



IBM Software Group

DB2 for z/OS Evolution of SQL Optimization V7, V8 and DB2 9

Yoichi Tsuji, Silicon Valley Laboratory



@business on demand.

Disclaimer

© Copyright IBM Corporation 2008. All rights reserved.
U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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.

IBM, the IBM logo, ibm.com, DB2, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml



Agenda

- *Main themes of DB2 for z/OS optimization development*
- *V7 updates – major PTF's*
- *V8 what's new*
- *V8 updates – major PTF's*
- *DB2 9 what's new*
- *Future visions*



Main Themes of DB2 for z/OS Optimization Development

- More efficient SQL and data base operations of DB2 for z/OS
 - ▶ Bind time to run time
- Plan stability
 - ▶ Base line: Stable access path selection for high performance
 - ▶ The “Plan stability” features
 - ▶ “REOPT”
 - ▶ “Optimization hint”
 - ▶ More statistics
- More features and tools to understand, diagnose and tune the access paths easier
 - ▶ IBM OSC and OE



V7 Updates

- **PQ73454 (UQ78681)** Aug., 2003
 - ✓ Allow IN-list “pushdown” to materialized views or table expressions
 - ✓ Allow IN-list and simple predicates “bubble-up” from correlated subquery
 - ▶ Introduced a ZAPRM **INLISTP**
 - ✓ V7 default INLISTP = 0 (disabled)
 - ✓ V8 and later default INLISTP = 50 (enabled)
 - ▶ An example of predicate “bubble-up”

```
SELECT * FROM Product P
  WHERE EXISTS (SELECT 1 FROM Manufacturer M
                WHERE P.ProdType = M.ProdType
                AND M.ProdType = 'bolt' )
```

→

```
SELECT * FROM Product P
  WHERE EXISTS (SELECT 1 FROM Manufacturer C
                WHERE P.ProdType = M.ProdType
                AND M.ProdType = 'bolt' )
  AND P.ProdType = 'bolt'
```

- ▶ In the above example, a predicate in the subquery, **M.ProdType = 'bolt'**, is “bubbled-up” to the parent, through the correlation predicate **P.ProdType = M.ProdType**.



V8 – What's New in SQL Optimization

- ***More stage 1 predicates***
- ***REOPT ONCE***
- ***Enhancement on multi-column sort merge join***
- ***In-line cardinality option for user-defined table function***
- ***New statistics collection feature (RUNSTATS)***
- ***Materialized Query Table (MQT)***
- ***Star join performance improvements***



V8 – More stage 1 predicates (CM and NFM)

- Removed restrictions for some simple predicates to be stage 1 (and indexable) that are not stage 1 in prior releases
 - ▶ By adding an implicit cast, when it does not cause “data loss”
 - ▶ By adding an extra “weakened” stage 1 predicate, when applicable.
- Mismatched data type and/or length between two arguments of a predicate
 - ▶ Typical examples (the literals can be host variables)
`DecimalColumn = Float_Literal`
`Column_Char_3 = Char_5_Literal`
 - ▶ Some restrictions still apply.
- Advantage of having more stage 1 predicates
 - ▶ Earlier application of filtering
 - ▶ Stage 1 predicates can be indexable
- Access paths can be changed by exploiting this feature
- More details : DB2 UDB for z/OS Version 8 Performance Topics, GS24-6465



V8 – Enhancement of multi-column sort merge join

- DB2 V8 supports a sort merge join with multiple sort keys more efficiently
 - ▶ Possible to apply more early filtering during the join with multiple join predicates
 - ▶ Sorting the inner table or not is determined by costing. No sort if there is a “good” index to support the order (it is still materialized).
 - ▶ Parallelism is enabled for more multi-column sort merge join cases.



V8 – REOPT ONCE

- New BIND option for dynamic SQL
- How it works
 - ▶ Defers access path selection until OPEN
 - ▶ Values of parameter markers on OPEN are used in optimization
 - ▶ Resulting access path cached in global dynamic statement cache
- Advantage
 - ▶ Lower overhead compared with re-optimizing always
 - ▶ More realistic filtering estimation by using the values given for the “first time” than using the default (simple guess)
- REOPT AUTO in DB2 9
 - ▶ Re-optimize “when needed”



V8 – In-line cardinality option for table UDF

- A user-defined table function (table UDF) is a “black box” for optimizer.
 - ▶ The `CARDINALITY` option in the `CREATE FUNCTION` can help only when a table UDF always returns a constant number of rows.
- In V8, a new option `CARDINALITY` can be specified to give a “hint” to the optimizer for a table UDF reference in a query.
 - ▶ Especially for performance sensitive query using table UDF's
 - ✓ When an application can capture or estimate the number of rows returned by the UDF.
 - ✓ For example, for a UDF reading an external file and returning it as if it were a DB2 table
- Example

```
SELECT * FROM
```

```
TABLE(UDF_To_Read_File(Arguments) CARDINALITY 5) AS XFILE;
```

where only a numeric constant (an integer) can follow the `CARDINALITY` keyword.



V8 – New RUNSTATS options

- In V8, distribution or correlation statistics can be collected for any combination of columns
- New COLGROUP specification
 - ▶ Single or multiple columns
 - ▶ With FREQVAL keyword,
 - Count is a required argument
 - Single or multi-column frequency in SYSIBM.SYSCOLDIST
 - ▶ Without FREQVAL
 - Single column cardinality in SYSIBM SYSCOLUMNS
 - Multi-column cardinality in SYSIBM.SYSCOLDIST
- Collecting additional multi-column cardinalities or single / multi-column frequencies can help improve and/or stabilize access path selection.
 - ▶ Multi-column cardinality → for a combination of EQUAL predicates
 - ▶ Single or multi-column frequencies → for one or more EQUAL predicates with literal constants



V8 – Materialized Query Table (MQT)

- Creating an MQT can dramatically improve the performance of complex queries involving joins of many tables
 - ▶ Tables are pre-joined and saved in the MQT
 - Typically for report generation queries in a DW workload
 - ▶ Especially effective when
 - A combination of the joined tables, predicates, and/or GROUP BY specification is shared by multiple queries
 - The joined result set is large and a sort is required for GROUP BY
- Optimizer evaluates the costs of access paths
 - ▶ With and without an MQT replacement in a query
- Important factors to use MQT efficiently
 - ▶ Good indexes are created on the MQT → common search / join keys on the MQT among the referencing queries.
 - ▶ “Good statistics” are collected on the MQT when it is refreshed.



V8 – Star Join performance enhancements

- Queries on a star schema
 - ▶ Star join method is enabled by system parameters
 - ✓ STARJOIN → subsystem level enablement of star join
 - ✓ SJTABLES → query level (not to apply star join for small queries)
 - ✓ SJMXPOOL → max size of star join in-memory pool
 - ▶ DB2 for z/OS star join support (V8)
 - ✓ Good for highly normalized star schema
 - ✓ Multi-column index on the fact table
- V8 enhancements
 - ▶ Improved heuristics for star join detection
 - ▶ Improved heuristics for star join sequence generation
 - ▶ New support for in-memory work file with indexing capability
 - ✓ Improved uses of “sparse index” in a star join context



V8 Updates – Major APARs in SQL optimization

- **PQ87390**
 - ▶ Improvement in access path optimization with OPTIMIZE FOR n ROWS
- **PK30857 + PK60939**
 - ▶ Improvements in index access costing
 - ▶ PK30857 fix in V8 is disabled (a zparm OPTXOIRC to enable)
 - ▶ General recommendation: Enable PK30857 by setting OPTXOIRC = YES.
 - ▶ OPTXOIRC was removed in DB2 9.
- **PK67356**
 - ▶ Improvement in access path selection for sort merge join
- **PK31798 + PK58321 + PK69079**
 - ▶ Improvements in access path optimization with nested loop joins
 - ▶ Recommended to apply PK69079 (V8 UK39138, DB2 9 UK39139) though not HIPER.
 - ▶ PK69079 introduced a zparm OPTJBPR (default OFF, to disable a part of fix) → Keep the default (OFF) for this parameter for V8 and DB2 9 unless otherwise directed by IBM service.



DB2 9 – What's New in SQL Optimization

- ***The “plan stability” feature***
- ***REOPT AUTO***
- ***Generalized sparse index and in-memory cache***
- ***Global optimization***
- ***Page range processing***
- ***Histogram statistics***
- ***Improved index cluster ratio***
- ***Index on expression***
- ***Dynamic index ANDing (star join)***
- ***Improved RID list processing (star join)***
- ***Other enhancements***
- ***Optimization Service Center***



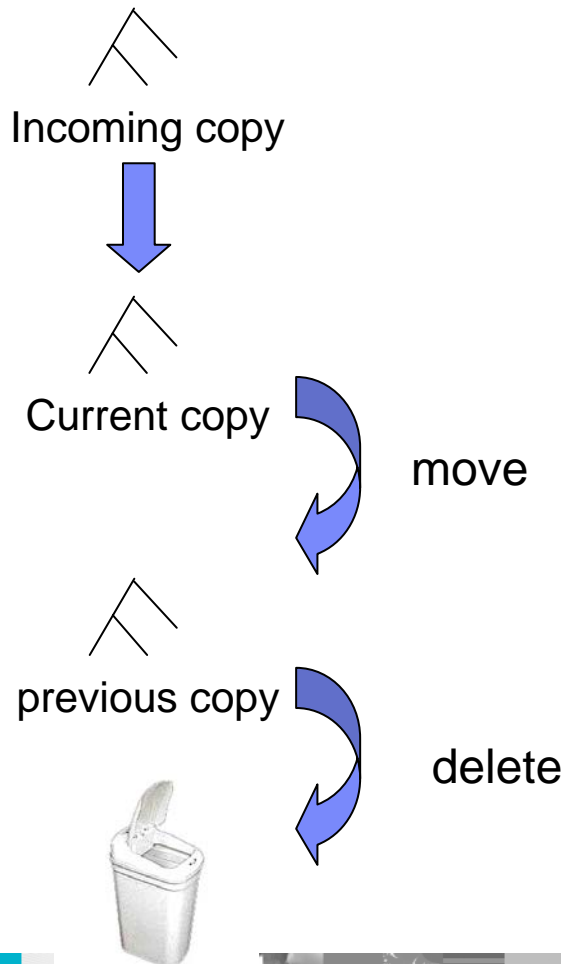
DB2 9 – Plan Stability Overview

- 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
- Supported as REBIND options
 - ▶ Default controlled by a ZPARM

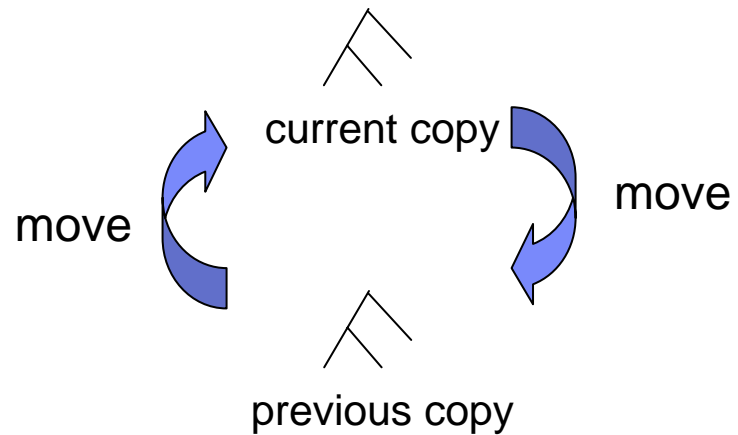


DB2 9 – Plan Stability “BASIC” support

REBIND ... PLANMGMT(BASIC)



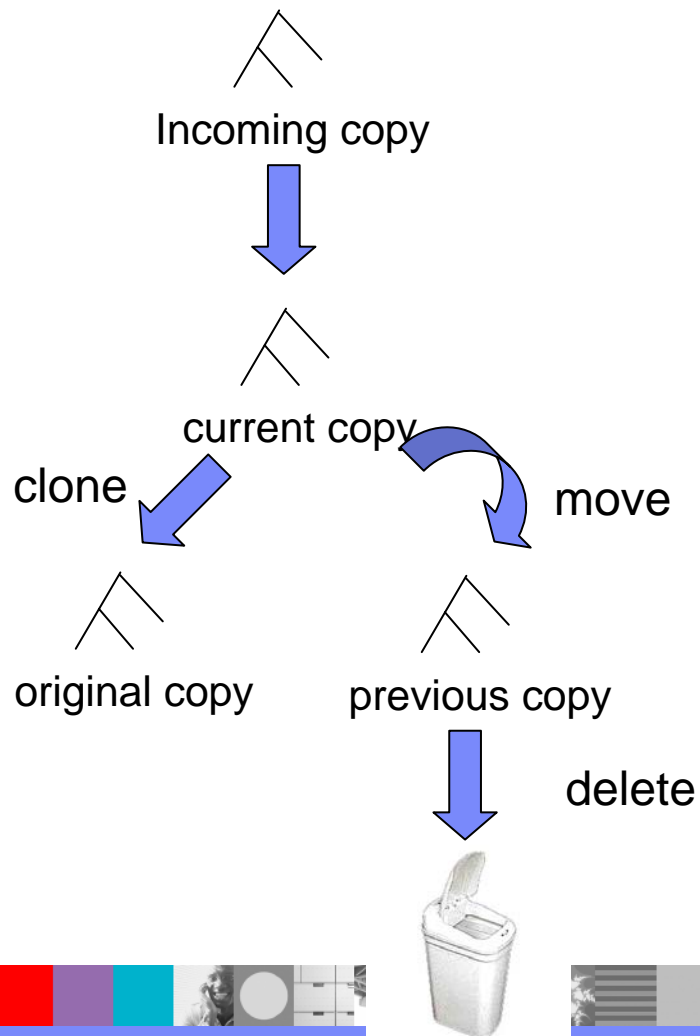
REBIND ... SWITCH(PREVIOUS)



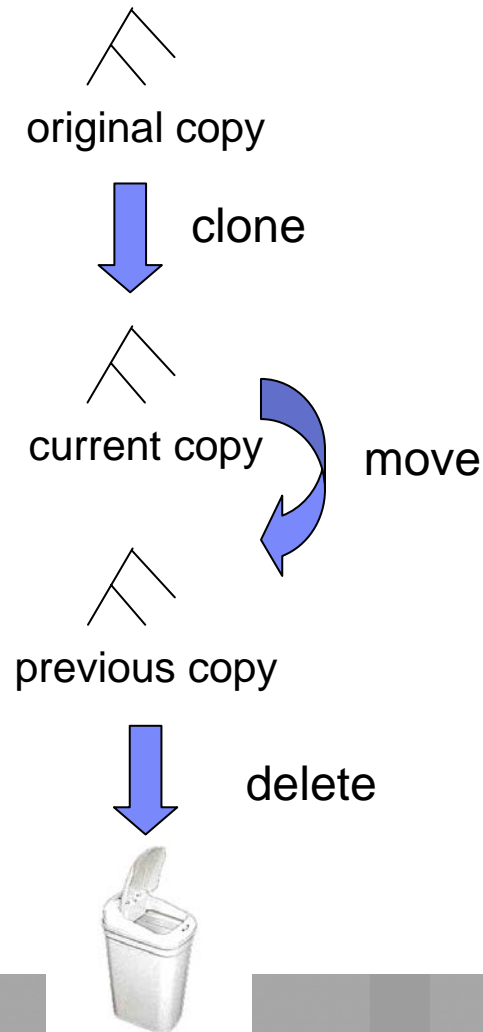
Actions occur from bottom to top

DB2 9 – Plan Stability “EXTENDED” support

REBIND ... PLANMGMT(EXTENDED)



REBIND ... SWITCH(ORIGINAL)



DB2 9 – REOPT AUTO

- V8 REOPT options
 - ▶ Dynamic SQL
 - REOPT(NONE, ONCE, ALWAYS)
 - ▶ Static SQL
 - REOPT(NONE, ALWAYS)

- V9 Addition for Dynamic SQL
 - ▶ Bind option REOPT(AUTO)



DB2 9 – 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



DB2 9 – Generalized Sparse Index and in-memory caching

- V4 introduced sparse index
 - ▶ for non-correlated subquery work files

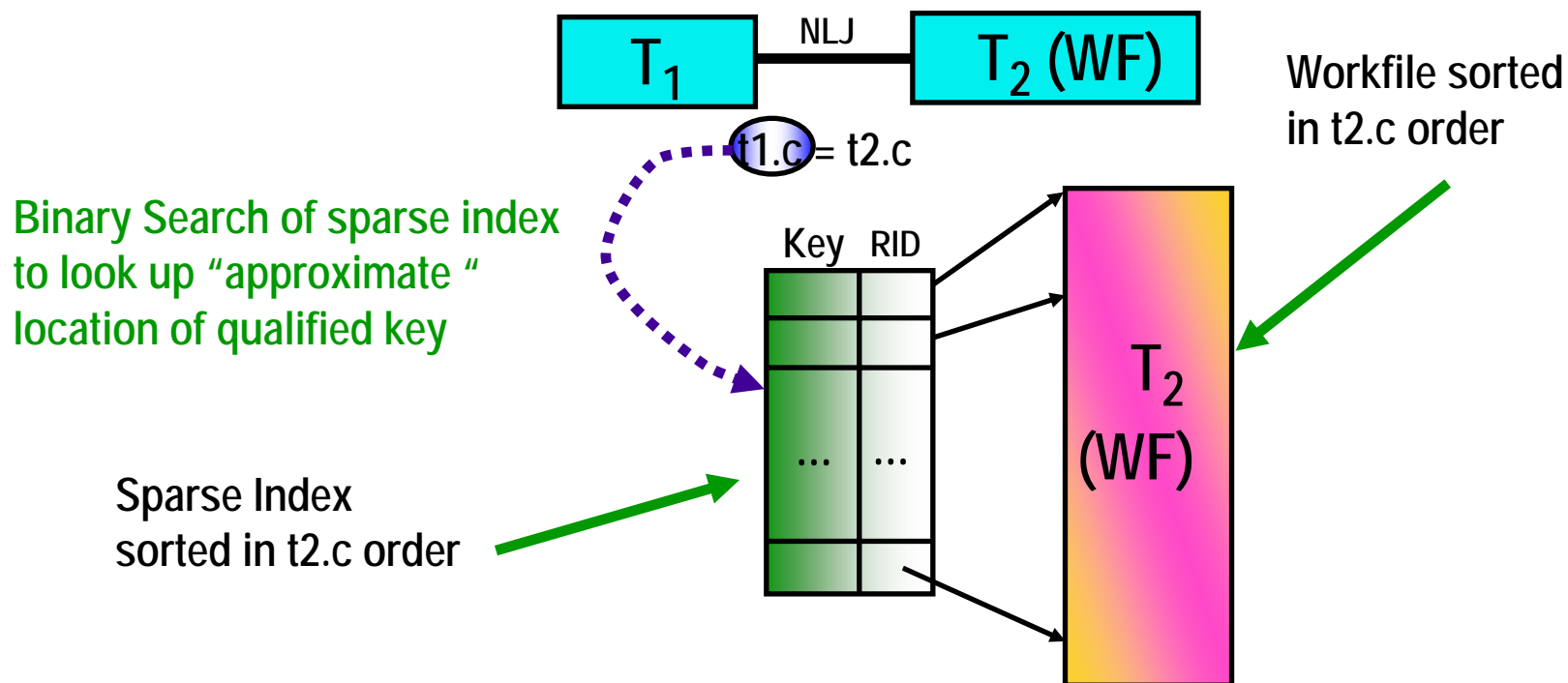
- V7 extended sparse index
 - ▶ for the materialized work files within star join

- V8 replaced sparse index
 - ▶ with in-memory data caching for star join
 - Runtime fallback to sparse index when memory is insufficient



DB2 9 – Generalized Sparse Index and in-memory caching

- How does a sparse index work ?
 - ▶ Example, a work file has 10,000 rows
 - 1,000 search keys can be stored in the space for “sparse index”
 - Sparse index is “binary searched” to find target location of search key
 - In average, 10 work file rows are scanned for each probe



DB2 9 – Generalized Sparse Index and in-memory caching

- In-memory data caching is extended to general cases (non-star join)
 - ▶ With enough space for data caching, work file data are kept in-memory.
 - ▶ A work file is NOT created, if data are cached in-memory.
 - ▶ Keys are “dense” (instead of “sparse”)

- V9 will use a local pool above the bar
 - ▶ Instead of a global pool used in V8 star join
 - ▶ Data caching storage management will be associated with each thread.
 - ✓ Which can reduce the potential storage contention.

- New ZPARM **MXDTCACH**
 - ▶ Specifies the maximum extent in MB, for data caching per thread.
 - ▶ Default is 20 MB.



DB2 9 – Global Optimization, scenario 1

- V8, Large Non-correlated subquery is materialized*

```
SELECT * FROM SMALL_TABLE S
WHERE S.C1 IN
      (SELECT B.C1 FROM BIG_TABLE B)
```

- “BIG_TABLE” is scanned and put into work file
- “SMALL_TABLE” is joined with the work file

- V9 may rewrite non-correlated subquery to correlated

- Much more efficient if scan / materialisation of BIG_TABLE was avoided
- Allows matching index access on BIG_TABLE

```
SELECT * FROM SMALL_TABLE S
WHERE EXISTS
      (SELECT 1 FROM BIG_TABLE B WHERE B.C1 = S.C1)
```



DB2 9 – Global Optimization, scenario 2

- V8, Large outer table scanned rather than using matching index access*

```
SELECT * FROM BIG_TABLE B
```

```
WHERE EXISTS
```

```
(SELECT 1 FROM SMALL_TABLE S WHERE S.C1 = B.C1)
```

- “BIG_TABLE” is scanned to obtain B.C1 value
- “SMALL_TABLE” gets matching index access

- V9 may rewrite correlated subquery to non-correlated

```
SELECT * FROM BIG_TABLE B
```

```
WHERE B.C1 IN
```

```
(SELECT S.C1 FROM SMALL_TABLE S)
```

- “SMALL_TABLE” scanned and put in work file
- Allows more efficient matching index access on BIG_TABLE

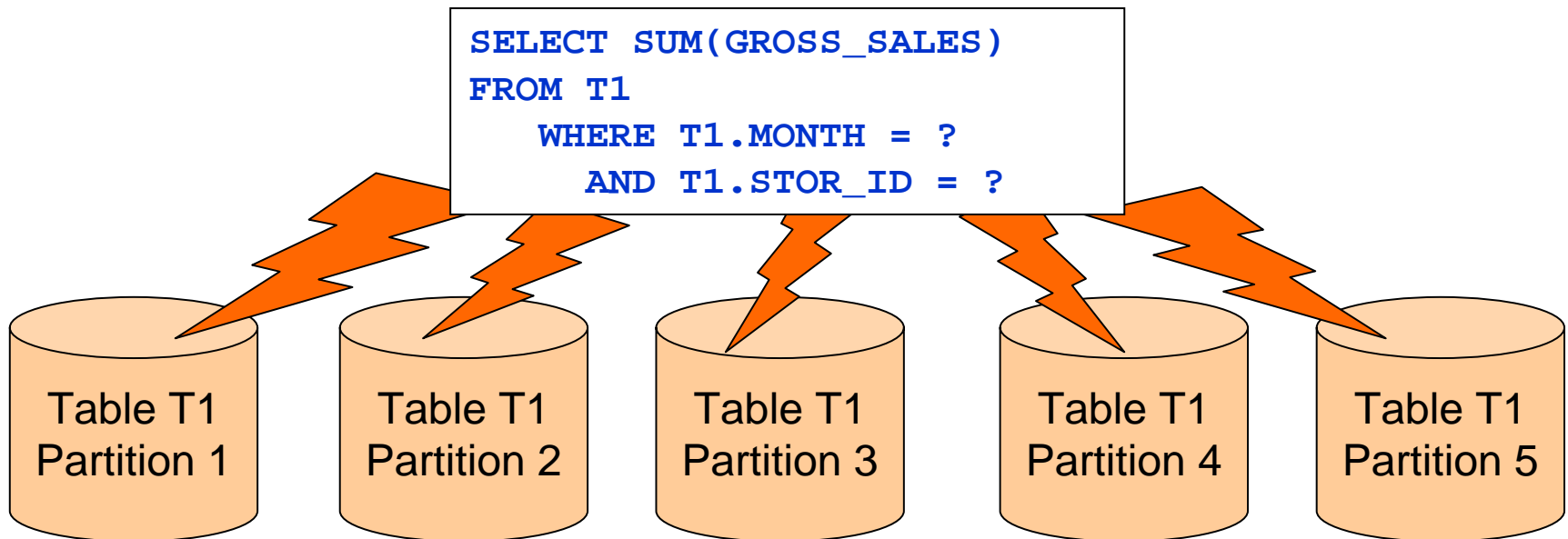


DB2 9 – Global Optimization, EXPLAIN output

- Additional row for materialized subquery block
 - ▶ Table type is "W" for "Workfile".
 - Name includes an indicator of the subquery QB number
 - Example → “DSNWFQB(02)”
 - ▶ When joined after the parent table
 - Sparse index can be used
 - PRIMARY_ACCESTYPE = ‘T’
- Additional column PARENT_PLANNO
 - ▶ Used with PARENT_QBLOCKNO to connect child QB to parent
 - ▶ V8 only contains PARENT_QBLOCKNO
 - Not possible to distinguish which plan step the child tasks belong to.

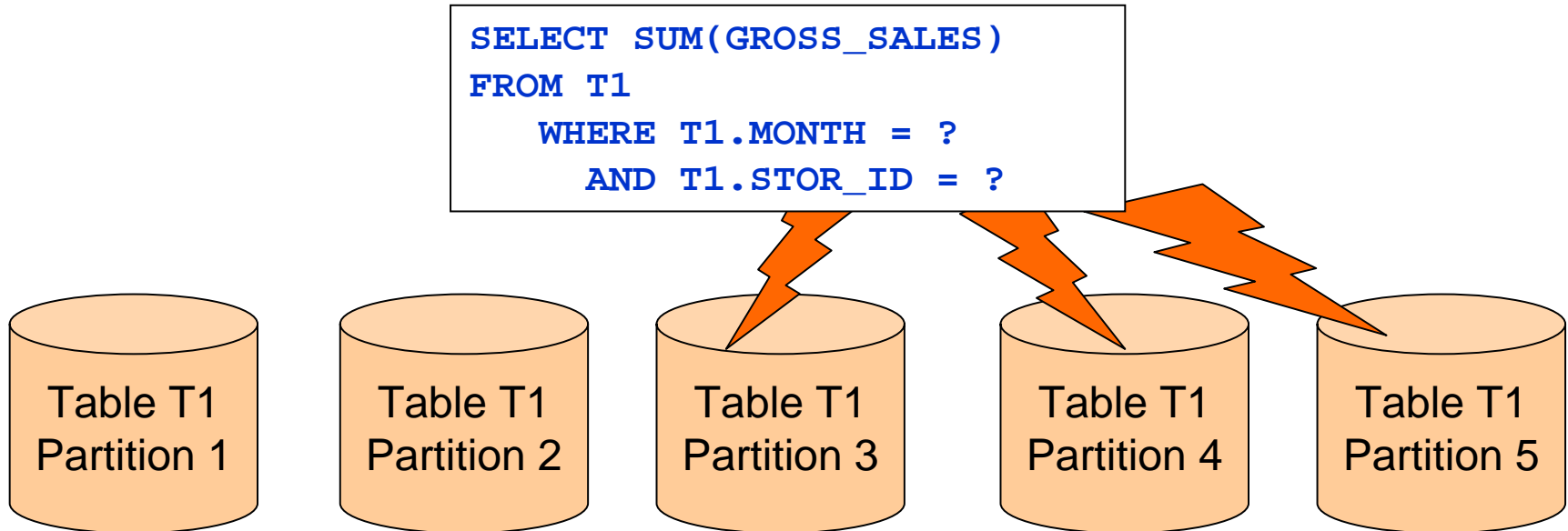


DB2 9 – Page Range Screening enhancements



- “Page range screening” = limiting the partitions accessed
- With DPSIs or tablespace scan of partitioned tablespace
 - ▶ beneficial to avoid accessing partitions with no qualifying rows
- Done using page range screening,
 - ▶ V8 support for local predicates on the leading partitioning key(s)
- **Reduces qualified rows read without indexing**

DB2 9 – Page Range Screening enhancements



- DB2 9 introduces two page range screening enhancements:
 - ▶ Join predicates
 - ▶ Non-matching predicates

DB2 9 – Page Range Screening enhancements

```
PARTITION BY (COL001 ASC)
(PART      1 VALUES('00000100'))
, PART     2 VALUES('00000200')
, PART     3 VALUES('00000300')
.....
, PART   999 VALUES('00099900')
, PART 1000 VALUES('00100000'))
```

```
SELECT *
FROM TABLEA A, TABLEB B
WHERE A.COL001 = B.COL001
AND   A.COL004 = B.COL004
```

Non-Indexed
partition key

DPSI key

- V8
 - ▶ All DPSI index parts accessed
 - page range screening for local predicates only

- V9
 - ▶ 1 DPSI index part on B accessed for each join row from A
 - join predicate(s) used for page range screening
 - 10X performance improvement in DB2 9 Redbook example

DB2 9 – Page Range Screening enhancements

```

PARTITION BY (COL001 ASC, COL002 ASC)
(PART 1 VALUES('00000100','00000002'),
 PART 2 VALUES('00000100','00000004'),
 PART 3 VALUES('00000100','00000006'),
 .....
 PART 50 VALUES('00000100','99999999'),
 PART 51 VALUES('00000200','00000002'),
 .....
 PART 1000 VALUES('99999999','99999999'))
  
```

```

SELECT SUM(COL008)
FROM TABLEA
WHERE COL002 = '00000001'
AND COL004 = '00000001'
  
```

Non-indexed
2nd part key

DPSI key

- V8, page range screening only applies to leading limit key(s)
 - ▶ 1000 DPSI parts must be probed
- V9, since only COL002 = '00000001' is required,
 - ▶ page range screening can be applied on 2nd limit key,
 - ▶ only 20 DPSI parts are probed (1 in every 50 parts)
 - 16X performance improvement in DB2 9 Redbook example

DB2 9 – Histogram statistics

- RUNSTATS
 - ▶ Maximum 100 quantiles for a column
 - ✓ The number of quantiles are adjusted automatically
 - ▶ Quantiles will be similar size but:
 - ✓ Will try to avoid big gaps inside quantiles
 - ✓ Null WILL have a separate quantile
- Supports column groups as well as single columns
- Think “frequencies” for high cardinality columns



DB2 9 – Histogram statistics, an example

- Suppose INTEGER is used to represent YEAR-MONTH

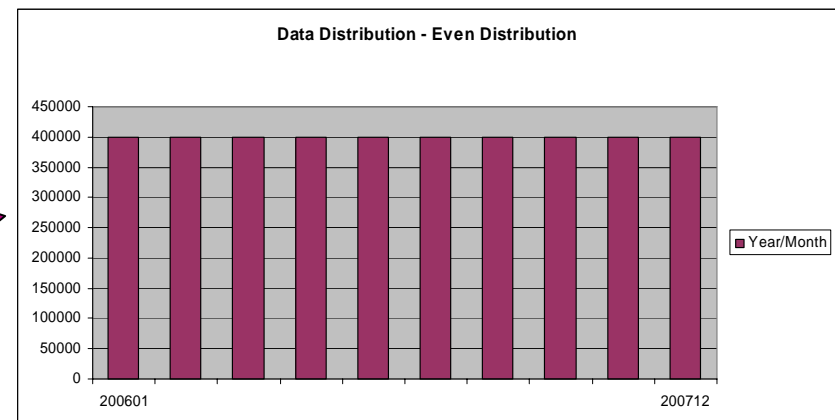
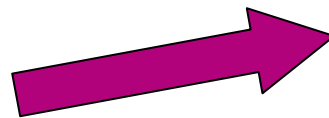
For a predicate

YEARMONTH BETWEEN 200601 AND 200612

Assuming data for 2006 & 2007

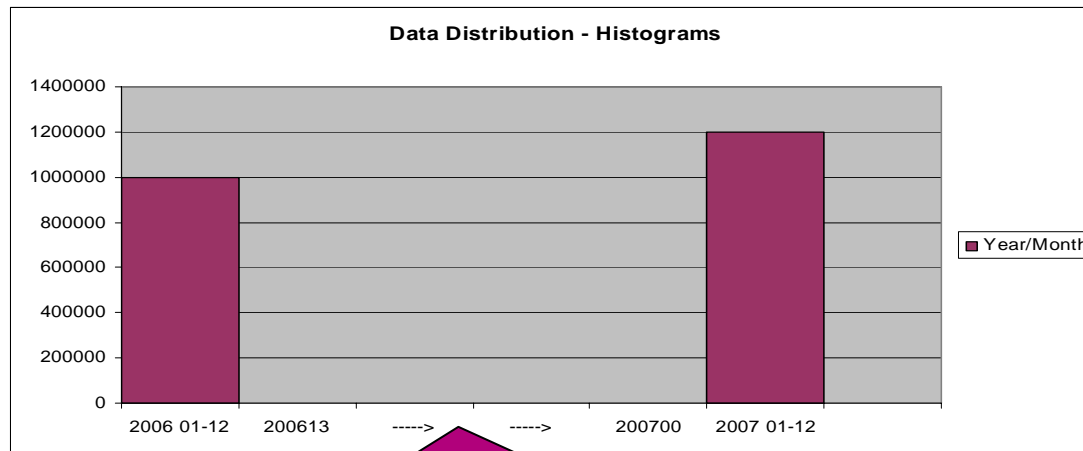
- $FF = (\text{high-value} - \text{low-value}) / (\text{high2key} - \text{low2key})$
- $FF = (200612 - 200601) / (200711 - 200602)$
- **10% of rows estimated to return**

Data assumed as evenly distributed between low and high range



DB2 9 – Histogram statistics, an example (cont.)

- ▶ Data only exists in ranges 200601-12 & 200701-12
 - Collect via histograms
 - 45% of rows estimated to return (more accurate)

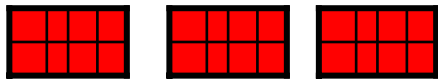


No data between
200613 & 200700

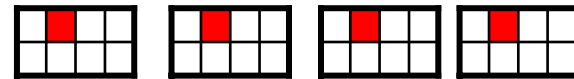
WHERE YEARMONTH BETWEEN 200601 AND 200612

DB2 9 – Improved Index Cluster Ratio

- New Clusterratio formula in DB2 9
 - ▶ Better awareness of prefetch range
 - ▶ More accurate CR for lower cardinality indexes
 - ▶ DB2 9 adds new statistic collected by RUNSTATS
 - DATAREPEATFACTOR differentiates density and sequential



Dense (and sequential)



Sequential (not dense)

Recommend RUNSTATS before mass REBIND in DB2 9

- A new zparm STATCLUS controls the formulas
- STATCLUS = ENHANCED to use the new formula (the default)
 - STATCLUS = STANDARD to use the V8 formula

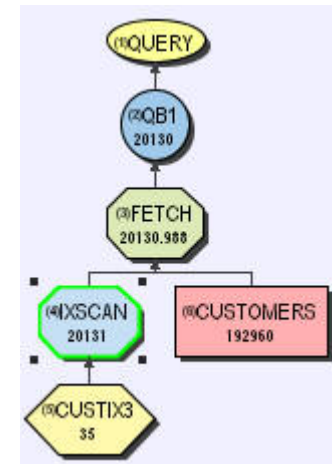
DB2 9 – Index on Expression

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

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

```
CREATE INDEX CUSTIX3 ON
CUSTOMER
(YEAR(BIRTHDATE) ASC)
```

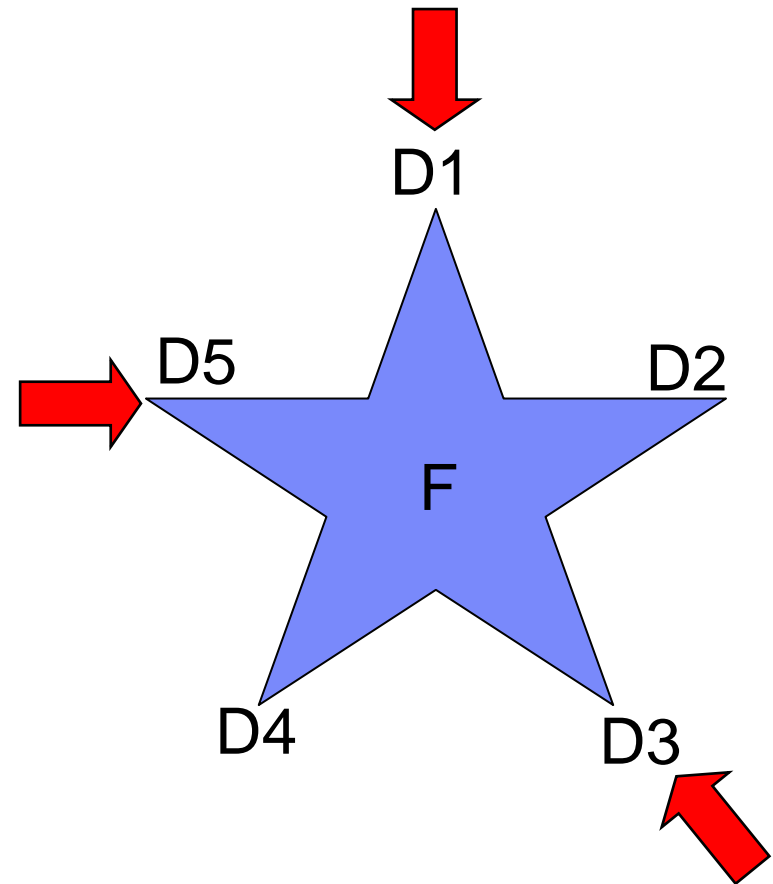
Previous FF = 1/25 (default)
 Now, RUNSTATS collects frequencies
 on the new index.
 → Improved FF accuracy.



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

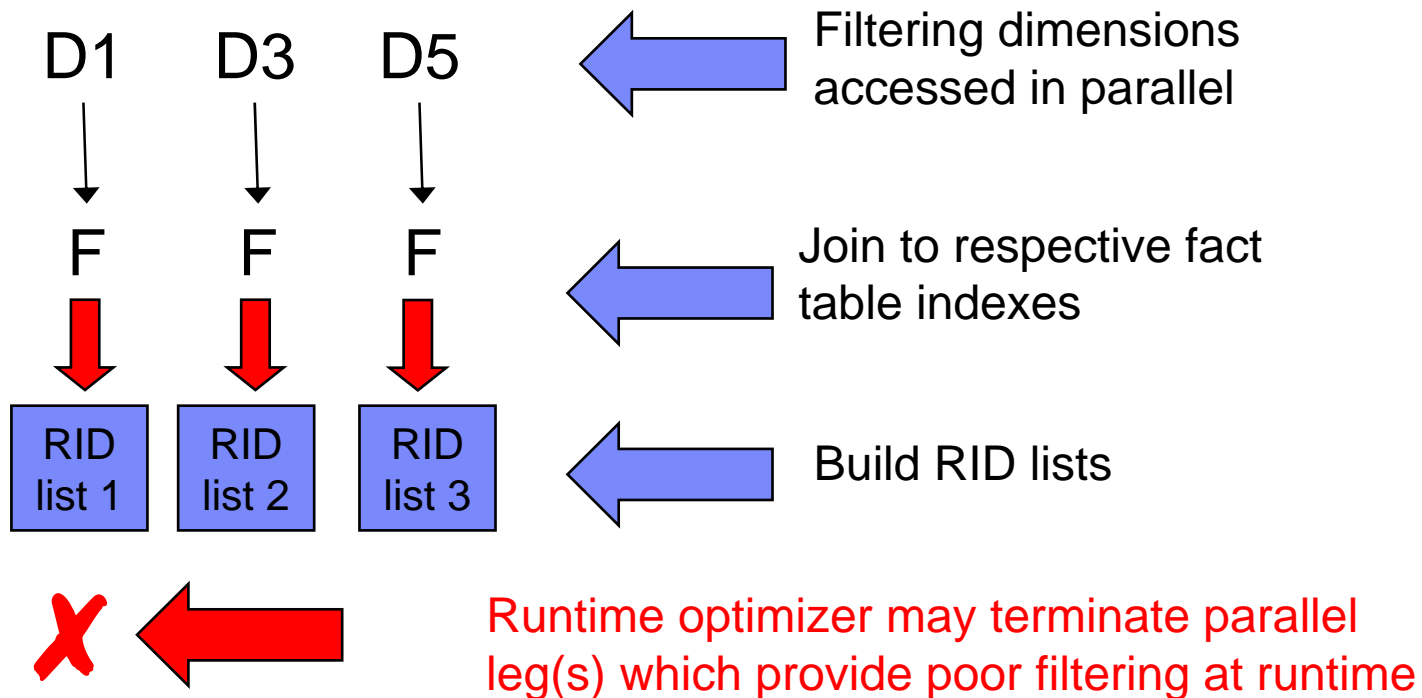
DB2 9 – Dynamic Index ANDing (star join)

- Complexity in a “star shaped” joins
 - ▶ Filtering can come from multiple tables
 - ▶ Current multi-index access does not support this situation directly
 - Will need Cartesian joins of multiple tables → can be a costly operation
- What is “dynamic index ANDing” ?
 - ▶ Perform pre-joins between individual filtering tables to collect RID lists
 - ▶ Apply ANDing on the RID lists
 - ▶ Access the center table F through the intersection of the RIDs
 - ▶ Join back the tables to retrieve data, if needed.



DB2 9 – Dynamic Index ANDing (star join)

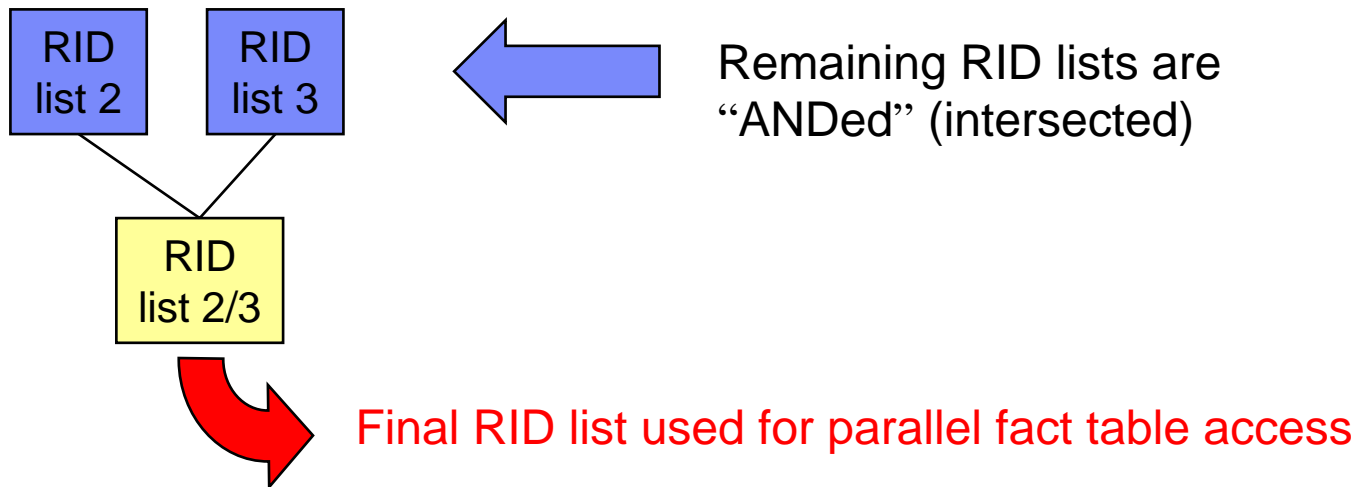
- Pre-joins to the FACT table
 - ▶ At runtime, exact filtering is known.



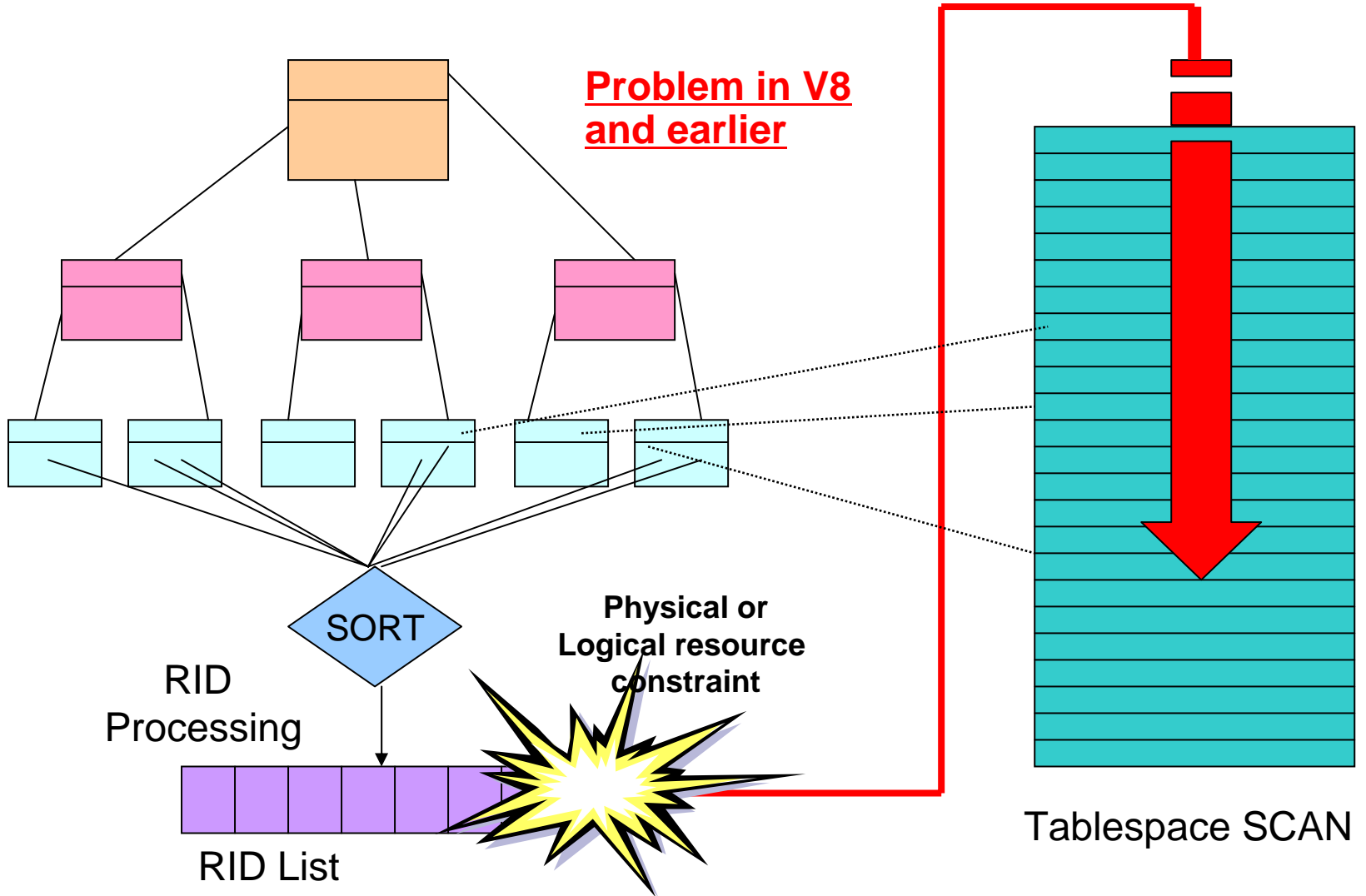
DB2 9 – Dynamic Index ANDing (star join)

- Fact table access
 - ▶ Intersect filtering RID lists
 - ▶ Access fact table ← from RID list
- Post fact table
 - ▶ Join back to dimension tables

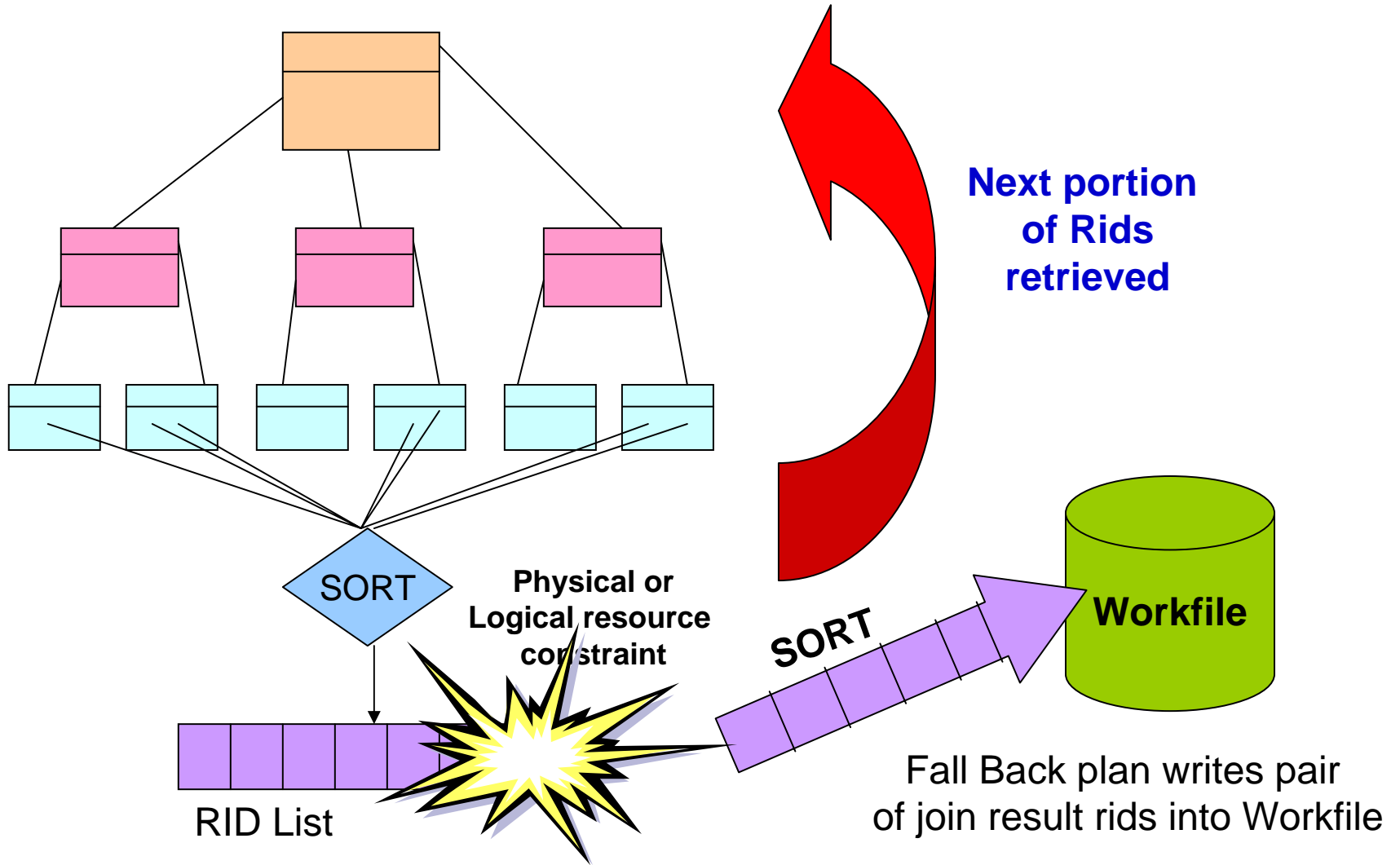
} Using parallelism



DB2 9 – Improved RID list processing (star join)



DB2 9 – Improved RID list processing (star join)



DB2 9 – Other enhancements

- Sequential access
 - ▶ Sequential prefetch only used for table space scan in DB2 9
 - ▶ For index access, only dynamic prefetch or list prefetch
 - ✓ Decision at run time by sequential detection
 - ✓ EXPLAIN output (PREFETCH in PLAN_TABLE) can still show 'S'
 - ▶ PK70398 Sequential detection enhancements
- Parallelism enhancements
 - ▶ In DB2 9, the access path with the minimum cost after parallelism is evaluates on multiple access paths
 - ▶ Determination of the degree in a parallel group can be on a non-leading table (V8 and earlier → determined on the leading table)
 - ▶ Key range for each degree on NPI is determined on histogram, if available → more even distribution of work



DB2 9 – Other enhancements

- Sort avoidance
 - ▶ Non-unique index can be used for DISTINCT
 - ▶ More index support for GROUP BY
 - ✓ An index X(C2,C1) can be used to support GROUP BY C1, C2
 - Without ORDER BY, the order of rows returned can change
 - SQL standard goes not guarantee an order with GROUP BY
- Sort improvements
 - ▶ Final sort step requiring 1 page will not allocate a work file
 - ▶ With FETCH FIRST n ROWS, if the requested “n” rows fits in a 32KB page,
 - ✓ Only the top “n” rows are kept in the order → in-memory
 - ✓ V8 and prior → the entire rows are sorted and stored in a work file, then the top “n” rows are returned.



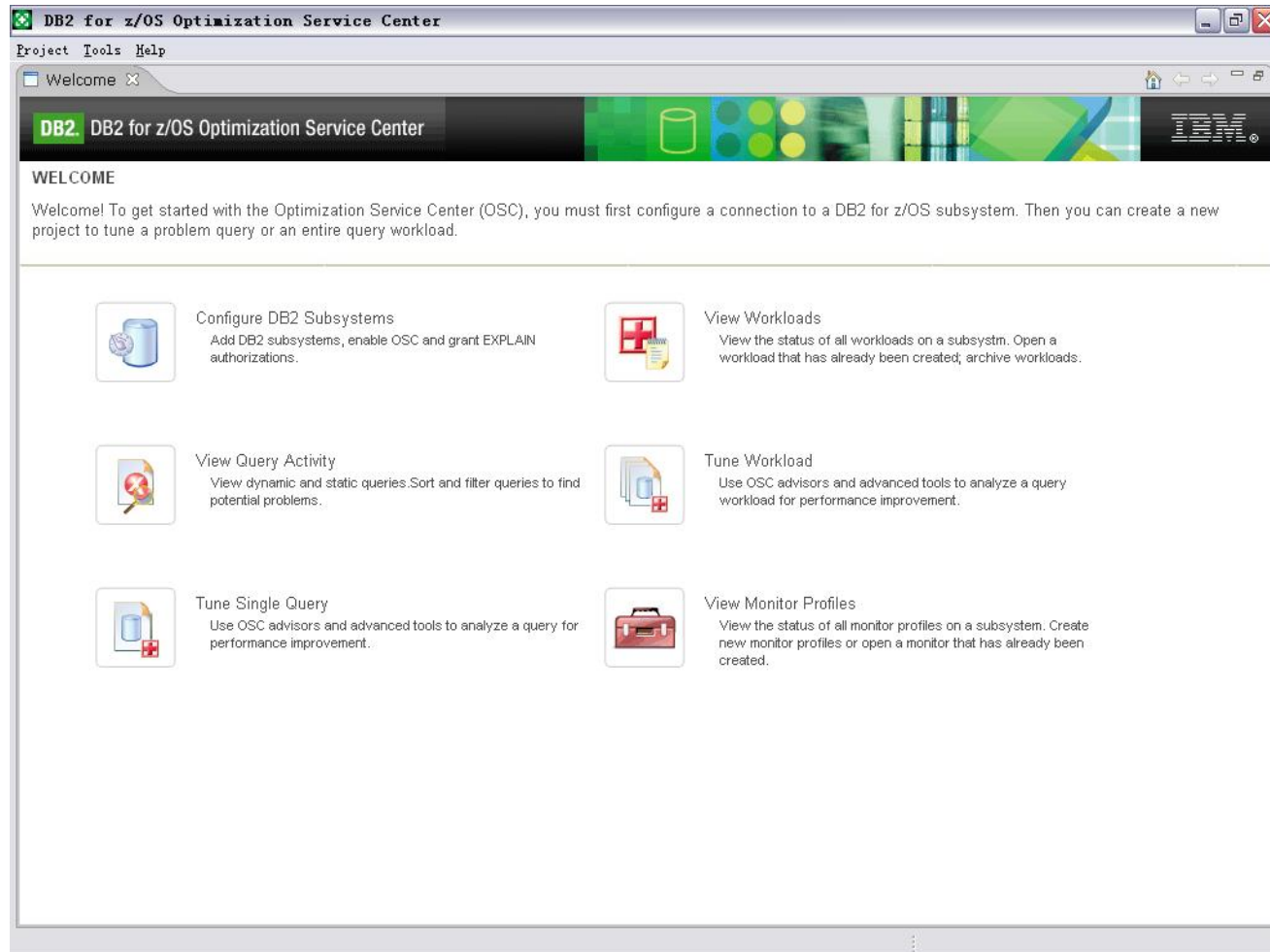
DB2 9 – Optimization Service Center

- V9 provides a more extensive “Optimization Service Center”
 - ▶ All the features of Visual Explain / Statistics Advisor
 - ▶ Single query or workload
 - ▶ Plus query monitor
 - ▶ Advisors – Statistics (OSC), Index (OE) and Query Design (OE)

- V8 compatible also



DB2 9 – OSC Welcome page



Future Visions

- More plan stability features → usability, availability and manageability
 - ▶ Plan stability for static and dynamic SQL
 - ▶ REOPT ONCE and AUTO for static SQL
 - ▶ Optimization hint by statement
- Optimizer base line → more plan stability with higher performance
 - ▶ Safe optimization (risk management for unknown factors)
 - ✓ Not just the cheapest → consider safer access paths
 - ▶ Statistics on views
 - ▶ New efficient access methods, for example
 - ✓ More predicate matching on an index
 - ✓ More efficient parallelism execution mechanism
- More automation
 - ▶ OSC, OE, DataStudio



THANK YOU

Any question ?

Yoichi Tsuji
tsujiy@us.ibm.com

