#SHAREorg

SHARE
Technology · Connections · Results

Explaining EXPLAIN: DB2 10 Edition

Session # 11953
Friday, August 10, 2012: 8:00 AM-9:00 AM

Willie Favero
Dynamic Warehouse on z/OS Swat Team (DB2 SME)
IBM Silicon Valley Laboratory
713-940-1132
wfavero@attglobal.net

my TWITTER
#db2zos

SHARE in Anaheim 2012

Notices

This information was developed for products and services offered in the U.S.A.

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to: IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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.

This information contains 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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

This presentation contains trade-marked IBM products and technologies. Refer to the following Web site:

http://www.ibm.com/legal/copytrade.shtml

What is EXPLAIN?

EXPLAIN captures detailed information about what could impact SQL performance

- or -

It gathers access path selection information

- or -

- A peek at what the optimizer might do! -

EXPLAIN
was introduced in
DB2 Version 1.2
on March 7, 1986

How is it Executed?

Can be dynamically prepared

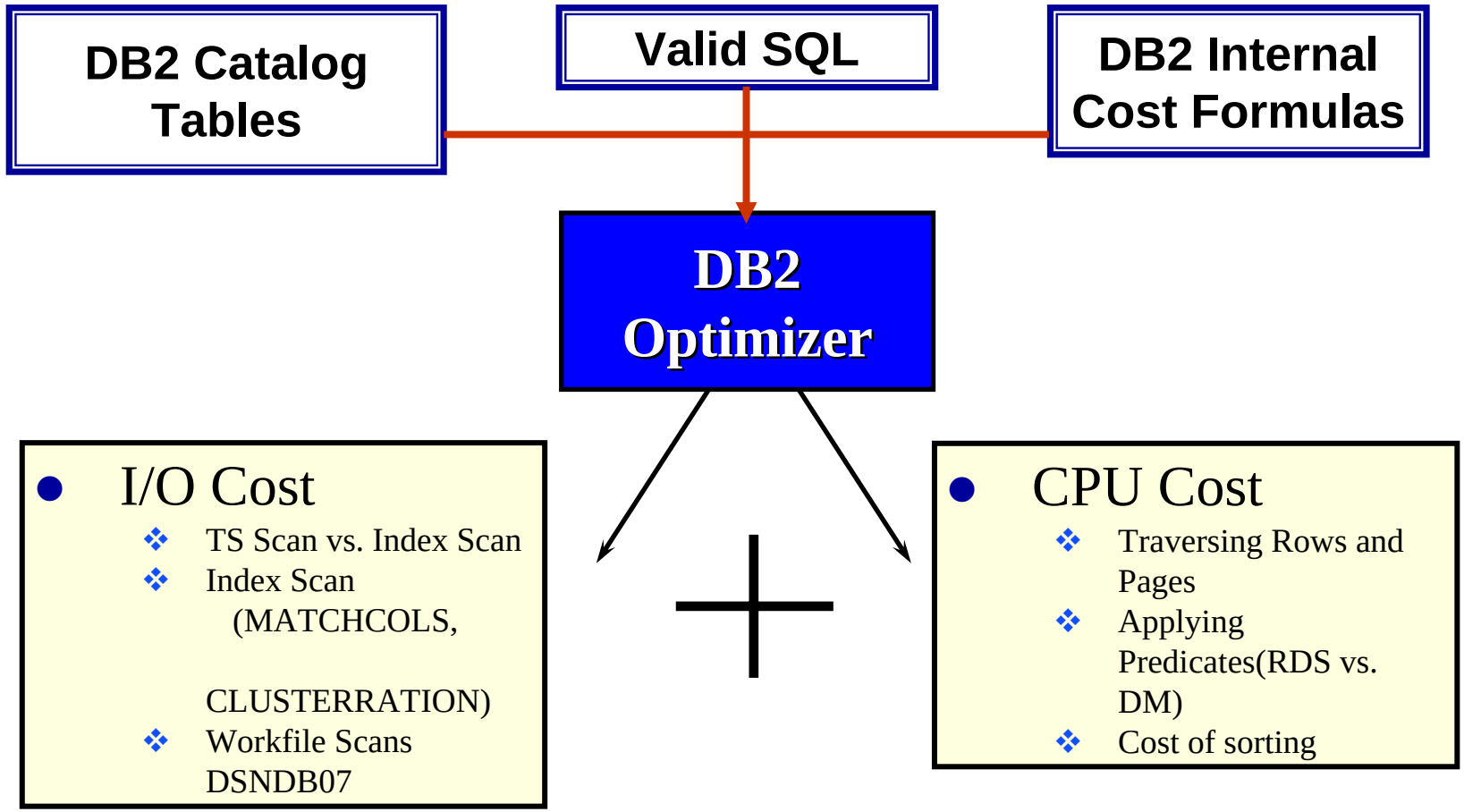
Run as a static SQL statement

BIND/REBIND command option

Automatically at automatic REBIND

Special register

Access Path Selection



= "***Least Cost Access Path***"

Your Output is Only as Good as Your Input

Statistics

RUNSTATS

Inline stats (REORG, LOAD, etc…)

And don't forget about Stats Advisor

SQL EXPLAIN

EXPLAIN PLAN SET QUERY=nnn *sql_statement*

EXPLAIN STMTCACHE ALL

EXPLAIN STMTCACHE STMTID *value*

EXPLAIN STMTCACHE STMTTOKEN *value*

EXPLAIN PACKAGE
COLLECTION *cccccc* PACKAGE *pppppp*

Only works with DB2 9 packages and above.
Available in Conversion Mode (CM)

Plan (Package) Management

- DSNZPARM keyword PLANMGMT
  - Need to specify BASIC or EXTENDED
    - BASIC
      - Access path information in repository
      - One access path retained
    - EXTENDED
      - Access path information in repository
      - Two access paths are retained; previous and original
  - EXPLAIN PACKAGE keyword COPY specifies CURRENT, PREVIOUS, or ORIGINAL should be used
    - HINT_USED contains which one
      - 0 (current copy of package)
      - 1 (previous copy of package)
      - 2 (the original copy of the package)

QUERYNO

- The column that “brings it all together”
  - Of the 18 (DB2 9 has 16 or V8 has 15) EXPLAIN tables, all but 1 has a QUERNO column
- Populated by
  - EXPLAIN SQL statement
  - EXPLAIN ALL
    SET QUERYNO=nnn <<<<< optional but should be used
    FOR .....

QUERYNO

- Populated by
  - When EXPLAIN (YES) or EXPLAIN (ONLY) is specified on the BIND/REBIND commands
    - QUERYNO is SQL statement number in pgm unless
      - Including QUERYNO keyword on SQL statement
      - SELECT ....
FROM .....
WHERE .....
WITH....
QUERYNO nnn
    - 32767 statement limit (is that really a problem)
    - Could simplify using OPTHINTS

Also used as QUERYNO in SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT

EXPLAIN(ONLY)

- Populate EXPLAIN tables without creating or replacing package
  - Available in Conversion Mode (CM)
  - Same cost as bind
  - Do not need the authority to run SQL in package to EXPLAIN package
  - Populate EXPLAIN tables even though EXPLAIN(NO) was originally specified
  - Test access path before replacing
    - Goes well with APREUSE(ERROR) or APCOMPARE(ERROR)

APREUSE(ERROR) – Reuse previous access path. If any statement cannot be reused, bind ends.

APCOMPARE(ERROR) –compare old and new access for each statement. Terminate if different.

EXPLAIN Tip

- DSNZPARM
  - ABEXP keyword on the DSN6SPRM macro
    - Should EXPLAIN processing take place at automatic rebind?
    - YES – Default; perform EXPLAIN processing at auto rebind
      - You want to specify YES
    - NO - no EXPLAIN processing at auto rebind
  - If package bound with EXPLAIN(NO), no EXPLAIN processing takes place at rebind regardless of this ZPARM's setting.
  - If the package was bound with EXPLAIN(YES) and ABEXP is YES EXPLAIN processing still happens during auto rebind even if the PLAN_TABLE doesn't exist, there's just no output produced.

EXPLAIN Special Register

- SET CURRENT EXPLAIN MODE =
  - NO - turns of capturing EXPLAIN information. This is the default
  - YES - turns feature on causing EXPLAIN information to be collected and plan tables to be updated.
  - EXPLAIN - this option is the same of YES. However, the dynamic SQL statements are not executed
  - host_variable – must be CHAR or VARCHAR and can only contain one of the above in uppercase and left justified
- Valid for eligible SQL only
  - SELECT, INSERT, and the searched form of UPDATE and DELETE.

For this one, you'll have to wait for new function mode (NFM) to use

EXPLAIN Privilege

- EXPLAIN is considered a “system privilege” that allows you to grant select privileges without granting execute on the SQL
  - EXPLAIN specifying any of these keywords
    - ALL/PLAN,
    - STMTCACHE ALL,
    - STMTCACHE STMTID,
    - STMTCACHE STMTTOKEN,
    - MONITORED STMTS,
  - Issue the SQL statements DESCRIBE TABLE and PREPARE
  - Specify BIND options EXPLAIN(ONLY) and SQLERROR(CHECK),
  - Explain dynamic SQL statements executing under special register CURRENT EXPLAIN MODE

EXPLAIN Privilege

- Syntax:
  - GRANT EXPLAIN TO …….
  - Can grant on authorization ID, role, or to PUBLIC
  - Can also specify WITH GRANT OPTION

Once again, for this one, you'll have to wait for new function mode (NFM) to use

Populates the PLAN_TABLE?

- EXPLAIN's will go here:
  - PLAN_TABLE
    - Describes access path of SQL state
    - Help better design SQL statem
    - Can be used for optimizati

No wait…
That doesn't seem correct?

BTW, Remember when everyone was trying to get to the hidden EXPLAIN tables?

Populate the PLAN_TABLE?

- Explain's output actually goes three places:
  - PLAN_TABLE
    - Describes access path of SQL stat
    - Help better design SQL statem
    - Can give optimization hint
- Added in DB2 Versio
  - DSN_STATEMN LE
    - Provides co tes
    - Cost in se nits and in milliseconds
    - For dynamic and static SQL statements
  - DSN_FUNCTION_TABLE
    - How DB2 resolves functions
    - One row for each function in an SQL statement

No, Still not correct…

Populate the PLAN_TABLE?

- And then there were 15 :
  - PLAN_TABLE, DSN_STATEMNT_TABLE, DSN_FUNCTION_TABLE
- Tables added in DB2 Versi
  - DSN_STATEMENT_CA ABLE
    - If you have dynamic t caching enabled, the statement cache table conta ormation about any SQL statements in t ent cache as a result of issuing the SQL state LAIN STATEMENT CACHE ALL.
      - You to consider having IFCIDs 316,317,and 318 active for ad al details.
  - DSN_DETCOST_TABLE
    - The detailed cost table contains information about detailed cost estimation of the mini-plans in a query.

No, But getting closer…

Populate the PLAN_TABLE?

- Tables used by EXPLAIN :
  - DSN_FILTER_TABLE
    - The filter table contains information ... predicates are used processing. during query
  - DSN_PGRANGE_TABLE
    - The page range table, ...RANGE_TABLE, contains information about q... ...artitions for all page range scans in a query.
  - DSN_PGRO... ...BLE
    - The par... ...up table, DSN_PGROUP_TABLE, contains informatio... ...bout the parallel groups in a query.

No, But getting closer…

Populate the PLAN_TABLE?

- Tables used by EXPLAIN :
  - DSN_PREDICAT_TABLE
    - The predicate table, DSN_PREDIC[illegible]E, contains information about all of the predi[illegible] query.
  - DSN_PTASK_TABLE
    - The parallel tasks table[illegible]TASK_TABLE, contains information about al[illegible]arallel tasks in a query.
  - DSN_QUERY_T[illegible]
    - The query[illegible]tains information about a SQL statement, and dis[illegible] statement before and after query transform[illegible]n.
    - Not populated for static SQL statements at BIND or REBIND with EXPLAIN(YES)

No, But getting closer…

Populate the PLAN_TABLE?

- Tables used by EXPLAIN :
  - DSN_SORTKEY_TABLE
    - The sort key table contains informat… sort keys for all of the sorts required by a query
  - DSN_SORT_TABLE
    - The sort table contains … on about the sort operations required by a query
  - DSN_STRUCT…
    - The struct… contains information about all of the query blocks i… .
  - DSN_VIEWREF_TABLE
    - The view reference table contains information about all of the views and materialized query tables that are used to process a query.

No, But getting closer…

Populate the PLAN_TABLE?

- Tables used as input to EXPLAIN :
  - DSN_VIRTUAL_INDEXES
    - The virtual indexes table enables or… tools to test the effect of creating and dropping in… the performance of particular queries
      - Added in DB2 Version 8
  - DSN_USERQ…_TABLE
    - contains… mation about optimization hints
      - Added …B2 10

No, But getting closer…

Populate the PLAN_TABLE?

- Tables used by EXPLAIN as of DB2 10:
  - DSN_KEYGTDIST_TABLE
    - This table contains non-uniform index expression statistic that are obtained dynamically by the DB2 optimizer.
  - DSN_COLDIST_TABLE
    - contains non-uniform column group statistics that are obtained dynamically by DB2 during non-index leaf pages
  - DSN_QUERYINFO_TABLE
    - Used for the IBM DB2 Analytics Accelerator

Maybe now…

DB2 10 Format

- EBCDIC EXPLAIN tables deprecated in DB2 10
- EXPLAIN tables in format before DB2 Version 8 disallowed
- Migration job SDSNSAMP(DSNTIJPM) has queries to discover non-compliant tables

- DDL for EXPLAIN tables
  - SDSNSAMP(DSNTESC) contains DDL to build all EXPLAIN tables
  - Data Studio (no-charge download)
  - Optim Query Tuner (charge)
  - Optim Query Workload Tuner (charge)

DSN_STATEMENT_CACHE_TABLE

The next few slides discuss how to use the dynamic statement cache

Activate Dynamic Statement Cache

- Activate Dynamic Statement Cache and make sure it's of adequate size
  - To activate the Dynamic Statement Cache:
    - Install panel
      - CACHE DYNAMIC SQL field

    or

    - DSNZPARM keyword
      - DSN6SPRM macro, keyword CACHEDYN

  - Statement cache must of size to hold the entire workload
    - Install panel
      - EDM STATEMENT CACHE field
        - 5000 to 1048576 KB allowed, default is 113386
        - The dynamic statement cache space is above the 2GB bar

    or

    - DSNZPARM
      - DSN6SPRM EDMSTMTC (specified in KB)

Start Trace for Cache

- Non-Data-Sharing -START TRACE command example

```
-STA TRACE(MON) CLASS(30) IFCID(316,317,318)
  DEST(SMF)
```

  - DSNW130I - MON TRACE STARTED, ASSIGNED TRACE NUMBER 04
  - DSN9022I - DSNWVCM1 '-STA TRACE' NORMAL COMPLETION

- Data-Sharing –START TRACE command example

```
-STA TRACE(MON) CLASS(30) IFCID(316,317,318)
  DEST(SMF)  SCOPE(GROUP)
```

  - DSNW130I - MON TRACE STARTED, ASSIGNED TRACE NUMBER 04
  - DSN9022I - DSNWVCM1 '-STA TRACE' NORMAL COMPLETION

* The DEST is optional on the –START TRACE command in the above examples.

Note:
Do not use a Monitor Class 29 trace. It only starts IFCIDs 316 & 318 when IFCID 317 is also required.

Start Trace for Cache

- IFCID 316
  - First 60 bytes of SQL statement plus identifying information and statistics (statement Id, authid, usage counters)
    - http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2rrd20228.htm
- IFCID 317
  - Used in addition to IFCID 316 to obtain the full SQL statement text
    - http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.xe_db2.doc/ko2rrd20228.htm
- IFCID 318
  - Contains no data and acts only as a switch for IFCID 316 to collect all available information
    - Stopping and Starting IFCID 318 will reset stats and cause a new interval to start

Minimal impact on CPU and minimal impact (if any) on SMF.

Capture Dynamic Statement Cache

- Execute the EXPLAIN STMTCACHE ALL
  - Issuing the above SQL statements results in one row being returned for each statement cached in the dynamic statement cache to the DSN_STATEMENT_CACHE_TABLE table. Each row returned contains identifying information about the cached statement along with statistics reflecting that statements execution. Statistics are returned ONLY when the IFCID 316 and 318 traces are active.
  - This SQL statement can be executed using SPUFI, DSNTEP2, DSNTEP4, or any number of programs that allow dynamically prepared SQL statements.
  - Ensure SCHEMA (SQLID if pre DB2 9) is set correct for the EXPLAIN tables you are using for processing.
  - Make sure you have the proper authorization to dump all the SQL in the dynamic statement cache.
    - For example, running the EXPLAIN STMTCACHE ALL with SYSADM authority will avoid this issue.

Stop Trace

- Stop the trace ONLY after verifying DSN_STATEMENT_CACHE_TABLE table appears to be accurate
- Stop trace

```
-STOP TRACE(MON) CLASS(30)
DSNW131I  - STOP TRACE SUCCESSFUL FOR
  TRACE NUMBER(S) 04
DSN9022I  - DSNWVCM1 '-STO TRACE' NORMAL
  COMPLETION
```

- Verify trace has been successfully stopped

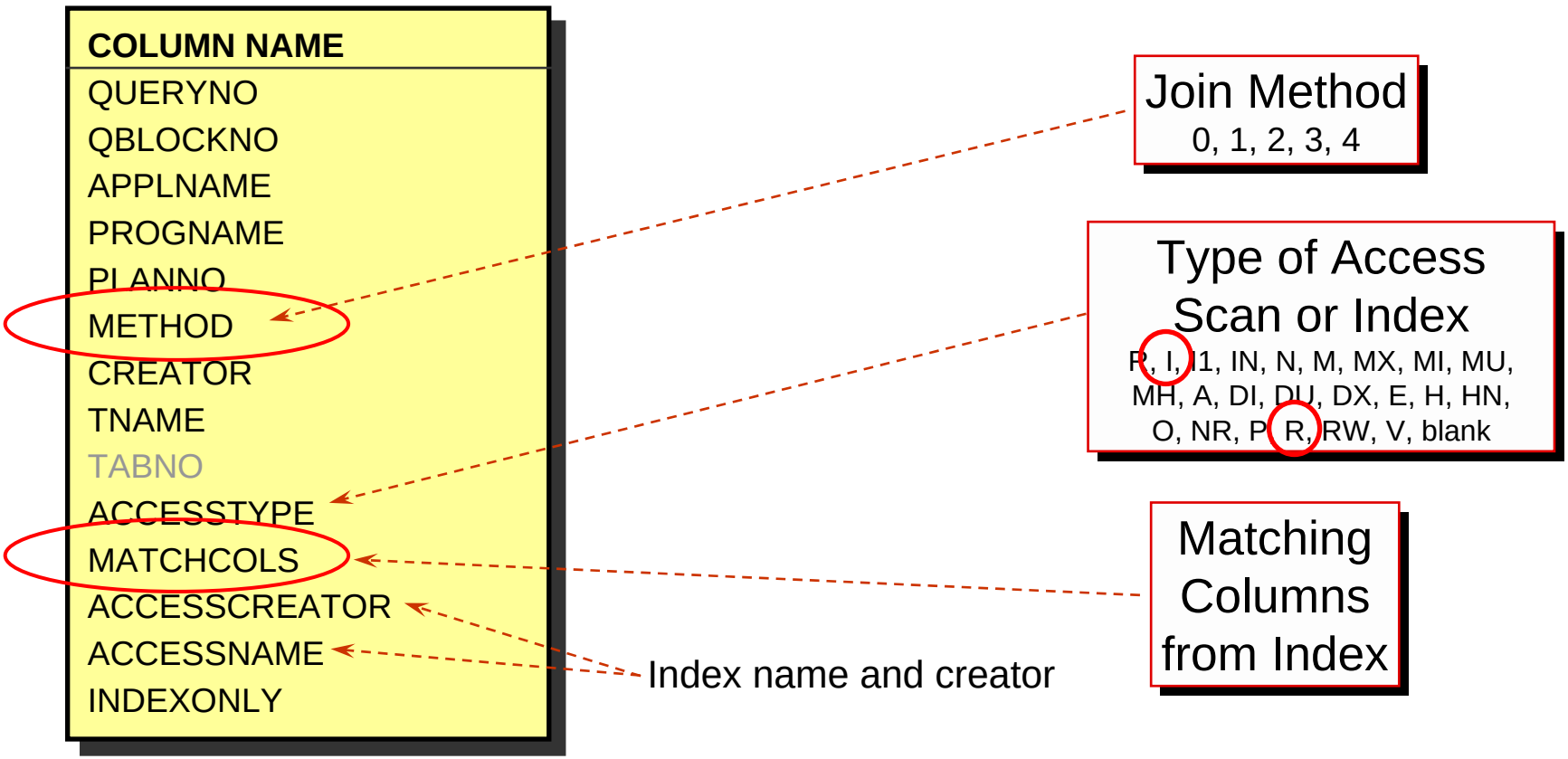
```
-DISPLAY TRACE(*)
```

PLAN_TABLE, etc…

- EXPLAIN Output tables:
  - You are responsible for creating all EXPLAIN tables
  - Ownership
    - For Dynamic - SQLID
    - For PLAN - Plan owner
    - For Package - Package owner
    - Can use ALIAS as of Version 8
  - Data in tables MUST be interpreted
    - Must Understand Column Meaning and Codes
  - EXPLAIN Won't Tell You Everything
    - Run Time Access Enhancements
    - Referential Integrity
    - Access to LOB values

The PLAN_TABLE

25 Column Format (V2.1)



Is a table space scan bad?
And, is an index access good? It's not all magic!!!!

Methods – Joins and Sorts

- METHOD
  - 0 - First table accessed
  - 1 - Nested Loop Join
  - 2 - Merge Scan Join
  - 3 - Sort
  - 4 - Hybrid Join

MATCHCOLS

IX(C1, C2, C3, C4)

C1=1
C2=2
C3>3
C4=4

ACCESSTYPE='I' and MATCHCOLS=3

C1=1
C3>2
C4=3

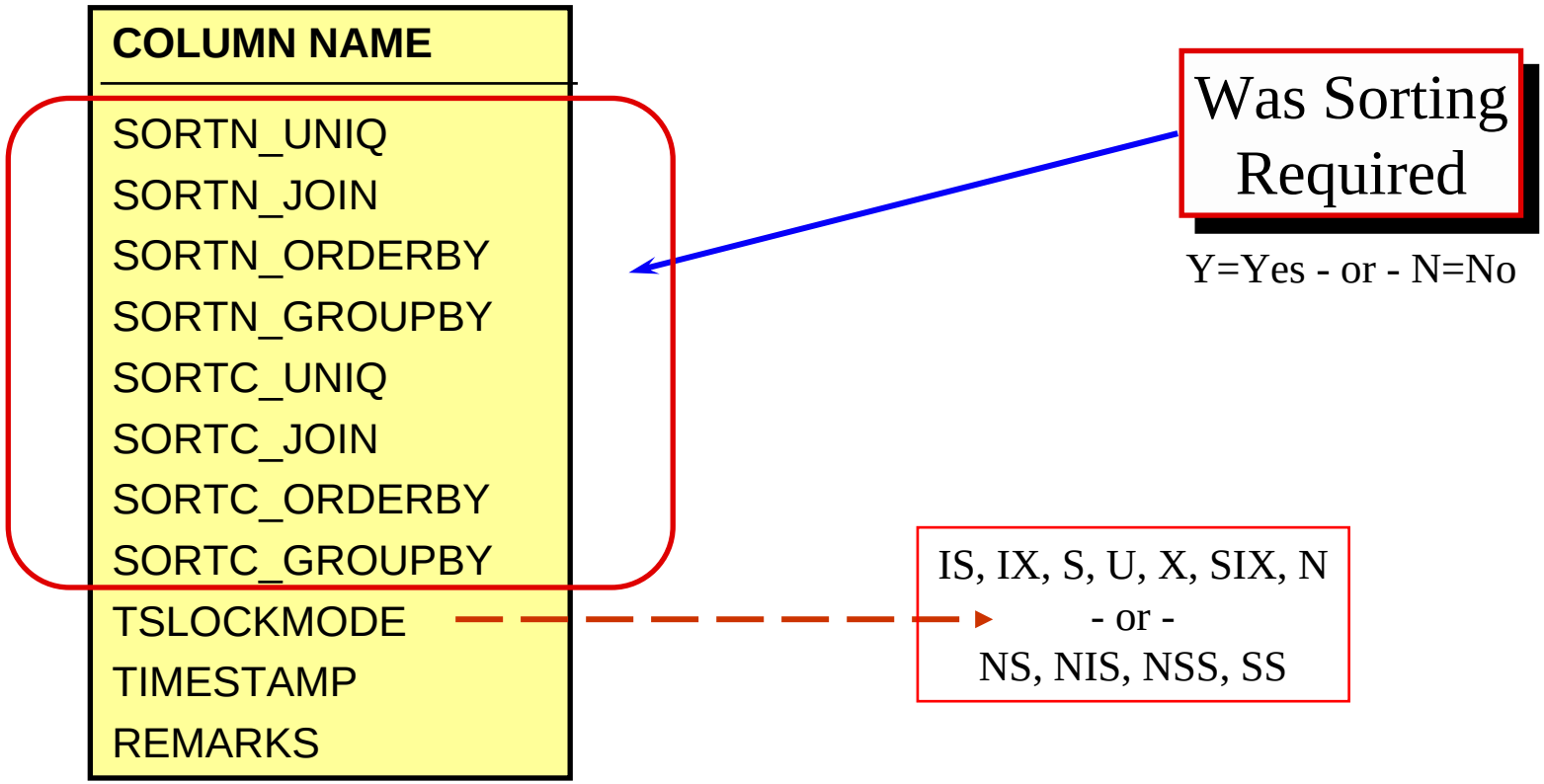
ACCESSTYPE='I' and MATCHCOLS=1

C5=1

ACCESSTYPE='I' and MATCHCOLS=0

The PLAN_TABLE

25 Column Format (V2.1)



The PLAN_TABLE

28 Column Format (V2.2)

COLUMN NAME
PREFETCH
COLUMN_FN_EVAL
MIXOPSEQ

PREFETCH → S, L, D, U, or Blank

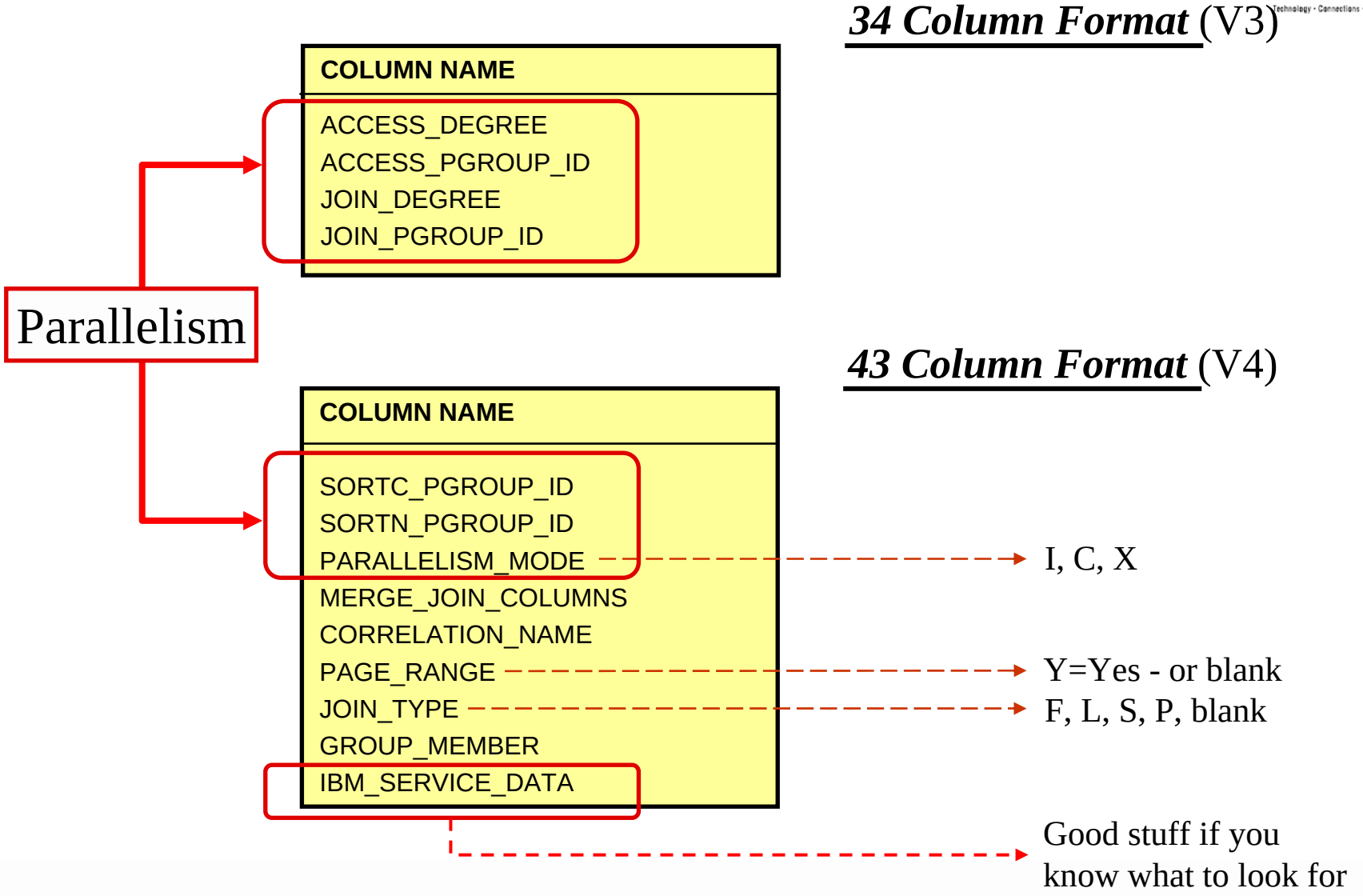
COLUMN_FN_EVAL → R, S, or Blank

30 Column Format (V2.3)

COLUMN NAME
VERSION
COLLID

For **Packages**

The PLAN_TABLE



Joins

- JOIN_TYPE
  - F - Full Outer Join
  - L - Left Outer Join
  - S - Star Join
  - blank - Inner Join or No Join

The PLAN_TABLE

CREATE DDL in Member DSNTESC in DSNSAMP

46 Column Format (V5)

COLUMN NAME
WHEN_OPTIMIZE
QBLOCK_TYPE
BIND_TIME

- WHEN_OPTIMIZE → Blank, B, R
- QBLOCK_TYPE → See List

49 Column Format (V6)

COLUMN NAME
OPTHINT
HINT_USED
PRIMARY_ACCESSTYPE

- OPTHINT → Hint identifier
- HINT_USED → Hint identifier
- PRIMARY_ACCESSTYPE → Blank, D, T

51 Column Format (V7)

COLUMN NAME
PARENT_QBLOCKNO
TABLE_TYPE

- TABLE_TYPE → F, Q, T, W, M, B, C, I, R, S

See List:

SELECT
INSERT
UPDATE
MERGE
DELETE
SELUPD
DELCUR
UPDCUR
CORSUB
TRUNCA
NCOSUB
TABLEX
TRIGGER
UNION
UNIONA
INTERS
INTERA
EXCEPT
EXCEPTA
PRUNED

The PLAN_TABLE

58 Column Format (V8)

COLUMN NAME
▪ TABLE_ENCODE
▪ TABLE_SCCSID
▪ TABLE_MCCSID
▪ TABLE_DCCSID
▪ ROUTINE_ID

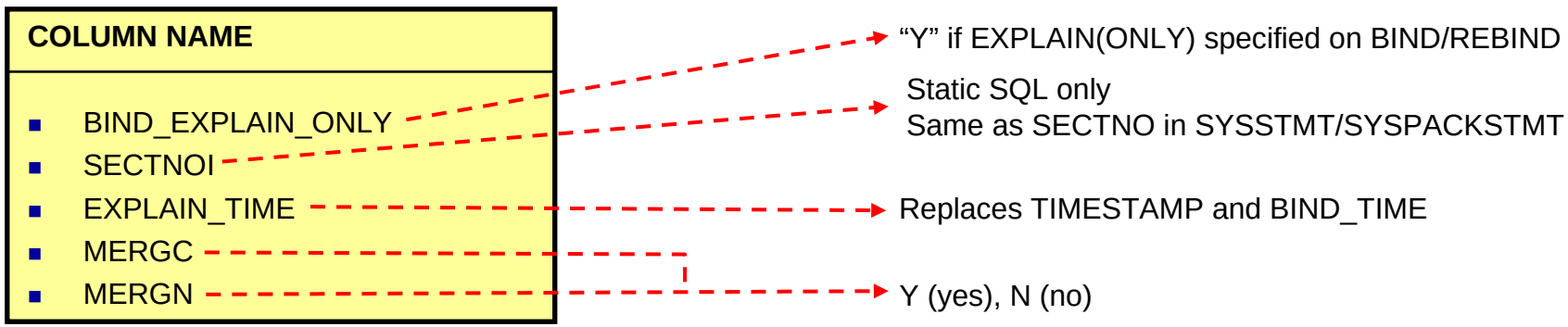
E – EBCDIC
A – ASCII
U – Unicode
M – multi CCSID sets

59 Column Format (DB2 9)

COLUMN NAME
▪ CTEREF
▪ STMTTOKEN
▪ PARENT_PLANNO

The PLAN_TABLE

64 Column Format (DB2 10)



Plan_Table Access

- Version 8 allows the use of an ALIAS to access PLAN_TABLE

```
CREATE ALIAS WILLIE.PLAN_TABLE  FOR

PRODUCTION.PLAN_TABLE
```

Willie runs query

```
EXPLAIN PLAN FOR SELECT….
```

Ordering PLAN_TABLE Data

- PLAN_TABLE should be sorted by:
  - EXPLAIN_TIME
    - When EXPLAIN information was captured
  - QUERYNO
    - QUERYNO clause on EXPLAIN
    - QUERYNO clause on a DML statement
    - Line number from source
  - QBLOCKNO
    - Query position in statement
      - 1 is outer most query block
  - PLANNO
    - Execution order of steps
  - MIXOPSEQ
    - Step order for multiple index operations

TIMESTAMP *&* BIND_TIME *have been deprecated. Use EXPLAIN_TIME.*

Which Rows Do You Want?

- Search for Plans:
  - APPLNAME = the application plan name
- Search for Packages:
  - PROGNAME = the package name
  - COLLID = the collection id
  - VERSION = the version identifier

Only applies to rows produced from some kind of BIND

Querying from Plan/Package

- For Plans:

```
SELECT *
    FROM YOUR_PLAN_TABLE
        WHERE APPLNAME = 'application_plan_name'
ORDER BY
        EXPLAIN_TIME, QUERYNO, QBLOCKNO, PLANNO,
        MIXOPSEQ;
```

- For Packages:

```
SELECT *
    FROM YOUR_PLAN_TABLE
        WHERE PROGNAME = 'package_name'
          AND COLLID = 'collection_id'
          AND VERSION = 'version_identifier'
ORDER BY
        EXPLAIN_TIME, QUERYNO, QBLOCKNO,
        PLANNO,MIXOPSEQ;
```

And Just for the Fun of It…

```
SELECT
 A.EXPLAIN_TIME, ',',  A.QUERYNO,',',
A.QBLOCKNO,',',
 A.PLANNO,',',
 SUBSTR(A.PROGNAME,1,22) AS PROGNAME,',
 SUBSTR(A.COLLID,1,22) AS COLLID,',',
 SUBSTR(A.VERSION,1,14) AS VERSION,',',
 SUBSTR(A.CREATOR,1,14) AS CREATOR,',',
 CASE A.METHOD
           WHEN 0 THEN 'FIRST'
           WHEN 1 THEN 'NLJOIN'
           WHEN 2 THEN 'MSJOIN'
           WHEN 3 THEN 'SORT'
           WHEN 4 THEN 'HYBRID'
                  ELSE 'UNKNWN'
      END AS METHOD,',',
 SUBSTR(TNAME,1,14) AS TABLENAME,',',
 BIGINT(C.CARDF) TCARDF, ',',
 BIGINT(C.NPAGESF) TNPAGESF, ',',
 C.PCTPAGES TPCTPAGES, ',', TABNO,',',
 B.PROCMS, ',', B.PROCSU, ',',
 B.COST_CATEGORY, ',',
 SUBSTR(B.REASON,1,25) AS REASON,',',
 ACCESSTYPE AS ACCTYP,',',
 MATCHCOLS AS MCOLS,',',
 SUBSTR(ACCESSCREATOR,1,14) AS ACCSSCRTR,',',
 SUBSTR(ACCESSNAME,1,14) AS ACCSSNM,',',
 CASE A.INDEXONLY
           WHEN 'Y' THEN 'INDEXONLY'
                    ELSE ' '
      END,',',
```

```
SORTN_UNIQ AS SN_U,',',
SORTN_JOIN AS SN_J,',',
SORTN_ORDERBY AS SN_O,',',
SORTN_GROUPBY AS SN_G,',',
SORTC_UNIQ AS SC_U,',',
SORTC_JOIN AS SC_J,',',
SORTC_ORDERBY AS SC_O,',',
SORTC_GROUPBY AS SC_G,',',
TSLOCKMODE,',‘,
CASE A.PREFETCH
           WHEN 'S' THEN 'SEQ'
           WHEN 'L' THEN 'LST'
           WHEN 'D' THEN 'DYN'
           WHEN 'U' THEN 'LPRID'
                    ELSE ' '
END AS PRFTCH,',', COLUMN_FN_EVAL AS COL_FN_E,
',',  MIXOPSEQ,',',
SUBSTR(A.VERSION,1,14) AS  VERSION,',',
SUBSTR(A.COLLID,1,14) AS COLLID,',',
ACCESS_DEGREE AS ACC_D,',',
ACCESS_PGROUP_ID AS ACC_PGID,',',
JOIN_DEGREE AS JN_D,',',
JOIN_PGROUP_ID AS JN_PGID,',',
SORTC_PGROUP_ID AS SC_PGID,',',
SORTN_PGROUP_ID AS SN_PGID,',',
PARALLELISM_MODE AS P_ISM_MD,',',
MERGE_JOIN_COLS AS M_J_COL,',',
SUBSTR(CORRELATION_NAME,1,10) AS  CORL_NM,',',
PAGE_RANGE,',',
```

```
SUBSTR(A.GROUP_MEMBER,1,8) AS GRP_MBR,',',
WHEN_OPTIMIZE,',',
QBLOCK_TYPE,',',
CASE A.JOIN_TYPE
           WHEN 'F' THEN 'FULLOJ'
           WHEN 'L' THEN 'LEFTOJ'
           WHEN 'P' THEN 'PAIRWS'
           WHEN 'S' THEN 'STAR'
           WHEN ' ' THEN 'INNER'
                  ELSE 'UNKNWN'
      END AS JNTYPE,',',
PRIMARY_ACCESSTYPE AS PRI_ACCES_TYPE,',',
PARENT_QBLOCKNO,',',
PARENT_PLANNO,',',
TABLE_TYPE,',',
TABLE_ENCODE,',',
CTEREF,',',
SUBSTR (STMTTOKEN,1,14) AS STMTTOKEN,',',
A.EXPLAIN_TIME,',',
MERGC,',',MERGN,',',
```


```
(CASE SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,17,1)),1,1)
      WHEN 'A' THEN 10
      WHEN 'B' THEN 11
      WHEN 'C' THEN 12
      WHEN 'D' THEN 13
      WHEN 'E' THEN 14
      WHEN 'F' THEN 15
ELSE
INTEGER(SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,17,1)),1,1))
END * POWER (16, 3) + CASE
SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,17,1)),2,1)
      WHEN 'A' THEN 10
      WHEN 'B' THEN 11
      WHEN 'C' THEN 12
      WHEN 'D' THEN 13
      WHEN 'E' THEN 14
      WHEN 'F' THEN 15
ELSE
INTEGER(SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,17,1)),2,1))
END * POWER (16, 2) + CASE
SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,18,1)),1,1)
      WHEN 'A' THEN 10
      WHEN 'B' THEN 11
      WHEN 'C' THEN 12
      WHEN 'D' THEN 13
      WHEN 'E' THEN 14
      WHEN 'F' THEN 15
ELSE
INTEGER(SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,18,1)),1,1))
END * POWER (16, 1) + CASE
SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,18,1)),2,1)
      WHEN 'A' THEN 10
      WHEN 'B' THEN 11
      WHEN 'C' THEN 12
      WHEN 'D' THEN 13
      WHEN 'E' THEN 14
      WHEN 'F' THEN 15
   ELSE
INTEGER(SUBSTR(HEX(SUBSTR(IBM_SERVICE_DATA,18,1)),2,1))
END * POWER (16, 0)) AS NUMCP_EXP,',',
    HEX(SUBSTR(IBM_SERVICE_DATA,23,1)) AS REASON,',',
    HEX(SUBSTR(IBM_SERVICE_DATA,25,2)) AS CPU,',',
    HEX(SUBSTR(IBM_SERVICE_DATA,69,4)) AS CPUSPEED,',',
    HEX(SUBSTR(IBM_SERVICE_DATA,13,4)) AS RIDPOOL,',',
    HEX(SUBSTR(IBM_SERVICE_DATA,9,4)) AS
    SORT_POOL_SIZE_HEX,
    SUBSTR(IBM_SERVICE_DATA,67,7) APAR,',',
    IBM_SERVICE_DATA


FROM    WILLIE.PLAN_TABLE A,
      DSN_STATEMNT_TABLE B,
      SYSIBM.SYSTABLES C


   WHERE C.DBNAME = 'BID1TB'
      AND C.NAME = A.TNAME
      AND C.CREATOR = A.CREATOR
      AND C.TYPE = 'T'
      AND A.QUERYNO = 91111



ORDER BY A.EXPLAIN_TIME, A.QUERYNO,A.QBLOCKNO,
    A.PLANNO, A.MIXOPSEQ ;
```

Using OPTHINT

- Create and index on PLAN_TABLE
  - Key = QUERYNO, APPLNAME, PROGNAME, VERSION, COLLID, OPTHINT
- On installation panel DSNTIP4
  - Set OPTIMIZATION HINTS to YES
- Update Plan_Table
- Set OPTHINT column to “some_value”
- Tell DB2 to use the hint
  - EXPLAIN(YES) and OPTHINT(’ some_value’)
    - or
  - SET CURRENT OPTIMIZATION HINT =' some_value ';
- Check Plan_Table and see if hint was used

References

Please Visit My DB2 for z/OS Blog

http://it.toolbox.com/blogs/db2zos/

Thank you!

SHARE
Technology · Connections · Results

Willie Favero

DB2 SME
Data Warehousing for System z Swat Team
IBM Silicon Valley Laboratory

My DB2 Blog
www.it.toolbox.com/blogs/db2zos/

http:www.WillieFavero.com