF+REORGFARINDREF (number of overflow rows since the last Reorg))/TOTALROWS > 5% in data sharing, >10% in non-data sharing
    - REORGINSERTS (number of records inserted since the last Reorg)/TOTALROWS > 25%
    - REORGDELETES (number of records deleted since the last Reorg)/TOTALROWS > 25%
    - EXTENTS (number of extents) > 254
    - REORGDISORGLOB (number of LOBs inserted since the last Reorg that are not perfectly chunked)/TOTALROWS > 50%
    - SPACE > 2 * (DATASIZE / 1024) (when free space is more than used space)
    - REORGMASSDELETE > 0 (mass deletes on seg tsp and DROP on multi-table tsps)
  - RUNSTATS
    - PERCDROP > 10%
    - SYSIBM.SYSLOBSTATS.ORGRATIO < 50% (changed to a value 0-100 in PQ96460 on V7/V8)
    - (NEARINDREF + FARINDREF) / CARDF > 10% non-data-sharing, > 5% if data sharing
    - FAROFFPOSF / CARDF > 10%
      - *Or, if index is a clustering index, CLUSTERRATIOF < 90% (irrelevant if predominantly random access)*
  - Other
    - Tsp is in adv reorg pending status (AREO*) as result of an ALTER TABLE stmnt
    - Index on the tsp is in adv REBUILD pend state (ARBDP) as result an ALTER stmnt

Reorg table space (incl. LOBs in V9) recommendations

- Consider running REORG TABLESPACE in the following situations:
  - Real-time statistics (TABLESPACESTATS)
    - REORGUNCLUSTINS (number of records inserted since the last Reorg that are not well-clustered)/TOTALROWS > 10%
      - *Irrelevant if predominantly random access*
      - *REORGUNCLUSTINS is only an indication of the insert behavior and is correlated to the cluster ratio only if there are no updates or deletes. To prevent DSNACCOR/X from triggering on these, identify such objects and put them in exception list*
    - (REORGNEARINDREF+REORGFARINDREF (number of overflow rows since the last Reorg))/TOTALROWS > 5% in data sharing, >10% in non-data sharing
    - REORGINSERTS (number of records inserted since the last Reorg)/TOTALROWS > 25%
    - REORGDELETES (number of records deleted since the last Reorg)/TOTALROWS > 25%
    - EXTENTS (number of extents) > 254
    - REORGDISORGLOB (number of LOBs inserted since the last Reorg that are not perfectly chunked)/TOTALROWS > 50%
    - SPACE > 2 * (DATASIZE / 1024) (when free space is more than used space)
    - REORGMASSDELETE > 0 (mass deletes on seg tsp and DROP on multi-table tsps)
  - RUNSTATS
    - PERCDROP > 10%
    - SYSIBM.SYSLOBSTATS.ORGRATIO < 50% (changed to a value 0-100 in PQ96460 on V7/V8)
    - (NEARINDREF + FARINDREF) / CARDF > 10% non-data-sharing, > 5% if data sharing
    - FAROFFPOSF / CARDF > 10%
      - *Or, if index is a clustering index, CLUSTERRATIOF < 90% (irrelevant if predominantly random access)*
  - Other
    - Tsp is in adv reorg pending status (AREO*) as result of an ALTER TABLE stmnt
    - Index on the tsp is in adv REBUILD pend state (ARBDP) as result an ALTER stmnt

Don't use RUNSTATS statistics as a trigger to consider running REORG

Reorg table space (incl. LOBs in V9) recommendations

- Consider running REORG TABLESPACE in the following situations:
  - Real-time statistics (TABLESPACESTATS)
    - REORGUNCLUSTINS (number of records inserted since the last Reorg that are not well-clustered)/TOTALROWS > 10%
      - *Irrelevant if predominantly random access*
      - *REORGUNCLUSTINS is only an indication of the insert behavior and is correlated to the cluster ratio only if there are no updates or deletes. To prevent DSNACCOR/X from triggering on these, identify such objects and put them in exception list*
    - (REORGNEARINDREF+REORGFARINDREF (number of overflow rows since the last Reorg))/TOTALROWS > 5% in data sharing, >10% in non-data sharing
    - REORGINSERTS (# of records inserted since the last Reorg)/TOTALROWS > 25%
    - REORGDELETES (# of records deleted since the last Reorg)/TOTALROWS > 25%
    - EXTENTS (number of extents) > 254
    - REORGDISORGLOB (number of LOBs inserted since the last Reorg that are not perfectly chunked)/TOTALROWS > 50%
    - SPACE > 2 * (DATASIZE / 1024) (when free space is more than used space)
    - REORGMASSDELETE > 0 (mass deletes on seg tsp and DROP on multi-table tsps)
  - Other
    - Tsp is in adv reorg pending status (AREO*) as result of an ALTER TABLE stmnt
    - Index on the tsp is in adv REBUILD pend state (ARBDP) as result an ALTER stmnt

Reorganizing LOBs in V7 and V8

- Generally not recommended
  - Only possible with SHRLEVEL NONE
  - Small performance gain that can be achieved is outweighed by
    - Loss of availability
    - Likelihood of increasing the size of the LOB table space
- With DB2 9's REORG support of LOBs with SHRLEVEL REFERENCE
  - Chunkiness (REORGDISORGLOB/TOTALROWS > *50%*
  - Space reclamation SPACE > *2* * (DATASIZE / 1024)

Reorg index recommendations

- Consider running REORG INDEX in the following cases:
  - Real-time statistics (INDEXSPACESTATS)
    - REORGPSEUDODELETES (number of index entries pseudo-deleted since the last Reorg)/TOTALENTRIES > 10% in non-data sharing, 5% if data sharing as pseudo-deleted entry can cause S-lock/unlock in Insert for unique index
    - REORGLEAFFAR (number of index leaf page splits since the last Reorg and the new leaf page far from the original leaf page)/NACTIVE > 10%
    - REORGINSERTS (number of index entries inserted since the last Reorg)/TOTALENTRIES > 25%
    - REORGDELETES (number of index entries inserted since the last Reorg)/TOTALENTRIES > 25%
    - REORGAPPENDINSERT / TOTALENTRIES > 20%
    - EXTENTS (number of extents) > 254
  - RUNSTATS
    - LEAFFAR / NLEAF > 10% (NLEAF is a column in SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXPART)
    - PSEUDO_DEL_ENTRIES / CARDF > 10% for non-data sharing and > 5% for data sharing
  - Other
    - The index is in advisory REORG-pending status (AREO*) or advisory-REBUILD-pending status (ARBDP) as the result of an ALTER statement

When is RUNSTATS needed?

- When the data changes sufficiently to warrant new statistics
  - REORG of tablespace or index (use inline stats!)
  - LOAD REPLACE of tablespace (use inline stats!)
  - After "significant" application changes for the tablespace or index
    - Periodically (weekly, monthly) except for read only data?
    - Application tracks updates with activity tables?
    - After percentage of pages changed since last RUNSTATS (RTS)?
- Understand implications for access paths!
- SHRLEVEL
  - REFERENCE drains writers
  - CHANGE runs like application with ISOLATION (UR)

    (claim reader for allocation duration)

New/Changed Data Statistics (V8)

- SPACEF at the table space level
  - 4096 partitions can hold a lot of data!
- 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
- AVGROWLEN at the table space/partition level
  - V7 only collected at the table level
  - Useful for estimating current number of rows of table space from file size without having to run RUNSTATS
  - Conversely, can calculate table space size allocation more accurately
  - UNLOAD utility space allocation
  - REORG & LOAD space allocation
    - work datasets
    - sort space

- SYSIBM.SYSTABLESPACE
  - AVGROWLEN
  - SPACEF
- SYSIBM.SYSTABLEPART_HIST
  - AVGROWLEN

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 new parameters
  - SORTDEVT (defaults to SYSALLDA)
  - 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

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 DB2 work file database

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:

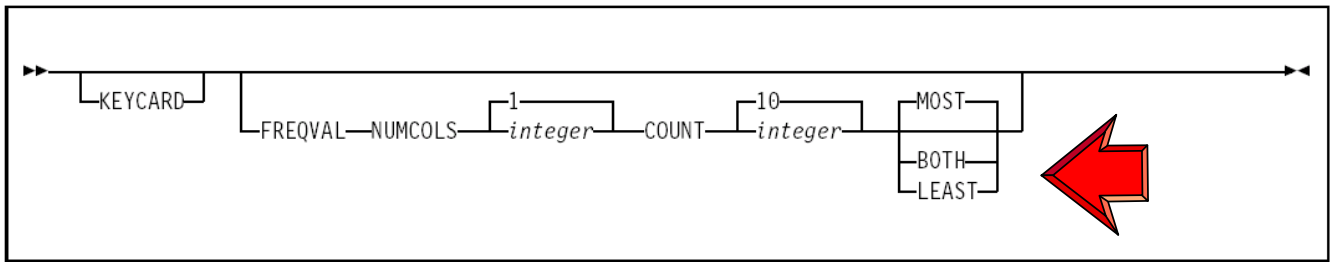
AGE_CATEGORY	FREQUENCY
INFANT	5%
CHILD	15%
ADOLESCENT	25%
ADULT	40%
SENIOR	15%

Distribution Statistics Enhanced

- Non-uniform distribution statistics on non-indexed columns
  - Now part of RUNSTATS
  - Significant performance improvement - no impact on DB2 work file and data only has to be scanned once
  - Uses external sort requiring 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.

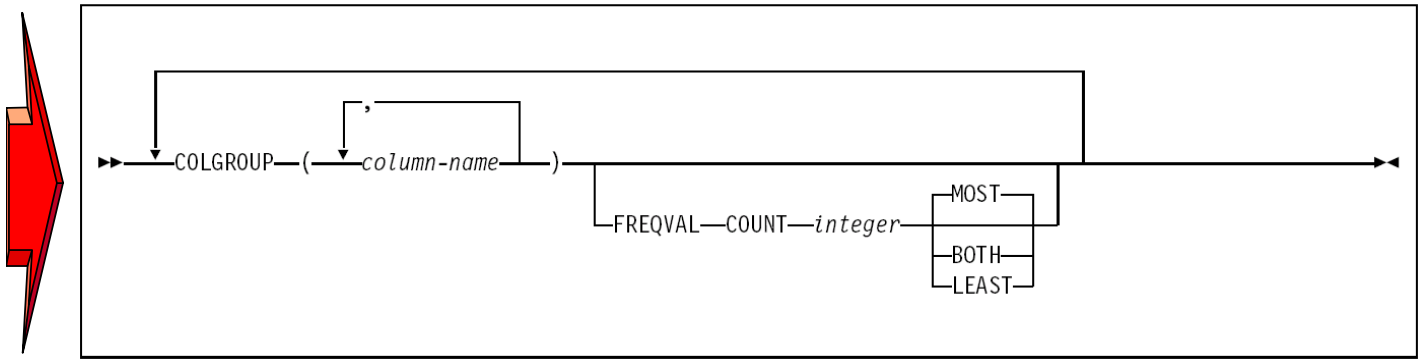
Distribution Statistics Enhanced

- Changed/new syntax



RUNSTATS INDEX
REBUILD, REORG INDEX

colgroup-spec:



RUNSTATS TABLESPACE

KEYCARD versus Distribution Statistics from an index

State	City	Zipcode
CA	San Jose	95123
CA	San Jose	95110
CA	San Jose	95141
CA	San Jose	95141
CA	Riverside	92504
CA	Riverside	92504
CA	Glendora	91741
TX	Austin	78732

- KEYCARD collects all of the distinct values in all of the 1 to *n* key column combinations
- So these are cardinality statisics across column sets... if we had a 3-column index on State, City, Zipcode:

Numcolumns	Card
1	2
2	4
3	6

- FREQVAL NUMCOLS 3 collects

Colvalue	Frequency
CA, San Jose, 95123	1/8 = 0.125
CA, San Jose, 95110	1/8 = 0.125
CA, San Jose, 95141	2/8 = 0.25
CA, Riverside, 92504	2/8 = 0.25
CA, Glendora, 91741	1/8 = 0.125
TX, Austin, 78732	1/8 = 0.125

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

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

Not *currently* collected via in-line statistics from LOAD and REORG

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.

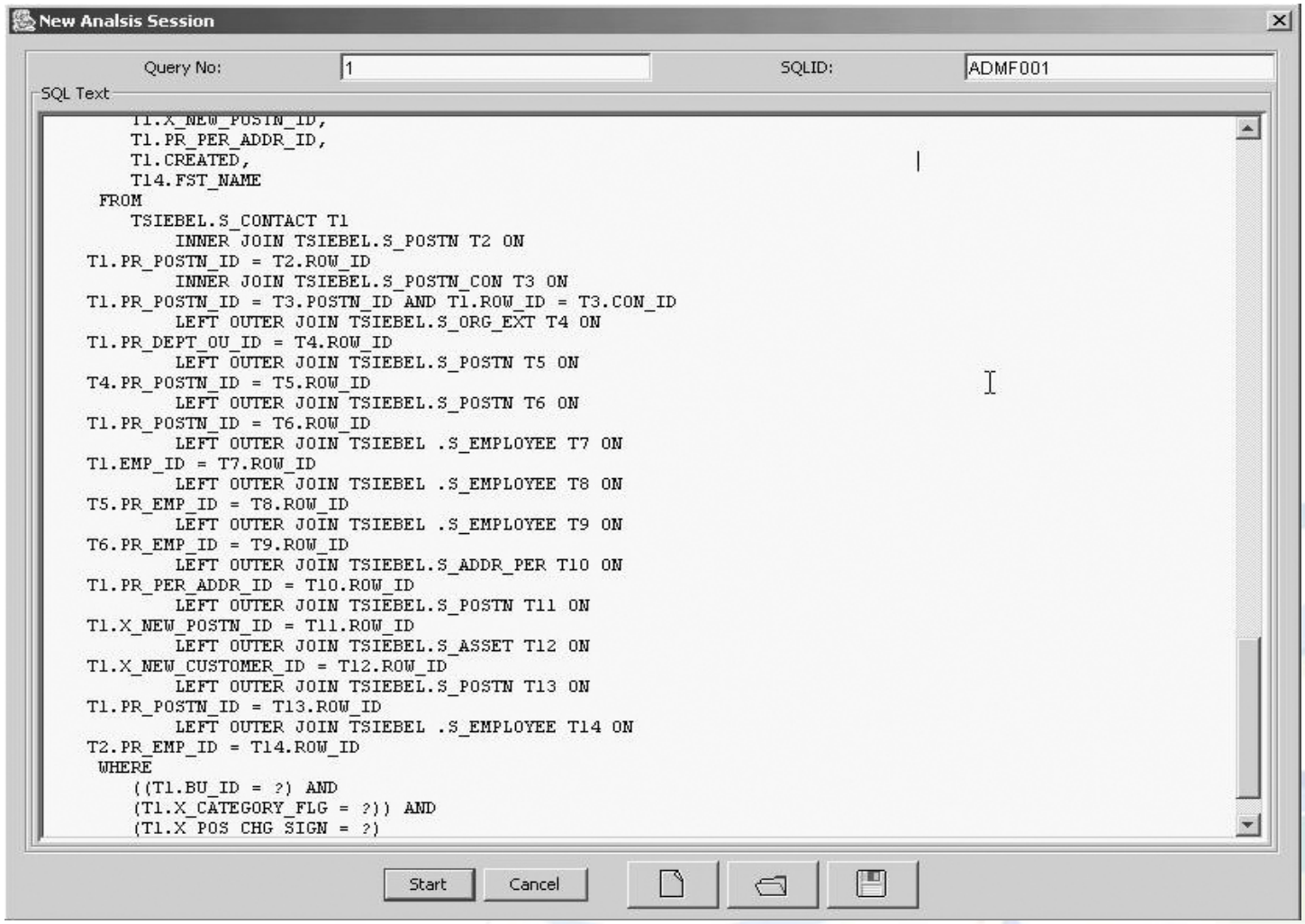
Flushing the dynamic statement cache

- RUNSTATS with UPDATE NONE REPORT NO
- Any statement in the Dynamic Statement Cache which is dependent on the affected 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.
- Granularity is at the table space/index level (not the table level)

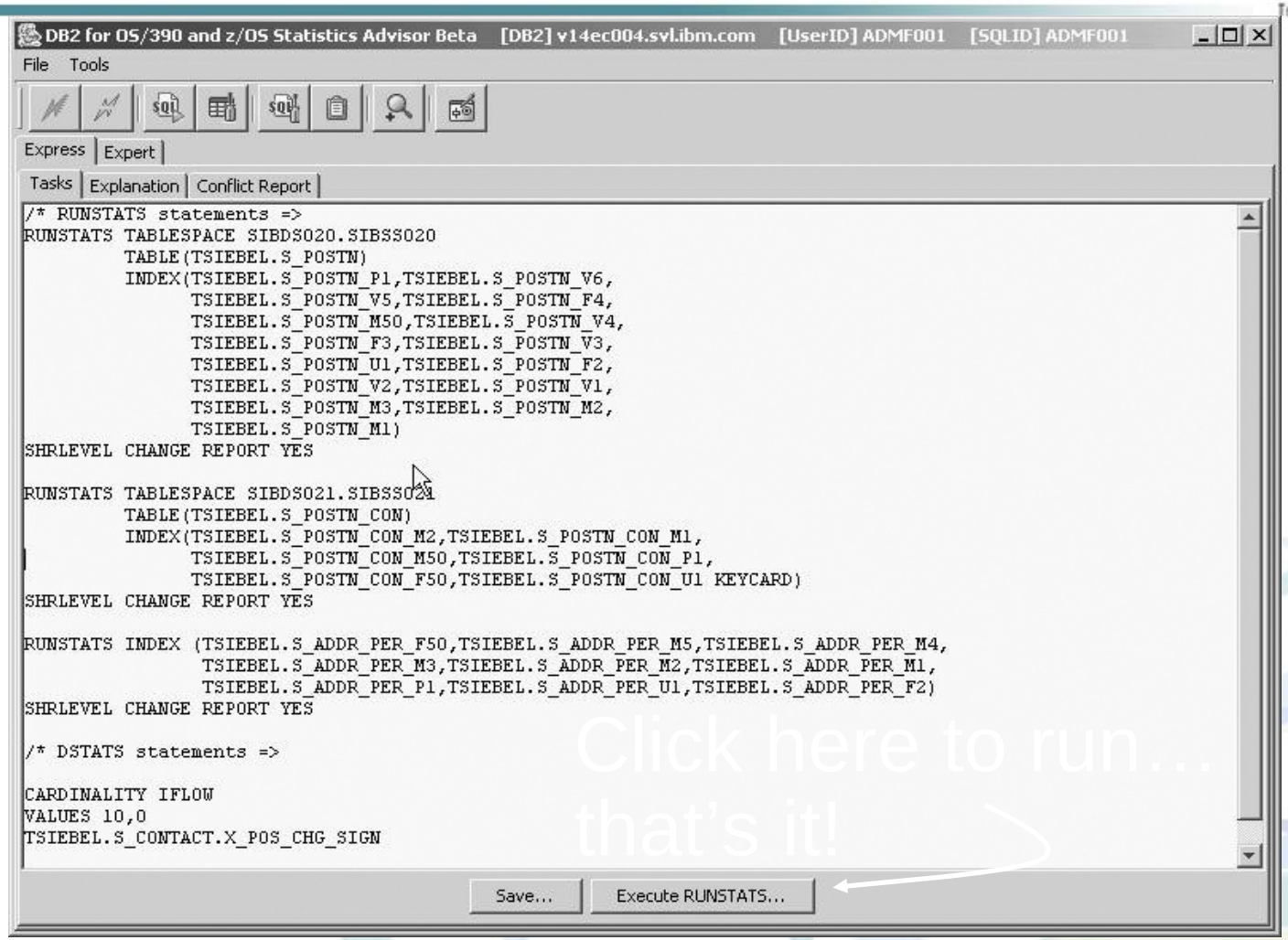
What statistics should I gather?

- 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)

Input SQL, Click start

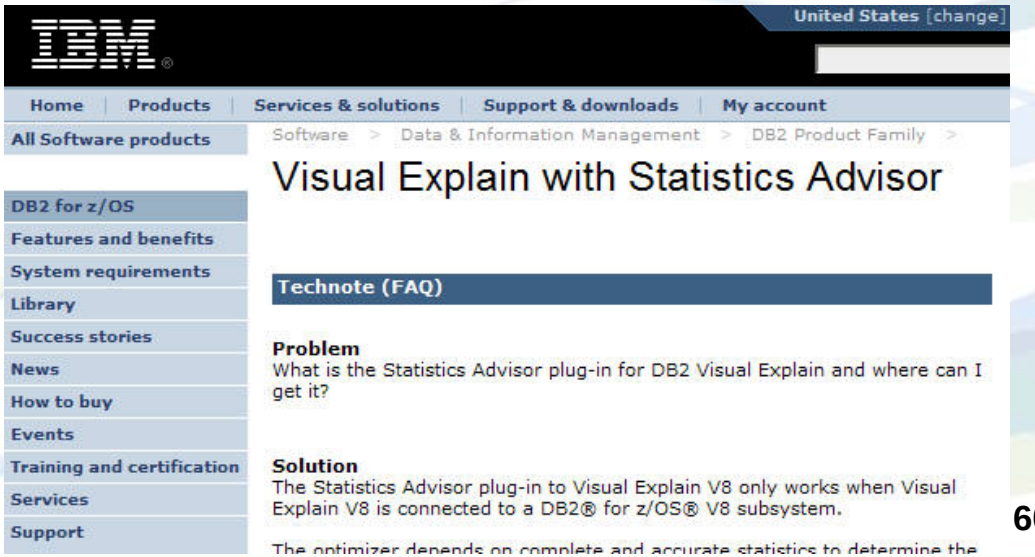


Suggestions for one Siebel query



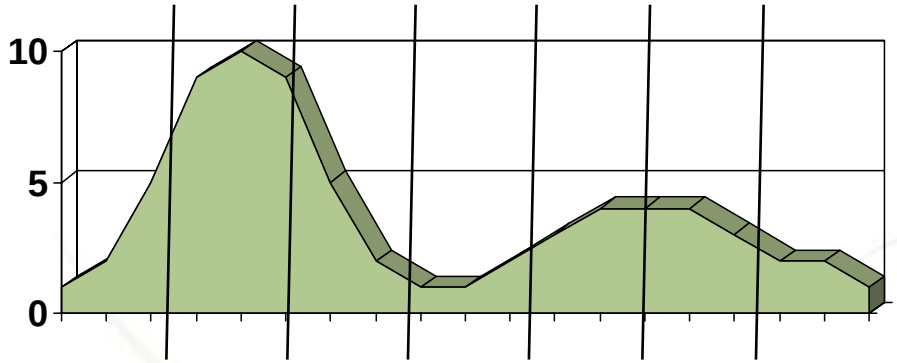
Statistics Advisor Current Status

- Statistics Advisor is integrated with VE now as a no-charge item
- 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



DB2 V9 for z/OS Changes to RUNSTATS

- New histogram statistics
  - Think of these as frequency distribution statistics on a range of data
  - Ideal for numeric, date, and time data types



- CPU reduction for RUNSTATS INDEX: 30-40%

Summary

- Why RUNSTATS?
- Invoking RUNSTATS
- Commonly asked questions (about the stats)
- Real-time Statistics
- Rebinding considerations
- Reorg recommendations
- When is RUNSTATS needed?
- New/changed data statistics
- New/changed index statistics
- Handling part level statistics for DPSIs
- Distribution Statistics Enhanced
- HISTORY statistics changes
- Flushing the dynamic statement cache
- What statistics should I gather?

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

DB2 UDB for z/OS information resources

- Information center
  http://publib.boulder.ibm.com/infocenter/dzichelp/index.jsp
- Information roadmap
  http://ibm.com/software/db2zos/roadmap.html
- DB2 UDB for z/OS library page
  http://ibm.com/software/db2zos/library.html
- Examples trading post
  http://ibm.com/software/db2zos/exHome.html
- DB2 for z/OS support
  http://ibm.com/software/db2zos/support.html
- Official Introduction to DB2 for z/OS
  http://ibm.com/software/data/education/bookstore

Disclaimers & Trademarks*

Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion.

You should not rely on such information to make business plans. Any discussion of OEM products is based upon information which has been publicly available and is subject to change.

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries: AIX, AS/400, DATABASE 2, DB2, OS/390, OS/400, ES/9000, MVS/ESA, Netfinity, RISC, RISC SYSTEM/6000, SYSTEM/390, SQL/DS, VM/ESA, IBM, Lotus, NOTES. The following terms are trademarks or registered trademarks of the MICROSOFT Corporation in the United States and/or other countries: MICROSOFT, WINDOWS, ODBC

Bryan F. Smith

IBM

bfsmith@us.ibm.com