



Mastering Access Path Management in Db2 for z/OS: Simplify, Optimize, Succeed

Denis Tronin

Product Management | Broadcom



Agenda

- Query Access Path
- EXPLAIN
- Plan Management Policies
- Access Path Optimization Hints
- Summary and Q&A



Access Path Overview

- Defines a **method database uses to retrieve data** for a SQL query
- Access path is determined by **Db2 Optimizer** based on factors like table statistics, available indexes, join methods, etc.
- Common access paths: *tablespace scans, index scans, index-only access, nested loop joins, etc.*
- Access path is set at:
 - During **PREPARE** for **dynamic** statements
 - During package **BIND/REBIND** for **static** statements



Access Path Influencers

- **Query Text**
 - *predicates, subqueries, joins, host vars, functions, etc.*
- **Available Indexes**
- **Object Statistics**
- **System setup**
 - *zParms, MIPS capacity, number of CPUs, BP sizes, etc.*
- **Optimizer hints and access path reuse**

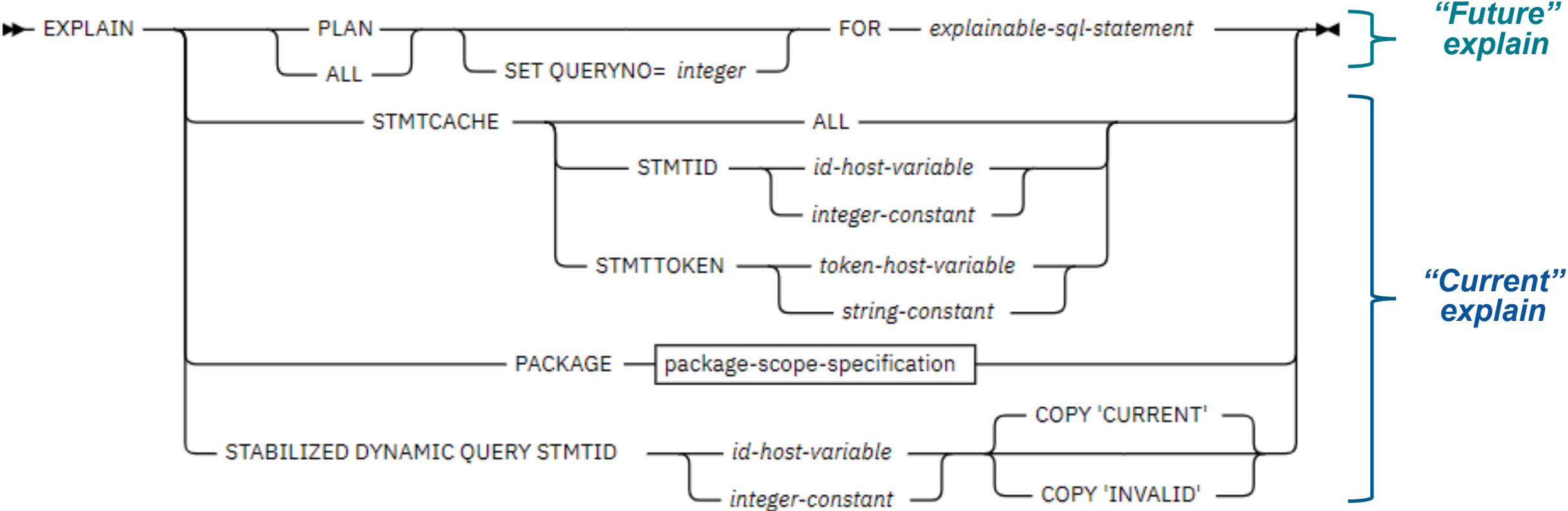


Getting the Access Path

- Query access path can be **externalized** for review and analysis using an EXPLAIN
 - Works only for DML statements
- EXPLAIN inserts rows to Db2-supplied user tables (“**explain tables**”)
 - Each row describes specifics query execution step
- Db2 populates explain tables:
 - when an EXPLAIN statement is executed
 - when BIND/REBIND a package with EXPLAIN(YES/ONLY) option
 - when running dynamic query and CURRENT EXPLAIN MODE = YES/EXPLAIN



EXPLAIN Statement



EXPLAIN tables

- As of Db2 13, there are 20 explain tables
 - These tables are user-specific, schema should match SQLID of the explainer

PLAN_TABLE	Major table – information about the access path methods Note: The only table populated for EXPLAIN PACKAGE
DSN_STATEMNT_TABLE	Estimated cost details (per statement)
DSN_FUNCTION_TABLE	Descriptions of used functions
DSN_STATEMENT_CACHE_TABLE	Dynamic SQL text with execution statistics (IFCID 316) Note: Populated with EXPLAIN STMTCACHE
DSN_PREDICAT_TABLE	Information about query predicates
DSN_STAT_FEEDBACK	Recommendations for capturing missing or conflicting statistics



Create EXPLAIN tables

- Sample DDL statements for creating explain tables can be found in *db2Lib.SDSNSAMP(DSNTESC)*
- Administrative stored procedure **SYSPROC.ADMIN_EXPLAIN_MAINT** can be used to create, drop or upgrade explain tables

Name	Type	Value	
MODE	VARCHAR(8)	RUN	+++
ACTION	VARCHAR(30)	STANDARDIZE_AND_CREATE	+++
MANAGE_ALIAS	VARCHAR(3)	*NULL*	+++
TABLE_SET	VARCHAR(1000)	DIAGNOSTICS	+++
AUTHID	VARCHAR(128)	*NULL*	+++
SCHEMA	VARCHAR(128)	*NULL*	+++
SCHEMA_ALIAS	VARCHAR(128)	*NULL*	+++
DATABASE	VARCHAR(8)	*NULL*	+++

Buttons: Set to Null, Load Values, Save Values, Reset Values



Access Path in PLAN_TABLE

- **Credentials:** *statement number, timestamp, plan, package ...*
- **Flow:** *query block, parent block, query block steps ...*
- **Access path:**
 - *access method (TB scan, IX scan..), matching cols...*
 - *join method (nested loop, merge scan..), join type (full, inner..)*
 - *sort operations for new and composite tables (unique, join, order by ..)*
- **Object:** *table name, table type (subquery, workfile..), index name ...*



Statement Cost in DSN_STATEMNT_TABLE

- **Credentials** (to correlate with PLAN_TABLE rows):
 - *Statement number, timestamp, plan, package..*
 - *Client fields, connection type, hashes ..*
- **Query:** *type (SELECT, INSERT..), encoding*
- **Cost estimation:**
 - *Cost **category** (A: enough stats vs. B: using defaults) and **reason** in case of Category B and using defaults (table cardinality, host variables, having clause..)*
 - ***Cost** (milliseconds, service units), overall estimated cost*
- **Access path status:** *Compare, Reuse*



Analyzing Access Path

Your SQL

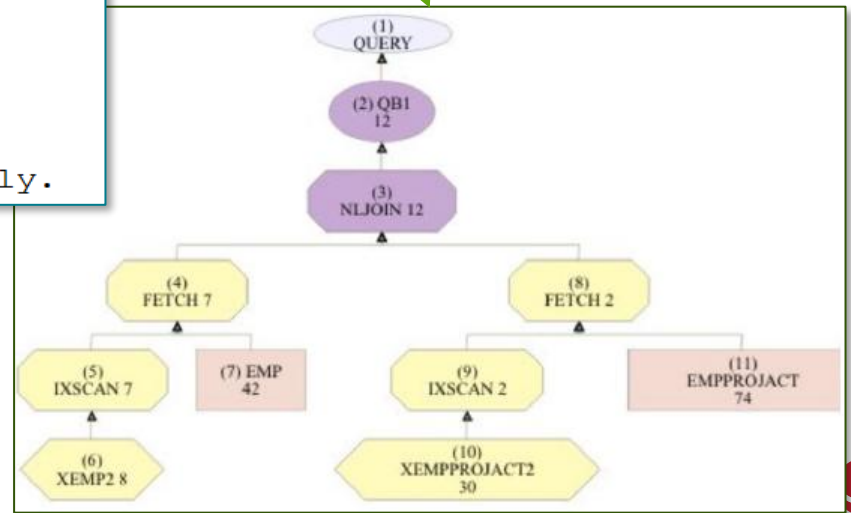
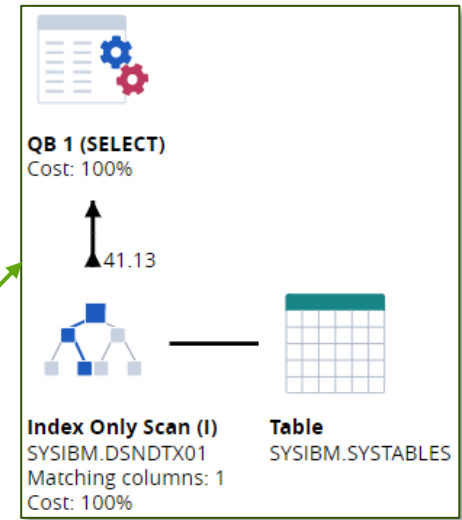
ACCESSCREATOR	ACCESSNAME	INDEXONLY	SORTN_UNIQ	SORTN_JOIN
SYSIBM	DSNDTX03	Y	N	N

Textual by Tooling

```

Line 01      Data accessed from the index, no table access needed
SQL operation: SELECT statement
Cost estimate: Milliseconds=1          Service units=9
              : Cost made without using default values.
In.....Table: SYSIBM.SYSTABLES
              Index: SYSIBM.DSNDTX01
Access Method: Matching index only scan using 1 column. Forward scan.
              Lock: Intent share
Accelerator:  Query is not eligible, reason 4. The query is not read only.
    
```

Visual by Tooling



Object Statistics

- **Up-to date** statistics helps Db2 to choose an efficient access path
 - But excessive and complex statistics (e.g. histograms) slows down Db2 optimizer – affects BIND and PREPARE
- Statistics is collected by running **RUNSTATS** utility or preferably inline as part of LOAD and REORG executions
 - Automate the stats collection (with tools like Database Analyzer, using ADMIN stored procedures, leveraging DSNACCOX, etc.)
 - Don't collect inline statistics as part of REORG INDEX as it may cause discrepancies / conflicting stats with already existing ones at table level
- Most of the object statistics columns in [Db2 catalog tables](#) is updatable
 - Preserve or migrate statistics as part of SQL tuning exercise, but not as way to establish a desired access path



Review Missing or Conflicting Statistics

- Explain populates **DSN_STAT_FEEDBACK** explain table to tell what statistics is better to be collected
 - Object, column(s), TYPE (cardinality, frequency..), REASON (basic, keycard..)
- Catalog table **SYSIBM.SYSSTATFEEDBACK** updated at stats intervals
 - STATFDBK_SCOPE enables stats collection (static/dynamic/all)
 - Only for tables with STATS_FEEDBACK set to 'Y' in SYSIBM.SYSTABLES
 - STATFDBK_PROFILE enables auto-creation of stats profiles
 - STATSINT zParm controls statistics collection interval
 - Successful RUNSTATS runs clean up the SYSSTATFEEDBACK table
 - Use ACCESS DB(*) SP(*) MODE(STATS) to force statistics externalization



Plan Management Policy

- Defines how many package copies (access paths) can exist
 - **EXTENDED** = Current, Previous, Original (+ Phase-out copies)
 - **BASIC** = Current and Previous
 - **OFF** = Current only
- REBIND option **PLANMGMT** sets the policy for the package
 - zParm PLANMGMT sets the default value (EXTENDED)
 - Also controls REBIND phase-in (enabled with EXTENDED)
- REBIND option **APRETAINDUP** to allow identical copies



Switch to an Inactive Package Copy

- Use REBIND PACKAGE with **SWITCH** option (PREVIOUS | ORIGINAL) to revert the access path to the previous (inactive) copies
 - **PREVIOUS** switches current and previous
 - **ORIGINAL** switches current to previous, and original to current
- SWITCH is **exclusive** and cannot be specified with any other REBIND options
- Use FREE PACKAGE with **PLANMGMTSCOPE** option to remove inactive package copies – only if this is what you want to do!
 - For example, copies that are more than -2 releases back, non-executable



Control Access Path Changes (1 – 4)

- Two strategies to control package changes at BIND/REBIND
- **APCOMPARE** – compare the access paths
 - **WARN** – proceed in case of a found mismatch
 - **ERROR** – terminate for an access path mismatch
- **APREUSE** – reuse the access paths
 - **WARN** – proceed in case of reuse failures
 - **ERROR** – terminate for reuse failures
- DSNT285I / DSNT286I – # of statements with AP matched / reused



Control Access Path Changes (2 – 4)

```
DSNT286I  ! ██████████ DSNTBBP2 REBIND FOR PACKAGE = ██████████ ,  
          USE OF APREUSE RESULTS IN:  
          1 STATEMENTS WHERE APREUSE IS SUCCESSFUL  
          0 STATEMENTS WHERE APREUSE IS EITHER NOT SUCCESSFUL  
            OR PARTIALLY SUCCESSFUL  
          0 STATEMENTS WHERE APREUSE COULD NOT BE PERFORMED  
          0 STATEMENTS WHERE APREUSE WAS SUPPRESSED BY OTHER HINTS.
```

```
DSNT285I  ! ██████████ DSNTBBP2 REBIND FOR PACKAGE = ██████████ ,  
          USE OF APCOMPARE RESULTS IN:  
          1 STATEMENTS WHERE COMPARISON IS SUCCESSFUL  
          0 STATEMENTS WHERE COMPARISON IS NOT SUCCESSFUL  
          0 STATEMENTS WHERE COMPARISON COULD NOT BE PERFORMED.
```



Control Access Path Changes (3 – 4)

- APCOMPARE(WARN)
 - Get new access path but report on changes
- APREUSE(WARN) + APCOMPARE(WARN)
 - Get old access where possible but report on changes
 - While a reuse can fail, a new access path might still match
- APREUSE(ERROR) + APCOMPARE(ERROR)
 - Prevent any access path changes

Use
REBIND
SWITCH to
revert the
changes



Control Access Path Changes (4 – 4)

- Combine APCOMPARE and APREUSE options with **EXPLAIN (ONLY)** or (YES) to analyze access path changes
- Review explain tables:
 - Successful reuse sets **PLAN_TABLE.HINT_USED = 'APREUSE'**
 - Failure details are written to **PLAN_TABLE.REMARKS** column
 - A reason code from **SQLCODE +395** for reuse failures
 - Mismatched **PLAN_TABLE** information for which comparison failed
 - Reuse & compare status details in **DSN_STATEMNT_TABLE**

```
REMARKS: APCOMPARE FAILURE (COLUMN: ACESSTYPE)
```

```
APREUSE FAILURE (REASON: 50) APCOMPARE FAILURE (COLUMN: ACCESS_DEGREE)
```



Dynamic SQL Statement Stabilization (1 – 4)

- Access path of a dynamic SQL statement can be **captured** into Db2 catalog and then **used for PREPARE**
 - Enabling access path stabilization and performance gains
- **SYSIBM.SYSDYNQRY** contains access path details for a statement
 - **Statement text, access path**, statement ID, stabilization group name
 - Validity flag, schema, APPLCOMPAT, current SQLID
 - Timestamp when stabilized, date when last used
- No PLANMGMT policy – only a **current copy** of the access path
 - One invalid copy is created when the same statement is re-stabilized
 - However, invalid copy can be EXPLAINed, useful for regression analysis



Dynamic SQL Statement Stabilization (2 – 4)

- **–START DYNQUERYCAPTURE** stabilizes the access path for selected dynamic statements
 - **STMTID / STMTTKN** to stabilize a **specific statement**
 - **THRESHOLD** to stabilize **multiple statements** with a minimum number of executions
 - **MONITOR** enables continuous collection
 - Use **–DISPLAY DYNQUERYCAPTURE** to identify and **–STOP DYNQUERYCAPTURE** to stop active monitors
 - **CURSQLID** limits with a specific SQLID
 - **SCOPE** enables capturing on all members of a data sharing group
- Ideal state – a “good” access path is stabilized



Dynamic SQL Statement Stabilization (3 – 4)

- Look at **dynamic statement cache** to identify queries for stabilization
 - EXPLAIN STMTCACHE ALL populates **DSN_STATEMENT_CACHE_TABLE**
 - Use your performance monitor (e.g. SYSVIEW for Db2)

R/DYNSQLST Dynamic SQL Statements in Cache							
SQL Text (16 bytes)	Cur Usrs	Act Cpys	Exec	Accum Elapsed	Accum CPU	Program Name	
SELECT SPACE FRO	0	0	8	0.000	0.000	RMA@PGTS	—
SELECT A.LOCAT	0	0	1	0.000	0.000	PZA#PACK	—

- Use **FREE STABILIZED DYNAMIC QUERY** to remove obsolete stabilized statements from the Db2 catalog



DB2 Command

```
====> -START DYNQUERYCAPTURE STBLGRP(DENIS) STMTID(3682)
====>
```

DSNX221I !■■■■ DSNXESTC DYNAMIC QUERY CAPTURE FOR COMMAND NUMBER 64 STARTED SUCCESSFULLY.

R/DYNSQLTX Selected Dynamic SQL Statement in Cache
 SORT N/A

Users	0	Stmt num	190	Program name	DSNESM68
Active copies	0	Executions	1	Transaction	■■■■■
Statement ID	00000E62	Gen stmt name	036E1A234F215E485A2E235C0		
Bind SQLID	■■■■■				
Table qual	■■■■■				
Owner user ID	■■■■■				
End user ID	■■■■■				
Workstn name	TSO			Caching membr	
Invalid Status	00	Current data	N	Cache lit rep	None
Isolation	RR	Current rules	D	Dynamic rules	R Current degree 1
Schema	■■■■■			Cur precision	N Csr with hold N
				Expansion rsn	None
				Accl eligible	No
				Query hash ID	365D594F4D5D50495C545C203

Stabilized ID	00000000000002BC1
Group name	DENIS

Time inserted	06/13/22 05:19:50	RID list occurrences	Limit	Storage
Stats started	06/13/22 03:51:46	Failed	0	0
		Paged out	0	0
		Noted	0	0
		Not skipped	0	0

SELECT SDQ_STMT_ID, STBLGRP, STMTTEXT, VALID, CURSCHEMA FROM SYSIBM.SYSDYNQRY WHERE STBLGRP='DENIS'

	SDQ_STMT_ID	STBLGRP	STMTTEXT	VALID	CURSCHEMA
1	11201	DENIS	SELECT COUNT(*) FROM PLAN_TABLE	Y	■■■■■

pages	5	Total	5.0
vs examined	0	Avg/Exec	0.0
vs processd	1		1.0



Optimization Hints

- System parameters
 - Star join processing: STARJOIN, SJTABLES
 - Query parallelism: PARAMDEG, CDSSRDEF
- Package **REOPT** bind/rebind option (NONE, ONCE, ALWAYS, AUTO)
 - Re-optimization at runtime using *host variable* and *spec register* values
- **PLAN_TABLE** Query Access Path Hints
- **Statement-level** Access Path Hints



PLAN_TABLE Access Path Hint (1 – 3)

- Applies to a **query** in the **package** using the access path stored in the **PLAN_TABLE**
 - QUERYNO, APPLNAME, PROGNAME, VERSION, COLLID must match
 - Include a QUERYNO clause into statement text to avoid ambiguity
 - Executing AUTH ID determines which PLAN_TABLE is used
- This is rather older technique, statement-level hints offer better usability and management



PLAN_TABLE Access Path Hint (2 – 3)

- **To create** a PLAN_TABLE hint:
 - Update the access path and set the *hint name* in OPTHINT column
 - BIND/REBIND a package with OPTHINT('hint name') for **static** statements
 - SET CURRENT OPTIMIZATION HINT = 'hint name' for **dynamic** statements
- **To validate:**
 - Explain and check if PLAN_TABLE(HINT_USED) contains the hint-name
 - Check SYSIBM.SYSPACKAGE(OPTHINT), SYSPACKSTMT(ACCESSPATH)='H'



```
SELECT COUNT ( * )
FROM SYSIBM.SYSTABLES QUERYNO 123
```

Access Path Analysis:

```
Cost: (ms) 1                (su) 12                (tc) 0.628

PObkNo: 0      PPlnNo: 0      QblkNo: 1      PlanNo: 1      MxOpSq: 0
Access: IXONLY TSLock: IS    Prefet: S      QblkTy: SEL    TBType: TABLE
```

```
UPDATE PLAN_TABLE SET (INDEXONLY, ACCESSCREATOR, ACCESSNAME, ACESSTYPE, OPTHINT) =
('N', '', '', 'R', 'TSCAN') WHERE QUERYNO=123;
```

```
SET CURRENT OPTIMIZATION HINT = 'TSCAN';
```

```
Cost: (ms) 43                (su) 1582                (tc) 705.770

PObkNo: 0      PPlnNo: 0      QblkNo: 1      PlanNo: 1      MxOpSq: 0
Access: TS SCN TSLock: IS    Prefet: S      QblkTy: SEL    TBType: TABLE
Tb#1 : SYSIBM.SYSTABLES
GrpMbr:        PgeRng:        ColFnE: R      WhnOpt:        PriAcc:
HintUs: TSCAN
Encode: UNICODE SCCSID: 367  MCCSID: 1208  DCCSID: 1200  VI_ACT: N
```



Statement-Level Hint (1 – 5)

- Use a **statement-level hint** to:
 - Select a specific access path
 - Override predicate filter factors
 - Set REOPT bind option and optimization zParms (star joins, parallelism)
- Applies to a statement with the **given SQL text** in scope of:
 - Specific package version
 - Specific package
 - System-wide



Statement-Level Hint (2 – 5)

- **To create a hint:**
 - Populate input **DSN_USERQUERY_TABLE** and relevant **explain tables**:
 - PLAN_TABLE specifies query access path selection details
 - DSN_PREDICAT_TABLE and DSN_PREDICATE_SELECTIVITY specify filter factors overrides for query predicates
 - Issue **BIND QUERY** command
- **To validate:**
 - Check details of the created hint in SYSIBM.SYSQUERY*
 - For packages, check if SYSIBM.SYSPACKSTMT contains ACCESSPATH='H'
 - Run EXPLAIN and check if HINT_USED column gets a hint query ID



DSN_USERQUERY_TABLE (3 – 5)

- Hint credentials columns
 - **Statement text, query ID** for SYSIBM.SYSQUERY* catalog tables reference
- Hint scope columns
 - System vs. package, collection/package/version
- Hint parameter columns
 - ACCESSPATH_HINT='Y' / SELECTVTY_OVERRIDE= 'Y' / OPTION_OVERRIDE='Y'
 - **Query number** to identify rows in related explain tables
 - REOPT, STARJOIN, SJTABLES, DEF_CURR_DEGREE, MAX_PAR_DEGREE



EXPLAIN STMTCACHE STMTID 3711;

```

Line 01          Data accessed from the index, no table access needed
SQL operation:  SELECT statement
Cost estimate:  Milliseconds=1          Service units=4
Access Method:  Non-matching index only scan using sequential prefetch. Forward scan.
-----
SELECT COUNT ( * )
FROM ██████████ . PLAN_TABLE

```

INSERT INTO DSN_USERQUERY_TABLE

(QUERYNO, QUERY_TEXT, HINT_SCOPE, OPTION_OVERRIDE, ACCESSPATH_HINT, SELECTVTY_OVERRIDE)

SELECT STMT_ID, STMT_TEXT, 0, 'N', 'Y', 'N' FROM DSN_STATEMENT_CACHE_TABLE WHERE STMT_ID=3711;

UPDATE PLAN_TABLE SET

(INDEXONLY, ACCESSCREATOR, ACCESSNAME, ACESSTYPE) = ('N', '', '', 'R') WHERE QUERYNO=3711;

```

DSN
BIND QUERY LOOKUP(NO)
DSNT280I  ! ██████████ BIND QUERY FOR QUERYNO = 3711 SUCCESSFUL
DSNT290I  ! ██████████ BIND QUERY COMMAND COMPLETED

```

```

Line 01          Data accessed from the table
SQL operation:  SELECT statement
Cost estimate:  Milliseconds=10        Service units=338
Access Method:  Table scan using sequential prefetch.
Hint:           Access path determined using statement-level hint, QUERYID 1

```



Statement-Level Hint (5 – 5)

- Clean up DSN_USERQUERY_TABLE after you are done
 - BIND QUERY takes every row – risk of overriding existing hints
- Create a separate set of EXPLAIN tables for creating hints
 - Use EXPLAININPUTSCHEMA to point to them (BIND QUERY parm)
- Use FREE QUERY command to remove a statement-level hint
- Keep in mind the precedence:
 1. Statement-level access path or parameters hints
 - package version → package → system-wide
 2. Statement-level predicate selectivity overrides



Summary and Q&A

- ✓ Use EXPLAIN to write efficient statements and troubleshoot performance problems
- ✓ Maintain accurate database statistics to get the best results
- ✓ If optimizer is not smart enough, use one of described techniques to influence selection of access path
- ✓ Once you are happy with performance, preserve the access path



Technical Education Direct from the Experts

In-Person

European MTE
Prague, Czech Republic
Apr. 21-23, 2026

Virtual

Virtual MTE
Held Virtually
June 23-25, 2026

In-Person

North American MTE
Plano, TX
Oct. 20-22, 2026

Join a Mainframe Technical Exchange

- Network with peers and Mainframe technical experts
- Participate in technical how-to sessions and hands-on workshops, hear product updates, provide feedback
- No registration fee!



"The event covered all the relevant topics, and the content was amazing. The hands-on experience was also a good way to get actual experience with the tools."

Senthil Mathur, Manager, Accenture



Thank you

Denis.Tronin@broadcom.com