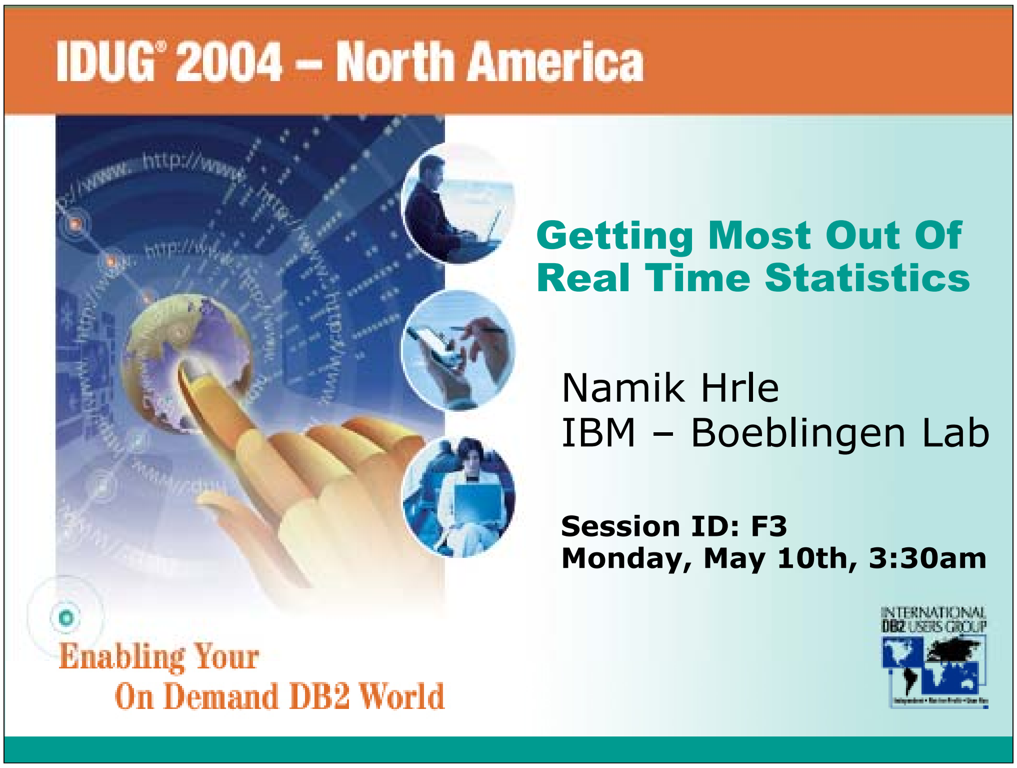


IDUG® 2004 – North America

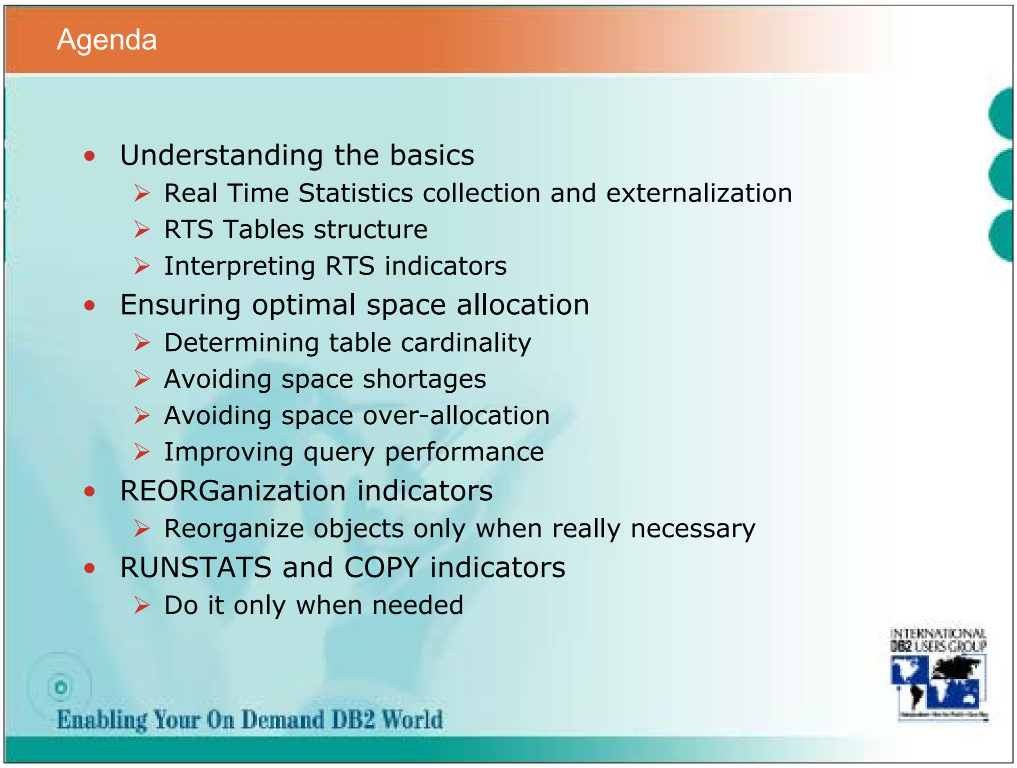
Getting Most Out Of Real Time Statistics

Namik Hrle
IBM – Boeblingen Lab

Session ID: F3
Monday, May 10th, 3:30am

Enabling Your On Demand DB2 World

The Real Time Statistics a vital building block on DB2's road to becoming autonomic, i.e. to be a database management system with zero administration and maintenance overhead. It provides a means of knowing when a preventive or corrective database administrative action should be performed and we can start reaping the feature's benefits today: the indicators are ready to be used by database administrators who prefer preventing problems to fixing them. This presentation will show you how.



Agenda

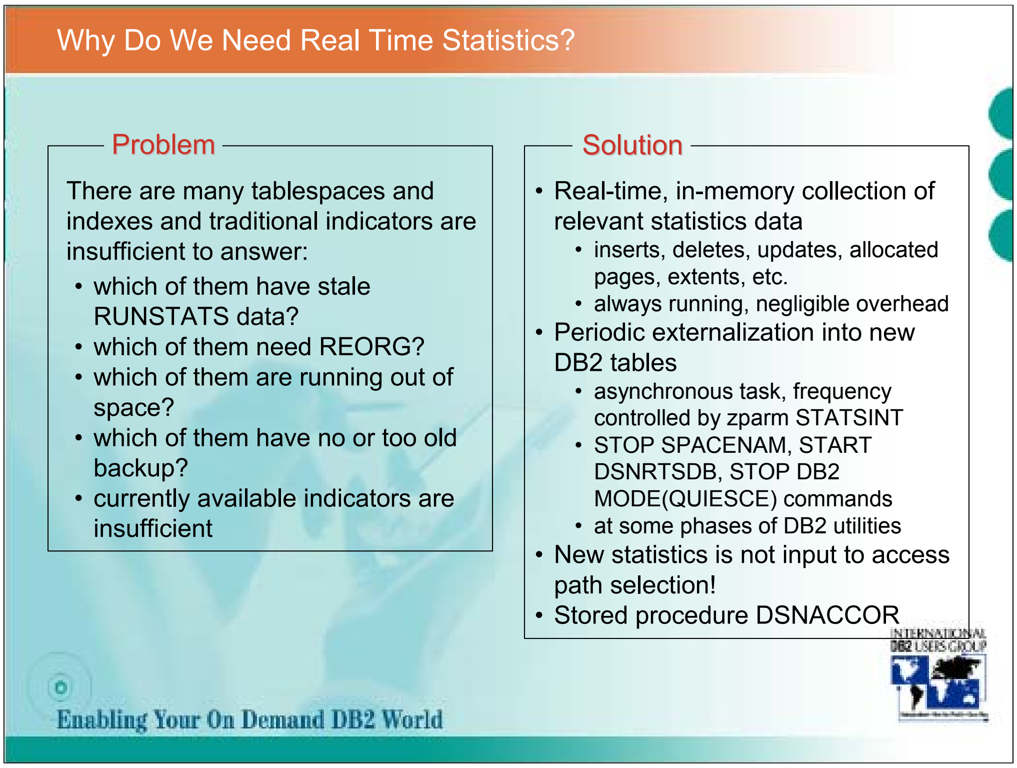
- Understanding the basics
  - Real Time Statistics collection and externalization
  - RTS Tables structure
  - Interpreting RTS indicators
- Ensuring optimal space allocation
  - Determining table cardinality
  - Avoiding space shortages
  - Avoiding space over-allocation
  - Improving query performance
- REORGanization indicators
  - Reorganize objects only when really necessary
- RUNSTATS and COPY indicators
  - Do it only when needed

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

DB2 UDB for OS/390 and z/OS Version 7 introduced a vital building block on its road to becoming autonomic, i.e. to be a database management system with zero administration and maintenance overhead: the Real Time Statistics.

This feature provides a means of knowing when a preventive or corrective database administrative action should be performed. The set of real time indicators can be input to an expert system that would prevent space shortages, optimize space usage and data access, back up objects in a timely manner and refresh the statistics used by the optimizer. But even without this automation, we can start reaping the feature's benefits today: the indicators are ready to be used by database administrators who prefer preventing problems to fixing them.



Why Do We Need Real Time Statistics?

Problem

There are many tablespaces and indexes and traditional indicators are insufficient to answer:

- which of them have stale RUNSTATS data?
- which of them need REORG?
- which of them are running out of space?
- which of them have no or too old backup?
- currently available indicators are insufficient

Solution

- Real-time, in-memory collection of relevant statistics data
  - inserts, deletes, updates, allocated pages, extents, etc.
  - always running, negligible overhead
- Periodic externalization into new DB2 tables
  - asynchronous task, frequency controlled by zparm STATSINT
  - STOP SPACENAM, START DSNRTSDB, STOP DB2 MODE(QUIESCE) commands
  - at some phases of DB2 utilities
- New statistics is not input to access path selection!
- Stored procedure DSNACCOR

Enabling Your On Demand DB2 World

With RTS, DB2 always generates in-memory statistics for each table space and index space. When the statistics is to be externalized, DB2 examines the in-memory data, calculates the new totals, updates the new real-time statistic tables with the new totals and resets the in-memory data. This process is an asynchronous task.

The tables where the real-time statistics is stored are SYSIBM.TABLESPACESTATS and SYSIBM.INDEXSPACESTATS in the DSNRTSDB.DSNRTSTS tablespace. For DDL see SDSNSAMP(DSNTESS).

The following conditions must be fulfilled if the in-memory statistics is to be externalized:

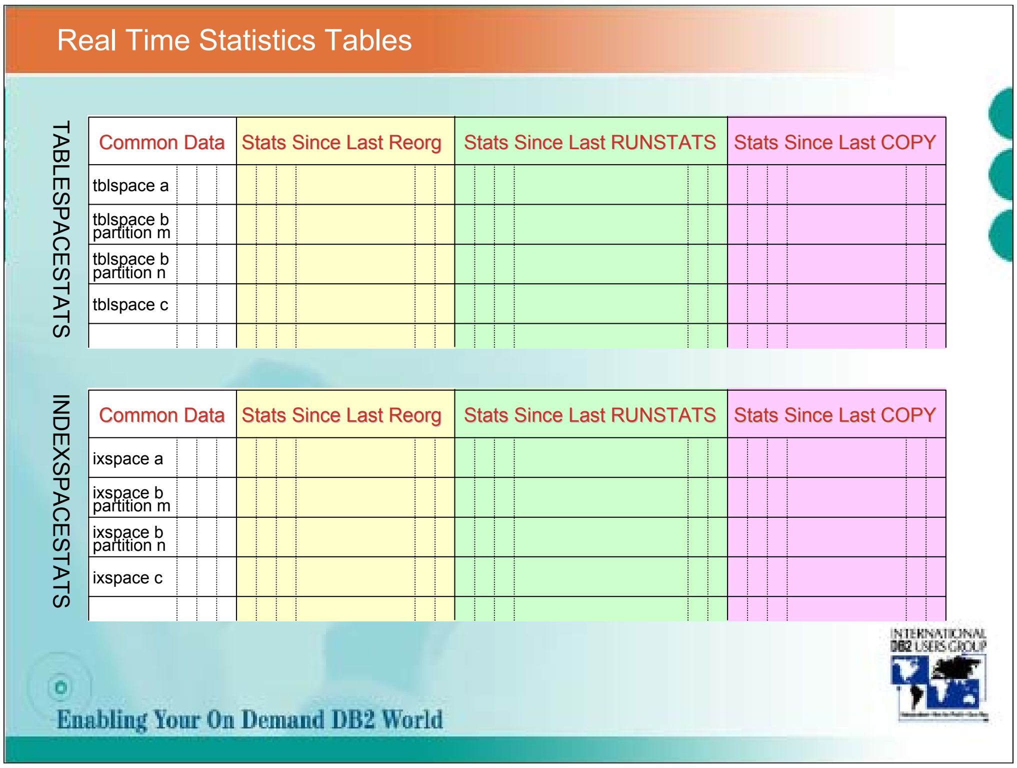
•The required database, table space, tables and indexes that will store the real-time statistics are created by the user. Note that these objects are not automatically created by DB2.

•These objects' attributes comply with required settings, i.e. the objects must be created with the specified object names, schema name and structure.

•The DSNRTSDB database (where the objects are created) must be started RW. Note that the database is implicitly stopped immediately after creation.

When externalizing in-memory statistics, DB2 inserts a row for each partition or non-partitioned page set. If a row already exists it will be updated. The absolute statistic values (for example, Total Rows) are replaced with the new values and incremental values are summed with the in-memory statistics.

STATSINT is specified in minutes. The default value is 30. It can be changed online.



Real Time Statistics Tables

TABLESPACESTATS

Common Data	Stats Since Last Reorg	Stats Since Last RUNSTATS	Stats Since Last COPY
tblspace a			
tblspace b partition m			
tblspace b partition n			
tblspace c			

INDEXSPACESTATS

Common Data	Stats Since Last Reorg	Stats Since Last RUNSTATS	Stats Since Last COPY
ixspace a			
ixspace b partition m			
ixspace b partition n			
ixspace c			

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

•Statistics is maintained for every non-partitioned tablespace and index and every partition of partitioned tablespace and index for which some changes occurred.

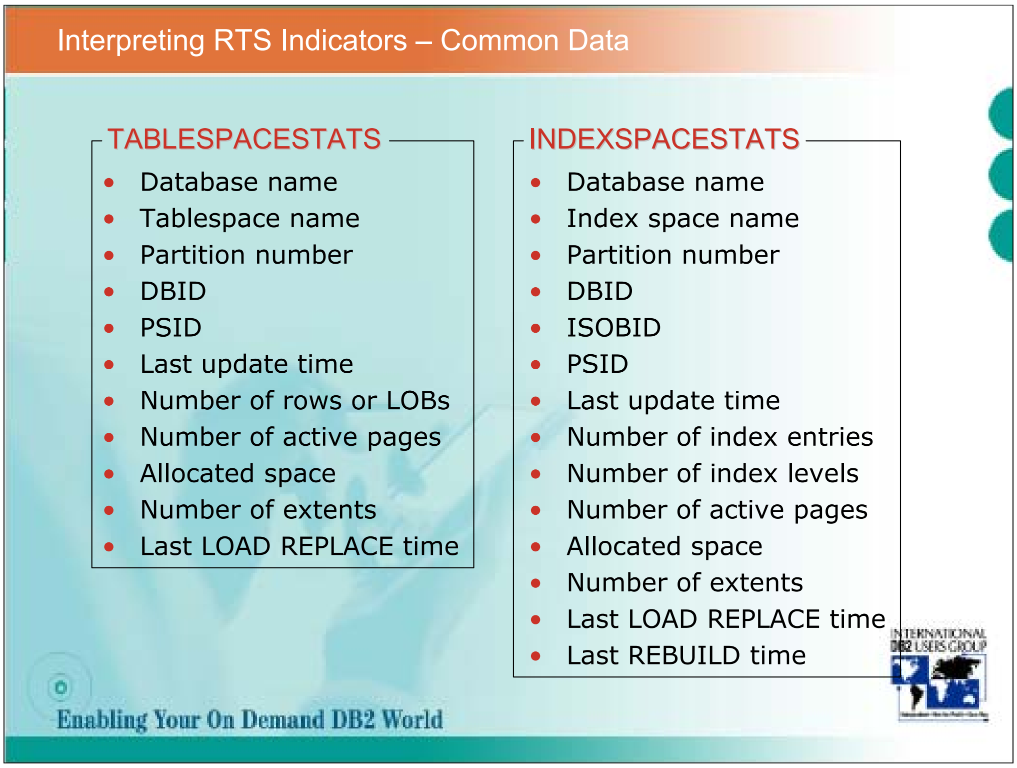
•The data can be grouped into object identification and common statistics, and one group for each: changes since the last REORG, changes since the last RUNSTATS and changes since the last COPY which are stored independently.

•DB2 utilities do not continuously update the new tables but before or after some of their phases.

•The new tables should be used for estimates and not for the exact values of the collected statistics.

•Note that TABLESPACESTATS and INDEXSPACESTATS data is not used by the DB2 Optimizer. You still need to run RUNSTATS to update the traditional catalog tables.

•DSNACCOR stored procedure will query the new tables to determine which DB2 objects should be reorganized, should have their statistics updated, should be image copied, have exceeded number of extents, or are in a restricted state. The default scope for this stored procedure is to scan all data in the tables and provide recommendations for any condition mentioned above. To override either the scope of the data queried or the type of recommendation evaluated, the caller will pass along the appropriate parameter with the call to the stored procedure. Parameters can also be passed that override the default threshold settings. Calls to the stored procedure that do not provide parameters will employ default parameters (or thresholds) where appropriate.



Interpreting RTS Indicators – Common Data

TABLESPACESTATS

- Database name
- Tablespace name
- Partition number
- DBID
- PSID
- Last update time
- Number of rows or LOBs
- Number of active pages
- Allocated space
- Number of extents
- Last LOAD REPLACE time

INDEXSPACESTATS

- Database name
- Index space name
- Partition number
- DBID
- ISOBID
- PSID
- Last update time
- Number of index entries
- Number of index levels
- Number of active pages
- Allocated space
- Number of extents
- Last LOAD REPLACE time
- Last REBUILD time

Enabling Your On Demand DB2 World

•For a given row UPDATESTATSTIME is updated with the current timestamp whenever the row is inserted or updated. One use of this column is to determine how old the statistics is and for how long a particular object has not been changed (this is not 100% reliable value, so do not use it for recovery purposes).

•The number of rows and the number of index entries are usually used as denominators in ratios indicating the percentage of affected rows and index entries respectively. The number of rows is equal to the number of index entries for single-table tablespaces.

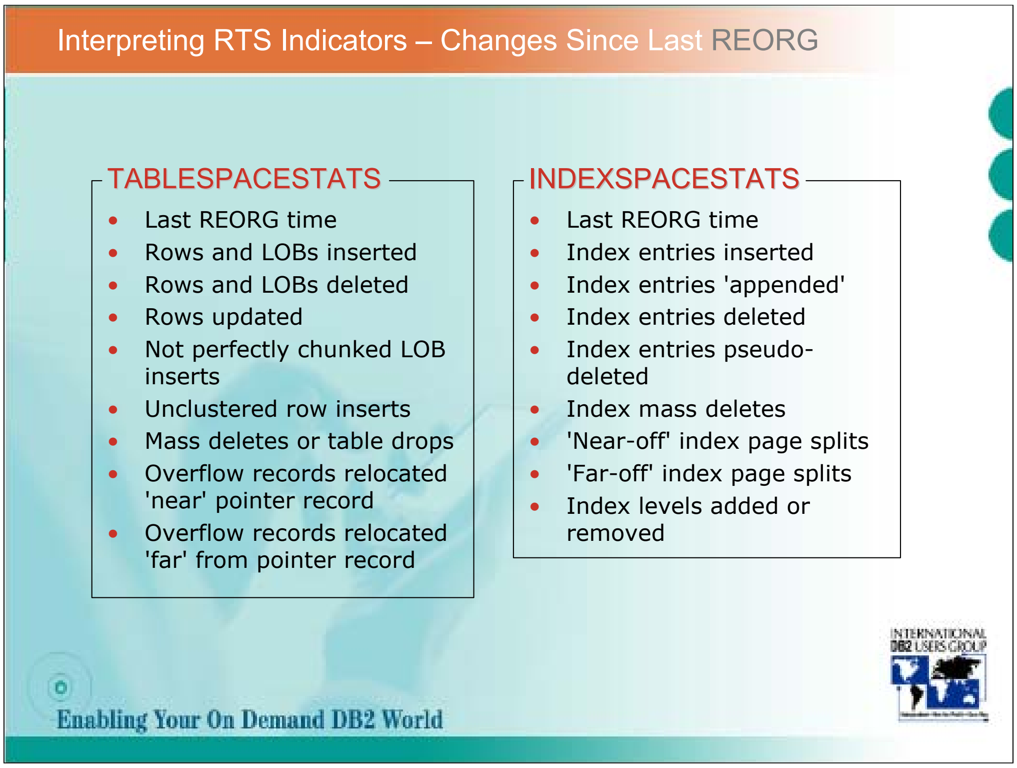
•Note that the field TOTALROWS is set to null until a reorg for that tablespace is done. This is understandable for tables that were created before RTS was enabled, but unfortunately it is true even for the tables that were created after the RTS was enabled.

•The number of active pages is equivalent to the number of preformatted pages (across all underlying data sets). It should be used as a denominator in ratios indicating the percentage of affected pages.

•The amount of allocated space (across all the underlying data sets) is given in KB.

•For multi-piece table/index spaces the number of extents is given for the last data set only. Should be used for adjusting primary and secondary allocation quantity when approaching extent limits.

•If index is enabled for COPY and the last REBUILD is more recent than the last COPY, it is a signal that a copy is due.



Interpreting RTS Indicators – Changes Since Last REORG

TABLESPACESTATS

- Last REORG time
- Rows and LOBs inserted
- Rows and LOBs deleted
- Rows updated
- Not perfectly chunked LOB inserts
- Unclustered row inserts
- Mass deletes or table drops
- Overflow records relocated 'near' pointer record
- Overflow records relocated 'far' from pointer record

INDEXSPACESTATS

- Last REORG time
- Index entries inserted
- Index entries 'appended'
- Index entries deleted
- Index entries pseudo-deleted
- Index mass deletes
- 'Near-off' index page splits
- 'Far-off' index page splits
- Index levels added or removed

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

•The indicators report changes since the last REORG or last LOAD REPLACE for tablespaces and, additionally, since the last REBUILD for indexes.

•Relatively large number of inserts, deletes and updates are not necessarily a reason for reorg if other threshold have not been reached. Nevertheless, if some queries perform badly and there were lots of rows inserted, deleted or updated since the last REORG (e.g. more than 20%), a new REORG could improve performance even if no other threshold is reached.

•A LOB is considered perfectly chunked if its pages are contained in the minimum number of chunks possible. Consider 10% as a threshold.

•A row is considered well clustered if it is inserted on a page within plus/minus 16 pages of the ideal candidate page.

•If the number of mass deletes or table drops (in case of segmented tablespace) is not zero, consider REORG.

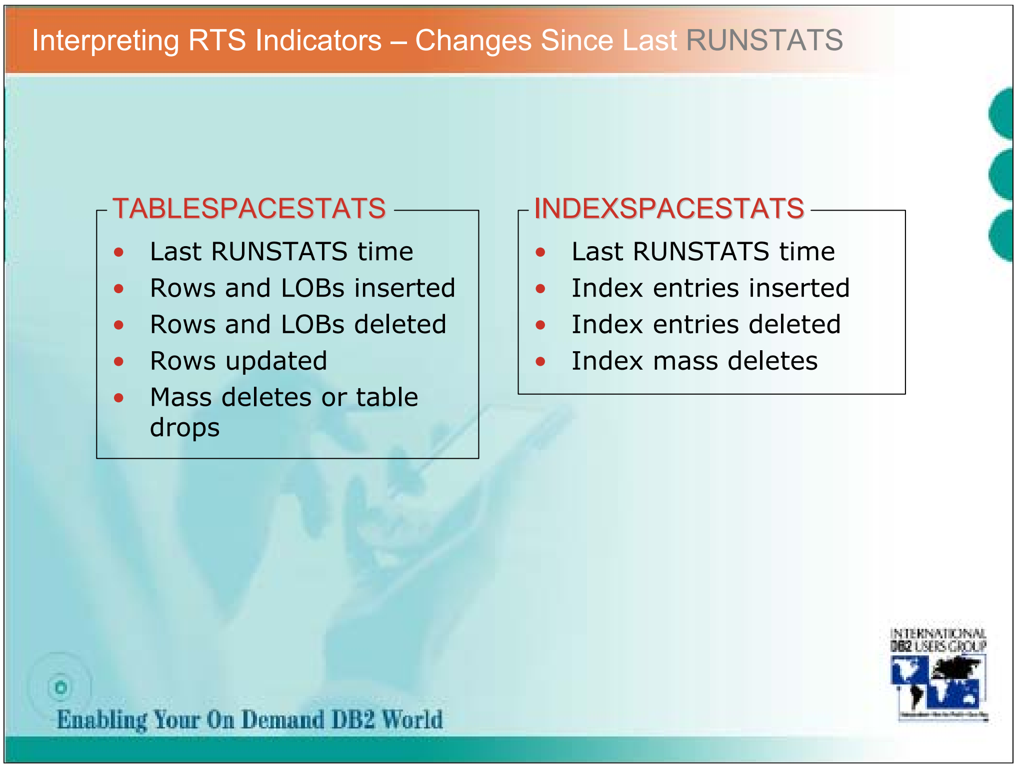
•For non-segmented tablespaces an overflow record is considered 'near' ('far' from) the pointer record if two pages differ by 16 or less (17 or more). For segmented tablespaces an overflow record is considered 'near' ('far' from) the pointer record if two pages differ by SEGSIZE*2 or less (SEGSIZE*2+1 or more). The sum of 'near' and 'far' overflow records should not be larger than 10% (non-data sharing), i.e. 5% (data sharing).

•The new index leaf pages as well as new non leaf pages could be stored in any available space within an index pageset when the rows are 'appended' (insert with the highest key). Reorg Index would rearrange them in the optimal physical sequence. Use 10% as a threshold.

•Threshold for pseudo-deletes should be 10% or less.

•An index split page is considered 'near-off' ('far-off') the present page if the difference is less than 16 (16 or more). If 10% of all pages are near or far-off, consider index reorg.

•If the number of index levels increased, look at the number of pseudo-deleted entries. A high number indicates a possibility to reduce the number of levels by index reorg.



Interpreting RTS Indicators – Changes Since Last RUNSTATS

TABLESPACESTATS

- Last RUNSTATS time
- Rows and LOBs inserted
- Rows and LOBs deleted
- Rows updated
- Mass deletes or table drops

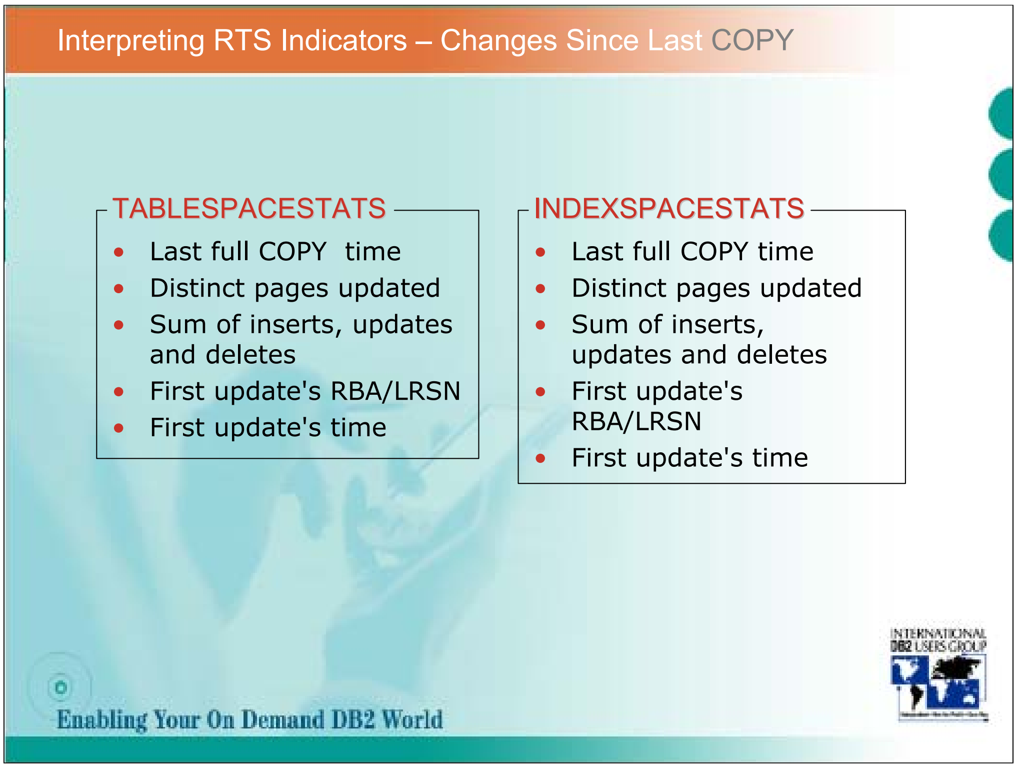
INDEXSPACESTATS

- Last RUNSTATS time
- Index entries inserted
- Index entries deleted
- Index mass deletes

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

These indicators are self-explanatory. Depending on the dynamics of changes to your tablespaces and index the optimal time for refreshing catalog statistics can be determined.



Interpreting RTS Indicators – Changes Since Last COPY

TABLESPACESTATS

- Last full COPY time
- Distinct pages updated
- Sum of inserts, updates and deletes
- First update's RBA/LRSN
- First update's time

INDEXSPACESTATS

- Last full COPY time
- Distinct pages updated
- Sum of inserts, updates and deletes
- First update's RBA/LRSN
- First update's time

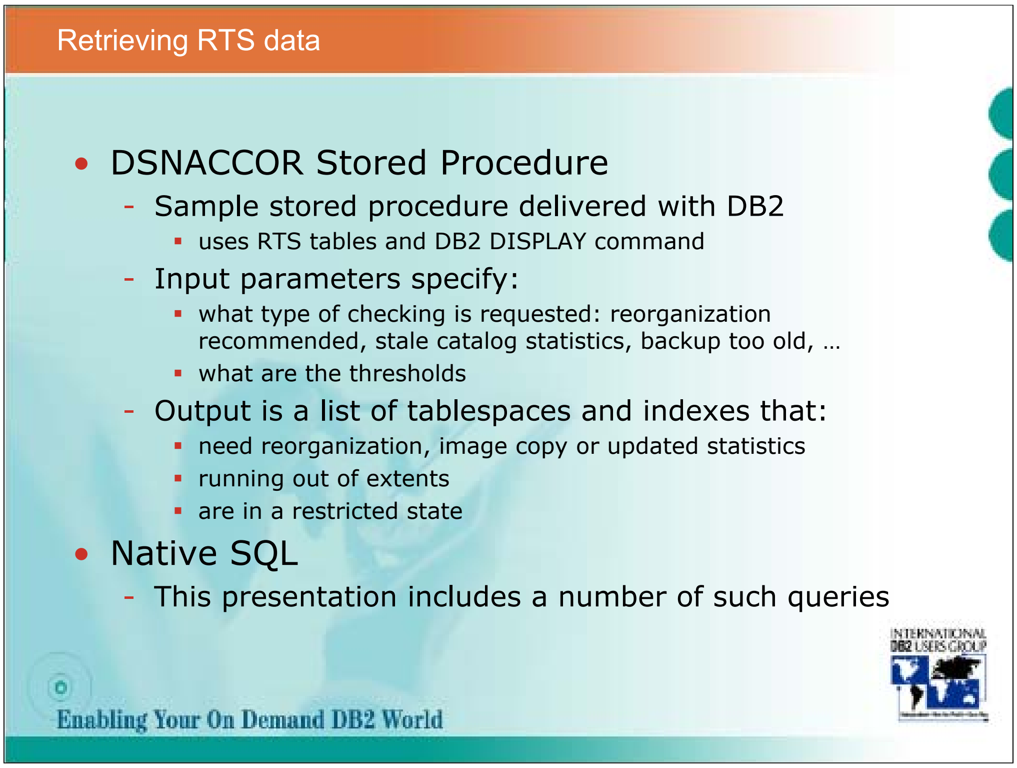
INTERNATIONAL DB2 USERS GROUP

Enabling Your On Demand DB2 World

•The last full COPY time should e compared with other utilities times to see is a new copy is due.

•The sum of inserts, updates and deletes approximates the number of log records that would need to be applied in a recovery situation.

•The first update's time and RBA/LRSN indicate how far in log recovery will have to go. You should consider a new copy if that point is not in active logs.



Retrieving RTS data

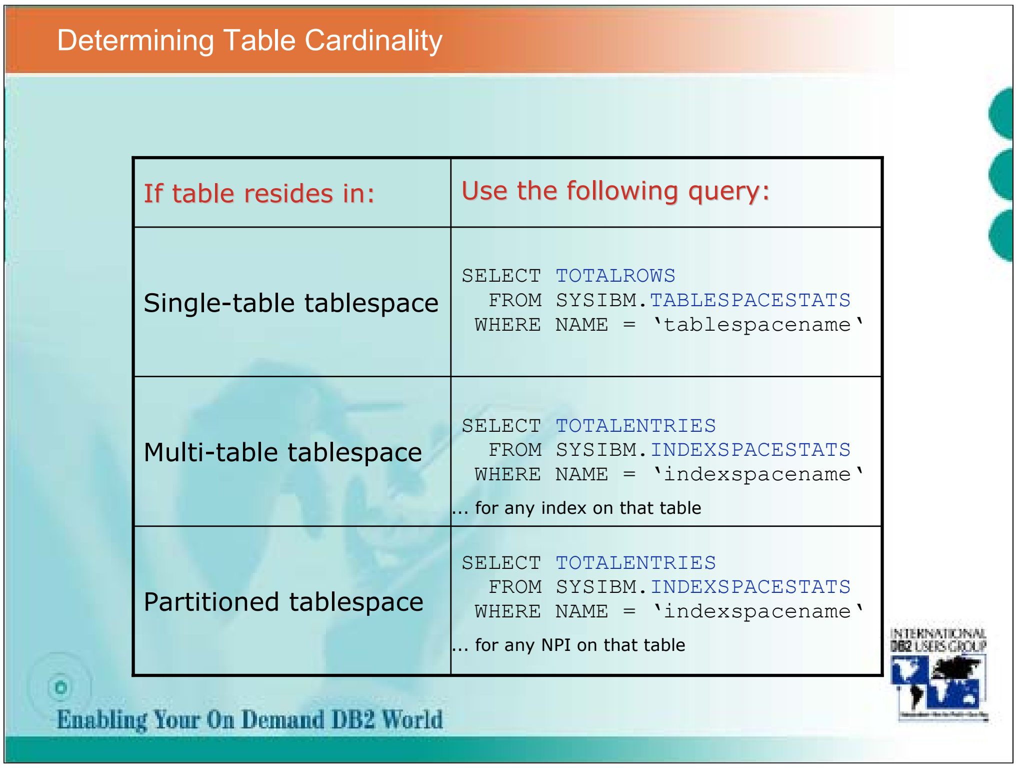
- DSNACCOR Stored Procedure
  - Sample stored procedure delivered with DB2
    - uses RTS tables and DB2 DISPLAY command
  - Input parameters specify:
    - what type of checking is requested: reorganization recommended, stale catalog statistics, backup too old, …
    - what are the thresholds
  - Output is a list of tablespaces and indexes that:
    - need reorganization, image copy or updated statistics
    - running out of extents
    - are in a restricted state
- Native SQL
  - This presentation includes a number of such queries

Enabling Your On Demand DB2 World

The content of the RTS tables can be retrieved directly by means of SQL SELECT statements or by using a special stored procedure DSNACCOR. It reads the RTS tables, applies thresholds on its entries (formulas are published in the DB2 Utility Guide and Reference) and returns a list of objects, that require RUNSTATS, REORG, COPY or have too many extents. This list also contains objects that are in some of restricted states (retrieved by appropriate DISPLAY commands).

DSNACCOR is well suited to be used by monitoring programs. Examples are Control Center and SAP on DB2 for z/OS exploits DSNACCOR in its Computing Center Management System. SAP CCMS creates alerts based on the objects lists returned by DSNACCOR. For example, the alert *Objects in restricted state* draws the DBA's attention to the objects that need to be fixed. All other alerts generated from DSNACCOR output can be resolved automatically by scheduling the appropriate jobs such as *RUNSTATS on objects that need new statistics* or *Online Reorg on suggested indexes* periodically in CCMS scheduling tool Planning Calendar.

An exception table is an optional, user-created DB2 table that you can use to place information in the INEXCEPTTABLE column of the recommendations result set. You can put any information in the INEXCEPTTABLE column, but the most common use of this column is to filter the recommendations result set. Each row in the exception table represents an object for which you want to provide information for the recommendations result set.



Determining Table Cardinality

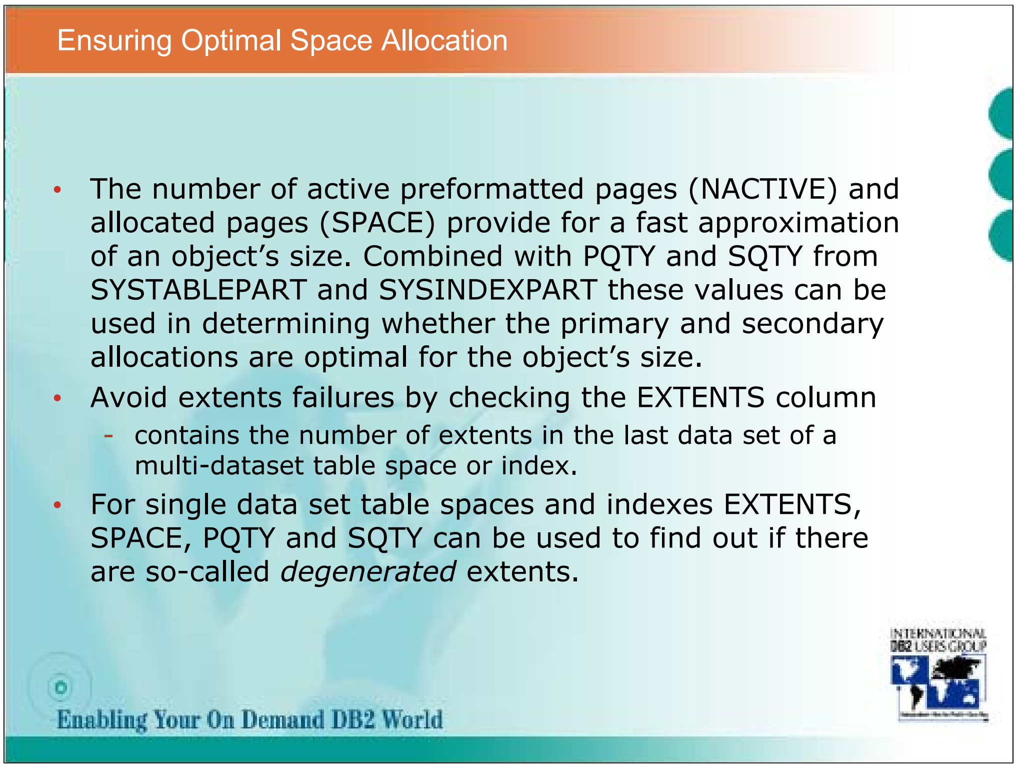
If table resides in:	Use the following query:
Single-table tablespace	`SELECT TOTALROWS FROM SYSIBM.TABLESPACESTATS WHERE NAME = 'tablespacename'`
Multi-table tablespace	`SELECT TOTALENTRIES FROM SYSIBM.INDEXSPACESTATS WHERE NAME = 'indexspacename'` … for any index on that table
Partitioned tablespace	`SELECT TOTALENTRIES FROM SYSIBM.INDEXSPACESTATS WHERE NAME = 'indexspacename'` … for any NPI on that table

Enabling Your On Demand DB2 World

•***TOTALROWS*** in TABLESPACESTATS contains the number of rows across all the tables in that table space at the time the corresponding TABLESPACESTATS entry was last time updated. Depending on the frequency of externalizing real-time statistics it can be a very good approximation of the actual number of rows. For single-table table spaces it is equivalent to the number of rows in the table and it is much faster to retrieve than SELECT COUNT(*). Of course, when the exact number of rows is needed, there is no better alternative than using the COUNT function. Note that the field TOTALROWS is set to null until a reorg for that tablespace is done. This is understandable for tables that were created before RTS was enabled, but unfortunately it is true even for the tables that were created after the RTS was enabled.

•For tables in multi-table table spaces the approximate cardinality can be derived from ***TOTALENTRIES*** in INDEXSPACESTATS that contains the number of index entries including duplicates, which means that any index on the table can be used in the derivation.

•Note that for a partitioned table the fastest way to get its cardinality is to retrieve TOTALENTRIES for any of its non-partitioning indexes. If there are no non-partitioning indexes, then sum TOTALENTRIES for all the index partitions or sum TOTALROWS for all the table space partitions.



Ensuring Optimal Space Allocation

- The number of active preformatted pages (NACTIVE) and allocated pages (SPACE) provide for a fast approximation of an object's size. Combined with PQTY and SQTY from SYSTABLEPART and SYSINDEXPART these values can be used in determining whether the primary and secondary allocations are optimal for the object's size.
- Avoid extents failures by checking the EXTENTS column
  - contains the number of extents in the last data set of a multi-dataset table space or index.
- For single data set table spaces and indexes EXTENTS, SPACE, PQTY and SQTY can be used to find out if there are so-called *degenerated* extents.

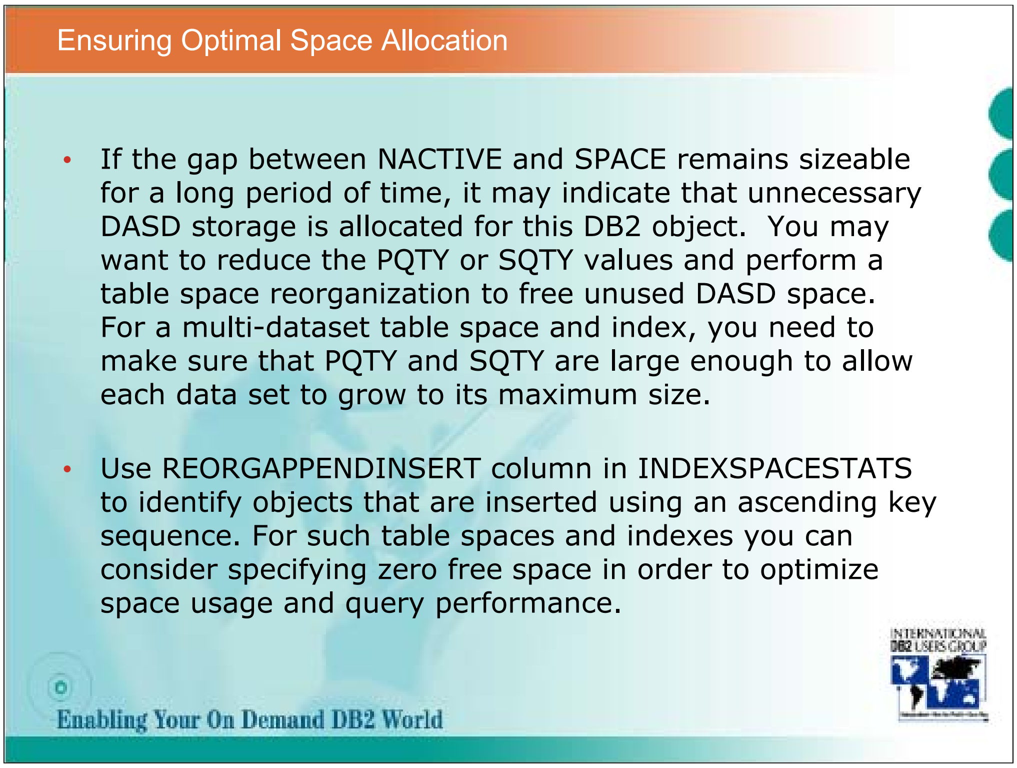
Enabling Your On Demand DB2 World
INTERNATIONAL DB2 USERS GROUP

•Note that the values are generally given in different units: NACTIVE is in number of pages (corresponding to the table space page size), SPACE is in number of kilobytes and PQTY, SQTY are in number of 4K blocks.

•Too small values for PQTY and SQTY can lead to an excessive number of extents and potentially cause extend request failures due to reaching the existing limits on the maximum number of extents per data set.

•Degenerated extents can appear due to storage fragmentation within a DASD volume. When a data set needs to be extended DB2 uses SQTY to add additional free space for a DB2 object. If the disk storage is fragmented, DB2 could use up to five extents (sum of which is equal to SQTY) to satisfy a single extend request.

•For multi-dataset table spaces and indexes, the total number of extents is not reported by the real time statistics tables (you can use LISTCAT for that purpose), but an approximation can be made by using PQTY, SQTY and SPACE.



Ensuring Optimal Space Allocation

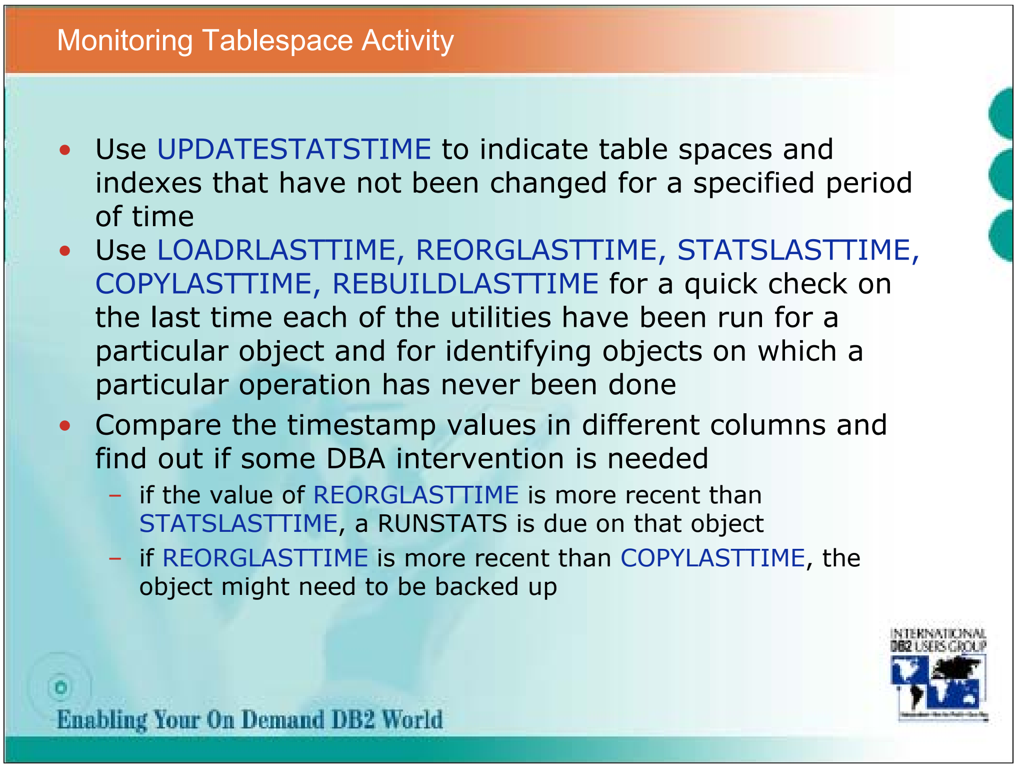
- If the gap between NACTIVE and SPACE remains sizeable for a long period of time, it may indicate that unnecessary DASD storage is allocated for this DB2 object. You may want to reduce the PQTY or SQTY values and perform a table space reorganization to free unused DASD space. For a multi-dataset table space and index, you need to make sure that PQTY and SQTY are large enough to allow each data set to grow to its maximum size.
- Use REORGAPPENDINSERT column in INDEXSPACESTATS to identify objects that are inserted using an ascending key sequence. For such table spaces and indexes you can consider specifying zero free space in order to optimize space usage and query performance.

Enabling Your On Demand DB2 World

•The gap between NACTIVE and SPACE is the reserved DASD storage that has not yet been preformatted by DB2. On an insert, DB2 looks for existing free space before preformatting additional space. Typically, when additional free space is needed, DB2 preformats two cylinders at a time. For very small table spaces (i.e. PQTY or SQTY is less than one cylinder), the preformat is two tracks at a time.

•The SPACE values are also maintained for the TEMP and the WORKFILE table spaces. Adding the SPACE value for all DB2 objects allows you to tell how much DASD storage is used to store permanent and temporary data for a DB2 system. Note that the indexes created on the Declare Global Temp Tables are not maintained in the INDEXSPACESTATS table. The storage for each temporary index is freed as soon as the TEMP table is dropped.

•The ***REORGAPPENDINSERT*** column in INDEXSPACESTATS contains the number of inserts into an index since the last REORG for which the key was higher than any other existing key. This can be used to identify objects that are inserted using an ascending key sequence. For such table spaces and indexes you can consider specifying zero free space in order to optimize space usage and query performance.



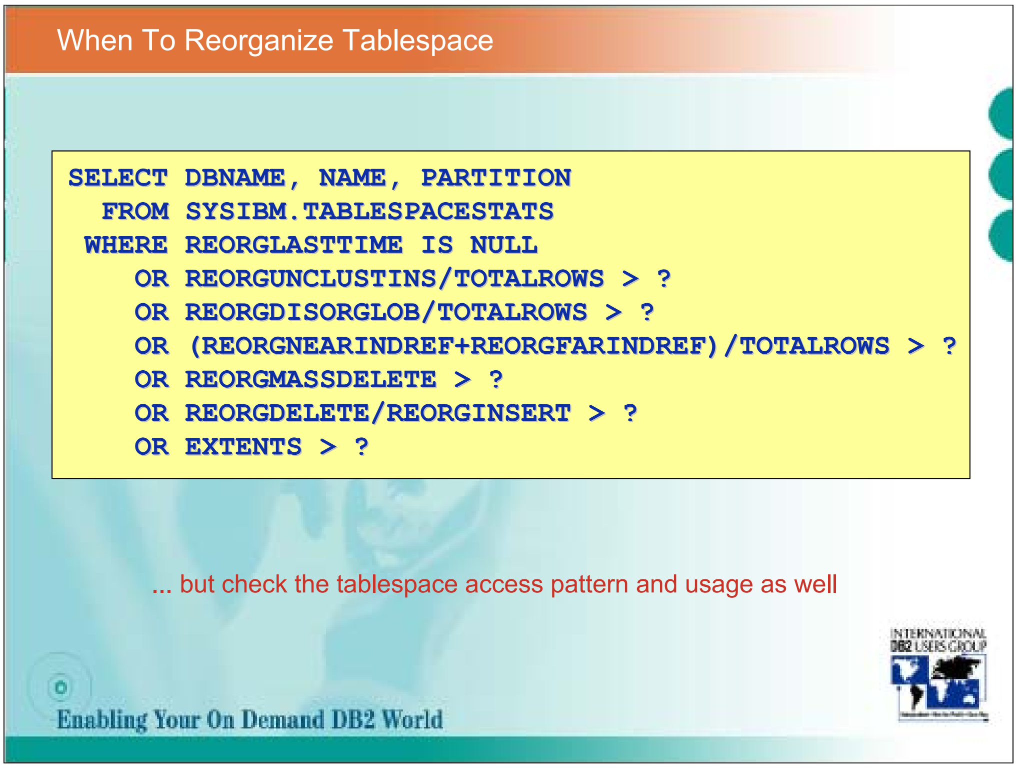
Monitoring Tablespace Activity

- Use UPDATESTATSTIME to indicate table spaces and indexes that have not been changed for a specified period of time
- Use LOADRLASTTIME, REORGLASTTIME, STATSLASTTIME, COPYLASTTIME, REBUILDLASTTIME for a quick check on the last time each of the utilities have been run for a particular object and for identifying objects on which a particular operation has never been done
- Compare the timestamp values in different columns and find out if some DBA intervention is needed
  - if the value of REORGLASTTIME is more recent than STATSLASTTIME, a RUNSTATS is due on that object
  - if REORGLASTTIME is more recent than COPYLASTTIME, the object might need to be backed up

Enabling Your On Demand DB2 World

•The UPDATESTATSTIME field contains the timestamp of the last update to the row in TABLESPACESTATS and INDEXSPACESTATS. As these tables are not updated for read-only objects, UPDATESTATSTIME can indicate table spaces and indexes that have not been changed for a specified period of time. Note that the column is also updated when the real-time statistics are reset due to executing DB2 database administration utilities (e.g. LOAD, REORG, RUNSTATS, COPY, REBUILD).

•Each row contains timestamps for the most recent executions of typical database administration utilities: ***LOADRLASTTIME***, ***REORGLASTTIME***, ***STATSLASTTIME***, ***COPYLASTTIME***, ***REBUILDLASTTIME***. You can use them for a quick check on the last time each of the utilities have been run for a particular object and for identifying objects on which a particular operation has never been done (a null value in the corresponding column). You can also compare the timestamp values in different columns and find out if some DBA intervention is needed. For example, if the value of REORGLASTTIME is more recent than STATSLASTTIME, a RUNSTATS is due on that object. Or, if REORGLASTTIME is more recent than COPYLASTTIME, the object might need to be backed up.



When To Reorganize Tablespace

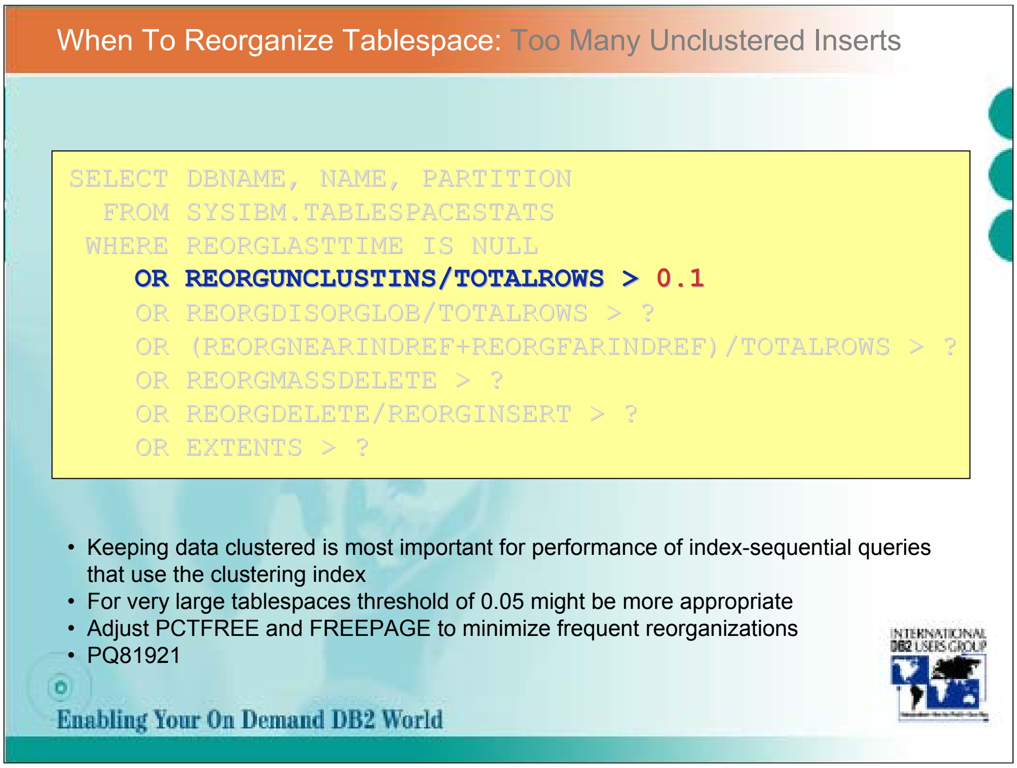
```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGUNCLUSTINS/TOTALROWS > ?
    OR REORGDISORGLOB/TOTALROWS > ?
    OR (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > ?
    OR REORGMASSDELETE > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

... but check the tablespace access pattern and usage as well

Enabling Your On Demand DB2 World

Reorganization can very significantly improve query performance and space utiliztaion, but it is a resource- and time-consuming operation that can also introduce serious contention problems into your system. This is why knowing exactly when an object‘s reorganization should be run is extremely important. The Real Time Statistics provides very reliable indicators that help you determine if a reorganization should be performed. In addition to that you need to find out if the statements set for that object would actually benefit from the reorganization. Namely, if all the accesses are through a unique, equal index (bringing only a single row at a time) reorganizing tablespace to reestablish clustering is an unnecessary overhead.

The question marks in the SELECT statement will need to be filled according to the specific customer‘s installation usage characteristics. The next pages will offer some values that should be appropriate in most cases.



When To Reorganize Tablespace: Too Many Unclustered Inserts

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGUNCLUSTINS/TOTALROWS > 0.1
    OR REORGDISORGLOB/TOTALROWS > ?
    OR (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > ?
    OR REORGMASSDELETE > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

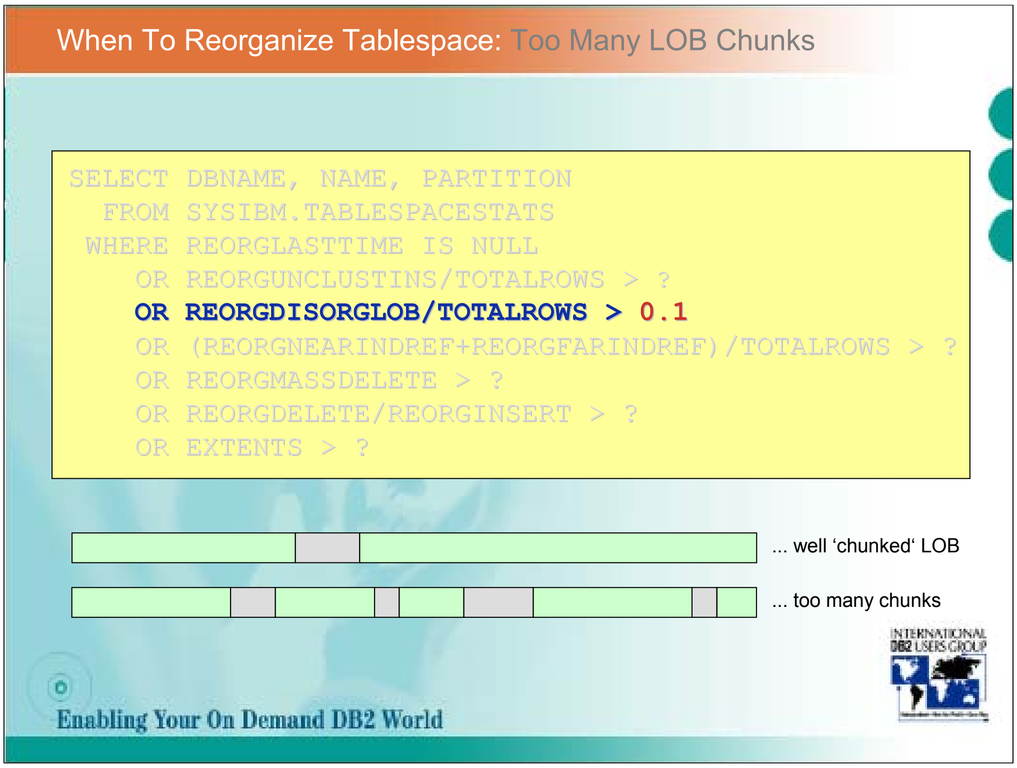
- Keeping data clustered is most important for performance of index-sequential queries that use the clustering index
- For very large tablespaces threshold of 0.05 might be more appropriate
- Adjust PCTFREE and FREEPAGE to minimize frequent reorganizations
- PQ81921

Enabling Your On Demand DB2 World

REORGUNCLUSTINS contains the number of inserts since the last REORG where the rows were inserted more than 16 pages away from their optimal position with respect to the clustering index. It should be used in the same context as the familiar CLUSTERRATIO from SYSINDEXES. If the number of such inserts comprises more than 10% of all rows (given in the TOTALROWS column), you should consider reorganizing the table space because the high number of unclustered inserts has a significant impact on performance of queries that access data sequentially via the clustering index. For large tablespaces the threshold can be lower, e.g. 5%.

Before REORG, you should consider adjusting PCTFREE and/or FREEPAGE to reserve more free space for inserts. This results in rows being inserted in the same sequence as the clustering index key values.

PQ81921 resolves the problem that the algorithm for computing the value of REORGUNCLUSTINS was based on rows that are inserted within 16 pages of the ideal page. This value was not the same as the CLUSTERRATIO indicated in SYSIBM.SYSINDEXES, and could have been misleading for customers when developing a method for Reorganizing tablespaces. Moreover when records are inserted in the correct sequence except the first one, the resulting value became incorrect after 18 inserts.



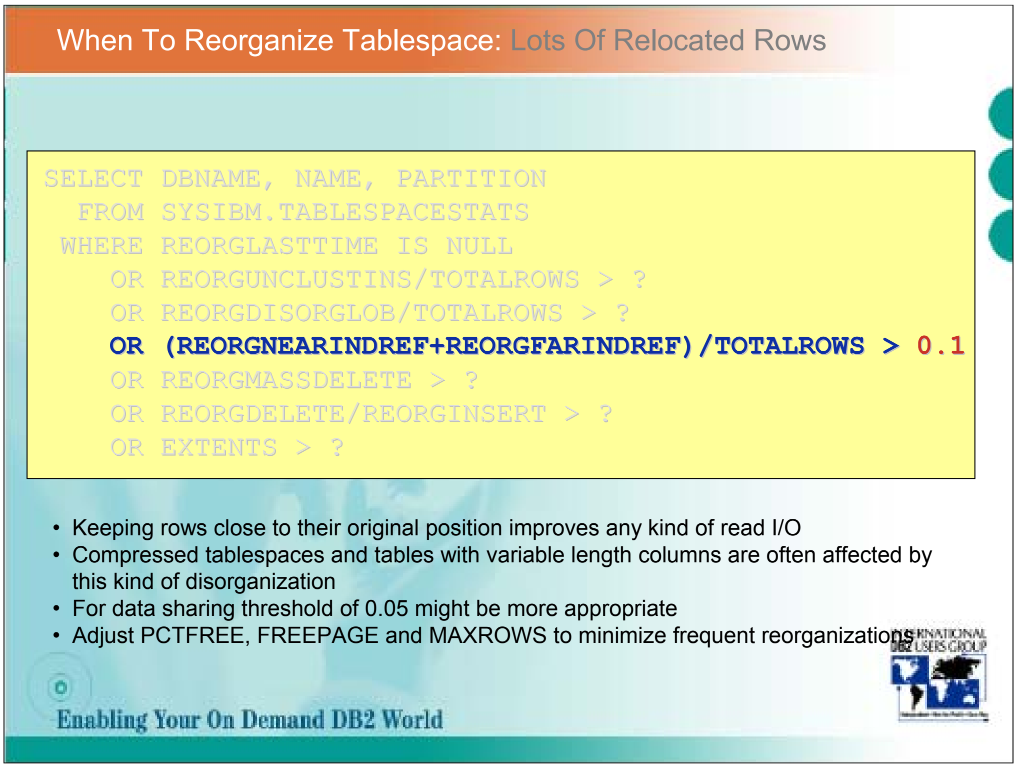
When To Reorganize Tablespace: Too Many LOB Chunks

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGUNCLUSTINS/TOTALROWS > ?
    OR REORGDISORGLOB/TOTALROWS > 0.1
    OR (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > ?
    OR REORGMASSDELETE > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

... well ‘chunked‘ LOB
... too many chunks
INTERNATIONAL DB2 USERS GROUP

Enabling Your On Demand DB2 World

REORGDISORGLOB contains the number of inserts involving LOBs (Large Objects) since the last REORG where the LOBs were inserted in more than the minimum required number of chunks. If the number of such inserts comprises more than 10% of all rows, reorganizing the table space would be recommended .



When To Reorganize Tablespace: Lots Of Relocated Rows

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGUNCLUSTINS/TOTALROWS > ?
    OR REORGDISORGLOB/TOTALROWS > ?
    OR (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > 0.1
    OR REORGMASSDELETE > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

- Keeping rows close to their original position improves any kind of read I/O
- Compressed tablespaces and tables with variable length columns are often affected by this kind of disorganization
- For data sharing threshold of 0.05 might be more appropriate
- Adjust PCTFREE, FREEPAGE and MAXROWS to minimize frequent reorganizations

Enabling Your On Demand DB2 World

If an update causes moving the row off its original position (due to insufficient space for the row extension), the counters ***REORGNEARINDREF*** or ***REORGFARINDREF*** get incremented, depending on how far away the relocated row needs to go. Therefore, these can be non-zero for table spaces that satisfy any of the following conditions:

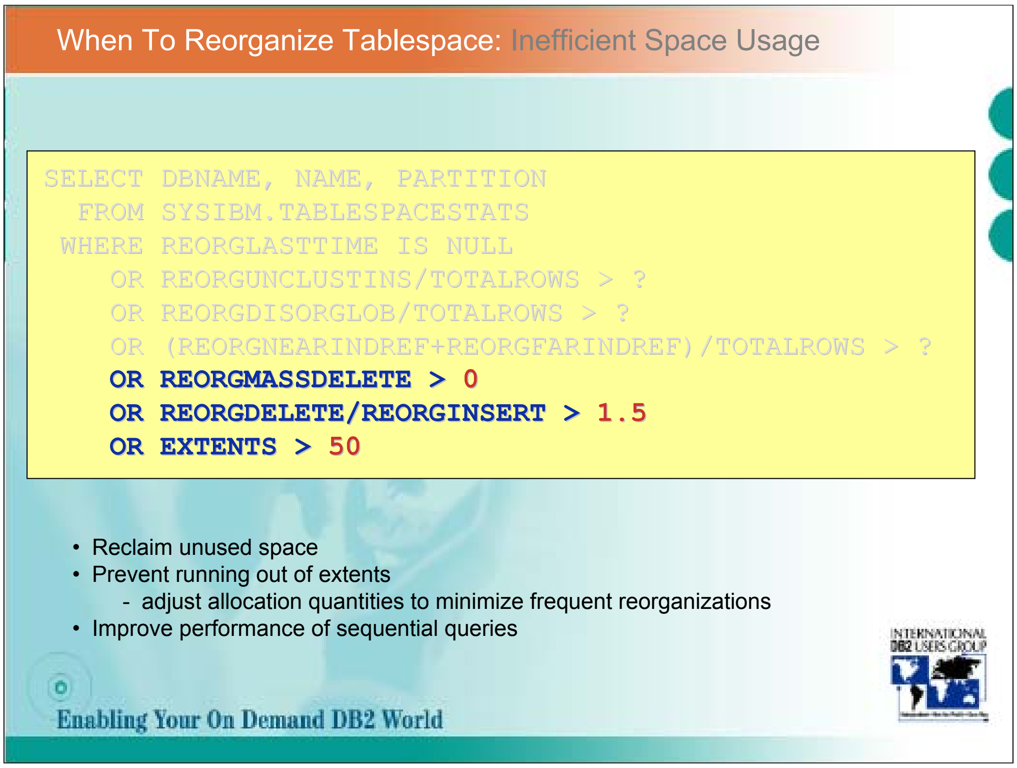
•One or more tables in the table space contain one or more variable length columns (i.e. VARCHAR or VARGRAPHIC).

•The table space has data compression enabled.

•One or more columns have been added to a fixed length table and the table space has not yet been reorganized.

For a relocated row, each reference requires accessing two data pages and possibly two disk I/Os. In a data sharing system, it may also result in an extra Coupling Facility request.

A table space reorganization is recommended if the sum of these two counters is more than 10% of all the rows. For data sharing environments a threshold of 5% would be more appropriate. Prior to the reorganization consider adjusting free space parameters. Larger PCTFREE and smaller MAXROWS settings reduce the chance of relocating rows to different pages. Large FREEPAGE values result in rows being relocated close to the original pages improving I/O efficiency (via more efficient prefetch) for queries that use the clustering index to access the rows sequentially. Having two pages close to each other also improves I/O efficiency for a single row access.



When To Reorganize Tablespace: Inefficient Space Usage

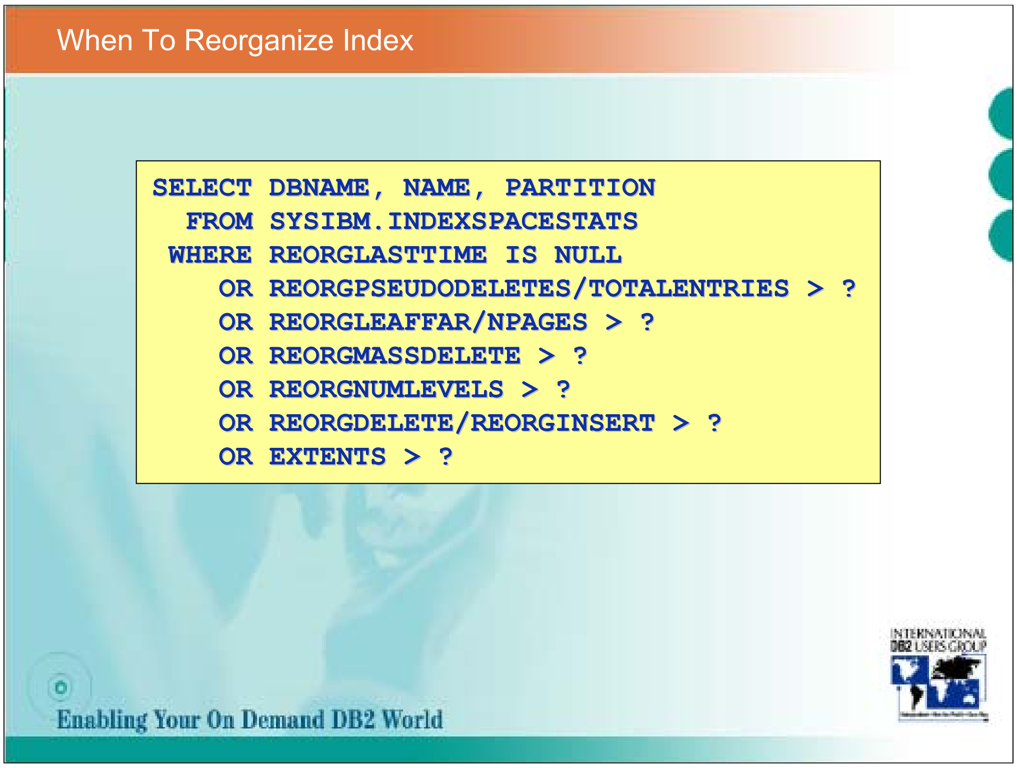
```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGUNCLUSTINS/TOTALROWS > ?
    OR REORGDISORGLOB/TOTALROWS > ?
    OR (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > ?
    OR REORGMASSDELETE > 0
    OR REORGDELETE/REORGINSERT > 1.5
    OR EXTENTS > 50
```

- Reclaim unused space
- Prevent running out of extents
  - adjust allocation quantities to minimize frequent reorganizations
- Improve performance of sequential queries

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

These indicators signal potential space utilization inefficiencies. A reorganization can reclaim the unused space, prevent running out of extents (after adjusting allocation quantities) and improve performance of sequential queries



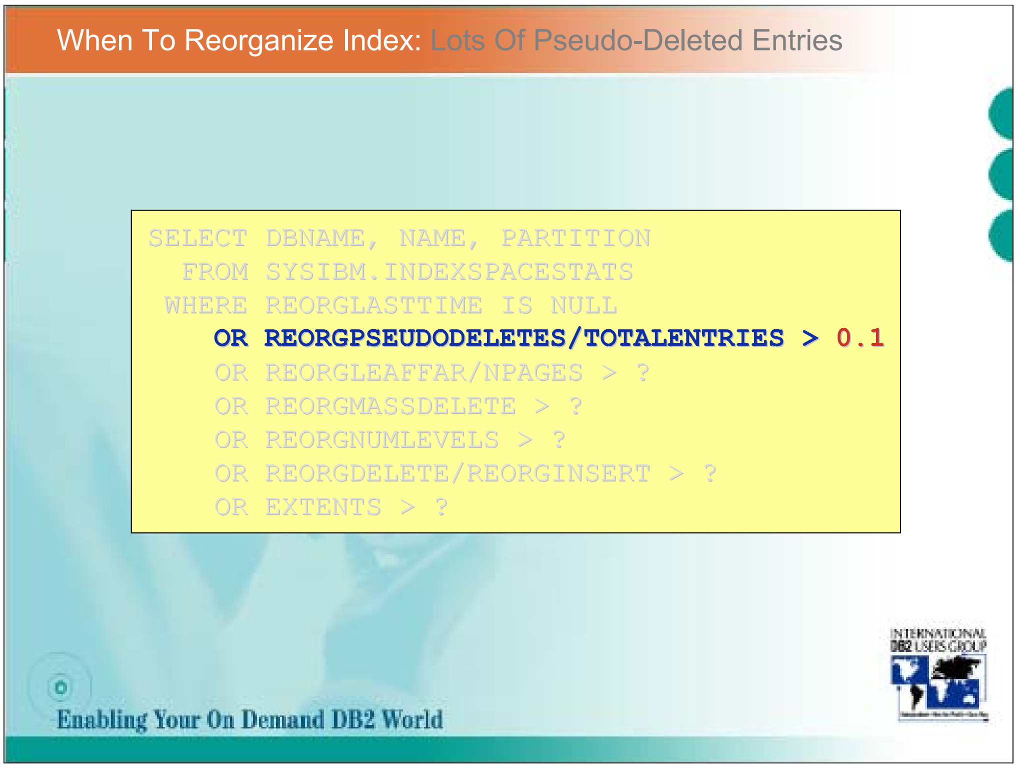
When To Reorganize Index

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGPSEUDODELETES/TOTALENTRIES > ?
    OR REORGLEAFFAR/NPAGES > ?
    OR REORGMASSDELETE > ?
    OR REORGNUMLEVELS > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

Enabling Your On Demand DB2 World
INTERNATIONAL DB2 USERS GROUP

Like in the reorganization of tablespaces case, your thresholds need to be adjusted to the specific workload.

For the REORGLASTTIME, REORGMASSDELETE, REORGDELETE/REORGINSERT and EXTENTS you can use the same thresholds as for the tablespaces.

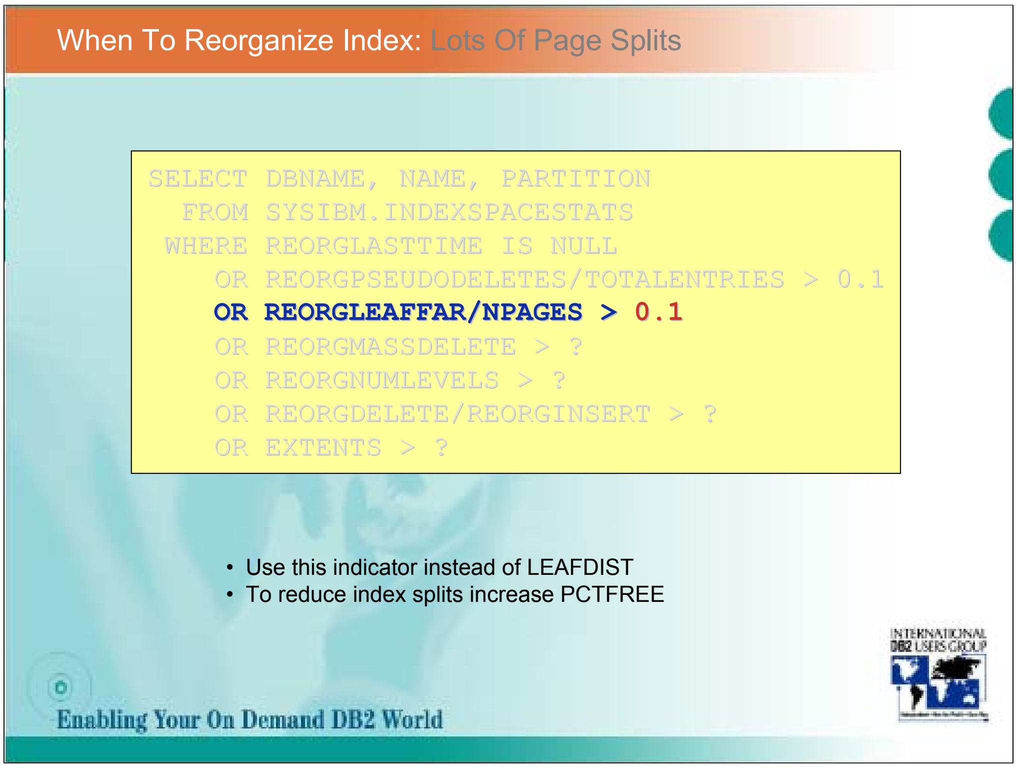


When To Reorganize Index: Lots Of Pseudo-Deleted Entries

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGPSEUDODELETES/TOTALENTRIES > 0.1
    OR REORGLEAFFAR/NPAGES > ?
    OR REORGMASSDELETE > ?
    OR REORGNUMLEVELS > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

Enabling Your On Demand DB2 World
INTERNATIONAL DB2 USERS GROUP

Another nice DB2 V7 addition is ***REORGPSEUDODELETES*** that records the number of pseudo-deleted entries in an index since the last REORG. Such entries inflate the index and if their number is larger than 10% of TOTALENTRIES, the index reorganization is recommended to improve queries that use index scan. Note that the counter is decremented whenever pseudo-deleted entries are removed



When To Reorganize Index: Lots Of Page Splits

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE REORGLASTTIME IS NULL
    OR REORGPSEUDODELETES/TOTALENTRIES > 0.1
    OR REORGLEAFFAR/NPAGES > 0.1
    OR REORGMASSDELETE > ?
    OR REORGNUMLEVELS > ?
    OR REORGDELETE/REORGINSERT > ?
    OR EXTENTS > ?
```

- Use this indicator instead of LEAFDIST
- To reduce index splits increase PCTFREE

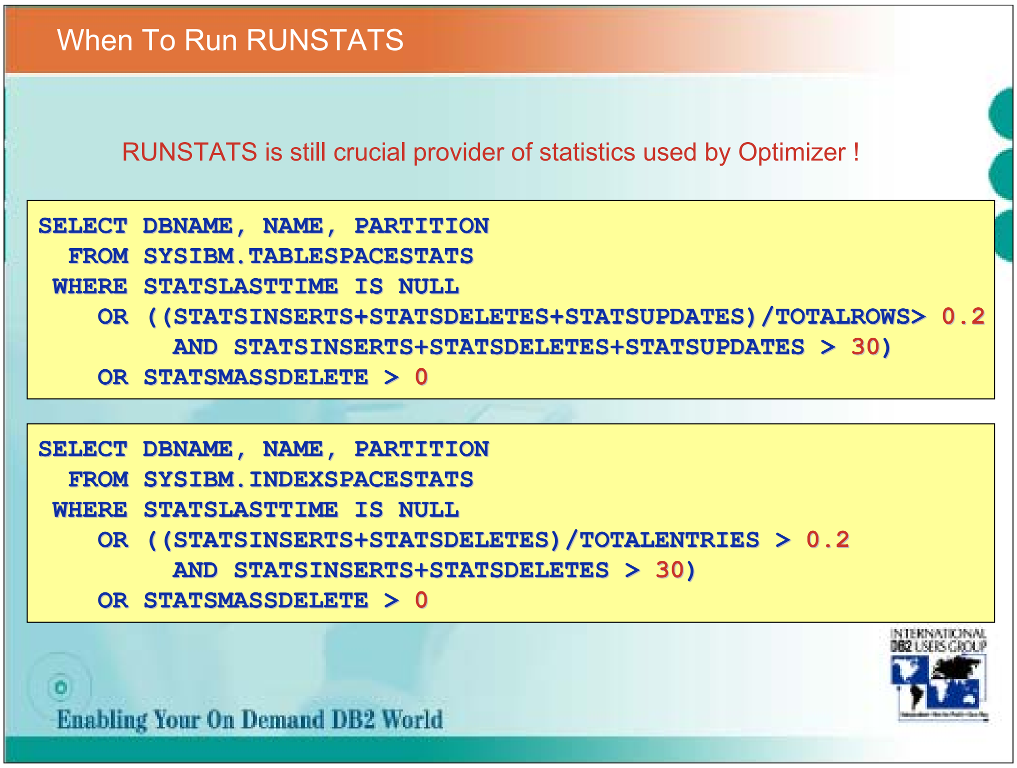
Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP

Prior to Version 7 the only index reorganization indicator was LEAFDIST from SYSINDEXPART that was often misleading and caused premature reorganizations. INDEXSPACESTATS now includes ***REORGLEAFNEAR*** and ***REORGLEAFFAR*** which are much more reliable index reorganization indicators. ***REORGLEAFNEAR*** records the number of index page splits where the resulting pages were located less than 16 pages apart. ***REORGLEAFFAR*** records the number of index page splits where the resulting pages were located more than 16 pages apart. If there is a free page next to the splitting page, none of the counters are incremented. This is why the number of index splits is equal or greater than the sum of the counters.

Index reorganization should be considered if REORGLEAFFAR is more than 10 percent of the number of index active pages (NPAGES).

To reduce the number of index splits, you should reserve more free space by increasing the PCTFREE value. This will result in reduction of synchronous disk I/Os caused by an index scan.



When To Run RUNSTATS

RUNSTATS is still crucial provider of statistics used by Optimizer !

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE STATSLASTTIME IS NULL
    OR ((STATSINSERTS+STATSDELETES+STATSUPDATES)/TOTALROWS> 0.2
          AND STATSINSERTS+STATSDELETES+STATSUPDATES > 30)
    OR STATSMASSDELETE > 0
```

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE STATSLASTTIME IS NULL
    OR ((STATSINSERTS+STATSDELETES)/TOTALENTRIES > 0.2
          AND STATSINSERTS+STATSDELETES > 30)
    OR STATSMASSDELETE > 0
```

Enabling Your On Demand DB2 World

In order to fully appreciate the value of the RTS feature, think of a typical administrative action such as reorganization. Database administrators (DBA) use a set of statistical values collected by the RUNSTATS utility and stored in the DB2 catalog to identify such tablespaces and indexes. Therefore these indicators must be current, i.e. the RUNSTATS utility needs to be performed in a timely manner. But, when is that? There are no indicators that would tell the DBA when the indicators' values are outdated and need to be refreshed. So, the DBAs resort to the only possible approach: they perform the RUNSTATS utility indiscriminately for all the objects in the database and do so in regular, frequent intervals. There are a number of problems with this approach:

•The RUNSTATS utility is not transparent to other database operations. It consumes CPU and acquires locks on the DB2 catalog that are incompatible with concurrent PREPARE and Data Definition Language (DDL) statements. This is especially relevant in large and complex systems such as SAP where a single application component includes more than 37000 tables and 45000 indexes (these numbers are valid for the most recent SAP release). As SAP installations typically consist of many components and each component is implemented in its test, development and production version, the number of objects for which RUNSTATS needs to be performed reaches 100000s.

•Due to its cost and impact to the interactive workload the RUNSTATS jobs are typically scheduled in a batch window, if one is available. But there are other operations that need to be performed at the same time and massive RUNSTATS compete with them which put pressure on the batch window duration.

•Creating, maintaining, scheduling and analysing the RUNSTATS jobs is not an efficient use of the DBA time whose skills are getting increasingly short and expensive.

•Depending on the dynamics of objects changes, no matter how often the RUNSTATS are performed the statistical values can quickly get outdated and hence misleading.

Without using Real Time Statistics RUNSTATS collected data that was necessary not only as the main input to the Optimizer, but also used as indicators when typical DBA operations are due: reorganization, copy, adjusting space allocation. With RTS the DBA aspect of RUNSTATS usage is no longer so important, but providing the fresh data for the Optimizer is equally important as before. So, it is important when the catalog statistics get stale and refresh it in a timely fashion.

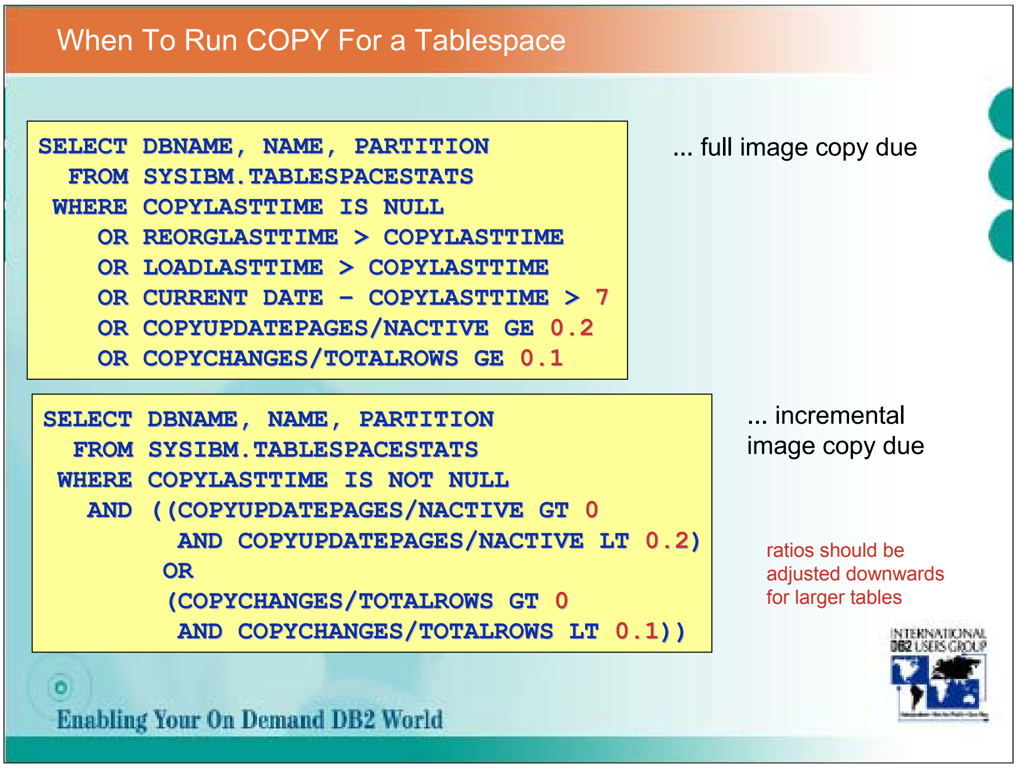
STATSINSERTS, ***STATSDELETES*** and ***STATSUPDATES*** contain the number of rows inserted, deleted and updated since the last time a RUNSTATS utility was executed. These statistics can be used to assess the frequency of changes for a particular object and a need for refreshing catalog statistics. Typically, if a sum of these counters is larger than 20% of the number of rows in the table space, a RUNSTATS utility should be executed.

STATSLASTTIME IS NULL

OR ((STATSINSERTS+STATSDELETES+STATSUPDATES)/TOTALROWS> 20

AND STATSINSERTS+STATSDELETES+STATSUPDATES> 30)

OR STATSMASSDELETE > 0



When To Run COPY For a Tablespace

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE COPYLASTTIME IS NULL
    OR REORGLASTTIME > COPYLASTTIME
    OR LOADLASTTIME > COPYLASTTIME
    OR CURRENT DATE - COPYLASTTIME > 7
    OR COPYUPDATEPAGES/NACTIVE GE 0.2
    OR COPYCHANGES/TOTALROWS GE 0.1
```

... full image copy due

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.TABLESPACESTATS
 WHERE COPYLASTTIME IS NOT NULL
   AND ((COPYUPDATEPAGES/NACTIVE GT 0
         AND COPYUPDATEPAGES/NACTIVE LT 0.2)
        OR
        (COPYCHANGES/TOTALROWS GT 0
         AND COPYCHANGES/TOTALROWS LT 0.1))
```

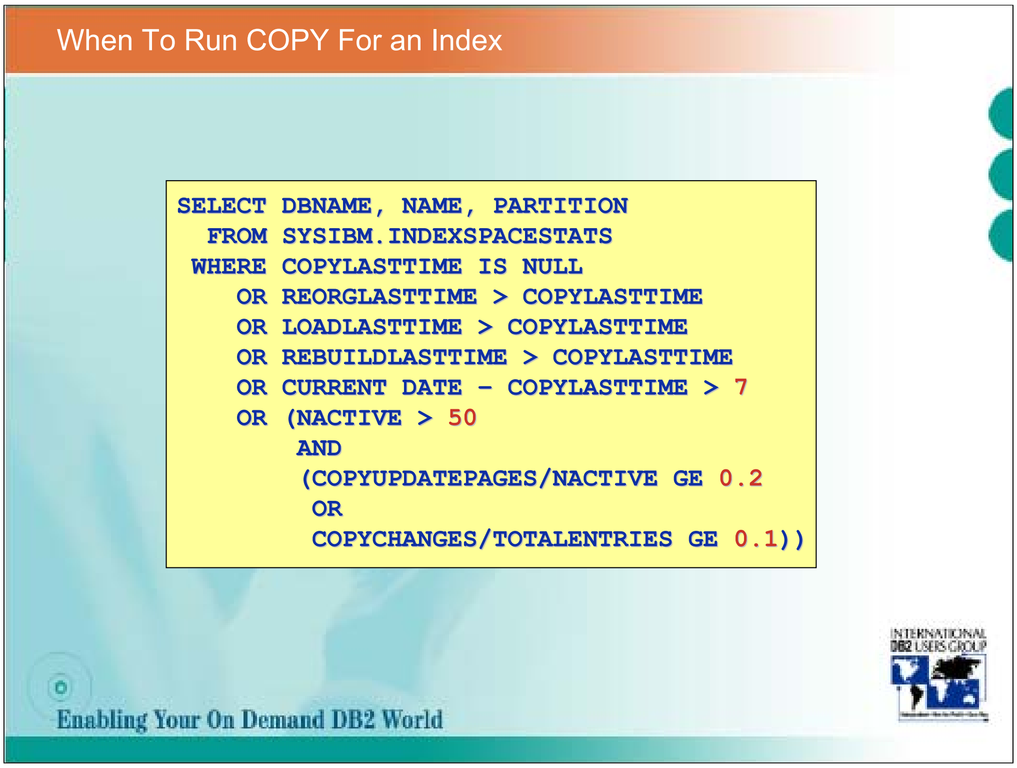
... incremental image copy due

ratios should be adjusted downwards for larger tables

INTERNATIONAL DB2 USERS GROUP
Enabling Your On Demand DB2 World

COPYUPDATEDPAGES and ***COPYCHANGES*** can be used to trigger backing up a particular table space or index. COPYUPDATEDPAGES is the number of distinct pages that have been changed since the last time COPY utility was run. If it is less than 20% (but more than 0%) of the active pages, you might want to schedule an incremental copy. Otherwise, a full copy would be more appropriate. COPYCHANGES is the number of inserts, deletes and updates since the last COPY. In other words, that is the number of log records that would need to be applied in the case of recovery. You might want to keep that number low by taking new image copies and speeding up potential recovery.

If the log records needed for recovery are not in the active logs, the entire recovery process might take longer than wanted. In order to avoid that check the value in ***COPYUPDATELRSN*** that contains the first LRSN or RBA after the last image copy was taken. If that log point is not in the active logs (use Print Log Map utility, DSNJU004 to find out), consider taking a new image copy.



When To Run COPY For an Index

```
SELECT DBNAME, NAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE COPYLASTTIME IS NULL
    OR REORGLASTTIME > COPYLASTTIME
    OR LOADLASTTIME > COPYLASTTIME
    OR REBUILDLASTTIME > COPYLASTTIME
    OR CURRENT DATE - COPYLASTTIME > 7
    OR (NACTIVE > 50
        AND
        (COPYUPDATEPAGES/NACTIVE GE 0.2
         OR
         COPYCHANGES/TOTALENTRIES GE 0.1))
```

Enabling Your On Demand DB2 World

The ratios should be adjusted downwards for larger indexes.

References

- SC26-9945-02 : DB2 UDB for OS/390 and z/OS Version 7 Utilities Guide and Reference
- SG24-6289-00 : Using the Utility Suite
- IDUG Solutions Journal: Volume 10, Number 3

IDUG® 2004 – North America

Thank You For Your Attention !

Getting Most Out Of Real Time Statistics

Session ID: F3

Namik Hrle
IBM Boeblingen Lab
hrle@de.ibm.com

Enabling Your On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP