



(*)



IBM DB2 for z/OS

Realtime-Statistics

(DB2-RTS)

(*) ist eingetragenes Warenzeichen der IBM International Business Machines Inc.

DB2 RTS Inhalte (Überblick)

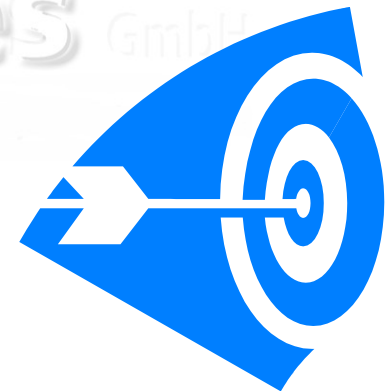
- Zielsetzung der „Real-time statistics“
- Was sind „Real-time Statistics“ (RTS) ?
- Benutzerdefinierte RTS Objekte
- Enable/Disable RTS Prozess
- Welche Statistiken werden über RTS gesammelt ?
- Wie legt DB2 die „in-memory statistics“ offen ?
- Wie beeinflussen SQL und Utilities die „Real-time statistics“ ?
- Was ist DSNACCOR/DSNACCOX ?
- Wie funktioniert DSNACCOR/DSNACCOX?
- Zusammenfassung

RTS Terminologie

CC 390	DB2 Control Center for OS/390 und z/OS
CCMS	SAP's Control Center Management System
CCSID	DB2 Code Page Identifier
CRM	„Customer Relationship Management“ Applikationen
DBA	Database Administrator
DSNACCOR	Eine DB2 Stored Procedure
ERP	Enterprise Resource Planning Applikationen
IFI	DB2's Instrumentation Facility Interface
LOB	Large Object
LRSN	Log Record Sequence Number
NPI	Non-Partitioned Index
QMF	IBM's Query Management Facility Program Product
ROT	„Rule of Thumb“
RTS	Real Time Statistics
SRB	Ein spezeller typ Prozess unter OS/390 und z/OS
WLM	OS/390 bzw. z/OS Workload Manager

RTS Ziele

- Z. Zt. haben DBA und Monitor Programme keine **genauen Daten Objekte, die Maintenance benötigen** zu identifizieren
 - Unnötige Maintenance für Objekte führt zu ineffizienter Nutzung der DBA Zeiten und
 - Verschwendet Zeit im Batch-Fenster
- **DB2 Systeme werden größer und komplexer**
 - Ein einzelnes DB2 besitzt mehr als 40,000 Tables/Indexes für viele ERP/CRM/... Package Anwendungen
 - Server Konsolidierung
 - Gut ausgebildete DBA sind zunehmend rar und teuer
- Ein Ziel ist **DB2 soweit wie möglich „Self-Managed“** zu haben

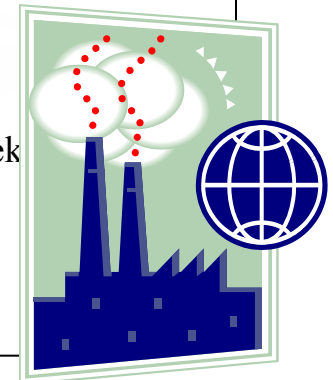


Was sind RTS ?

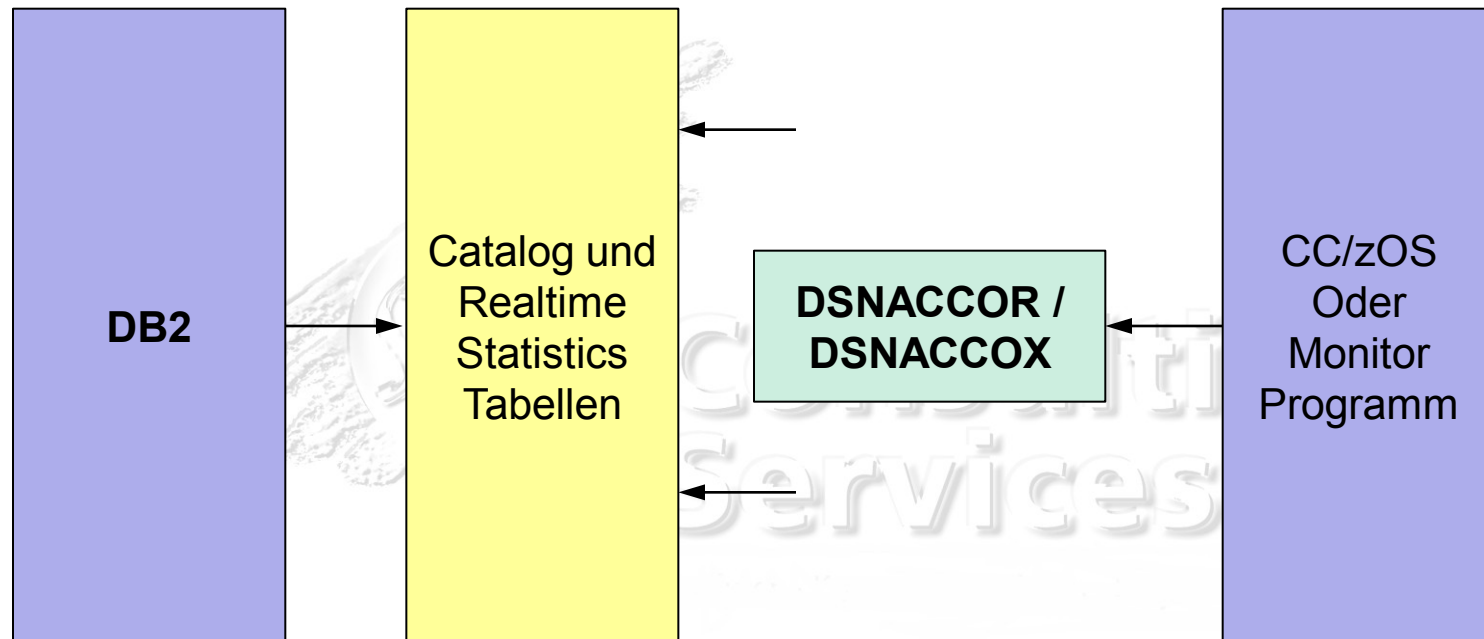
- Sammelt „real-time statistics“ für Tabellen- und Indexspeicher
- Hält Statistiken im Speicher und schreibt diese in bestimmten Zeitabständen auf die RTS Tabellen
 - SYSIBM.SYSTABLESPACESTATS
 - SYSIBM.SYSINDEXSPACESTATS
- Eine DB2 Stored Procedure (DSNACCOR/DSNACCOX) greift auf die RTS Tabellen zu, um die Objekete zu identifizieren, die
 - REORG
 - RUNSTATS
 - IMAGE COPY

benötigen

- DB2 Control Center der andere Monitor Programme (z.B. DB2 Automation Tool, SAP CCMS, ...) nutzen DSNACCOR/DSNACCOX um die DB2 Objek verwalten
- In V7 ausgelieferte APARs: PQ48447,PQ48448 and PQ56256; DSNACCOR wurde im PQ46859 ausgeliefert



RTS Übersicht



- DB2 **RTS Manager** schreibt und modifiziert die Real-time Statistics (timer interval)
- **DSNACCOR/DSNACCOX** (stored procedure) analysiert den DB2 Katalog und die Real-time Statistics

„User defined“ RTS Objekte

Unique Index

SYSIBM.SYSTABLESPACESTATS
DSNRTX01: dbid.psid.partition.instance

Unique Index

SYSIBM.SYSINDEXSPACESTATS
DSNRTX02: dbid.isobid.partition.instance

SYSIBM.SYSTABLESPACESTATS

SYSIBM.SYSINDEXSPACESTATS

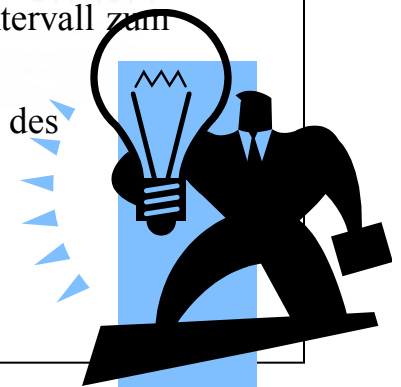
DSNDB06.SYSRTSTS (segmented)

RTS Tables und Indexe

- **SYSIBM.SYSTABLESPACESTATS**
 - Enthält Statistiken für Tablespaces
 - Eine „row“ pro Tablespace/Partition
 - Braucht einen „unique index“ SYSIBM.DSNRTX01
- **SYSIBM.SYSINDEXSPACESTATS**
 - Enthält Statistiken für Indexes (ausgenommen TEMP Indexe)
 - Eine „row“ pro Indexspace/Partition
 - Braucht einen „unique index“ SYSIBM.SYSINDEXSPACESTATS_IX
- Die **Tabellen** müssen sich **in einem „segmented table space“** (SYSRTSTS) befinden
- **SYSIBM.SYSRTSTS** wird **im Katalog in der Datenbank DSNDB06** angelegt
- Database, Tablespace, Table und Index Definitionen werden **AUTOMATISCH** bei Installation der DB2 9 angelegt.

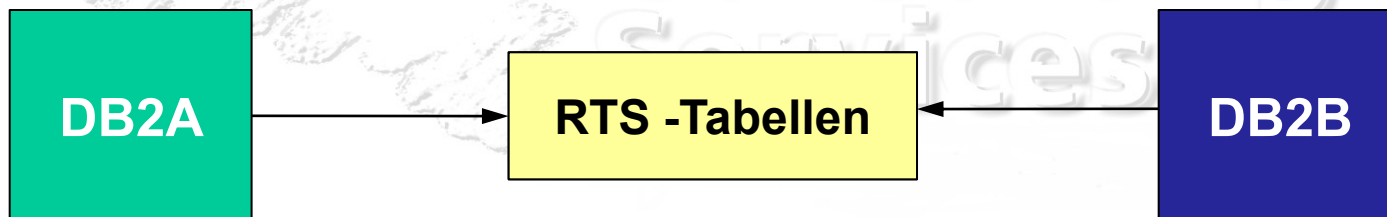
Richtlinien für RTS Objekte

- Die **Größe der RTS Objekte** hängt von der Größe der Speicherobjekte in DB2 ab
 - Eine „row“ für jede Partition oder jeden „non-partitioned table/index-space“
- **RTS Objekte sollten im Speicher gehalten werden** (z.B. eigener Buffer Pool) um die Effizienz der Änderungen an den Statistiken so hoch wie möglich zu halten.
- Statistiken für das Directory von DB2 werden nicht gesammelt.
- Für “partitioned spaces” generiert DB2 Informationen pro Partition. Dennoch muss man das Intervall, für das die Statistiken in die RTS-Tabellen geschrieben werden sollen, selbst festlegen.
- Das **Feld in den ZPARMS von DB2** heisst **REAL TIME STATS** aus dem Panel DSNTIPO.
- In einem “data sharing environment” hat jedes “member” sein eigenes Intervall zum Schreiben der “real-time statistics”.
- Zum Verändern des Intervalls: Ändern der RTS-Statistiken: Modifizieren des Systemparameters STATSINT. Default Intervall ist 30 Minuten.
- Möglich sind: **1 bis 1.440 Minuten**



Enable/Disable RTS

- Viele Spalten in den “real-time statistics” Tabellen zeigen die Häufigkeit an, in der eine Operation zwischen dem letzten mal, an dem ein utility lief und wann die “real-time statistics” geschrieben wurden, an.
- **Für jedes Objekt für das “real-time statistics” gesammelt werden sollen** Sollte das entsprechende Utility gestartet werden (REORG, RUNSTATS, LOAD REPLACE, REBUILD INDEX bzw. COPY), um die Basisdaten aus denen die Delta-Daten kalkuliert werden können, zu erstellen
- Um die “real-time statistics” effizient nutzen zu können, muss man verstehen, **wann DB2 sie sammelt und ausschreibt** und welche Faktoren im System diese Statistiken steuern.



- **Zuweisung von RTS Blöcken** für jedes geänderte Objekt
- **Freigabe von RTS Blöcken**, wenn Pagesets/Partitions geschlossen werden und nachdem die Statistiken auf die RTS Tables geschrieben sind
- **„In-memory statistics“ werden IMMER gesammelt**, auch wenn RTS nicht „enabled“ ist

Ausschreiben der RTS „in-memory“ Statistiken

- **STOP DATABASE(DSNDB06) SPACENAM(<spacename>)** Kommando
 - Löschen der „in-memory statistics“ für alle Zielobjekte und Schreiben nach RTS-Tabellen
 - Für die Datenbank DSBRTSDB erfolgt keine Externalisierung
- **STOP DATABASE(<dbn<me>) SPACNAM(<spacename>)**
 - Löschen aller „in-memory statistics“
 - Die „real time statistics“ Aufzeichnungen von <dbname>.<spacename> werden externalisiert
- **Am Ende des Zeitintervalls, das bei der Installation angegeben wurde** (STATSINT Systemparameter).
- **STOP DB2 MODE(QUIESCE)**
Aber: Bei STOP DB2 MODE(FORCE), DB2 schreibt keine Statistiken. Sie gehen verloren...
- **Eine Utility Operation** (e.g. LOAD, REORG, RUNSTATS, COPY, REBUILD, RECOVER)

Empfehlung: “real-time statistics objects” sollten nicht Bestandteil der “Utility Liste” sein



Tablespace Statistiken

UPDATESTATTIME	Timestamp der Eintragung der Statistik
NACTIVE	Anzahl aktiver (z.B. „preformatted“) Pages im TS bzw. Partition
NPAGES	Anzahl unterschiedlicher pages mit aktiven „rows“ im TS / Partition
EXTENTS	Anzahl „extents“ im TS: der letzte DS für einen gestückelten „simple/segmented“ TS. NULL heist „ubekannt“!
LOADLASTTIME	Timestamp des letzten LOAD REPLACE
REORGLASTTIME	Timestamp des letzten REORG
REORGINSETS	Die Anzahl Sätze/LOBs, die in den TS/Partition mit einem LOAD (ohne REPLACE) oder einem REORG eingefügt wurden
REORGDELETES	s.o.
REORGUPDATES	s.o.
REORGDISORGLOB	für LOBs s.o.
REORGUNCLUSTINS	Anzahl Sätze, die ausserhalb der „clustering page range“ abgelegt sind. Ein Insert gilt als „unclustered“, wenn die „row“ mehr als 16 Pages von der idealen Page entfernt gespeichert wurde. Immer = 0 für einen TS ohne Index
REORGMASDELETE	Anzahl vom Massen-Deletes aus einem segmented bzw. LOB TS
REORGNEARINDREF	# von „overflow records“ nahe dem „pointer“ Satz seit dem letzten REORG „Near“ heisst für TS <16 Pages, für STS < SEGSIZE*2.
REORGFARINDREF	# von „overflow records“ weit entfernt vom „pointer“ Satz seit dem letzten REORG. „far“ heisst für „simple“ TS >16 Pages, für STS > (SEGSIZE*2) +1.

Tablespace Statistiken

STATSLASTTIME	Timestamp des letzten RUNSTATS
STATSINSERTS	wie REORGINSERTS nun für RUNSTATS
STATSDELETES	wie REORGDELETES nun für RUNSTATS
STATSUPDATES	wie REORGUPDATS nun für RUNSTATS
STATSMASSDELETES	wie REORMASSDELETES nun für RUNSTATS
COPYLASTTIME	Timestamp des letzten FC/IC COPY
COPYUPDATEDPAGES	# Pages, die seit dem letzten COPY verändert wurden
COPYCHANGES	# Pages, die über INSERT, UPDATE, DELETE oder LOAD seit dem Letzten COPY verändert wurden
COPYUPDATELRSN	Die LRSN/RBA des ersten UPDAT nach dem letzten COPY
COPYUPDATETIME	Timestamp des ersten UPDATE nach dem letzten COPY
IBMREQD	Y heisst, die „row“ kommt vom MRM Tape
DBID	Database Identifier (DBID Spalte in SYSIBM.SYSTABLESPACE)
PSID	Tablespace Identifier (PSID Spalte in SYSIBM.SYSTABLESPACE)
PARTITION	Partitionnummer, = 0 für „non-partitioned“ TS, ansonsten Partionnummer
INSTANCE	zeigt eine Verbindung zu einer DS Instanz an (1 oder 2)
SPACE	zugewiesener Speicher für TS/Partition in K Bytes
TOTALROWS	Anzahl der „rows“ / LOBs im TS oder der Partition
DATASIZE	# Bytes einer „data row“ in den Datenrows/LOB-Rows
UNCOMPRESSED DATASIZE	# Bytes einer „data row“ in den Daten-/LOB-Rows, wenn die Daten unkomprimiert sind
DBNAME	Name der Datenbank für das „mapping“ einer DB auf ihre Statistiken
NAME	Name des Tablespace (TS) für das „mapping“ einer DB auf seine Statistiken

REORG eines TS ?

- **Umstrukturieren der Daten** („simple TS“ auf STS, auf PTS)
- **Komprimieren** der Daten/Partitionen
- **Phys. Entfernen von „rows“** nach DROP oder einem „mass delete“ auf eine Tabelle
- **Keine Möglichkeit mehr Daten in Tabellen einzufügen**, wegen
 - 255-Extents, Limit von 59 Volumes erreicht
- **Konsolidieren der Extents** wegen Performance für sequentielle Zugriffe
- **Zurückgewinnen und neu Festlegen von freiem Speicher** für das „clustering“
- **„Clustering“** der Daten – d.h. „sequential access“-Verbesserungen
- **Entfernen von „overflow records“**
- **„Fast unload“**

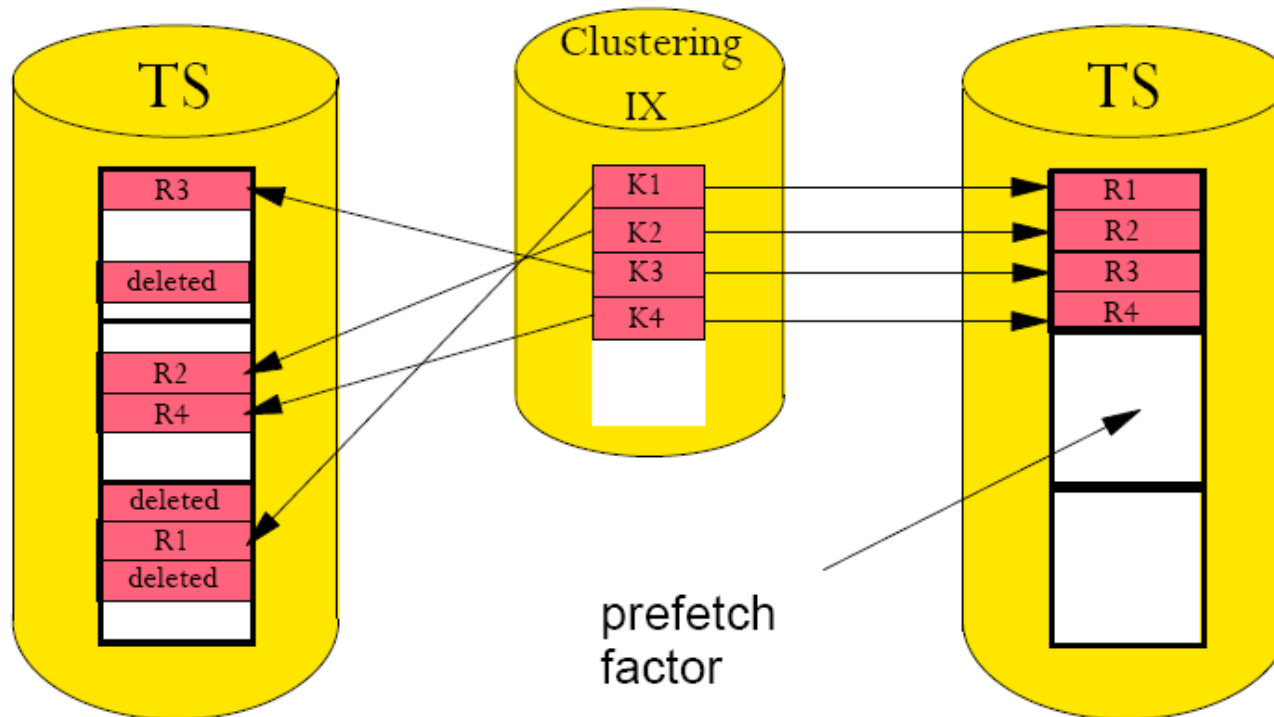


Identifizieren der Objekte für REORG

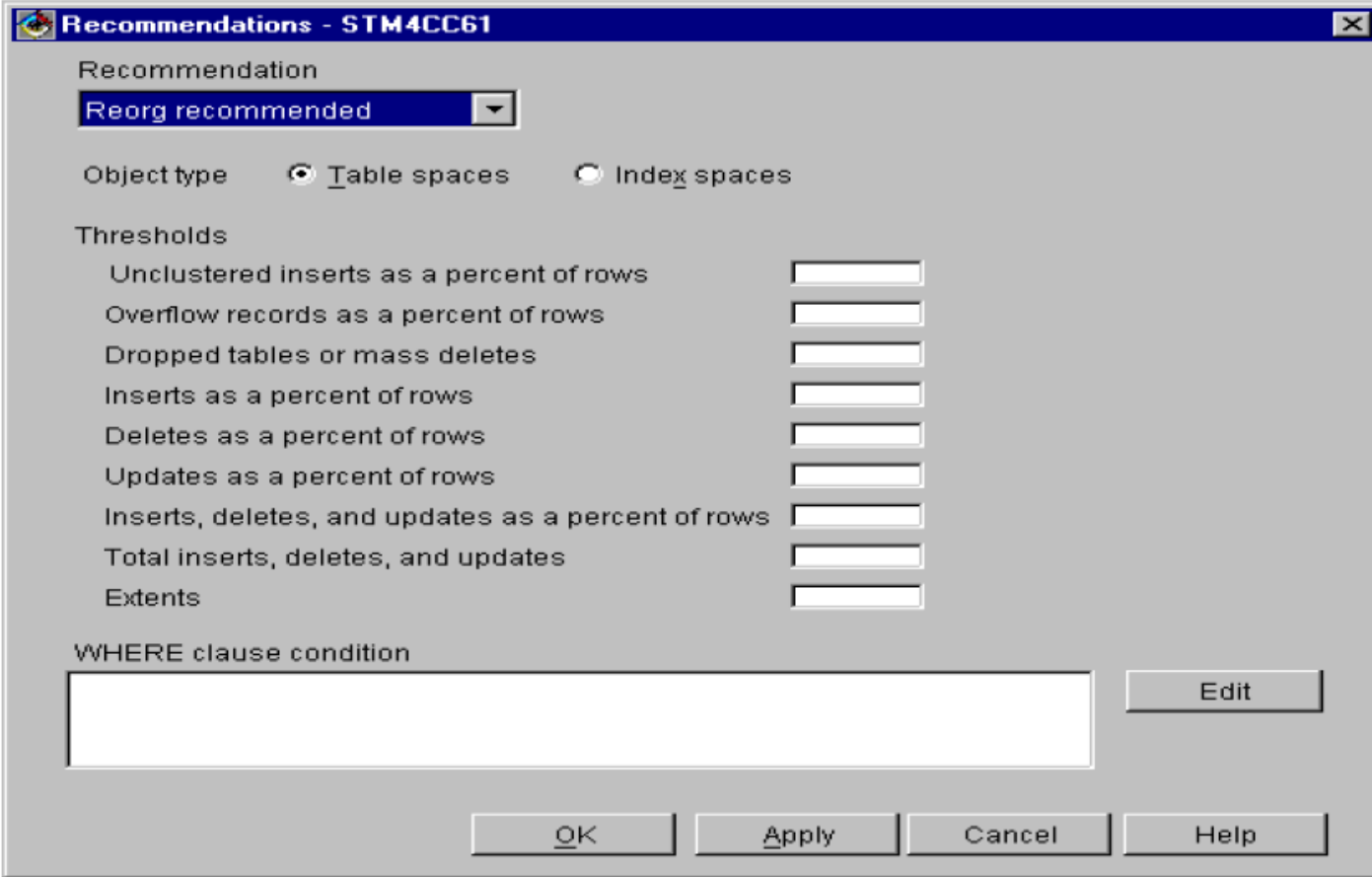
- **Inserts/updates/deletes** Menge an Update Aktivitäten seit dem letzten REORG
- **Uncluster_Inserts** Identifizieren der „rows“, die nicht in der Reihenfolge der „clustering keys“ eingefügt werden konnten
- **Deletes / Mass Deletes / Drops** REORG, um Speicher freizugeben, der von „deleted rows“ besetzt ist
- **NearIndRef and FarIndRef** Anzahl „overflow records“ seit dem letzten REORG
REORG zum Entfernen der „overflows“ und zur Wiederherstellung von „free space“
MAXROWS zur Begrenzung der Maximalzahl von „rows“ innerhalb einer Page
- **Extents** Vermeiden des Erreichens des „dataset extent“ Limits
Konsolidieren der Extents aus Performancegründen
- **Disorg_LOB** Anzahl LOBs, die eingefügt aber nicht richtig gruppiert werden konnten

Optimieren von Daten-Clustern

Uncluster_Inserts – Organisieren der Daten in einer „clustering“ Sequenz



Ein einfaches Panel für REORG Objekte



Recommendations - STM4CC61

Recommendation
Reorg recommended

Object type Table spaces Index spaces

Thresholds

Unclustered inserts as a percent of rows	<input type="text"/>
Overflow records as a percent of rows	<input type="text"/>
Dropped tables or mass deletes	<input type="text"/>
Inserts as a percent of rows	<input type="text"/>
Deletes as a percent of rows	<input type="text"/>
Updates as a percent of rows	<input type="text"/>
Inserts, deletes, and updates as a percent of rows	<input type="text"/>
Total inserts, deletes, and updates	<input type="text"/>
Extents	<input type="text"/>

WHERE clause condition

Edit

OK Apply Cancel Help

Identifizieren der Objekte für RUNSTATS / COPY

- Die **ZugriffspfadAuswahl** einer Query basiert auf den Katalog Statistiken, die vom RUNSTATS Utility zur Verfügung gestellt werden
- Die **Query Performance** hängt von der Genauigkeit der Tabellen- und Indexstatistiken aus dem Katalog ab
- **Folgende „real time statistics“** identifizieren Objekte, die RUNSTATS benötigen:
 - **Inserts/updates/deletes** seit dem letzten RUNSTATS
 - **Mass Deletes/Drops** seit dem letzten RUNSTATS
- Man braucht eine Methode, um festzustellen, ob ein **weiterer COPY** die Zeit eines „data recovery“ reduzieren könnte
 - **Anzahl inserts/updates/deletes**, die erfolgen müssen, wenn Daten aus einem „image copy“ übernommen wurden
 - **Vermeiden von Zugriffen auf „archive logs“**
- Nutzen Sie **folgende „real time statistics“** zur Identifikation von Objekten für COPY
 - **CopyChanges** Anzahl von inserts/updates/deletes, die erneut durchgeführt werden müssten, wenn Daten aus dem letzten IC zurückkopiert werden
 - **Distinct Updated Pages** Anzahl Pages, die während der Zuführung des Log Prozesses zusätzlich verändert werden müssten
 - **UpdateLRSN / Update timestamp** Identifizieren des Zusatzes aus dem Log Startpunkt nach dem Rücksichern der Daten aus dem letzten IC

Indexspace Statistiken

UPDATESTATTIME	Timestamp der Eintragung der Statistik
NLEVELS	Anzahl EBENEN im Index-Baum
NPAGES	Anzahl unterschiedlicher pages mit aktiven „rows“ in der zugehörigen Tabelle
NLEAF	# „leaf-pages“ im Index
NACTIVE	# aktiver Pages im IX-Space/Partiuon (= Anzahl „preformatted“ Pages)
SPACE	Speicherplatz in KB, der dem IX/Partition zugewiesen wurde
EXTENTS	Anzahl „extents“ im TS: der letzte DS für einen gestückelten „simple/segmented“ TS. NULL heist „ubekannt“!
LOADRLASTTIME	Timestamp des letzten LOAD REPLACE
REBUILDLASTTIME	Timestamp des letzten REBUILD
REORGLASTTIME	Timestamp des letzten REORG INDEX
REORGINSERTS	Die Anzahl Einträge in den IX/Partition mit einem LOAD REPLACE oder einem REORG, bzw. einem REBUILD INDEX eingefügt wurden
REORGDELETES	s.o.
REORGAPPENDINSERT	# der getätigten Indexeinträge, deren Schlüssel grösser ist, als der maximale „key“-Wert; seit dem letzten REORG, LOAD REPLACE oder REBUILD
REORGPSEUDODELETES	# IX-Einträge, die seit dem letzten REORG, REBUILD oder LOAD REPLACE „pseudo-deleted“ wurden
REORGMASSEDELETE	Anzahl vom Massen-Deletes aus einem segmented bzw. LOB TS seit dem letzten REORG/LOAD REPLACE
REORGLEAFNEAR	# von „leaf pages“ nahe den vorausgehenden Pages seit dem letzten REORG, REBUILD, LOAD REPLACE. „Near“ heisst für „leaf pages“ die Differenz ist 1 (optimal), mindestens aber 2-16.

Indexspace Statistiken

REORGLAFFAR	# von „leaf pages“ entfernt zu den vorausgehenden Pages seit dem letzten REORG, REBUILD, LOAD REPLACE. „Far“ heisst für „leaf pages“ die Differenz ist mindestens > 16..
REORGNUMLEVELS	# Levels, die zum/aus dem IX-Baum seit dem letzten REORG, REBUILD, LOAD REPLACE hinzugefügt/entfernt wurden
STATSLASTTIME	Timestamp des letzten RUNSTATS
STATSINSERTS	wie REORGINSERTS nun für RUNSTATS
STATSDELETES	wie REORGDELETES nun für RUNSTATS
STATSMASSDELETES	wie REORGMASSDELETES nun für RUNSTATS
COPYLASTTIME	Timestamp des letzten FC/IC COPY
COPYUPDATEDPAGES	# Pages, die seit dem letzten COPY verändert wurden
COPYCHANGES	# Pages, die über INSERT, UPDATE, DELETE oder LOAD seit dem letzten COPY verändert wurden
COPYUPDATELRSN	Die LRSN/RBA des ersten UPDAT nach dem letzten COPY
COPYUPDATETIME	Timestamp des ersten UPDATE nach dem letzten COPY
IBMREQD	Y heisst, die „row“ kommt vom MRM Tape
DBID	Database Identifier (DBID Spalte in SYSIBM.SYSINDEXES)
ISOBID	Interner Identifikator des IX Page Set Descriptors
PSID	Tablespace Identifier (PSID Spalte in SYSIBM.SYSINDEXES)
PARTITION	Partitionnummer, = 0 für „non-partitioned“ TS, ansonsten Partionnummer
INSTANCE	zeigt eine Verbindung zu einer DS Instanz an (1 oder 2)

Indexspace Statistiken

TOTALENTRIES	Anzahl der Einträge im IX/Partition, inkl. Duplikate
DBNAME	Name der Datenbank
NAME	Name des IX
CREATOR	Schema des INDEX
INDEXSPACE	Name des IX-Space
LASTUSED	Datum, wann der IX zum letzten Mal für SELECT, FETCH, „searched UPDATE“ / „searched DELETE genutzt wurde, bzw. in RI Constraint-Prüfungen involviert war. Der „default „ ist 1/1/0001

DB2 Utilities nutzen ebenfalls die RTS Statistiken

Warum REORG eines Indexspace ?

- **Entfernen von „keys“** nach einem „mass delete“ auf den Index
- **Keine Möglichkeit mehr „keys“ in den Index einzufügen**, wegen
 - 255-Extents, Limit von 59 Volumes erreicht
- **Zurückgewinnung von freiem Speicher**
- **neu Festlegen von freiem Speicher** zur Reduktion von „index splits“
- **neu Festlegen von freien Pages** zur Reduktion der Distanz zwischen den „index split“-Pages (wegen Vorteilen beim sequentiellen Zugriff)
- **„Clustering“** der Daten – d.h. „sequential access“-Verbesserungen
- **Entfernen von „pseudo-deleted keys“**
- **Optimieren der Indexstruktur:** „rebalancing“ der „keys“ in „leaf“ und „non-leaf“ Pages
- **Konsolidieren der Extents** für bessere sequentielle Zugriffperformance

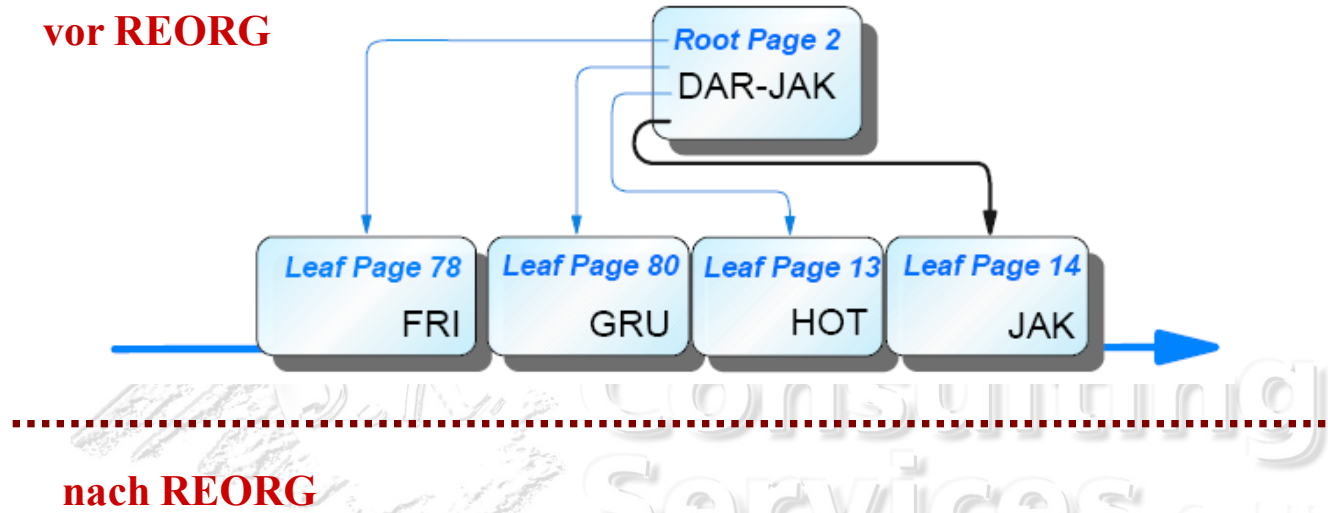


Identifizieren der IX-Objekte für REORG

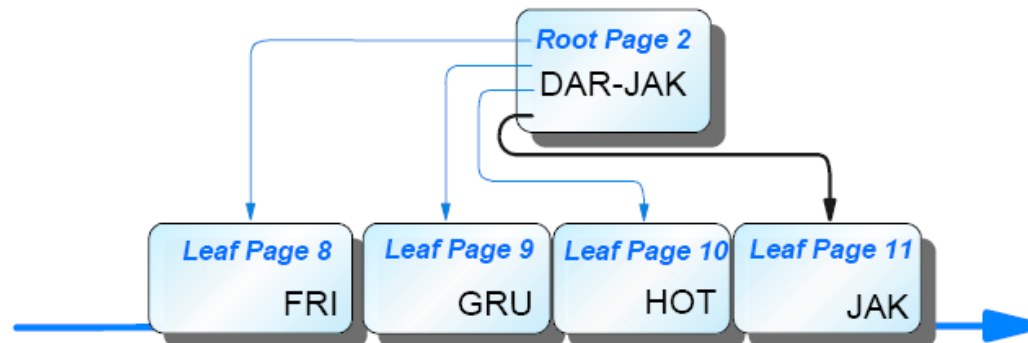
- **Inserts/updates/deletes** Menge an Änderungsaktivitäten seit dem letzten REORG
- **Pseudo / Mass Deletes** REORG, um Speicher freizugeben, der von „deleted keys“ besetzt ist
- **Leaf_Near / Leaf_Far** Gibt Auskunft über die Effizienz von Index-Scans
Mehr freie Pages zu reservieren, hilft, die Leaf_Far-Werte zu reduzieren
- **NLevels / ReorgNumLevels** REORG kann helfen die Ebenen des Index-Baums zu reduzieren
- **Append_Inserts** die Anzahl „keys“, die am Ende des Index einhefügt wurden
Es gibt keinen Grund, „free space“ bzw. „free pages“ zu reservieren, wenn alle Inserts Append_Inserts sind
- **Extents** Vermeiden des Erreichens des „dataset extent“ Limits
Konsolidieren der Extents aus Performancegründen

IX-Scan: Leaf_Near & Leaf_Far

vor REORG



nach REORG



Wann werden Statistiken für die DB2-Objekte gesammelt?

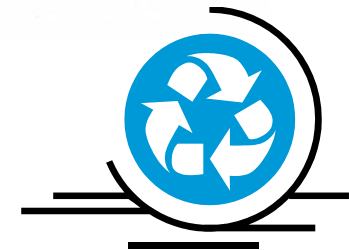
- **Bei neu erzeugten Tablespaces und Indexes**
 - Rows werden in die RTS Tables zum CREATE-Zeitpunkt eingefügt
 - Last_Load_Replace wird auf den CREATE Timestamp gesetzt
 - Last_REORG/STATS/COPY wird auf NULL gesetzt
- **Tablespaces und Indexes existieren schon bevor RTS eingeschaltet wird**
 - Rows werden geschrieben, wenn die **Objekte zum erstenmal verändert** werden
 - o Zum nächsten STATSINT Timer Intervall
 - o Alle Statistikwerte werden auf NULL gesetzt (außer Nactive, Space, Extent)
 - o Last_REORG/STATS/COPY/Load_Replace werden auf NULL gesetzt
 - o Statistics values will be set after the first REORG, RUNSTATS, or COPY
 - Es gibt **keine RTS „rows“ für „Read Only“** Objekte

Wie beeinflusst SQL die RTS?

- **CREATE/DROP TABLESPACE** Insert/delete einer „row“ in SYSIBM.TABLESPACESTATS
- **Insert** Erhöht für Spalten Inserts, TotalROWS, die Copy Changes die Zähler kann Nactive, Space, Extents, Uncluster_Inserts, Distinct Updated Pages, Update LRSN, Update Timestamp ändern
- **Update** Erhöht die Updates, Copy Changes Zähler kann NearIndRef/FarIndRef, Nactive, Space, Extents für VARCHAR verändern kann Distinct Updated Pages, Update LRSN und Timestamp ändern
- **Delete** Erhöht die Deletes und Copy Changes Zähler
- **Delete ohne WHERE Klausel oder DROP TABLE auf „Segmented Tablespaces“** Erhöht den Mass Deletes/Drops Zähler
- **Rollback**
 - bei Insert erhöht den Deletes Zähler
 - bei Delete erhöht den Inserts Zähler
 - bei Update erhöht den Updates ZählerMass Delete/Drop Table zählt den Mass Deletes/Drops Zähler nicht nach unten
- Statistikzähler werden sich verändert **während eines DB2 Restart**
- **Triggers** können einen Update der Statistiken für andere Tabellen verursachen

Wie beeinflusst SQL die RTS?

- **CREATE/DROP INDEX** Insert/delete einer „row“ in SYSIBM.INDEXSPACESTATS
- **Insert** Erhöht für Spalten Inserts, TotalEntries die Zähler kann Append_Inserts, LeafNear, LeafFar, ReorgNumLevels, Nactive, Space, Extents ändern
- **Delete** Erhöht die Deletes Zähler kann PseudoDeletes und ReorgNumLevels ändern
- **COPY YES indexes (Insert/Delete)** verwaltet Copy Changes, Distinct Updated Pages, Update LRSN, Update Timestamp
- **Delete ohne WHERE Klausel oder DROP TABLE auf „Segmented Tablespaces“** Erhöht den Mass Deletes/Drops Zähler
- **Rollback / Restart** wie bei den TS Statistiken



Wie beeinflussen Utilities die RTS?

- **REORG**
 - Setzt Last_REORG_Timestamp
 - Setzt die REORG bezogenen Statistiken zurück
 - Die „Log apply“ Änderungen für den „online REORG werden wie Inserts/Deletes/Updates behandelt
- **RUNSTATS**
 - Setzt Last_RUNSTATS_Timestamp
 - setzt die RUNSTATS bezogenen Statistiken zurück
- **COPY**
 - Setzt Last_COPY_Timestamp
 - setzt die COPY bezogenen Statistiken zurück
- **LOAD REPLACE**
 - Setzt Last_Load_Replace Timestamp
 - Setzt die REORG bezogenen Statistiken zurück

Wie beeinflussen Utilities die RTS?

- **REORG/LOAD REPLACE PART**
 - Setzt die REORG Statistiken für „non-partitioned“ Indexes nicht zurück
 - Statistiken für NPIs werden „updated“ wie INSERT/ DELETE
- **COPY mit der DSNUM(n) Option** auf einem „Linearen“ Tablespace
 - Setzt den Last_Copy_Timestamp nicht zurück
 - setzt COPY bezogene Statistiken nicht zurück
- **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**
 - Setzt Last_Rebuild_Index_Timestamp
 - Setzt REORG bezogene Statistiken zurück
- **Online LOAD Resume**
 - Behandelt wie Inserts

Welche RTS Infos nutzen die Utilities?

- **DB2 utilities nutzen die “real-time statistics”** zum Optimieren der Verarbeitung und ihrer Funktionen.
 - Zur Kalkulation WIE die Verarbeitung stattfinden soll (effizienter als RUNSTATS Statistiken oder Statistiken aus dem Katalog)
 - Eliminiert einige Abhängigkeiten mit dem Ablauf von RUNSTATS, der verarbeitungsintensiv und zeitaufwändig sein kann
- **Sind die “real-time statistics” verfügbar** und der Systemparameter UTSORTAL ist auf YES gestellt, so helfen sie folgenden Utilities:
 - CHECK DATA / CHECK INDEX
 - REBUILD INDEX
 - REORG TABLESPACE
 - RUNSTATS
- DB2 meldet DSNU3343I, wenn keine “real-time statistics” verfügbar sind (IX oder TS).
- DB2 versucht nun “real time statistic” zu sammeln ... wenn das geht
- Sind keine “real-time statistics” verfügbar, greift DB2 auf RUNSTATS Schätzungen zurück

Welche RTS Infos nutzen die Utilities?

Tablespace und Index Eigenschaften

- Utilities sammeln normalerweise Informationen über die IX /TS Charakteristiken. Diese Information wird zur Kalkulation von Statistiken genutzt, die helfen das Verarbeiten der Daten für die Utilities effizienter zu gestalten
- Utilities lesen die Spalte TOTALROWS in der Tabelle SYSIBM.SYSTABLESPACESTATS und die Anzahl zugehöriger “index keys” aus TOTALENTRIES in der Tabelle SYSIBM.SYSINDEXSPACESTATS.
- Diese Statistiken werden verwendet, um die Anzahl von Sätzen für SORTs und die Größe der Work-Datasets zu kalkulieren.

Empfehlung:

Um die Utilities nicht mit inkorrekten Daten arbeiten zu lassen, wenn die TS von Utilities, wie DSN1COPY oder anderen Utilities, die nicht von DB2 Kontrolliert werden können, ersetzt werden, kann die entsprechende Spalteninformation auf NULL gesetzt werden.

Ist die entsprechende Spalteninformation NULL, so wird die Anzahl von Sätzen auf der Basis der RUNSTATS Statistiken geschätzt.

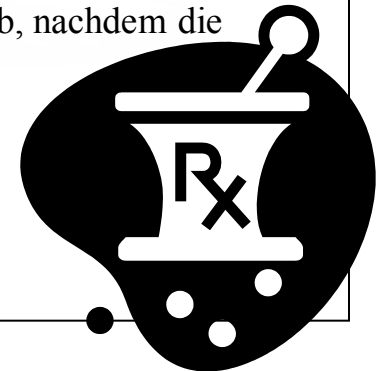
Die Spalten werden neu initialisiert, sobald der nächste REORG TABLESPACE, LOAD REPLACE, oder REBUILD INDEX abläuft.

Genauigkeit der RTS Statistiken

- Immer über das „**timer interval**“ verzögert
 - Gesteuert über ZPARM STATSINT (default 30 Minuten)
- **Verlust aller „in-memory“ Statistiken**, wenn DB2 abstürzt oder bei STOP DB2 MODE(FORCE)
- Statistiken können nicht aufgezeichnet werden, wenn die **DSNRTSDB gestopped oder die „statistics tables“ nicht verfügbar** sind
- Man muß einen **REORG, RUNSTATS, COPY** durchführen, um einen Bezugspunkt zu erhalten
- Manche Statistikwerte sind für ein **Utility Restart Scenario** ungültig (gekennzeichnet mit NULL)
- Statistiken können ungenau sein, wenn **Utilities von Drittanbietern** laufen, ohne die „in-memory statistics“ zurückzusetzen
- Nur die „**physical space statistics**“ (z.B. Nactives, Space, Extents) werden für die DSNDB07 und die TEMP Databases verwaltet

Richtlinien zum SQL/Utility Zugriff auf RTS Statistiken

- **Vermeiden Sie „Timeouts“ / „Deadlocks“** auf dem RTS Manager
 - Nutzen Sie „Uncommitted Read“ Lock Isolation beim zugriff auf RTS Tables
 - Nutzen Sie SHRLEVEL CHANGE für die Utilities REORG, RUNSTATS, COPY auf die RTS Objekte
- **Mischen Sie nie RTS Objekte** mit anderen ser Objekten in einer Utility Liste
 - Wenn, dann werden die RTS Statistiken für alle Objekte in dieser Liste nicht zurückgesetzt
- Bei **Disaster Recovery**
 - Das Recover der RTS Objekte sollte nach dem Recover der DB2 Katalog- und „directory“ erfolgen
 - Setzen Sie explizit einen START DATABASE(DSNRTSDB) ab, nachdem die RTS Objekte „recovered“ sind



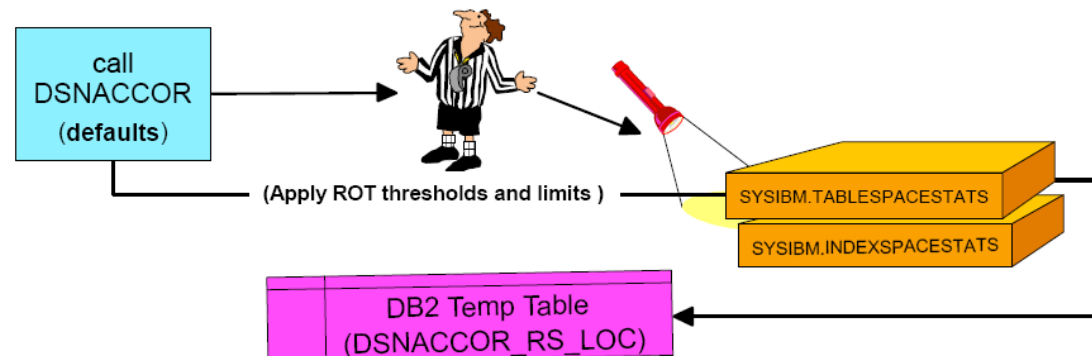
Was ist DSNACCOR/DSNACCOX?

- **Eine DB2 Stored Procedure**, die auf die RTS Tables zugreift
- Und **IFI Calls** ausführt
 - Um den **-DISPLAY** Status auf den DB2 Objekten zu erhalten
- **Wichtigste Zielsetzung**
 - Empfehlung für jedes DB2 Objekt, das entweder: REORG, RUNSTATS, IMAGE COPY benötigt

Wie arbeitet DSNACCOR/DSNACCOX ?

DSNACCOR/DSNACCOX fragt **default-gesteuert beide** RTS Tables ab

- Angewendet werden Algorithmen, die die ROT Schwellwerte und die Limits aus den Objekten in den Tables anwenden
- Liefert Info über alle Objekte, die REORG, RUNSTATS oder COPY benötigen



Zusammenfassung

- Nutzen Sie die **REAL TIME** Statistiken
- **Die REAL TIME** Statistiken helfen der DBA oder Monitor Programmen Objekte, die mit „database maintenance“ Utilities behandelt werden müssen, zu identifizieren
 - Effiziente Nutzung der DBA-Zeiten
 - Verbesserung der System und Anwendungsperformance
 - Reduzieren des Batch-Fensters
- **DSNACCOR** stellt einen einfachen Weg zur Identifikation der DB2 Objekte dar, die
 - REORG/RUNSTATS/COPY Utilities benötigen
 - Sich in „restricted access states“ (z.B. COPY pending, ...) befinden
- „**Real-time statistics**“ bieten eine Basis für DB2 sich zukünftig in die Richtung eines „self-managed“ DBMS zu bewegen
- Eine **zukünftige Nutzung der RTS** durch den DB2 Query Optimizer und DB2 Utilities scheint möglich

