IDUG

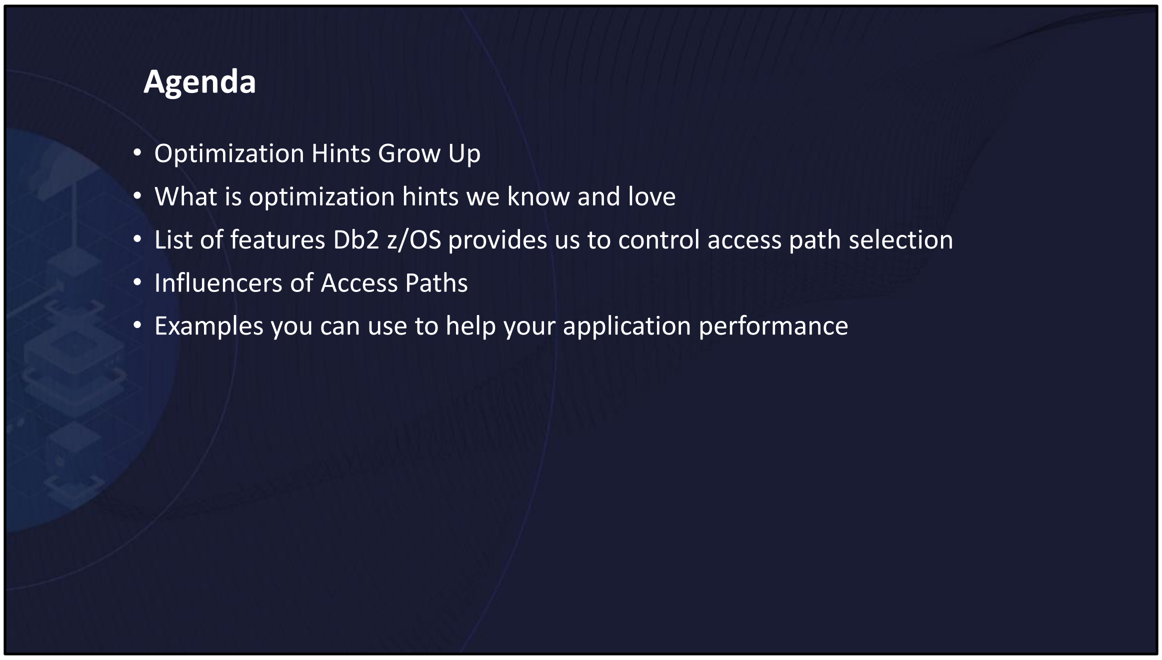
IDUG VIRTUAL
2021 NA **Db2** Tech Conference

Optimization Hints Grow Up

Chad Reiber, BMC Software

Db2 z/OS

Welcome to Optimization Hints Grow Up … Chad Reiber BMC Software



Agenda

- Optimization Hints Grow Up
- What is optimization hints we know and love
- List of features Db2 z/OS provides us to control access path selection
- Influencers of Access Paths
- Examples you can use to help your application performance

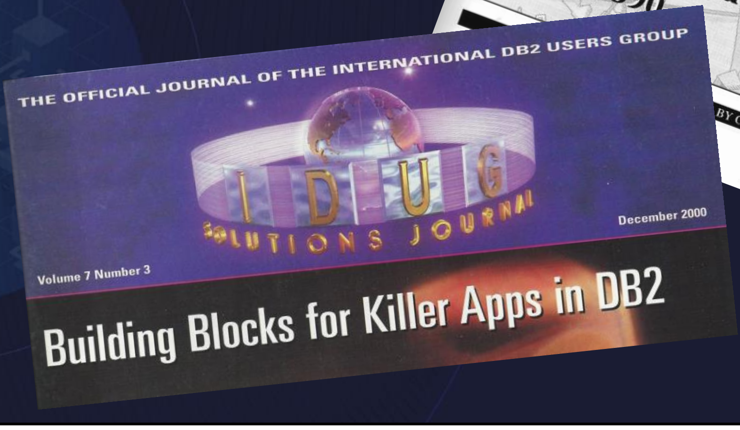
Optimization Hints Grow Up
What is optimization hints we know and love
List of features Db2 z/OS provides us to control access paths selection
Influencers of Access Paths
Examples you can use to help your application performance

IBM Db2 has given us several ways to influence access paths. Whether it is to fallback to older access paths that worked. Tell Db2 how you want the query to access the data or even use AI to help with access path selection. This presentation will show us

IBM Db2 has given us several ways to influence access paths release after release. This presentation will look at what those feature/functions were in the past, how they have changed, and what new ways we have to determine the appropriate access path.

Seedy Underworld of Db2

- Article I wrote in 2000
  - That's right Y2K …

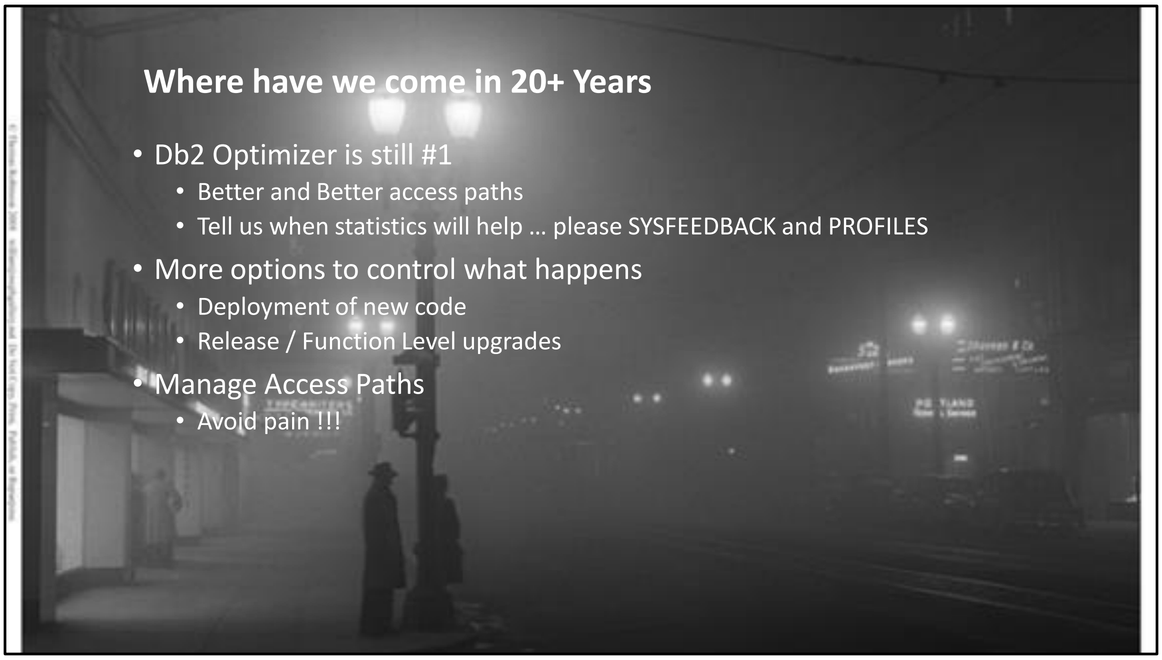

THE OFFICIAL JOURNAL OF THE INTERNATIONAL DB2 USERS GROUP
IDUG
SOLUTIONS JOURNAL
December 2000
Volume 7 Number 3
Building Blocks for Killer Apps in DB2

The Seedy Underworld of DB2 UDB for OS/390

BY CHAD REIBER

change the path without their knowledge, causing a painful morning. What about the possibility of imple-

Back in 2000, I wrote an article for the IDUG Journal titled the Seedy Underworld of Db2. It discussed the new feature in DB2 Version 6 where our world class Db2 Optimizer could be influenced by the deal making backroom of the optimization hints. It was controversial the time but over the different DB2 releases and twenty years, IBM has given us several ways to influence the access paths, fallback to older access paths and just recently to use AI to help with access path determination. In this presentation, I will highlight where we have been and where we are now with getting the access path we want and need.
Many times, DBAs and Db2 Performance people are asked to improve performance without the benefit of the applications being able to make SQL changes. This is just another bunch of features/functions, that come with Db2 z/OS, individuals can used to make a difference to the end user.



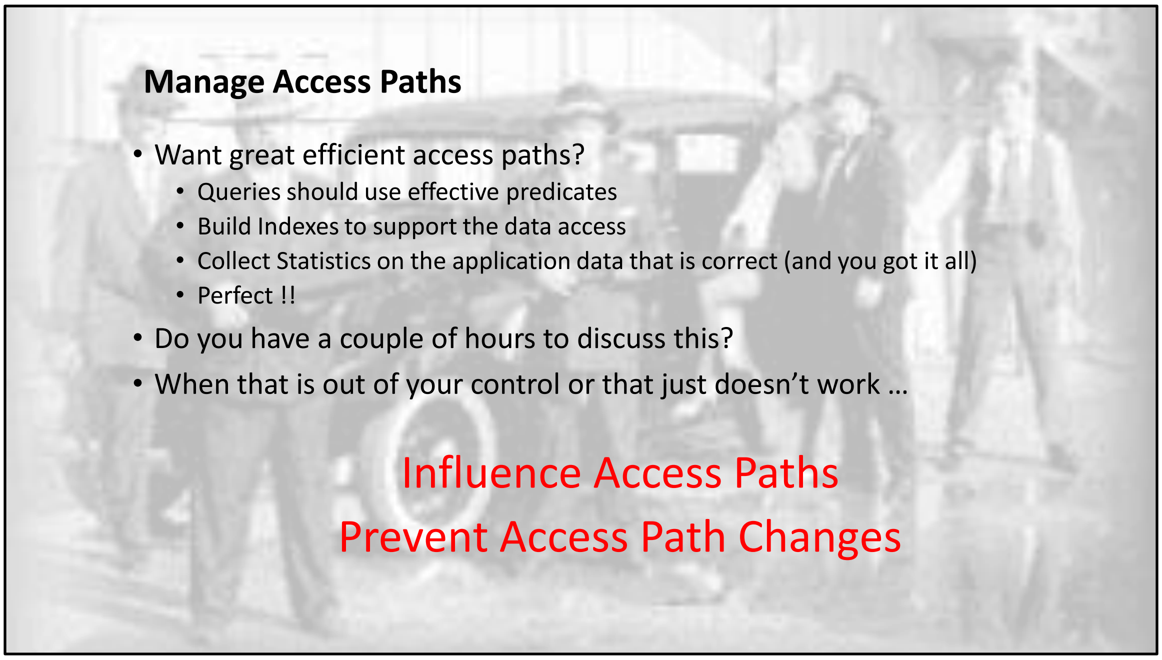
Where have we come in 20+ Years

- Db2 Optimizer is still #1
  - Better and Better access paths
  - Tell us when statistics will help … please SYSFEEDBACK and PROFILES
- More options to control what happens
  - Deployment of new code
  - Release / Function Level upgrades
- Manage Access Paths
  - Avoid pain !!!

Are we still in the shadows with influencing the optimizer.

IBM and Db2 continues to innovate when it comes to the DBMS, this includes helping to get the best result for our end users.

While the Optimize is still number 1, there are times where we might need to control what happens with the access paths. Avoid the Monday Morning headaches, or worse yet middle of the day headaches

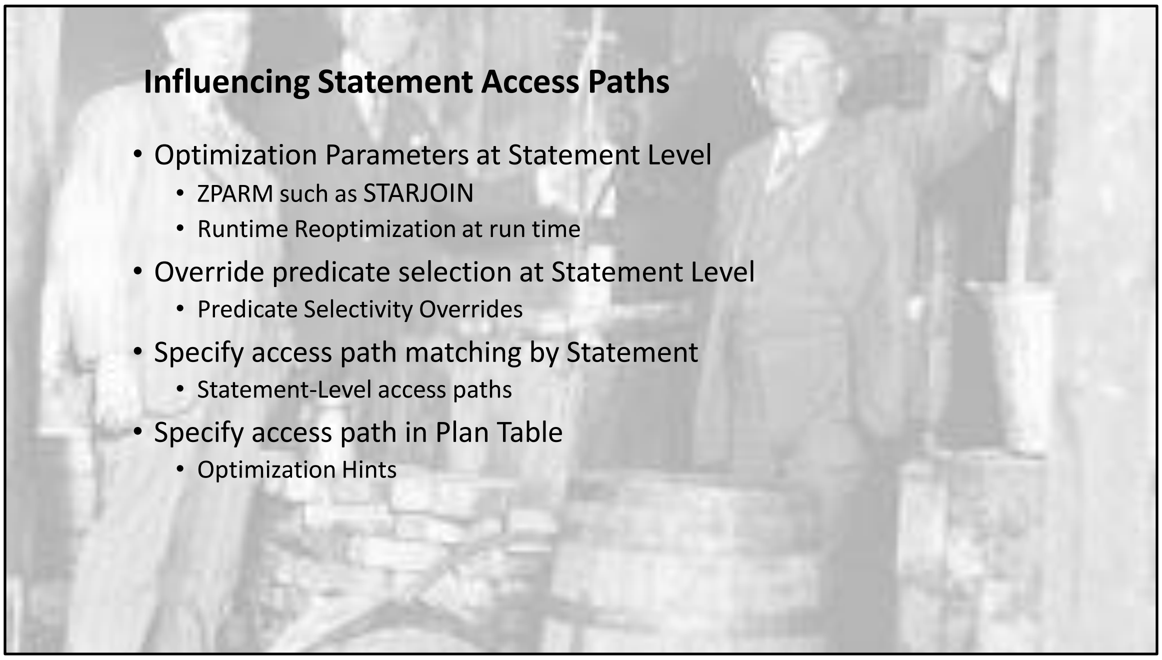


Manage Access Paths

- Want great efficient access paths?
  - Queries should use effective predicates
  - Build Indexes to support the data access
  - Collect Statistics on the application data that is correct (and you got it all)
  - Perfect !!
- Do you have a couple of hours to discuss this?
- When that is out of your control or that just doesn't work …

Influence Access Paths

Prevent Access Path Changes

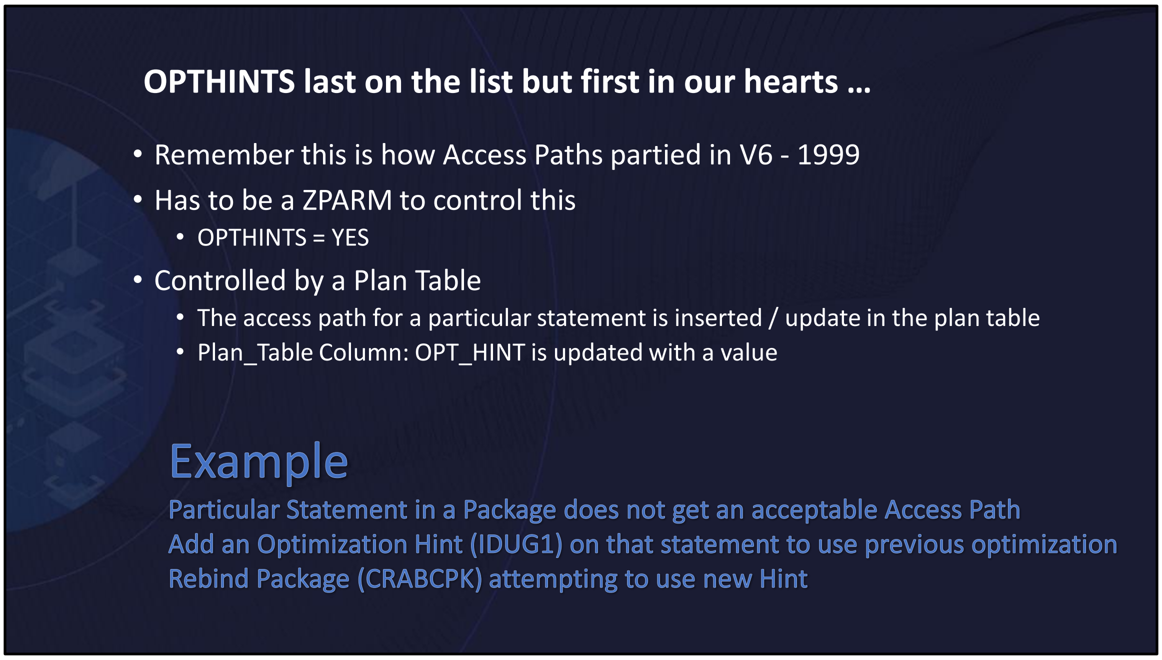


Influencing Statement Access Paths

- Optimization Parameters at Statement Level
  - ZPARM such as STARJOIN
  - Runtime Reoptimization at run time
- Override predicate selection at Statement Level
  - Predicate Selectivity Overrides
- Specify access path matching by Statement
  - Statement-Level access paths
- Specify access path in Plan Table
  - Optimization Hints

There are four specific ways we can help influence statement access paths with in Db2.

There are number of SQL coding techniques that can help as well. I will not be going over those with this presentation. But things like catalog statistic changes can help the optimizer make a decision you really need.



OPTHINTS last on the list but first in our hearts …

- Remember this is how Access Paths partied in V6 - 1999
- Has to be a ZPARM to control this
  - OPTHINTS = YES
- Controlled by a Plan Table
  - The access path for a particular statement is inserted / update in the plan table
  - Plan_Table Column: OPT_HINT is updated with a value

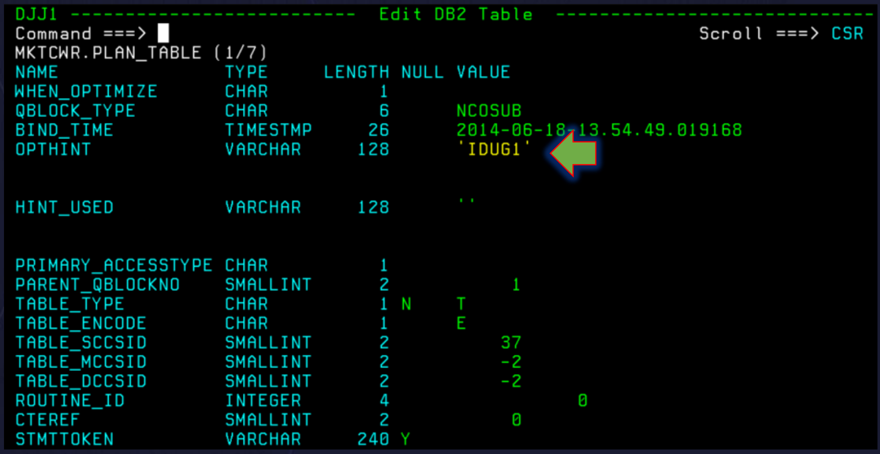
Example

Particular Statement in a Package does not get an acceptable Access Path
Add an Optimization Hint (IDUG1) on that statement to use previous optimization
Rebind Package (CRABCPK) attempting to use new Hint

Lets start at the beginning. The article I wrote in 2000 on optimization hints. Which really came out with Db2 V6 in 1999.

An example is helpful here.

Add OPTHINT (IDUG1) to the Package Qualifier's Plan Table



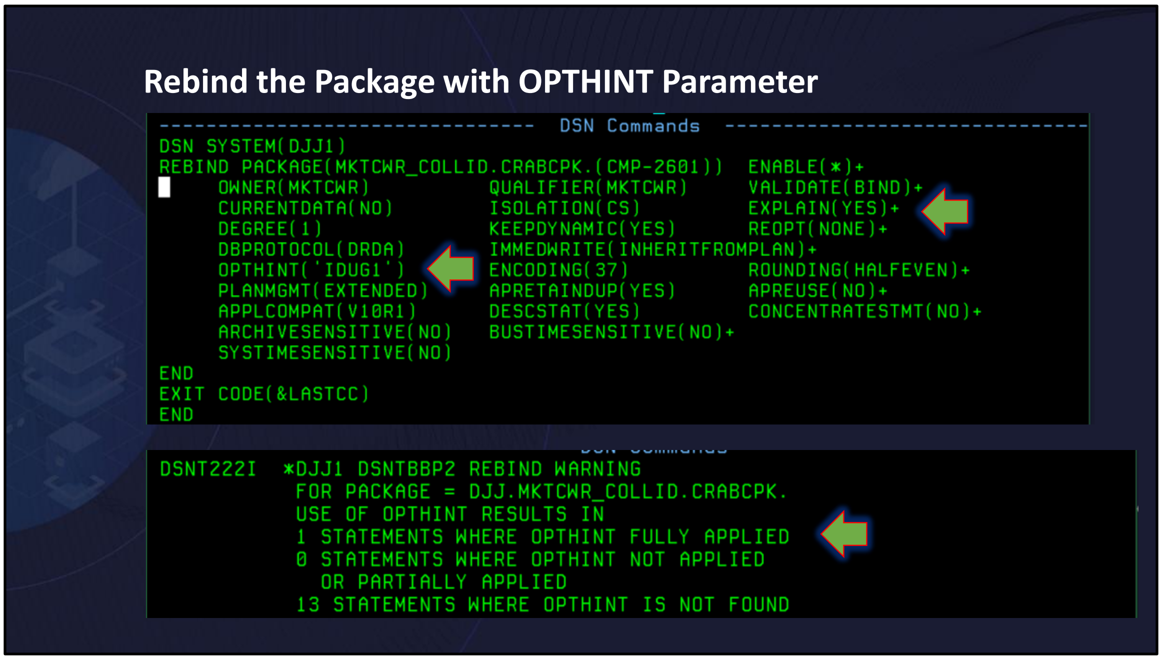
```
DJJ1 -------------------------  Edit DB2 Table  ----------------------------
Command ===> █                                                 Scroll ===> CSR
MKTCWR.PLAN_TABLE (1/7)
NAME                TYPE      LENGTH NULL  VALUE
WHEN_OPTIMIZE       CHAR           1
QBLOCK_TYPE         CHAR           6       NCOSUB
BIND_TIME           TIMESTMP      26       2014-06-18-13.54.49.019168
OPTHINT             VARCHAR      128       'IDUG1'


HINT_USED           VARCHAR      128       ''


PRIMARY_ACCESSTYPE  CHAR           1
PARENT_QBLOCKNO     SMALLINT       2              1
TABLE_TYPE          CHAR           1 N     T
TABLE_ENCODE        CHAR           1       E
TABLE_SCCSID        SMALLINT       2           37
TABLE_MCCSID        SMALLINT       2           -2
TABLE_DCCSID        SMALLINT       2           -2
ROUTINE_ID          INTEGER        4                    0
CTEREF              SMALLINT       2              0
STMTTOKEN           VARCHAR      240 Y
```

Here is the view of the plan table that I am going to update with my optimization hint.

OPTHINT is updated to include IDUG1

What you don't see here is the 'key' of the plan table, package name, query number etc. That is important as the rebind will look to match that to implement the hint.

We will see that in the next couple of slides



Rebind the Package with OPTHINT Parameter

```
-------------------------------- DSN Commands --------------------------------
DSN SYSTEM(DJJ1)
REBIND PACKAGE(MKTCWR_COLLID.CRABCPK.(CMP-2601))  ENABLE(*)+
     OWNER(MKTCWR)           QUALIFIER(MKTCWR)     VALIDATE(BIND)+
     CURRENTDATA(NO)         ISOLATION(CS)         EXPLAIN(YES)+
     DEGREE(1)               KEEPDYNAMIC(YES)      REOPT(NONE)+
     DBPROTOCOL(DRDA)        IMMEDWRITE(INHERITFROMPLAN)+
     OPTHINT('IDUG1')        ENCODING(37)          ROUNDING(HALFEVEN)+
     PLANMGMT(EXTENDED)      APRETAINDUP(YES)      APREUSE(NO)+
     APPLCOMPAT(V10R1)       DESCSTAT(YES)         CONCENTRATESTMT(NO)+
     ARCHIVESENSITIVE(NO)    BUSTIMESENSITIVE(NO)+
     SYSTIMESENSITIVE(NO)
END
EXIT CODE(&LASTCC)
END
```

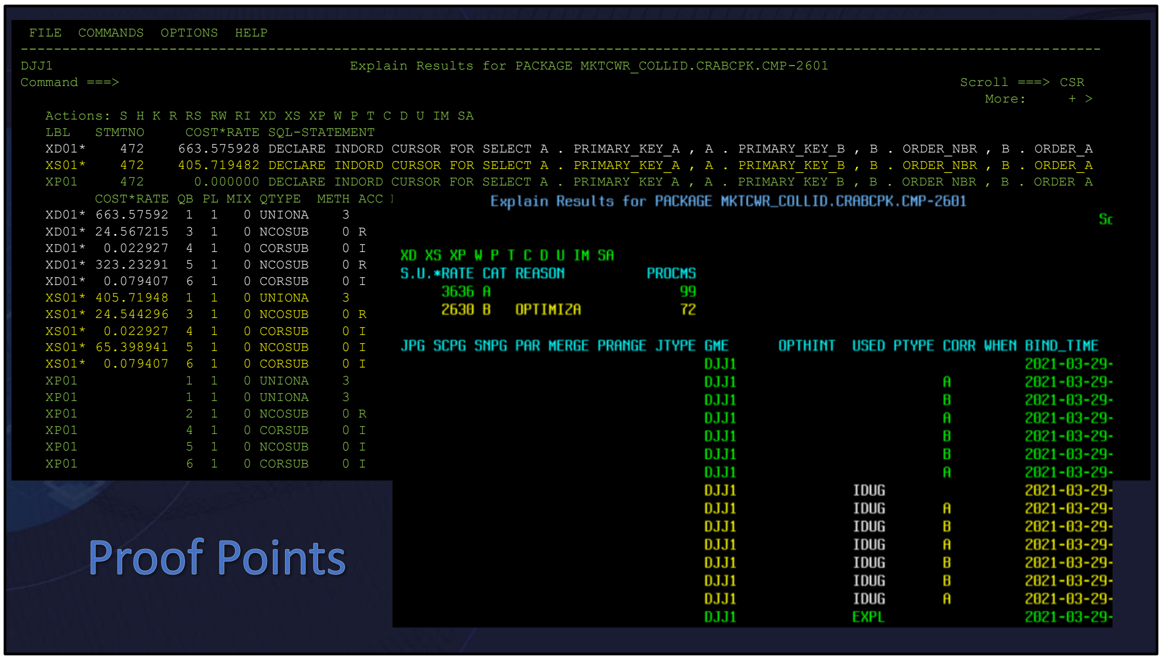
```
DSNT222I  *DJJ1 DSNTBBP2 REBIND WARNING
           FOR PACKAGE = DJJ.MKTCWR_COLLID.CRABCPK.
           USE OF OPTHINT RESULTS IN
           1 STATEMENTS WHERE OPTHINT FULLY APPLIED
           0 STATEMENTS WHERE OPTHINT NOT APPLIED
             OR PARTIALLY APPLIED
           13 STATEMENTS WHERE OPTHINT IS NOT FOUND
```

Want to implement that optimization hint?

In a static environment you need to rebind with the keyword of OPTHINT, and provide the opthint name we put into the plan table. In this case IDUG1.

You see the messages from the rebinds – back in 1999, the messages were not so nice.

Couple other points here, I am saying EXPLAIN(YES) because I want to validate that my opthint is actually used



Proof Points

```
 FILE  COMMANDS  OPTIONS  HELP
------------------------------------------------------------------------------------------------------------------------------------
DJJ1                                    Explain Results for PACKAGE MKTCWR_COLLID.CRABCPK.CMP-2601
Command ===>                                                                                                         Scroll ===> CSR
                                                                                                                        More:     + >
   Actions: S H K R RS RW RI XD XS XP W P T C D U IM SA
   LBL   STMTNO     COST*RATE SQL-STATEMENT
   XD01*    472    663.575928 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , B . ORDER_NBR , B . ORDER_A
   XS01*    472    405.719482 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , B . ORDER_NBR , B . ORDER_A
   XP01     472      0.000000 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , B . ORDER_NBR , B . ORDER_A
         COST*RATE QB PL MIX QTYPE   METH ACC
   XD01* 663.57592  1  1   0 UNIONA     3
   XD01* 24.567215  3  1   0 NCOSUB     0 R
   XD01*  0.022927  4  1   0 CORSUB     0 I
   XD01* 323.23291  5  1   0 NCOSUB     0 R
   XD01*  0.079407  6  1   0 CORSUB     0 I
   XS01* 405.71948  1  1   0 UNIONA     3
   XS01* 24.544296  3  1   0 NCOSUB     0 R
   XS01*  0.022927  4  1   0 CORSUB     0 I
   XS01* 65.398941  5  1   0 NCOSUB     0 I
   XS01*  0.079407  6  1   0 CORSUB     0 I
   XP01             1  1   0 UNIONA     3
   XP01             1  1   0 UNIONA     3
   XP01             2  1   0 NCOSUB     0 R
   XP01             4  1   0 CORSUB     0 I
   XP01             5  1   0 NCOSUB     0 I
   XP01             6  1   0 CORSUB     0 I
```

```
           Explain Results for PACKAGE MKTCWR_COLLID.CRABCPK.CMP-2601
                                                                          Sc

XD XS XP W P T C D U IM SA
S.U.*RATE CAT REASON           PROCMS
     3636 A                        99
     2630 B    OPTIMIZA            72

JPG SCPG SNPG PAR MERGE PRANGE JTYPE GME      OPTHINT  USED PTYPE CORR WHEN BIND_TIME
                                     DJJ1                                   2021-03-29-
                                     DJJ1                           A       2021-03-29-
                                     DJJ1                           B       2021-03-29-
                                     DJJ1                           A       2021-03-29-
                                     DJJ1                           B       2021-03-29-
                                     DJJ1                           B       2021-03-29-
                                     DJJ1                           A       2021-03-29-
                                     DJJ1              IDUG                 2021-03-29-
                                     DJJ1              IDUG         A       2021-03-29-
                                     DJJ1              IDUG         B       2021-03-29-
                                     DJJ1              IDUG         A       2021-03-29-
                                     DJJ1              IDUG         B       2021-03-29-
                                     DJJ1              IDUG         B       2021-03-29-
                                     DJJ1              IDUG         A       2021-03-29-
                                     DJJ1              EXPL                 2021-03-29-
```

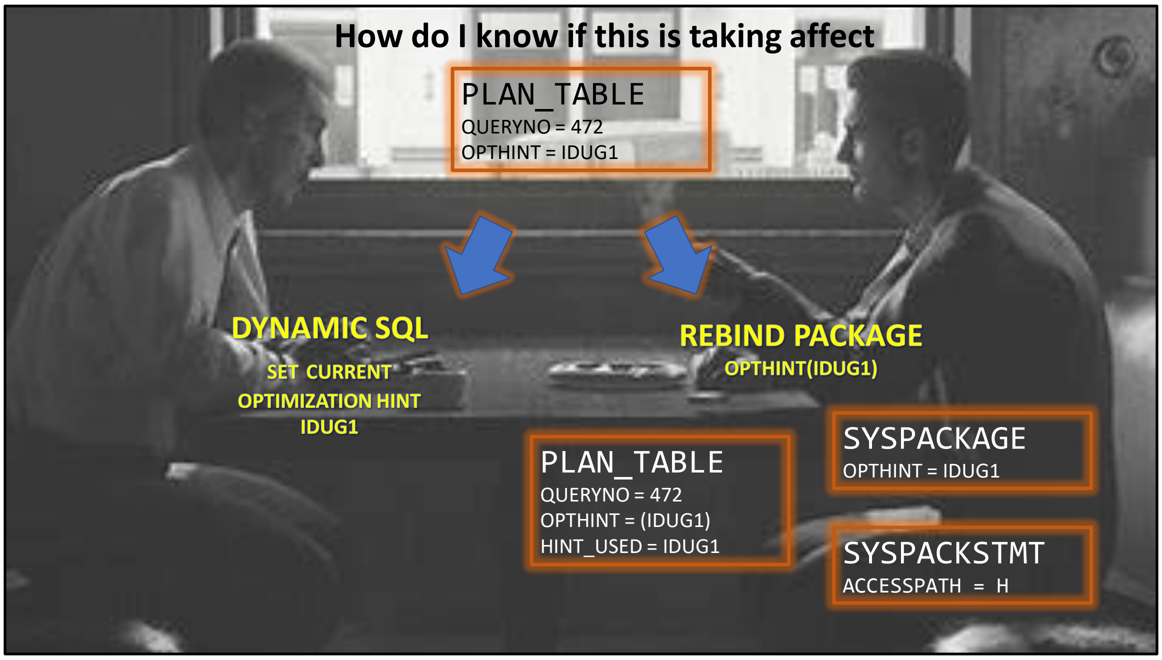
Lets see this in action.

This is just a way to look at “explains”. You will have your way to look at them. Using a BMC way.

XD entry here is the statement that I just passed to the optimizer to see what access path it would choose for my statement. Focus on the cost, 663, that is from the cost table. Below that is the access path query blocks, you see a scan (an R) followed by index (I), scan, and another index access.

Now look at the XS entry – that is from the plan table after I did an explain using my hints. That shows a cost of 409 and if you look at the query blocks, one tablespace scan and three index accesses.

The last entry is our plan management that IBM provided a number of release ago. It is what is keep in the catalog and directory. This product does an EXPLAIN PACKAGE and you see in the other screen. The explain output shows the different HINTS used, for catalog and directory explains it does update this column just so you know.



How do I know if this is taking affect

PLAN_TABLE
QUERYNO = 472
OPTHINT = IDUG1

DYNAMIC SQL
SET CURRENT
OPTIMIZATION HINT
IDUG1

REBIND PACKAGE
OPTHINT(IDUG1)

PLAN_TABLE
QUERYNO = 472
OPTHINT = (IDUG1)
HINT_USED = IDUG1

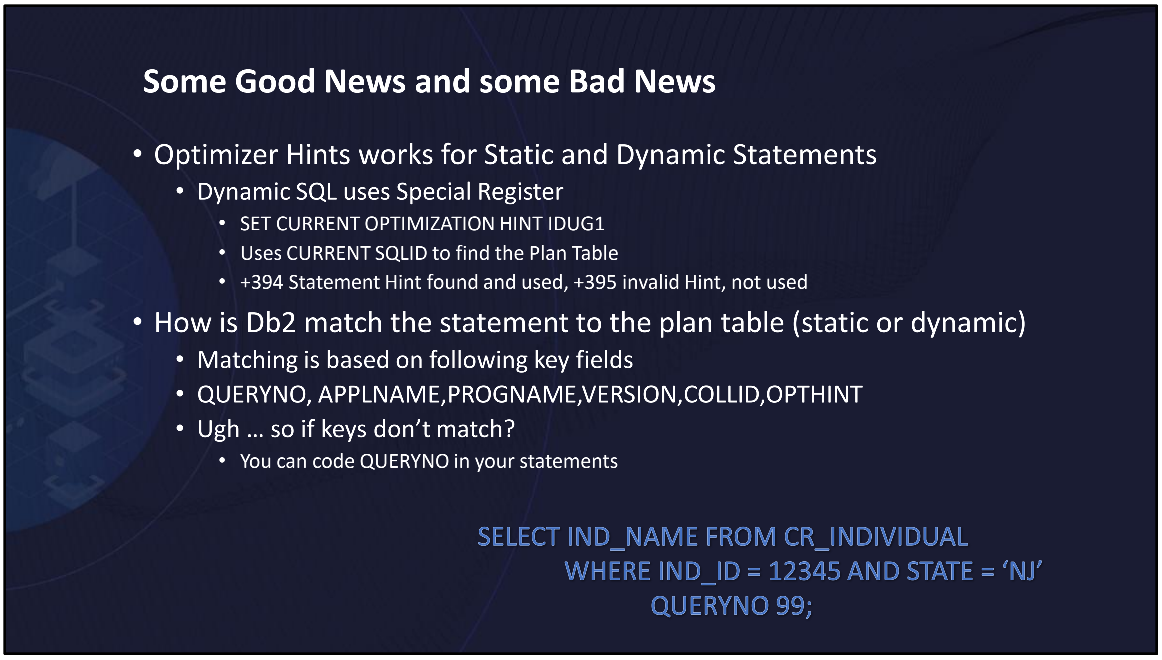
SYSPACKAGE
OPTHINT = IDUG1

SYSPACKSTMT
ACCESSPATH = H

Review what is going on with Optimization Hints by updating the plan table.

We will see updating the plan table is required in other solutions but not by updating the current query number
SYSPACKAGE
    OPTHINT column contains the IDUG1
SYSPACKSTMT
    ACCESSPATH column contains H
PLAN_TABLE
    We updated OPT_HINT with IDUG1
    HINT_USED column will also have IDUG1 if the hint was valid

Problems with this … what if plan table goes away and you need to rebind? But still want that access path. Need a plan B.

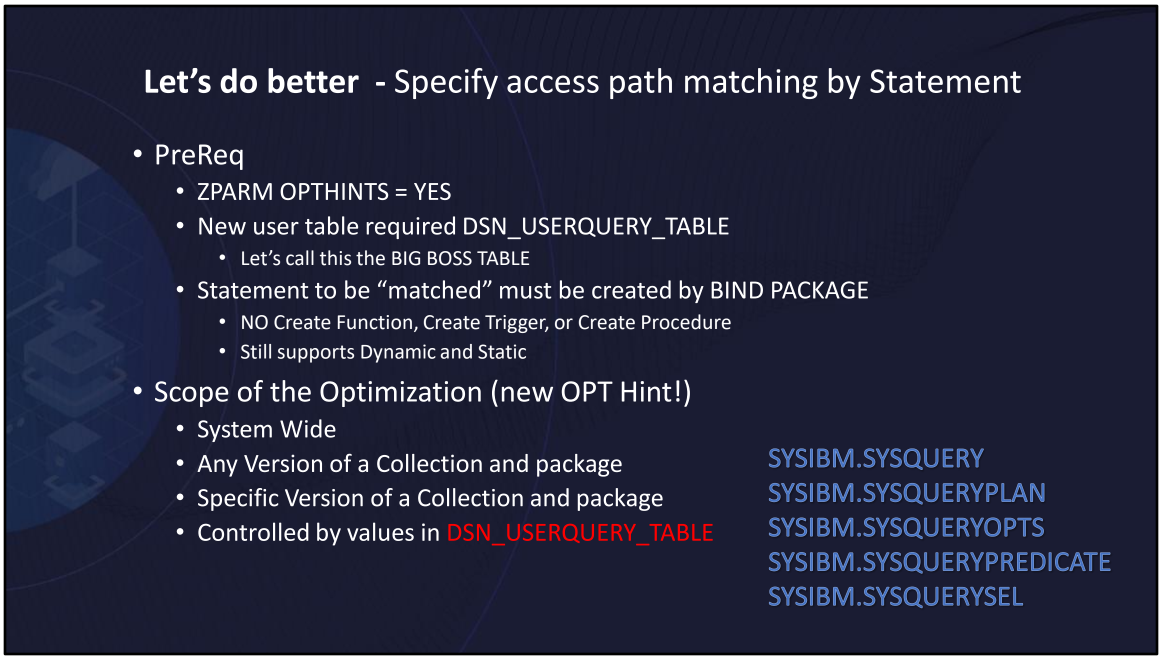


Some Good News and some Bad News

- Optimizer Hints works for Static and Dynamic Statements
  - Dynamic SQL uses Special Register
    - SET CURRENT OPTIMIZATION HINT IDUG1
    - Uses CURRENT SQLID to find the Plan Table
    - +394 Statement Hint found and used, +395 invalid Hint, not used
- How is Db2 match the statement to the plan table (static or dynamic)
  - Matching is based on following key fields
  - QUERYNO, APPLNAME,PROGNAME,VERSION,COLLID,OPTHINT
  - Ugh … so if keys don't match?
    - You can code QUERYNO in your statements

```
SELECT IND_NAME FROM CR_INDIVIDUAL
        WHERE IND_ID = 12345 AND STATE = 'NJ'
                QUERYNO 99;
```

Hold the phone … matching on queryno, applname, …. Opthint – get that from the rebind, go it. That could mean I could have many opthints and just change at bind time. Could be cool. But what if I change the code that could cause the queryno to change. You are right, you would lose the hint on new query numbers. Taking it to dynamic SQL, how can that work?



Let's do better - Specify access path matching by Statement

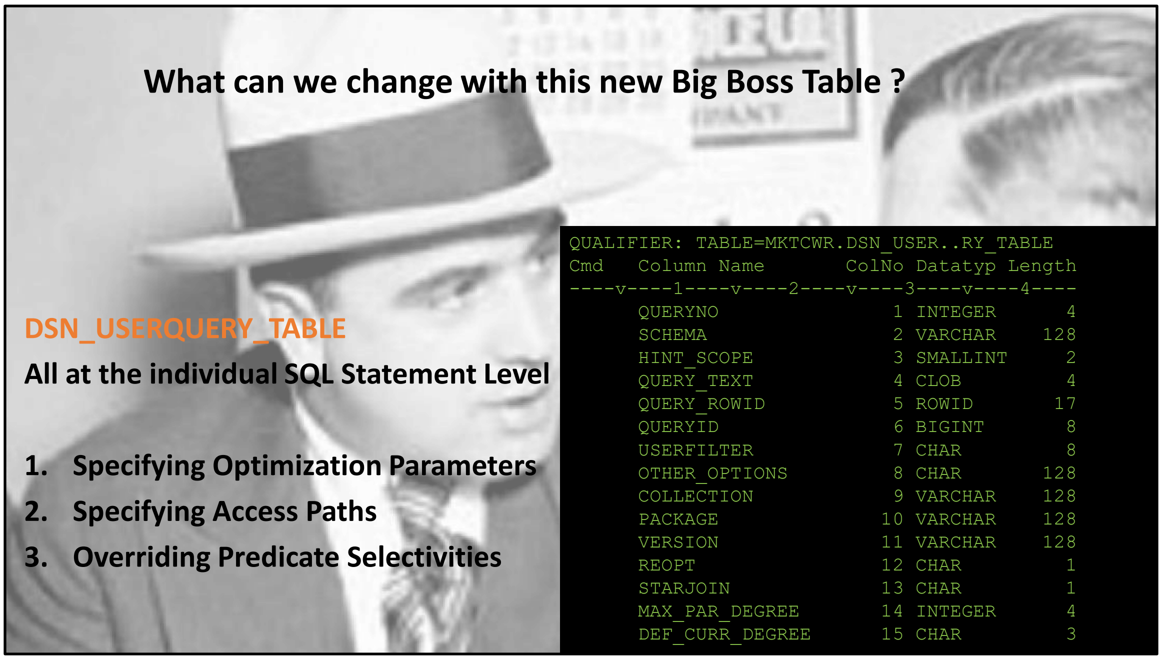
- PreReq
  - ZPARM OPTHINTS = YES
  - New user table required DSN_USERQUERY_TABLE
    - Let's call this the BIG BOSS TABLE
  - Statement to be "matched" must be created by BIND PACKAGE
    - NO Create Function, Create Trigger, or Create Procedure
    - Still supports Dynamic and Static
- Scope of the Optimization (new OPT Hint!)
  - System Wide
  - Any Version of a Collection and package
  - Specific Version of a Collection and package
  - Controlled by values in DSN_USERQUERY_TABLE

SYSIBM.SYSQUERY
SYSIBM.SYSQUERYPLAN
SYSIBM.SYSQUERYOPTS
SYSIBM.SYSQUERYPREDICATE
SYSIBM.SYSQUERYSEL

Along comes Db2 V10 and V11 and we can make these statement level access path changes better. And lets add some additional way Db2 can help.

CREATE statements for the tables and associated indexes in members DSNTESC and DSNTESH of the *prefix*.SDSNSAMP library.

Statement level hints were added in DB2 V10. It was a completely new method of providing hints. Based on the access paths kept within the Db2 Catalog.



What can we change with this new Big Boss Table ?

DSN_USERQUERY_TABLE

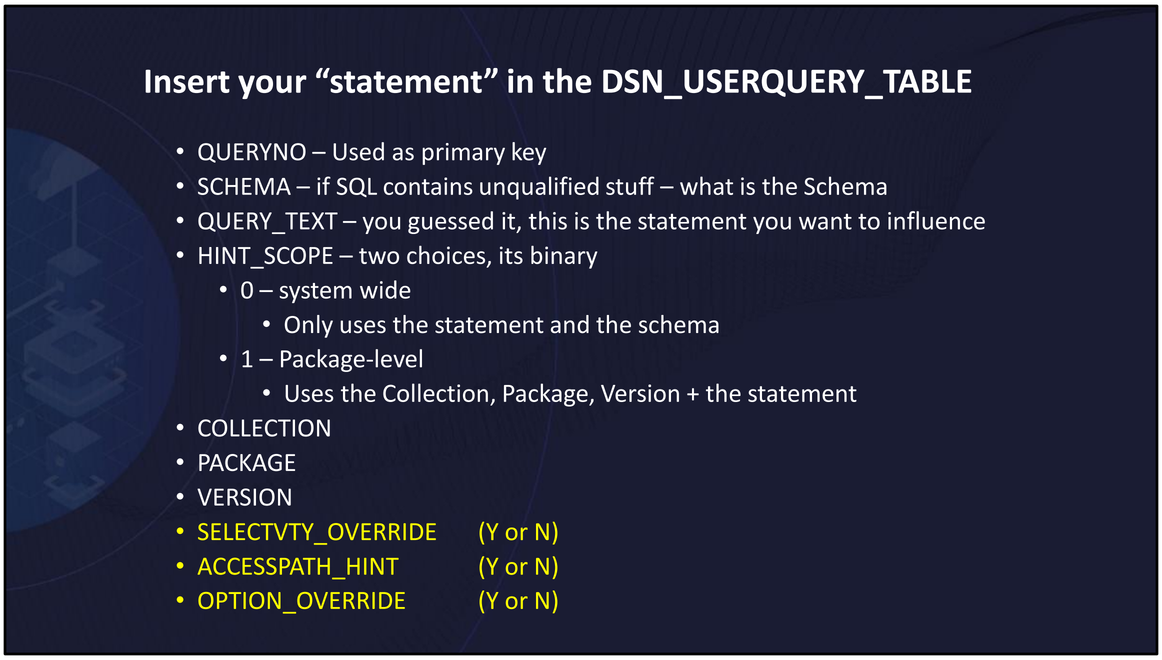
All at the individual SQL Statement Level

1. **Specifying Optimization Parameters**
2. **Specifying Access Paths**
3. **Overriding Predicate Selectivities**

```
QUALIFIER: TABLE=MKTCWR.DSN_USER..RY_TABLE
Cmd   Column Name        ColNo Datatyp Length
----v----1----v----2----v----3----v----4----
      QUERYNO                1 INTEGER      4
      SCHEMA                 2 VARCHAR    128
      HINT_SCOPE             3 SMALLINT     2
      QUERY_TEXT             4 CLOB         4
      QUERY_ROWID            5 ROWID       17
      QUERYID                6 BIGINT       8
      USERFILTER             7 CHAR         8
      OTHER_OPTIONS          8 CHAR       128
      COLLECTION             9 VARCHAR    128
      PACKAGE               10 VARCHAR    128
      VERSION               11 VARCHAR    128
      REOPT                 12 CHAR         1
      STARJOIN              13 CHAR         1
      MAX_PAR_DEGREE        14 INTEGER      4
      DEF_CURR_DEGREE       15 CHAR         3
```

CREATE statements for the tables and associated indexes in members DSNTESC and DSNTESH of the *prefix*.SDSNSAMP library.

Statement level hints were added in DB2 V10. It was a completely new method of providing hints. Based on the access paths kept within the Db2 Catalog.



Insert your “statement” in the DSN_USERQUERY_TABLE

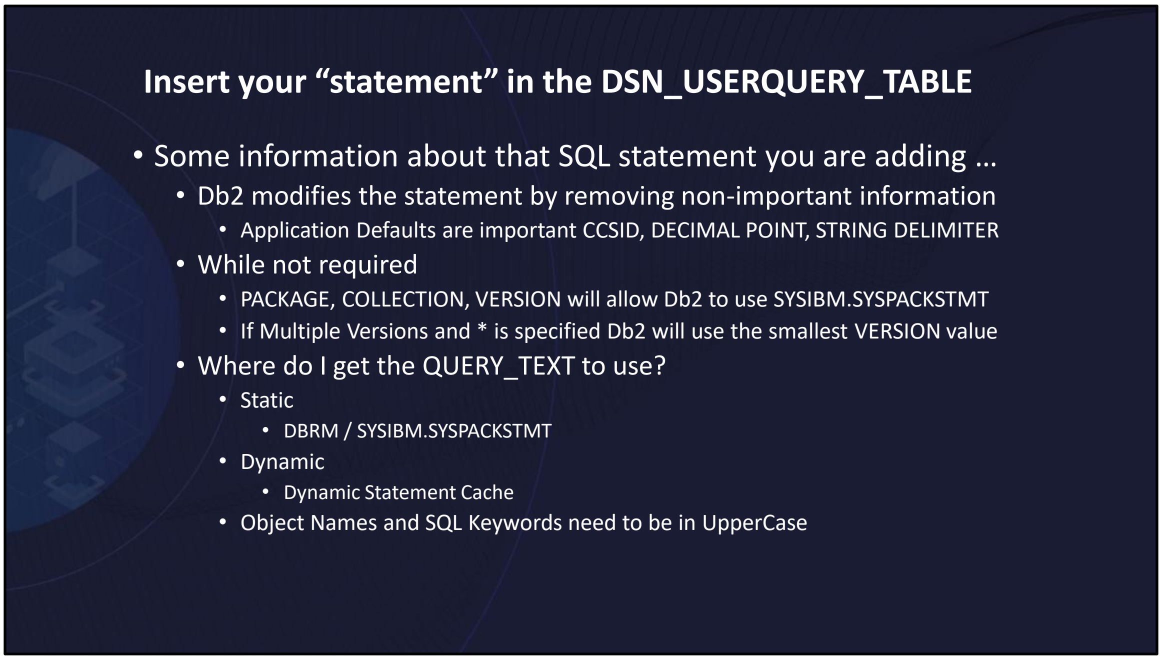
- QUERYNO – Used as primary key
- SCHEMA – if SQL contains unqualified stuff – what is the Schema
- QUERY_TEXT – you guessed it, this is the statement you want to influence
- HINT_SCOPE – two choices, its binary
  - 0 – system wide
    - Only uses the statement and the schema
  - 1 – Package-level
    - Uses the Collection, Package, Version + the statement
- COLLECTION
- PACKAGE
- VERSION
- SELECTVTY_OVERRIDE (Y or N)
- ACCESSPATH_HINT (Y or N)
- OPTION_OVERRIDE (Y or N)

You can wildcard the VERSION.

QUERYNO Specify any value that does not correlate to PLAN_TABLE rows and does not already exist in another DSN_USERQUERY_TABLE row. The QUERYNO value is used only for the primary key of DSN_USERQUERY_TABLE.SCHEMA. If the SQL statement contains unqualified object names that might resolve to different default schemas, insert the schema name that identifies the unqualified database objects. If the statement contains unqualified objects names because it might apply to different schemas at different times, you must create separate hints or overrides for each possible SCHEMA value. If the statement contains only fully qualified object names, the SCHEMA value is not required. However, you can still insert a SCHEMA value to help you identify that the hint relates to a certain schema.QUERY_TEXT.Insert the text of the statement whose access path you want to influence. The text that you provide must match the statement text that Db2 uses when binding static SQL statements and preparing dynamic SQL statements.
HINT_SCOPE Insert a value to specify that context in which to match the statement.
SELECTVTY_OVERRIDE Specify a value to indicate whether selectivity overrides are specified. Unless you specifically want to enable both option overrides and selectivity overrides for the same statement, specify 'N' in this column. If you want to specify

both types of overrides, specify 'Y'.
ACCESSPATH_HINT Specify a value of 'N' to indicate that no access path is specified. You cannot specify both access paths and selectivity overrides for the same statements.
OPTION_OVERRIDE Specify a value of 'Y' to indicate that statement-level optimization parameters are specified.

SELECTVTY_OVERRIDE is for predicate selectivities
ACCESSPATH_HINT is for a hint that is in the plan table – this is stand alone, you can not combine that with either of the other overrides
Option_overirid is for statement-level optimization parameters.

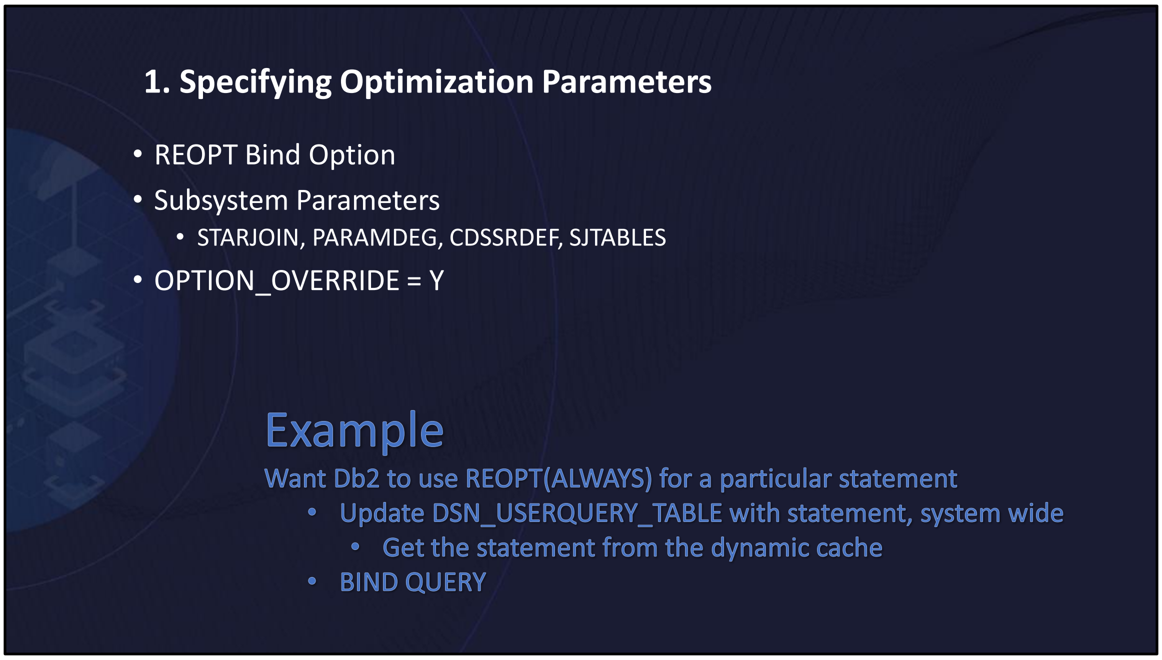

Insert your “statement” in the DSN_USERQUERY_TABLE

- Some information about that SQL statement you are adding …
  - Db2 modifies the statement by removing non-important information
    - Application Defaults are important CCSID, DECIMAL POINT, STRING DELIMITER
  - While not required
    - PACKAGE, COLLECTION, VERSION will allow Db2 to use SYSIBM.SYSPACKSTMT
    - If Multiple Versions and * is specified Db2 will use the smallest VERSION value
  - Where do I get the QUERY_TEXT to use?
    - Static
      - DBRM / SYSIBM.SYSPACKSTMT
    - Dynamic
      - Dynamic Statement Cache
    - Object Names and SQL Keywords need to be in UpperCase

For static SQL statements, and for dynamic statements that are prepared with the DYNAMICRULES(BIND) option, specify the following columns that specify package information for the statement in DSN_USERQUERY_TABLE:PACKAGE
COLLECTION
VERSION
These values are not strictly required. However, when these values are specified, Db2 uses parsing information from the SYSIBM.SYSPACKSTMT catalog table to modify the statement text. If the values are unspecified, or the matching package is not found during the BIND QUERY processing, Db2 uses the values that are specified in the application defaults module.
As part of the BIND QUERY process, Db2 validates that the package if specified, contains matching statement text. If the statement text does not match, Db2 issues message DSNT281I and the BIND QUERY command fails.
When multiple versions of the package exist, and you specify * for the value of the VERSION column. Db2 uses package information from the SYSIBM.SYSPACKSTMT catalog table that has the smallest value in the VERSION column to modify the statement text. If other versions of the package use different options, it is possible that for matching to fail for statements from the other versions.
When the package context is not specified in DSN_USERQUERY_TABLE, Db2 uses the

applications default module to modify the statement text. However, the statement text is not validated against statements in a particular package.
When you populate the QUERY_TEXT column in DSN_USERQUERY_TABLE, select the parsed query text from the following locations:For static SQL statements, select the statement text from the DBRM or from the SYSIBM.SYSPACKSTMT catalog table.
For dynamic SQL statements, select the statement text from the dynamic statement cache. For statements that are eligible for replacement of literal values by the ampersand symbol (&), extract the statement text after Db2 replaces literal values.
It is possible to specify the text directly in an INSERT statement (such as by copying from the source code for your application).

It is possible to specify the text directly in an INSERT statement (such as by copying from the source code for your application). However, that approach reduces the likelihood of successful matching of statements to the hint.



1. Specifying Optimization Parameters

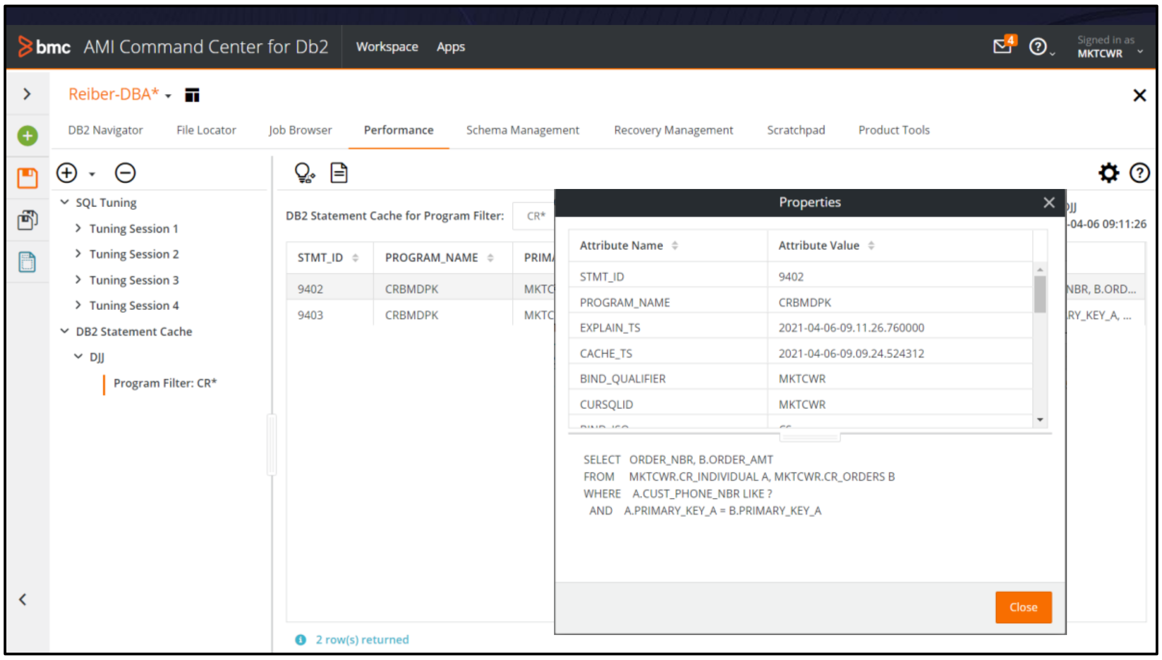
- REOPT Bind Option
- Subsystem Parameters
  - STARJOIN, PARAMDEG, CDSSRDEF, SJTABLES
- OPTION_OVERRIDE = Y

Example

Want Db2 to use REOPT(ALWAYS) for a particular statement

- Update DSN_USERQUERY_TABLE with statement, system wide
  - Get the statement from the dynamic cache
- BIND QUERY

STARJOIN subsystem parameter
PARAMDEG subsystem parameter (MAX_PAR_DEGREE column)
CDSSRDEF subsystem parameter (DEF_CURR_DEGREE column)
SJTABLES subsystem parameter

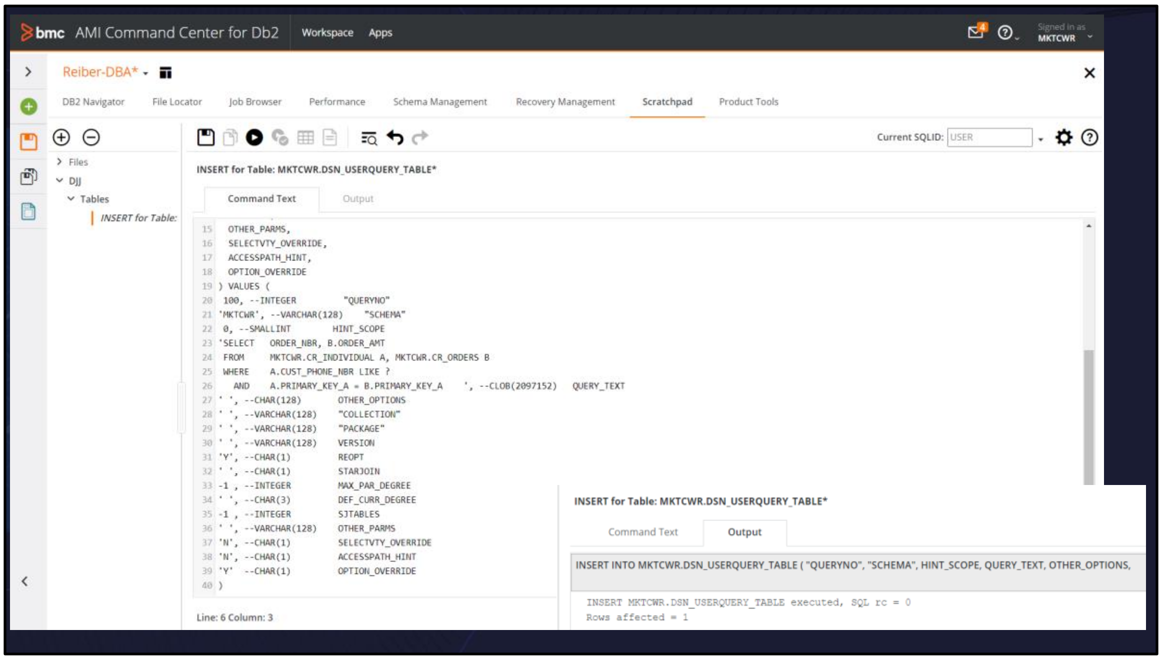


STMT_ID	PROGRAM_NAME
9402	CRBMDPK
9403	CRBMDPK

Attribute Name	Attribute Value
STMT_ID	9402
PROGRAM_NAME	CRBMDPK
EXPLAIN_TS	2021-04-06-09.11.26.760000
CACHE_TS	2021-04-06-09.09.24.524312
BIND_QUALIFIER	MKTCWR
CURSQLID	MKTCWR

```
SELECT   ORDER_NBR, B.ORDER_AMT
FROM     MKTCWR.CR_INDIVIDUAL A, MKTCWR.CR_ORDERS B
WHERE    A.CUST_PHONE_NBR LIKE ?
 AND     A.PRIMARY_KEY_A = B.PRIMARY_KEY_A
```

There are many ways to get the statement from dynamic cache but going to use this to look at the statements based on package/program name. Here I have my statement and I will cut and paste it to my update of the big boss table


INSERT for Table: MKTCWR.DSN_USERQUERY_TABLE*

```
 OTHER_PARMS,
 SELECTVTY_OVERRIDE,
 ACCESSPATH_HINT,
 OPTION_OVERRIDE
) VALUES (
 100, --INTEGER         "QUERYNO"
'MKTCWR', --VARCHAR(128)    "SCHEMA"
 0, --SMALLINT        HINT_SCOPE
'SELECT   ORDER_NBR, B.ORDER_AMT
 FROM     MKTCWR.CR_INDIVIDUAL A, MKTCWR.CR_ORDERS B
 WHERE    A.CUST_PHONE_NBR LIKE ?
   AND    A.PRIMARY_KEY_A = B.PRIMARY_KEY_A    ', --CLOB(2097152)   QUERY_TEXT
' ', --CHAR(128)       OTHER_OPTIONS
' ', --VARCHAR(128)    "COLLECTION"
' ', --VARCHAR(128)    "PACKAGE"
' ', --VARCHAR(128)    VERSION
'Y', --CHAR(1)         REOPT
' ', --CHAR(1)         STARJOIN
-1 , --INTEGER         MAX_PAR_DEGREE
' ', --CHAR(3)         DEF_CURR_DEGREE
-1 , --INTEGER         SJTABLES
' ', --VARCHAR(128)    OTHER_PARMS
'N', --CHAR(1)         SELECTVTY_OVERRIDE
'N', --CHAR(1)         ACCESSPATH_HINT
'Y'  --CHAR(1)         OPTION_OVERRIDE
)
```

Line: 6 Column: 3

INSERT for Table: MKTCWR.DSN_USERQUERY_TABLE*

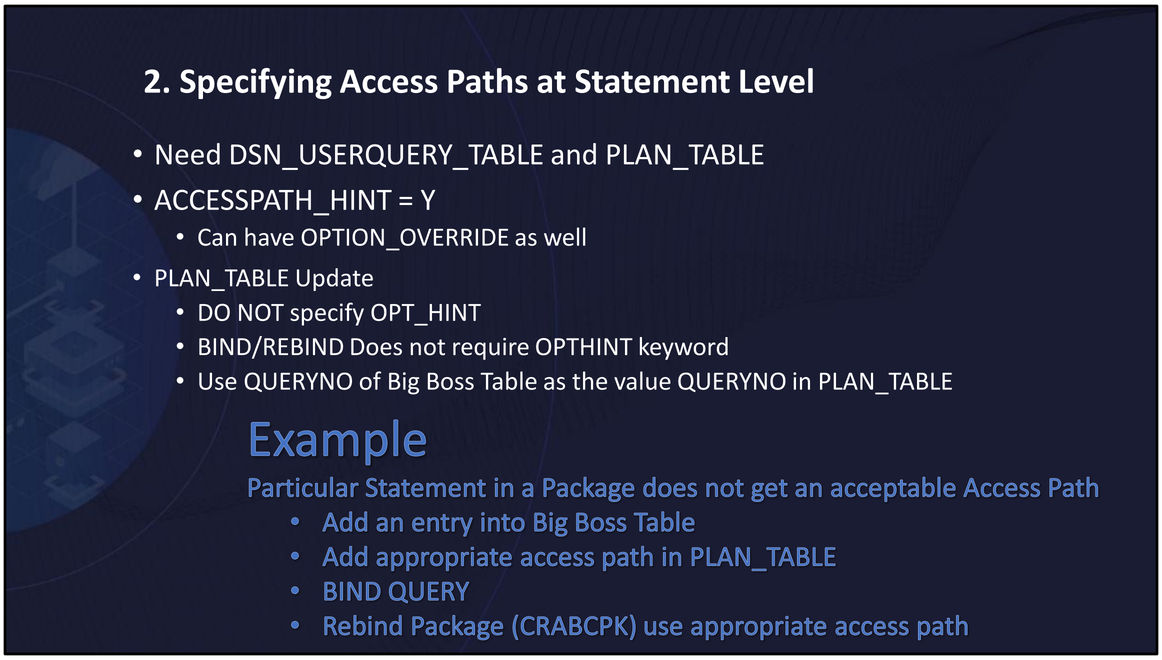
```
INSERT INTO MKTCWR.DSN_USERQUERY_TABLE ( "QUERYNO", "SCHEMA", HINT_SCOPE, QUERY_TEXT, OTHER_OPTIONS,

INSERT MKTCWR.DSN_USERQUERY_TABLE executed, SQL rc = 0
Rows affected = 1
```

I went to dynamic statement cache to find the statement I was interested in. and copied that statement into the insert into the big boss table.

Made it system wide by setting the HINT SCOPE to 0.

This is not based on a static package so that is left blank.

Remember I am going for Option Override with this statement.



2. Specifying Access Paths at Statement Level

- Need DSN_USERQUERY_TABLE and PLAN_TABLE
- ACCESSPATH_HINT = Y
  - Can have OPTION_OVERRIDE as well
- PLAN_TABLE Update
  - DO NOT specify OPT_HINT
  - BIND/REBIND Does not require OPTHINT keyword
  - Use QUERYNO of Big Boss Table as the value QUERYNO in PLAN_TABLE

Example

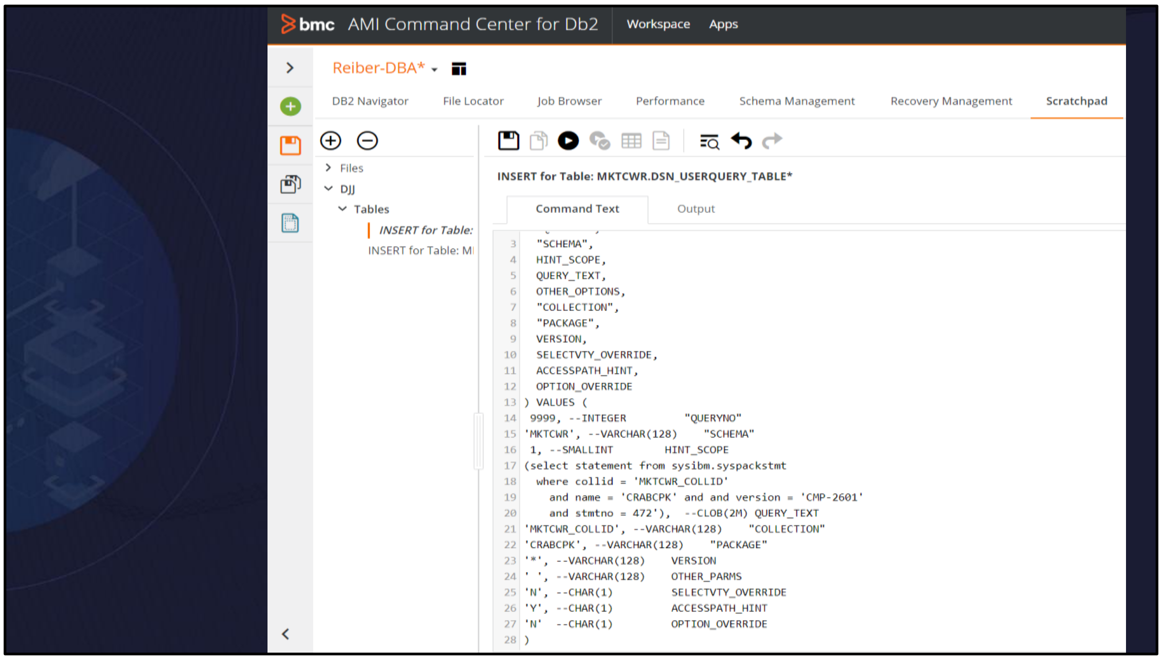
Particular Statement in a Package does not get an acceptable Access Path

- Add an entry into Big Boss Table
- Add appropriate access path in PLAN_TABLE
- BIND QUERY
- Rebind Package (CRABCPK) use appropriate access path

Here we are going to tell Db2 this is the access path I want you to use … you do that by specifying the access path in the plan table.

With any of these “hints” you need to have the particular access path in some plan table someplace. You typically can not just code up an access path, there are specific data IBM provides in the plan table for it to work.

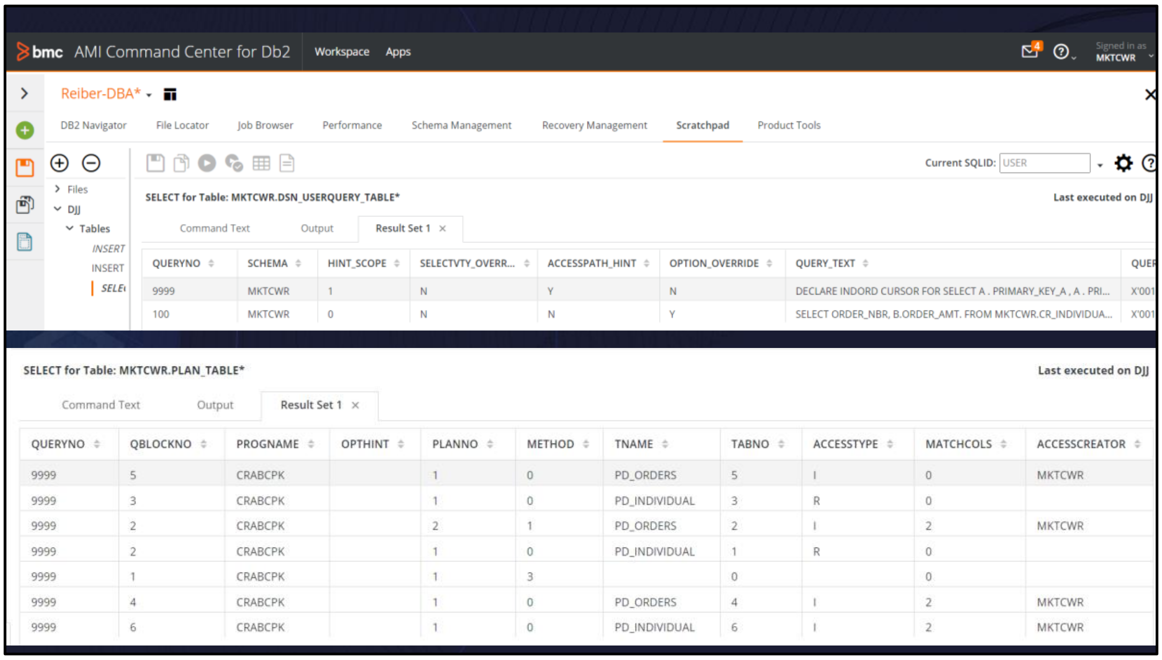
This the version of the old V6 Optimization Hints but a bit different where you don’t need to match on query number.



INSERT for Table: MKTCWR.DSN_USERQUERY_TABLE*

```
  "SCHEMA",
  HINT_SCOPE,
  QUERY_TEXT,
  OTHER_OPTIONS,
  "COLLECTION",
  "PACKAGE",
  VERSION,
  SELECTVTY_OVERRIDE,
  ACCESSPATH_HINT,
  OPTION_OVERRIDE
) VALUES (
 9999, --INTEGER          "QUERYNO"
'MKTCWR', --VARCHAR(128)    "SCHEMA"
 1, --SMALLINT         HINT_SCOPE
(select statement from sysibm.syspackstmt
  where collid = 'MKTCWR_COLLID'
    and name = 'CRABCPK' and and version = 'CMP-2601'
    and stmtno = 472'),  --CLOB(2M) QUERY_TEXT
'MKTCWR_COLLID', --VARCHAR(128)    "COLLECTION"
'CRABCPK', --VARCHAR(128)    "PACKAGE"
'*', --VARCHAR(128)    VERSION
' ', --VARCHAR(128)    OTHER_PARMS
'N', --CHAR(1)         SELECTVTY_OVERRIDE
'Y', --CHAR(1)         ACCESSPATH_HINT
'N'  --CHAR(1)         OPTION_OVERRIDE
)
```

Remember the old opt hints you needed the query number to match up the row in the plan table to provide the hint. Here db2 is using the statement itself to match up. But to get the statement I am copying the statement right from sys pack stmt.



SELECT for Table: MKTCWR.DSN_USERQUERY_TABLE*

Last executed on DJJ

QUERYNO	SCHEMA	HINT_SCOPE	SELECTVTY_OVERR...	ACCESSPATH_HINT	OPTION_OVERRIDE	QUERY_TEXT	QUER
9999	MKTCWR	1	N	Y	N	DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRI...	X'001
100	MKTCWR	0	N	N	Y	SELECT ORDER_NBR, B.ORDER_AMT. FROM MKTCWR.CR_INDIVIDUA...	X'001

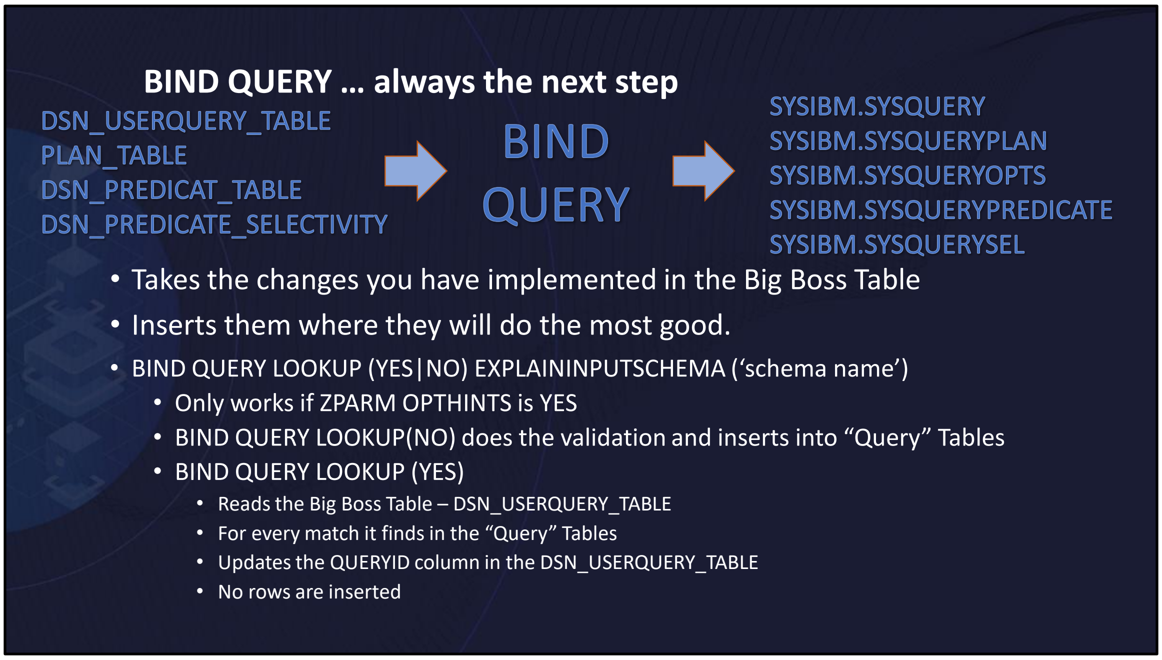
SELECT for Table: MKTCWR.PLAN_TABLE*

Last executed on DJJ

QUERYNO	QBLOCKNO	PROGNAME	OPTHINT	PLANNO	METHOD	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSCREATOR
9999	5	CRABCPK		1	0	PD_ORDERS	5	I	0	MKTCWR
9999	3	CRABCPK		1	0	PD_INDIVIDUAL	3	R	0	
9999	2	CRABCPK		2	1	PD_ORDERS	2	I	2	MKTCWR
9999	2	CRABCPK		1	0	PD_INDIVIDUAL	1	R	0	
9999	1	CRABCPK		1	3		0		0	
9999	4	CRABCPK		1	0	PD_ORDERS	4	I	2	MKTCWR
9999	6	CRABCPK		1	0	PD_INDIVIDUAL	6	I	2	MKTCWR

Now that the big boss table is updated with a query no of your choice. Where is the access path we want the statement to use. It is of course in the plan table.

Using query number 9999 … matches the statement with the plan table access path. Yes both of these tables must have the same schema/owner.



BIND QUERY … always the next step

DSN_USERQUERY_TABLE
PLAN_TABLE
DSN_PREDICAT_TABLE
DSN_PREDICATE_SELECTIVITY

→ BIND QUERY →

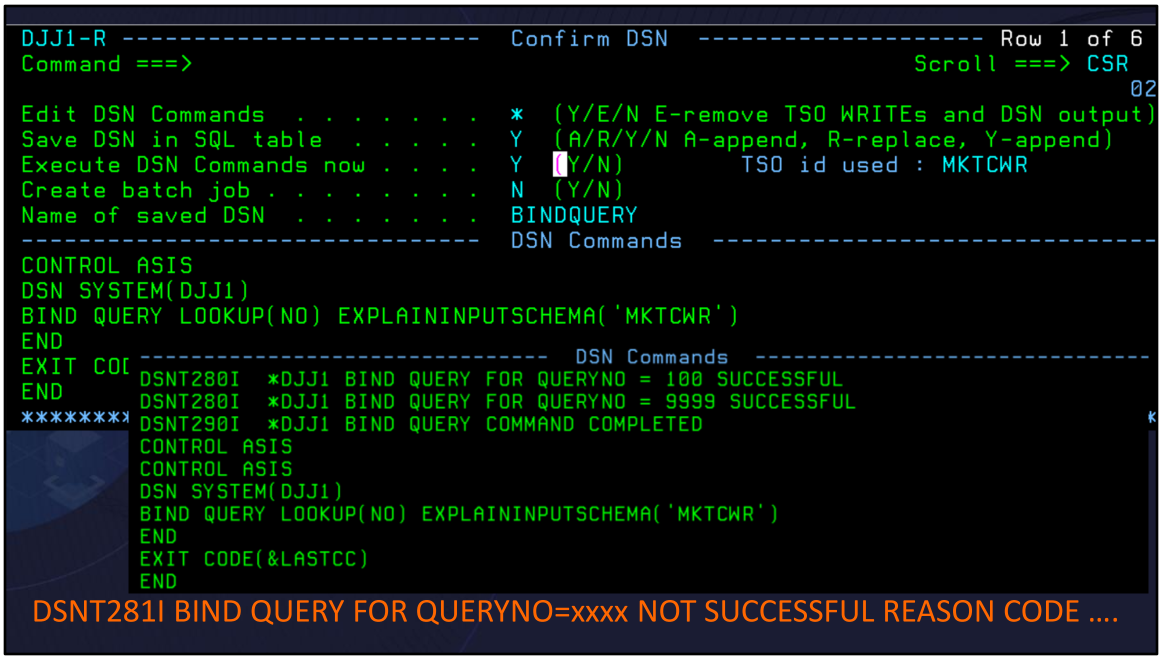
SYSIBM.SYSQUERY
SYSIBM.SYSQUERYPLAN
SYSIBM.SYSQUERYOPTS
SYSIBM.SYSQUERYPREDICATE
SYSIBM.SYSQUERYSEL

- Takes the changes you have implemented in the Big Boss Table
- Inserts them where they will do the most good.
- BIND QUERY LOOKUP (YES|NO) EXPLAININPUTSCHEMA ('schema name')
  - Only works if ZPARM OPTHINTS is YES
  - BIND QUERY LOOKUP(NO) does the validation and inserts into "Query" Tables
  - BIND QUERY LOOKUP (YES)
    - Reads the Big Boss Table – DSN_USERQUERY_TABLE
    - For every match it finds in the "Query" Tables
    - Updates the QUERYID column in the DSN_USERQUERY_TABLE
    - No rows are inserted

Nothing happens until you execute the BIND QUERY statement. It takes the information from our DSN tables

The important BIND QUERY command is BIND QUERY LOOKUP(NO) – doesn't really make sense but you LOOKUP NO.

EXPLAININPUTSCHEMA keyword comes into play when the tables such as DSN USERQUERY or PLAN TABLE are different than the userid that is issuing the commands.



```
DJJ1-R -------------------------  Confirm DSN  -------------------- Row 1 of 6
Command ===>                                                   Scroll ===> CSR
                                                                            02
Edit DSN Commands . . . . . . . .  *  (Y/E/N E-remove TSO WRITEs and DSN output)
Save DSN in SQL table  . . . . . . Y  (A/R/Y/N A-append, R-replace, Y-append)
Execute DSN Commands now . . . . . Y  (Y/N)        TSO id used : MKTCWR
Create batch job . . . . . . . . . N  (Y/N)
Name of saved DSN  . . . . . . . . BINDQUERY
--------------------------------   DSN Commands  -------------------------------
CONTROL ASIS
DSN SYSTEM(DJJ1)
BIND QUERY LOOKUP(NO) EXPLAININPUTSCHEMA('MKTCWR')
END
EXIT COI
END
********
```

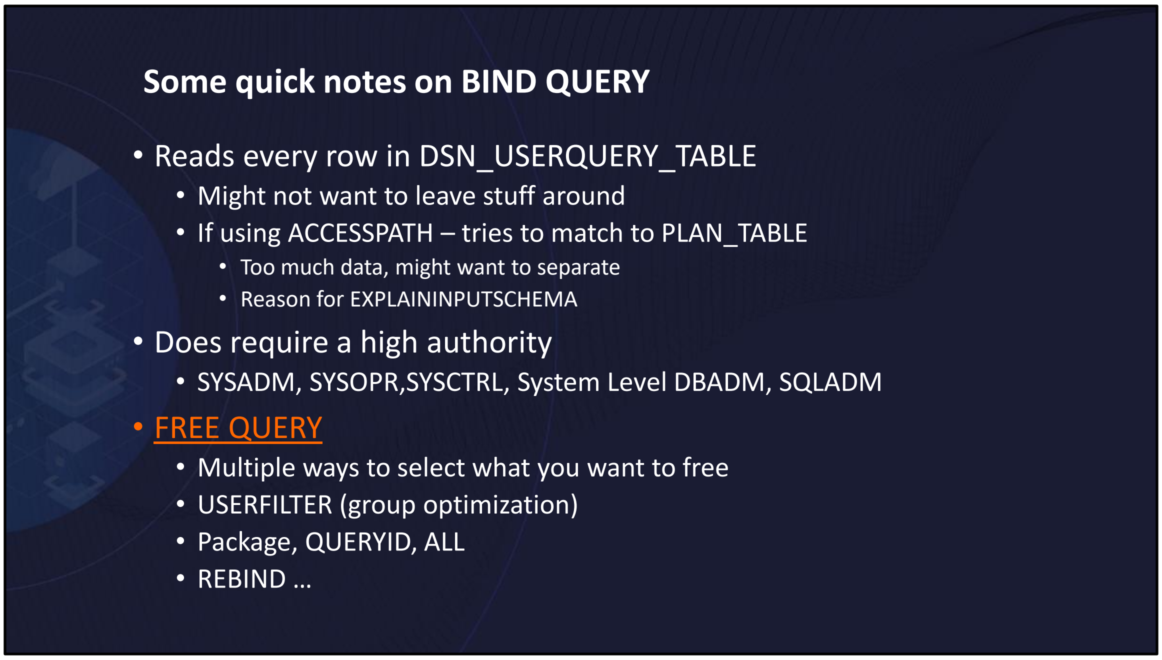
```
--------------------------------  DSN Commands  ------------------------------
DSNT280I  *DJJ1 BIND QUERY FOR QUERYNO = 100 SUCCESSFUL
DSNT280I  *DJJ1 BIND QUERY FOR QUERYNO = 9999 SUCCESSFUL
DSNT290I  *DJJ1 BIND QUERY COMMAND COMPLETED
CONTROL ASIS
CONTROL ASIS
DSN SYSTEM(DJJ1)
BIND QUERY LOOKUP(NO) EXPLAININPUTSCHEMA('MKTCWR')
END
EXIT CODE(&LASTCC)
END
```

DSNT281I BIND QUERY FOR QUERYNO=xxxx NOT SUCCESSFUL REASON CODE ….

I updated the big boss table, I added the access path to the plan table. Now I want to implement my work.

On the direction of the big boss, going to pull the trigger and issue the bind query

Things can go wrong … and Db2 will tell you.

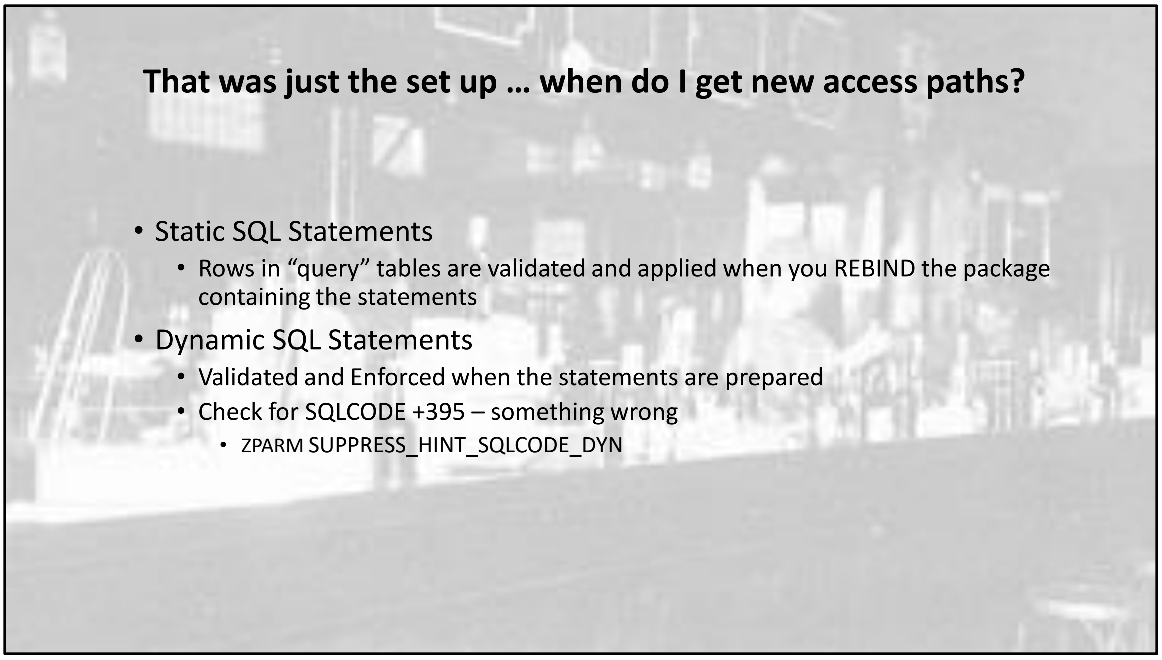

Some quick notes on BIND QUERY

- Reads every row in DSN_USERQUERY_TABLE
  - Might not want to leave stuff around
  - If using ACCESSPATH – tries to match to PLAN_TABLE
    - Too much data, might want to separate
    - Reason for EXPLAININPUTSCHEMA
- Does require a high authority
  - SYSADM, SYSOPR,SYSCTRL, System Level DBADM, SQLADM
- FREE QUERY
  - Multiple ways to select what you want to free
  - USERFILTER (group optimization)
  - Package, QUERYID, ALL
  - REBIND …

Some notes on BIND QUERY … not a command we issue every day.

But what if we want to clean up stuff that is now in the catalog and get in our way going forward?

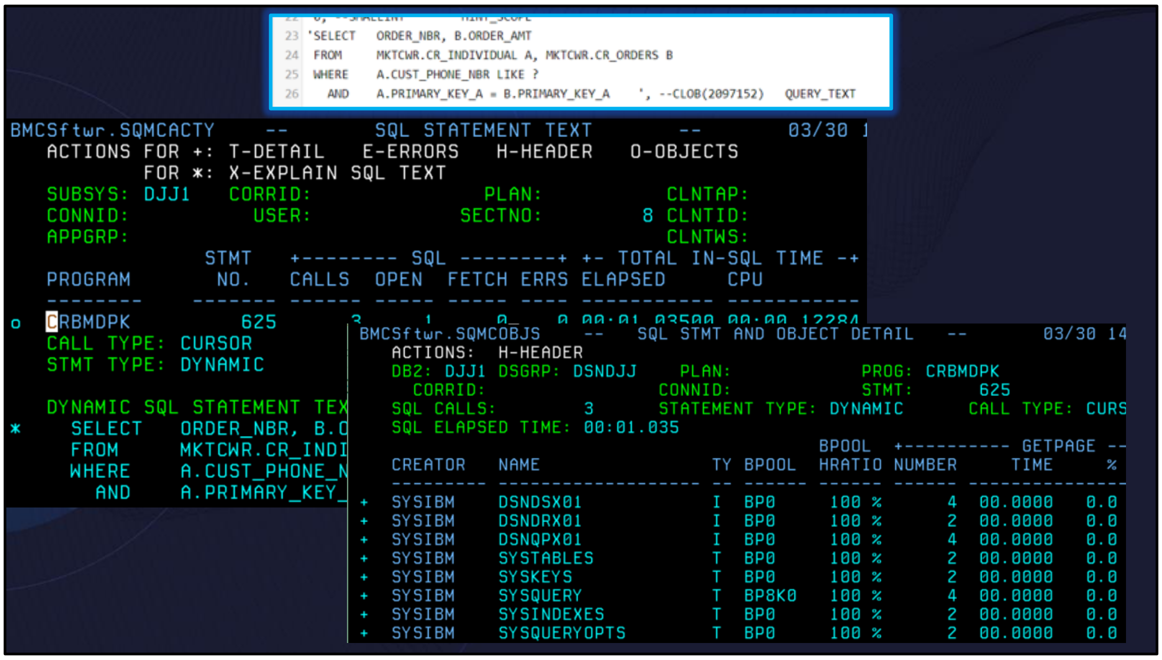
FREE QUERY will remove entries.



That was just the set up … when do I get new access paths?

- Static SQL Statements
  - Rows in “query” tables are validated and applied when you REBIND the package containing the statements
- Dynamic SQL Statements
  - Validated and Enforced when the statements are prepared
  - Check for SQLCODE +395 – something wrong
    - ZPARM SUPPRESS_HINT_SQLCODE_DYN

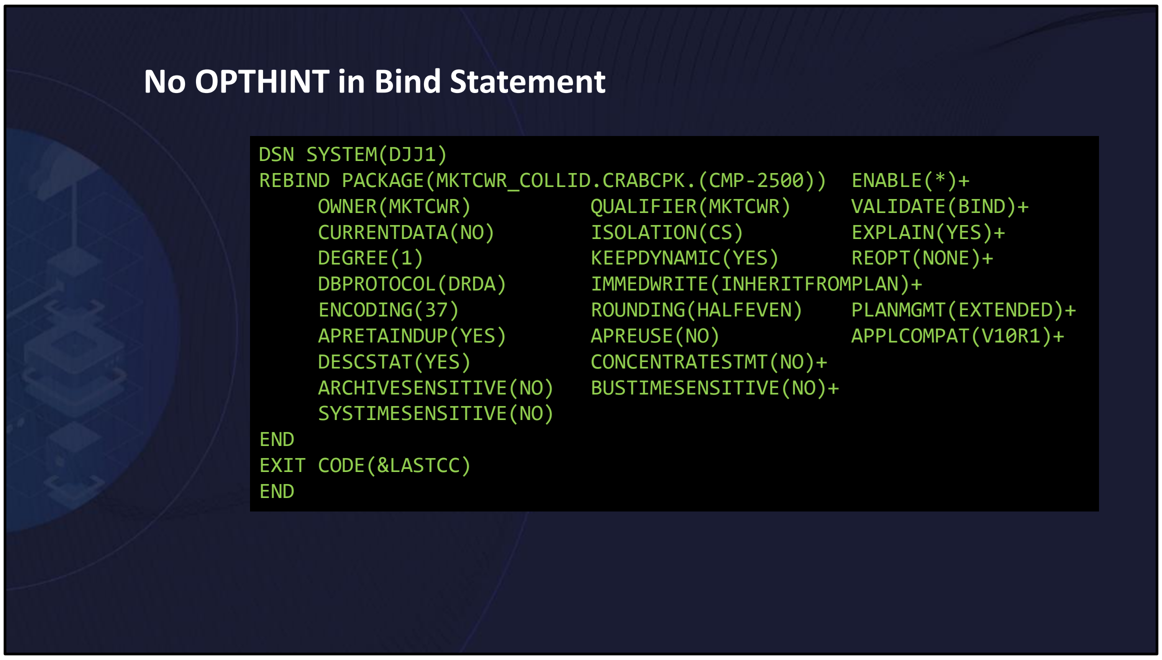
BIND Query did nothing else but put rows in the QUERY catalog tables. We have to get them to help us.



```
 0, --SMALLINT          HINT_SCOPE
'SELECT   ORDER_NBR, B.ORDER_AMT
 FROM     MKTCWR.CR_INDIVIDUAL A, MKTCWR.CR_ORDERS B
 WHERE    A.CUST_PHONE_NBR LIKE ?
   AND    A.PRIMARY_KEY_A = B.PRIMARY_KEY_A    ', --CLOB(2097152)   QUERY_TEXT
```

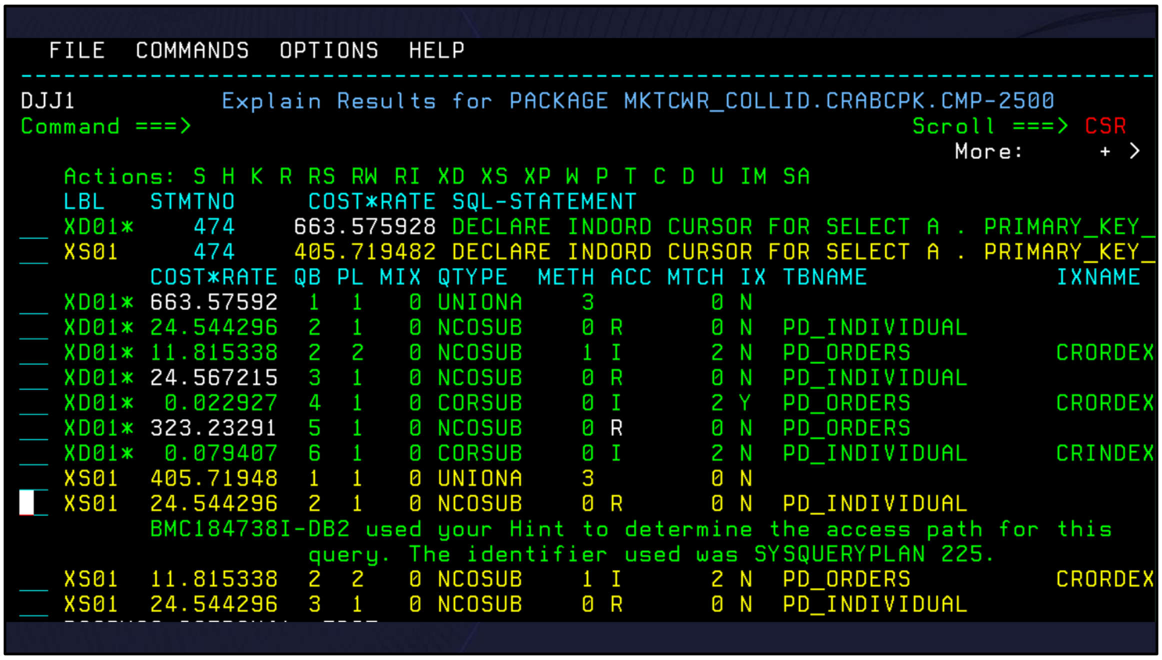
```
BMCSftwr.SQMCACTY    --        SQL STATEMENT TEXT        --       03/30 1
   ACTIONS FOR +: T-DETAIL   E-ERRORS   H-HEADER   O-OBJECTS
           FOR *: X-EXPLAIN SQL TEXT
   SUBSYS: DJJ1    CORRID:                PLAN:           CLNTAP:
   CONNID:           USER:               SECTNO:         8 CLNTID:
   APPGRP:                                                 CLNTWS:
                 STMT    +-------- SQL --------+ +- TOTAL IN-SQL TIME -+
   PROGRAM        NO.    CALLS  OPEN  FETCH ERRS ELAPSED      CPU
   --------     ------- ------ ----- ----- ---- ----------- -----------
o  CRBMDPK         625       3     1     0_    0 00:01.03500 00:00.12284
   CALL TYPE: CURSOR
   STMT TYPE: DYNAMIC

   DYNAMIC SQL STATEMENT TEX
*    SELECT    ORDER_NBR, B.O
     FROM      MKTCWR.CR_INDI
     WHERE     A.CUST_PHONE_N
       AND     A.PRIMARY_KEY_
```

```
BMCSftwr.SQMCOBJS    --   SQL STMT AND OBJECT DETAIL   --        03/30 14
   ACTIONS:  H-HEADER
   DB2: DJJ1 DSGRP: DSNDJJ    PLAN:                PROG: CRBMDPK
     CORRID:                 CONNID:               STMT:       625
   SQL CALLS:          3     STATEMENT TYPE: DYNAMIC      CALL TYPE: CURS
   SQL ELAPSED TIME: 00:01.035
                                                BPOOL  +---------- GETPAGE --
   CREATOR   NAME                  TY BPOOL  HRATIO NUMBER      TIME       %
   --------- ------------------- -- ------ ------ ------ ---------------
+  SYSIBM    DSNDSX01              I  BP0      100 %      4  00.0000    0.0
+  SYSIBM    DSNDRX01              I  BP0      100 %      2  00.0000    0.0
+  SYSIBM    DSNQPX01              I  BP0      100 %      4  00.0000    0.0
+  SYSIBM    SYSTABLES             T  BP0      100 %      2  00.0000    0.0
+  SYSIBM    SYSKEYS               T  BP0      100 %      2  00.0000    0.0
+  SYSIBM    SYSQUERY              T  BP8K0    100 %      4  00.0000    0.0
+  SYSIBM    SYSINDEXES            T  BP0      100 %      2  00.0000    0.0
+  SYSIBM    SYSQUERYOPTS          T  BP0      100 %      2  00.0000    0.0
```



No OPTHINT in Bind Statement

```
DSN SYSTEM(DJJ1)
REBIND PACKAGE(MKTCWR_COLLID.CRABCPK.(CMP-2500))  ENABLE(*)+
     OWNER(MKTCWR)          QUALIFIER(MKTCWR)     VALIDATE(BIND)+
     CURRENTDATA(NO)        ISOLATION(CS)         EXPLAIN(YES)+
     DEGREE(1)              KEEPDYNAMIC(YES)      REOPT(NONE)+
     DBPROTOCOL(DRDA)       IMMEDWRITE(INHERITFROMPLAN)+
     ENCODING(37)           ROUNDING(HALFEVEN)    PLANMGMT(EXTENDED)+
     APRETAINDUP(YES)       APREUSE(NO)           APPLCOMPAT(V10R1)+
     DESCSTAT(YES)          CONCENTRATESTMT(NO)+
     ARCHIVESENSITIVE(NO)   BUSTIMESENSITIVE(NO)+
     SYSTIMESENSITIVE(NO)
END
EXIT CODE(&LASTCC)
END
```

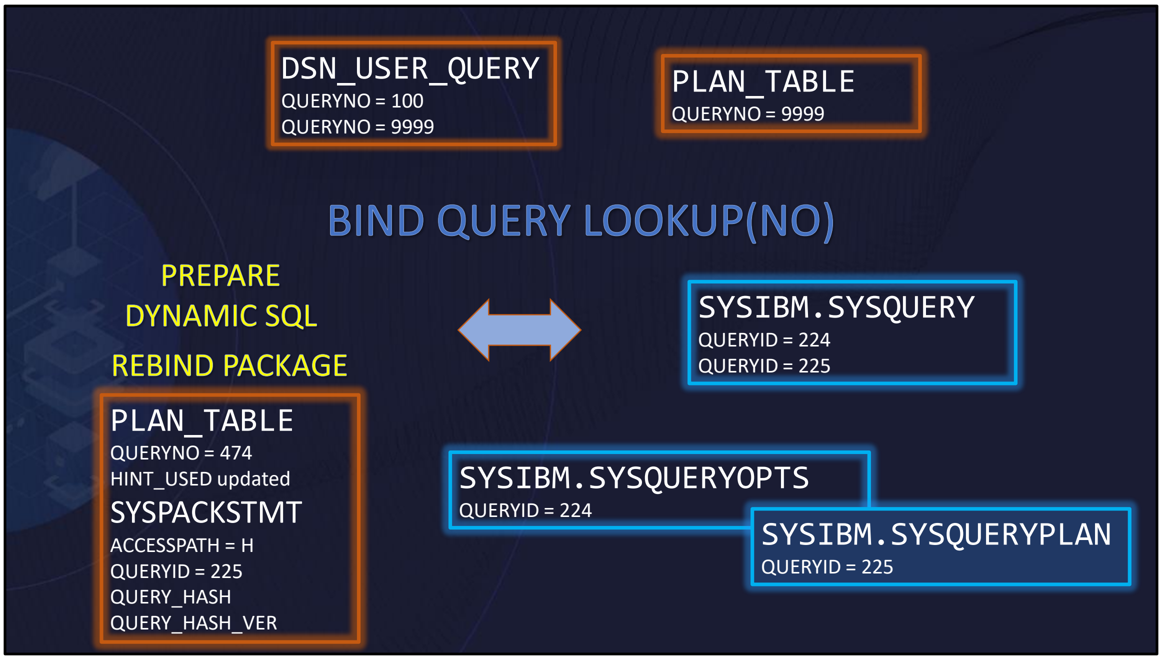


```
  FILE  COMMANDS  OPTIONS  HELP
------------------------------------------------------------------------------
DJJ1          Explain Results for PACKAGE MKTCWR_COLLID.CRABCPK.CMP-2500
Command ===>                                                   Scroll ===> CSR
                                                                  More:     + >
   Actions: S H K R RS RW RI XD XS XP W P T C D U IM SA
   LBL   STMTNO      COST*RATE SQL-STATEMENT
__ XD01*    474     663.575928 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_
__ XS01     474     405.719482 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_
         COST*RATE QB PL MIX QTYPE   METH ACC MTCH IX TBNAME             IXNAME
__ XD01* 663.57592  1  1   0 UNIONA     3         0 N
__ XD01* 24.544296  2  1   0 NCOSUB     0 R       0 N  PD_INDIVIDUAL
__ XD01* 11.815338  2  2   0 NCOSUB     1 I       2 N  PD_ORDERS          CRORDEX
__ XD01* 24.567215  3  1   0 NCOSUB     0 R       0 N  PD_INDIVIDUAL
__ XD01*  0.022927  4  1   0 CORSUB     0 I       2 Y  PD_ORDERS          CRORDEX
__ XD01* 323.23291  5  1   0 NCOSUB     0 R       0 N  PD_ORDERS
__ XD01*  0.079407  6  1   0 CORSUB     0 I       2 N  PD_INDIVIDUAL      CRINDEX
__ XS01  405.71948  1  1   0 UNIONA     3         0 N
__ XS01  24.544296  2  1   0 NCOSUB     0 R       0 N  PD_INDIVIDUAL
         BMC184738I-DB2 used your Hint to determine the access path for this
                    query. The identifier used was SYSQUERYPLAN 225.
__ XS01  11.815338  2  2   0 NCOSUB     1 I       2 N  PD_ORDERS          CRORDEX
__ XS01  24.544296  3  1   0 NCOSUB     0 R       0 N  PD_INDIVIDUAL
```

Proof points with rebinds … with my update to big boss table I did not provide a version. So any rebind of any version of the package would pick up my access path.

In this version of the package the statement I needed to change was statement 474 – save type of thing.

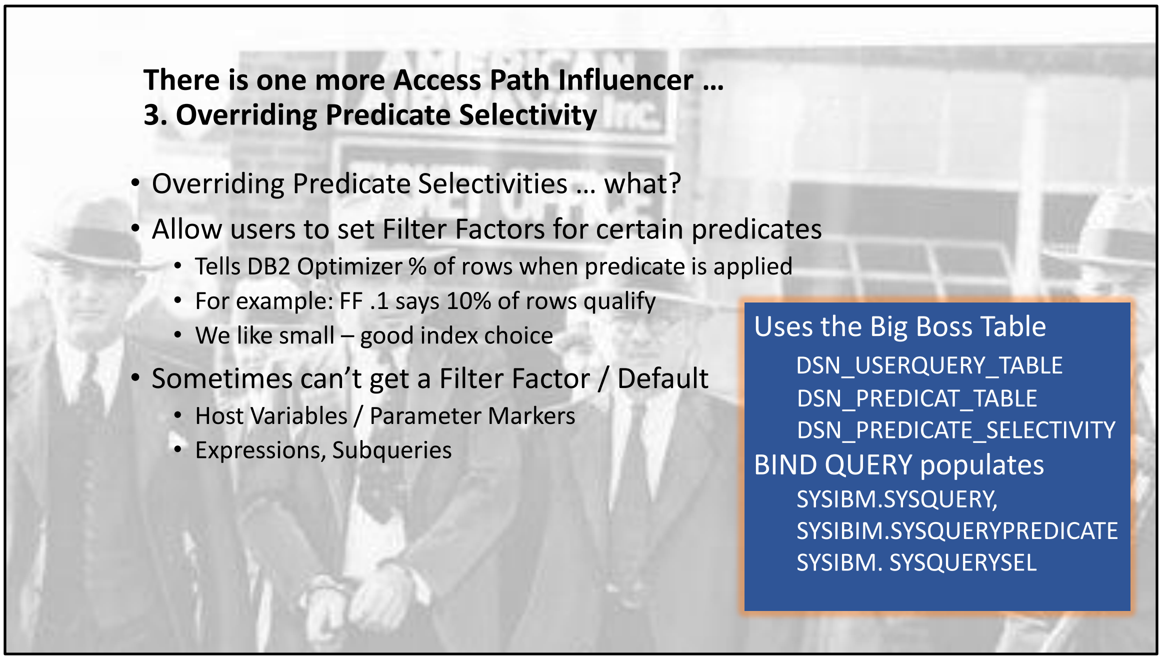
Notice however that the hint that was used to determine the access path points to the catalog table SYSQUERYPLAN 225. 225 being the statement in that table.



Connection from DSN_USER_QUERY to SYSQUERY is lost once BIND QUERY LOOKUP NO is run. That is why you might want to run LOOKUP YES

But this is a review of what is happening out there.

You need to make sure sysibm.sysquery is maintained or you can have stale stuff out there.



There is one more Access Path Influencer …
3. Overriding Predicate Selectivity

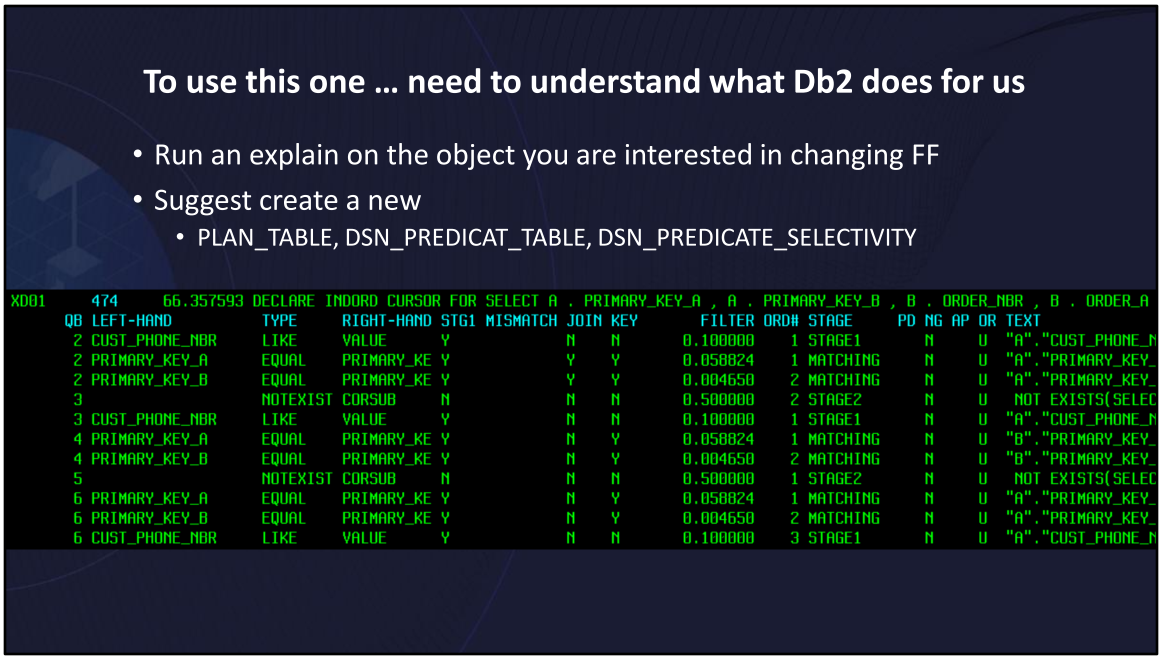
- Overriding Predicate Selectivities … what?
- Allow users to set Filter Factors for certain predicates
  - Tells DB2 Optimizer % of rows when predicate is applied
  - For example: FF .1 says 10% of rows qualify
  - We like small – good index choice
- Sometimes can't get a Filter Factor / Default
  - Host Variables / Parameter Markers
  - Expressions, Subqueries

Uses the Big Boss Table
DSN_USERQUERY_TABLE
DSN_PREDICAT_TABLE
DSN_PREDICATE_SELECTIVITY
BIND QUERY populates
SYSIBM.SYSQUERY,
SYSIBIM.SYSQUERYPREDICATE
SYSIBM. SYSQUERYSEL

you can override these default filter factors for certain predicates by creating selectivity overrides. Each *predicate selectivity override* describes the selectivity of a particular predicate in a particular SQL statement. When a statement contains more than one predicate, you can create separate selectivity overrides for each predicate in the statement.
A statement that is issued many times might have different filtering characteristics at different times. A predicate that filters many rows with one literal value might filter far fewer rows when the literal value is different. Therefore, a single set of overrides for a statement might not adequately describe the filtering of the predicates across all executions. So, you can create more than one set of overrides for each statement. Each set of overrides is a *selectivity instance*. Each selectivity instance has a weight value. The weight value represents the percentage of executions of the statement in which you expect that the predicates to have the selectivities of that instance.

Some typical defaults:
• COL=constant 0.04
• COL<>constant 0.96
• COL op constant 0.33
(where op means >, <,<=etc.)

- COL LIKE constant 0.10



To use this one … need to understand what Db2 does for us

- Run an explain on the object you are interested in changing FF
- Suggest create a new
  - PLAN_TABLE, DSN_PREDICAT_TABLE, DSN_PREDICATE_SELECTIVITY

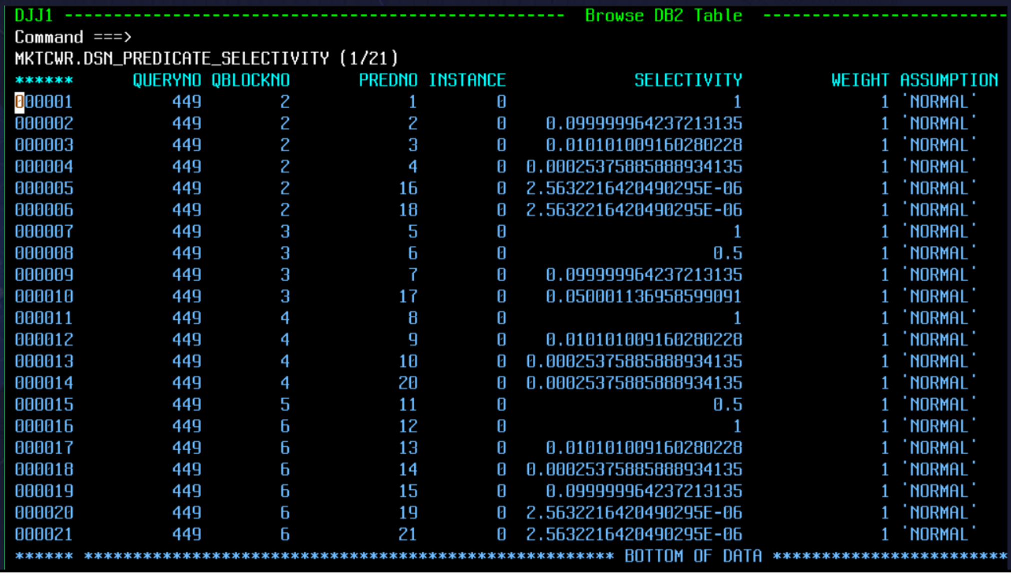
```
XD01     474     66.357593 DECLARE INDORD CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , B . ORDER_NBR , B . ORDER_A
      QB LEFT-HAND          TYPE     RIGHT-HAND STG1 MISMATCH JOIN KEY       FILTER ORD# STAGE     PD NG AP OR TEXT
       2 CUST_PHONE_NBR     LIKE     VALUE      Y             N    N       0.100000    1 STAGE1       N     U  "A"."CUST_PHONE_N
       2 PRIMARY_KEY_A      EQUAL    PRIMARY_KE Y             Y    Y       0.058824    1 MATCHING     N     U  "A"."PRIMARY_KEY_
       2 PRIMARY_KEY_B      EQUAL    PRIMARY_KE Y             Y    Y       0.004650    2 MATCHING     N     U  "A"."PRIMARY_KEY_
       3                    NOTEXIST CORSUB     N             N    N       0.500000    2 STAGE2       N     U   NOT EXISTS(SELEC
       3 CUST_PHONE_NBR     LIKE     VALUE      Y             N    N       0.100000    1 STAGE1       N     U  "A"."CUST_PHONE_N
       4 PRIMARY_KEY_A      EQUAL    PRIMARY_KE Y             N    Y       0.058824    1 MATCHING     N     U  "B"."PRIMARY_KEY_
       4 PRIMARY_KEY_B      EQUAL    PRIMARY_KE Y             N    Y       0.004650    2 MATCHING     N     U  "B"."PRIMARY_KEY_
       5                    NOTEXIST CORSUB     N             N    N       0.500000    1 STAGE2       N     U   NOT EXISTS(SELEC
       6 PRIMARY_KEY_A      EQUAL    PRIMARY_KE Y             N    Y       0.058824    1 MATCHING     N     U  "A"."PRIMARY_KEY_
       6 PRIMARY_KEY_B      EQUAL    PRIMARY_KE Y             N    Y       0.004650    2 MATCHING     N     U  "A"."PRIMARY_KEY_
       6 CUST_PHONE_NBR     LIKE     VALUE      Y             N    N       0.100000    3 STAGE1       N     U  "A"."CUST_PHONE_N
```

Lets look at predicates of a statement and what the optimizer does with these guys

When we do a rebind, prepare a dynamic statement – the optimizer will externalize what it looked at to come up with an access path.

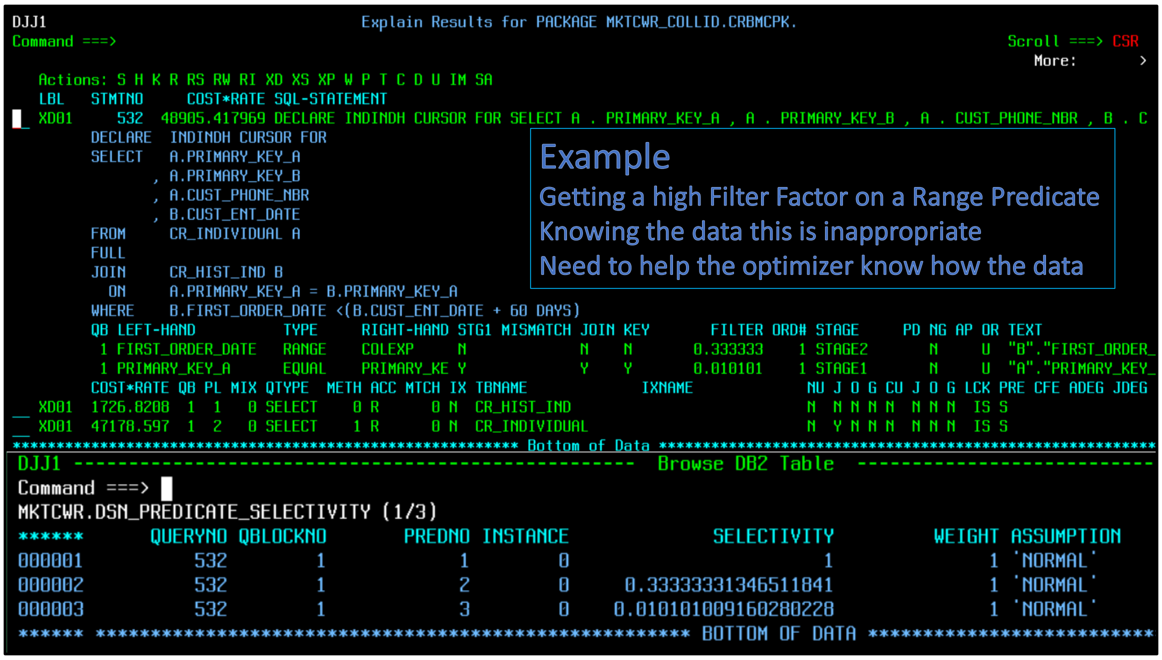
So lets start with the predicates that a statement uses. All that filter factor or selectivity factor. And a bunch of other stuff which is good to know but I am not going to cover in this presentation. Here I have 11 predicates that you can see if you look at the query.

After Explain DSN_PREDICATE_SELECTIVITY



```
DJJ1 --------------------------------------------------  Browse DB2 Table  ------------------------
Command ===>
MKTCWR.DSN_PREDICATE_SELECTIVITY (1/21)
******       QUERYNO QBLOCKNO       PREDNO INSTANCE              SELECTIVITY         WEIGHT ASSUMPTION
000001           449        2            1        0                        1              1 'NORMAL'
000002           449        2            2        0    0.099999964237213135               1 'NORMAL'
000003           449        2            3        0    0.010101009160280228               1 'NORMAL'
000004           449        2            4        0  0.00025375885888934135              1 'NORMAL'
000005           449        2           16        0  2.5632216420490295E-06              1 'NORMAL'
000006           449        2           18        0  2.5632216420490295E-06              1 'NORMAL'
000007           449        3            5        0                        1              1 'NORMAL'
000008           449        3            6        0                      0.5              1 'NORMAL'
000009           449        3            7        0    0.099999964237213135               1 'NORMAL'
000010           449        3           17        0    0.050001136958599091               1 'NORMAL'
000011           449        4            8        0                        1              1 'NORMAL'
000012           449        4            9        0    0.010101009160280228               1 'NORMAL'
000013           449        4           10        0  0.00025375885888934135              1 'NORMAL'
000014           449        4           20        0  0.00025375885888934135              1 'NORMAL'
000015           449        5           11        0                      0.5              1 'NORMAL'
000016           449        6           12        0                        1              1 'NORMAL'
000017           449        6           13        0    0.010101009160280228               1 'NORMAL'
000018           449        6           14        0  0.00025375885888934135              1 'NORMAL'
000019           449        6           15        0    0.099999964237213135               1 'NORMAL'
000020           449        6           19        0  2.5632216420490295E-06              1 'NORMAL'
000021           449        6           21        0  2.5632216420490295E-06              1 'NORMAL'
****** ***************************************************** BOTTOM OF DATA ************************
```

If I look at that query in the DSN Predicate Selectivity table (updated by Explain). You might not have this table allocated in your system, but behind the scenes the optimizer is looking at this … you will see actually 21 different predicate selections. Why because sometime the optimizer makes some decisions and add more predicates that make sense and help determine the access path.



```
DJJ1                                        Explain Results for PACKAGE MKTCWR_COLLID.CRBMCPK.
Command ===>                                                                                      Scroll ===> CSR
                                                                                                        More:     >
   Actions: S H K R RS RW RI XD XS XP W P T C D U IM SA
   LBL   STMTNO     COST*RATE SQL-STATEMENT
█_ XD01     532  48905.417969 DECLARE INDINDH CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , A . CUST_PHONE_NBR , B . C
         DECLARE  INDINDH CURSOR FOR
         SELECT   A.PRIMARY_KEY_A
                , A.PRIMARY_KEY_B
                , A.CUST_PHONE_NBR
                , B.CUST_ENT_DATE
         FROM     CR_INDIVIDUAL A
         FULL
         JOIN     CR_HIST_IND B
           ON     A.PRIMARY_KEY_A = B.PRIMARY_KEY_A
         WHERE    B.FIRST_ORDER_DATE <(B.CUST_ENT_DATE + 60 DAYS)
         QB LEFT-HAND          TYPE     RIGHT-HAND STG1 MISMATCH JOIN KEY       FILTER ORD# STAGE     PD NG AP OR TEXT
          1 FIRST_ORDER_DATE   RANGE    COLEXP     N             N    N       0.333333    1 STAGE2        N     U  "B"."FIRST_ORDER_
          1 PRIMARY_KEY_A      EQUAL    PRIMARY_KE Y             Y    Y       0.010101    1 STAGE1        N     U  "A"."PRIMARY_KEY_
         COST*RATE QB PL MIX QTYPE  METH ACC MTCH IX TBNAME             IXNAME             NU J O G CU J O G LCK PRE CFE ADEG JDEG
__ XD01  1726.8208  1  1   0 SELECT    0 R       0 N  CR_HIST_IND                          N  N N N N  N N N  IS S
__ XD01  47178.597  1  2   0 SELECT    1 R       0 N  CR_INDIVIDUAL                        N  Y N N N  N N N  IS S
********************************************************* Bottom of Data ****************************************************
DJJ1 -------------------------------------------------  Browse DB2 Table  ---------------------------
Command ===> █
MKTCWR.DSN_PREDICATE_SELECTIVITY (1/3)
******      QUERYNO QBLOCKNO        PREDNO INSTANCE              SELECTIVITY          WEIGHT ASSUMPTION
000001          532        1             1        0                        1               1 'NORMAL'
000002          532        1             2        0       0.33333331346511841              1 'NORMAL'
000003          532        1             3        0     0.010101009160280228               1 'NORMAL'
****** **************************************************** BOTTOM OF DATA **************************
```

Example

Getting a high Filter Factor on a Range Predicate

Knowing the data this is inappropriate

Need to help the optimizer know how the data

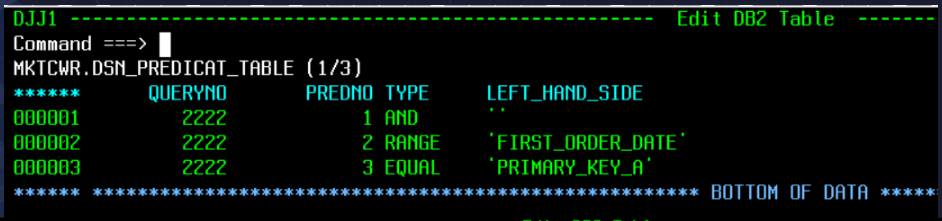
Taking an easier statement – with two predicates.

Showing two predicates, turn into three predicate selections

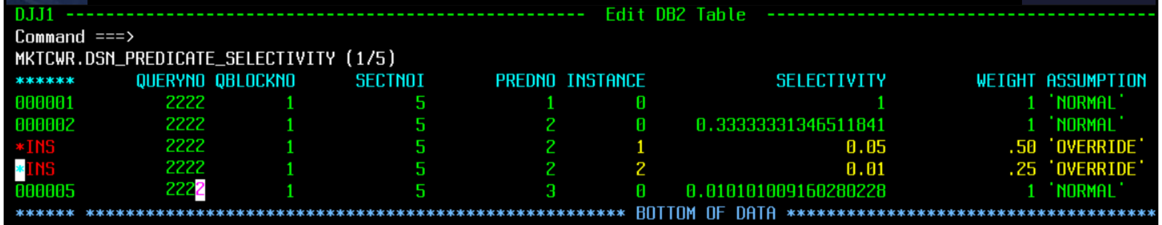
For predicate two, the optimizer has assigned a filter factor of 33% meaning 33% of all rows in that predicate will selected. 33% is a default filter factor because I have a range predicate and a column expression.

I might know better and could get a different path is I was to change the filter factor.

Db2 Explain will populate (if exists):
DSN_PREDICAT_TABLE
DSN_PREDICATE_SELECTIVITY

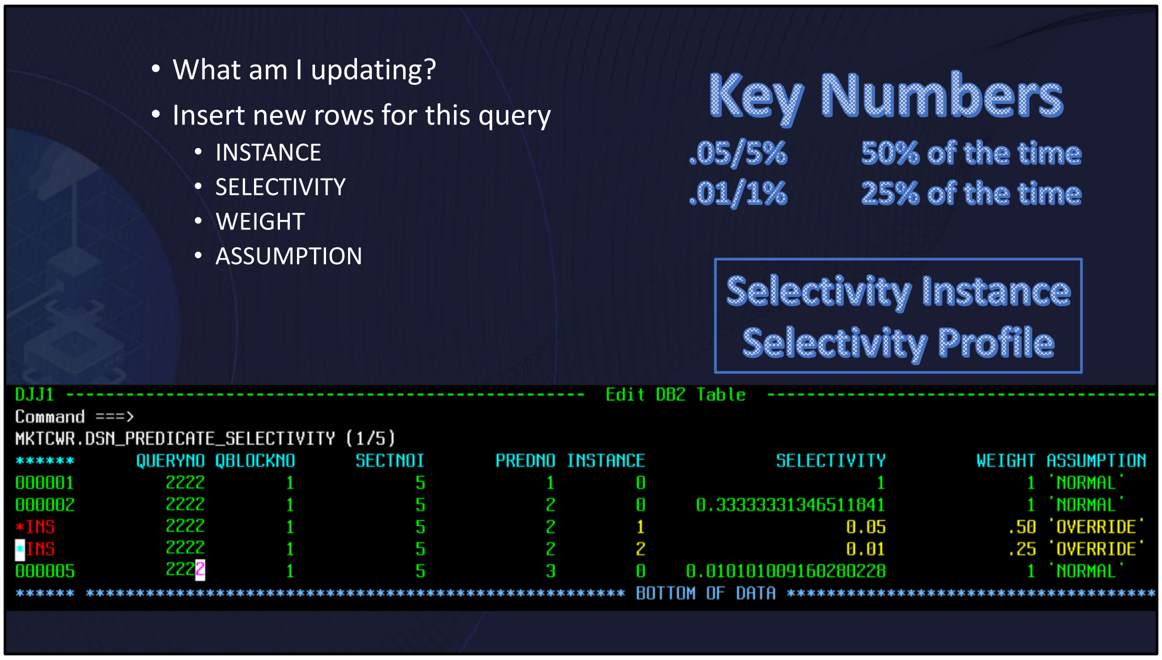


```
DJJ1 --------------------------------------------------- Edit DB2 Table -------
Command ===>
MKTCWR.DSN_PREDICAT_TABLE (1/3)
******      QUERYNO       PREDNO TYPE     LEFT_HAND_SIDE
000001         2222            1 AND      ''
000002         2222            2 RANGE    'FIRST_ORDER_DATE'
000003         2222            3 EQUAL    'PRIMARY_KEY_A'
****** ***************************************************** BOTTOM OF DATA *****
```



```
DJJ1 --------------------------------------------- Edit DB2 Table -----------------------------------
Command ===>
MKTCWR.DSN_PREDICATE_SELECTIVITY (1/5)
******      QUERYNO QBLOCKNO      SECTNOI         PREDNO INSTANCE           SELECTIVITY       WEIGHT ASSUMPTION
000001         2222        1            5              1        0                     1            1 'NORMAL'
000002         2222        1            5              2        0    0.3333333134651184 1          1 'NORMAL'
*INS           2222        1            5              2        1                  0.05          .50 'OVERRIDE'
*INS           2222        1            5              2        2                  0.01          .25 'OVERRIDE'
000005         2222        1            5              3        0   0.010101009160280228           1 'NORMAL'
****** ************************************************ BOTTOM OF DATA ************************************
```

Using that data make required changes



- What am I updating?
- Insert new rows for this query
  - INSTANCE
  - SELECTIVITY
  - WEIGHT
  - ASSUMPTION

Key Numbers

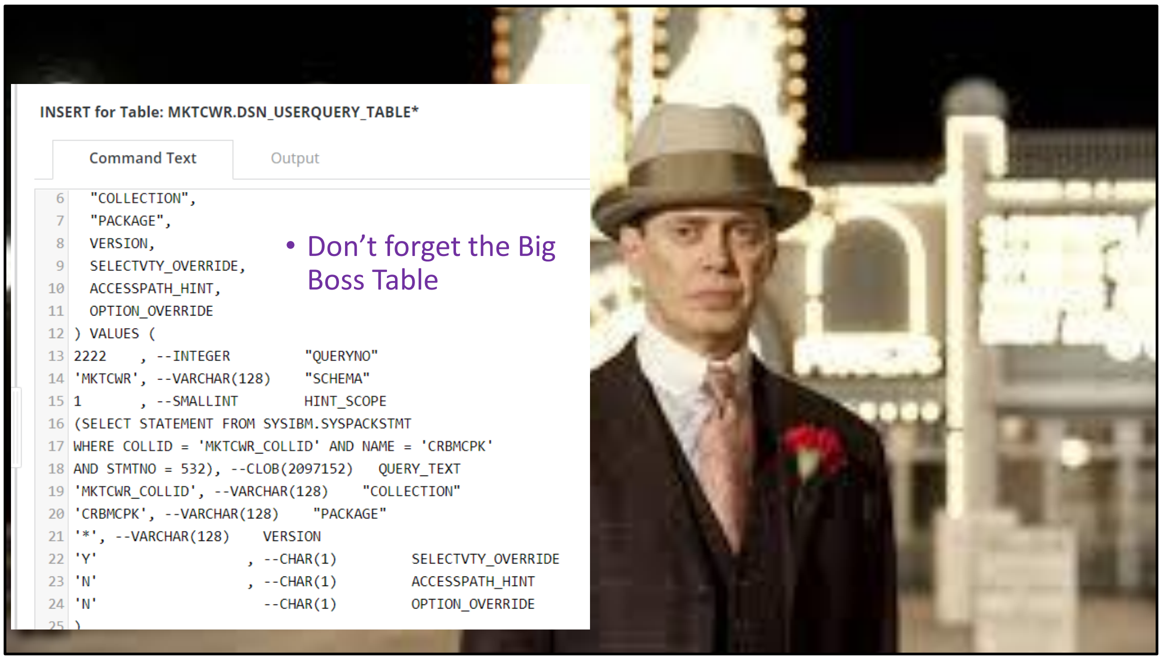
.05/5%	50% of the time
.01/1%	25% of the time

Selectivity Instance
Selectivity Profile

```
DJJ1 ---------------------------------------------------- Edit DB2 Table ---------------------------------------
Command ===>
MKTCWR.DSN_PREDICATE_SELECTIVITY (1/5)
******      QUERYNO QBLOCKNO       SECTNOI        PREDNO  INSTANCE            SELECTIVITY           WEIGHT  ASSUMPTION
000001         2222        1             5             1         0                      1                1  'NORMAL'
000002         2222        1             5             2         0    0.33333331346511841                1  'NORMAL'
*INS           2222        1             5             2         1                   0.05              .50  'OVERRIDE'
*INS           2222        1             5             2         2                   0.01              .25  'OVERRIDE'
000005         2222        1             5             3         0   0.010101009160280228                1  'NORMAL'
****** ****************************************************** BOTTOM OF DATA *************************************
```

Lets think why are we doing this in the first place. The filter factor the optimizer is using is not the best value because the data tells a different story.

It could be the data tells multiple stories. What these inserts tell you. This predicate has three different scenarios. One scenario is 5% of the data qualifies for that predicate 50% of the time. 1% of data qualifies 25% of the time. The rest of the time use the defaults.

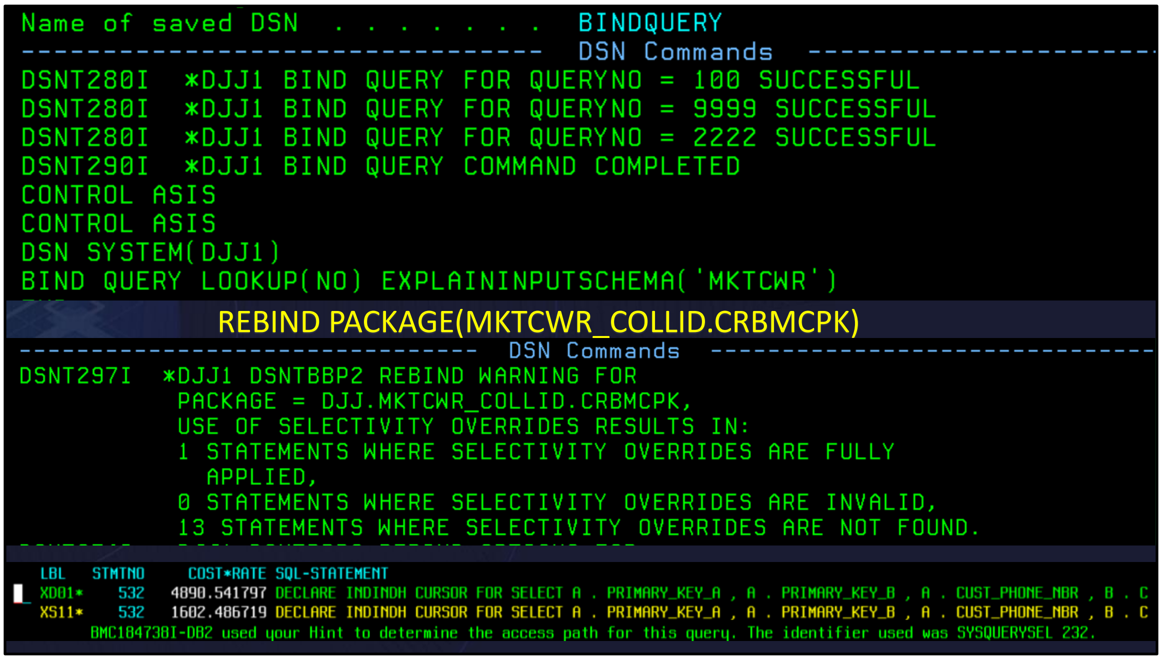


INSERT for Table: MKTCWR.DSN_USERQUERY_TABLE*

```
  "COLLECTION",
  "PACKAGE",
  VERSION,
  SELECTVTY_OVERRIDE,
  ACCESSPATH_HINT,
  OPTION_OVERRIDE
) VALUES (
2222     , --INTEGER          "QUERYNO"
'MKTCWR', --VARCHAR(128)      "SCHEMA"
1        , --SMALLINT         HINT_SCOPE
(SELECT STATEMENT FROM SYSIBM.SYSPACKSTMT
WHERE COLLID = 'MKTCWR_COLLID' AND NAME = 'CRBMCPK'
AND STMTNO = 532), --CLOB(2097152)   QUERY_TEXT
'MKTCWR_COLLID', --VARCHAR(128)     "COLLECTION"
'CRBMCPK', --VARCHAR(128)     "PACKAGE"
'*', --VARCHAR(128)     VERSION
'Y'                     , --CHAR(1)        SELECTVTY_OVERRIDE
'N'                     , --CHAR(1)        ACCESSPATH_HINT
'N'                       --CHAR(1)        OPTION_OVERRIDE
)
```

- Don’t forget the Big Boss Table

First needed to override the predicate selectivity table … but the big boss table is always involved. What statement are you changing?
Global?
Select the Selectivity Override



```
Name of saved DSN  . . . . . . . .  BINDQUERY
--------------------------------  DSN Commands  ----------------------
DSNT280I  *DJJ1 BIND QUERY FOR QUERYNO = 100 SUCCESSFUL
DSNT280I  *DJJ1 BIND QUERY FOR QUERYNO = 9999 SUCCESSFUL
DSNT280I  *DJJ1 BIND QUERY FOR QUERYNO = 2222 SUCCESSFUL
DSNT290I  *DJJ1 BIND QUERY COMMAND COMPLETED
CONTROL ASIS
CONTROL ASIS
DSN SYSTEM(DJJ1)
BIND QUERY LOOKUP(NO) EXPLAININPUTSCHEMA('MKTCWR')
```

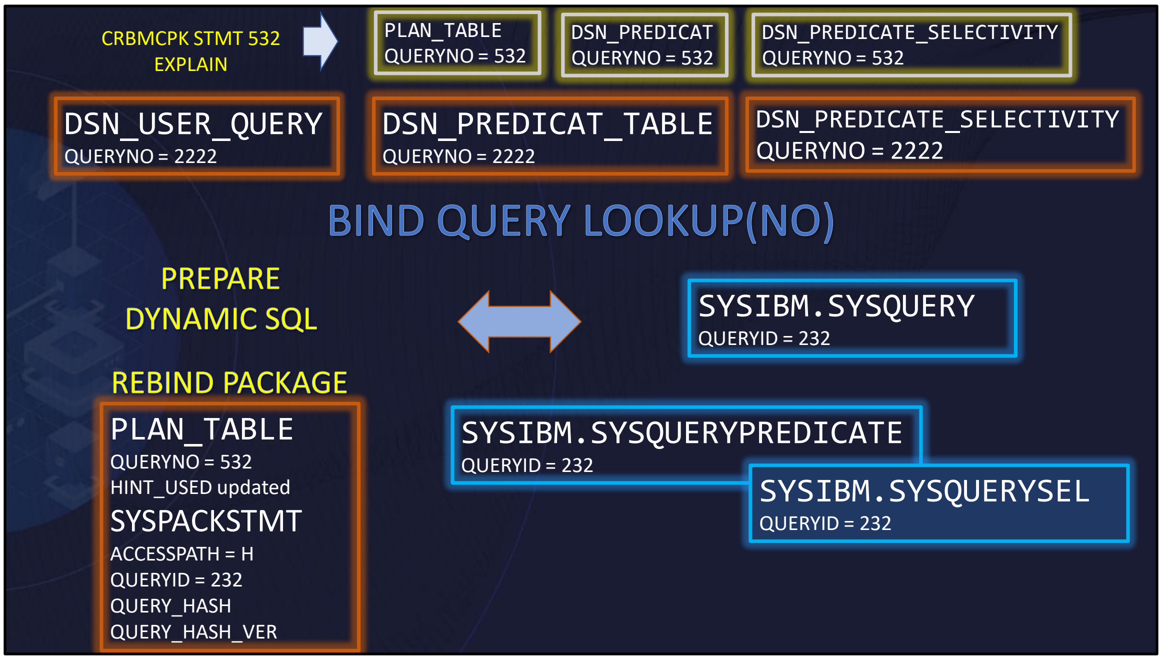
REBIND PACKAGE(MKTCWR_COLLID.CRBMCPK)

```
--------------------------------  DSN Commands  ------------------------------
DSNT297I  *DJJ1 DSNTBBP2 REBIND WARNING FOR
           PACKAGE = DJJ.MKTCWR_COLLID.CRBMCPK,
           USE OF SELECTIVITY OVERRIDES RESULTS IN:
           1 STATEMENTS WHERE SELECTIVITY OVERRIDES ARE FULLY
             APPLIED,
           0 STATEMENTS WHERE SELECTIVITY OVERRIDES ARE INVALID,
           13 STATEMENTS WHERE SELECTIVITY OVERRIDES ARE NOT FOUND.
```

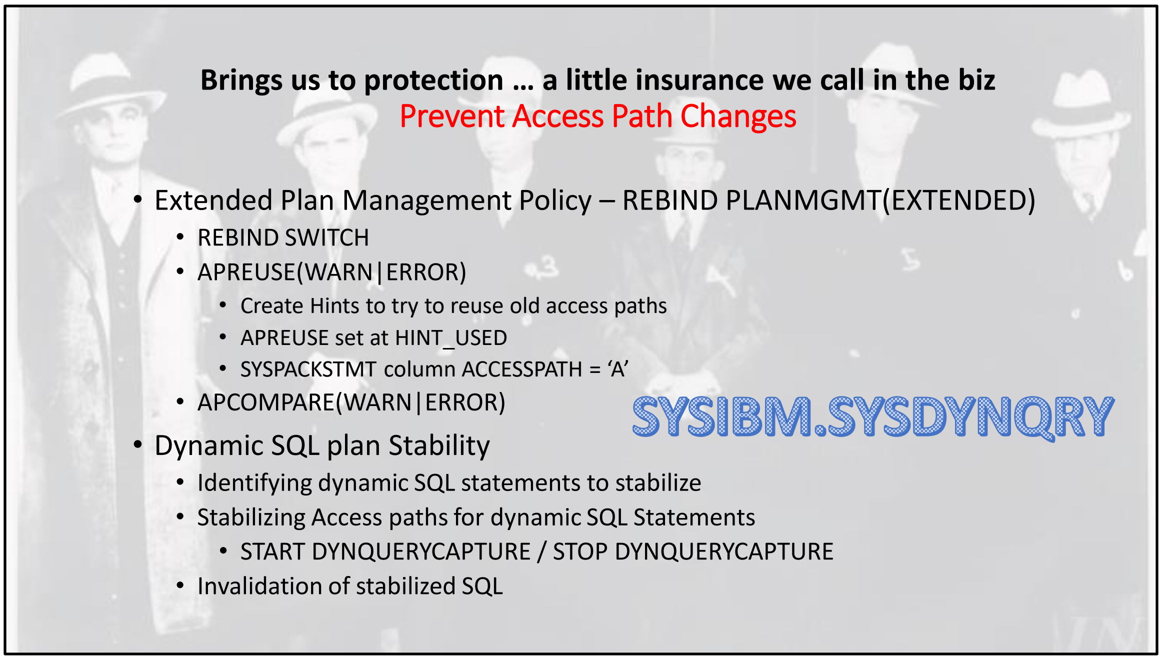
```
  LBL    STMTNO     COST*RATE SQL-STATEMENT
  XD01*     532   4890.541797 DECLARE INDINDH CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , A . CUST_PHONE_NBR , B . C
  XS11*     532   1602.486719 DECLARE INDINDH CURSOR FOR SELECT A . PRIMARY_KEY_A , A . PRIMARY_KEY_B , A . CUST_PHONE_NBR , B . C
        BMC184738I-DB2 used your Hint to determine the access path for this query. The identifier used was SYSQUERYSEL 232.
```

BIND Query, REBIND Package, Proof Points

Notice BIND Query will process all the rows in the Big Boss Table. So could have duplicates and get new query numbers.


CRBMCPK STMT 532
EXPLAIN
PLAN_TABLE
QUERYNO = 532
DSN_PREDICAT
QUERYNO = 532
DSN_PREDICATE_SELECTIVITY
QUERYNO = 532
DSN_USER_QUERY
QUERYNO = 2222
DSN_PREDICAT_TABLE
QUERYNO = 2222
DSN_PREDICATE_SELECTIVITY
QUERYNO = 2222
BIND QUERY LOOKUP(NO)
PREPARE
DYNAMIC SQL
REBIND PACKAGE
SYSIBM.SYSQUERY
QUERYID = 232
PLAN_TABLE
QUERYNO = 532
HINT_USED updated
SYSPACKSTMT
ACCESSPATH = H
QUERYID = 232
QUERY_HASH
QUERY_HASH_VER
SYSIBM.SYSQUERYPREDICATE
QUERYID = 232
SYSIBM.SYSQUERYSEL
QUERYID = 232

Review …



Brings us to protection … a little insurance we call in the biz

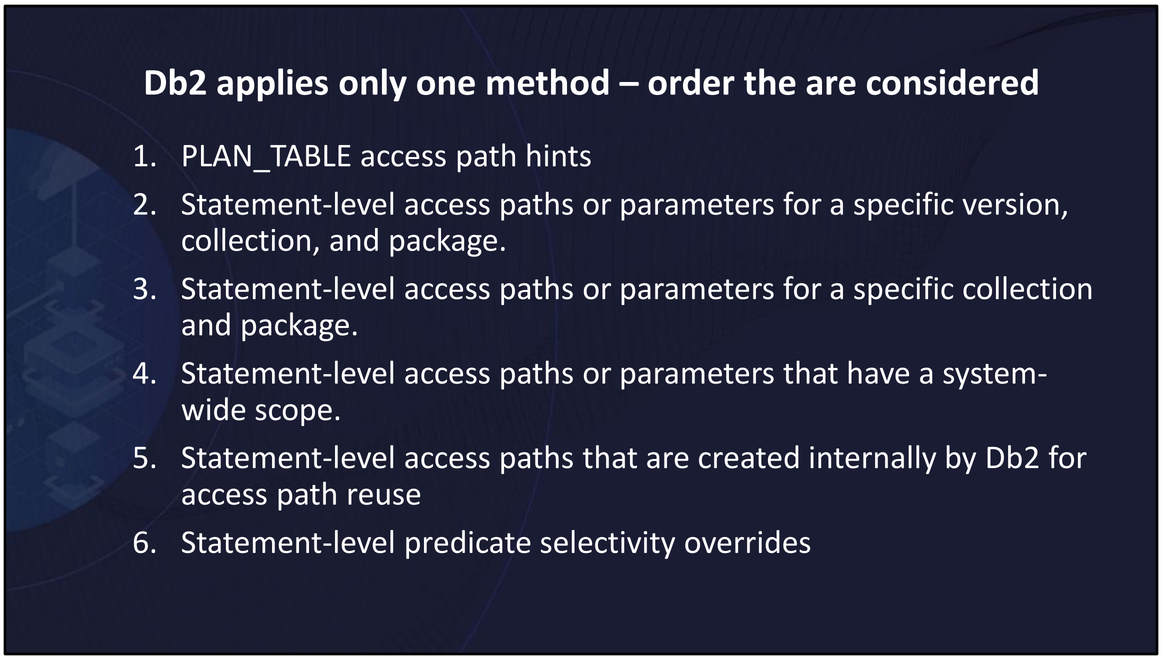
Prevent Access Path Changes

- Extended Plan Management Policy – REBIND PLANMGMT(EXTENDED)
  - REBIND SWITCH
  - APREUSE(WARN|ERROR)
    - Create Hints to try to reuse old access paths
    - APREUSE set at HINT_USED
    - SYSPACKSTMT column ACCESSPATH = ‘A’
  - APCOMPARE(WARN|ERROR)
- Dynamic SQL plan Stability
  - Identifying dynamic SQL statements to stabilize
  - Stabilizing Access paths for dynamic SQL Statements
    - START DYNQUERYCAPTURE / STOP DYNQUERYCAPTURE
  - Invalidation of stabilized SQL

SYSIBM.SYSDYNQRY

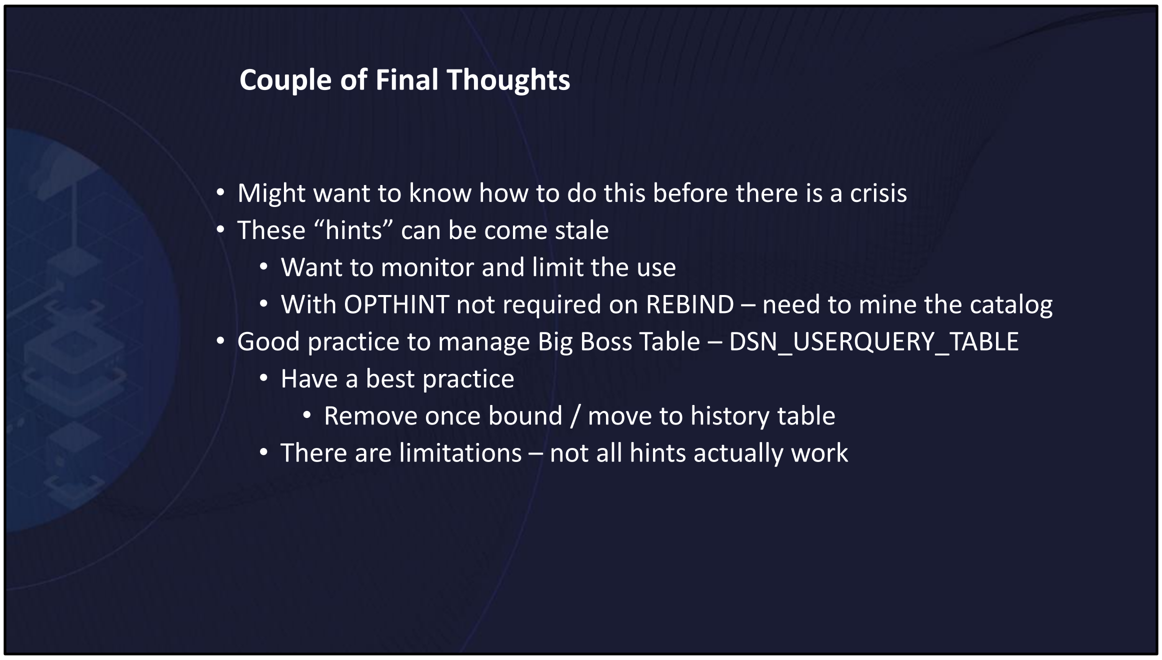
When you enable *dynamic SQL plan stability*, Db2 stores statement cache structures for specified dynamic SQL statements in the Db2 catalog. Whenever a *stabilized dynamic SQL statement* is not present in the dynamic statement cache when issued, Db2 can load the statement cache structures from the Db2 catalog and avoid the full prepare operation. The goal is to achieve access path stability comparable to static SQL statements for repeating cached dynamic SQL statements.

However, stabilizing dynamic SQL statements involves tradeoffs. Access path changes often improve performance, so you trade away those potential performance improvements for stability. The stabilized dynamic SQL statements also use storage space in the Db2 catalog to store the run time structures.



Db2 applies only one method – order the are considered

1. PLAN_TABLE access path hints
2. Statement-level access paths or parameters for a specific version, collection, and package.
3. Statement-level access paths or parameters for a specific collection and package.
4. Statement-level access paths or parameters that have a system-wide scope.
5. Statement-level access paths that are created internally by Db2 for access path reuse
6. Statement-level predicate selectivity overrides



Couple of Final Thoughts

- Might want to know how to do this before there is a crisis
- These “hints” can be come stale
  - Want to monitor and limit the use
  - With OPTHINT not required on REBIND – need to mine the catalog
- Good practice to manage Big Boss Table – DSN_USERQUERY_TABLE
  - Have a best practice
    - Remove once bound / move to history table
  - There are limitations – not all hints actually work

When you enable *dynamic SQL plan stability*, Db2 stores statement cache structures for specified dynamic SQL statements in the Db2 catalog. Whenever a *stabilized dynamic SQL statement* is not present in the dynamic statement cache when issued, Db2 can load the statement cache structures from the Db2 catalog and avoid the full prepare operation. The goal is to achieve access path stability comparable to static SQL statements for repeating cached dynamic SQL statements.

However, stabilizing dynamic SQL statements involves tradeoffs. Access path changes often improve performance, so you trade away those potential performance improvements for stability. The stabilized dynamic SQL statements also use storage space in the Db2 catalog to store the run time structures.

Speaker: Chad Reiber

Company: BMC Software

Email Address: Chad_Reiber@bmc.com

Don't forget to fill out a session evaluation!

In 1983 **Chad Reiber** started in IT with AT&T as a developer of data generator software. From there he moved into the database arena as a database administrator for IMS, IDMS, and Db2. Chad has been working with Db2 since 1987. Chad began working for BMC Software in 1997 as a Software Consultant and primarily supports all aspects of BMC's Mainframe products for the Northeast Db2 community.
My email address is chad_reiber@bmc.com and my twitter account is @creiber11