Exciting Optimizer and SQL Performance Enhancements in DB2 9 for z/OS and Beyond

James Guo
IBM Silicon Valley Lab

August 3, 2010 3 pm – 4 pm
Session Number 7966

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

Agenda

- Plan Stability
- Indexing Enhancements
- General Query Performance Enhancements
- Histogram Statistics
- Generalized sparse index and in-memory data cache
- REOPT AUTO
- V10 Query Performance Enhancements

Plan Stability

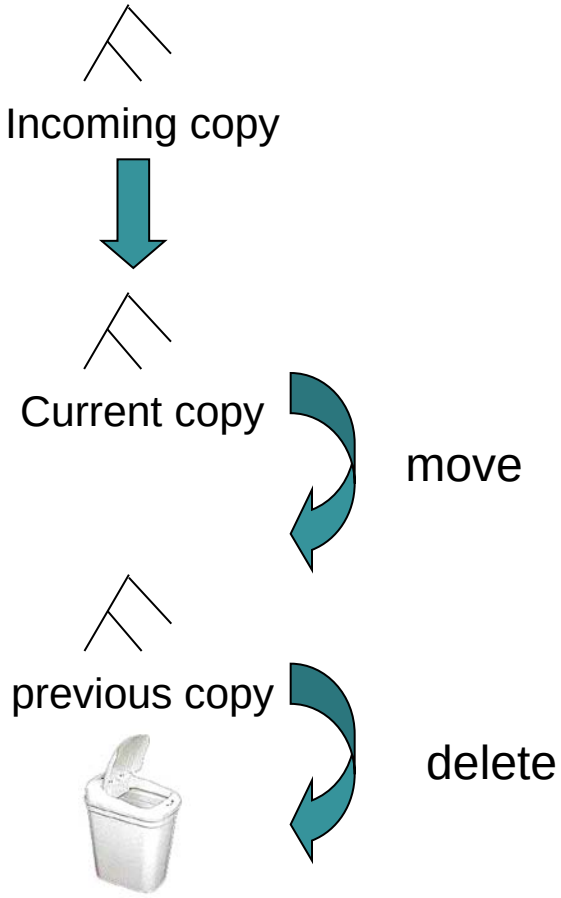
Plan Stability Overview

- Ability to backup your static SQL packages

- At REBIND
  - Save old copies of packages in Catalog/Directory
  - Switch back to previous or original version
- Two flavors
  - BASIC
    - 2 copies: Current and Previous
  - EXTENDED
    - 3 copies: Current, Previous, Original
  - Default controlled by a ZPARM
  - Also supported as REBIND options

Plan Stability - BASIC support

REBIND … PLANMGMT(BASIC)



REBIND … SWITCH(PREVIOUS)

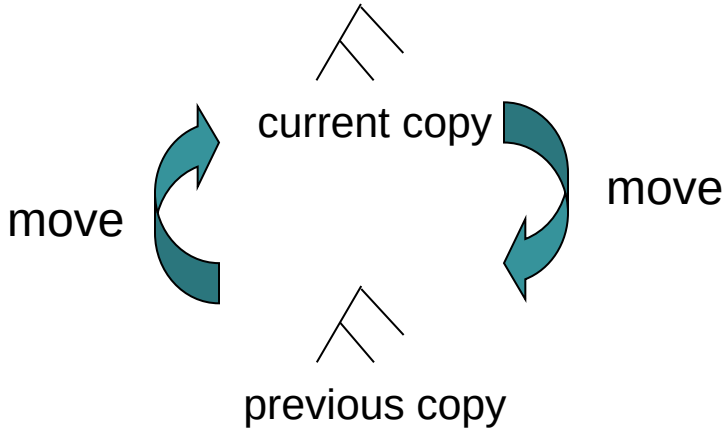
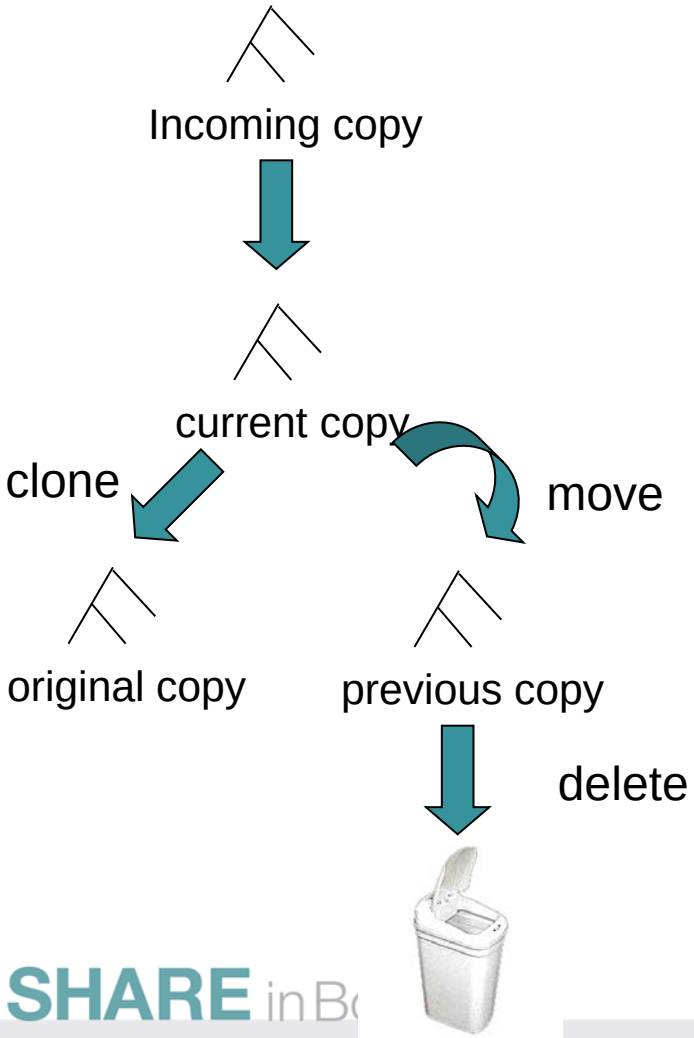


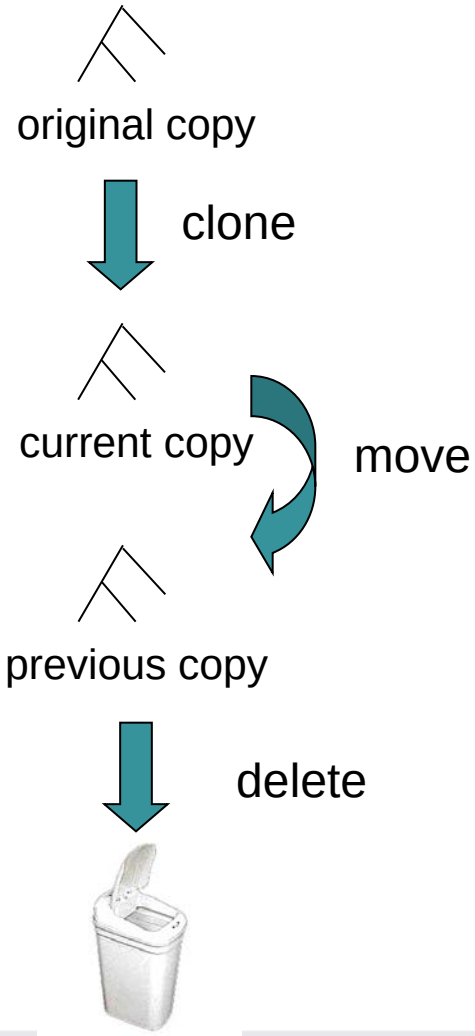
Chart is to be read from bottom to top

Plan Stability - EXTENDED support

REBIND … PLANMGMT(EXTENDED)



REBIND … SWITCH(ORIGINAL)



Access Plan Stability Notes

- REBIND PACKAGE …
  - PLANMGMT (BASIC)

  2 copies: Current and Previous
  - PLANMGMT (EXTENDED)

  3 copies: Current, Previous, Original

- REBIND PACKAGE …
  - SWITCH(PREVIOUS)

  Switch between current & previous
  - SWITCH(ORIGINAL)

  Switch between current & original

- Most bind options can be changed at REBIND
  - *But a few must be the same …*

- FREE PACKAGE …
  - PLANMGMTSCOPE(ALL) – Free package completely
  - PLANMGMTSCOPE(INACTIVE) – Free old copies

- Catalog support
  - SYSPACKAGE reflects active copy
  - SYSPACKDEP reflects dependencies of all copies
  - Other catalogs (SYSPKSYSTEM, …) reflect metadata for all copies

- Invalidation and Auto Bind
  - Each copy invalidated separately

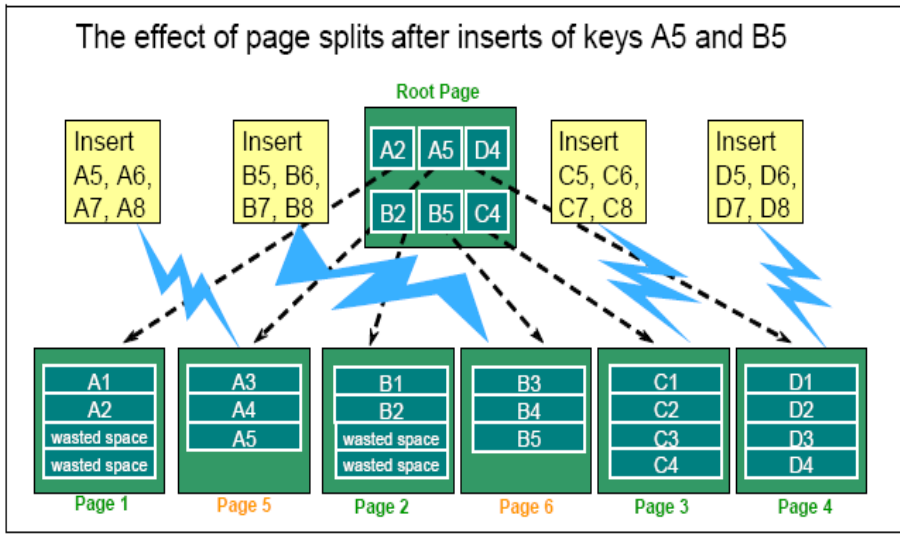
Indexing Enhancements

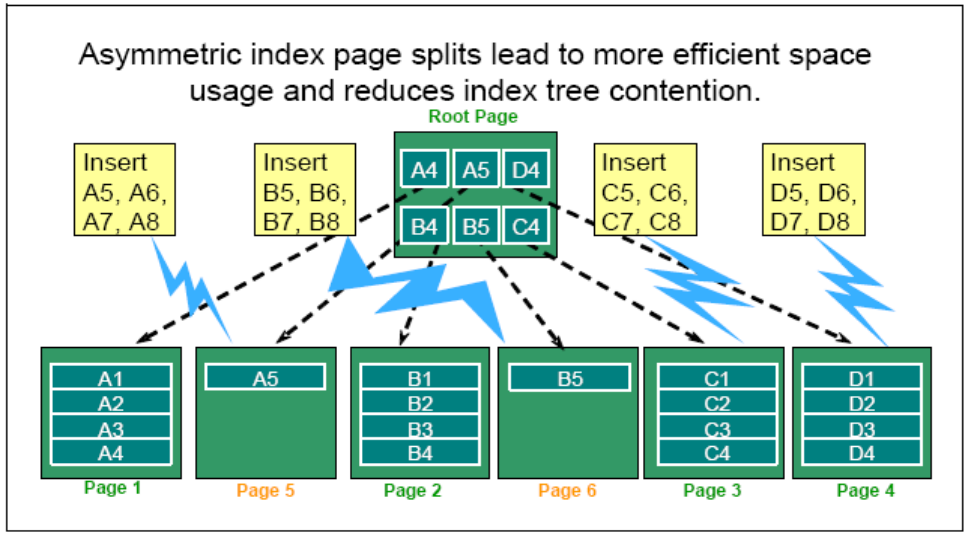
Insert/Update/Delete Performance

- DB2 9 addresses several traditional problem areas for high bandwidth INSERT/UPDATE/DELETE workloads.
  - Log Latch Contention (LC 19) and LRSN Spin (NFM & DS)
  - IX Leaf Page Split Overhead
  - Free Space Search Overhead
  - IX and DATA hot spots
- Table Space APPEND Option (can ALTER on and off)
- Not Logged Tablespaces
- Asymmetric Leaf Page Split
- Randomized Index Key
- Larger Index Page Sizes
- Increased Index Look-aside

- Up to 2x increased logging rate
- 10x reduction in LC19 waits
- Adjust LOGBUFF accordingly

Asymmetric Index Page Split (NFM)



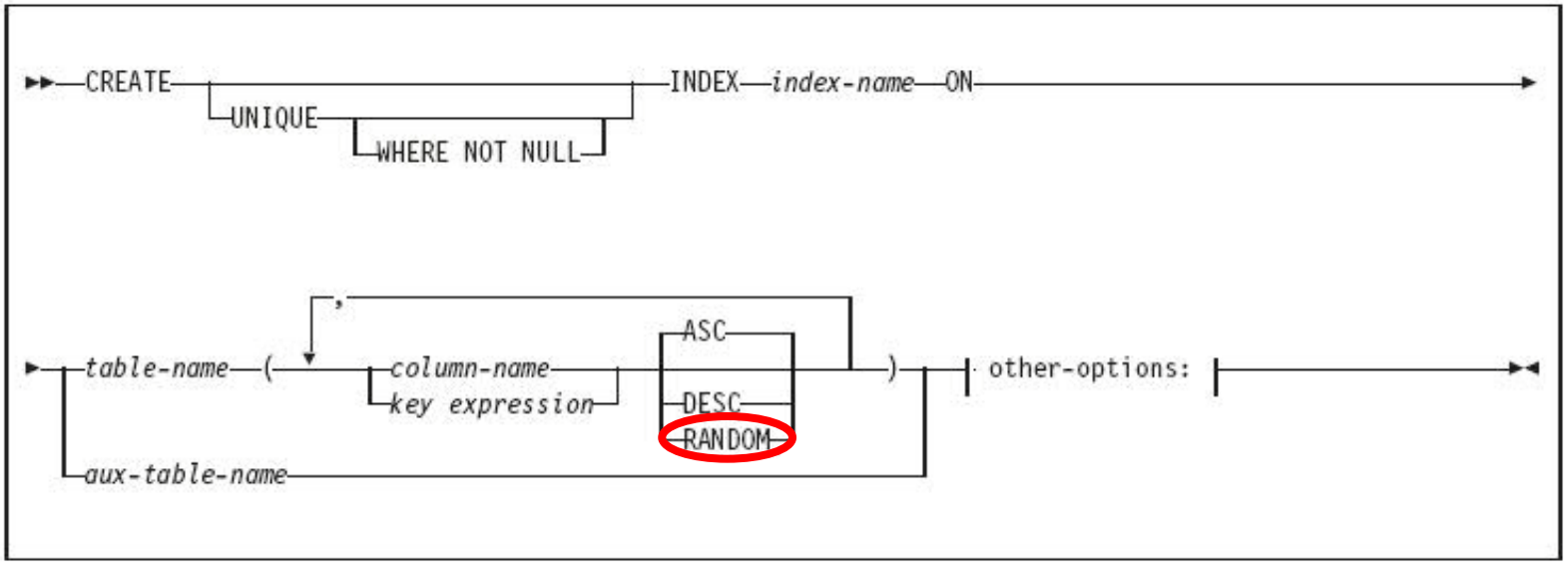


- Index split roughly 50/50 (prior to DB2 9)
- Sequential inserts → ~50% free space

- Up to 50% reduction in IX page splits
- Up to 20% reduction in DB2 CPU
- Up to 30% reduction in DB2 ET

- New algorithm dynamically accommodates a varying pattern of inserts
- Up to 90/10 split
- Effective across multiple inserting threads (due to tracking at the page level).
- Improve space utilization and reduce contention.

Randomized Index Key (NFM)



- Lock contention relief
- LC 6 relief

Vs.

- Additional getpages
- Additional read/write I/Os
- Increased lock requests

- Cannot support order
- Can provide dramatic improvement or degradation!
- Recommend making randomized indexes bufferpool resident
- Can be any one or more columns of an IX key

Index Compression (NFM)

- Always stored as 4k page on disk
- Best with high BP hit ratio

Difference between data and index compression

	Data	Index
Level of compression	Row	Page (1)
CPU overhead (who is charged for comp/decomp)	In Acctg	In Acctg and/or DBM1 SRB
Comp in DASD	Yes	Yes
Comp in BP and Log	Yes	No
Comp Dictionary	Yes	No (2)
'Typical' Comp Ratio CR	10 - 90%	25 - 75% (3)

Use DSN1COMP utility to predict index compression ratio.

Larger Index page Sizes (NFM)

- 8K, 16K, or 32K page
  - Up to 8 times less index split (16x with asym. IX splits)
- Good for heavy inserts to reduce index splits
  - Especially recommended if high LC6 contention in data sharing
    - 2 forced log writes per split in data sharing
  - Or high LC254 contention in non data sharing shown in IFCID57

- Lower NLEAF & NLEVELS (more keys per page)
- Exploitation of larger page sizes (> 8K) more likely without index compression
- Better IX look-a-side and getpage avoidance
- Can result in increased (or decreased) I/O overhead

- Up to 50% CPU & 40% ET reduction in DS
- Up to 20% CPU & 30% ET reduction in non DS

Index *Look-aside* (CM)

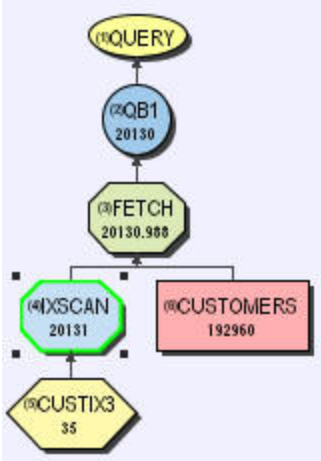
- In V8
  - Insert – clustering index only
  - Delete – no index lookaside
- In V9,
  - Insert & Delete – now possible for additional indexes where CLUSTERRATIO >= 80%
  - IX Update = Delete + Insert
- Potential for big reduction in index getpages and thus CPU time
  - Benchmark Example - Heavy insert
    - Large table, 3 indexes, all in ascending index key sequence,
    - 0+6+6=12 index Getpages per average insert in V8
    - 0+1+1=2 in V9
- Big winner for seq. insert, update or delete patterns

Index on Expression

- DB2 9 supports “index on expression”
  - Can turn a stage 2 predicate into indexable

```
SELECT *
FROM CUSTOMERS
WHERE YEAR(BIRTHDATE) = 1971

CREATE INDEX ADMF001.CUSTIX3
    ON ADMF001.CUSTOMERS
(YEAR(BIRTHDATE) ASC)
```



Name	Value
Input RIDs	192960
Index Leaf Pages	241
Matching Predicates	Filter Factor
ADMF001.CUSTOMERS.= CAST(1971 AS INTEGER)	0.1043
Scanned Leaf Pages	26
Output RIDs	20131
Total Filter Factor	0.1043
Matching Columns	1

Previous FF = 1/25
Now, RUNSTATS collects frequencies. Improved FF accuracy

Index Enhancement - Tracking Usage

- Additional indexes require overhead for
  - Utilities
    - REORG, RUNSTATS, LOAD etc
  - Data maintenance
    - INSERT, UPDATE, DELETE
  - Disk storage
  - Optimization time
    - Increases optimizer's choices
- But identifying unused indexes is a difficult task
  - Especially in a dynamic SQL environment

Tracking Index Usage

- RTS records the index last used date.
  - SYSINDEXSPACESTATS.LASTUSED
    - Updated once in a 24 hour period
      - *RTS service task updates at 1st externalization interval (set by STATSINT) after 12PM.*
    - if the index is used by DB2, update occurs.
    - If the index was not used, no update.

- "Used", as defined by DB2 as:
  - As an access path for query or fetch.
  - For searched UPDATE / DELETE SQL statement.
  - As a primary index for referential integrity.
  - To support foreign key access

General Query Performance Enhancements

GROUP BY Sort Avoidance

- Improved sort avoidance for GROUP BY
  - Reorder GROUP BY columns to match available index

```
SELECT … FROM T1
GROUP BY C2, C1      ←GROUP BY in C2, C1 sequence
Index 1 (C1, C2)     ←Index in C1, C2 sequence
```

  - Remove 'constants' from GROUP BY ordering requirement

```
SELECT … FROM T1
WHERE C2 = 5         ←C2 Constant
GROUP BY C2, C1
```

    - ordering requirement reduced to just C1

GROUP BY Sort Avoidance

- Continued….
  - Allow swapping of ordering columns using transitive closure

```
SELECT … FROM T1, T2
WHERE T1.C1 = T2.C1
GROUP BY T1.C1, T2.C3
```

←Contains T1 & T2

    - ordering requirement changed to T2.C1, T2.C3

  - Improvement for 'partially ordered' cases with unique index

```
SELECT C1, C2+C3, C4 FROM T1
GROUP BY 1, 2, 3
```

    - if we have unique index on C4, C1
      - *Sort can be avoided*

GROUP BY Sort Avoidance Implications

- Implications of improved sort avoidance for GROUP BY
  - May improve query performance!!!
  - Data may be returned in a different order
    - Always been true in any DB2 release
      - *Also true in other DBMSs*
    - Relational theory states that order is NOT guaranteed without ORDER BY

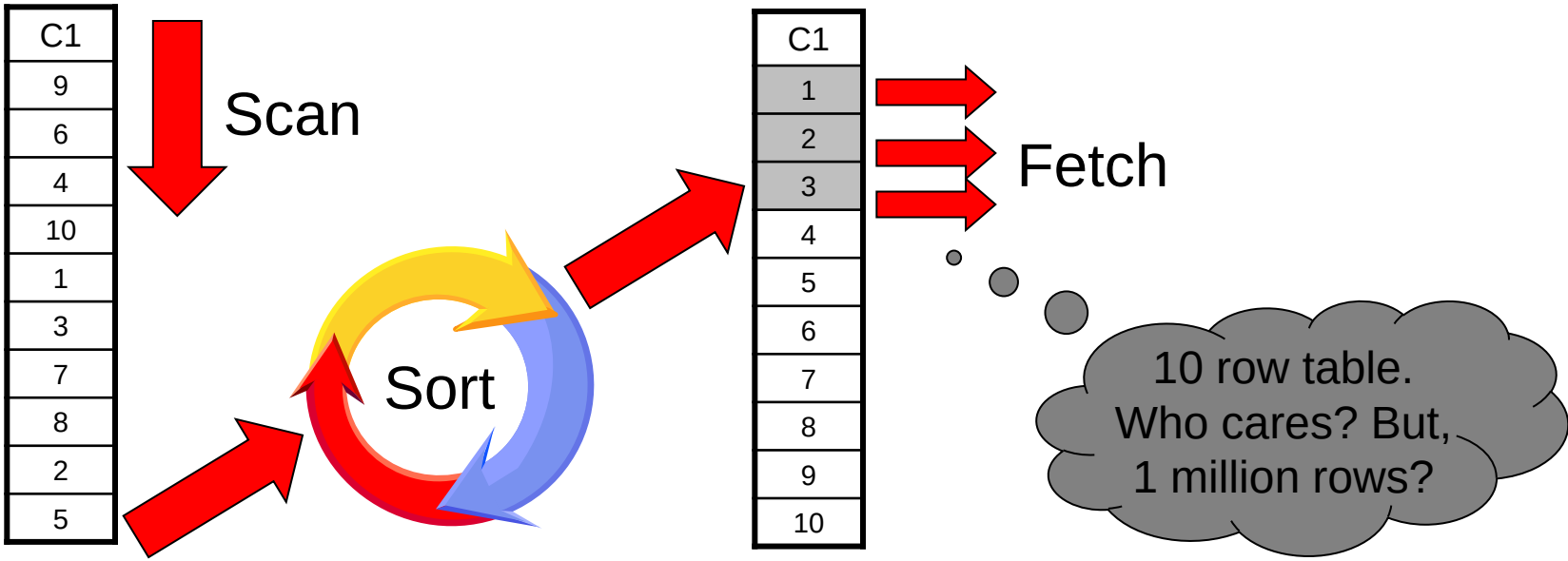
Sort Improvements

- Reduced workfile usage for very small sorts
  - Final sort step requiring 1 page will NOT allocate workfile
- More efficient sort with FETCH FIRST clause
  - V8 and prior,
    - Sort would continue to completion
    - Then return only the requested ‘n’ rows
  - From V9,
    - If the requested ‘n’ rows will fit into a 32K page,
      - *As the data is scanned,*
        - *Only the top ‘n’ rows are kept in memory*
        - *Order of the rows is tracked*
        - *No requirement for final sort*

FETCH FIRST V8 Example

- Sort is not avoided via index
  - Must sort all qualified rows

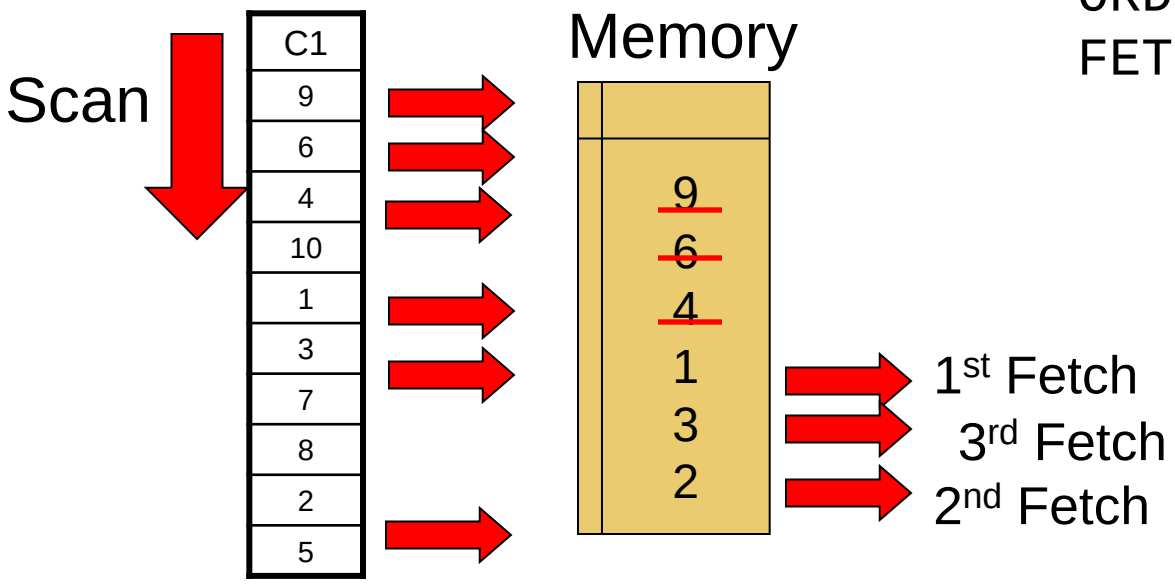
```
SELECT C1
FROM  T
ORDER BY C1
FETCH FIRST 3 ROWS ONLY
```



FETCH FIRST DB2 9 Example

- Sort is not avoided via index
  - But in-memory swap avoids sort
    - Pointers maintain order

```
SELECT C1
FROM  T
ORDER BY C1
FETCH FIRST 3 ROWS ONLY
```



Suggestion: Add FETCH FIRST when subset is required.

Dynamic Prefetch Enhancements

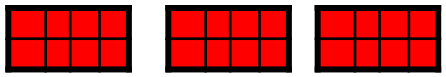
Sequential Prefetch	Dynamic Prefetch
Chosen at bind/prepare time	Detected at runtime
Requires hit to a triggering page	Tracks sequential access pattern
Only prefetch in one direction	Prefetch forward or backward
Used for tablespace scan & LOBs	Used for index & index+data access

- Seq. Pref. cannot fall back to Dyn. Pref. at run time
- Plan table may still show ‘S’ for IX + Data access

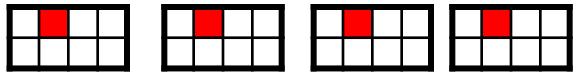
- ET reductions between 5-50% measured at SVL
- 10-75% reduction in synchronous I/O’s

Clusterratio Enhancement

- New Clusterratio formula in DB2 9
  - Including new DATAREPEATFACTOR statistic
    - Differentiates density and sequential

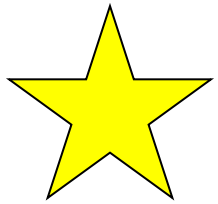


Dense (and sequential)



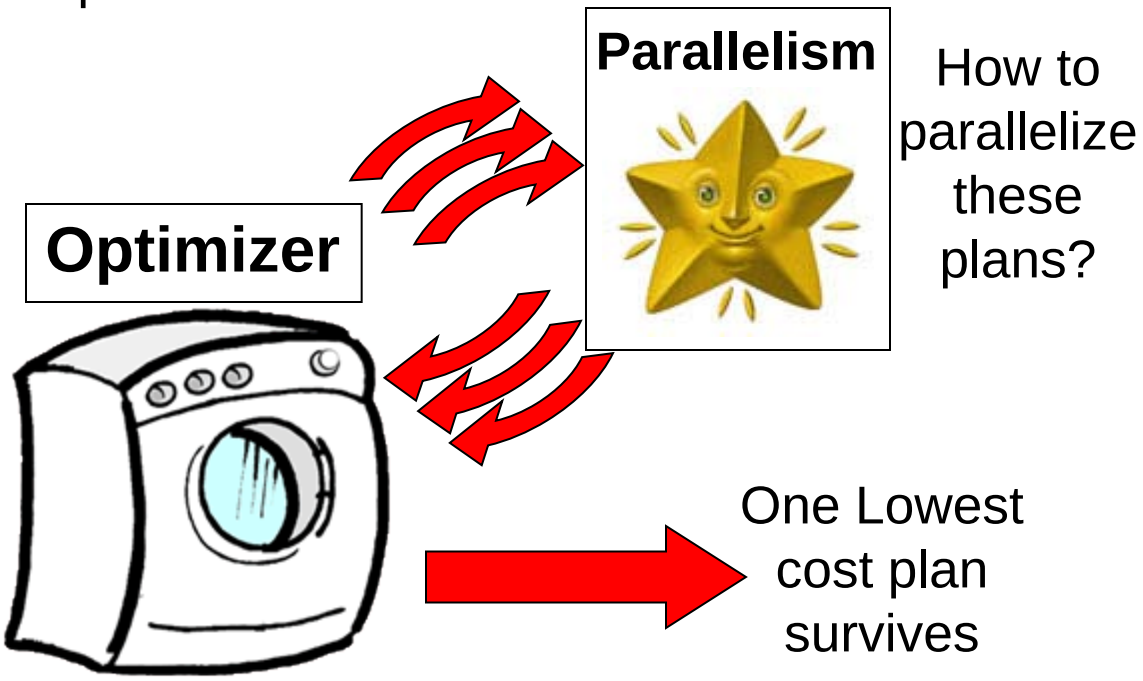
Sequential (not dense)

- Controlled by zparm STATCLUS
  - ENHANCED is default
  - STANDARD disables, and is NOT recommended
- Recommend RUNSTATS before mass REBIND in DB2 9



Parallelism Enhancements

- In V8
  - Lowest cost is BEFORE parallelism
- In DB2 9
  - Lowest cost is AFTER parallelism
    - Only a subset of plans are considered for parallelism



Additional Parallelism Enhancements

- In V8
  - Degree cut on leading table (exception star join)
- In DB2 9
  - Degree can cut on non-leading table
    - Benefit for leading workfile, 1-row table etc.
  - Histogram statistics exploited for more even distribution
    - For index access with NPI
  - CPU bound query degree <= # of CPUs * 4
    - <= # of CPUs in V8

Histogram Statistics

RUNSTATS Histogram Statistics

- **RUNSTATS** will produce equal-depth histogram
  - Each quantile (range) will have approx same number of rows
    - Not same number of values
  - Another term is range frequency

- Example
  - 1, 3, 3, 4, 4, 6, 7, 8, 9, 10, 12, 15 (sequenced)
  - Lets cut that into 3 quantiles.
    - 1, 3, 3, 4 ,4 6,7,8,9 10,12,15

Seq No	Low Value	High Value	Cardinality	Frequency
1	1	4	3	5/12
2	6	9	4	4/12
3	10	15	3	3/12

RUNSTATS Histogram Statistics Notes

- RUNSTATS
  - Maximum 100 quantiles for a column
  - Same value columns WILL be in the same quantile
  - Quantiles will be similar size but:
    - Will try to avoid big gaps inside quantiles
    - Highvalue and lowvalue may have separate quantiles
    - Null WILL have a separate quantile
- Supports column groups as well as single columns
- Think “frequencies” for high cardinality columns

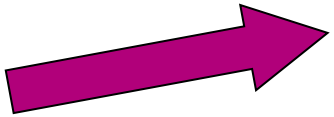
Histogram Statistics Example

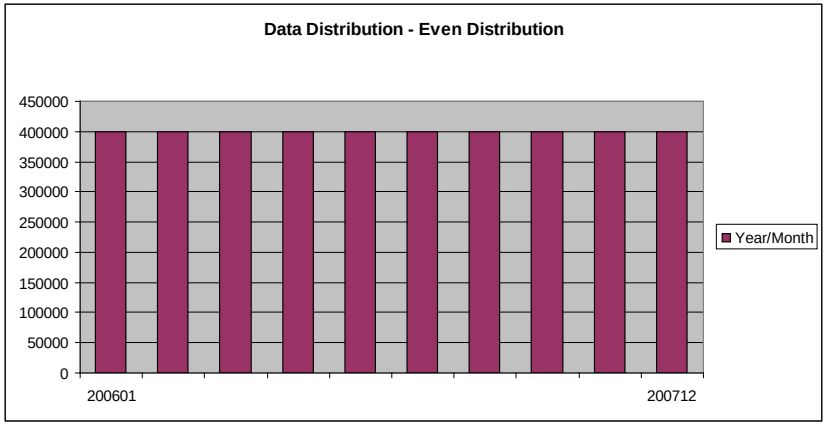
- SAP uses INTEGER (or VARCHAR) for YEAR-MONTH

WHERE YEARMONTH BETWEEN 200601 AND 200612

  - Assuming data for 2006 & 2007
    - *FF = (high-value – low-value) / (high2key – low2key)*
    - *FF = (200612 – 200601) / (200711 – 200602)*
    - ***10% of rows estimated to return***

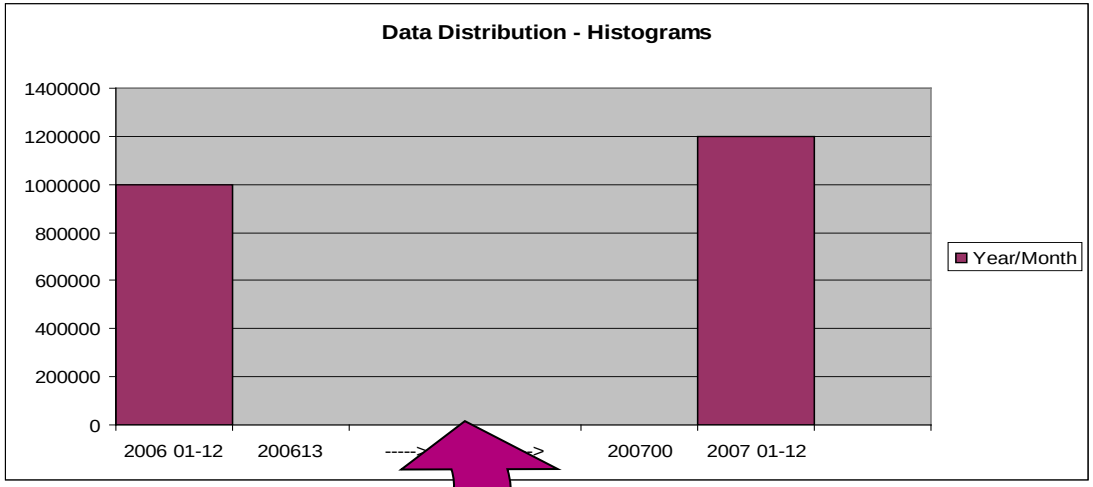
Data assumed as evenly distributed between low and high range





Histogram Statistics Example

- Example (cont.)
  - Data only exists in ranges 200601-12 & 200701-12
    - Collect via histograms
      - *45% of rows estimated to return*



No data between
200613 & 200700

WHERE YEARMONTH BETWEEN 200601 AND 200612

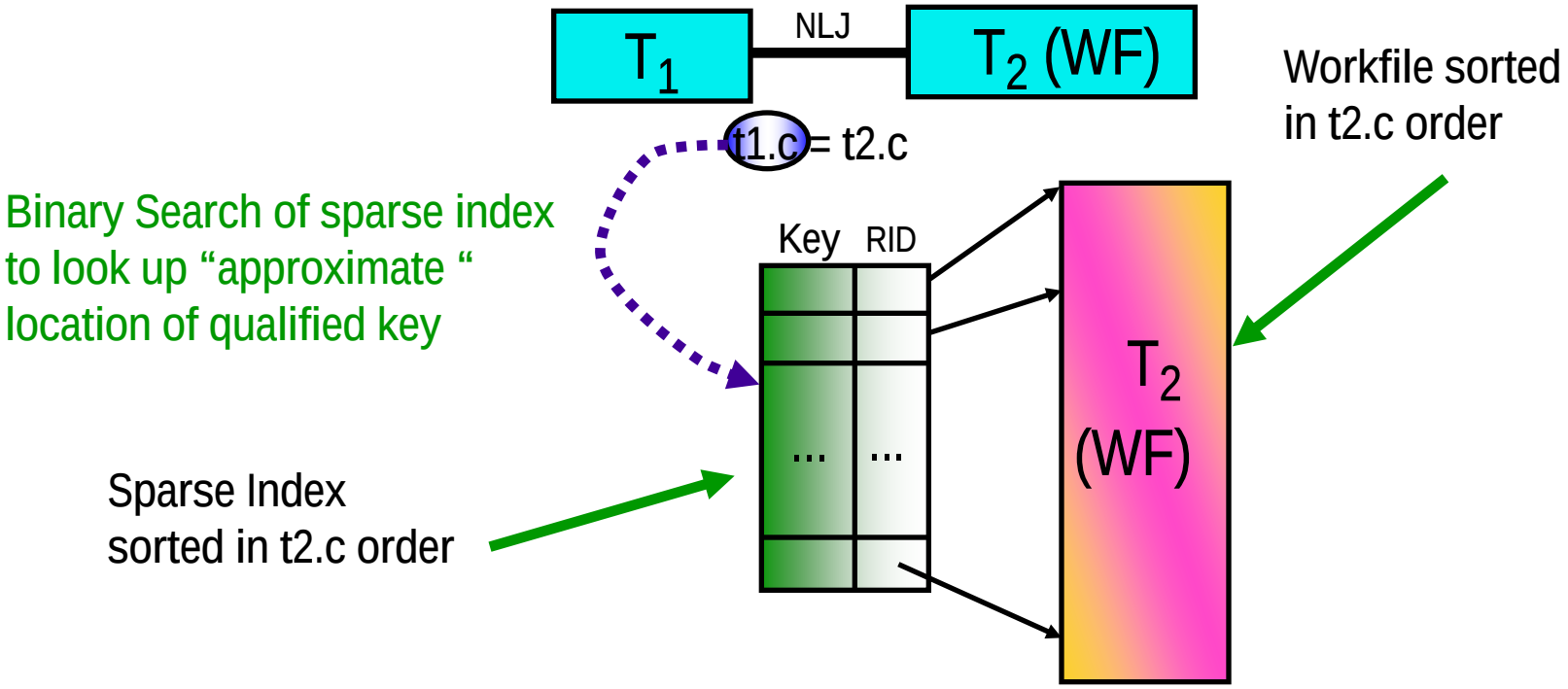
Generalized Sparse Index and In-memory Data Caching

Pre-V9 Sparse Index & in-memory data cache

- V4 introduced sparse index
  - for non-correlated subquery workfiles
- V7 extended sparse index
  - for the materialized work files within star join
- V8 replaced sparse index
  - with in-memory data caching for star join
    - Runtime fallback to sparse index when memory is insufficient

How does Sparse Index work?

- Sparse index may be a subset of workfile (WF)
  - Example, WF may have 10,000 entries
    - Sparse index may have enough space (240K) for 1,000 entries
    - Sparse index is “binary searched” to find target location of search key
    - At most 10 WF entries are scanned

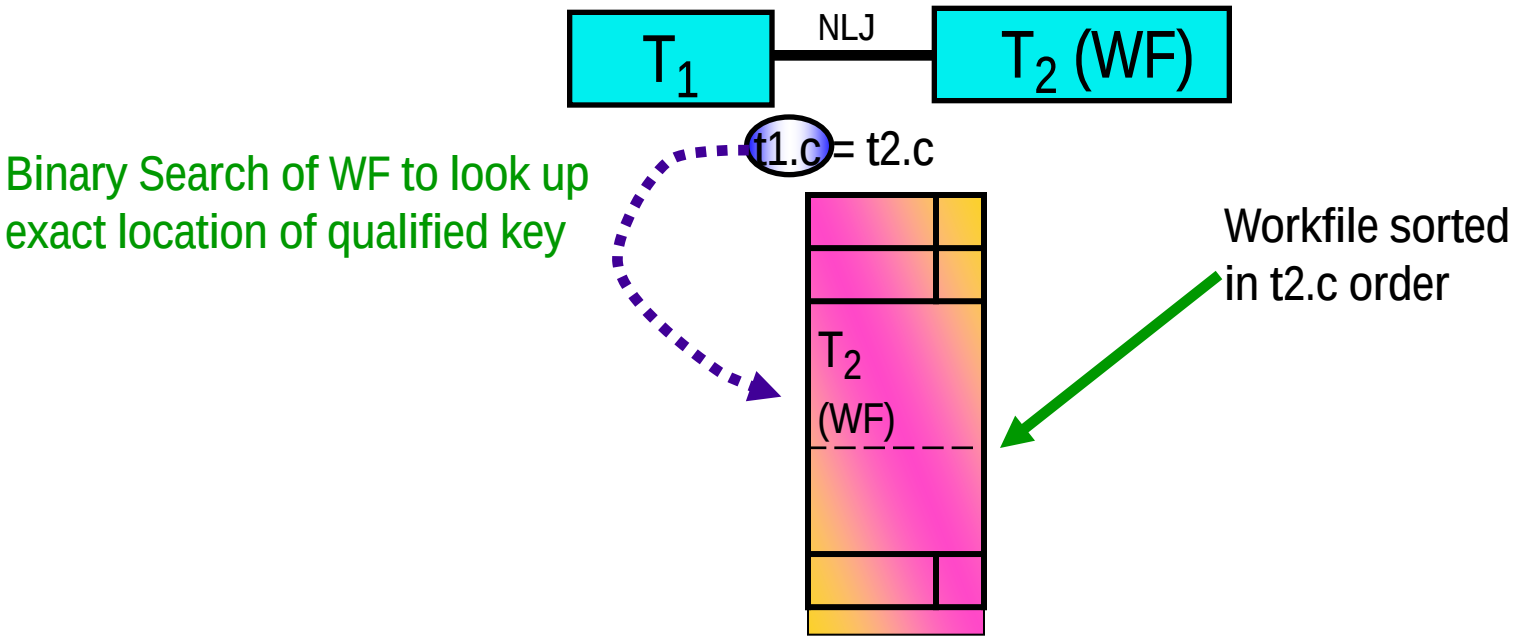


Data Caching vs Sparse Index

- Data Caching
  - Also known as In-Memory WF
  - Is a runtime enhancement to sparse index
- Sparse Index/In-Memory WF
  - Extended to non-star join in DB2 9
- New ZPARM MXDTCACH
  - Maximum extent in MB, for data caching per thread
  - If memory is insufficient
    - Fall-back to sparse index at runtime

How does In-Memory WF work?

- Whereas sparse index may be a subset of WF
  - IMWF contains the full result (not sparse)
  - Example, WF may have 10,000 entries
    - IMWF is “binary searched” to find target location of search key



Benefit of Data Caching

- All tables lacking an index on join column(s):
  - Temporary tables
  - Subqueries converted to joins
  - …..any table

- V9 also supports multi-column sparse index

REOPT Auto Based On Parameter Marker Change

REOPT enhancement for dynamic SQL

- V8 REOPT options
  - Dynamic SQL
    - REOPT(NONE, ONCE, ALWAYS)
  - Static SQL
    - REOPT(NONE, ALWAYS)
- V9 Addition for Dynamic SQL
  - Bind option REOPT(AUTO)

Dynamic SQL REOPT - AUTO

- For dynamic SQL with parameter markers
  - DB2 will automatically reoptimize the SQL when
    - Filtering of one or more of the predicates changes dramatically
      - *Such that table join sequence or index selection may change*
    - Some statistics cached to improve performance of runtime check
  - Newly generated access path will replace the global statement cache copy.

- First optimization is the same as REOPT(ONCE)
  - Followed by analysis of the values supplied at each execution of the statement

V10 Query Performance Enhancements Overview

DB2 10 Query Enhancements

- CPU time reductions for queries, batch, & transactions
- SQL enhancements: Moving Sum, Moving Average, temporal, timestamp, implicit cast, SQL PL, …
- pureXML improvements
- Access improvements: Index include columns, hash, index list prefetch, workfile spanned records, …
- Optimization techniques
  - Remove parallelism restrictions and more even parallel distribution. Increased zIIP usage.
  - In-memory techniques for faster query performance
  - Access path stability and control
- Analysis: instrumentation, Data Studio & Optim Query Tuner
- Advanced query acceleration techniques
  - IBM Smart Analytics Optimizer

Safe Query Optimization

- V10 - Consider the uncertainty of predicate filtering when selecting an index
  - Uncertain predicate filtering from non-uniform data, host variables or parameter markers
  - DB2 might choose an index that has slightly higher cost estimate if that index has a higher cost certainty

- V10 - List prefetch enhancement
  - RID list processing continues in work file when DB2 exhausts the RID pool resources (avoid R scan)

Range List Index Scan - problem to be solved

Table PHONEBOOK

last name	first name	phone	street	city	…

```
SELECT *
   FROM PHONEBOOK
WHERE (LASTNAME='SMITH' AND
FIRSTNAME>='JOHN') OR
(LASTNAME>'SMITH')
ORDER BY LASTNAME, FIRSTNAME
FETCH FIRST 10 ROWS;
```

Index IX1 on (lastname, firstname)

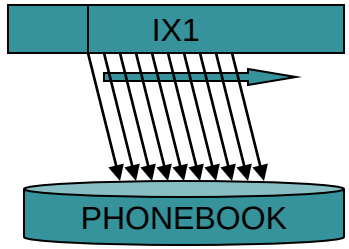
Current possible access path

1. Table space scan
2. Non-matching index access
3. Multi-index access (index ORing)

Both need to retrieve all the qualified rows, sort them for ORDER BY , then return 10 first ordered result

Ideal access path

Direct index access and avoid ORDER BY sort
Terminate the process after getting first 10 rows



Range List Index Scan

- V 10 – DB2 introduces new access method, range list access, to process OR predicates
  - Range list access is only considered when
    - OR predicate is on leading table
    - OR predicate is stage 1 predicate
    - Each disjunct has at least one matching column
- New access type ‘NR’ in PLAN_TABLE

Range List Index Scan Benefits

- An index access with matching predicates. It can narrow down the search scope comparing to tablespace scan or non-matching index scan.
- It is a single index access instead of multiple indexes access (index ORing). Index is exploited once.
- It allows index key ordering to be maintained which is significantly important to data-dependent pagination application. If the index satisfies order by ordering, sort can be avoided.
- Process can be terminated early if only part of result set is required (e.g. with FETCH FIRST n ROWS ONLY clause).

IN-list Predicate Enhancements

- Index matching on multiple IN-list predicates.
- Predicate transitive closure for IN-list predicates.
- List prefetch.
- New access type – ACTYPE - 'IN' and new table type TBTYPE – 'I'
  - If more than one matching IN-list predicates
  - Each IN-list predicate is an in-memory table

View/Table Expression Merge Enhancement

- More Merge scenarios (instead of physical materialization) for View/Table Expressions
  - Especially in outer join.
    - More join sequence can be considered.
    - Can apply predicates early
- In general a more aggressive Merge strategy for View/Table Expressions is preferable.

Stage 2 Predicate Pushdown to Stage 1

- Example:

```
CREATE TABLE T1(C1,C2,C3)
CREATE INDEX IX1 ON T1(C1,C2)

SELECT *
FROM T1 WHERE T1.C1 >0
AND T1.C2+1=5 AND T1.C3+2=4 ;
```

Matching predicate

IM pushdown

DM pushdown

Query Parallelism Enhancements

- What are the enhancements to reduce query elapsed time?
  - Dynamic record range partitioning
  - Straw model
  - Removal of some parallelism restrictions
  - SMJ with sparse index on inner table work file
- When the enhancements are not eligible?
  - Sysplex parallelism
  - Full outer join queryblock
  - IO parallelism

Dynamic Record Range Partitioning

- Why record range partitioning?
  - Key range partitioning is determined at bind time
  - Key range may not be cut evenly due to data skew, data correlation and out of date statistics
    - *Query elapsed time not optimal due to unbalanced amount of work in parallel child tasks*

- What is dynamic record range partitioning?
  - Dynamically materialize the intermediate result in joins
    - *Result may fit in in-memory work file*
  - Based on the number of records in the composite table
  - Divide the result into ranges with equal number of records
  - Up to x times reduction in elapsed time with x parallel degree

- If used, dsn_pgroup_table field RANGEKIND = ‘R’

Parallelism Straw Model

- Difference vs. non-straw model
  - Number of ranges (elements) > number of degree
  - Number of parallel tasks = number of degree
    - True for both straw and non straw model
  - Each parallel task continues on the next available range after it finishes the current one
  - Parallel tasks stop after all the ranges are processed

- When it is used?
  - Parallel group cost is not too small
  - Leading table is index access and the colcard is not too small
  - Leading table is R-scan and the number of pages is not too small

- If used, dsn_pgroup_table field STRAW_MODEL = 'Y'

Remove Restrictions for Parallelism

- Parallelism no longer disabled when parallel group contains work file from
  - Materialized view
  - Materialized table expression
- Parallelism no longer disabled in the last parallel group of the top query block when multi-row fetch is used
  - This restriction is removed under the condition that the cursor is **read only**
- Parallelism no longer disabled when queryblock contains OLAP functions
  - OLAP functions are still processed at parent side

Disclaimer/Trademarks

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information on the new product is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information on the new product is for informational purposes only and may not be incorporated into any contract. The information on the new product is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion. *

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the pages of the presentation:

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: AIX, AS/400, DataJoiner, DataPropagator, DB2, DB2 Connect, DB2 Extenders, DB2 OLAP Server, DB2 Universal Database, Distributed Relational Database Architecture, DRDA, eServer, IBM, IMS, iSeries, MVS, Net.Data, OS/390, OS/400, PowerPC, pSeries, RS/6000, SQL/400, SQL/DS, Tivoli, VisualAge, VM/ESA, VSE/ESA, WebSphere, z/OS, zSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Other company, product, or service names may be trademarks or service marks of others.

Key Details About DB2 10: Getting Ready

Prerequisites: migrate from DB2 9 for z/OS or DB2 for z/OS V8

- z/OS V1.10 SMS-managed DB2-managed DB2 catalog
- System z10, z9, z890, z990, and above (no z800, z900)
- DB2 Connect 9 FP1, 9.7 FP3 for many 10 functions, FP2 beta
- IMS 10 & 11 (not 9) CICS compilers (See announcement)
- Info APARs for migration II14477 (9), II14474 (8)
- SPE PK56922 PK69411 PK61766 PK85956 PM04680 PK87280 PK87281 PM08102 PM08105
- Premigration check DSNTIJPA PM04968

Items deprecated in earlier versions eliminated: more for V8 mig.

- Private protocol → DRDA (DSNTP2DP, PK92339, PK64045)
- Old plans and packages V5 or before → REBIND
- Plans containing DBRMs → packages PK62876 PK79925 (V8)
- ACQUIRE(ALLOCATE) → ACQUIRE(USE)
- Old plan table formats → DB2 V8 or 9, Unicode, 59 cols PK85068
- BookManager use for DB2 publications → Info Center, pdf

Exciting Optimizer & SQL Performance Enhancements in DB2 9 for z/OS and Beyond

James Guo, guojw@us.ibm.com
IBM Silicon Valley Lab