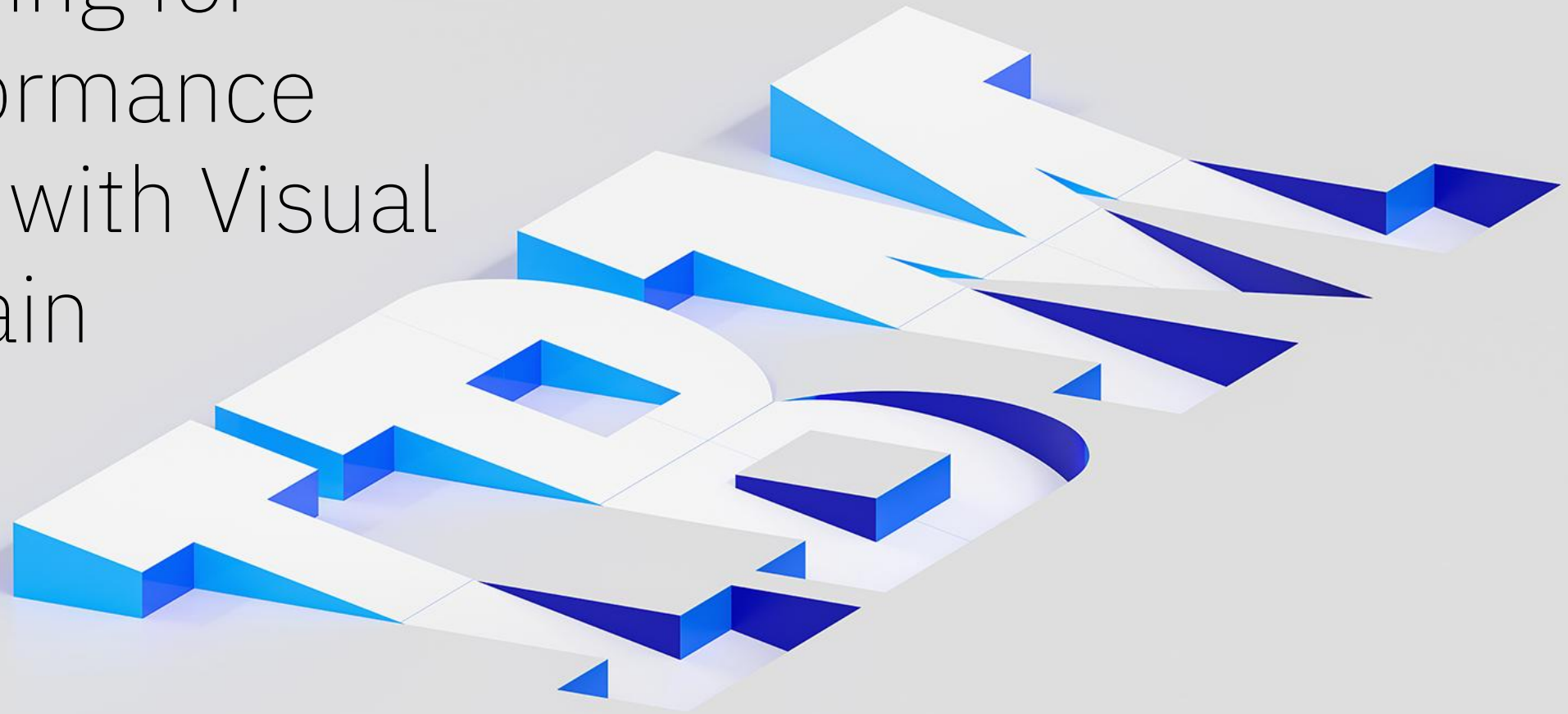
Panning for Performance Gold with Visual Explain



~~Ryan Moeller~~
~~Staff Software Engineer, Db2 for i~~ rmoeller@ibm.com

Morten Buur Rasmussen
Power Performance Specialist, IBM Expert Labs Morten.buur.rasmussen@dk.ibm.com

IBM i

© Copyright IBM Corporation 2026

Ryan Moeller - Software Developer - Db2 for i SQL Query Optimizer

Thanks for borrowing the deck

It is no problem, I had reserved this time slot anyway

© Copyright IBM Corporation 2026

Key Points

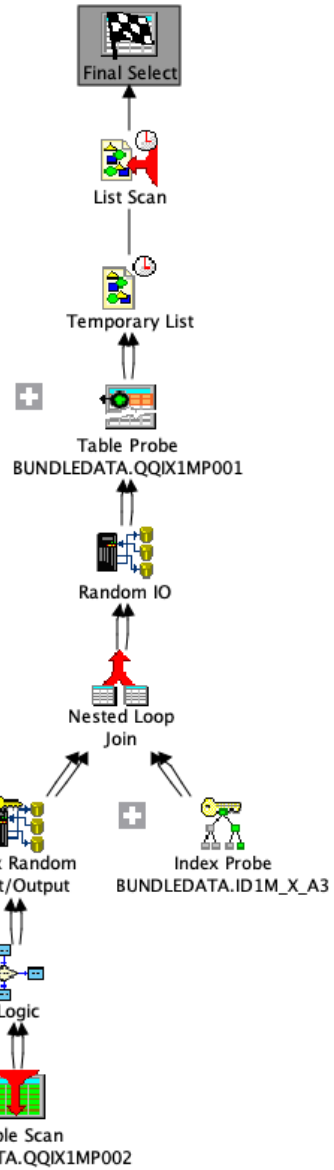
- An introduction to Visual Explain (VE), and how to read a VE diagram
- How to capture data for a VE diagram
- The parts (nodes) of a query and their roles
- Analyze and understand the optimizer’s choices through Visual Explain diagrams

© Copyright IBM Corporation 2026

What is Visual Explain?

- Diagram that depicts the set of steps (called nodes) the database takes to process a query request

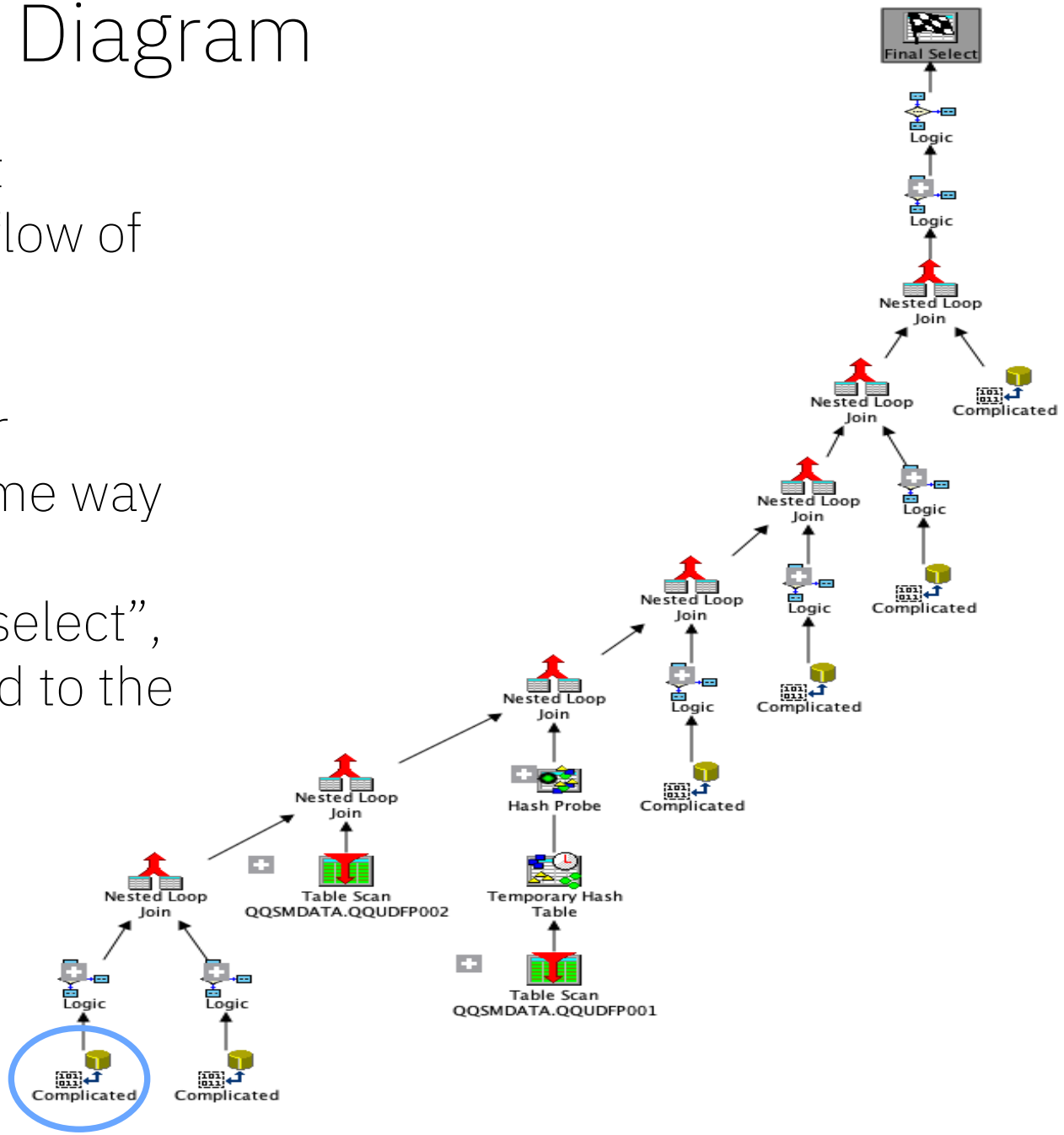
- Includes many metrics relevant to a performance discussion:
  - Estimated: time, IO requests
  - Selectivity and cardinality
  - Indexes used (or not used!)
  - Much more…



© Copyright IBM Corporation 2026

How to Read a VE Diagram

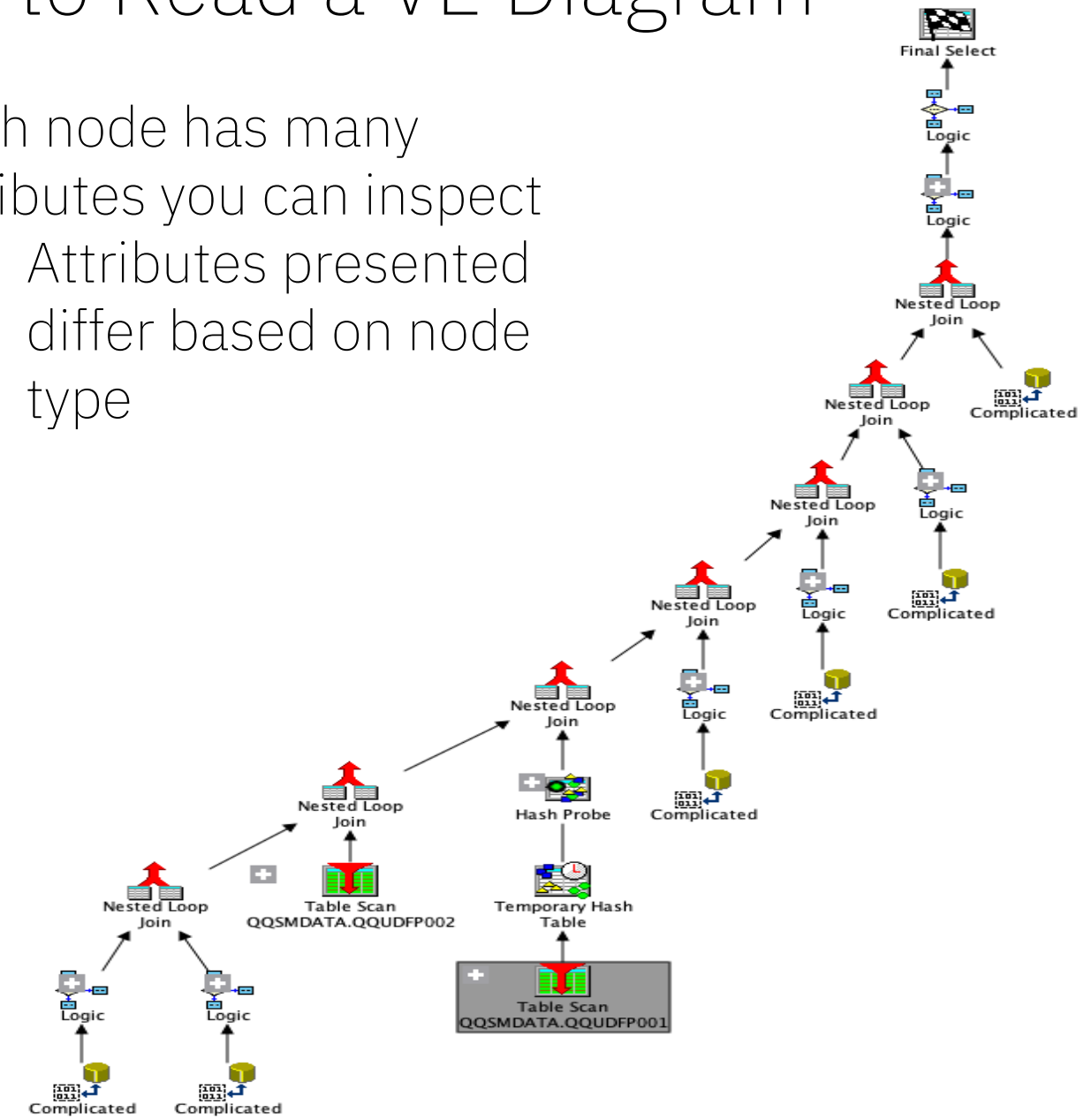
- Bottom-up, left-to-right
  - Arrows depict the flow of data
- Each node processes or manipulates data in some way
- Always ends in a “final select”, where rows are returned to the application



© Copyright IBM Corporation 2026

How to Read a VE Diagram

- Each node has many attributes you can inspect
  - Attributes presented differ based on node type

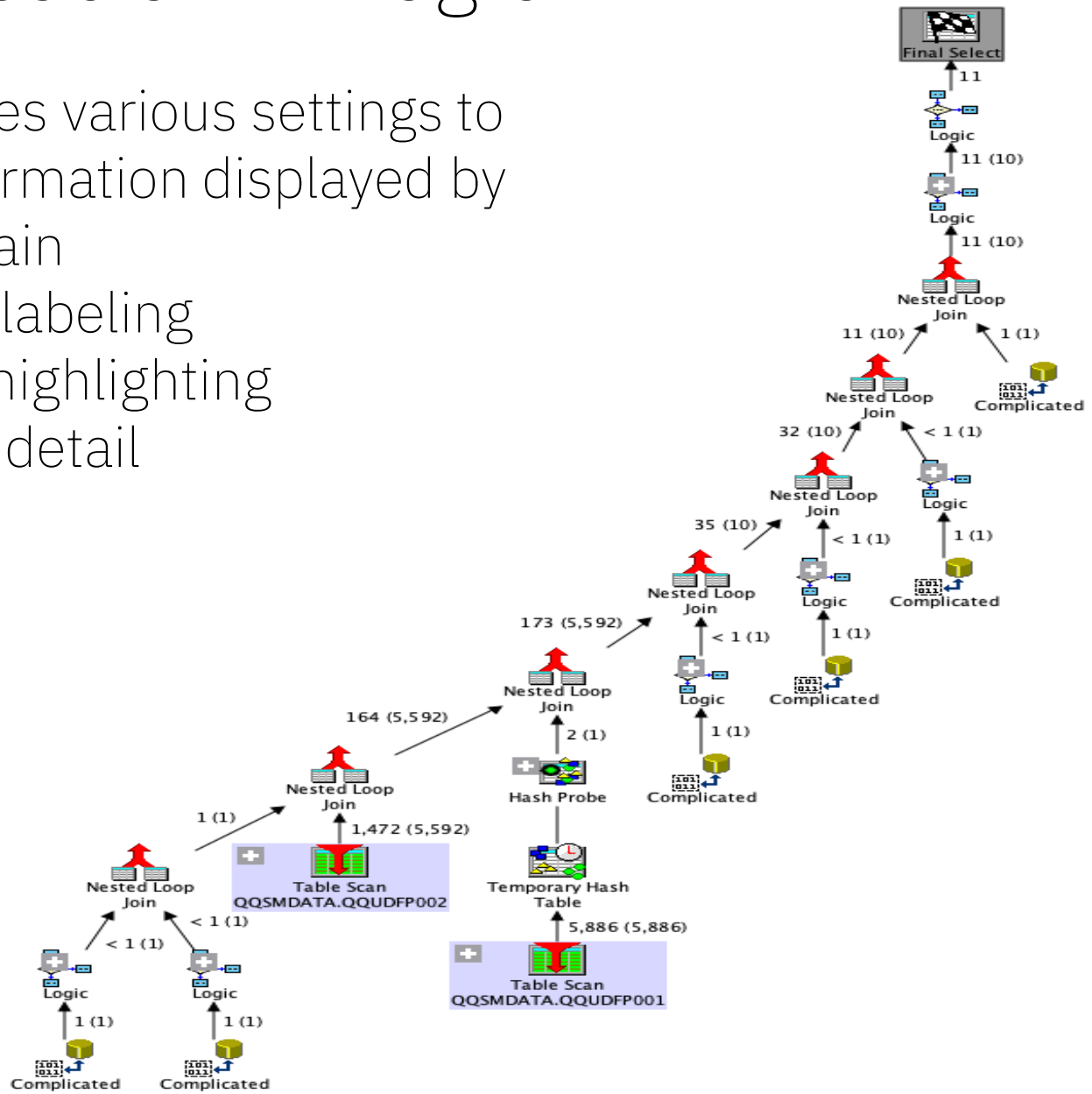


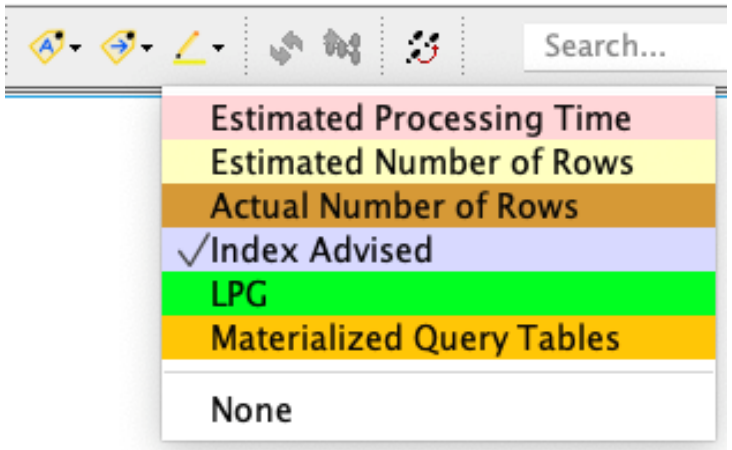
Attribute	Value
Estimated Time Information (Star	
Processing Time(ms)	62.04
Cumulative Time(ms)	0
Additional Table Info	
Total Rows in Table	5,886
Table Size(bytes)	5,192,416
Active Table Rows	5,886
Deleted Table Rows	0
Estimated rows selected and quei	
Rows Selected Per Plan Step Iteration	5,886
Rows Processed During Last Plan Step	5,886
Plan Step Iterations	1
Total Rows Selected	5,886
Total Rows Processed	5,886
Optimize for N Rows	All
Percent Selectivity	100
Cumulative Percent Selectivity	100
Fetch N Rows	All
Estimated Cost Information Abou	
Processing Time(ms)	62.04
I/O Or CPU Bound	I/O Bound
CPU Cost(ms)	.746
I/O Cost(ms)	62.04
I/O Count	33
Average IO Read Time (ms)	7.755
I/O Cost Reduction	No
PreLoad Relation	Yes
Memory Used(bytes)	4,145,152
Memory Constrained	No
Cumulative Memory Constrained	No

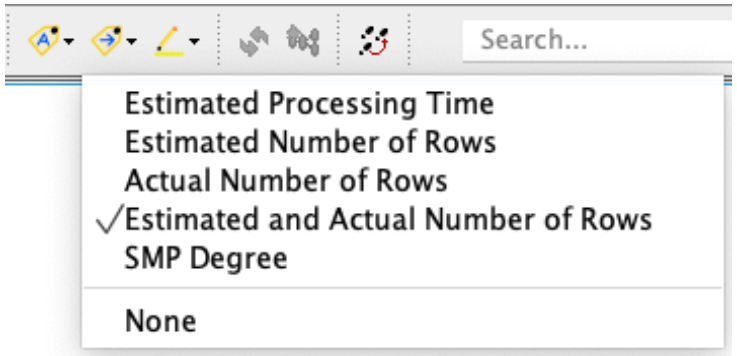
© Copyright IBM Corporation 2026

How to Read a VE Diagram

- ACS provides various settings to modify information displayed by Visual Explain
  - Arrow labeling
  - Node highlighting
  - Graph detail







© Copyright IBM Corporation 2026

Capturing Data for Visual Explain

- Many ways to generate Visual Explain data:
  - Run SQL Scripts:
    - Explain
    - Run and Explain
    - Explain While Running
  - DB Performance Monitors
  - Plan Cache Snapshots
  - Live Plan Cache

© Copyright IBM Corporation 2026

Run SQL Scripts and Visual Explain

-  **Explain**
  - Shows the plan implementation

-  **Run and Explain**
  - Runs the query, displaying the result set
  - Shows the plan implementation

-  **Explain While Running**
  - Runs the query, displaying the result set
  - Shows plan implementation with actual row count

© Copyright IBM Corporation 2026

Database Monitors (Performance Monitor)

- Captures detailed information about query optimization and execution
  - Information stored in database file
  - Contains actual row count information for Visual Explain
  - Much more than just Visual Explain information!

- Created using:
  - STRDBMON/ENDDBMON, or
  - SQL Performance Center in ACS

- Generally scoped to a single job
  - Very heavyweight

- More info on available data in DBMON format: https://www.ibm.com/docs/en/i/7.5?topic=formats-sql-view

© Copyright IBM Corporation 2026

Plan Cache Snapshots

- Moment-in-time view of the Plan Cache
  - Can capture various subsets of the Plan Cache:
    - Entire Plan Cache
    - Top N plans for a given metric (CPU usage, IO read operations)
    - Specific QRO hash

- Creating using:
  - SQL Performance Center, or
  - QSYS2.DUMP_PLAN_CACHE, or
  - QSYS2.DUMP_PLAN_CACHE_TOPN

- Contains plan metrics averaged over all query executions

- Creates a database file using the Database Monitor format
  - Contains similar information

© Copyright IBM Corporation 2026

Live Plan Cache

- View of the plan cache in real-time
  - Contains additional information due to ”live” nature:
    - Top three longest executions per query (can be changed up to 50)
    - List of jobs that utilized the plan

- Can only be viewed with the SQL Performance Center

- Can’t be saved into database file

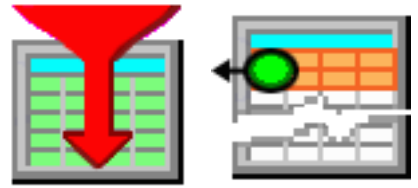
© Copyright IBM Corporation 2026

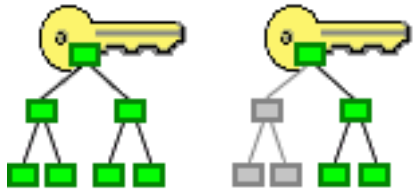
Query Anatomy

© Copyright IBM Corporation 2026

Selection

- Driven by the SELECT, WHERE, and HAVING keywords
  - SELECT: What data to grab
  - WHERE: What data to include/exclude
  - HAVING: What data to include/exclude after grouping
- Two permanent sources of data: tables and indexes
  - Many types of temporary data sources
- Two main actions: scan and probe
  - A scan looks at every row in the structure
  - A probe looks at a specific set of rows
- Need to consider views, MQTs, and derived/sparse indexes

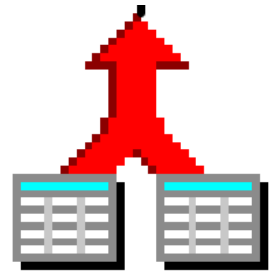




© Copyright IBM Corporation 2026

Joins

- Driven by the JOIN keyword
  - Also driven by subselects and CTEs
- Many different types of joins
  - Inner, left/right/full outer, exception, Cartesian
- Join types influence the set of possible join orders
  - Ex: left outer forces join order
- Without proper schema design, join estimation can pose a major pain point for optimization



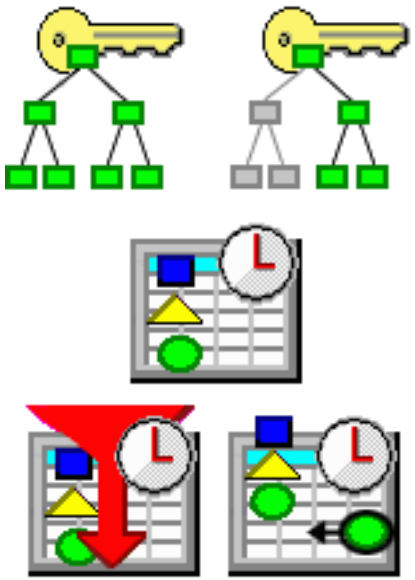
© Copyright IBM Corporation 2026

Ordering

- Driven by the ORDER BY keyword

- Data must be copied into a temporary sorted list
  - Consumes memory
  - Entire “feeding tree” must be run before rows can be selected out of the sorted list
    - Can’t guarantee data is sorted until all data is seen

- Radix index can service the sorting request with no sorted list needed
  - Radix index structure is inherently sorted

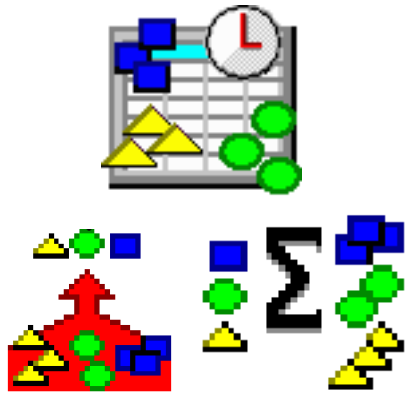


© Copyright IBM Corporation 2026

Grouping

- Driven by the GROUP BY and DISTINCT keywords

- Various grouping strategies exist:
  - Use hash table to distinct/aggregate values
  - Use “distinctor” to distinct values
  - Use index to handle grouping



- Radix indexes and EVIs can service grouping
  - If radix index is built over GROUP BY column
  - EVI can contain built-in aggregations

© Copyright IBM Corporation 2026

Row Number Lists and Bitmaps

- Large rows can be expensive to process, causing high memory usage
  - Optimizer must fit plan into “fair share” of memory
- Data must be copied into temporary structures during processing
  - Sorting, grouping
- Utilizing row numbers can prevent/delay copying of large data
  - Can also build bitmap against table


```
SELECT FIRSTNAME, LASTNAME, BIO, PROFILEPIC
FROM USERS
ORDER BY FIRSTNAME DESC, LASTNAME DESC;
```

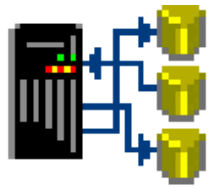
USERS			
FIRSTNAME CHAR(50)	LASTNAME CHAR(100)	BIO CHAR(1000)	PROFILEPIC BLOB(1M)

© Copyright IBM Corporation 2026

Miscellaneous Nodes

- UDF/UDTFs that are marked DETERMINISTIC can have a cache built over top
  - Significantly improves performance for functions with a low input cardinality

- Page faults can be avoided with Random I/O and Clustered I/O nodes
  - You can't control this, but it can be interesting to know!



- Symmetric Multiprocessing (SMP) requires additional nodes to function:
  - Range node: Divides work evenly between threads
  - Buffer node: Provides work to thread as needed

© Copyright IBM Corporation 2026

Visual Explain Analysis

© Copyright IBM Corporation 2026

Getting Started

- Start with most expensive queries
  - Sum of all runs, not individual
  - Plan Cache snapshots are your friend

Start Time	Most Expensive Time (sec)	Total Processing Time (sec)	Total Times Run	Average Processing Time (sec)	Statement
2024-07-30 13:37:31.513246	0.4606	26.1162	1248	0.0209	SELECT SQL_ID, SKIPSTMT, SKIPCHKSM, SETUP, IGNFAIL, NUM_
2024-07-30 12:51:59.847728	8.9678	8.9678	1	8.9678	select a.partkey,c.mpk from star1g.part_dim a left outer join

Much bigger, more expensive query

Smaller query runs much more often, has more runtime!
Much better analysis target!

© Copyright IBM Corporation 2026

Indexing Strategy

- Indexes are crucial to database performance
  - Significantly speeds up selection, sorting, and grouping
  - Excessive indexes can significantly harm performance

- How to analyze index usage?
  - Wide net:
    - Plan Cache Snapshot - “Analyze” functionality
    - MTI_INFO SQL Service
    - QSYS2.SYSIXADV
  - Fine tuning:
    - Visual Explain

© Copyright IBM Corporation 2026

Indexing Strategy

```
MERGE INTO ALLOBJ_USERS AU USING (
        SELECT AUTHORIZATION_NAME,
               USER_CREATOR,
               STATUS,
               NO_PASSWORD_INDICATOR,
               USER_DEFAULT_PASSWORD,
               SPECIAL_AUTHORITIES,
               GROUP_PROFILE_NAME,
               SUPPLEMENTAL_GROUP_LIST,
               TEXT_DESCRIPTION
            FROM QSYS2.USER_INFO
            WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%'
                  OR AUTHORIZATION_NAME IN (SELECT USER_PROFILE_NAME
                        FROM QSYS2.GROUP_PROFILE_ENTRIES
                        WHERE GROUP_PROFILE_NAME IN (SELECT AUTHORIZATION_NAME
                                    FROM QSYS2.USER_INFO
                                    WHERE SPECIAL_AUTHORITIES LIKE '%*ALLOBJ%'))
            ORDER BY AUTHORIZATION_NAME
    ) LIVE ON AU.USER_NAME = LIVE.AUTHORIZATION_NAME
    WHEN NOT MATCHED THEN INSERT VALUES (
            . . .
        )
```

Most Expensive Time	Total Processing Time	Total Times Run	Average Processing Time	Indexes Advised	Statement
24.3269	495.8078	25	19.8323	1	MERGE INTO

© Copyright IBM Corporation 2026

Indexing Strategy

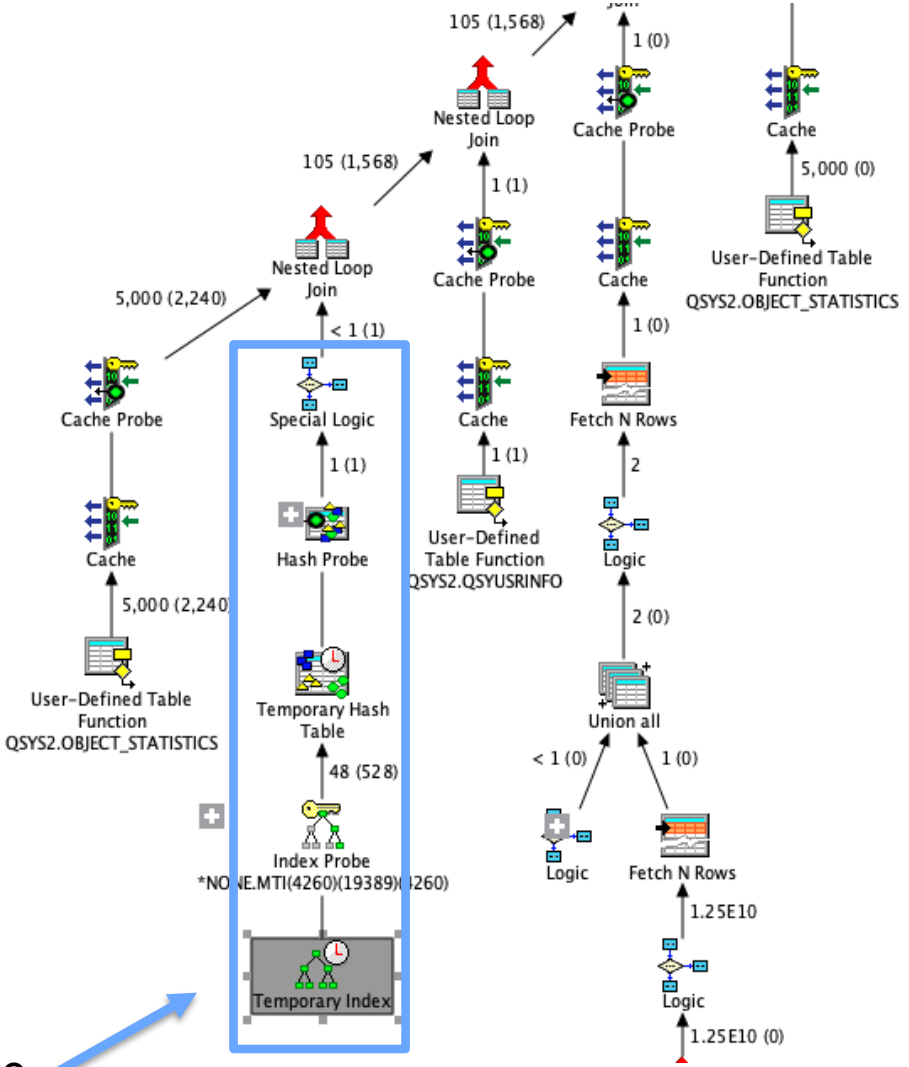
Index and Statistics Advisor

Indexes | Statistics

The following indexes were advised:

Table	Schema	Columns	Index Type
ALLOBJ_USERS	CAROLNEW	USER_NAME	BINARY RADIX

Estimated Time Information (Start Time, Total Time)	
Estimated Population Time	0
I/O Or CPU Bound	Equivalent
CPU Cost(ms)	0
I/O Cost(ms)	0
I/O Count	0
Index Info	
Number of Index Entries	48
Key Cardinality	47
Size of Index, in Bytes	147,456
Amortization Value	1
List of Key Columns	ALLOBJUSRS_19.USER_NAME
Key Size(bytes)	19
Shareable	Yes
Index was Reused	Yes



Leg accounts for 0.05% of estimated query runtime

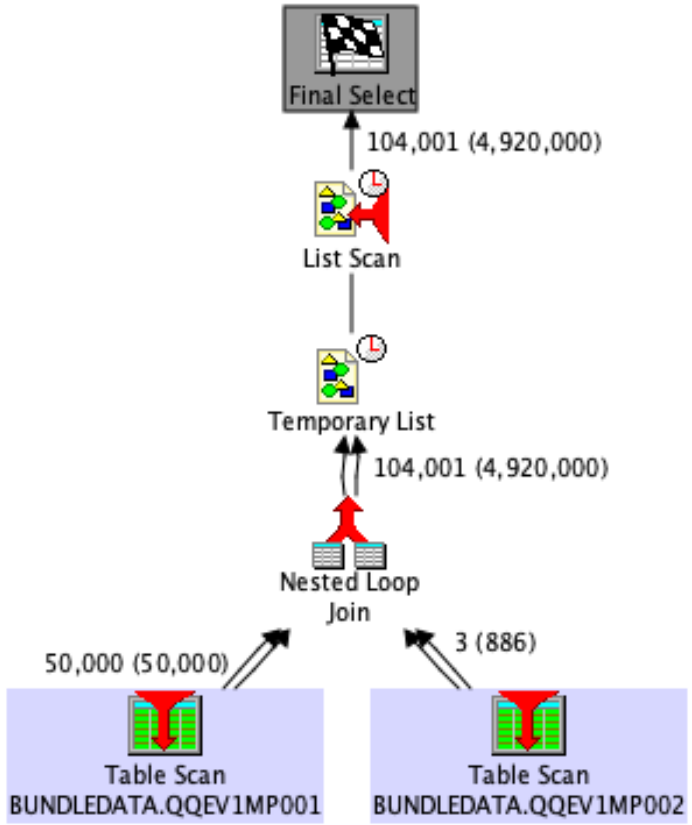
© Copyright IBM Corporation 2026

Indexing Strategy

```
SELECT XXX.KEYFLD,
       YYY.KEYFLD
    FROM BUNDLEDATA.QQEV1MP001 XXX,
         BUNDLEDATA.QQEV1MP002 YYY
    WHERE (XXX.TSTAMP1 = YYY.TSTAMP1
           AND XXX.DFLOAT2 = YYY.DFLOAT2
           AND XXX.CHAR3 IS NULL
           AND YYY.VCHAR2 > ?
           AND YYY.VCHAR2 = ?
           AND ? IS NOT NULL
           AND XXX.VCHAR2 IS NOT NULL)
        OR (XXX.CHAR3 = YYY.CHAR3
           AND XXX.TSTAMP1 = YYY.TSTAMP1
           AND XXX.CHAR3 IN (?, ?, ?, ?, ?, ?)
           AND XXX.CHAR3 = ?
           AND YYY.VCHAR2 != ?)
        OR (XXX.TSTAMP1 = YYY.TSTAMP1
           AND XXX.CHAR3 = YYY.CHAR3
           AND XXX.CHAR3 IS NULL
           AND YYY.VCHAR2 > ?
           AND YYY.VCHAR2 = ?
           AND ? IS NOT NULL
           AND XXX.VCHAR2 IS NOT NULL)
        OR . . .
```

Most Expensive Time	Total Processing Time	Total Times Run	Average Processing Time	Indexes Advised	Statement
87.8603	92.7554	9	10.3061	6	SELECT XX

Indexing Strategy

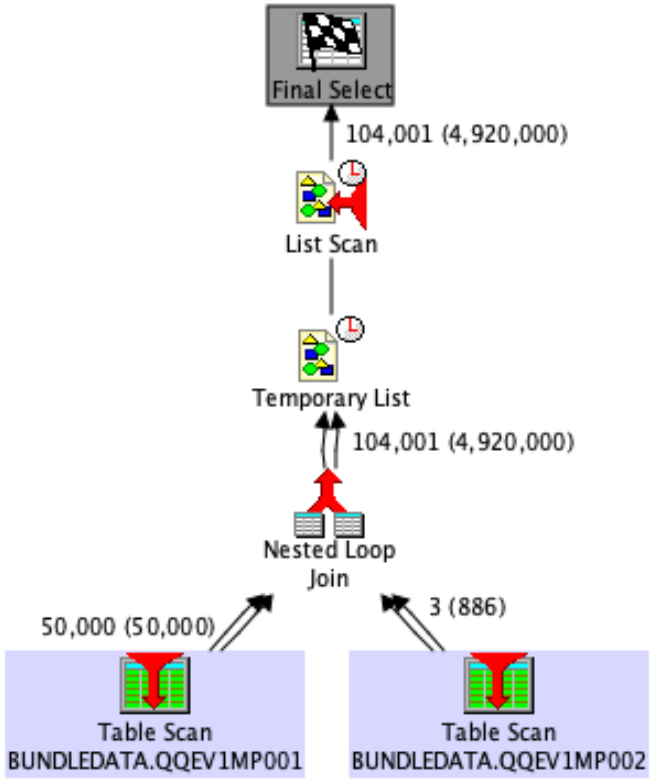


How do I know which indexes to create?

Table	Schema	Columns	Index Type
QQEV1MP001	BUNDLEDATA	DFLOAT2, VCHAR2, CHAR3	BINARY RADIX
QQEV1MP001	BUNDLEDATA	CHAR3, TSTAMP1	BINARY RADIX
QQEV1MP001	BUNDLEDATA	TSTAMP1, DFLOAT2	BINARY RADIX
QQEV1MP002	BUNDLEDATA	CHAR3, TSTAMP1, VCHAR2	BINARY RADIX
QQEV1MP002	BUNDLEDATA	DFLOAT2, VCHAR2, CHAR3	BINARY RADIX
QQEV1MP002	BUNDLEDATA	TSTAMP1, DFLOAT2, VCHAR2	BINARY RADIX

© Copyright IBM Corporation 2026

Indexing Strategy



Estimated rows selected and query join info	
Rows Selected Per Plan Step Iteration	50,000
Rows Processed During Last Plan Step	50,000
Plan Step Iterations	1
Total Rows Selected	50,000
Total Rows Processed	50,000
Optimize for N Rows	All
Percent Selectivity	100

Estimated rows selected and query join info	
Rows Selected Per Plan Step Iteration	2.08
Rows Processed During Last Plan Step	49,200
Plan Step Iterations	50,000
Total Rows Selected	104,000
Total Rows Processed	2.46E9
Optimize for N Rows	All
Percent Selectivity	.004

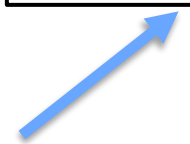
Estimated Time Information	
Processing Time(ms)	134.672

Estimated Time Information	
Processing Time(ms)	283,574

© Copyright IBM Corporation 2026

Indexing Strategy

Most Expensive Time	Total Processing Time	Total Times Run	Average Processing Time
87.8603	92.7554	9	10.3061



Why is the peak runtime so high?

Why is the temporary storage so high?

Temporary Storage Used (MB)	108

"Average Synchronous DB Reads"	"Average Synchronous DB Writes"	"Average Asynchronou s DB Reads"	"Average Asynchronou s DB Writes"	"Average Page Faults "
38.4444	0.0000	58.6666	0.0000	38.5555

© Copyright IBM Corporation 2026

Indexing Strategy

- If a Maintained Temporary Index exists:
  - Good sign that a permanent index should be created, but not perfect
  - Investigate with MTI_INFO
    - Reference count, MTI size

- If an Advised Index entry exists:
  - Created by the optimizer when a new index would allow a new optimization strategy to be tried
    - Could be a very good/bad strategy, but we don’t know yet!
  - Approach advised indexes with scrutiny

© Copyright IBM Corporation 2026

Indexing Strategy

- If a Maintained Temporary Index exists:
  - Good sign that a permanent index should be created, but not perfect
  - Investigate with MTI_INFO
    - Reference count, MTI size

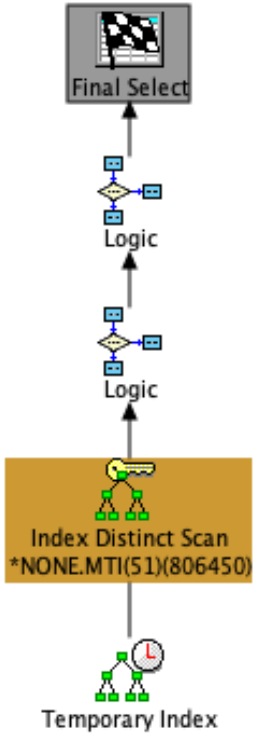
```
WITH TOP_5_MTI AS (
        SELECT TABLE_SCHEMA, TABLE_NAME, REFERENCE_COUNT,
               KEY_DEFINITION, MTI_SIZE, QRO_HASH_JSON
            FROM TABLE (
                    QSYS2.MTI_INFO(TABLE_SCHEMA => 'PAY700')
                )
            ORDER BY REFERENCE_COUNT DESC LIMIT 5
    )

SELECT * FROM TOP_5_MTI,
         JSON_TABLE(QRO_HASH_JSON,
                    'lax $.QRO_HASH_LIST'
                    COLUMNS(EXTRACTED_QRO_HASH CHAR(8) PATH 'lax $')
         );
```


© Copyright IBM Corporation 2026

Indexing Strategy

TABLE_SCHEMA	TABLE_NAME	REFERENCE_COUNT	KEY_DEFINITION	PLAN_ID
PAY700	PS_PAYGROUP_TBL	11	COUNTRY	330820
PAY700	PS_PAYGROUP_TBL	11	COUNTRY	330825
PAY700	PS_CM_EVALUAT...	9	EVALUATION_T...	330405
PAY700	PS_CM_EVALUAT...	9	EVALUATION_T...	330413
PAY700	PS_CM_EVALUAT...	9	EVALUATION_T...	330441




```
CALL QSYS2.DUMP_PLAN_CACHE(FILESCHEMA => 'RMOELLER',
                           FILENAME => 'DUMPPLANID',
                           PLAN_IDENTIFIER => 330820
```

Actual Runtime Information	
Optimization Time (ms)	4
Longest Key Range Estimate (ms)	0
Key Range Estimate Timed Out	No
Average Run Time (ms)	73
Average Statement Open Time (ms)	Not Available
Average Statement Fetch Time (ms)	73
Average Statement Close Time (ms)	Not Available
Average Rows Fetched	10,310
Total Times Query Was Run	2
Total Time For All Runs (ms)	147
Average CPU Time (ms)	5

© Copyright IBM Corporation 2026

SMP: Helping or Hiding Performance Problems?

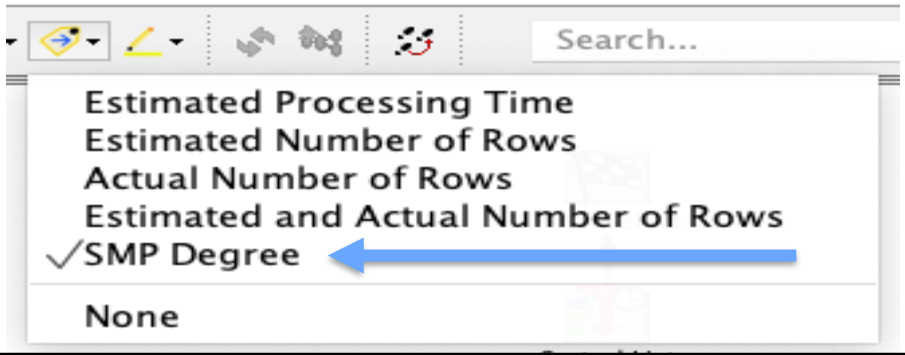
- Symmetric Multi-Processing (SMP) allows DB2 for i to process queries with multiple threads
  - Can significantly increase query performance

- Number of threads available scales with available CPU cores
  - With PARALLEL_DEGREE set to *OPTIMIZE, maximum is equal to the number of CPU cores allocated to the partition

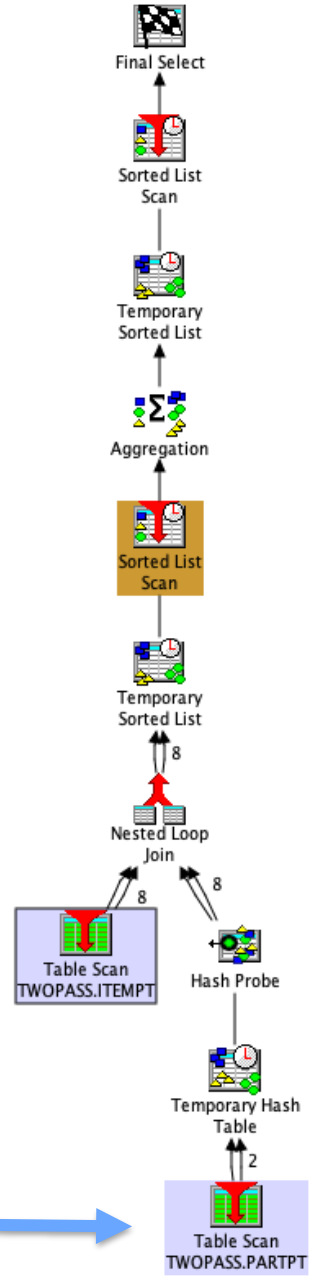
- SMP can function as a band-aid to performance problems
  - Doesn’t fix underlying problem, just applies a temporary solution

© Copyright IBM Corporation 2026

SMP: Helping or Hiding Performance Problems?



Estimated rows selected and query join info	
Rows Selected Per Plan Step Iteration	3,334,008
Rows Processed During Last Plan Step	1.2E7
Plan Step Iterations	1
Total Rows Selected	3,334,008
Total Rows Processed	1.2E7
Optimize for N Rows	All
Percent Selectivity	27.777
DB2 SMP Parallel Information	
Parallel Degree Used	8
Estimated rows selected and query join info	
Rows Selected Per Plan Step Iteration	133,333
Rows Processed During Last Plan Step	400,000
Plan Step Iterations	1
Total Rows Selected	133,333
Total Rows Processed	400,000
Optimize for N Rows	All
Percent Selectivity	33.333
DB2 SMP Parallel Information	
Parallel Degree Used	2

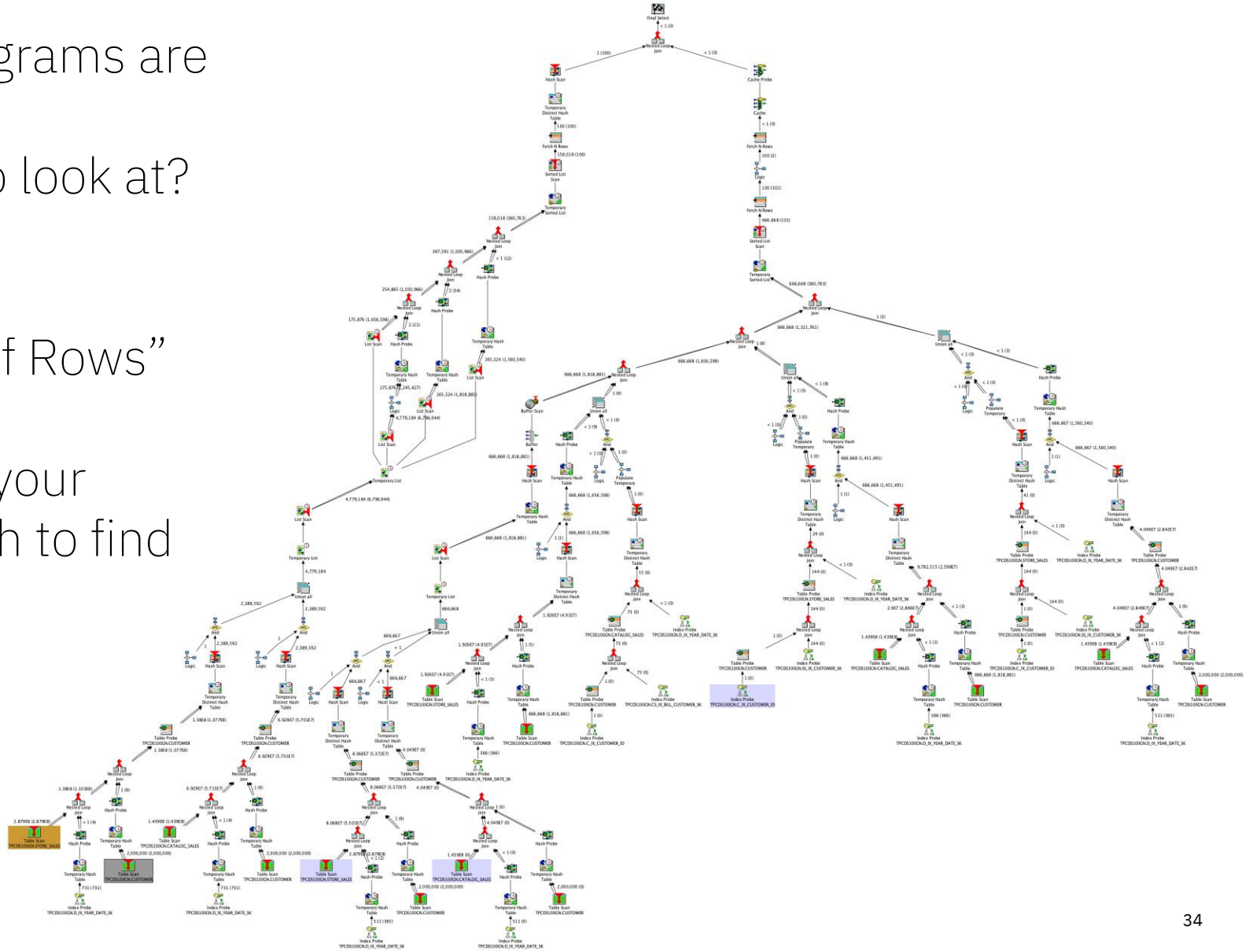


© Copyright IBM Corporation 2026

Navigating Large Trees (and more SMP)

- Sometimes Visual Explain diagrams are massive…
  - How do you know what to look at?

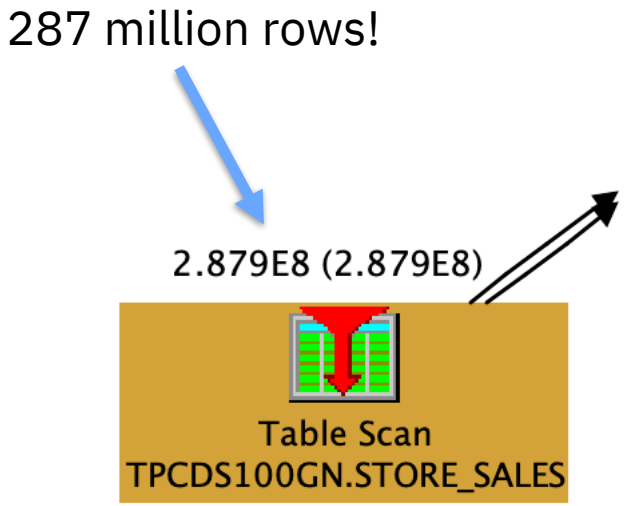
- Two tips:
  - Enable “Actual Number of Rows” highlighting
  - Use the tree structure to your advantage – binary search to find the expensive subtree



© Copyright IBM Corporation 2026

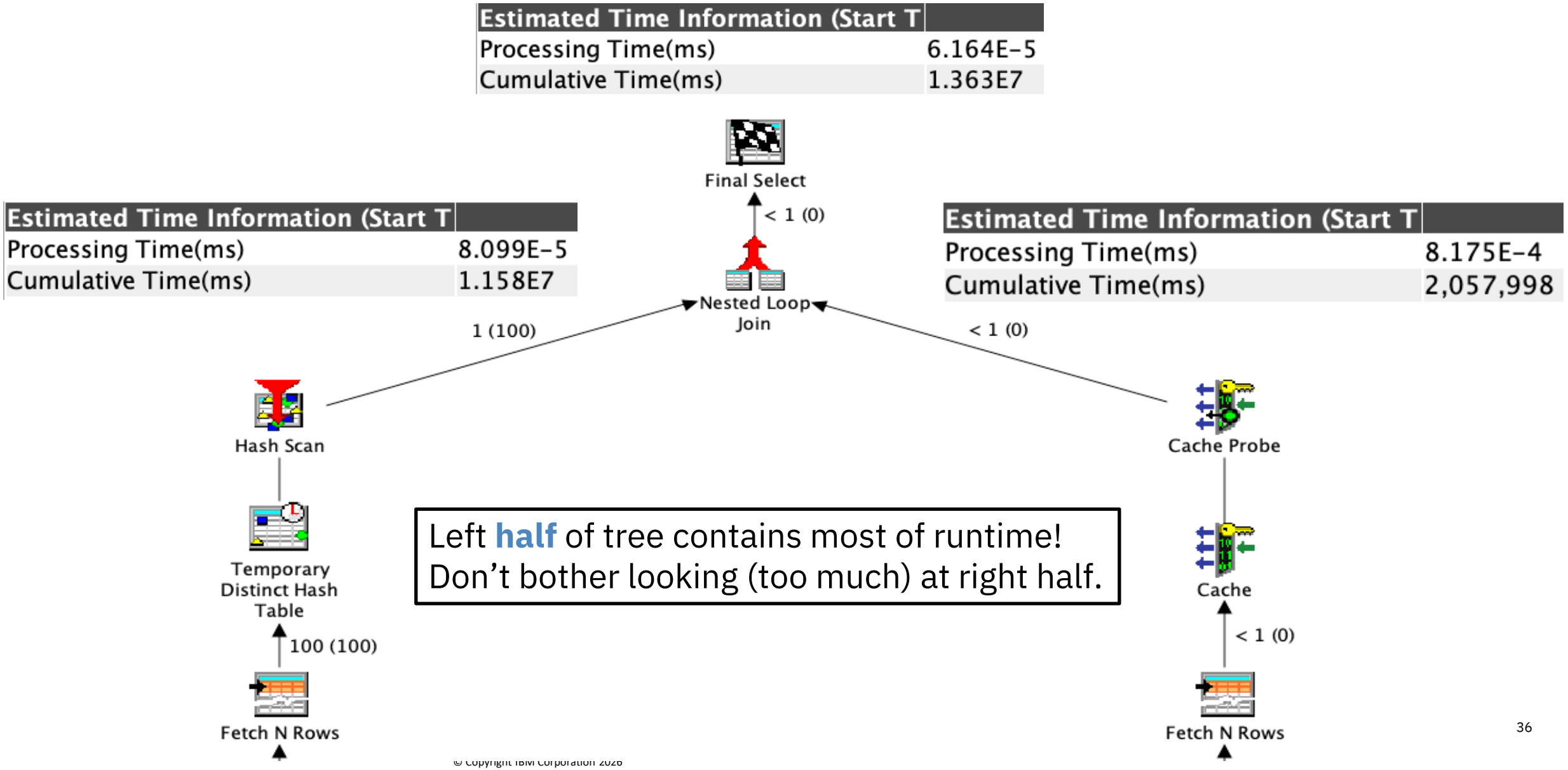
Navigating Large Trees (and more SMP)

- "Actual Number of Rows" highlighting
  - The largest number of rows isn't always going to be the most expensive part of the tree, but it's a good clue
  - If multiple tables have the same count, only highlights left-most table



© Copyright IBM Corporation 2026

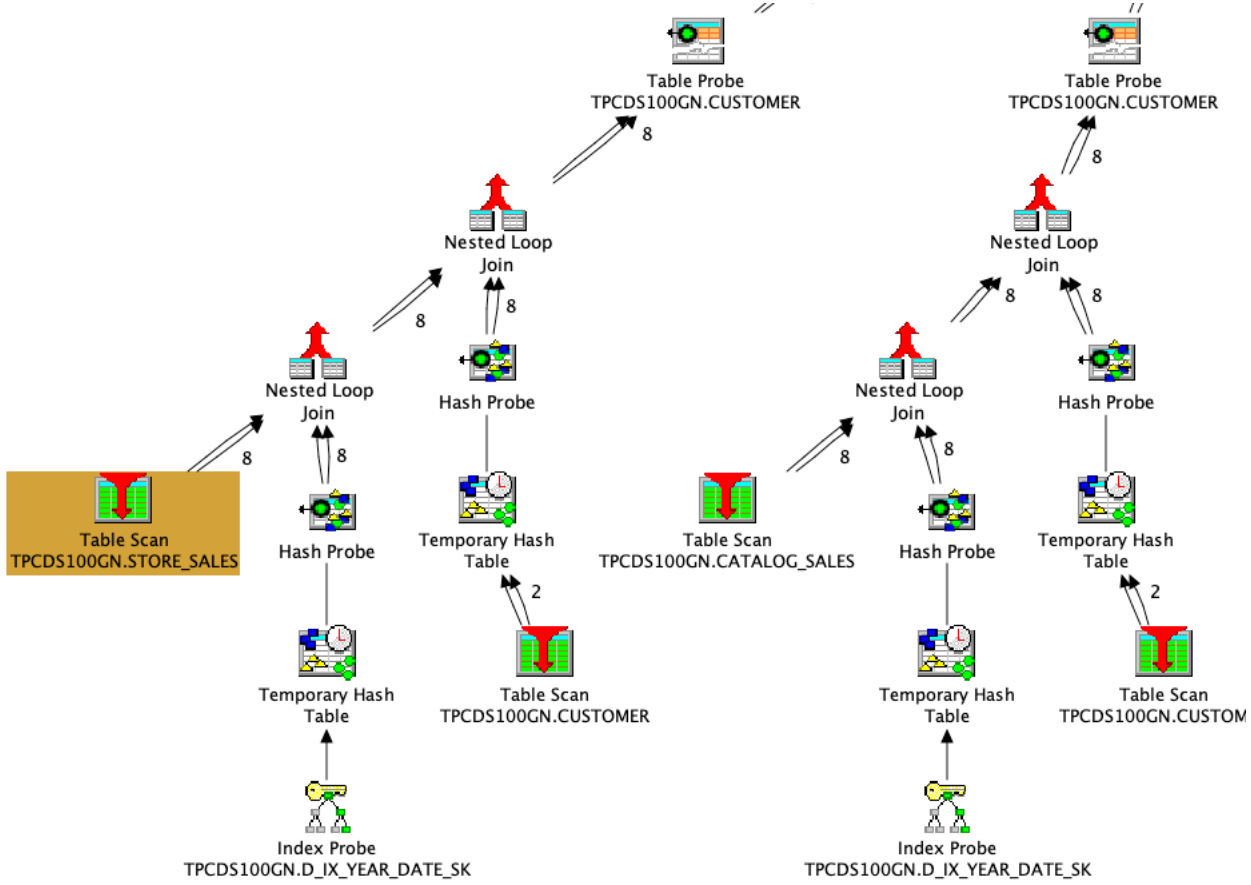
Navigating Large Trees (and more SMP)



© Copyright IBM Corporation 2026

Navigating Large Trees (and more SMP)

- Most expensive operations are table probes against CUSTOMER table
  - Quick to find by splitting up tree by subtrees
- Query looks well-structured and has proper indexing
  - SMP not hiding any issues
- SMP was a very good fit for this query:
  - Lots and lots of data
  - Very long-running
  - Expensive nodes are CPU-bound



© Copyright IBM Corporation 2026

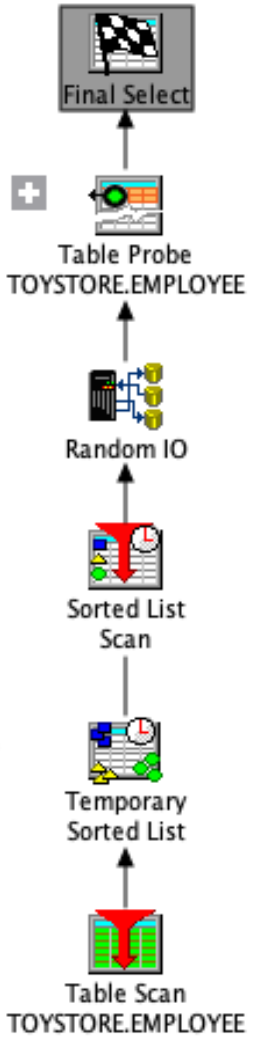
Temporary Data Structure Latency

- "Why does my query become **way** slower when I add ordering or grouping?"

- When a temporary data structure is built, it must be fully populated before it can be used
  - Ex: If asked to sort a series of words, you don't know the first word in the list until you've sorted all the words!

- If first-row latency is important (ex: GUI application):
  - QAQQINI option OPTIMIZATION_GOAL set to *FIRSTIO
  - Index can provide much faster sorting/grouping

Scans every row from EMPLOYEE first, then returns rows to the Sorted List Scan



© Copyright IBM Corporation 2026

I/O Bound Queries

- Each node in Visual Explain contains an estimated CPU and IO time:
  - Higher of the two is the total “cost” of the node

- I/O bound nodes may benefit from object (table, index) being stored on SSD
  - CHGPFM UNIT(*SSD) to store on SSD
  - Query runs faster, less I/O time spent waiting
  - Can run even faster with SMP, increases threading capabilities

Estimated Cost Information About the Plan Perform	
Processing Time(ms)	33,564
I/O Or CPU Bound	I/O Bound
CPU Cost(ms)	12,190
I/O Cost(ms)	67,129
I/O Count	221,308
Average IO Read Time (ms)	8.492
I/O Cost Reduction	No
PreLoad Relation	No
Memory Used(bytes)	1.397E7
Memory Constrained	No
Cumulative Memory Constrained	No

© Copyright IBM Corporation 2026

Searching for Poor SQL Practices

- ACS GUI is great, but it doesn’t offer every trick for performance deep dives

- Database Monitor table format contains VE data – directly query it!

- Poor SQL practices to avoid:
  - Creating intermediate tables in QTEMP
  - Poor join conditions
    - Substrings and concatenated strings are common – ouch!
  - Using logic within predicates
    - UDFs and CASE statements

© Copyright IBM Corporation 2026

Searching for Poor SQL Practices

- Let's look for queries using QTEMP, and total up the temporary storage being wasted by each query

```
SELECT (SELECT QQ1000L FROM RMOELLER.QZG0000926 Z
           WHERE QQRID = 1000 AND Z.QQJFLD = X.QQJFLD) STATEMENT_TEXT,
       SUM((SELECT QQINT03 FROM RMOELLER.QZG0000926 Z
           WHERE QQRID = 3019 AND Z.QQJFLD = X.QQJFLD)) TEMP_STORAGE_USED_MB
    FROM RMOELLER.QZG0000926 X
    WHERE QQJFLD IN (SELECT QQJFLD
                FROM RMOELLER.QZG0000926
                WHERE QQRID = 1000
                     AND QQ1000L LIKE '%QTEMP%')
GROUP BY (SELECT QQ1000L FROM RMOELLER.QZG0000926 Z
             WHERE QQRID = 1000 AND Z.QQJFLD = X.QQJFLD)
ORDER BY TEMP_STORAGE_USED DESC;
```

STATEMENT_TEXT	TEMP_STORAGE_USED_MB
DECLARE C1 CURSOR FOR SELECT A . QRECN , A . QTITY , A . QTISTY , ...	537
UPDATE QTEMP.PROPS SET INTERFACE = ? WHERE PROPERTIES_NAME IN (sel...	320
INSERT INTO EUTSMART.TEST_DESCRIPTION_AUX_TABLE(TESTNAME, PROPERTI...	304

© Copyright IBM Corporation 2026

Other Performance/Visual Explain Topics

- Memory pool size
  - Temporary storage usage
- When to use Encoded Vector instead of Binary Radix Indexes
- Environmental options (JDBC/ODBC settings, QAQQINI)
- UDF/UDTF analysis
  - Deterministic UDF caching
- Much more…

© Copyright IBM Corporation 2026

Expert Labs

Want a pro to handle your Visual Explain?

The Db2 for i SQL Performance Workshop provides hands-on education to learn Visual Explain & other ACS SQL Performance Center tooling

ibm.biz/Db2iExpertLabs or email morten.buur.Rasmussen@dk.ibm.com

© Copyright IBM Corporation 2026

Everyone can be successful with SQL

Db2 for i SQL Tutor

CLASS IN SESSION

- Aggregation of all Gist Examples
- Aggregation of all iSee video blogs, for example: **Convert Spool files to PDF easy with SQL**
- Different perspectives to easily find what you want

ibm.biz/Db2foriSQLTutor

© Copyright IBM Corporation 2026

Thank You!

© Copyright IBM Corporation 2026