



DB2 for z/OS

# What's new in DB2 9 for z/OS for Applications

**Patrick Bossman**

**bossman@us.ibm.com**

**Senior software engineer**

**IBM Silicon Valley Lab**



# Disclaimer

© Copyright IBM Corporation [current year]. 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](http://www.ibm.com/legal/copytrade.shtml)

# Agenda

- **Plan Stability**
- **REOPT AUTO**
- **Histogram Statistics**
- **Global Query Optimization**
- **Generalized sparse index and in-memory data cache**
- **Dynamic Index ANDing**
- **Indexing Enhancements**
- **Optimization Service Center**
- **Misc Optimization Enhancements**



# Access Path Stability

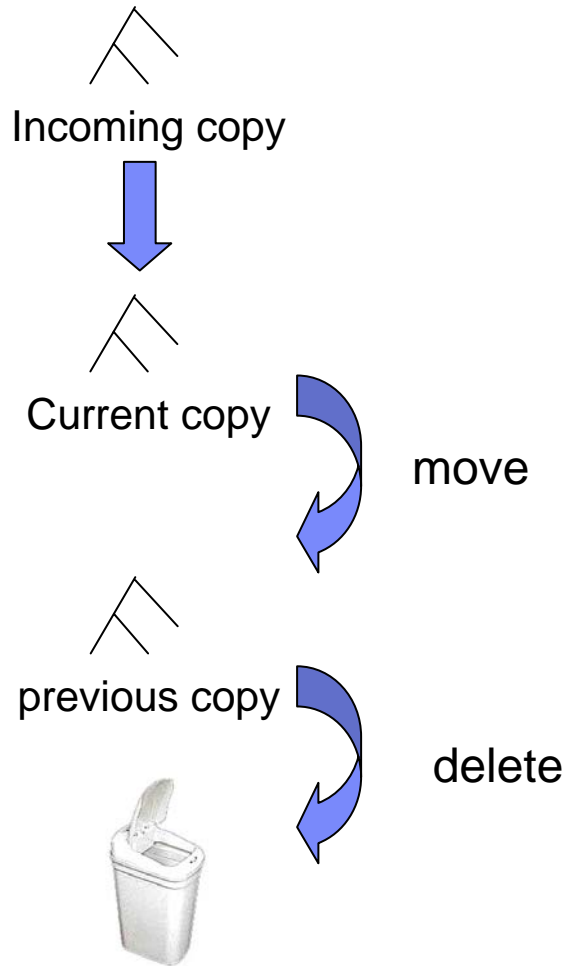


# Plan Stability Overview

- **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)

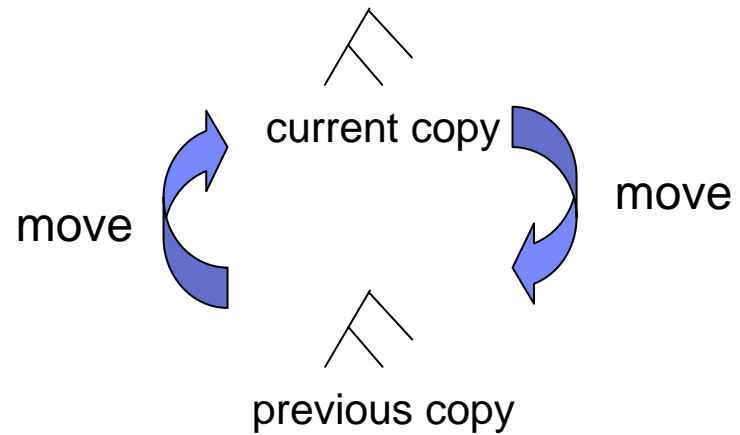
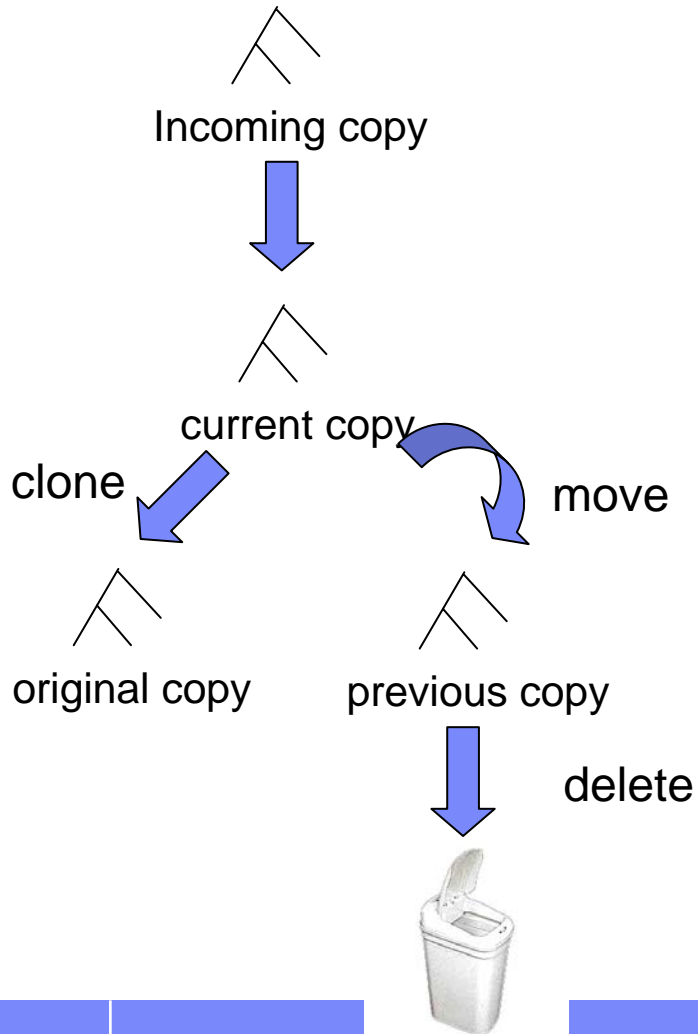


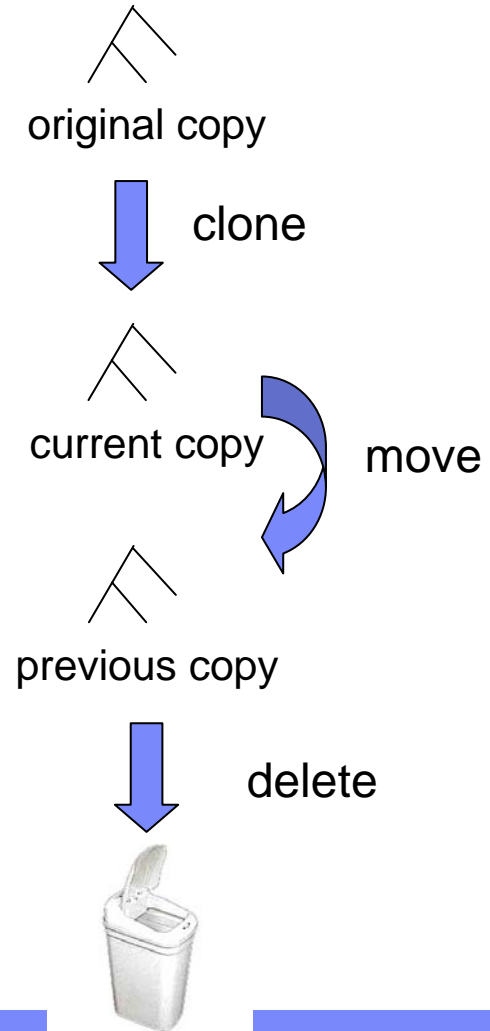
Chart is to be read from bottom to top

# 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



# 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



# Histogram statistics





# 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

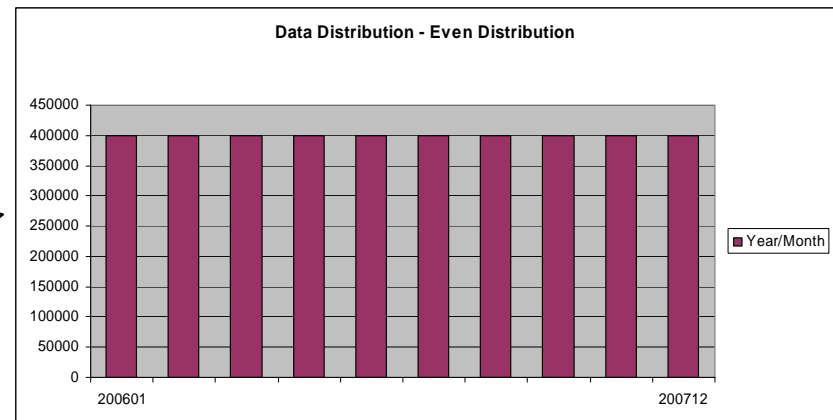
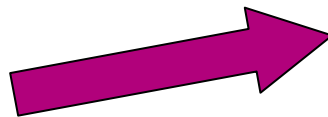
## Histogram Statistics Example

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

WHERE YEARMONTH BETWEEN 200601 AND 200612

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

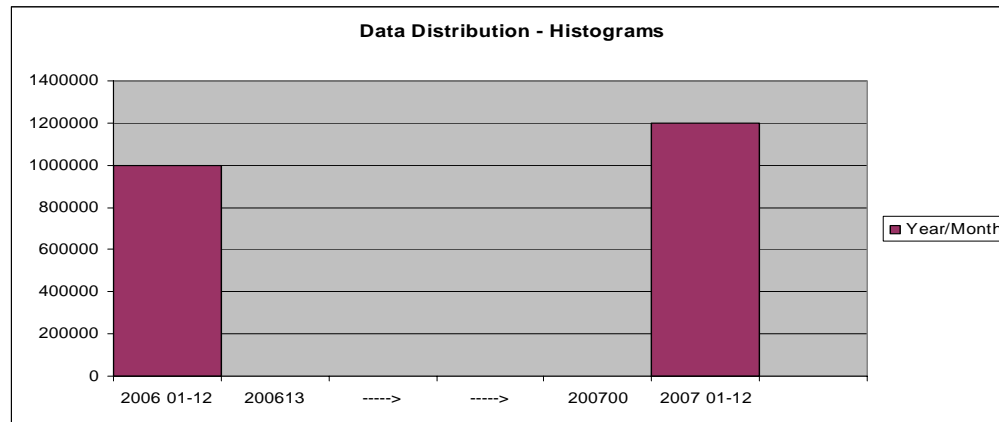
Data assumed as evenly distributed between low and high range



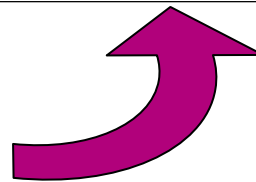
# Histogram Statistics Example

## ■ 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
```



# Global Query Optimization



## Problem Scenario 1

- **V8, Large Non-correlated subquery is materialized\***

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

\* Assumes subquery is not transformed to join

- “BIG\_TABLE” is scanned and put into workfile
- “SMALL\_TABLE” is joined with the workfile

- **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 A
WHERE EXISTS
      (SELECT 1 FROM BIG_TABLE B WHERE B.C1 = A.C1)
```

## Problem Scenario 2

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

```
SELECT * FROM BIG_TABLE A
```

```
WHERE EXISTS
```

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

- “BIG\_TABLE” is scanned to obtain A.C1 value
- “SMALL\_TABLE” gets matching index access

\* Assumes subquery is not transformed to join

- **V9 may rewrite correlated subquery to non-correlated**

```
SELECT * FROM BIG_TABLE A
```

```
WHERE A.C1 IN
```

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

- “SMALL\_TABLE” scanned and put in workfile
- Allows more efficient matching index access on BIG\_TABLE

## Virtual Tables

- **A new way to internally represent subqueries**
  - Represented as a Virtual table
    - Allows subquery to be considered in different join sequences
    - May or may not represent a workfile
  - Apply only to subqueries that cannot be transformed to joins

Correlated or non-correlated?.....I shouldn't have to care!

## EXPLAIN Output

- **Additional row for materialized “Virtual Table”**
  - Table type is "W" for "Workfile".
    - Name includes an indicator of the subquery QB number
      - Example → “DSNWF(02)”
  - Non-materialized Virtual tables will not be shown in EXPLAIN output.
- **Additional column PARENT\_PLANNO**
  - Used with PARENT\_QBLOCKNO to connect child QB to parent
  - V8 only contains PARENT\_QBNO
    - Not possible to distinguish which plan step the child tasks belong to.



# Generalizing Sparse Index and In-Memory Data Cache

## Pre-V9 Sparse Index & in-memory data cache

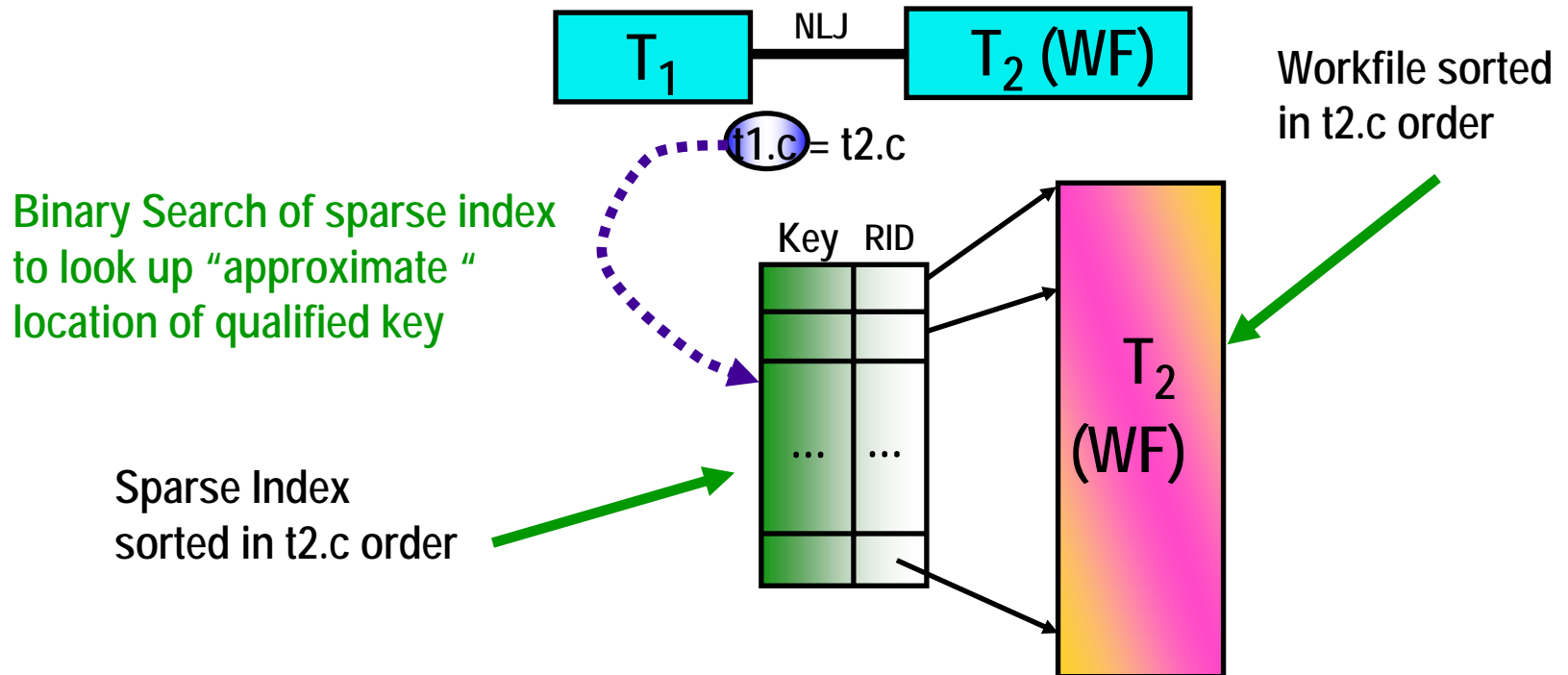
- **V4 introduced sparse index**
  - for non-correlated subquery workfiles
  
- **V7 extended sparse index**
  - for the materialized work files within star join
  
- **V8 replaced sparse index**
  - with in-memory data caching for star join
    - Runtime fallback to sparse index when memory is insufficient

## 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

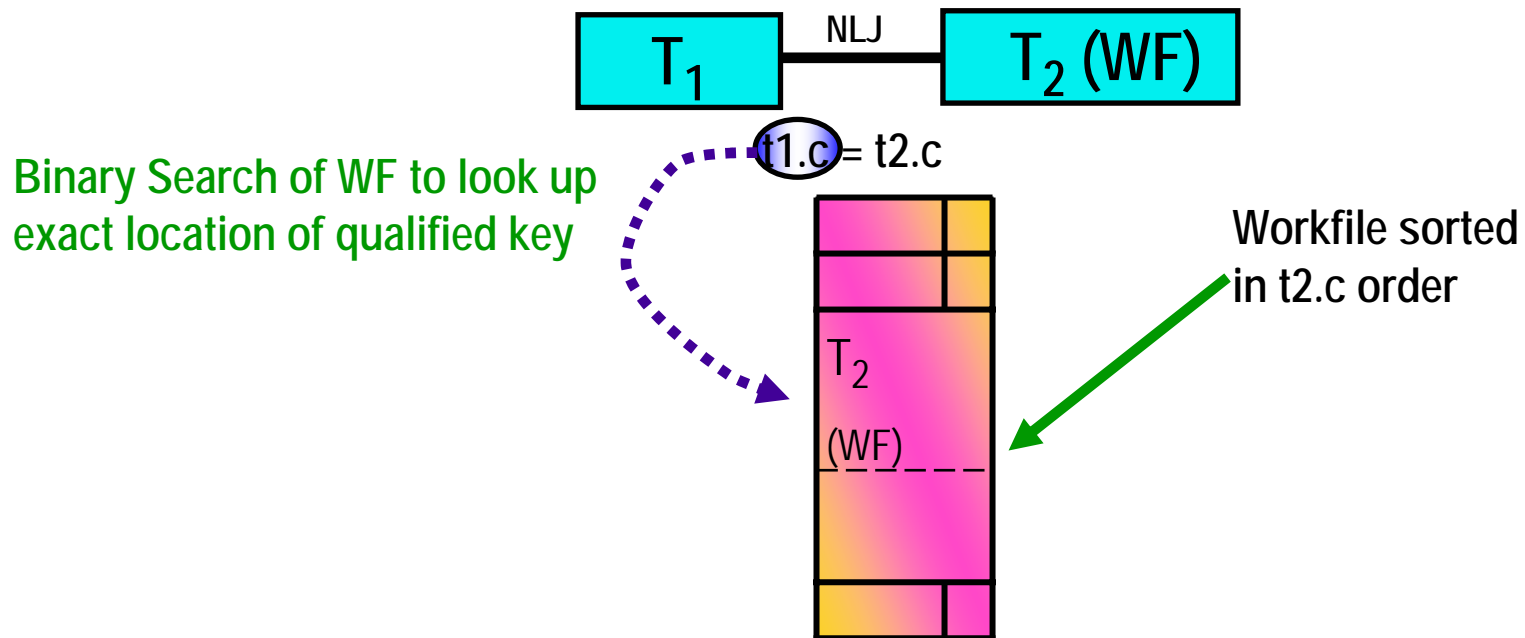
- Extended to non-star join in DB2 9

## ■ 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**

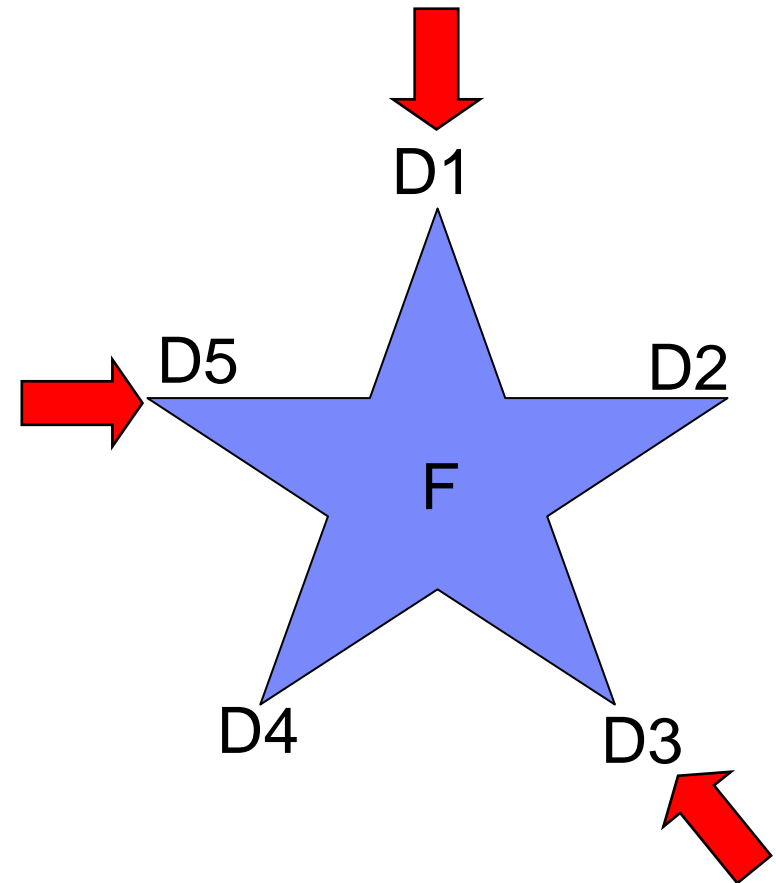


# Dynamic Index ANDing for Star Schema



# Dynamic Index ANDing Challenge

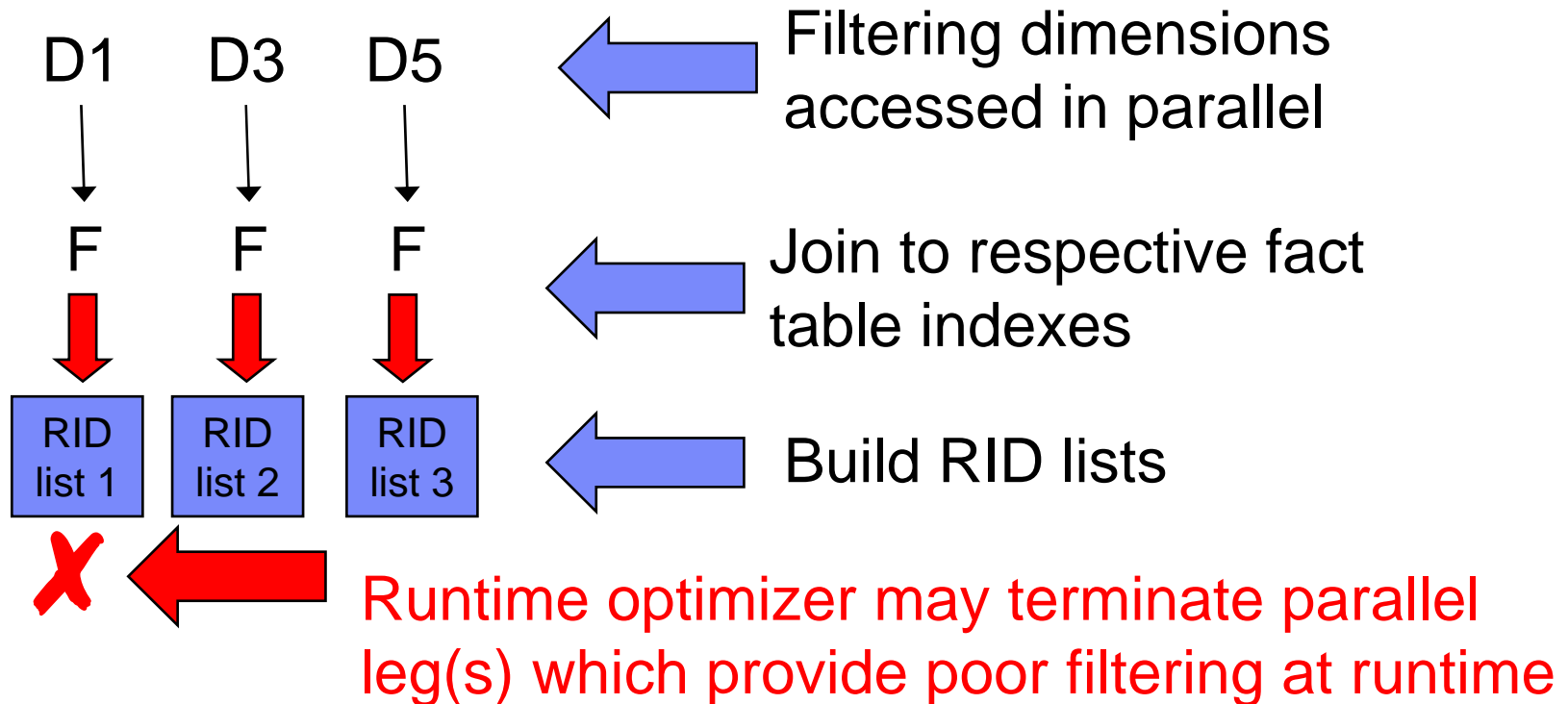
- **Filtering may come from multiple dimensions**
  - Creating multi-column indexes to support the best combinations is difficult



## Index ANDing – Pre-Fact

- **Pre-fact table access**

- Filtering may not be (truly) known until runtime



## Index ANDing – Fact and Post-Fact

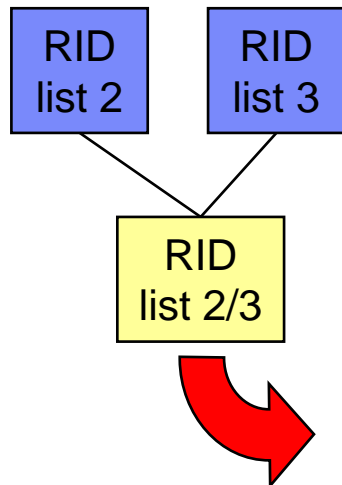
### ■ Fact table access

- Intersect filtering RID lists
- Access fact table
  - From RID list

Using parallelism

### ■ Post fact table

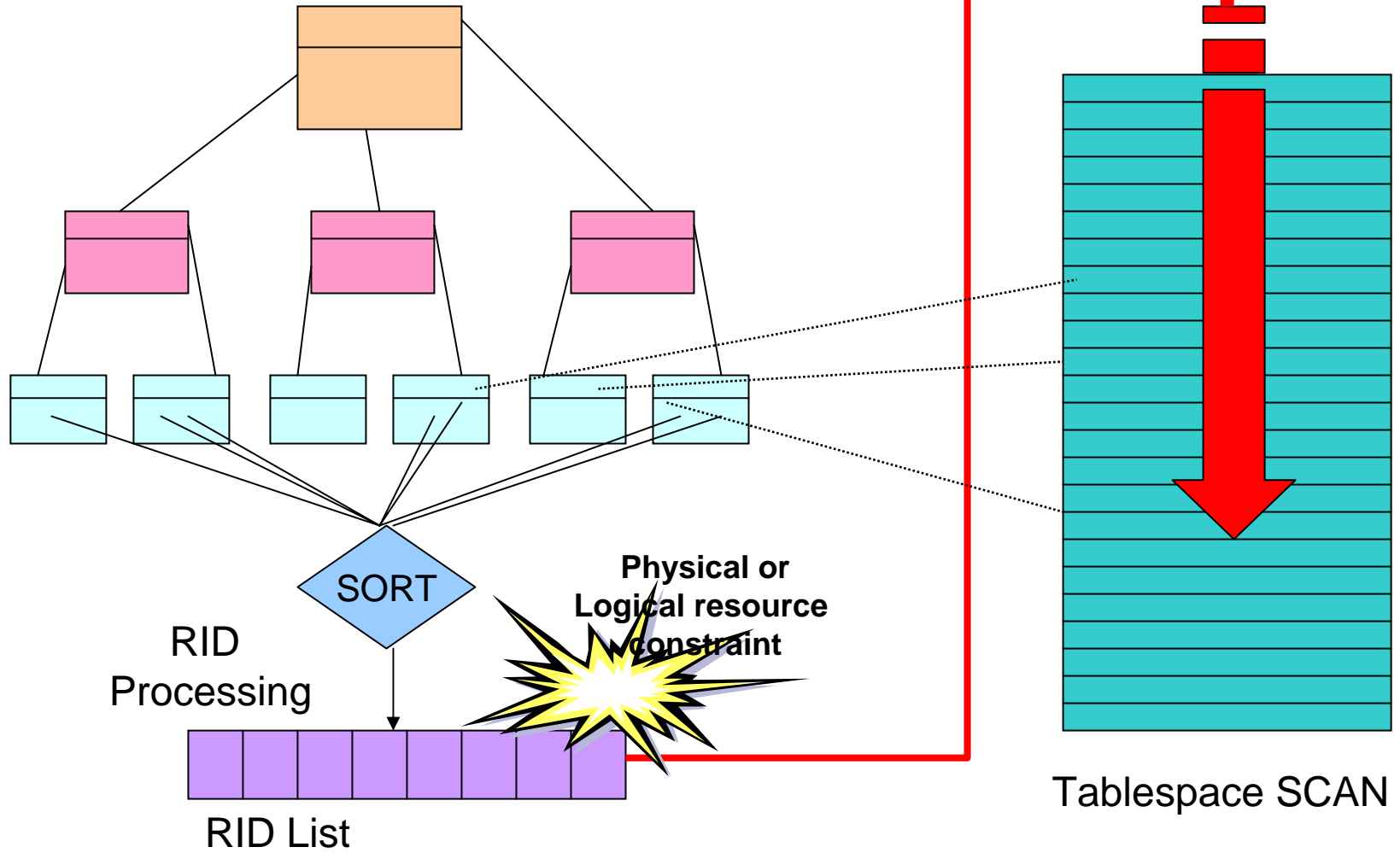
- Join back to dimension tables



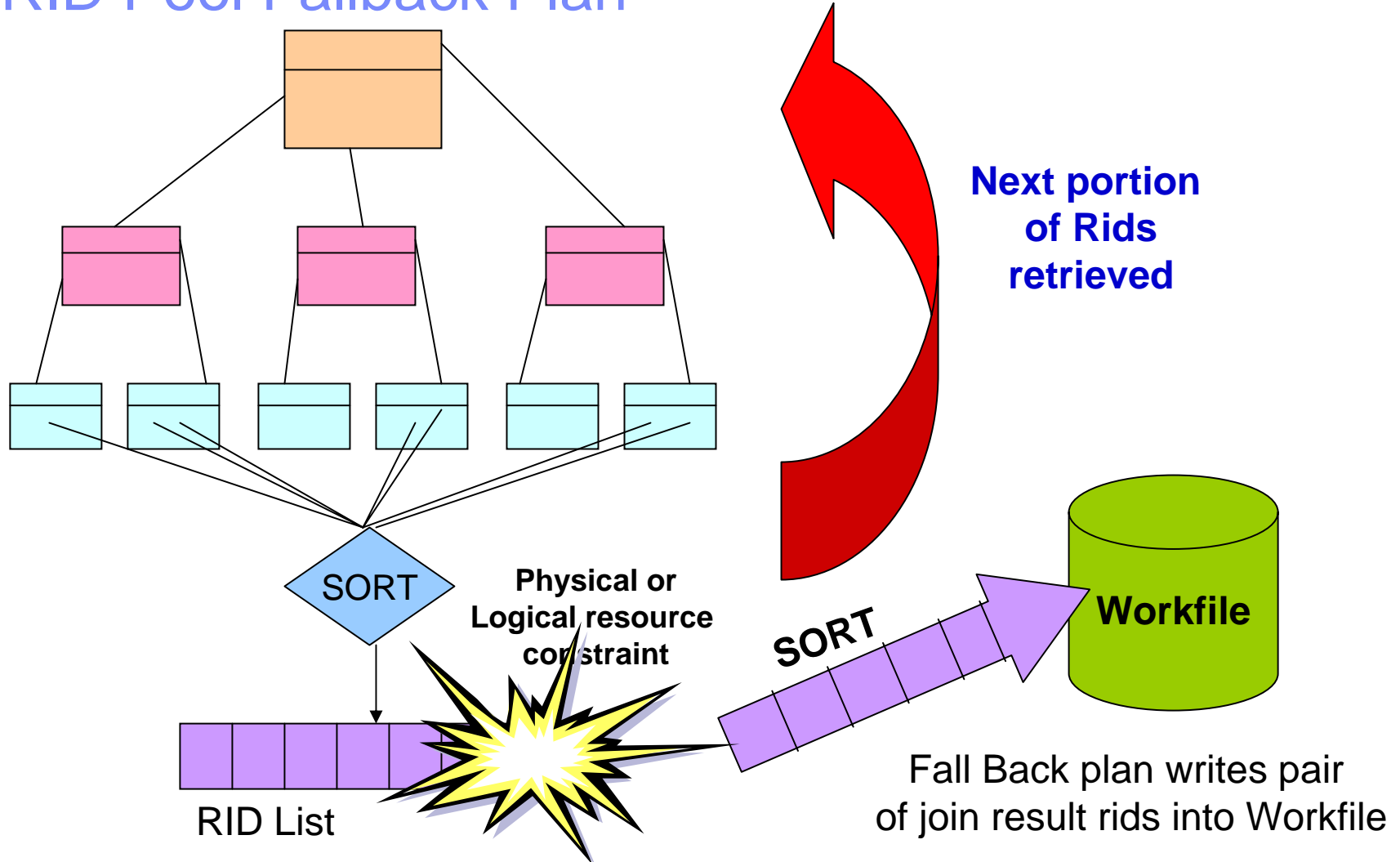
Remaining RID lists are  
“ANDed” (intersected)

Final RID list used for parallel fact table access

# V8 RID Pool failure = TS Scan



# V9 RID Pool Fallback Plan



## Dynamic Index Anding Highlights

- **Pre-fact table filtering**
  - Filtering dimensions accessed concurrently
  
- **Runtime optimization**
  - Terminate poorly filtering legs at runtime
  
- **More aggressive parallelism**
  
- **Fallback to workfile for RID pool failure**
  - Instead of r-scan



# Indexing Enhancements



## Index on Expression

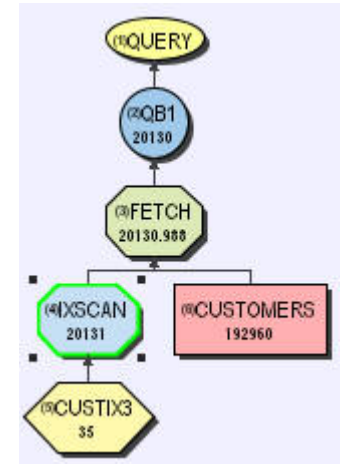
- **DB2 9 supports “index on expression”**

- 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)
```

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

## 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



# Optimization Service Center

## Optimization Service Center (OSC)

- **V9 provides enhanced “Optimization Service Center”**
  - All the features of Visual Explain
    - Single query or workload
    - Plus
      - query monitor
      - Workload Statistics Advisors
      - Query Annotation
      - Improved Query reports
      - Visual plan hint

## VE → OSC Feature comparison

<b>Functions</b>	<b>Visual Explain</b>	<b>Optimization Service Center</b>	<b>Requires DB2 9?</b>
<b>Queries from Cache, Catalog Query Formatter, Annotation</b>	Yes	Yes	
<b>Access Plan Graph</b>	Yes	Yes	
<b>Visual Plan Hint</b>		Yes	
<b>Query Statistics Advisor</b>	Yes	Yes	
<b>Workload Statistics Advisor</b>		Yes	
<b>Profile based Monitoring</b>		Yes	Yes

**DB2.** DB2 for z/OS Optimization Service Center**WELCOME**

Welcome! To get started with the Optimization Service Center (OSC), you must first configure a connection to a DB2 for z/OS subsystem. Then you can create a new project to tune a problem query or an entire query workload.

**Configure DB2 Subsystems**

Add DB2 subsystems, enable OSC and grant EXPLAIN authorizations.

**View Workloads**

View the status of all workloads on a subsystem. Open a workload that has already been created; archive workloads.

**View Query Activity**

View dynamic and static queries. Sort and filter queries to find potential problems.

**Tune Workload**

Use OSC advisors and advanced tools to analyze a query workload for performance improvement.

**Tune Single Query**

Use OSC advisors and advanced tools to analyze a query for performance improvement.

**View Monitor Profiles**

View the status of all monitor profiles on a subsystem. Create new monitor profiles or open a monitor that has already been created.

[www-306.ibm.com/software/data/db2/zos/downloads/osc.html](http://www-306.ibm.com/software/data/db2/zos/downloads/osc.html)

## OSC Basic Features

**Query text**

There are several options to tune the selected query. Format or categorization, or cost analysis.

Query ▾ Advisors ▾ Tools ▾ EXPLAIN timestamp:

EXPLAIN options:  Run EXPLAIN

SELECT CUSTNO FROM CUSTOMER WHERE CUSTNO = ?

- Query Annotation
- Access Plan Graph
- Visual Plan Hint
- Query Reports
- Gather Service Information

## OSC Query Annotation

- **DB2 9 OSC will take an unformatted query**
  - And format this, and add costing detail
    - Table cardf, Column cardinalities, FFs etc

Original Transformed

**Query Annotation**

Indicate which annotation to display and customize your view. Selecting a row will highlight all of the relevant rows for predicate will highlight all of the join predicate rows in both of the joined tables. Click Reset Text to return to the origin

Annotation to display:

Formatted Query	Annotation
SELECT ADMF001.CUSTOMERS.CUSTNO	
FROM ADMF001.CUSTOMERS	CARDF=192,960 QUALIFIED_ROWS=2,473.846 NPAGESF=3,860
WHERE ( ADMF001.CUSTOMERS.CITY = ?	COLCARDF=26 MAX_FREQ=76.418% FF=0.038
AND ADMF001.CUSTOMERS.GENDER = ?	COLCARDF=3 MAX_FREQ=(missing) FF=0.333
)	

# OSC Access Plan Graph

Graphically displays the access path

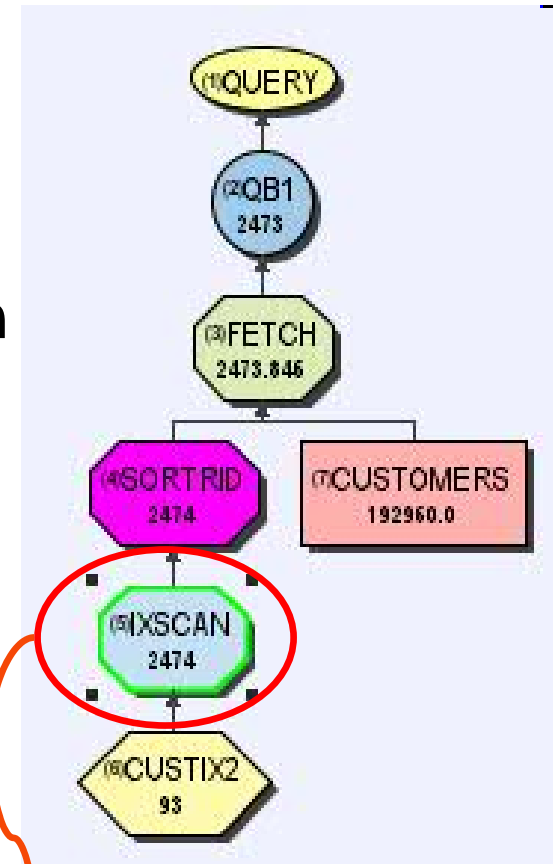
Node Descriptor

Index Scan: IXSCAN

- iscan
  - Matching\_Predicates
    - @ ADMF001.CUSTOMERS.CITY=(EXPR)
  - Screening\_Predicates
    - @ ADMF001.CUSTOMERS.GENDER=(EXPR)

Show attribute explanation Views: cost\_estim

Name	Value
Input RIDs	192960
Index Leaf Pages	473
Matching Predicates	Filter Factor
ADMF001.CUSTOMERS.CITY=(E...	0.0385
Scanned Leaf Pages	19
Screening Predicates	Filter Factor
ADMF001.CUSTOMERS.GENDER...	0.3333
Output RIDs	2474
Total Filter Factor	0.0128
Matching Columns	1



Highlight node to display detail

## Visual Plan Hint

- **Manually altering PLAN\_TABLE for hints is difficult**
  
- **Visual Plan Hint permits**
  - Graphically alter the explain table
  
- **More details included as part of**
  - Optimization hints presentation by Patrick Bossman
    - Google: control your own destiny with optimization hints

## OSC Query Reports

- **Consolidates query tuning data into**
  - Predicate report
  - Table report
  - Index report

### Table Report

```

-----
TABLE_SPACE      NACTIVEF      PARTS          SEGSIZE        PG_SIZE
TPTEST.TPTSTTS1 4680.0         1              0              4
-----
TABLE            CARDF          NPAGESF        TABNO          QUALROWS
ADMF001.CUSTOMERS 192960         3860           1             2473.846
-----
INDEX           CLU   UR  NLEAF      NLEVEL  CR    KEYCOLNAME  COLCARDF  MCARDF
ADMF001.CUSTIX1C Y     U   526        3        1.0  CUSTNO      192960    192960
ADMF001.CUSTIX2  N     D   473        3        0.353 CITY        26         26
                                   STATE        9          31
                                   GENDER       3          93
-----
COLUMN_GROUP  MCARDF
(CITY,STATE) 31.0
-----

```



# Misc Optimization enhancements

# Sort Avoidance Improvements

## ■ Improved Sort avoidance for **DISTINCT**

- From V9, DISTINCT can avoid sort using duplicate index
  - APAR PK71121 – Avoid WF creation for zero rows

## ■ Sort avoidance for **GROUP BY**

- Order of GROUP BY columns re-arranged to match index
  - Data may be returned in a different order
    - Relational theory states that order is NOT guaranteed without ORDER BY

GROUP BY **C2**, **C1** ← GROUP BY in C2, C1 sequence

Index 1 (**C1**, **C2**) ← Index in C1, C2 sequence

# Sort Improvements

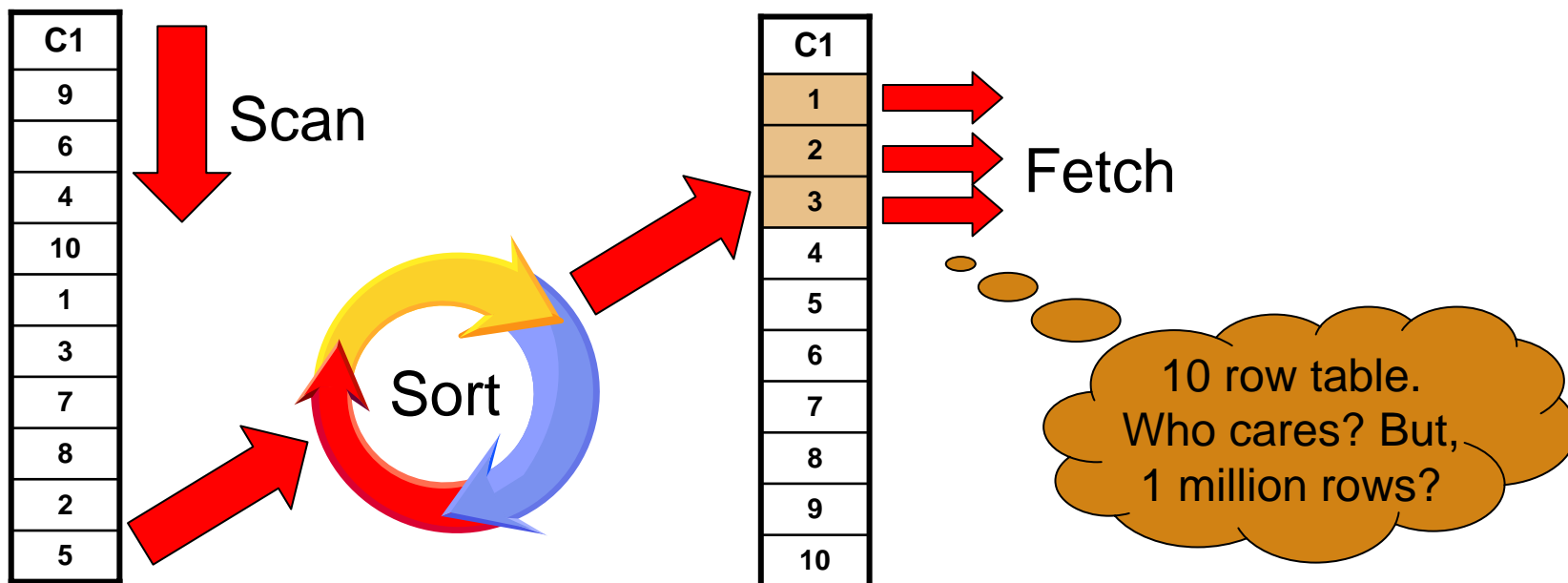
- **Reduced workfile usage for very small sorts**
  - Final sort step requiring 1 page will NOT allocate workfile
  
- **More efficient sort with FETCH FIRST clause**
  - V8 and prior,
    - Sort would continue to completion
    - Then return only the requested 'n' rows
  - From V9,
    - If the requested '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
```



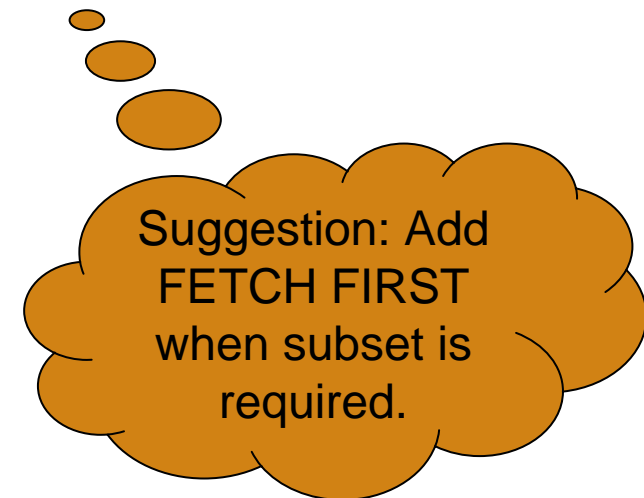
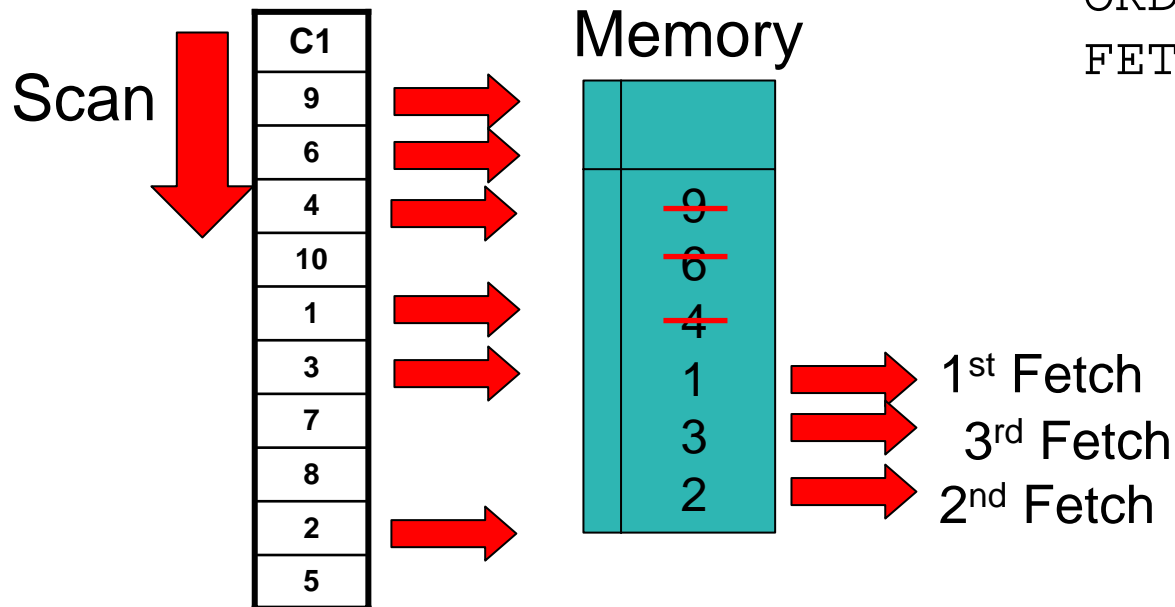
## FETCH FIRST DB2 9 Example

- Sort is not avoided via index

- But in-memory swap avoids sort

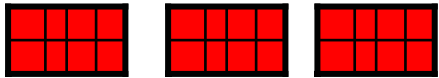
- Pointers maintain order

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

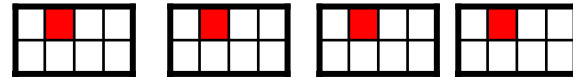


# Clusterratio Enhancement

- **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

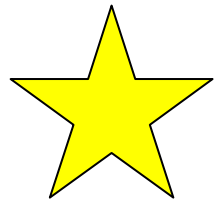


Dense (and sequential)



Sequential (not dense)

- **Recommend RUNSTATS before mass REBIND in DB2 9**



## Sequential Access

- **Sequential prefetch only used for tablespace scan in V9**
  - Dynamic prefetch used instead for other access paths
    - Dynamic prefetch tracks sequential access at runtime
    - Sequential prefetch is based upon bind/prepare prediction
      - At runtime, data may not be page sequential
  - PK70398 Sequential detection enhancements
    - Improved tracking for the clustered range

# Parallelism Enhancements

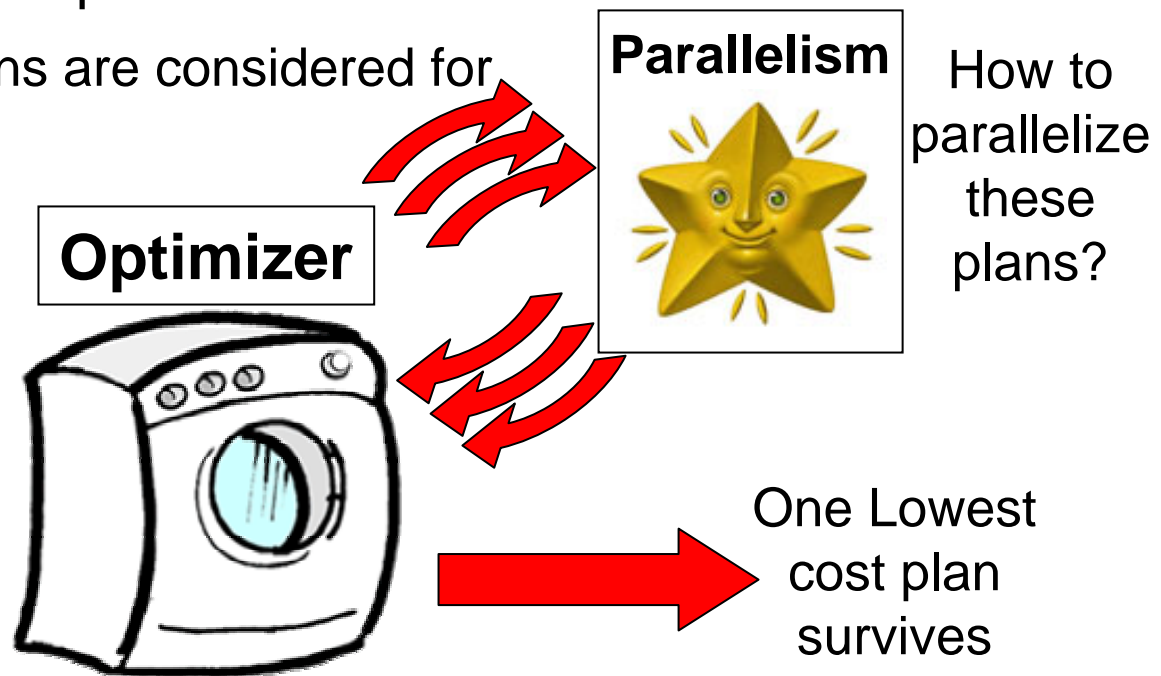
## ■ In V8

–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  $\leq$  # of CPUs \* 4
  - $\leq$  # of CPUs in V8



DB2 for z/OS

# What's new in DB2 9 for z/OS for Applications

**Patrick Bossman**

**bossman@us.ibm.com**

**Senior software engineer**

**IBM Silicon Valley Lab**