Application Optimization in DB2 for z/OS V10

Jim Wankowski
CA Technologies
Agenda

• Real-Time Statistics
• Performance Profiles
• Maintenance for Performance
  • Reorganization
  • RUNSTATS
    • Autonomic Statistics
• Managing Access Paths
  • Statement Level Optimization
  • Profiles and access paths
  • Virtual Indexes
  • Production Modeling
Application Performance

- Many factors can influence application performance
  - SQL (of course!)
  - But also critical:
    - Machine size
    - Subsystem parameters
      - BP
      - RID Pool
      - Sort Pool
    - Database physical design
    - Proper maintenance
- New automated features are making it easier to manage access paths and application performance.
Real-Time statistics

- Real-Time Statistics are gathered in memory for all DB2 objects including catalog tables.

- Eventually externalized to 2 catalog tables:
  - SYSIBM.SYSTABLESPACESTATS
  - SYSIBM.SYSINDEXSPACESTATS
When are Statistics Externalized?

- The RTS manager externalizes statistics at the end of the time interval specified during DB2 installation
  - STATSINT (ZPARM) - default – 30 minutes
- During Utility operations
  - LOAD REPLACE
  - REORG
  - REBUILD INDEX
  - RUNSTATS (UPDATE ALL)
  - COPY
  - RECOVERY
- UPDATE, INSERT, DELETE increments counters
- START or STOP DATABASE commands issued
- STOP DB2 command issued
DSNACCOX

- Sample stored Procedure that makes recommendations to help you maintain your DB2 databases
  - Reads from catalog and statistics tables
  - Recommends when you should reorganize, image copy, or update statistics for table spaces or index spaces
  - Indicates when a data set has exceeded a specified threshold for the number of extents that it occupies.
  - Indicates whether objects are in a restricted state
- V10
  - Improved formulas over DSNACCOR
  - Installed and configured by installation job DSNTIJRT
Establish Baseline Statistics

- Run utilities to establish a baseline for RTS
  - REORG
  - RUNSTATS
  - LOAD REPLACE
  - REBUILD INDEX
  - COPY
- RTS can then calculate deltas
- Improves accuracy of DSNACCOX
Performance Profiles

Profiles enable you to monitor the use of system resources and to control performance-related subsystem parameters in particular contexts on your DB2 subsystem. Each monitored context is defined by a set of criteria called a profile.

- Profiles can:
  - V9
    - Set/disable optimization parameters for SQL statements
    - Model a production environment in test for EXPLAIN processing
    - Set thresholds for Query Acceleration
  - V10
    - Monitor remote threads and connections
    - Manage copies of SQL access paths
Profile Tables

- Created by install job DSNTISG
- **DSNOPTDB.DSNOPTTS**
  - SYSIBM.DSN_PROFILE_TABLE
  - SYSIBM.DSN_PROFILE_ATTRIBUTES
  - SYSIBM.DSN_PROFILE_HISTORY
  - SYSIBM.DSN_PROFILE_ATTRIBUTES_HISTORY
- Indexes
  - SYSIBM.DSN_PROFILE_TABLE_IX_ALL
  - SYSIBM.DSN_PROFILE_TABLE_IX2_ALL
  - SYSIBM.DSN_PROFILE_ATTRIBUTES_IX_ALL
Profile Tables

- **SYSIBM.DSN_PROFILE_TABLE**
  - Profile definition and filtering
  - Insert values based on type of profile being created
  - `PROFILEID` – Unique identifier for profile
  - `PROFILE_ENABLED` column turns profile on/off

- **SYSIBM.DSN_PROFILE_ATTRIBUTES**
  - Controls the attributes of the actions to be applied to an associated profile entry
    - Production Simulation
    - Set/Disable Optimization Parameters
    - Monitor threads
    - Manage Access Paths
Profile Table Filtering

- Profiles for SQL Statement Context
  - Authorization ID and IP address
    - AUTHID,LOCATION
  - Plan name, collection ID, and package name
    - Plannname, COLLID, and PKGNAME
  - Plan Name
    - PLANNNAME
Profile Table Filtering

- **Profiles for System resources**
  - Client IP Address or Domain Name
    - LOCATION
  - Role or Authorization ID
    - ROLE or AUTHID
  - Collection identifier and/or package name
    - COLLID and or PKGNAME
  - Location name, or location alias
    - LOCATION
  - Client application name, user ID, or workstaiton ID
    - CLIENT_APPLNAME, CLIENT_USERID or CLIENT_WRKSTNNAME
Using Profiles to Optimize Subsystem Parameters for SQL

- Profiles can be used to control certain subsystem parameters when executing SQL
- Entries in `SYSIBM.DSN_PROFILE_ATTRIBUTES`

<table>
<thead>
<tr>
<th>Keyword</th>
<th><code>ATTRIBUTE_n</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>NPAGES THRESHOLD</td>
<td>Set <code>ATTRIBUTE2</code> to an integer specifying the page threshold for index access (NPGTHRSIH)</td>
</tr>
<tr>
<td>STARJOIN</td>
<td>Set <code>ATTRIBUTE1</code> to DISABLE or ENABLE Star Join processing (STARJOIN)</td>
</tr>
<tr>
<td>MIN STAR JOIN TABLES</td>
<td>Set <code>ATTRIBUTE2</code> to 3-225 to specify minimum number of tables for star join processing. (SJTABLES)</td>
</tr>
</tbody>
</table>
Creating a Profile

Example

Specify NPAGES threshold for subsystem

- Insert Row in DSN_PROFILE_TABLE

<table>
<thead>
<tr>
<th>PROFILEID</th>
<th>LOCATION</th>
<th>ROLE</th>
<th>AUTHID</th>
<th>PRDID</th>
<th>COLLID</th>
<th>PKGNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>TPX.CA.COM</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
</tr>
</tbody>
</table>

- Insert row in to DSN_PROFILE_ATTRIBUTES

<table>
<thead>
<tr>
<th>PROFILEID</th>
<th>KEYWORDS</th>
<th>ATTRIBUTE1</th>
<th>ATTRIBUTE2</th>
<th>ATTRIBUTE3</th>
<th>ATTRIBUTE_TIMESTAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>NPAGES</td>
<td>Null</td>
<td>100</td>
<td>Null</td>
<td>2012-09-05...</td>
</tr>
</tbody>
</table>
Start/Stop Profiles

- START PROFILE command
  - Starts any profile in DSN_PROFILE_TABLE where PROFILE_ENABLED='Y'

```
PTPDBCDD 16.0 ---- DB2 Command Processor ---- 02-20-13 18:29
COMMAND ==> _

-START PROFILE

DSNT741I !S10A DSNT1SDV START PROFILE IS COMPLETED.
DSN9022I !S10A DSNT1STR 'START PROFILE' NORMAL COMPLETION

******************************************************************************
TOP OF DATA******************************************************************************
******************************************************************************
```

***************************************************************************************
BOTTOM OF DATA ***************************************************************************************
• Verify Profile started
  • -DIS PROFILE

| PTPDBCDD 16.0 ---- DB2 Command Processor ---- 08-06-12 15:37 |
| COMMAND ==> SCROLL ==> PAGE |
| User ID: WANJA03 |
| -DIS PROFILE |

************************************************************************** TOP OF DATA **************************************************************************
DSNT753I !S10A DSNT1DSP DISPLAY PROFILE REPORT FOLLOWS:  
  STATUS = ON  
  TIMESTAMP = 2012-08-06-15.36.46.966532  
  PUSHLIFTS = 0 OUT OF 10000  
  DISPLAY PROFILE REPORT COMPLETE.  
DSN9022I !S10A DSNT1DSP 'DISPLAY PROFILE' NORMAL COMPLETION
************************************************************************** BOTTOM OF DATA **************************************************************************

• -STOP PROFILE command
  • Terminates all active profiles
Profile History

- **SYSIBM.DSN_PROFILE_HISTORY**
  - Contains all of the profiles that were in effect at some point in time
  - Updated by IBM optimization tools

- **SYSIBM.DSN_PROFILE_ATTRIBUTES_HISTORY**
  - Contains the attributes for all the profiles that were in effect at some point in time.
  - Same columns as SYSIBM.DSN_PROFILE_ATTRIBUTES, except for the REMARKS column, which is named STATUS instead.
    - STATUS column indicates whether the profile was accepted, and when a profile was rejected contains information about the reason for the rejection.
HOW MAINTENANCE IMPACTS PERFORMANCE
Keep data Organized

Data needs to be organized in order to obtain optimum access paths and avoid tablespace scans

- What causes data to become fragmented?
  - Insert/Updates
    - Check PCTFREE and FREEPAGE
  - VARCHAR fields being updated
When to Reorganize

Tablespace

- Cluster Ratio < 95%
  - SYSINDEXES
    - CLUSTERRATIOF < 95%
  - SYSINDEXPART
    - FAROFFPOS > 10% of CARD
- Excessive row relocation
  - SYSTABLEPART
    - NEARINDREF+FARINDREF > 10% of CARD
- Excessive extents (>254)
- Excessive drop space
  - Simple TS only
    - PERCDROP > 10%
- LOB tablespaces
  - SYSLOBSTATS
    - ORGRATIO > 2

Index

- Excessive distance between LEAF pages
  - SYSINDEXPART
    - LEAFDIST > 200
    - Can cause Pre-fetch to be disabled
    - Should be monitored for growth
    - LEAFFAR > 10% of NLEAF (SYSINDEXES)
Monitor for Reorgs

• **Real-Time Statistics**
  • DSNACCOR (V9)
  • DSNACCOX (V10)

• **Manual**
  • DSNTESP
    • Set of catalog queries for monitoring various performance issues
    • Member of highlvl.SDSNSAMP
    • Queries 15-21 for reorg

• **REORG Utility**
  • REPORT ONLY option
  • Tablespace Reorg
    • OFFPOSLIMIT = (NEAROFFPOSF + FAROFFPOSF) × 100 / CARDF
      • SYSIBM.SYSINDEXPART
    • INDREFLIMIT = (NEARINDREF + FARINDREF) × 100 / CARDF
      • SYSIBM.SYSTABLEPART
  • Index Reorg
    • LEAFDISTLIMIT
Statistics

Accurate statistics are a critical for accurate access path determination.

- **Two types of statistics:**
  - **Access path statistics**
    - Used by BIND/PREPARE in its process of optimization to determine access path
  - **Space**
    - Used to monitor space usage
    - Assist in capacity planning
    - Determine when to reorganize
When to run RUNSTATS?

- After a table is loaded *
- After an index is physically created
- After a table space is reorganized if inline statistics were not collected
- After running extensive updates, deletions, or insertions in a table space
- After running any of the following utilities without collecting inline statistics: RECOVER TABLESPACE, REBUILD INDEX, or REORG INDEX
- Before running REORG with the OFFPOSLIMIT, INDREFLIMIT, or LEAFDISTLIMIT options
- After running the ALTER TABLE ROTATE PARTITION statement run RUNSTATS with REORG.
Rebind After RUNSTATS?

- Did stats change dramatically?
- **General Guidelines:**
  - CLUSTERRATIOD changes to less than 80% (a value of 0.80)
  - NLEAF changes more than 20% from the previous value.
  - NLEVELS changes (only if it was more than a two-level index to begin with).
  - NPAGES changes more than 20% from the previous value.
  - NACTIVEF changes more than 20% from the previous value.
  - The range of HIGH2KEY to LOW2KEY range changes more than 20% from the range previously recorded.
  - Cardinality changes more than 20% from previous range.
  - Distribution statistics change the majority of the frequent column values, or a new value appears in the frequently occurring values.
Automatic Statistics

- Automated real-time statistics collection
  - Requires Administrative Task Scheduler
- New Catalog Tables (V10)
  - SYSIBM.SYSTABLES_PROFILES
    - RUNSTATS Profiles
  - SYSIBM.SYSAUTOTIMEWINDOWS
    - Controls batch window for RUNSTATS
  - SYSIBM.SYSAUTOALERTS
    - Alerts generated by ADMIN_UTL_MONITOR
  - SYSIBM.SYSAUTORUNS_HIST
    - Historical runtime info for UTL procedures
RUNSTATS Profiles (V10)

• A saved set of options for the RUNSTATS utility that apply for a particular table. Required for Autonomic RUNSTATS

• Options stored in PROFILE_TEXT column in SYSIBM.SYSTABLES_PROFILES
  - COLUMN
  - COLGROUP
  - FREQVAL
  - COUNT
  - MOST
  - BOTH
  - LEAST
  - INDEX
  - KEYCARD
  - NUMCOLS
  - COUNT
  - HISTOGRAM
  - NUMQUANTILES
Managing RUNSTATS Profiles

- ADMIN_UTL_MONITOR will create a profile for every table monitored based off of existing statistics.
- Managing profiles with RUNSTATS
  - RUNSTATS syntax:
    - SET PROFILE
      - Specify options
    - SET PROFILE FROM EXISTING STATS
    - UPDATE PROFILE
    - DELETE PROFILE
Enabling Automatic Statistics Collection

• Series of Stored Procedures to monitor for alerts and schedule RUNSTATS
  • SYSPROC.ADMIN_UTL_MONITOR
  • SYSPROC.ADMIN_UTL_EXECUTE
  • SYSPROC.ADMIN_TASK_ADD
  • SYSPROC.ADMIN_UTL_MODIFY
Administrative Task Scheduler

- Started task, activated at DB2 startup
  - Calls `ADMIN_UTL_MONITOR` as scheduled for statistics monitoring.
    - Calls the `ADMIN_UTL_EXECUTE` to solve alerts according to tasks added by `ADMIN_UTL_MONITOR` and `ADMIN_UTL_EXECUTE` stored procedures.
  - Calls `ADMIN_UTL MODIFY` as scheduled to remove old alert history information from the SYSIBM.SYSAUTOALERTS catalog table.
Enabling Automatic Statistics Collection cont.

- **SYSPROC.ADMIN_UTL_MONITOR**
  - Analyze existing catalog stats and generates alerts if out of date or incomplete
  - Adds entries into
    - SYSIBM.SYSAUTOALERTS
    - SYSIBM.SYSAUTORUNS_HIST
    - SYSTABLES_PROFILES
      - Will create profiles based off of catalog stats if one doesn’t exist.
  - Reads RUNSTATS profiles to determine the set of columns and indexes to check
  - Invokes ADMIN_TASK_ADD to schedule ADMIN_UTL_EXECUTE
Enabling Automatic Statistics Collection cont.

- **SYSPROC.ADMIN_U TL_EXECUTE**
  - Invokes RUNSTATS to resolve alerts identified by ADMIN_U TL_MONITOR
    - Reads from SYSIBM.SYSAUTOALERTS
    - Invokes RUNSTATS within batch window
    - Uses the options that are defined in RUNSTATS profiles to invoke the RUNSTATS stored procedure within defined maintenance windows and resolves the problems
Enabling Automatic Statistics Collection cont.

- **SYSPROC.ADMIN_UTL_MODIFY**
  - Cleans Up
    - SYSIBM.SYSAUTORUNS_HIST
    - SYSIBM.SYSAUTOALERTS
Controlling Batch Window

- **SYSIBM.SYSAUTOTIMEWINDOWS**
  - Controls when RUNSTATS allowed to execute
    - `ADMIN_UTIL_EXECUTE`
    - `ADMIN_UTL_EXECUTE` reads from this table

- **Example:**
  Allow Runstats the first day of each month from 12 midnight to 12 noon, allowing 5 parallel tasks:

  ```sql
  INSERT INTO SYSIBM.SYSAUTOTIMEWINDOWS(DB2_SSID, MONTH_WEEK, MONTH, DAY, FROM_TIME, TO_TIME, ACTION, MAX_TASKS)
  VALUES(S10A,'M',NULL,1,'00:00','12:00:00','RUNSTATS',5);
  ```
MANAGING ACCESS PATHS
Optimizer Choices

• Optimization
  • Influenced by catalog statistics
  • Memory
  • Machine size
Monitoring SQL Access Paths

Explain Tables

- **Native Tables**
  - PLAN_TABLE
  - DSN_FUNCTION
  - DSN_STATEMENT

- **Tables used by IBM Tools**
  - DSN_PREDICAT_TABLE
  - DSN_STRUCT_TABLE
  - DSN_PGROUP_TABLE
  - DSN_PTASK_TABLE
  - DSN_FILTER_TABLE
  - DSN_DETCOST_TABLE
  - DSN_SORT_TABLE
  - DSN_SORTKEY_TABLE
  - DSN_PGRANGE_TABLE
  - DSN_VIEWREF_TABLE
  - DSN_QUERY_TABLE
Maintaining Access Paths

- **Plan management policies**
  - enable you to specify whether DB2 saves historical information about the access paths for SQL statements.
  - **OFF**
    - DB2 does not save access path information.
  - **BASIC**
    - DB2 saves information about the current and one additional access paths.
  - **EXTENDED**
    - DB2 saves information about the current and two additional access paths.
Specifying a Plan Management Policy

• **Subsystem Parameter**
  • This value applies to all statements member-wide unless they are overridden by a value specified by one of the other methods.

• **Bind Option**
  • For static or dynamic SQL statements, you can bind or rebind packages and specify the PLANMGMT bind option.

• **Profile Keyword**
  • When you use profiles to specify the plan management policy, you do not need to rebind the package that contains the statements for the policy and scope to apply.
Maintaining Access Path Information

**BIND/REBIND Package (V10)**

- **PLANMGMT(OFF,BASIC,EXTENDED)**
  - Stores access path information
- **APCOMPARE(NONE,WARN,ERROR)**
  - Check if access path will change
- **APREUSE (NONE,NO,ERROR)**
  - Try to reuse access path
- Package must have been created in V9 or later
- **SYSIBM.SYSPACKCOPY**
  - **APRETAINDUP(YES/NO)**
  - Contains relevant package information for static SQL statements for all copies of a package
Check Access Path Before REBIND

- REBIND EXPLAIN(ONLY)
- APREUSE(ERROR)
- APCOMPARE(WARN)

Rebind will maintain existing access path

Index dropped prior to rebind. Rebind would fail

Refer to Plan_Table “REMARKS” column

REMARKS
APREUSE FAILURE (REASON: 13) APCOMPARE FAILURE (COLUMN: ACCESSTYPE)
Reverting to a saved Access Path

- **REBIND PACKAGE**
  - **SWITCH (PREVIOUS,ORIGINAL)**
    - Reverts access path to an older version.
  - **PREVIOUS**
    - The existing active copy takes the place of the previous copy
    - The existing previous copy takes the place of the active copy.
  - **ORIGINAL**
    - The existing active copy replaces the previous copy.
    - The existing previous copy is discarded.
Using Profiles to Maintain Access Paths

- Overrides PLANMGMT and PLANMGMTSCOPE bind options
- SYSIBM.DSN_PROFILE_TABLE
  - Insert rows to identify package

<table>
<thead>
<tr>
<th>PROFILEID</th>
<th>AUTHID</th>
<th>PLANNAME</th>
<th>AUTHID</th>
<th>PRDID</th>
<th>COLLID</th>
<th>PKGNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>*</td>
<td>*</td>
<td>null</td>
<td>null</td>
<td>BIGEMP</td>
<td>BIGEMP</td>
</tr>
</tbody>
</table>

- SYSIBM.DSN_PROFILE_ATTRIBUTES
  - Insert rows to correspond to profile name

<table>
<thead>
<tr>
<th>PROFILEID</th>
<th>KEYWORDS</th>
<th>ATTRIBUTE1</th>
<th>ATTRIBUTE2</th>
<th>ATTRIBUTE3</th>
<th>ATTRIBUTE_TIMESTAMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>PLANMGMT</td>
<td>EXTENDED</td>
<td>NULL</td>
<td>NULL</td>
<td>2012-09-05...</td>
</tr>
</tbody>
</table>
Advanced Optimization

• **Optimization Hints**
  • Used to maintain specific access path
  • Must be turned on at install time
    • OPPTHINTS=YES
  • Need to modify PLAN_TABLE
    • Additional columns
    • Manually update data
Advanced Optimization cont.

• **Statement Level Optimization (V10)**
  • DSN_USERQUERY_TABLE
    • Insert SQL info
      • QUERYNO - Refers back to hint in Plan_table
      • QUERY_TEXT
      • HINT_SCOPE (0,1)
        • 0 – Use text only from QUERY_TEXT
        • 1 – Use additional COLLECTION, PACKAGE, and VERSION info.
BIND QUERY Command

- Creates a hint for every DSN_USERQUERY_TABLE row
- Inserts data into:
  - SYSIBM.SYSQUERY
    - 1 row for each optimization hint
  - SYSIBM.SYSQUERYPLAN
    - 1 row for each hint which enforces an access path
  - SYSIBM.SYSQUERYOPTS
    - 1 row for each hint that specifies optimization parameters.
Index Simulation

- **SYSIBM.DSN_VIRTUAL_INDEXES**
  - Allows simulation of adding or dropping an index.
  - EXPLAIN will show access paths with both virtual and real indexes.
  - Populated by optimization tools
z/OS Production Modeling (V10)

- Need more than production stats
  - Optimizer considers
    - CPU Speed
    - Number of Processors
    - BP Size
    - RID Pool
    - Sort Pool
  - Support optimizer overrides
    - V9 APAR PM26475
    - V10 APAR PM26973
  - New ZPARMs
    - SIMULATED_CPU_SPEED
    - SIMULATED_COUNT
  - SYSIBM.DSN_PROFILE_ATTRIBUTES
    - SORT_POOL_SIZE
    - MAX_RIDBLOCKS
    - Bufferpools
Gather Production Data

- **Copy Prod Stats**
  - Highlvl.SDSNSAMP(DSNTESP)
    - Queries 11-14
- **Buffer Pool allocations**
  - -DISPLAY BUFFERPOOL
- **Obtain CPU speed, number of processors, Rid pool, and sort pool**
  - **MDLPROD.COPY** (Run on PROD)
    - captures the CPU speed, number of processors, Rid pool, and sort pool from the IBM_SERVICE_DATA column in PLAN_TABLE
    - Values converted to integer
Update System Parameters

- Make a copy of DSNTIJUZ and add
  - SIMULATED_CPU_SPEED
    - Captured from previous step
  - SIMULATED_CPU_COUNT
    - Captured from previous step
  - NPGTHRSH
    - Copy from prod environment
  - PARAMDEG
    - Copy from prod environment
  - STARJOIN
    - Copy from prod environment
Create Profile with Production processor info

• **SYSIBM.DSN_PROFILE_TABLE**
  - Create unique PROFILEID
    
    ```sql
    INSERT INTO SYSIBM.DSN_PROFILE_TABLE(PROFILEID)
    VALUES(1234);
    ```

• **SYSIBM.DSN_PROFILE_ATTRIBUTES**

    ```sql
    INSERT INTO SYSIBM.DSN_PROFILE_ATTRIBUTES
    (PROFILEID,KEYWORDS,ATTRIBUTE1,ATTRIBUTE2) VALUES (1234, 'BP0',NULL, 25000);
    INSERT INTO SYSIBM.DSN_PROFILE_ATTRIBUTES
    (PROFILEID,KEYWORDS,ATTRIBUTE1,ATTRIBUTE2) VALUES (1234, 'SORT_POOL_SIZE',NULL, 307200);
    INSERT INTO SYSIBM.DSN_PROFILE_ATTRIBUTES
    (PROFILEID,KEYWORDS,ATTRIBUTE1,ATTRIBUTE2) VALUES (1234, 'MAX_RIDBLOCKS',NULL, 300);
    ```


• **MDLTEST.COPY** (Run on Test)
  - SQL to insert and create profile in DSN_PROFILE_ATTRIBUTES that incorporates production values into test
Start Profile

- Issue `-START PROFILE`
- Verify that it is active:

```sql
SELECT PROFILE_ENABLED FROM SYSIBM.DSN_PROFILE_TABLE
WHERE PROFILEID=4714
```

1 ROWS RETRIEVED
PROFILE_ENABLED
Y

*************************************************************************** BOTTOM OF DATA **
SELECT SUBSTR(KEYWORDS,1,14) KEYWORDS, ATTRIBUTE2, 
SUBSTRSTATUS,1,51) STATUS
FROM SYSEMBM.DSN_PROFILE_ATTRIBUTES_HISTORY PAH
WHERE PAH.ATTRIBUTE_TIMESTAMP =
(SELECT MAX(ATTRIBUTE_TIMESTAMP)
FROM SYSEMBM.DSN_PROFILE_ATTRIBUTES_HISTORY PAH2
WHERE PAH2.PROFILEID = PAH.PROFILEID)
AND PROFILEID = 4714;

<table>
<thead>
<tr>
<th>KEYWORDS</th>
<th>ATTRIBUTE2</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_RIDBLOCKS</td>
<td>300</td>
<td>ACCEPTED</td>
</tr>
<tr>
<td>SORT_POOL_SIZE</td>
<td>307200</td>
<td>ACCEPTED</td>
</tr>
<tr>
<td>BP8K0</td>
<td>2500</td>
<td>ACCEPTED</td>
</tr>
<tr>
<td>BP0</td>
<td>250000</td>
<td>ACCEPTED</td>
</tr>
</tbody>
</table>
Summary

- DB2 9 & 10 introduced a lot of new features to make managing access paths much easier.
- Try introducing these new features into your environments with a few applications.
THANK YOU!
Jim Wankowski
CA Technologies
James.Wankowski@CA.Com
Application Optimization in DB2 for z/OS V10