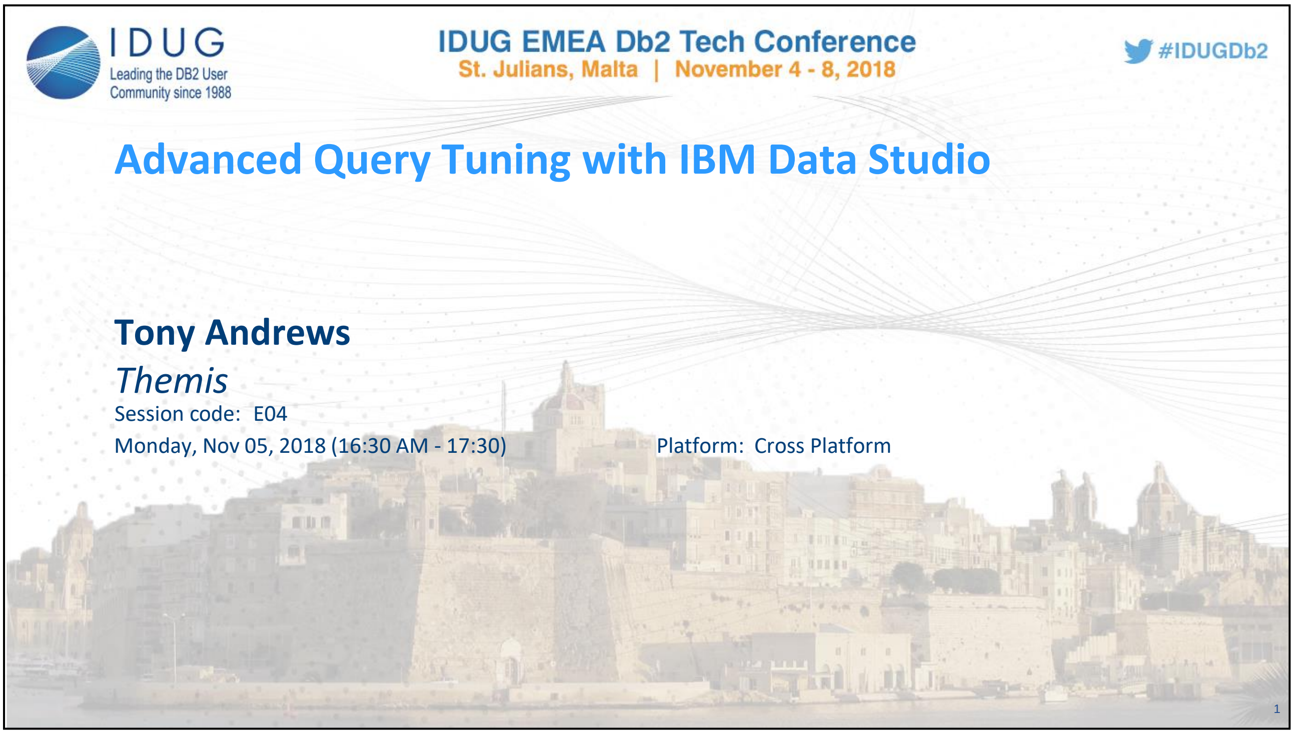


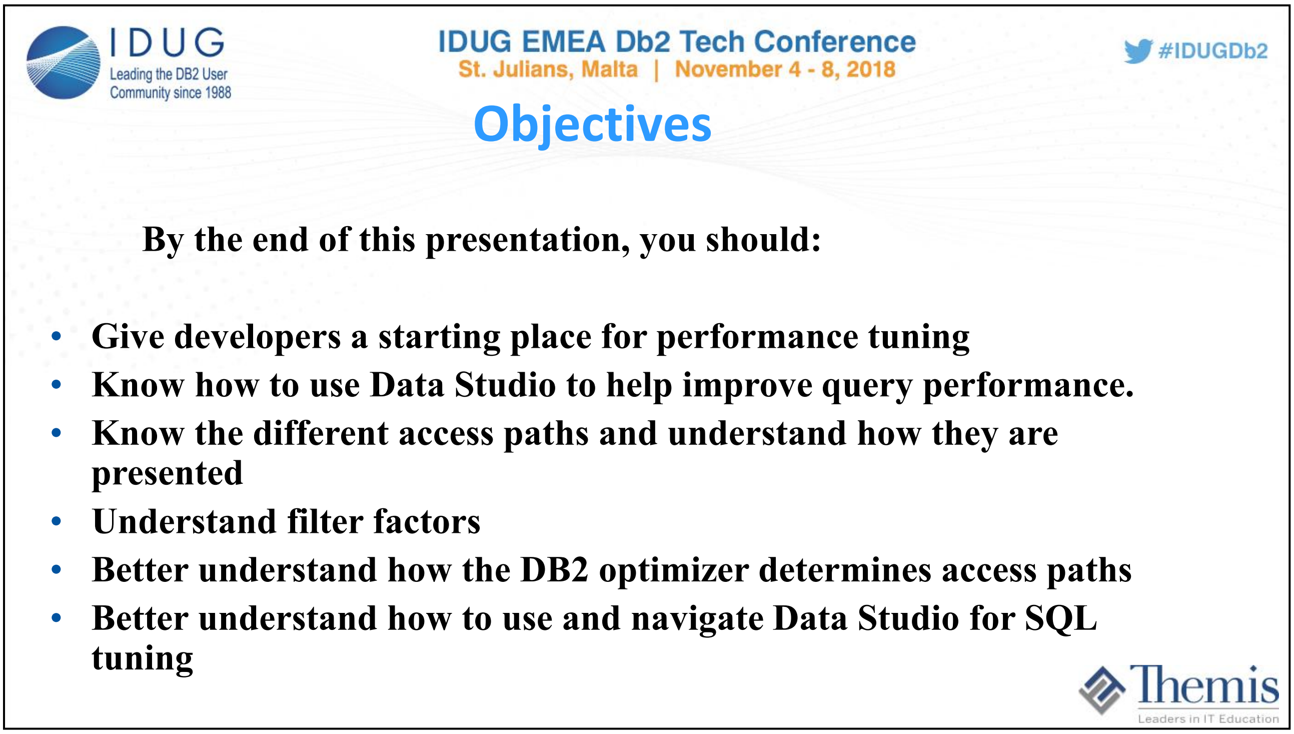
Advanced Query Tuning with IBM Data Studio

Tony Andrews

Themis

Session code: E04

Monday, Nov 05, 2018 (16:30 AM - 17:30) Platform: Cross Platform

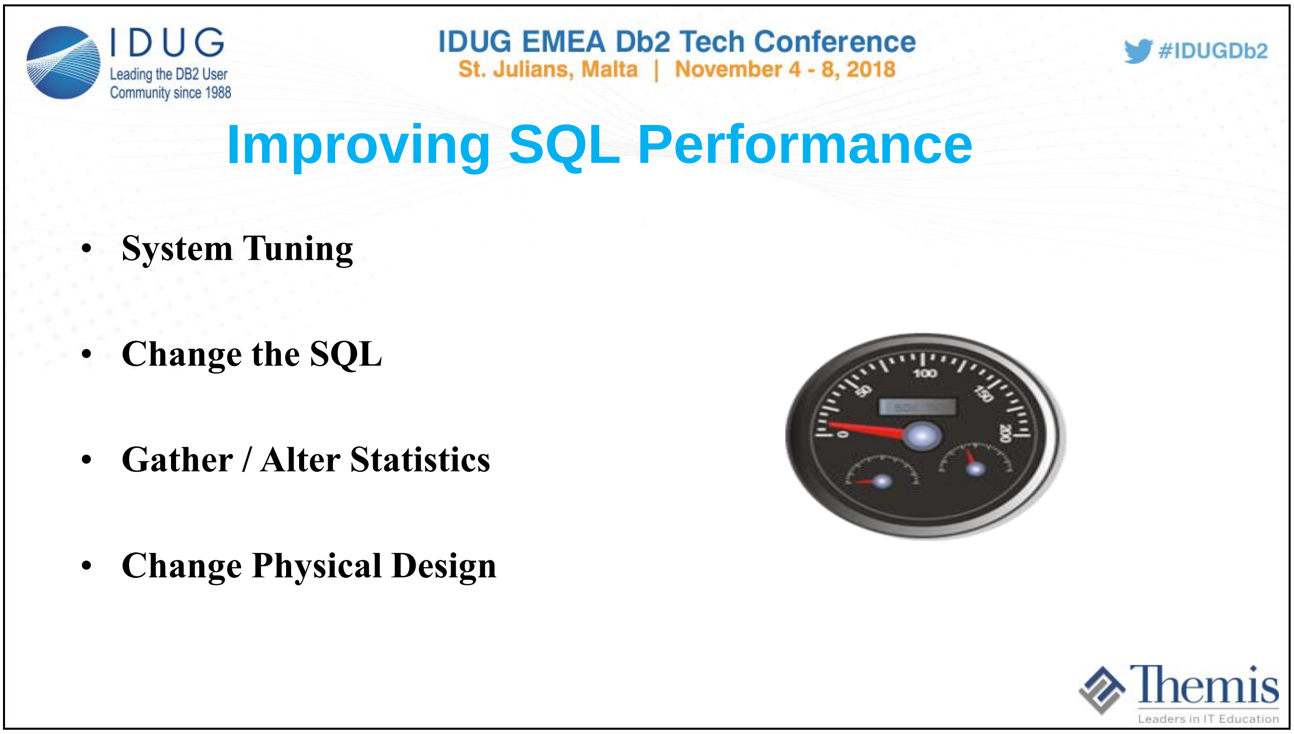


Objectives

By the end of this presentation, you should:

- Give developers a starting place for performance tuning
- Know how to use Data Studio to help improve query performance.
- Know the different access paths and understand how they are presented
- Understand filter factors
- Better understand how the DB2 optimizer determines access paths
- Better understand how to use and navigate Data Studio for SQL tuning

Themis
Leaders in IT Education



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

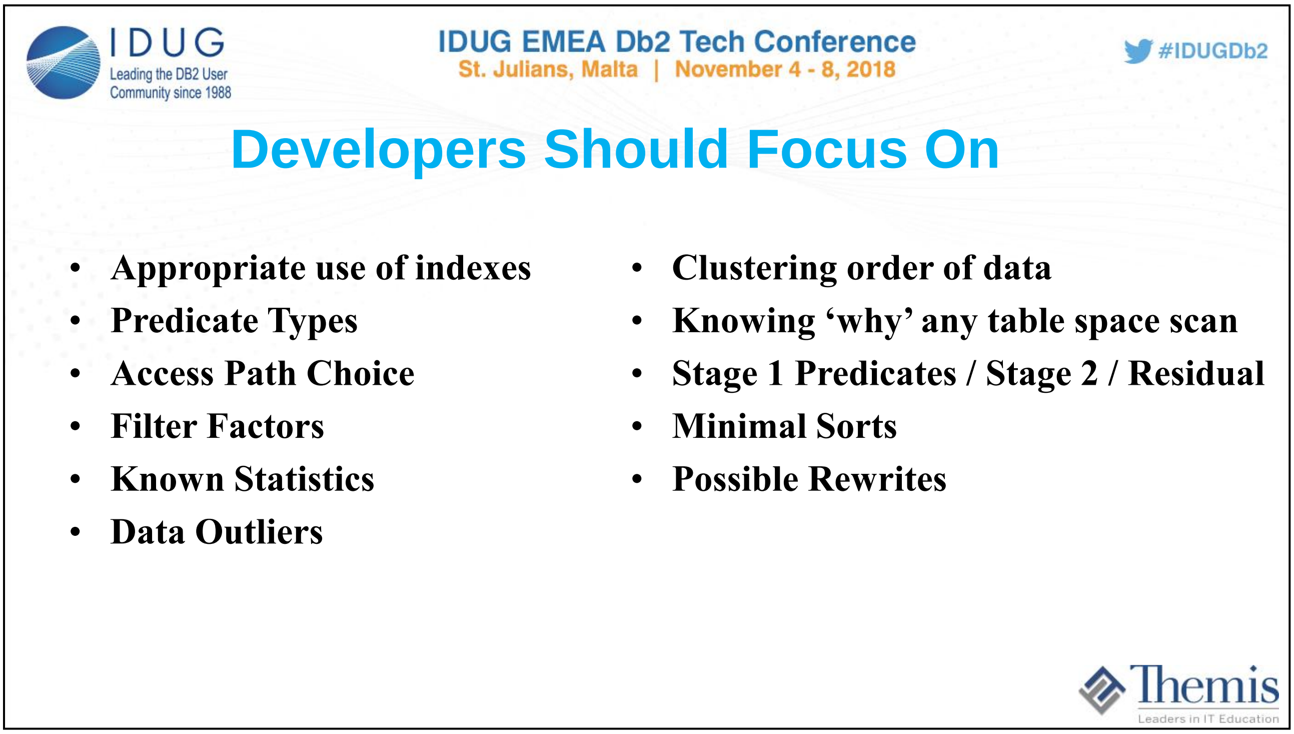
Improving SQL Performance

- **System Tuning**
- **Change the SQL**
- **Gather / Alter Statistics**
- **Change Physical Design**

0
50
100
150
200

Themis
Leaders in IT Education

Improving SQL performance can be done in one of at least 4 ways. System tuning may be done to adjust the parameters under which the DB2 subsystem operates to effectively match the workload. Altering system parameters, tuning temporary space, and adjusting bufferpool sizes and thresholds are all examples of this type of tuning. An appropriately tuned system can affect an improvement in performance. Most of the time, however, other factors dominate a tuning scenario. The SQL itself must be written in a way that may be processed efficiently by the database. An appropriate level of statistics about the data must be gathered to tell the optimizer about the nature of the data being accessed. Lastly, the way the physical objects are defined must be aligned with the types of queries that are to be performed.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Developers Should Focus On

- **Appropriate use of indexes**
- **Predicate Types**
- **Access Path Choice**
- **Filter Factors**
- **Known Statistics**
- **Data Outliers**
- **Clustering order of data**
- **Knowing 'why' any table space scan**
- **Stage 1 Predicates / Stage 2 / Residual**
- **Minimal Sorts**
- **Possible Rewrites**

Themis Leaders in IT Education

Developers should focus on all of these areas when tuning a query.

No amount of system tuning, however, can recover the resources wasted by a poor database design or poor access paths generated by the DB2 optimizer. This presentation focuses on optimization and tuning at the SQL level. In general, we want the optimizer to generate an access path that eliminates as much data from consideration as early as possible in the process, takes advantage of indexes, and has up to date statistics to help its access path choices..

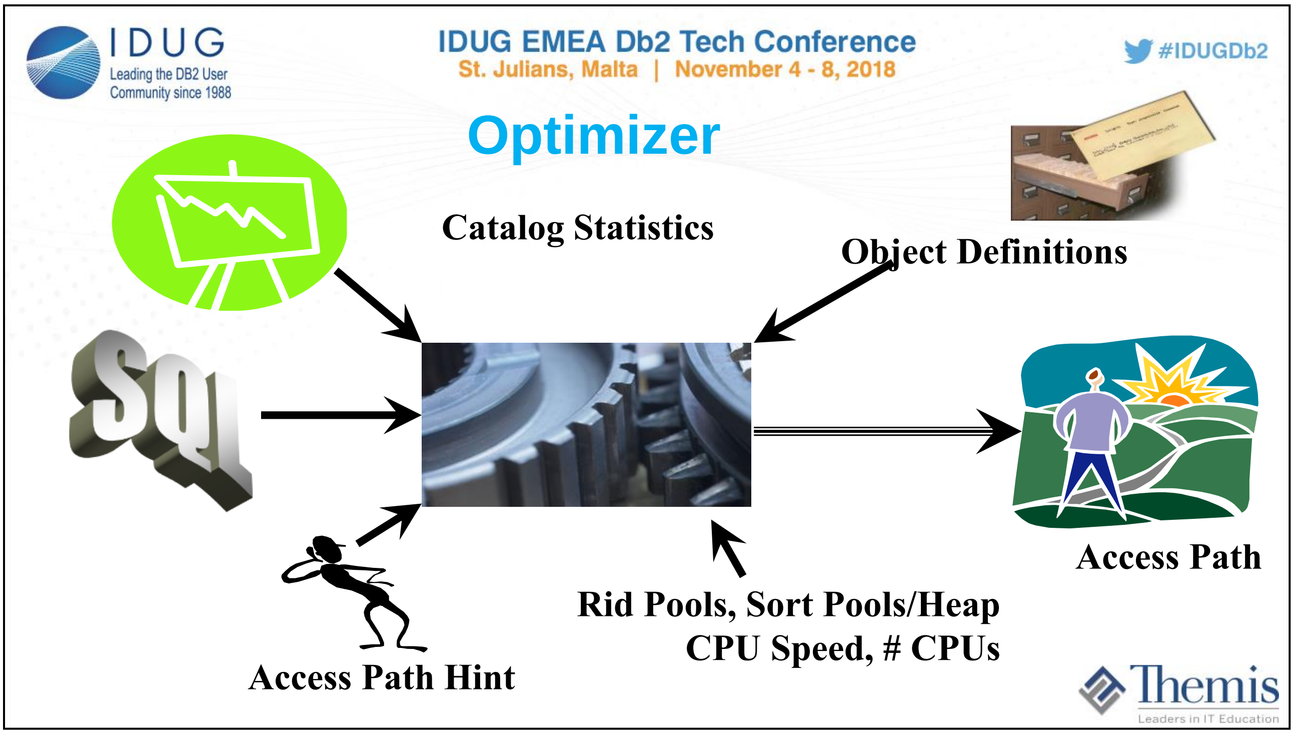
Creating appropriate indexes on columns or groups of columns that are commonly used to identify needed data can significantly reduce the I/O and CPU needed to retrieve a result.

Paying attention to well written predicates are important (Stage 1 vs Stage 2)

Paying attention to any filter factors that are not close to the reality of data being processed.

Knowing the clustering order of the tables is so important. Having tables clustered by primary keys is not always what we want for applications. How is the table queried? What are the SQL statements and joins to a table?

Look at the queries that gather medium to large result sets, not OLTP queries to help determine.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Optimizer

Catalog Statistics

Object Definitions

Access Path

Rid Pools, Sort Pools/Heap
CPU Speed, # CPUs

Access Path Hint

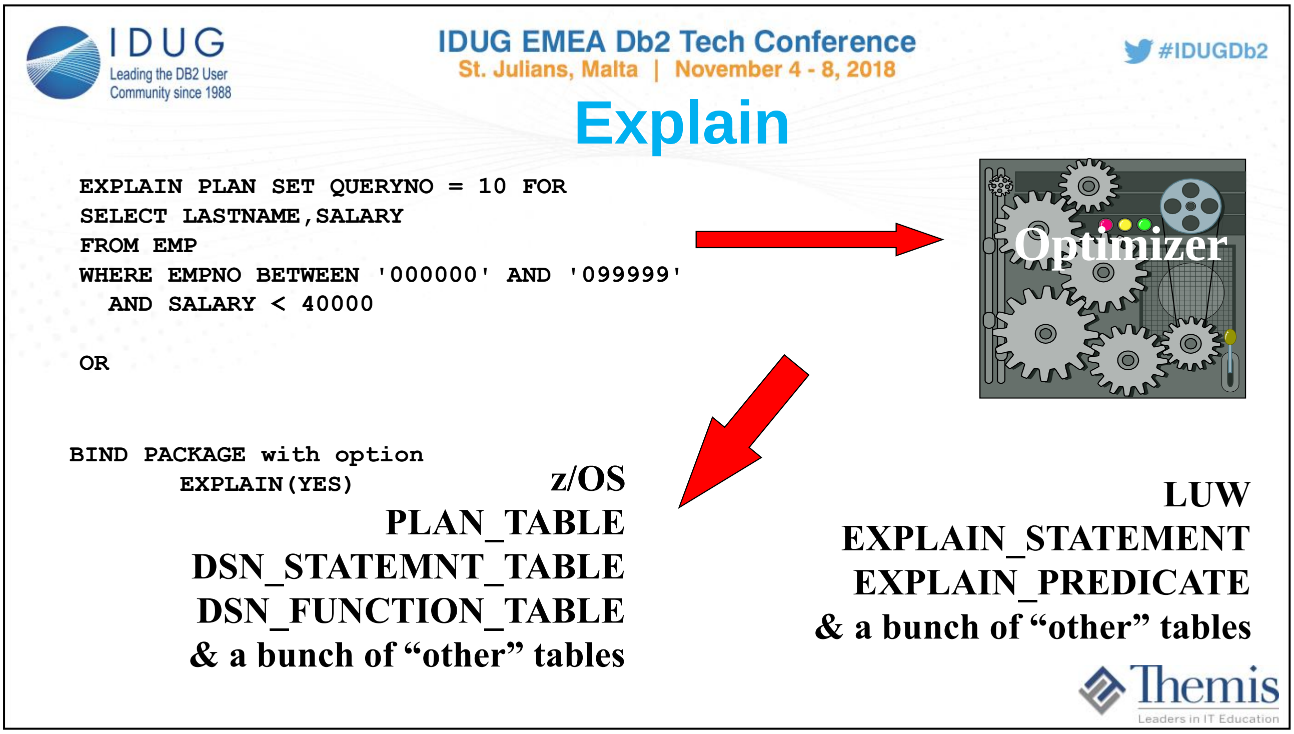
Themis Leaders in IT Education

Through the Data Manipulation Language (DML) the user of a DB2 database supplies the “WHAT”; that is, the data that is needed from the database to satisfy the business requirements. DB2 then uses the information in the DB2 Catalog to resolve “WHERE” the data resides. The DB2 Optimizer is then responsible for determining the all important “HOW” to access the data most efficiently.

Ideally, the user of a relational database is not concerned with how the system accesses data. This is probably true for an end user of DB2, who writes SQL queries quickly for one-time or occasional use. It is less true for developers who write application pro-grams and transactions, some of which will be executed thou-sands of times a day. For these cases, some attention to DB2 access methods can significantly improve performance.
DB2’s access paths can be influenced in four ways:

- By rewriting a query in a more efficient form.
- By creating, altering, or dropping indexes.
- By updating the catalog statistics that DB2 uses to estimate access costs.
- By utilizing Optimizer Hints.

As you can see there are many variables that can affect an access path choice outside of the actual SQL code. It is always best to have a test environment that looks like production (at least data statistics wise) so when queries and packages are eventually promoted to production, there are no last minute surprise/different access paths.



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Explain

```
EXPLAIN PLAN SET QUERYNO = 10 FOR
SELECT LASTNAME,SALARY
FROM EMP
WHERE EMPNO BETWEEN '000000' AND '099999'
  AND SALARY < 40000

OR


BIND PACKAGE with option
        EXPLAIN(YES)
```

z/OS
PLAN_TABLE
DSN_STATEMNT_TABLE
DSN_FUNCTION_TABLE
& a bunch of “other” tables

LUW
EXPLAIN_STATEMENT
EXPLAIN_PREDICATE
& a bunch of “other” tables

The process of asking the DB2 optimizer to describe an access path that was chosen (or will be chosen) for a query is called an *explain.* When we run an explain, the output is placed in DB2 tables that we may then view.

A PLAN_TABLE is a regular DB2 table that holds results of an EXPLAIN. IBM’s *SQL Reference Guide* contains a format for the PLAN TABLE and a description of all the columns. Each user running an explain needs access to a plan table either by owning one directly or through a secondary authid. In DB2 Version 8, aliases may also be used to allow users to share a single set of explain tables.

Although the plan table is required to run explains, there are also several other explain tables which may optionally be created to hold explain data. These extra tables will be populated during an explain if they exist.

The DSN_STATEMNT_TABLE contains information about the total perceived cost of the query being explained. This cost data may be compared for several iterations of refinement for a query to see if it might improve the performance. Costs may also be tracked over time.

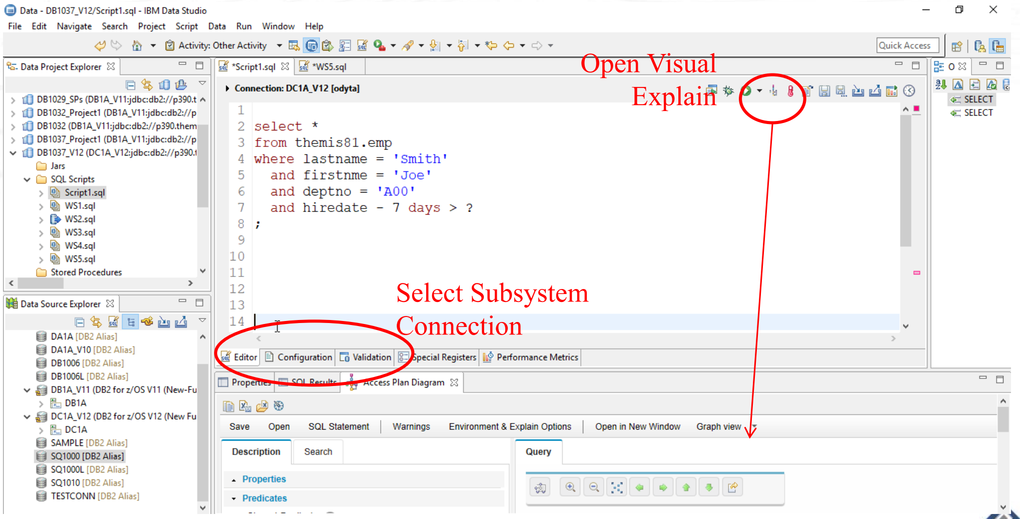
Additionally, as of V8 there are many more explain tables that will be populated if they are created. IBM has recently documented the contents of some of these extra tables. Their primary purpose is to supply additional information for optimization products like V8 Visual Explain, V9 Optimization Service Center (OSC) and Data Studio, and V10 Data Studio.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Explaining Queries



Open Visual Explain

Select Subsystem Connection

Connection: DC1A_V12 [odyta]

```
select *
from themis81.emp
where lastname = 'Smith'
  and firstnme = 'Joe'
  and deptno = 'A00'
  and hiredate - 7 days > ?
;
```

Themis
Leaders in IT Education

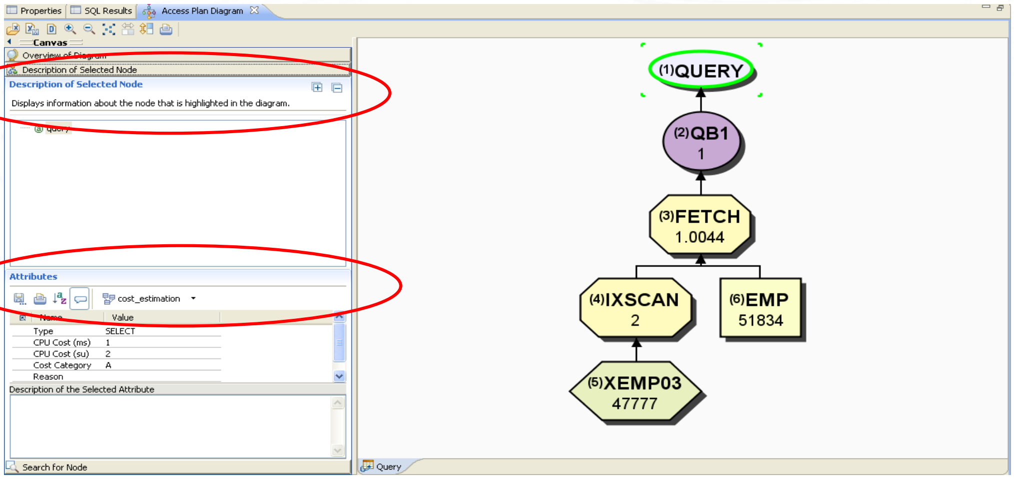
You can run your query from any database connection. Select Configuration for which sub system you want to execute against. You
must be connected to the sub system. In this example 2 sub systems are currently connected to.

You type in your query and click on the ‘Open Visual Explain’ button (top right)

The Visual Explain gets generated (bottom right)

To expand the Visual Explain, click the maximize

z/OS Data Studio Access Path Graphs



Description of Selected Node

Displays information about the node that is highlighted in the diagram.

Attributes

cost_estimation

Name	Value
Type	SELECT
CPU Cost (ms)	1
CPU Cost (su)	2
Cost Category	A
Reason	

Description of the Selected Attribute

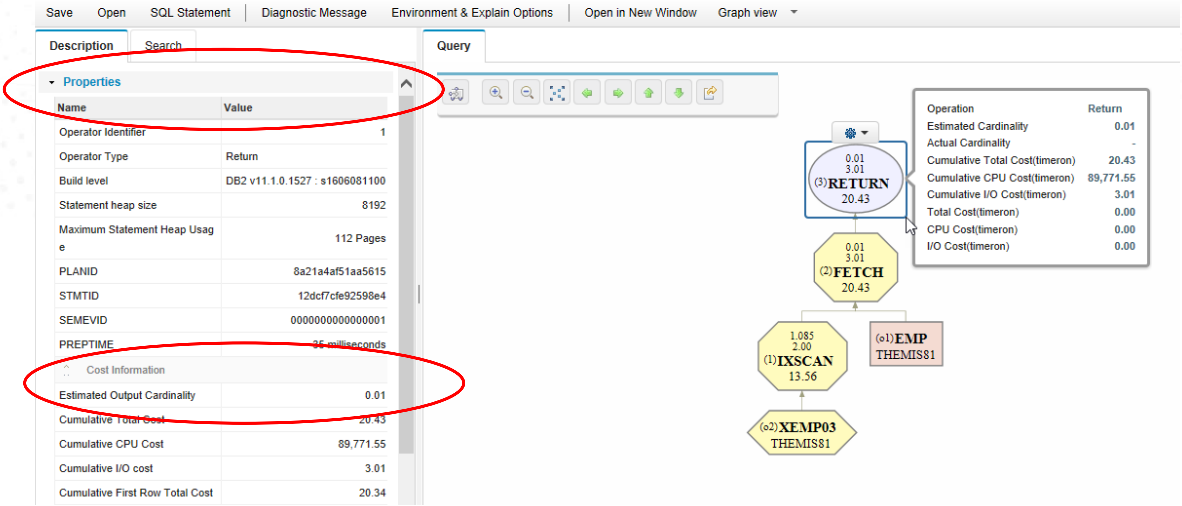
The access path graph shows the explain data for a query. Access path graphs are read left to right, bottom to top. Each of the nodes on the graph represent a source of data or an operation on data as it moves towards the result set. Each node may be clicked to provide details about that node on the left side of the screen.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

LUW Data Studio Access Path Graphs



Name	Value
Operator Identifier	1
Operator Type	Return
Build level	DB2 v11.1.0.1527 : s1606081100
Statement heap size	8192
Maximum Statement Heap Usag e	112 Pages
PLANID	8a21a4af51aa5615
STMTID	12dcf7cfe92598e4
SEMEVID	0000000000000001
PREPTIME	35 milliseconds
Cost Information	
Estimated Output Cardinality	0.01
Cumulative Total Cost	20.43
Cumulative CPU Cost	89,771.55
Cumulative I/O cost	3.01
Cumulative First Row Total Cost	20.34

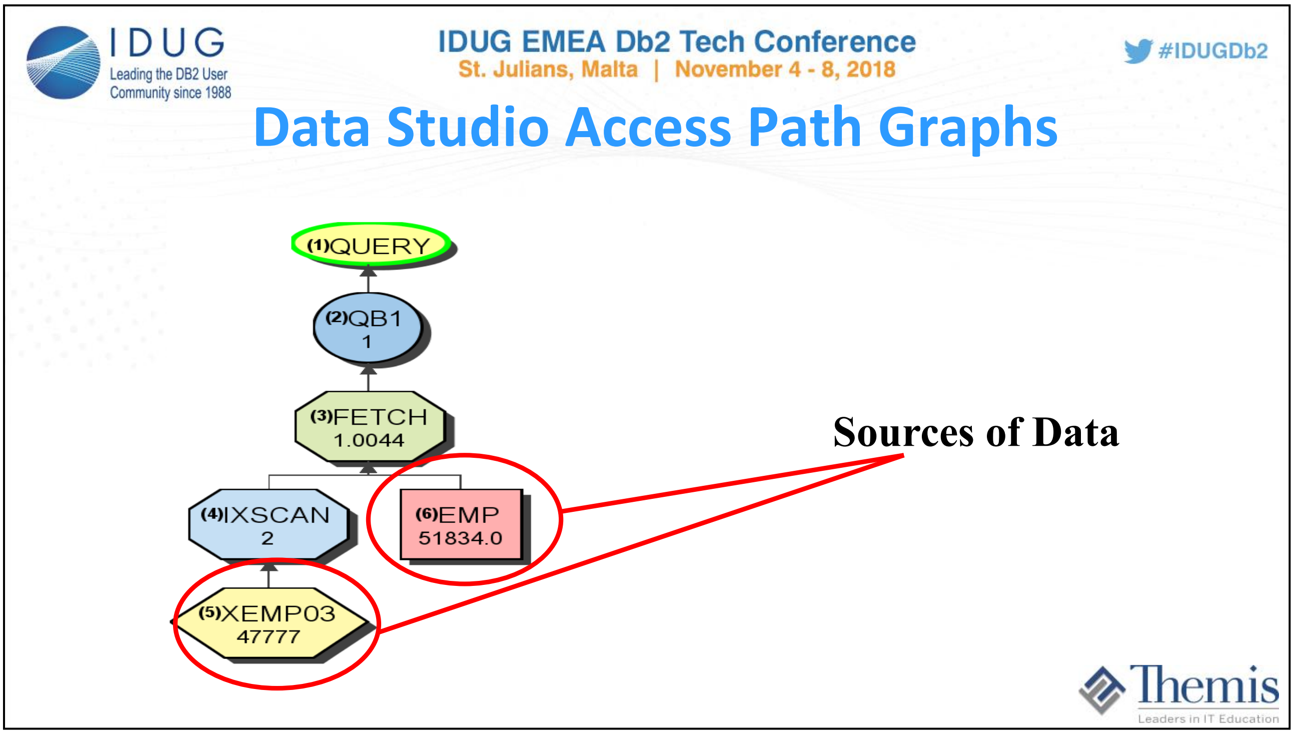
Operation	Return
Estimated Cardinality	0.01
Actual Cardinality	-
Cumulative Total Cost(timeron)	20.43
Cumulative CPU Cost(timeron)	89,771.55
Cumulative I/O Cost(timeron)	3.01
Total Cost(timeron)	0.00
CPU Cost(timeron)	0.00
I/O Cost(timeron)	0.00

Themis
Leaders in IT Education

The is the same query except explained in LUW. Notice:

Costing is in timerons: This is some proprietary algorithm that contains some weighing factors between I/O and CPU.

Bottom numbers in the visual explain nodes are timeron costings. Top numbers are cardinality numbers. For example the IXSCAN
node shows 3 numbers.
- 13.56 for the timeron cost
- 1.085 estimated cardinality of RIDs that meet the criteria
- 2.00 cumulative I/O timeron cost



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Access Path Graphs

Sources of Data

Themis Leaders in IT Education

In this graph, there are two nodes that indicate they are a source of data. Data is accessed by doing an index scan of the XEMP03 index. Once appropriate rows are identified then the data in the EMP table is retrieved using the identifiers from the index.

If one of these nodes is highlighted by clicking on it the catalog data about the object is displayed on the left side of the screen. Statistical information is displayed as well as the timestamp when statistics were last gathered when connected to a Db2 z/OS subsystem.

In Db2 LUW, there are different numbers that show within the nodes. For IXSCAN and FETCH nodes:
Costing is in timerons:
This is some proprietary algorithm that contains some weighing factors between I/O and CPU.
Bottom numbers in the visual explain nodes are timeron costings. Top numbers are cardinality numbers.
For example the IXSCAN
node for this same query in LUW shows 3 numbers.
- 13.56 for the timeron cost
- 1.085 estimated cardinality of RIDs that meet the criteria
- 2.00 cumulative I/O timeron cost

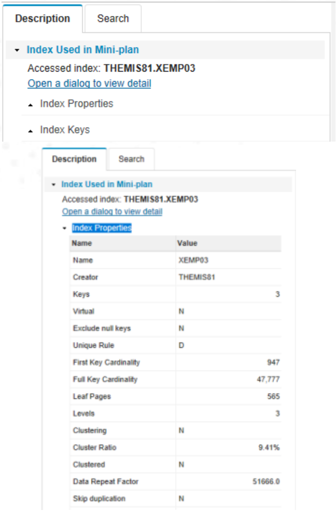
By navigating the tree at the top left of the screen, information may be viewed about the table, tablespace and any other indexes that exist on the referenced table.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Access Path Graphs



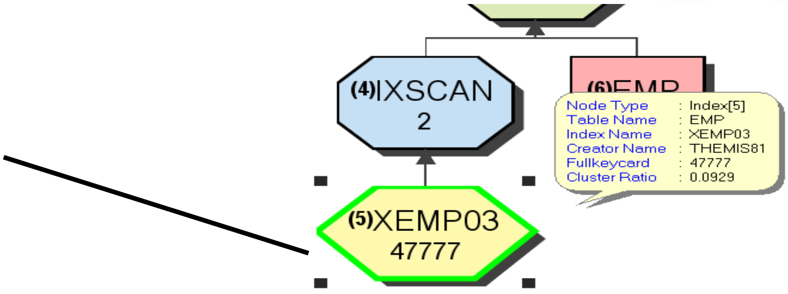
Description | Search

- Index Used in Mini-plan
  Accessed index: **THEMIS81.XEMP03**
  Open a dialog to view detail
  - Index Properties
  - Index Keys

Description | Search

- Index Used in Mini-plan
  Accessed index: **THEMIS81.XEMP03**
  Open a dialog to view detail
  - Index Properties

Name	Value
Name	XEMP03
Creator	THEMIS81
Keys	3
Virtual	N
Exclude null keys	N
Unique Rule	D
First Key Cardinality	947
Full Key Cardinality	47,777
Leaf Pages	565
Levels	3
Clustering	N
Cluster Ratio	9.41%
Clustered	N
Data Repeat Factor	51666.0
Skip duplication	N

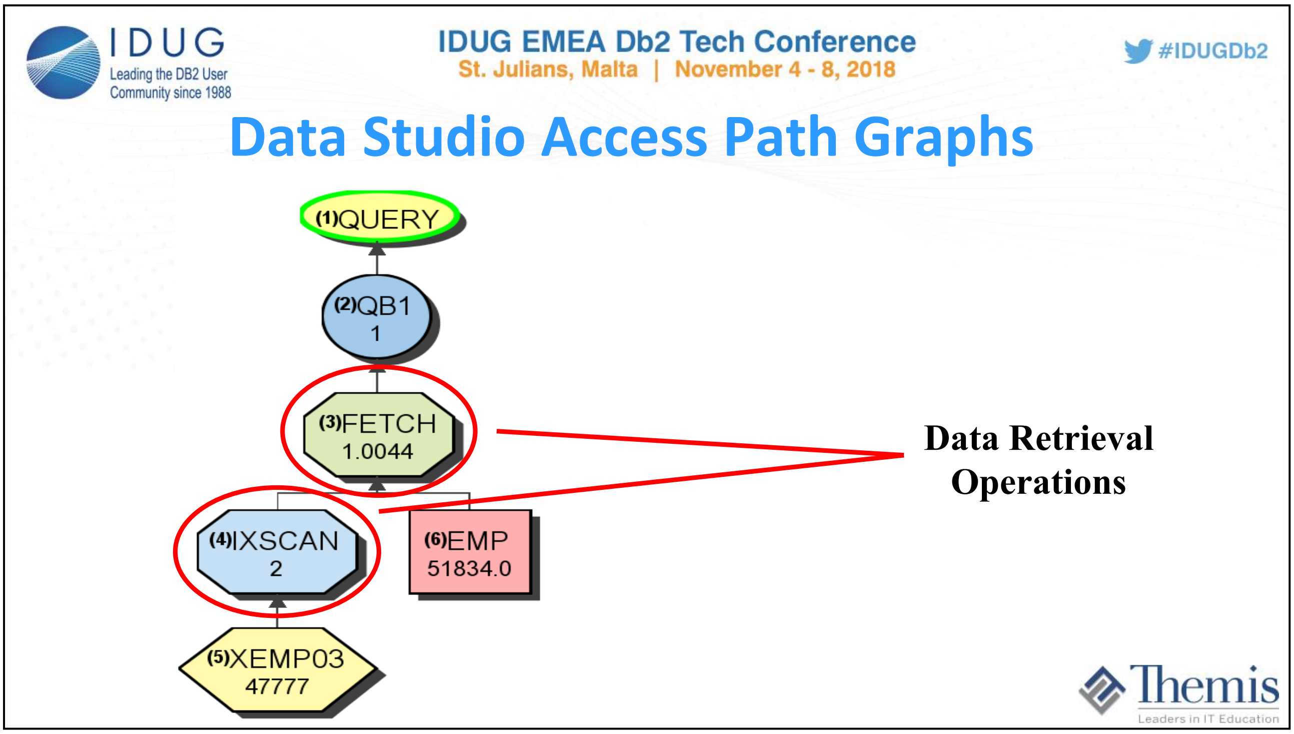


Node Type : Index[5]
Table Name : EMP
Index Name : XEMP03
Creator Name : THEMIS81
Fullkeycard : 47777
Cluster Ratio : 0.0929

Themis
Leaders in IT Education

If one of these nodes is highlighted by clicking on it, the catalog data about the object is displayed on the left side of the screen. Statistical information is displayed as well as the timestamp when statistics were last gathered.

By navigating the tree at the top left of the screen, information may be viewed about the table, tablespace and any other indexes that exist on the referenced table.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Access Path Graphs

Data Retrieval Operations

Themis Leaders in IT Education

The other nodes on the diagram represent operations on the data. If one of these nodes is selected the left side of the screen will show the metrics that DB2 used in determining that this was the appropriate access method. Predicate level data is shown as well as row estimates for how many rows will be passed to the next operation. These estimates may then be compared to reality to determine if the optimizer made a good choice.

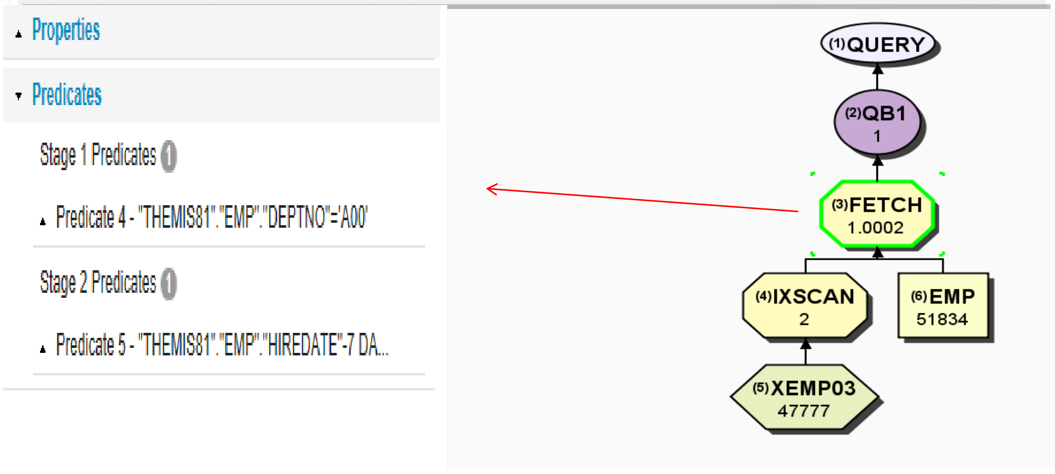
Where do predicates get applied?
1 At the index level (IXSCAN) index predicates
2 At the data level (FETCH) data predicages

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Stage 1, Stage 2 Predicates



- Properties
- Predicates
  - Stage 1 Predicates 1
    - Predicate 4 - "THEMIS81"."EMP"."DEPTNO"='A00'
  - Stage 2 Predicates 1
    - Predicate 5 - "THEMIS81"."EMP"."HIREDATE"-7 DA...

Themis
Leaders in IT Education

Predicates in z/OS get pit into 2 categories, with stage 2 predicates on average about 10% more expensive to evaluate. Many stage 2 predicates can be rewritten as stage 1 predicates, some cannot. It is always important to know if there exists any stage 2 predicates within the query.

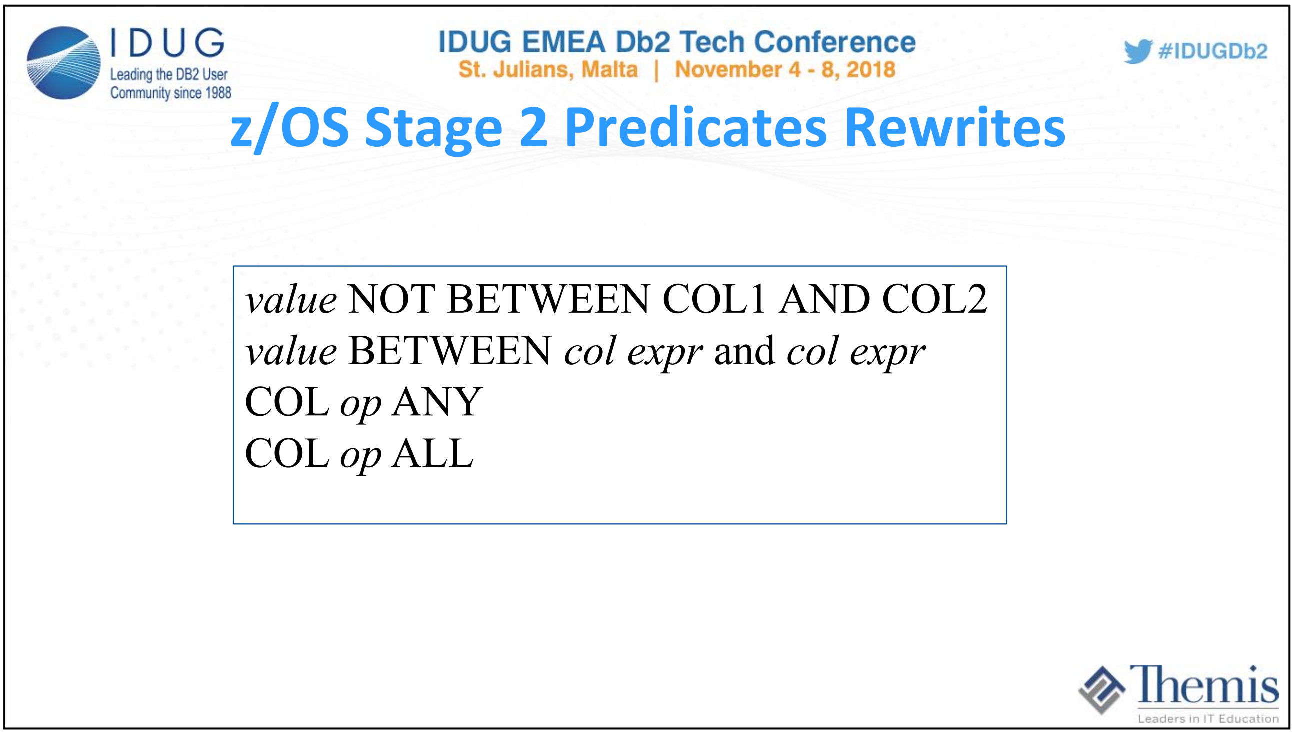
Db2 LUW does not have stage 2 predicates. They do however have residual predicates, being the most expensive of predicates from this platform. Predicate types for LUW:

1) Range delimiting predicates: The start and stop key values in an index search. Evaluated by the index manager.

2) Index sargable: Other index column predicates. These predicates are also evaluated by the index manager.

3) Data sargable: Predicates applied not at the index level by the index manager, but evaluated at the data level through the Data Management Services (DMS). Similar to z/OS stage 1.

4) Residual predicates: Predicates applied last and sometime requiring additional I/O (such as large objects). They also are predicates that often times can be rewritten more efficiently. For example: Subqueries with ANY, ALL, SOME, or IN). These predicates are evaluated by Relational Data Services (RDS) and are the most expensive of the four categories of predicates. Similar to z/OS stage 2.



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Stage 2 Predicates Rewrites

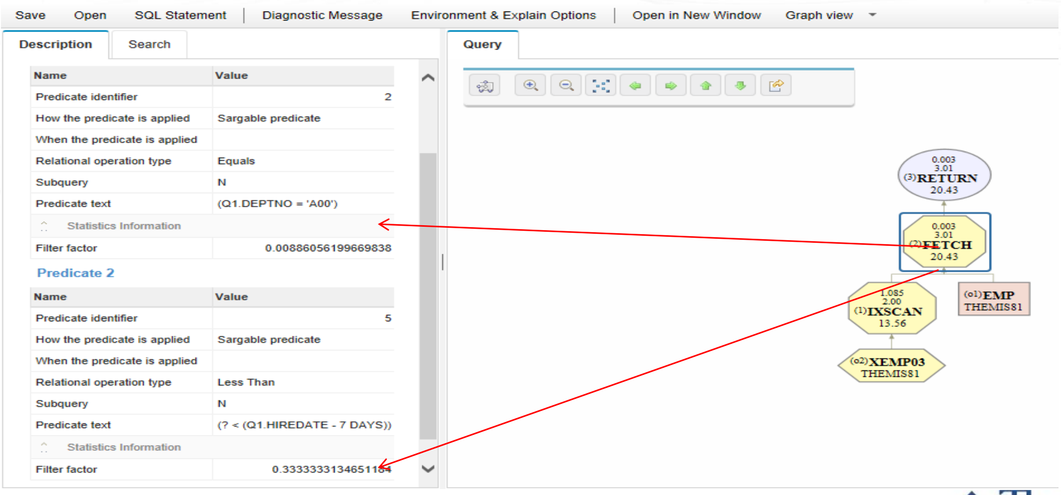
value NOT BETWEEN COL1 AND COL2
value BETWEEN *col expr* and *col expr*
COL *op* ANY
COL *op* ALL

There are far fewer stage 2 predicates than in previous releases of Db2. Here a few of the many that remain that can be controlled and rewritten by developers.

The following will be rewritten by the optmizer:

Where deptno not between value and value

LUW Predicates – Sargeable (similar to S1)



Name	Value
Predicate identifier	2
How the predicate is applied	Sargable predicate
When the predicate is applied	
Relational operation type	Equals
Subquery	N
Predicate text	(Q1.DEPTNO = 'A00')
Statistics Information	
Filter factor	0.00886056199669838

Predicate 2

Name	Value
Predicate identifier	5
How the predicate is applied	Sargable predicate
When the predicate is applied	
Relational operation type	Less Than
Subquery	N
Predicate text	(? < (Q1.HIREDATE - 7 DAYS))
Statistics Information	
Filter factor	0.333333313465118 4

Predicates in z/OS get pit into 2 categories, with stage 2 predicates on average about 10% more expensive to evaluate. Many stage 2 predicates can be rewritten as stage 1 predicates, some cannot. It is always important to know if there exists any stage 2 predicates within the query.

Db2 LUW does not have stage 2 predicates. They do however have residual predicates, being the most expensive of predicates from this platform. Predicate types for LUW:

1) Range delimiting predicates: The start and stop key values in an index search. Evaluated by the index manager.

2) Index sargable: Other index column predicates. These predicates are also evaluated by the index manager.

3) Data sargable: Predicates applied not at the index level by the index manager, but evaluated at the data level through the Data Management Services (DMS). Similar to z/OS stage 1.

4) Residual predicates: Predicates applied last and sometime requiring additional I/O (such as large objects). They also are predicates that often times can be rewritten more efficiently. For example: Subqueries with ANY, ALL, SOME, or IN). These predicates are evaluated by Relational Data Services (RDS) and are the most expensive of the four categories of predicates. Similar to z/OS stage 2.

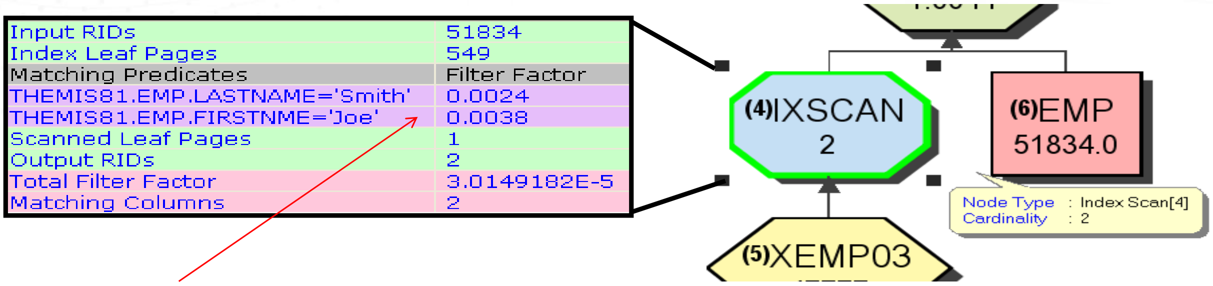
Notice here that the HIREDATE – 7 DAYS is a stage 2 predicate in z/OS, yet Sargable in LUW.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Access Path Graphs



Input RIDs	51834
Index Leaf Pages	549
Matching Predicates	Filter Factor
THEMIS81.EMP.LASTNAME='Smith'	0.0024
THEMIS81.EMP.FIRSTNME='Joe'	0.0038
Scanned Leaf Pages	1
Output RIDs	2
Total Filter Factor	3.0149182E-5
Matching Columns	2

Where do the filter factors come from ?
Do any frequency values exists for LASTNAME data?
Is it defaulting to 1/column cardinality?
FIRSTNME has 260 values. 1/260 = .0038

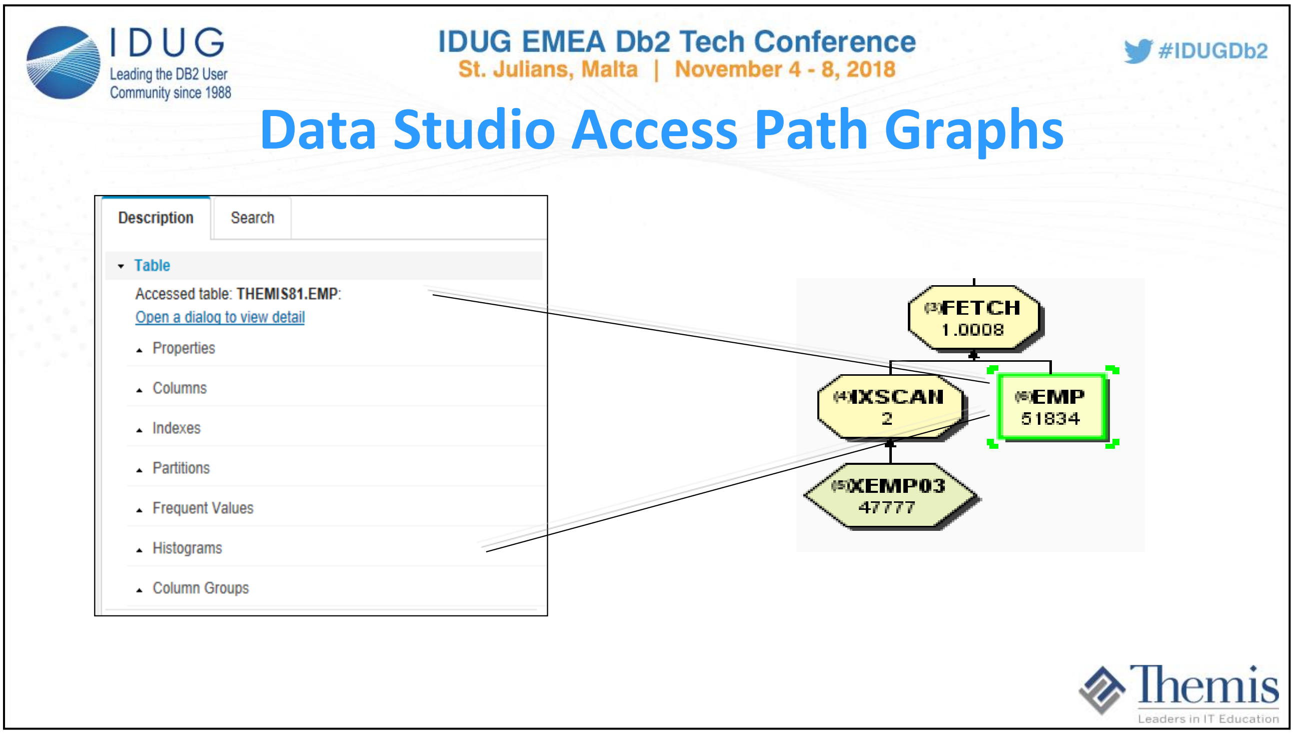
Themis
Leaders in IT Education

This screen shows filter factors for the index predicates. How did the optimizer come up with .0024 for the LASTNAME column?

In the statistics, it shows there are 947 different LASTNAME values. This query is looking for 1 named ‘Smith’. Since LASTNAME is the first column of an index, it automatically calculates the top 10 names in the data. The value ‘Smith’ is one of them and has .0024 percent of the data.

For FIRSTNME = ‘Joe’, all Db2 knows is that there are 260 different first names, and the query is looking for 1 named ‘Joe’. Without any further information of specific first name values, it assumes they are all evenly distributed, and takes 1/260 = .0038.

This 1/cardinality is common when using host variables and/or parameter markers ‘?’.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Data Studio Access Path Graphs

Description | Search

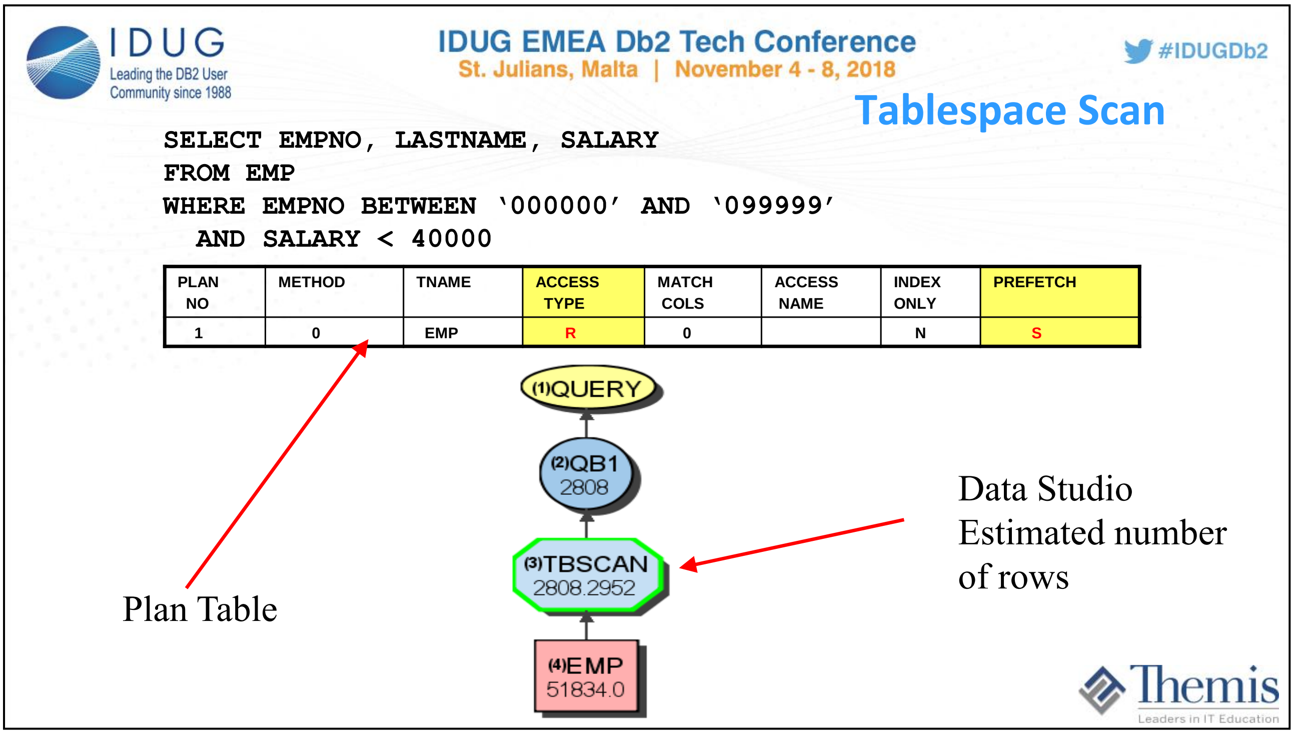
- Table

Accessed table: **THEMIS81.EMP**:
Open a dialog to view detail

- Properties
- Columns
- Indexes
- Partitions
- Frequent Values
- Histograms
- Column Groups

Themis Leaders in IT Education

A great thing about clicking on the table node is the ability to see everything about the table:
-Table level statistics and object information
-Column attributes and statistics
-Indexes and their columns
-Tablespace attributes
- Partitioning Information
-Any special statistics like (Freqval, Histogram, Colgroup)



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Tablespace Scan

```
SELECT EMPNO, LASTNAME, SALARY
FROM EMP
WHERE EMPNO BETWEEN '000000' AND '099999'
   AND SALARY < 40000
```

PLAN NO	METHOD	TNAME	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	PREFETCH
1	0	EMP	R	0		N	S

Plan Table

Data Studio Estimated number of rows

Themis Leaders in IT Education

Tablespace scans are illustrated through EXPLAIN by ACCESSTYPE = R and PREFETCH = S.

The first query on the previous page illustrates a tablespace scan. The query has a predicate, however there are no matching indexes on the HIREDATE column.

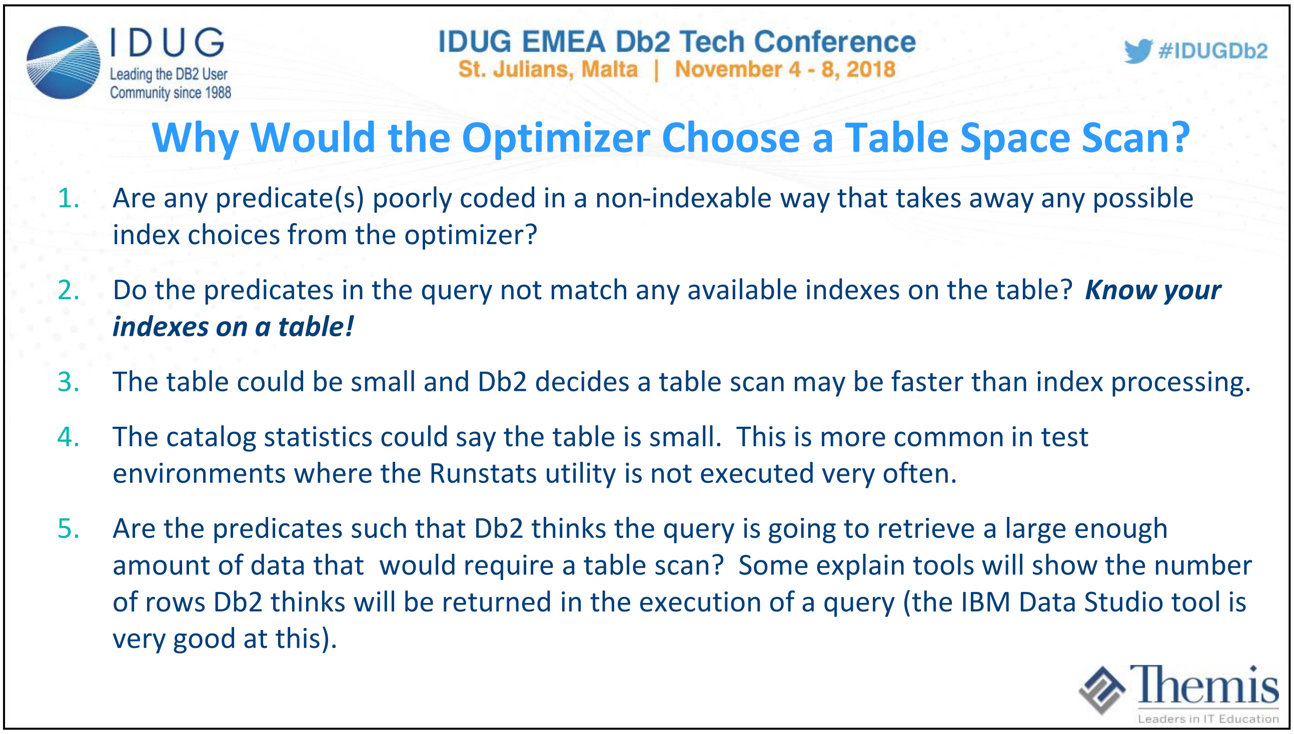
Through the PLAN_TABLE output, there is no way to differentiate a segmented from a non-segmented tablespace scan. A table-space must have been created as segmented in order to benefit from the performance advantages. To change from non-segmented to segmented after a table has been created, it must be dropped and recreated.

When Tablespace Scans are Appropriate

Tablespace scan access is selected by DB2 typically when;

- ♦ A matching index scan is not possible because there are no indexes or there are no predicates to that match the index columns.
- ♦ A high percentage of the rows in the table qualify.
- ♦ The indexes that have matching predicates have low cluster ratios, making them efficient only when a small number of rows qualify.
- ♦ The table is small, or the statistics say the table is small

The Visual Explain access path graph for a tablespace scan is also shown. Notice that Visual Explain gives estimates of how many rows will remain at each level.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

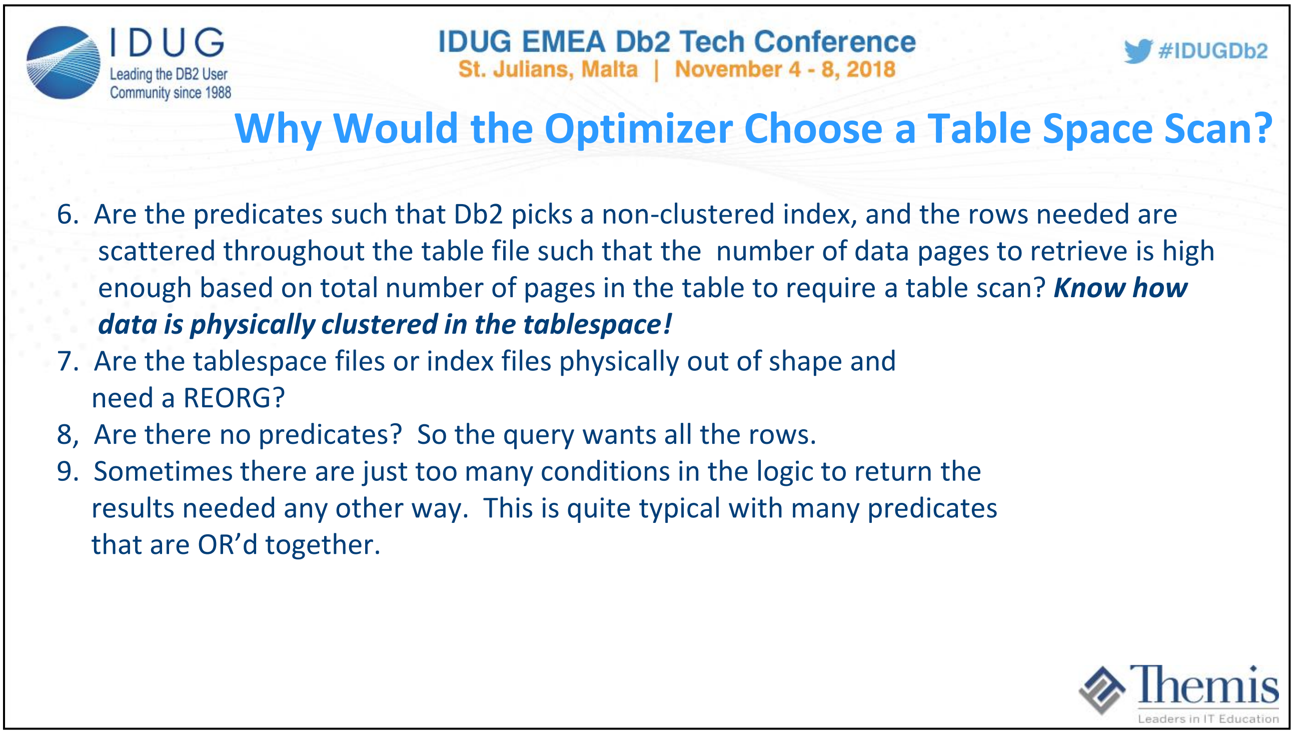
Why Would the Optimizer Choose a Table Space Scan?

1. Are any predicate(s) poorly coded in a non-indexable way that takes away any possible index choices from the optimizer?
2. Do the predicates in the query not match any available indexes on the table? ***Know your indexes on a table!***
3. The table could be small and Db2 decides a table scan may be faster than index processing.
4. The catalog statistics could say the table is small. This is more common in test environments where the Runstats utility is not executed very often.
5. Are the predicates such that Db2 thinks the query is going to retrieve a large enough amount of data that would require a table scan? Some explain tools will show the number of rows Db2 thinks will be returned in the execution of a query (the IBM Data Studio tool is very good at this).

Themis Leaders in IT Education

Developers should:

- Know their data (cardinalities, columns with uneven distributions of data, etc.)
- Know the indexes on tables involved in their queries
- Know the clustering order of data in their tables
- Know the partitioning of data in their tables

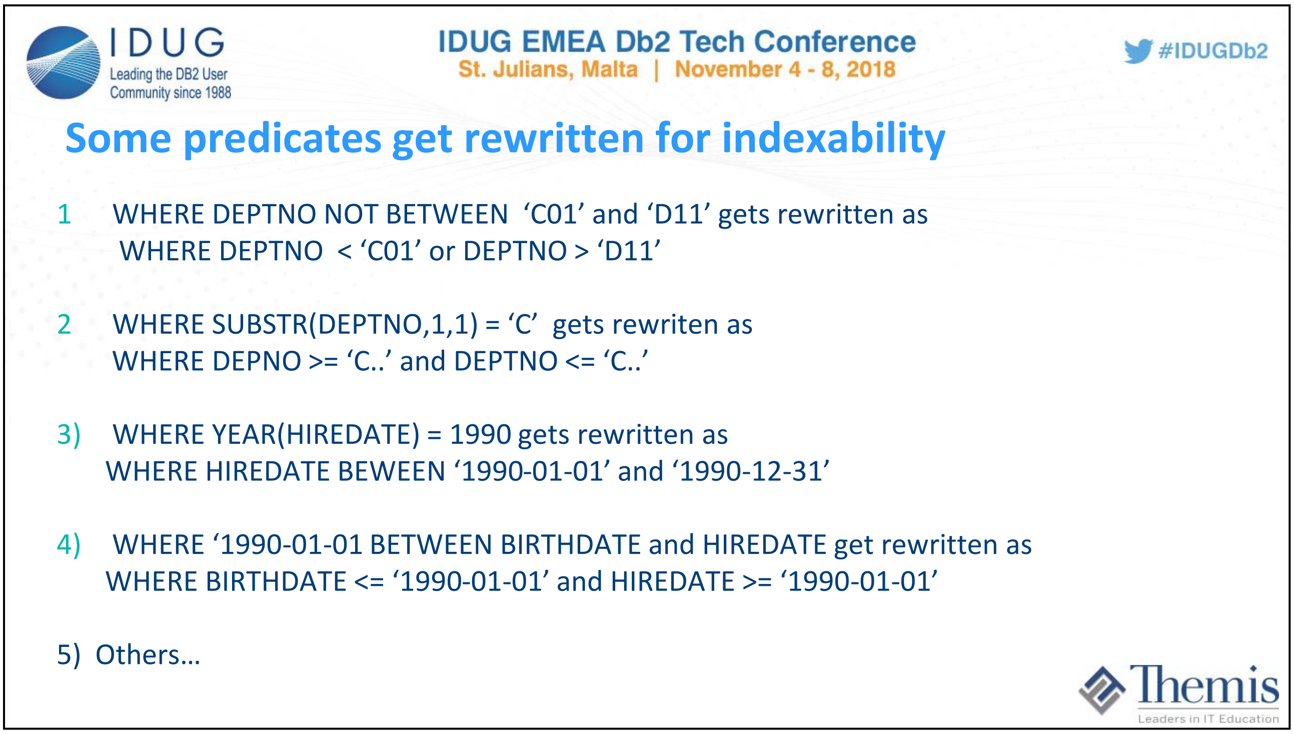


Why Would the Optimizer Choose a Table Space Scan?

6. Are the predicates such that Db2 picks a non-clustered index, and the rows needed are scattered throughout the table file such that the number of data pages to retrieve is high enough based on total number of pages in the table to require a table scan? ***Know how data is physically clustered in the tablespace!***
7. Are the tablespace files or index files physically out of shape and need a REORG?

8, Are there no predicates? So the query wants all the rows.

9. Sometimes there are just too many conditions in the logic to return the results needed any other way. This is quite typical with many predicates that are OR’d together.



Some predicates get rewritten for indexability

1 WHERE DEPTNO NOT BETWEEN ‘C01’ and ‘D11’ gets rewritten as
WHERE DEPTNO < ‘C01’ or DEPTNO > ‘D11’

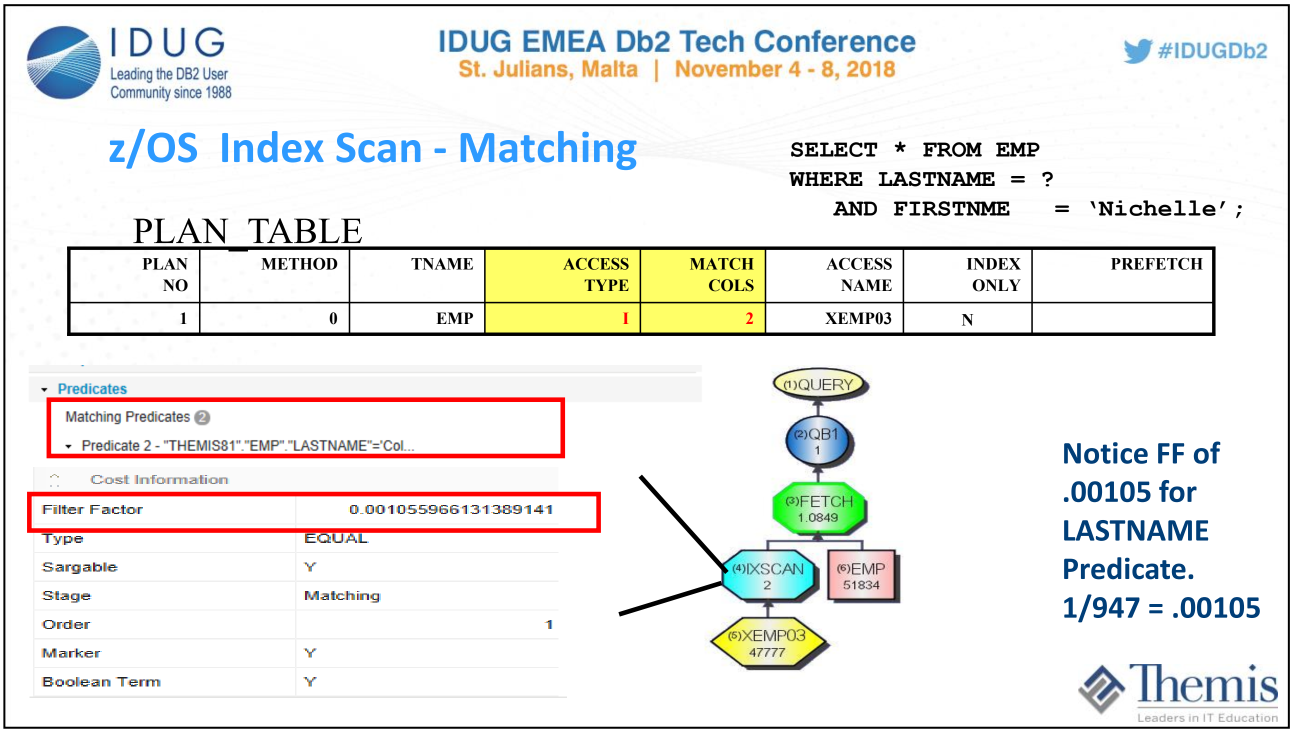
2 WHERE SUBSTR(DEPTNO,1,1) = ‘C’ gets rewriten as
WHERE DEPNO >= ‘C..’ and DEPTNO <= ‘C..’

3) WHERE YEAR(HIREDATE) = 1990 gets rewritten as
WHERE HIREDATE BEWEEN ‘1990-01-01’ and ‘1990-12-31’

4) WHERE ‘1990-01-01 BETWEEN BIRTHDATE and HIREDATE get rewritten as
WHERE BIRTHDATE <= ‘1990-01-01’ and HIREDATE >= ‘1990-01-01’

5) Others…

Themis
Leaders in IT Education



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Index Scan - Matching

```
SELECT * FROM EMP
WHERE LASTNAME = ?
  AND FIRSTNME    = 'Nichelle';
```

PLAN_TABLE

PLAN NO	METHOD	TNAME	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	PREFETCH
1	0	EMP	I	2	XEMP03	N	

Predicates

Matching Predicates 2

Predicate 2 - "THEMIS81"."EMP"."LASTNAME"='Col...

Cost Information

Filter Factor	0.001055966131389141
Type	EQUAL
Sargable	Y
Stage	Matching
Order	1
Marker	Y
Boolean Term	Y

Notice FF of .00105 for LASTNAME Predicate. 1/947 = .00105

Themis Leaders in IT Education

Matching Index scans are depicted in the PLAN_TABLE by ACCESSTYPE = I, I1, N, or MX and MATCHCOLS > 0.

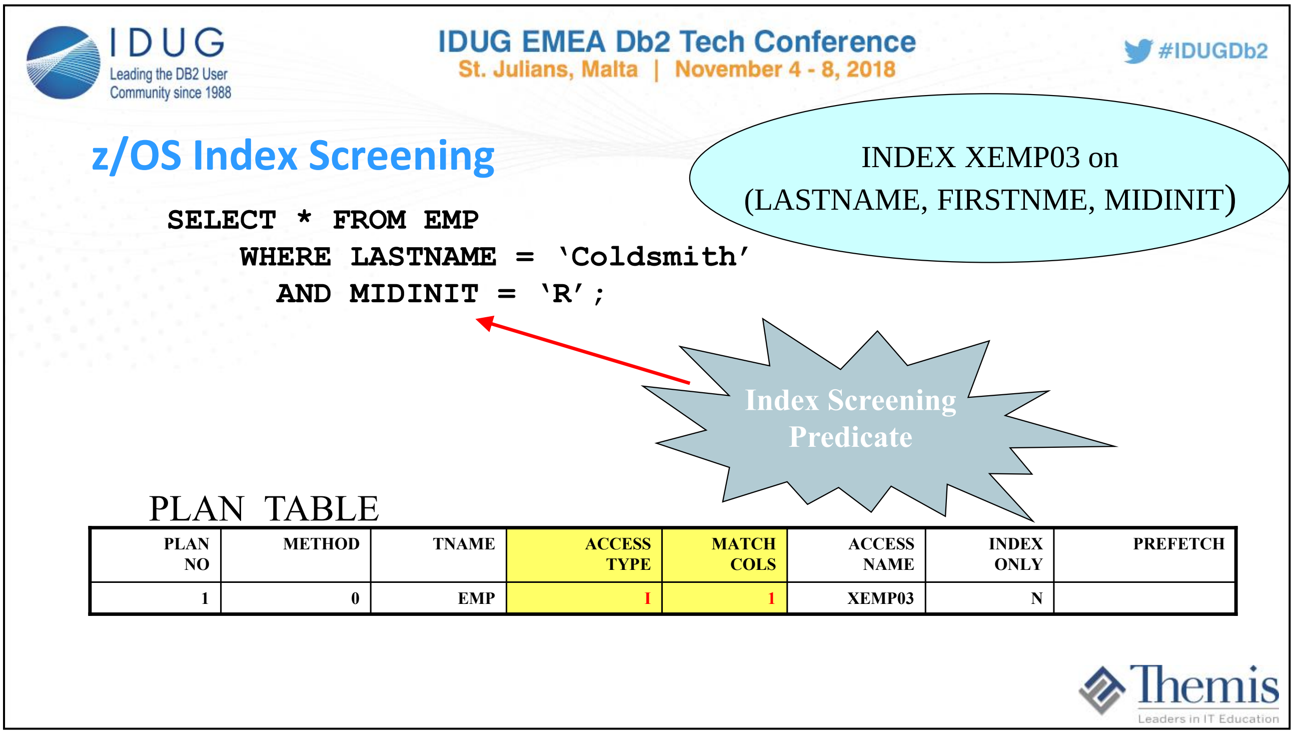
For a Matching Index Scan, DB2 has determined that the query uses predicates that match index columns. In general, the matching predicates on the leading index columns are equal or IN predicates. The predicate that matches the final index column can be an equal, IN, or a range predicate (<, <=, >, >=, LIKE, or BETWEEN).

The query on the previous page illustrates matching index access. Assume the table EMP has an index; XEMP03 on (LASTNAME, FIRSTNME, MIDINIT). The index XEMP03 is the chosen access path for this query, with MATCHCOLS = 2. There are two equal predicates on the first two columns of the index.

In Visual Explain the IXSCAN detail shows which predicates were used to match columns along with their filter factors. Row estimates are computed and displayed based on the available statistics for the table and index. This slide only shows the first matching predicate.

The value of MATCHCOLS is used to determine the number of columns DB2 can match to predicates in the query. Typically, index access will be more efficient the greater the number of matching columns.

Effort placed on proper index design can have a huge return on investment in terms of the efficiency of DB2's ability to ut*i*lize matching indexes to query predicates.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Index Screening

```
SELECT * FROM EMP
      WHERE LASTNAME = 'Coldsmith'
         AND MIDINIT = 'R';
```

INDEX XEMP03 on
(LASTNAME, FIRSTNME, MIDINIT)

Index Screening Predicate

PLAN TABLE

PLAN NO	METHOD	TNAME	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	PREFETCH
1	0	EMP	I	1	XEMP03	N	

Themis Leaders in IT Education

- **Index Screening**

Index screening predicates are specified on index key columns, but are not part of the matching columns used to scan the index structure. These screening predicates improve index access by reducing the number of rows that qualify while searching the index.

Assume the table EMP has an index; XEMP03 on (LASTNAME, FIRSTNME, MIDINIT);

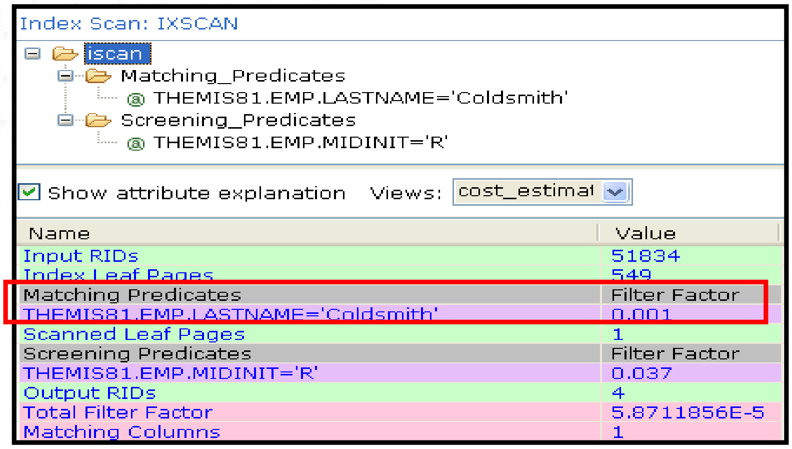
The query on the previous page illustrates DB2s ability to use one of the two predicates matching against the index, i.e. with MATCHCOLS = 1. Once DB2 determines that a symbolic key entry matches on the predicate LASTNAME = ?, the predicate MIDINIT = ? can be applied during the index scan to further qualify rows. This is the process known as index screening. If a row meets the criteria of these screening predicates, the row will be retrieved. Once the data row has been retrieved, predicates for columns not in the index can be applied.

- ***When Index Screening is used***

♦ When there are predicates available to apply against columns in the index to further qualify rows.

♦ The PLAN_TABLE does not directly tell when an index is screened. However, if the MATCHCOLS is less than the number of in-dex key columns, this indicates index screening is possible. Visual Explain does flag predicates where index screening is used.

z/OS Index Screening (cont)

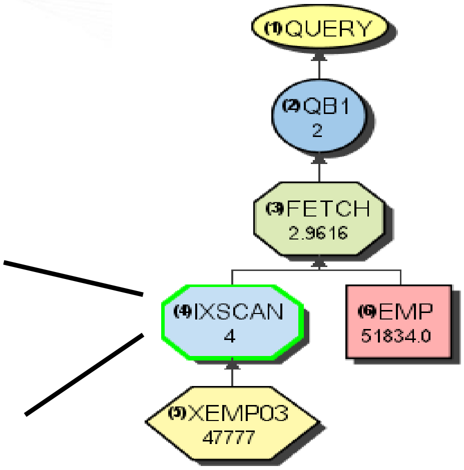


Index Scan: IXSCAN

- iscan
  - Matching_Predicates
    - THEMIS81.EMP.LASTNAME='Coldsmith'
  - Screening_Predicates
    - THEMIS81.EMP.MIDINIT='R'

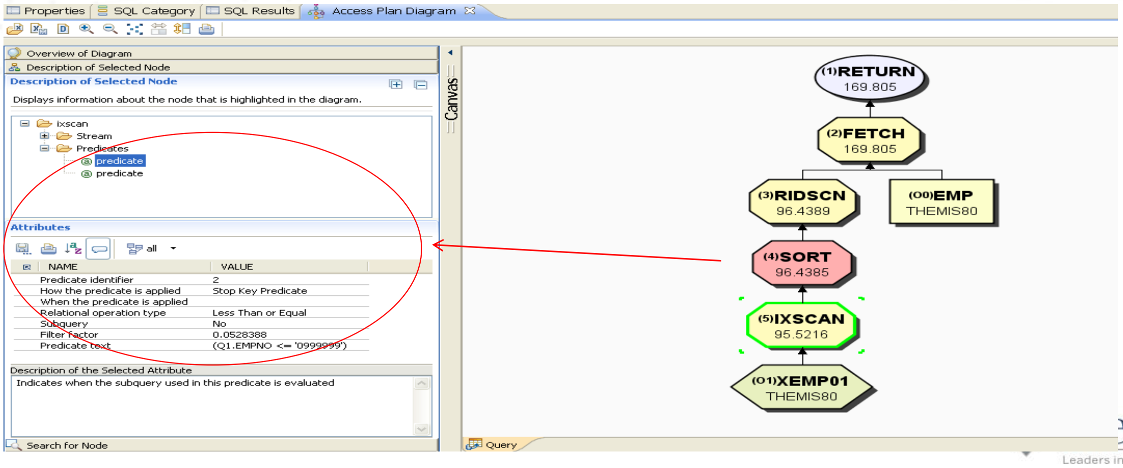
☑ Show attribute explanation Views: cost_estimat

Name	Value
Input RIDs	51834
Index Leaf Pages	549
Matching Predicates	Filter Factor
THEMIS81.EMP.LASTNAME='Coldsmith'	0.001
Scanned Leaf Pages	1
Screening Predicates	Filter Factor
THEMIS81.EMP.MIDINIT='R'	0.037
Output RIDs	4
Total Filter Factor	5.8711856E-5
Matching Columns	1



LUW Index Scan - Matching (Start/Stop Keys)

```
SELECT * FROM EMP
WHERE EMPNO BETWEEN '000000' and '099999'
```



Description of Selected Node

Displays information about the node that is highlighted in the diagram.

- ixscan
  - Stream
  - Predicates
    - predicate
    - predicate

Attributes

all

NAME	VALUE
Predicate identifier	2
How the predicate is applied	Stop Key Predicate
When the predicate is applied	
Relational operation type	Less Than or Equal
Subquery	No
Filter factor	0.0528388
Predicate text	(Q1.EMPNO <= '0999999')

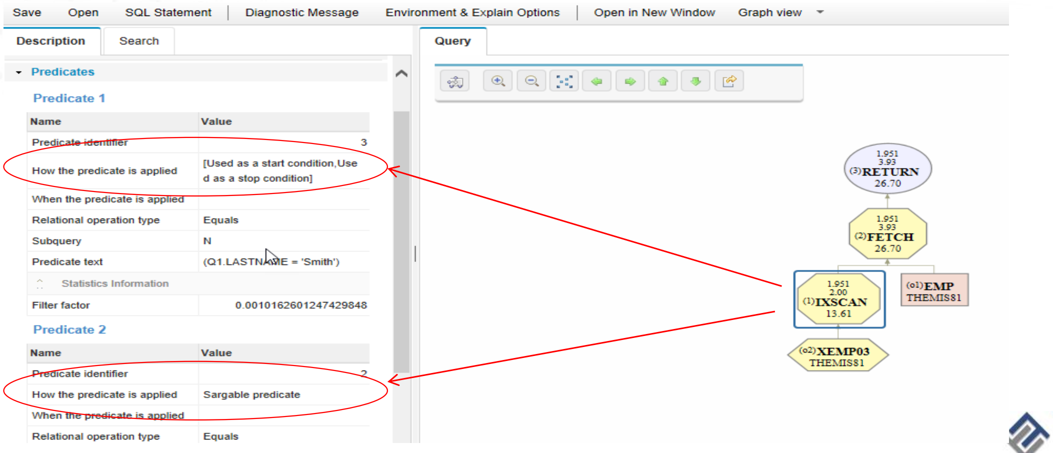
Description of the Selected Attribute

Indicates when the subquery used in this predicate is evaluated

You know when you have matching index access by seeing start and stop keys in the index predicate information

LUW Index Screening

```
SELECT * FROM EMP
WHERE LASTNAME = 'Smith'
   AND MIDINIT = 'R'
```



Predicates

Predicate 1

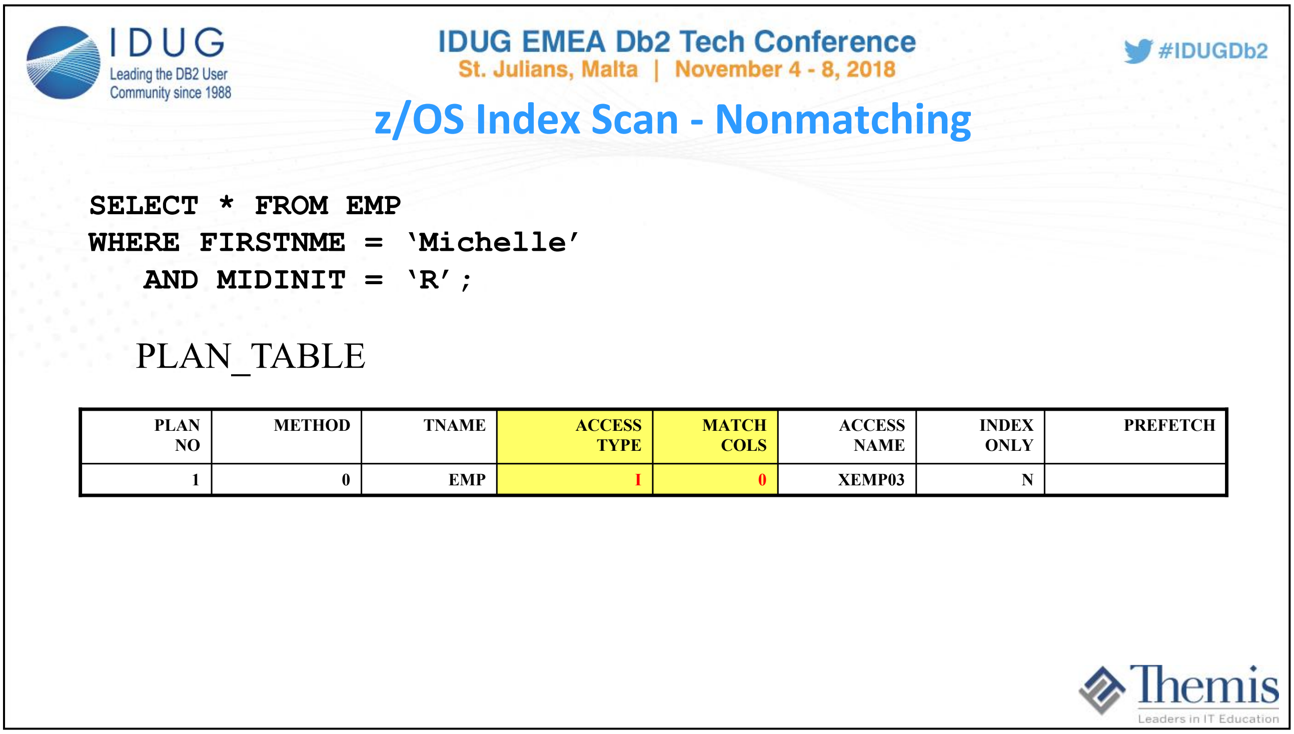
Name	Value
Predicate identifier	3
How the predicate is applied	[Used as a start condition,Used as a stop condition]
When the predicate is applied	
Relational operation type	Equals
Subquery	N
Predicate text	(Q1.LASTNAME = 'Smith')
Statistics Information	
Filter factor	0.0010162601247429848

Predicate 2

Name	Value
Predicate identifier	2
How the predicate is applied	Sargable predicate
When the predicate is applied	
Relational operation type	Equals

Themis
Leaders in IT Education

Index screening in LUW shows a Start position (matching of 1 in this example) and then the other index predicate as a Sargable predicate.



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Index Scan - Nonmatching

```
SELECT * FROM EMP
WHERE FIRSTNME = 'Michelle'
   AND MIDINIT = 'R';
```

PLAN_TABLE

PLAN NO	METHOD	TNAME	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	PREFETCH
1	0	EMP	I	0	XEMP03	N	

Nonmatching Index scans are described through EXPLAIN by ACCESSTYPE = I and MATCHCOLS = 0.

In Visual Explain it is possible to see which columns are used as screening predicates in a nonmatching index scan. Notice that MATCHCOLS shows up as zero in both the PLAN_TABLE and Visual Explain.

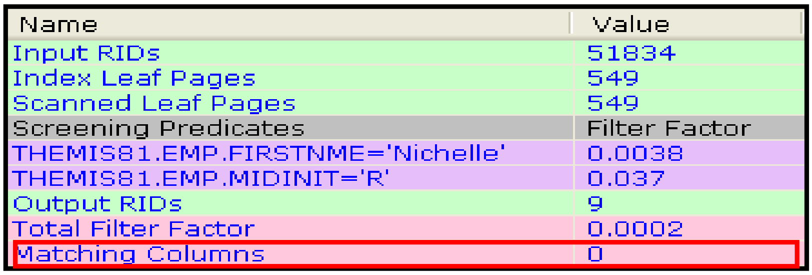
When Nonmatching Index Access is used

Because there is little or no filtering, a nonmatching index scan is used. Sometimes called an index scan:

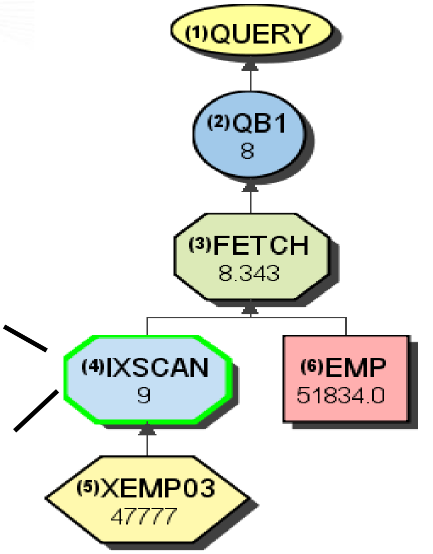
♦ When index screening is provided. In this case not all the data pages are accessed, but all index pages are accessed. Then only those data pages that DB2 has determined qualified based on the scanning and evaluating of the index predicates.

z/OS Index Scan - Nonmatching

```
SELECT * FROM EMP
WHERE FIRSTNME = ‘Michelle’
AND MIDINIT = ‘R’;
```



Name	Value
Input RIDs	51834
Index Leaf Pages	549
Scanned Leaf Pages	549
Screening Predicates	Filter Factor
THEMIS81.EMP.FIRSTNME='Nichelle'	0.0038
THEMIS81.EMP.MIDINIT='R'	0.037
Output RIDs	9
Total Filter Factor	0.0002
Matching Columns	0



Note that on the visual Explain, a node ‘IXSCAN’ shows whether there is matching or not. So it is very important to always click on the node to evaluate any matching and/or screening.

0 matching should be an indicator for further analysis to determine ‘Why’ no matching.

0 matching means the whole index file will be scanned.

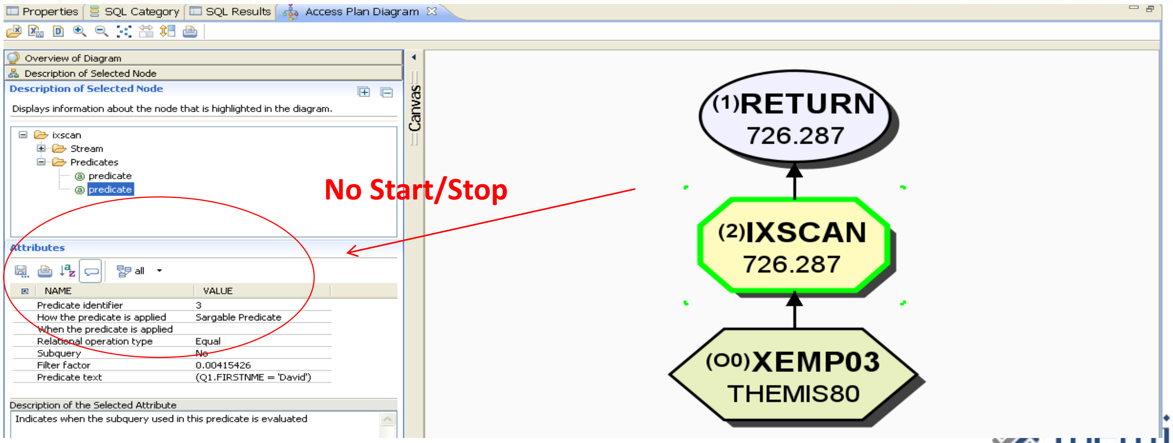
IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

LUW Index Scan – Non Matching (Sargable)

```
SELECT LASTNAME* FROM EMP
WHERE FIRSTNME = ‘David’ and MIDINIT = ‘A’
```



Displays information about the node that is highlighted in the diagram.

NAME	VALUE
Predicate identifier	3
How the predicate is applied	Sargable Predicate
When the predicate is applied	
Relational operation type	Equal
Subquery	No
Filter factor	0.00415426
Predicate text	(Q1.FIRSTNME = 'David')

Description of the Selected Attribute

Indicates when the subquery used in this predicate is evaluated

No Start/Stop

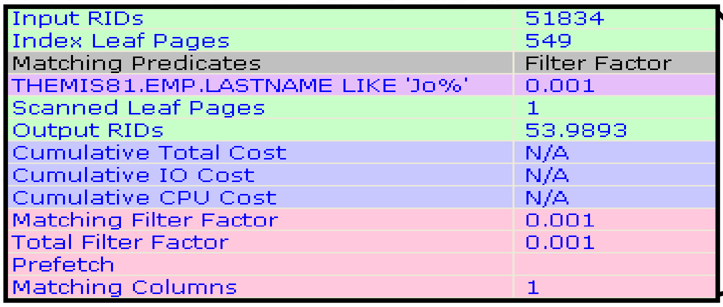
When you do not code the leading column of the index, and DB2 still chooses to use the index, the other Predicates show up as non starting and stopping keys. In this case SARGEABLE predicates against the index column(s).

IDUG
Leading the DB2 User Community since 1988

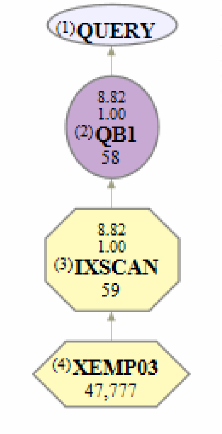
IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Index Only Access



Input RIDs	51834
Index Leaf Pages	549
Matching Predicates	Filter Factor
THEMIS81.EMP.LASTNAME LIKE 'Jo%'	0.001
Scanned Leaf Pages	1
Output RIDs	53.9893
Cumulative Total Cost	N/A
Cumulative IO Cost	N/A
Cumulative CPU Cost	N/A
Matching Filter Factor	0.001
Total Filter Factor	0.001
Prefetch	
Matching Columns	1



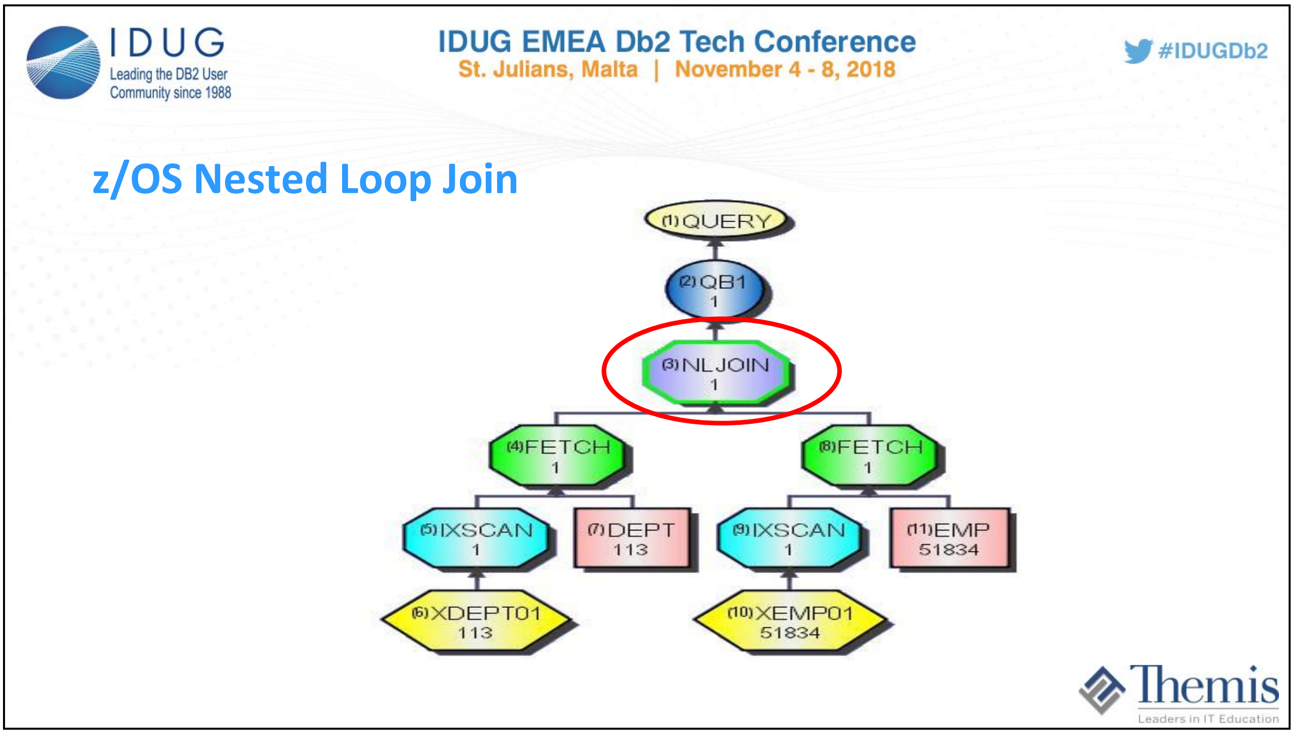
```
SELECT LASTNAME, FIRSTNME, MIDINIT
FROM EMP
WHERE LASTNAME LIKE 'Jo%'
```

Themis
Leaders in IT Education

If all the columns needed for a particular table in a query are available in an index, the optimizer may be able to qualify and retrieve the columns from the index without going to the tablespace at all. This is called *index only access* and may provide a significant performance improvement, particularly when many rows need to be evaluated using a non-clustered index.

In this example only 1 column in XEMP03 is being used to qualify rows, but placing the FIRSTNME and MIDINIT columns in the index provides significant benefit to this query.

Note that there is no table node for the EMP table in this diagram since only the index is used.



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

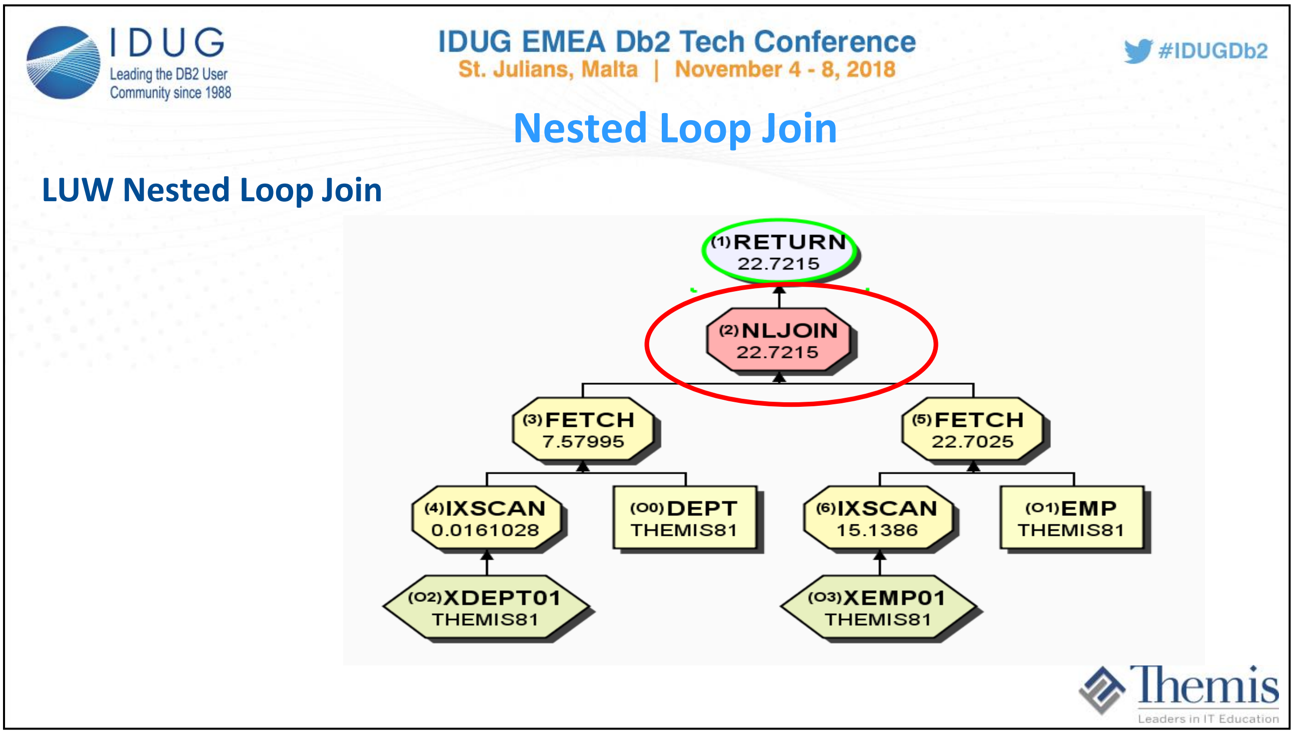
#IDUGDb2

z/OS Nested Loop Join

Themis
Leaders in IT Education

A nested loop join is described in the PLAN_TABLE through METHOD = 1. An inner join is described in the PLAN_TABLE through JOIN_TYPE = ‘ ‘. An outer join is described in the PLAN_TABLE through JOIN_TYPE <> ‘ ‘.

In Visual Explain there will be a join node labeled NLJOIN to describe a nested loop join. In the example shown the optimizer estimates that only 1 row will qualify from the department table and match to only 1 row on the employee table. This likely explains why this join method was chosen.



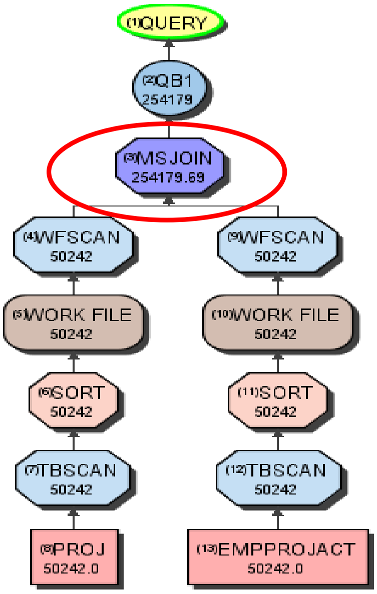
Nested Loop Join

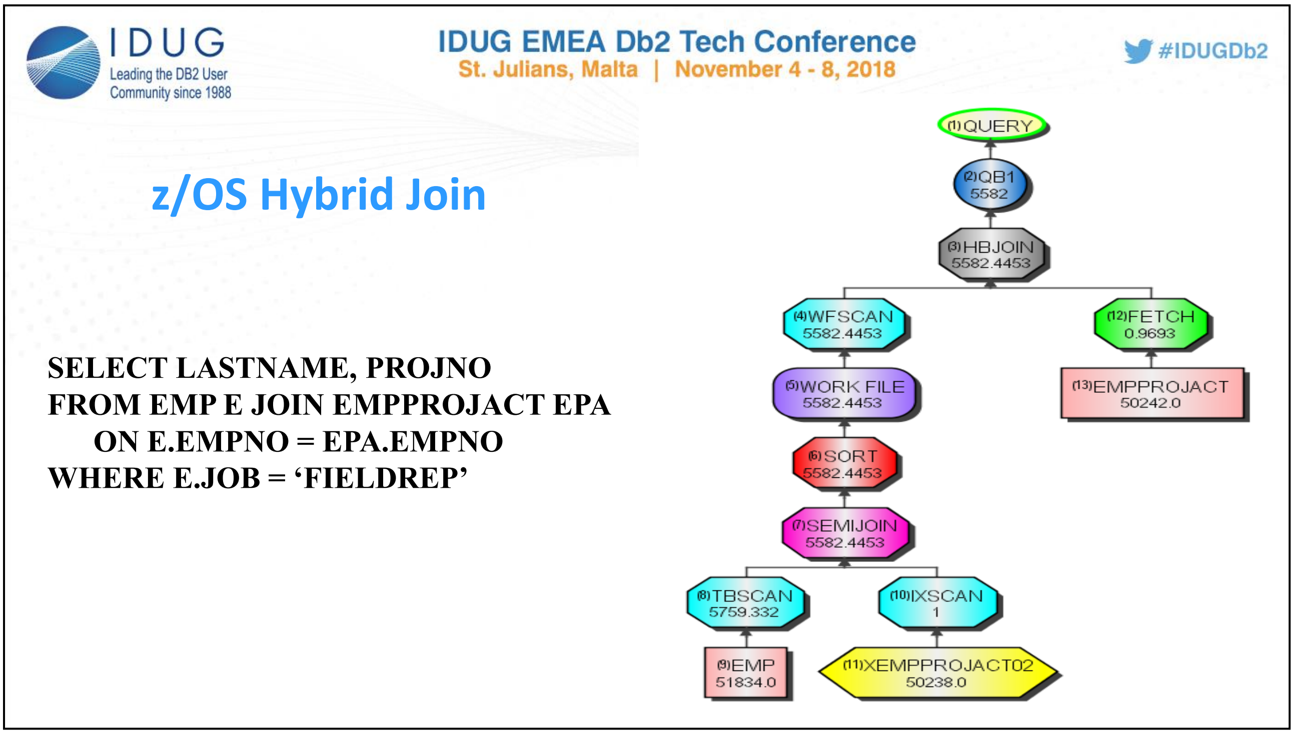
LUW Nested Loop Join

A nested loop join is described in the PLAN_TABLE through METHOD = 1. An inner join is described in the PLAN_TABLE through JOIN_TYPE = ‘ ‘. An outer join is described in the PLAN_TABLE through JOIN_TYPE <> ‘ ‘.

In Visual Explain there will be a join node labeled NLJOIN to describe a nested loop join. In the example shown the optimizer estimates that only 1 row will qualify from the department table and match to only 1 row on the employee table. This likely explains why this join method was chosen.

z/OS Merge Scan Join



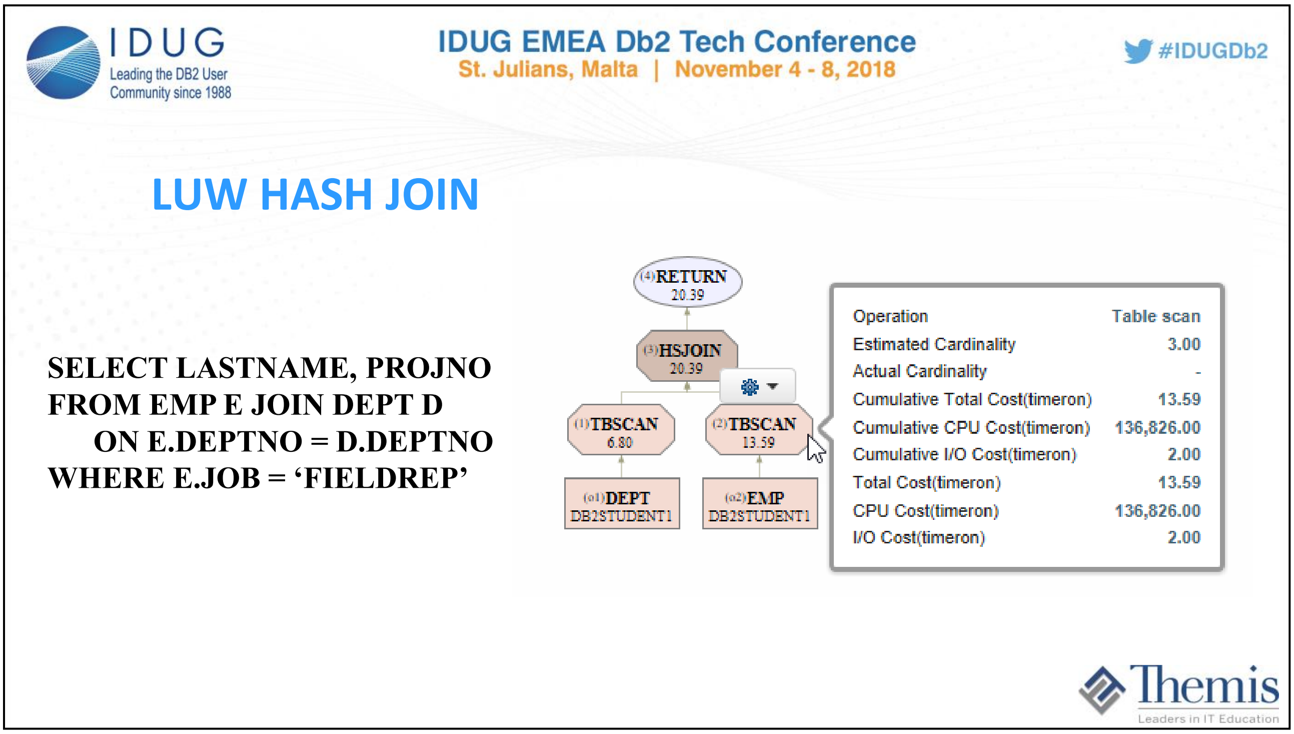


z/OS Hybrid Join

```
SELECT LASTNAME, PROJNO
FROM EMP E JOIN EMPPROJACT EPA
    ON E.EMPNO = EPA.EMPNO
WHERE E.JOB = 'FIELDREP'
```

Hybrid join works well when:

- The outer table qualifies many rows
- A non-clustered index exists on the join key for the inner table
- Many rows will be eliminated as a result of the join



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

LUW HASH JOIN

```
SELECT LASTNAME, PROJNO
FROM EMP E JOIN DEPT D
     ON E.DEPTNO = D.DEPTNO
WHERE E.JOB = 'FIELDREP'
```

Operation	Table scan
Estimated Cardinality	3.00
Actual Cardinality	-
Cumulative Total Cost(timeron)	13.59
Cumulative CPU Cost(timeron)	136,826.00
Cumulative I/O Cost(timeron)	2.00
Total Cost(timeron)	13.59
CPU Cost(timeron)	136,826.00
I/O Cost(timeron)	2.00

Themis
Leaders in IT Education

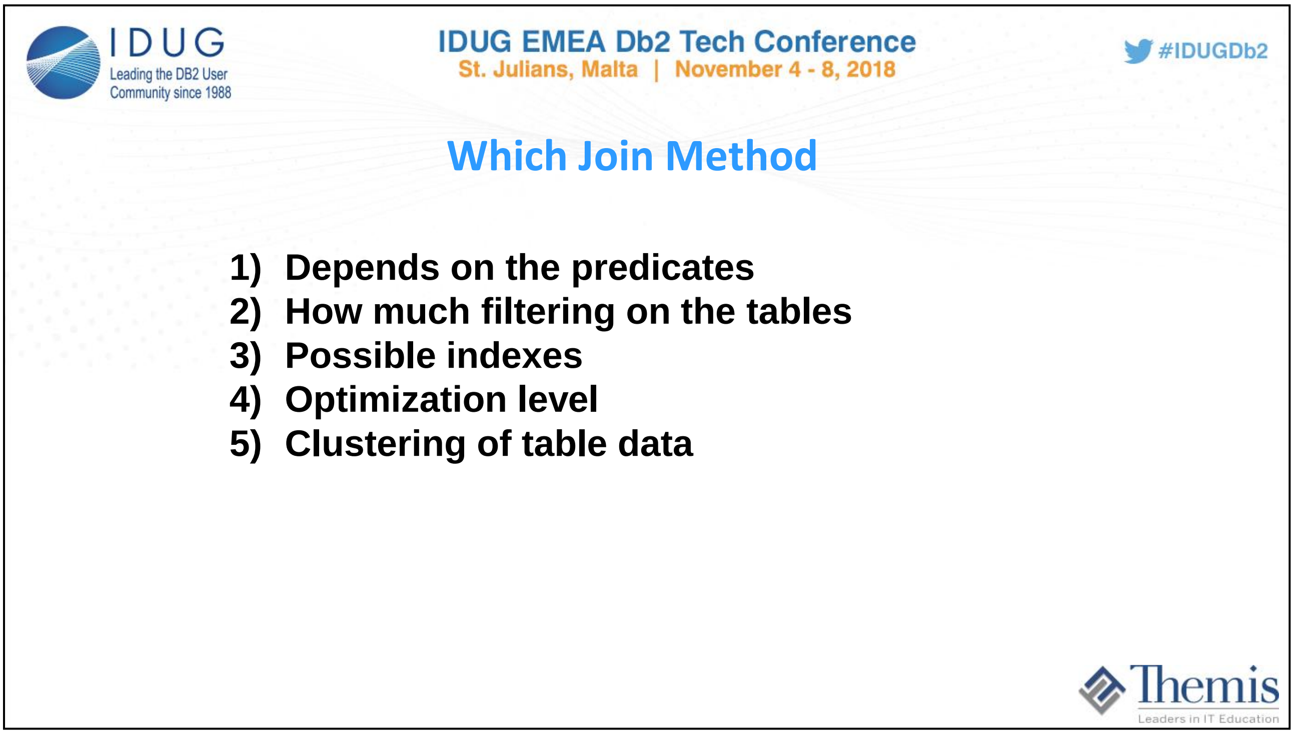
For batch jobs that process large portions of tables, hash joins are often the best choice. Nested loop joins are good when there are indexable predicates on the inner table and the inner table is clustered in the same order as the outer table. Otherwise, hash joins can perform much better. This is because there is one pass of each table and sorts are not needed. A single pass of a table using prefetch is the best you can do if you need to process the entire table.

DB2 builds a hash table in memory by reading one of the tables (the inner table – the right hand side of the access path graph. The hash table is a set of memory blocks (called buckets).

- Each key is hashed by an algorithm and the key and needed columns are stored in the corresponding bucket in the hash table.
- More than one key can hash to the same bucket. DB2 will maintain a chain of those keys.

The second table (outer or left and side) is read

- The keys are hashed and the bucket is searched for a matching key.
- For each matching key, DB2 will pass the joined data to the next step in the access path graph.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Which Join Method

1) Depends on the predicates
2) How much filtering on the tables
3) Possible indexes
4) Optimization level
5) Clustering of table data

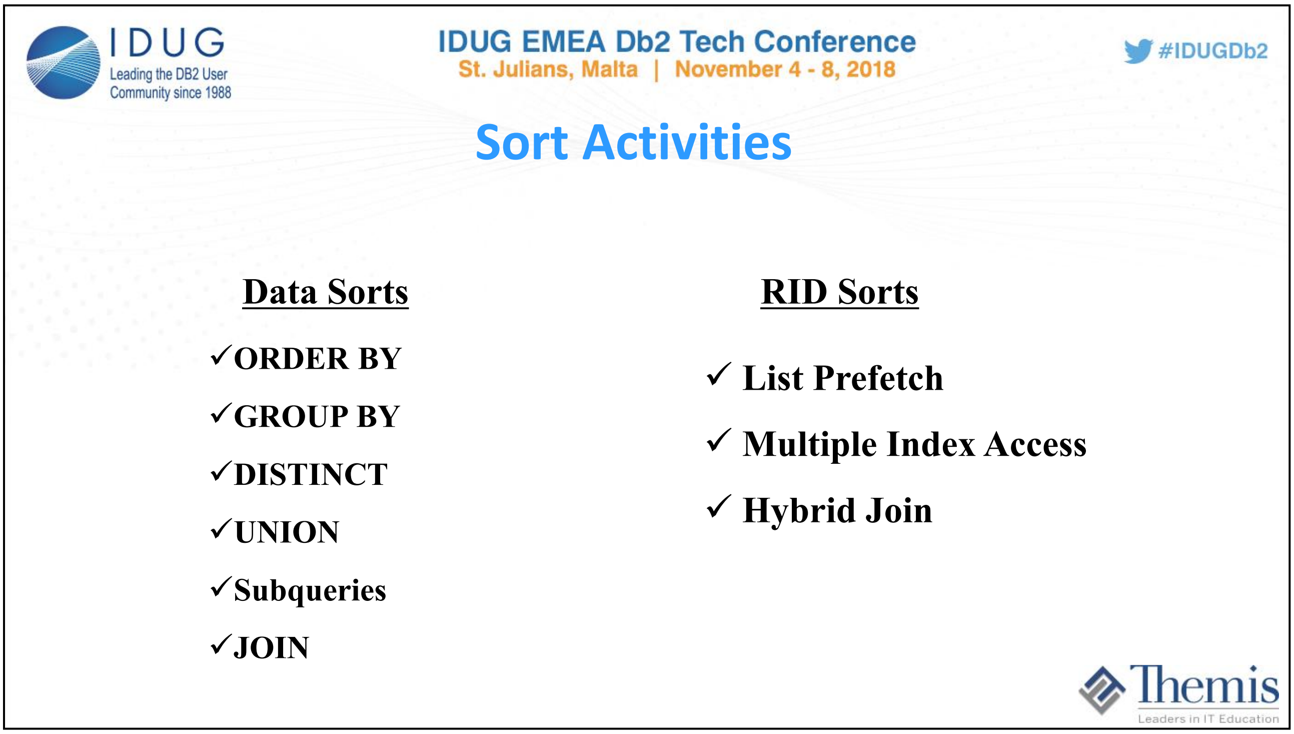
Themis Leaders in IT Education

Hybrid join works well when:

•The outer table qualifies many rows

•A non-clustered index exists on the join key for the inner table

•Many rows will be eliminated as a result of the join



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Sort Activities

Data Sorts

- ✓ORDER BY
- ✓GROUP BY
- ✓DISTINCT
- ✓UNION
- ✓Subqueries
- ✓JOIN

RID Sorts

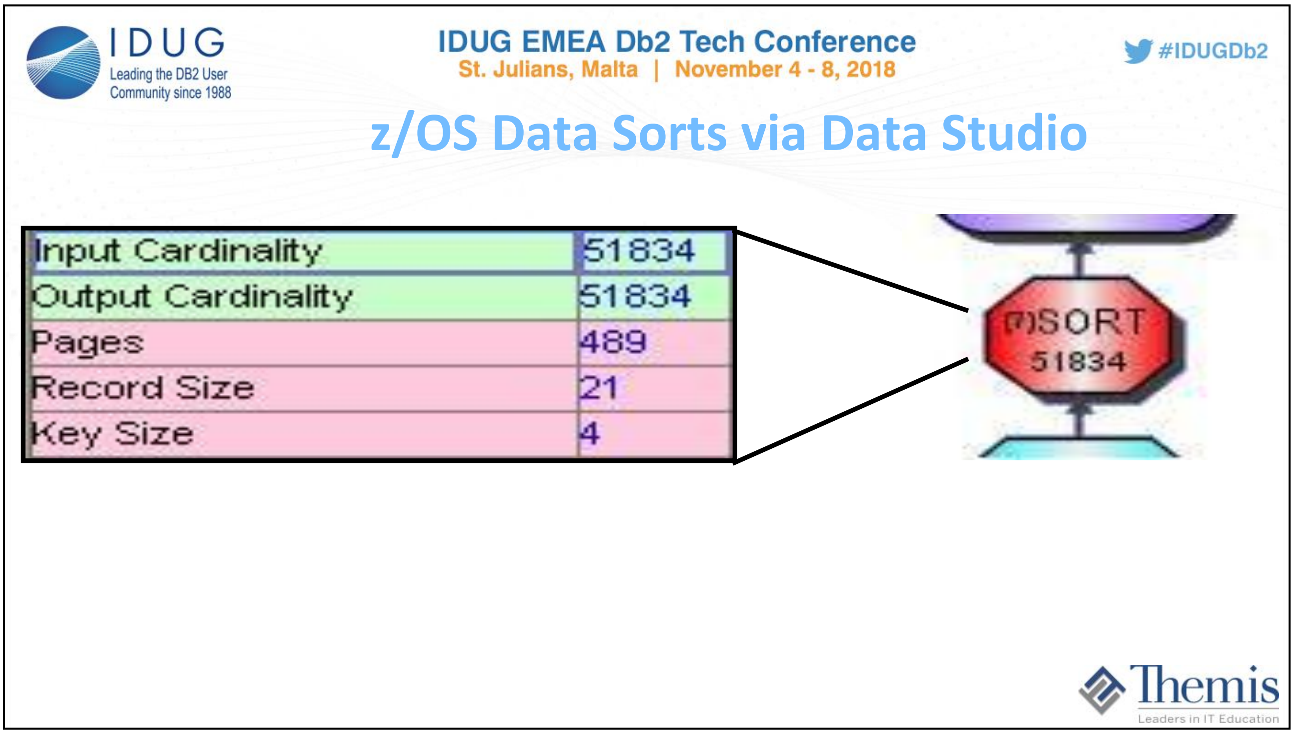
- ✓ List Prefetch
- ✓ Multiple Index Access
- ✓ Hybrid Join

Themis
Leaders in IT Education

Whenever a sort is seen in the explain output, 2 questions should always come to mind:

1. Is the sort needed in this query

2. If so, how many rows are going into the sort. The more rows and columns selected, the more expensive the sort.



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

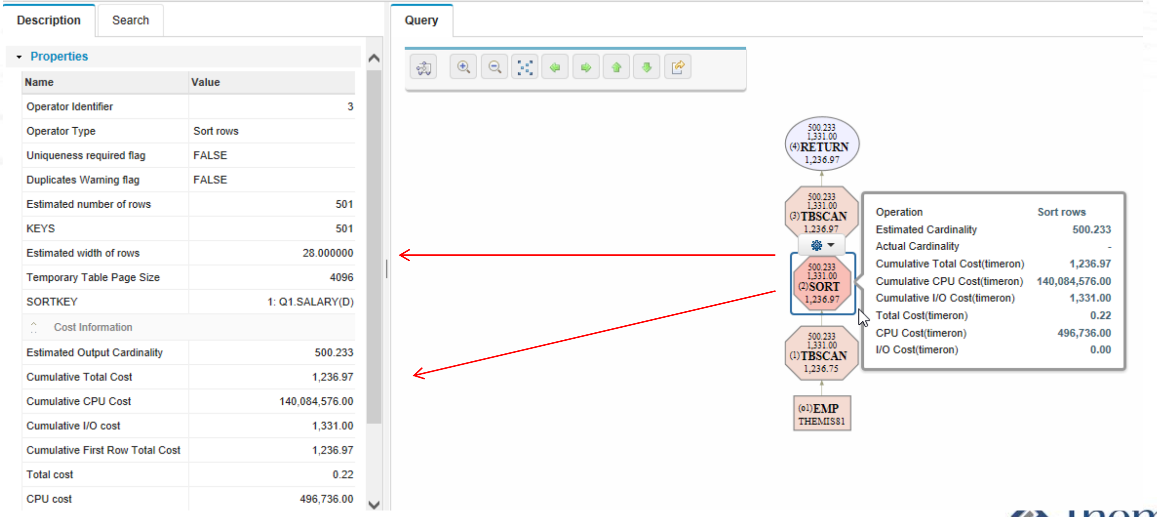
z/OS Data Sorts via Data Studio

Input Cardinality	51834
Output Cardinality	51834
Pages	489
Record Size	21
Key Size	4

Themis
Leaders in IT Education

Sorts are represented in Visual Explain by red nodes that indicate the reason for the sort and an estimate of the number of records to be processed by the sort. The number 51834 represents the guesstimated number of rows going into the sort based on the ‘Where’ criteria.

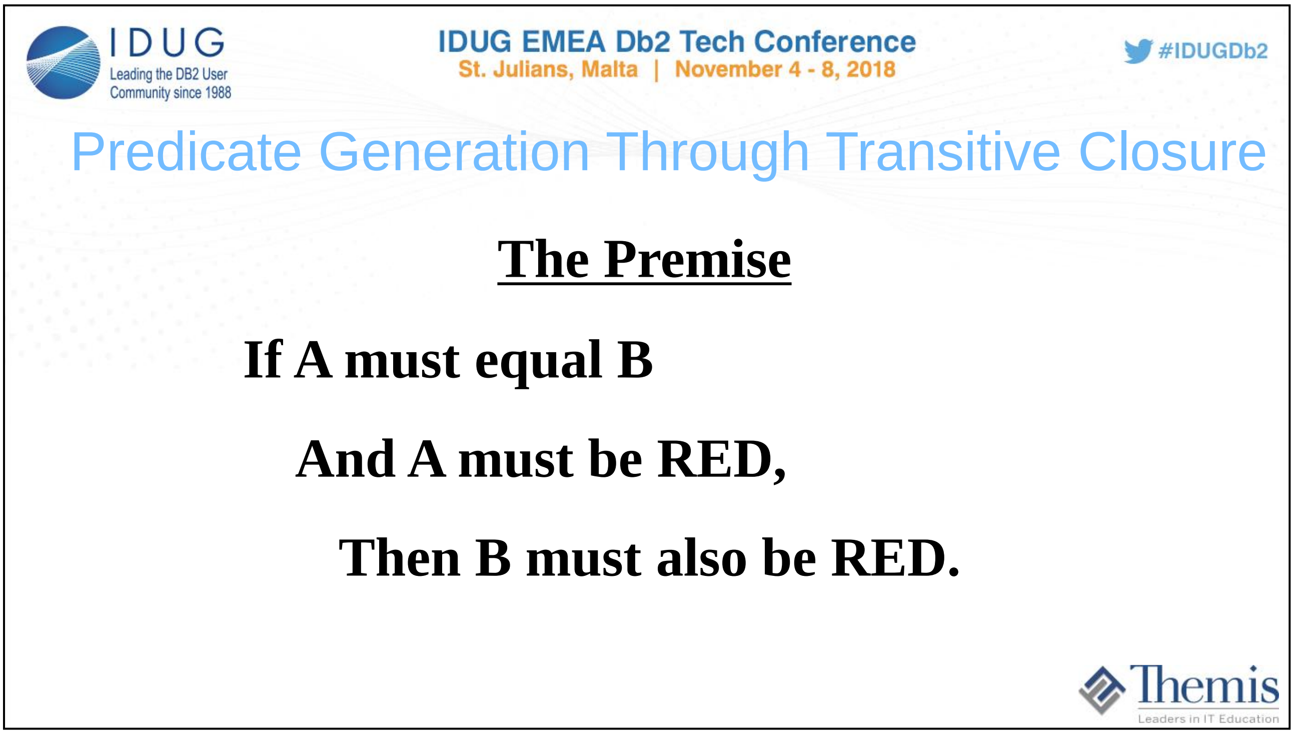
LUW Data Sorts via Data Studio



Name	Value
Operator Identifier	3
Operator Type	Sort rows
Uniqueness required flag	FALSE
Duplicates Warning flag	FALSE
Estimated number of rows	501
KEYS	501
Estimated width of rows	28.000000
Temporary Table Page Size	4096
SORTKEY	1: Q1.SALARY(D)
Cost Information	
Estimated Output Cardinality	500.233
Cumulative Total Cost	1,236.97
Cumulative CPU Cost	140,084,576.00
Cumulative I/O cost	1,331.00
Cumulative First Row Total Cost	1,236.97
Total cost	0.22
CPU cost	496,736.00

Operation	Sort rows
Estimated Cardinality	500.233
Actual Cardinality	-
Cumulative Total Cost(timeron)	1,236.97
Cumulative CPU Cost(timeron)	140,084,576.00
Cumulative I/O Cost(timeron)	1,331.00
Total Cost(timeron)	0.22
CPU Cost(timeron)	496,736.00
I/O Cost(timeron)	0.00

Sorts are represented in Visual Explain by red nodes that indicate the reason for the sort and an estimate of the number of records to be processed by the sort.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Predicate Generation Through Transitive Closure

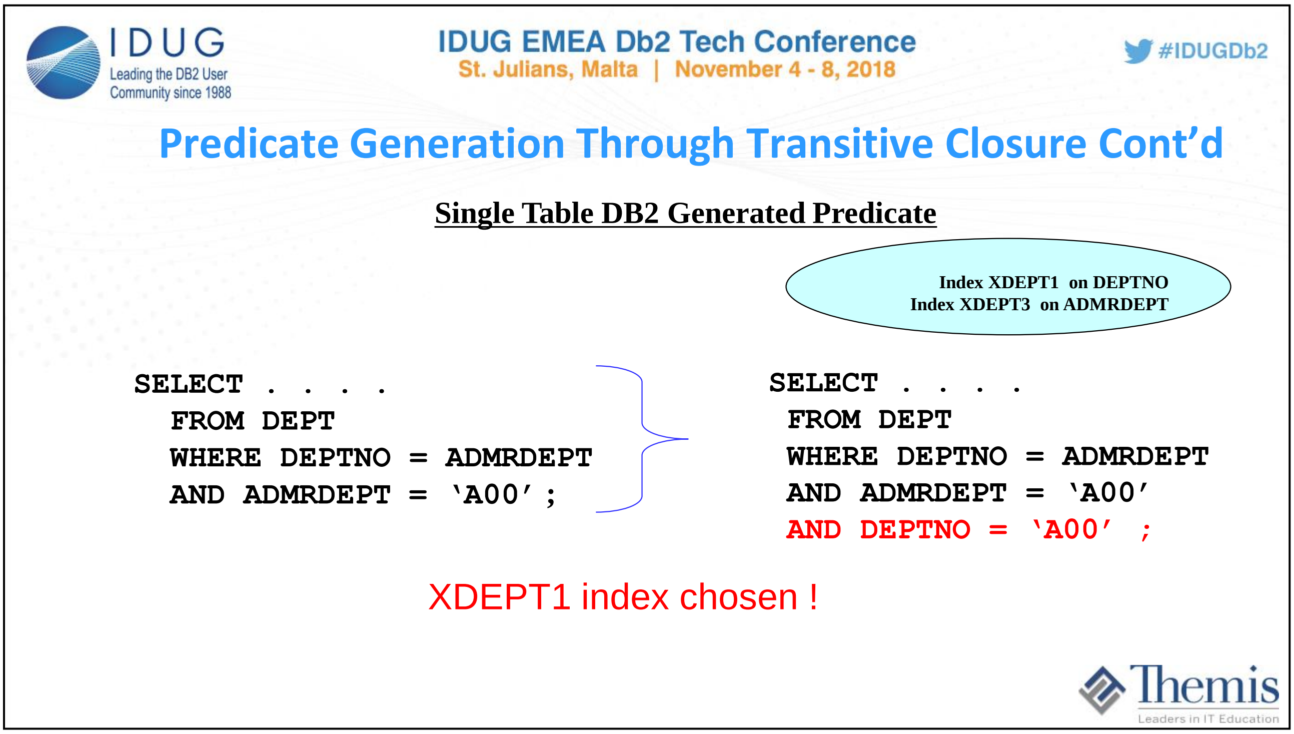
The Premise

If A must equal B

And A must be RED,

Then B must also be RED.

Themis
Leaders in IT Education



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Predicate Generation Through Transitive Closure Cont'd

Single Table DB2 Generated Predicate

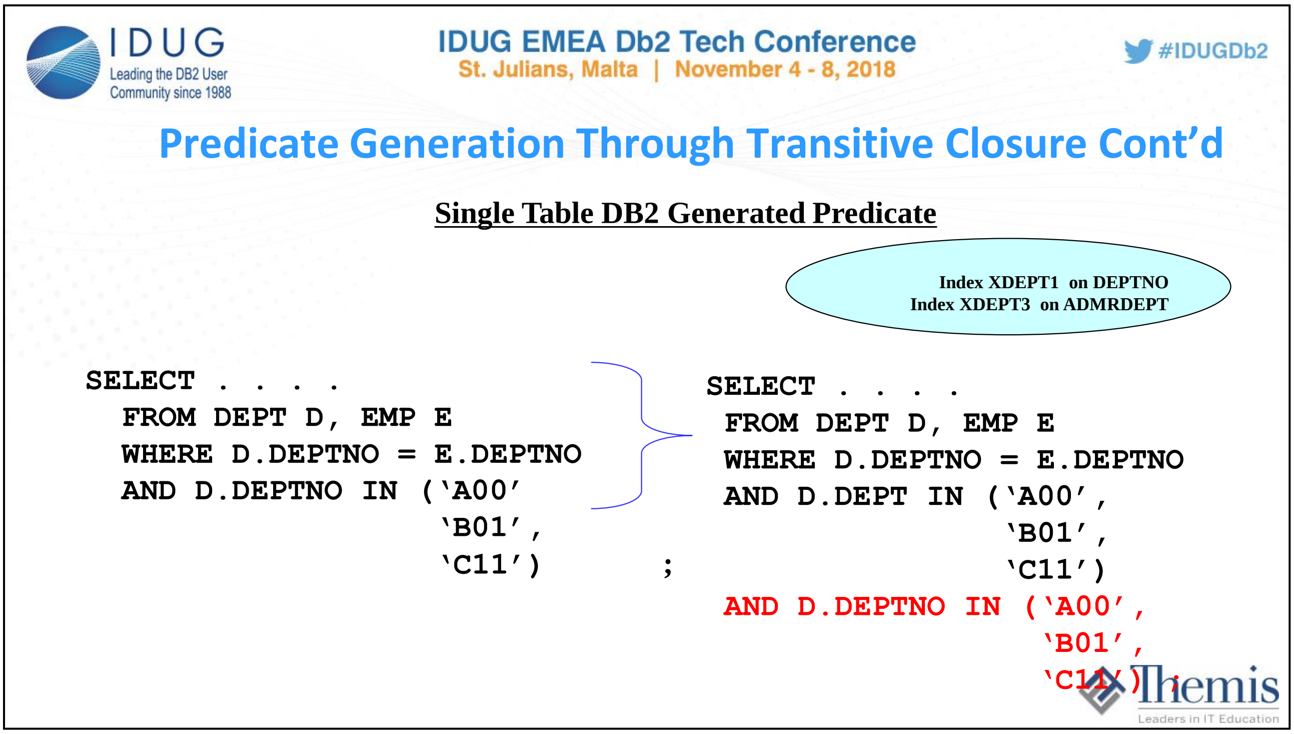
Index XDEPT1 on DEPTNO
Index XDEPT3 on ADMRDEPT

```
SELECT . . . .
  FROM DEPT
  WHERE DEPTNO = ADMRDEPT
  AND ADMRDEPT = 'A00' ;
```

```
SELECT . . . .
 FROM DEPT
 WHERE DEPTNO = ADMRDEPT
 AND ADMRDEPT = 'A00'
 AND DEPTNO = 'A00' ;
```

XDEPT1 index chosen !

Themis
Leaders in IT Education



Predicate Generation Through Transitive Closure Cont’d

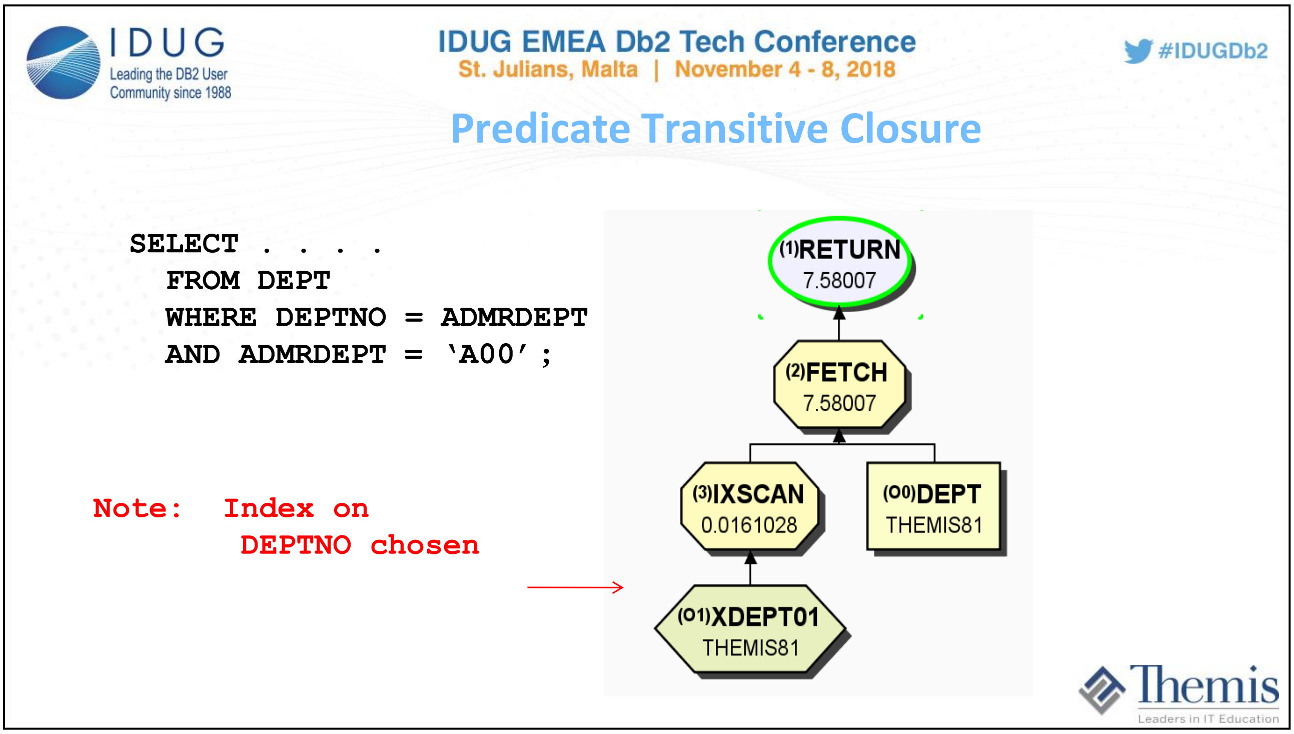
Single Table DB2 Generated Predicate

Index XDEPT1 on DEPTNO
Index XDEPT3 on ADMRDEPT

```
SELECT . . . .
  FROM DEPT D, EMP E
  WHERE D.DEPTNO = E.DEPTNO
  AND D.DEPTNO IN (‘A00’
                   ‘B01’,
                   ‘C11’)      ;
```

```
SELECT . . . .
 FROM DEPT D, EMP E
 WHERE D.DEPTNO = E.DEPTNO
 AND D.DEPT IN (‘A00’,
                ‘B01’,
                ‘C11’)
 AND D.DEPTNO IN (‘A00’,
                  ‘B01’,
                  ‘C11’);
```

Themis
Leaders in IT Education



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Predicate Transitive Closure

```
SELECT . . . .
   FROM DEPT
   WHERE DEPTNO = ADMRDEPT
   AND ADMRDEPT = 'A00' ;
```

Note: Index on
DEPTNO chosen

Themis Leaders in IT Education

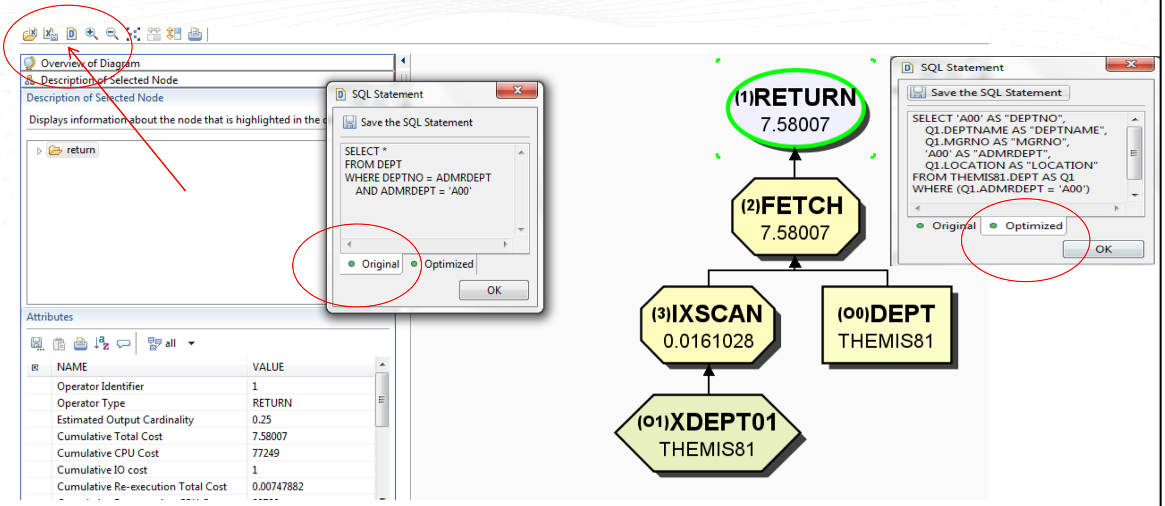
Visual Explain will show all predicates used by the optimizer, both those included in the query and those generated by predicate transitive closure.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

LUW Predicate Transitive Closure



```
SELECT *
FROM DEPT
WHERE DEPTNO = ADMRDEPT
   AND ADMRDEPT = 'A00'
```

NAME	VALUE
Operator Identifier	1
Operator Type	RETURN
Estimated Output Cardinality	0.25
Cumulative Total Cost	7.58007
Cumulative CPU Cost	77249
Cumulative IO cost	1
Cumulative Re-execution Total Cost	0.00747882

```
SELECT 'A00' AS "DEPTNO",
   Q1.DEPTNAME AS "DEPTNAME",
   Q1.MGRNO AS "MGRNO",
   'A00' AS "ADMRDEPT",
   Q1.LOCATION AS "LOCATION"
FROM THEMIS81.DEPT AS Q1
WHERE (Q1.ADMRDEPT = 'A00')
```

Themis
Leaders in IT Education

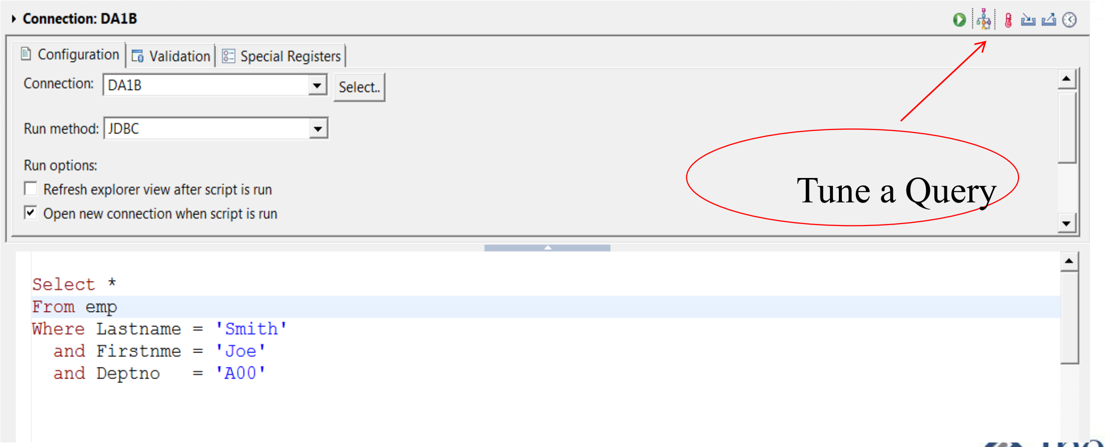
Visual Explain will show all predicates used by the optimizer, both those included in the query and those generated by predicate transitive closure.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Tuning a Query



Connection: DA1B

Connection: DA1B

Run method: JDBC

Run options:

- □ Refresh explorer view after script is run
- ☑ Open new connection when script is run

Tune a Query

```
Select *
From emp
Where Lastname = 'Smith'
  and Firstnme = 'Joe'
  and Deptno    = 'A00'
```

nemis
Leaders in IT Education

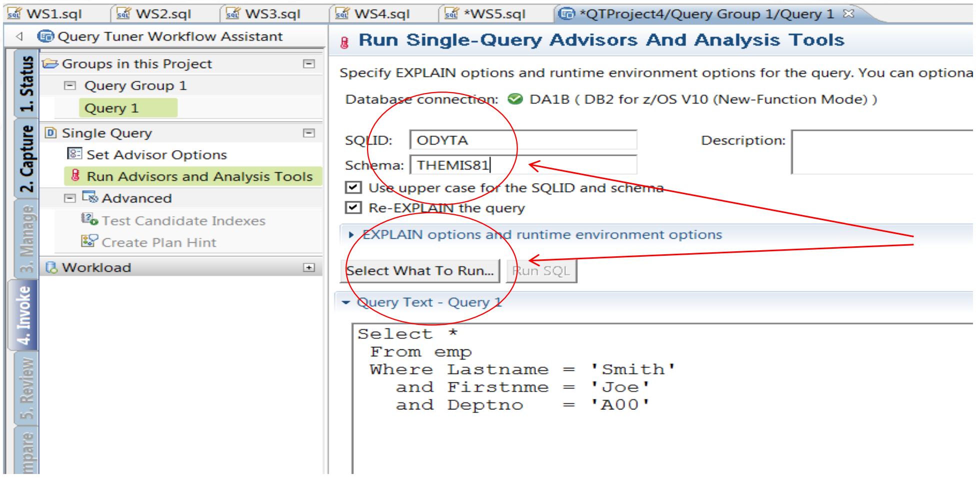
In z/OS you have to go into ‘Tune a query’ in order to see the original and transformed SQL.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Tuning a Query



Run Single-Query Advisors And Analysis Tools

Specify EXPLAIN options and runtime environment options for the query. You can optiona

Database connection: DA1B (DB2 for z/OS V10 (New-Function Mode))

SQLID: ODYTA

Schema: THEMIS81

Description:

☑ Use upper case for the SQLID and schema

☑ Re-EXPLAIN the query

▸ EXPLAIN options and runtime environment options

▾ Query Text - Query 1

```
Select *
 From emp
 Where Lastname  = 'Smith'
    and Firstnme = 'Joe'
    and Deptno   = 'A00'
```

Themis
Leaders in IT Education

Notice some of the options are inaccessible in the free standalone version. Click on the ‘Select What to Run’to initiate further tuning of the SQL statement.

Also notice settings for:

SQLID - Set to the owner of plan table s for the tool to use

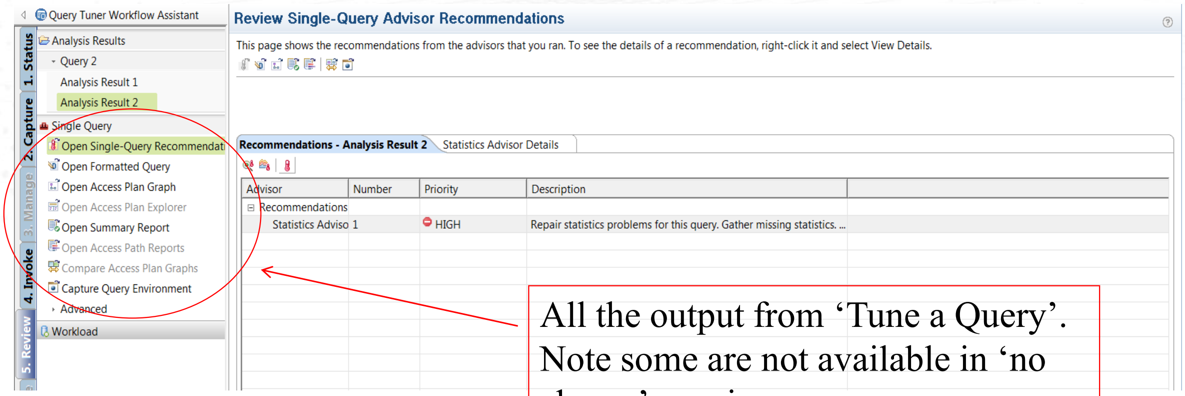
SCHEMA – Set to the owner of tables involved in the query

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Tuning a Query Output



Review Single-Query Advisor Recommendations

This page shows the recommendations from the advisors that you ran. To see the details of a recommendation, right-click it and select View Details.

Recommendations - Analysis Result 2 | Statistics Advisor Details

Advisor	Number	Priority	Description
⊟ Recommendations			
Statistics Adviso	1	⊖ HIGH	Repair statistics problems for this query. Gather missing statistics. ...

All the output from ‘Tune a Query’. Note some are not available in ‘no charge’ version.

Themis
Leaders in IT Education

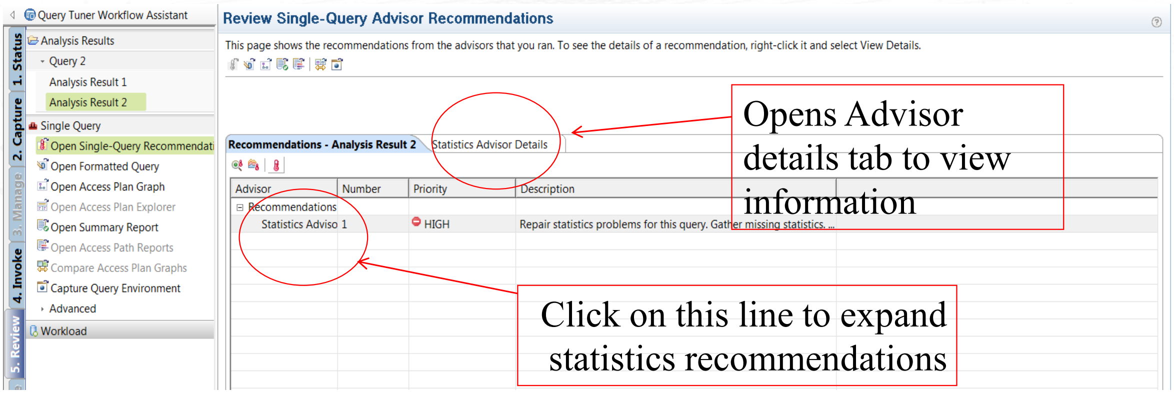
After executing the query tuning, options open up for analysis (Formatted Query, Access Plan Graph, Summary Report, Statistics Advisors, etc.)

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Statistics Advisor



Review Single-Query Advisor Recommendations

This page shows the recommendations from the advisors that you ran. To see the details of a recommendation, right-click it and select View Details.

Recommendations - Analysis Result 2
Statistics Advisor Details

Advisor	Number	Priority	Description
⊟ Recommendations			
Statistics Adviso	1	● HIGH	Repair statistics problems for this query. Gather missing statistics. ...

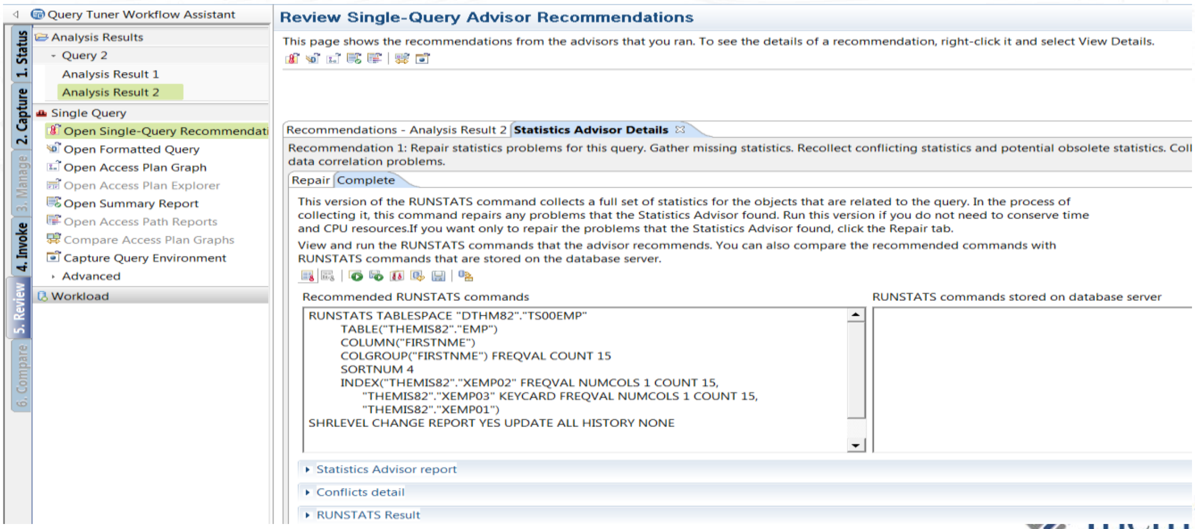
Opens Advisor details tab to view information

Click on this line to expand statistics recommendations

Themis
Leaders in IT Education

Statistics Advisories are noted by priorities. Each can be clicked on to provide more details and specific recommendations.

Statistics Advisor Summary Report



Review Single-Query Advisor Recommendations

This page shows the recommendations from the advisors that you ran. To see the details of a recommendation, right-click it and select View Details.

Recommendation 1: Repair statistics problems for this query. Gather missing statistics. Recollect conflicting statistics and potential obsolete statistics. Coll data correlation problems.

This version of the RUNSTATS command collects a full set of statistics for the objects that are related to the query. In the process of collecting it, this command repairs any problems that the Statistics Advisor found. Run this version if you do not need to conserve time and CPU resources.If you want only to repair the problems that the Statistics Advisor found, click the Repair tab.

View and run the RUNSTATS commands that the advisor recommends. You can also compare the recommended commands with RUNSTATS commands that are stored on the database server.

Recommended RUNSTATS commands

```
RUNSTATS TABLESPACE "DTHM82"."TS00EMP"
      TABLE("THEMIS82"."EMP")
      COLUMN("FIRSTNME")
      COLGROUP("FIRSTNME") FREQVAL COUNT 15
      SORTNUM 4
      INDEX("THEMIS82"."XEMP02" FREQVAL NUMCOLS 1 COUNT 15,
          "THEMIS82"."XEMP03" KEYCARD FREQVAL NUMCOLS 1 COUNT 15,
          "THEMIS82"."XEMP01")
SHRLEVEL CHANGE REPORT YES UPDATE ALL HISTORY NONE
```

RUNSTATS commands stored on database server

Leaders in IT Education

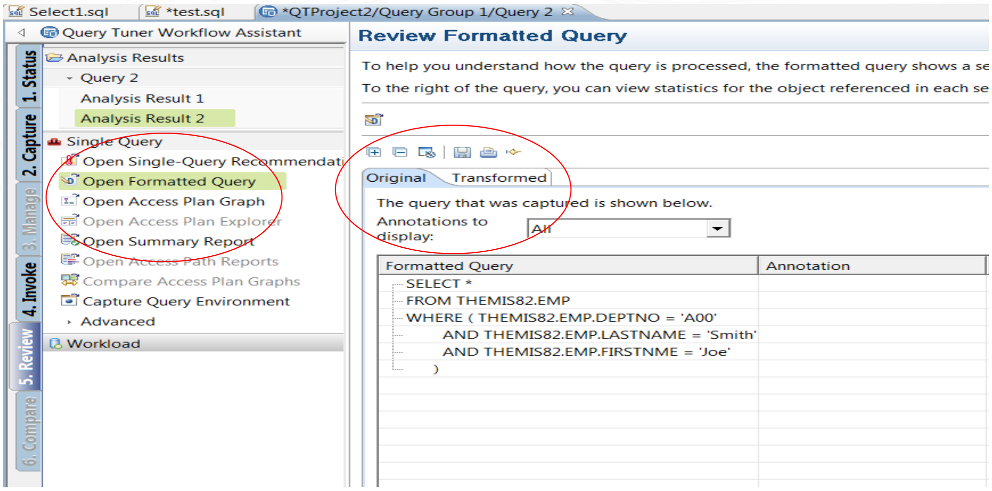
Statistics Advisories are noted by priorities. Each can be clicked on to provide more details And specific recommendations.

IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Tune a query – Query Transformation



Review Formatted Query

To help you understand how the query is processed, the formatted query shows a se
To the right of the query, you can view statistics for the object referenced in each se

The query that was captured is shown below.

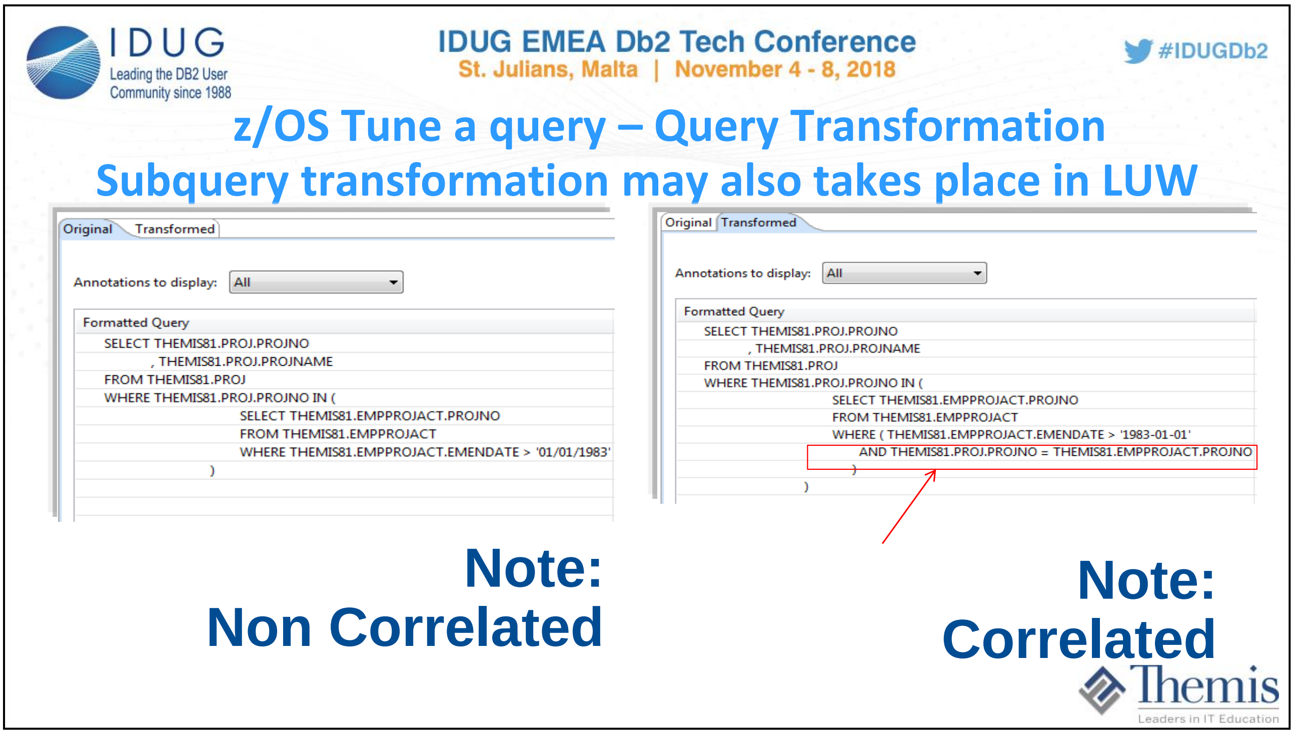
Annotations to display: All

Formatted Query	Annotation
SELECT *	
FROM THEMIS82.EMP	
WHERE (THEMIS82.EMP.DEPTNO = 'A00'	
AND THEMIS82.EMP.LASTNAME = 'Smith'	
AND THEMIS82.EMP.FIRSTNME = 'Joe'	
)	

Themis
Leaders in IT Education

Open formatted query is one the best features in query tuning. DB2 optimization has always rewritten (transformed) queries a t times in the past, and now the rewrite (if any) is being externalized.

This is especially important with the V9 Global Query Optimization enhancement.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS Tune a query – Query Transformation
Subquery transformation may also takes place in LUW

Original | Transformed

Annotations to display: All

Formatted Query

```
SELECT THEMIS81.PROJ.PROJNO
       , THEMIS81.PROJ.PROJNAME
FROM THEMIS81.PROJ
WHERE THEMIS81.PROJ.PROJNO IN (
                    SELECT THEMIS81.EMPPROJACT.PROJNO
                    FROM THEMIS81.EMPPROJACT
                    WHERE THEMIS81.EMPPROJACT.EMENDATE > '01/01/1983'
              )
```

Original | Transformed

Annotations to display: All

Formatted Query

```
SELECT THEMIS81.PROJ.PROJNO
       , THEMIS81.PROJ.PROJNAME
FROM THEMIS81.PROJ
WHERE THEMIS81.PROJ.PROJNO IN (
                    SELECT THEMIS81.EMPPROJACT.PROJNO
                    FROM THEMIS81.EMPPROJACT
                    WHERE ( THEMIS81.EMPPROJACT.EMENDATE > '1983-01-01'
                        AND THEMIS81.PROJ.PROJNO = THEMIS81.EMPPROJACT.PROJNO
                        )
                 )
```

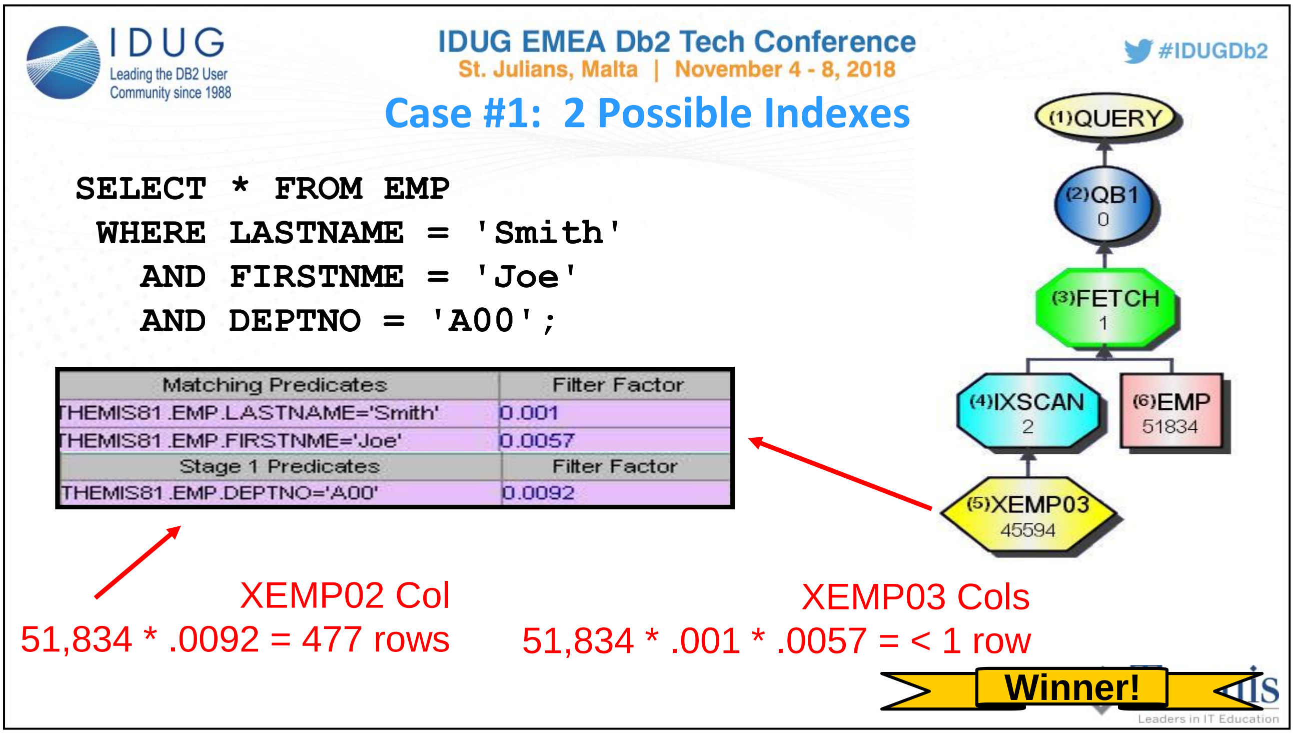
**Note:
Non Correlated**

**Note:
Correlated**

Themis
Leaders in IT Education

Open formatted query is one the best features in query tuning. DB2 optimization has always rewritten (transformed) queries at times in the past, and now the rewrite (if any) is being externalized. But by no means should we cut and paste the transformed query into the code. This is because along with the Transformed rewrite are other 'hidden' pieces of information that get passed to the access path selection step. The transformed query does tell the whole tranformation.

This is especially important with the V9 Global Query Optimization enhancement.



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

Case #1: 2 Possible Indexes

```
SELECT * FROM EMP
 WHERE LASTNAME = 'Smith'
   AND FIRSTNME = 'Joe'
   AND DEPTNO = 'A00';
```

Matching Predicates	Filter Factor
THEMIS81.EMP.LASTNAME='Smith'	0.001
THEMIS81.EMP.FIRSTNME='Joe'	0.0057
Stage 1 Predicates	**Filter Factor**
THEMIS81.EMP.DEPTNO='A00'	0.0092

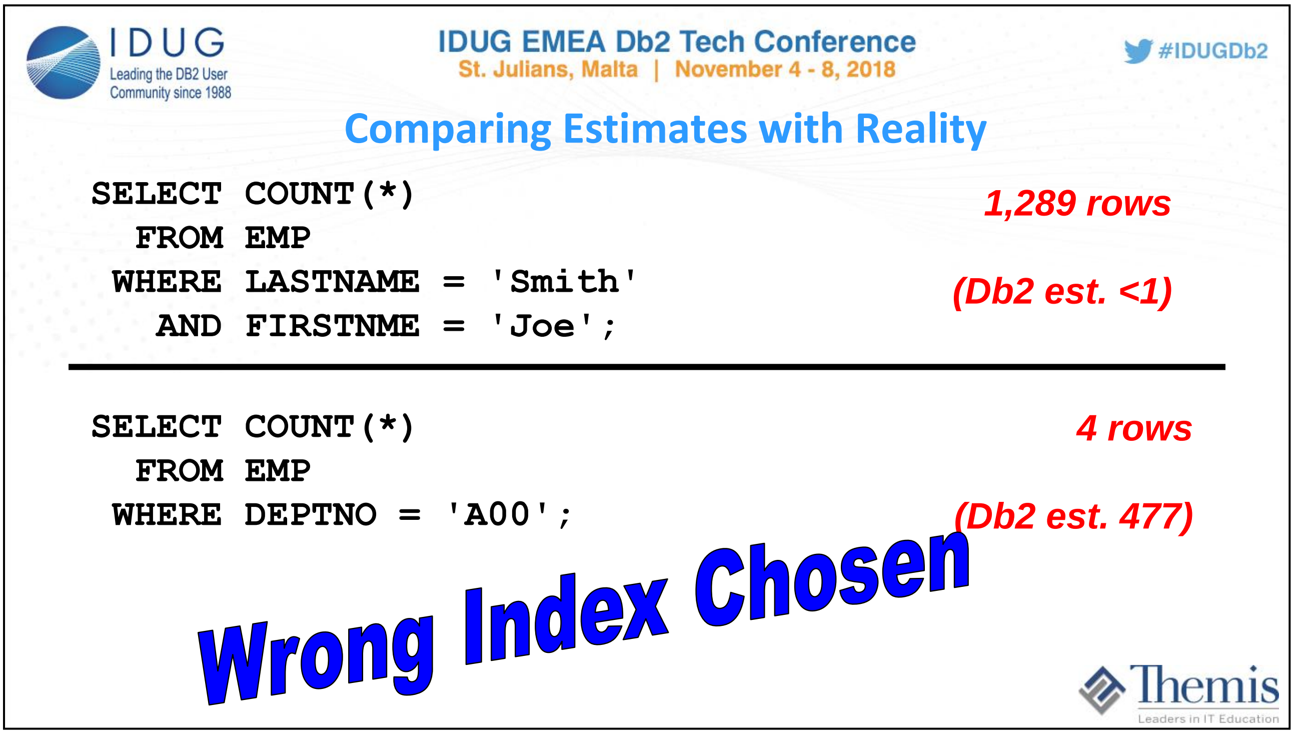
XEMP02 Col
51,834 * .0092 = 477 rows

XEMP03 Cols
51,834 * .001 * .0057 = < 1 row

Winner!

IBM Data Studio can often tell us the ‘Why’ of a certain access path choice. In this example DB2 chose the XEMP03 index because based on statistics, it was showing the lest amount of rows that met the criteria. So the explain looks good, but could run better ……

In this example there are two possible indexes that could be used to retrieve the desired result. Index XEPM02 contains column DEPTNO while index XEMP03 contains a concatenation of LASTNAME, FIRSTNME and MIDINIT. Db2 could choose to use multiple index access, but this is unlikely unless neither index provides a high degree of filtering. By computing the filter factors for the 2 available indexes using the formula for uniform distribution, Db2 determines that about 447 rows will qualify for the DEPTNO predicate and less than one row will qualify for the combination of lastname and firstname requested. Since XEMP03 is perceived to narrow the result to one row, it is chosen as the index to be used for qualifying the rows.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Comparing Estimates with Reality

```
SELECT COUNT(*)
  FROM EMP
 WHERE LASTNAME = 'Smith'
   AND FIRSTNME = 'Joe';
```

1,289 rows

(Db2 est. <1)

```
SELECT COUNT(*)
  FROM EMP
 WHERE DEPTNO = 'A00';
```

4 rows

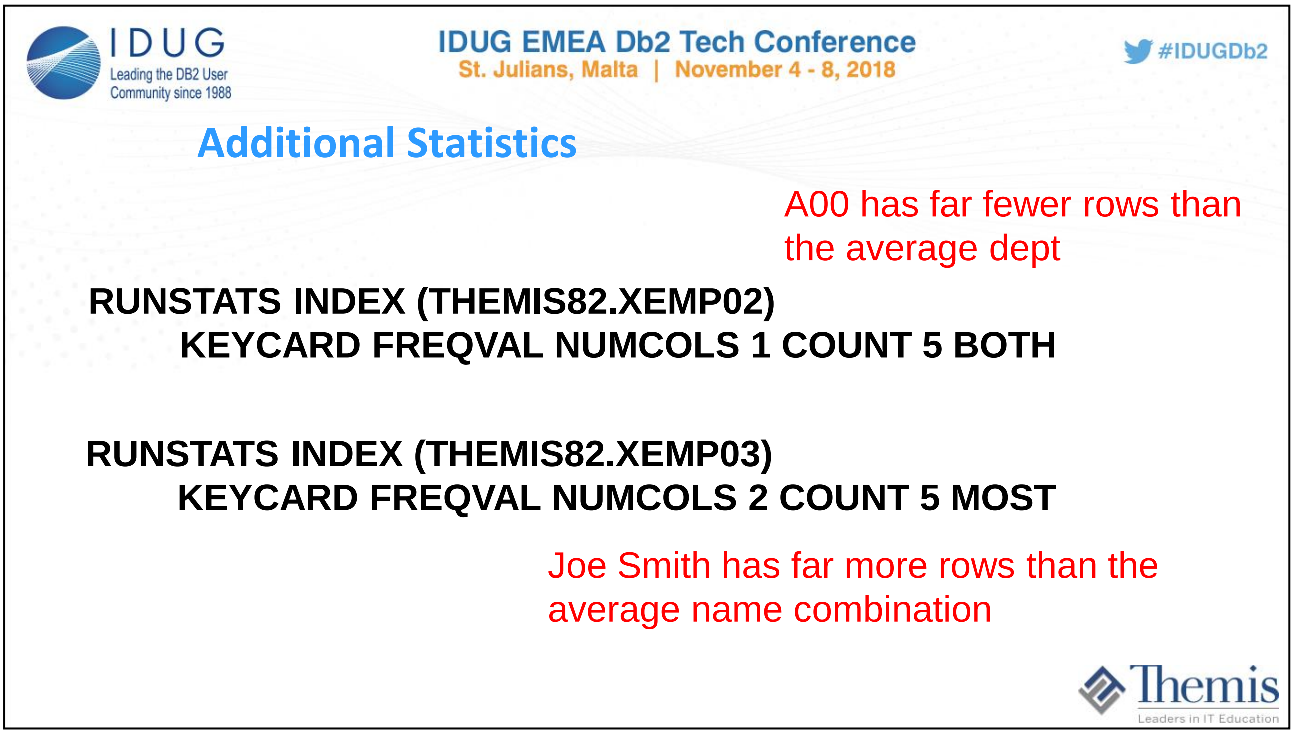
(Db2 est. 477)

Wrong Index Chosen

Themis Leaders in IT Education

If this query is not performing optimally, it may be useful to compare the filter factors presented in Data Studio with actual row counts on the tables involved. In this case, the optimizer's estimate of 1 row for index XEMP03 was incorrect (1289 rows actually exist). The estimate for XEMP02 of 477 qualifying rows was also incorrect (actual count was only 4).

In this case a poor filter factor estimate led to the wrong index being selected.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Additional Statistics

A00 has far fewer rows than the average dept

```
RUNSTATS INDEX (THEMIS82.XEMP02)
      KEYCARD FREQVAL NUMCOLS 1 COUNT 5 BOTH

RUNSTATS INDEX (THEMIS82.XEMP03)
      KEYCARD FREQVAL NUMCOLS 2 COUNT 5 MOST
```

Joe Smith has far more rows than the average name combination

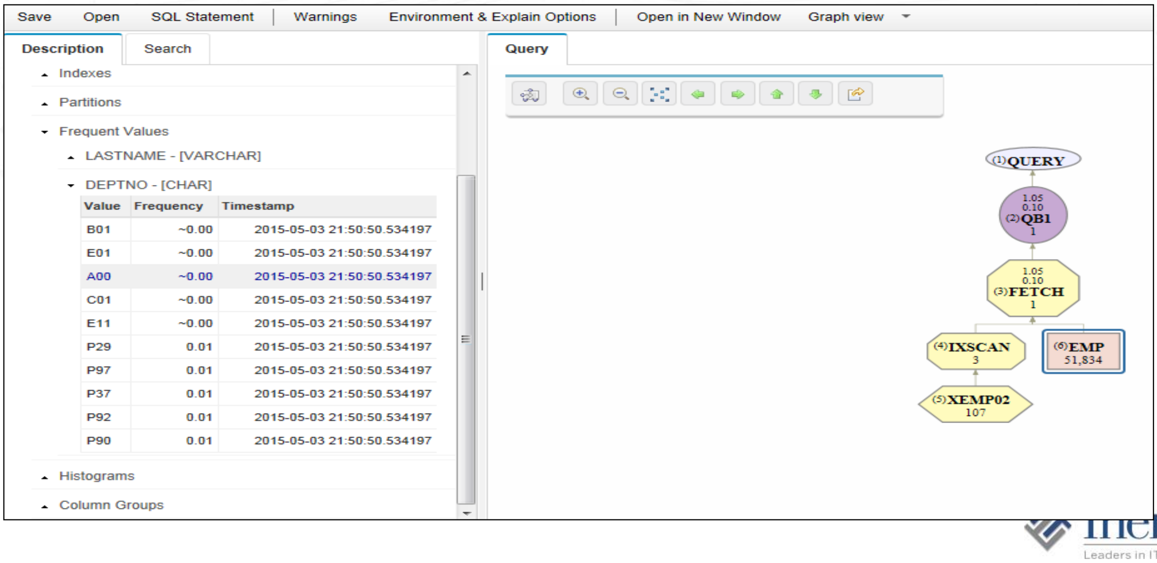
Themis
Leaders in IT Education

Uneven distribution of data in this table is the reason for the poor filter factor estimates. In this case department A00 has far fewer occurrences than the average department. Joe Smith also occurred far more frequently than the average name combination. Additional index statistics may be gathered to inform the optimizer of the statistical outliers. In the first control card, the runstats utility is being requested to capture both the 5 most frequently occurring values and the 5 least frequently occurring values for DEPTNO and store these values in SYSCOLDIST. The second control card is requesting the top 5 combinations for the first two columns of XEMP03 (LASTNAME and FIRSTNME).

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Frequency Stats



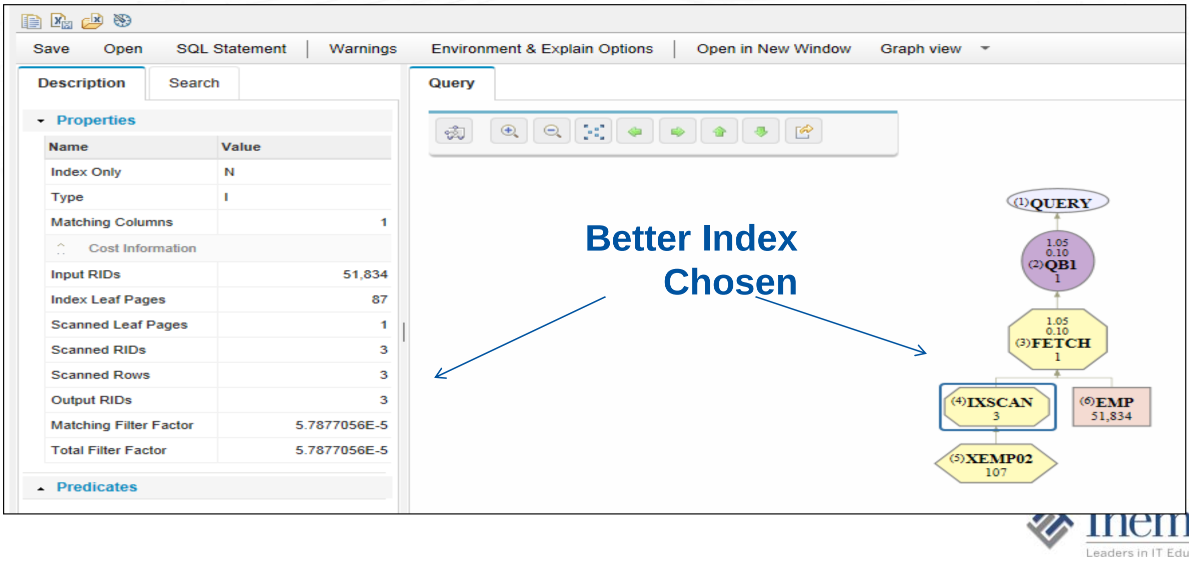
- Indexes
- Partitions
- Frequent Values
  - LASTNAME - [VARCHAR]
  - DEPTNO - [CHAR]

Value	Frequency	Timestamp
B01	~0.00	2015-05-03 21:50:50.534197
E01	~0.00	2015-05-03 21:50:50.534197
A00	~0.00	2015-05-03 21:50:50.534197
C01	~0.00	2015-05-03 21:50:50.534197
E11	~0.00	2015-05-03 21:50:50.534197
P29	0.01	2015-05-03 21:50:50.534197
P97	0.01	2015-05-03 21:50:50.534197
P37	0.01	2015-05-03 21:50:50.534197
P92	0.01	2015-05-03 21:50:50.534197
P90	0.01	2015-05-03 21:50:50.534197

- Histograms
- Column Groups

Once these statistics have been gathered they may be viewed using Data Studio in the Coldist folder for the column statistics. The appropriate filter factor for department A00 is .000077 rather than .0092 as calculated using uniform distribution rules. This leads Db2 to estimate that about 4 rows will qualify for this predicate, which is exactly what the actual count revealed.

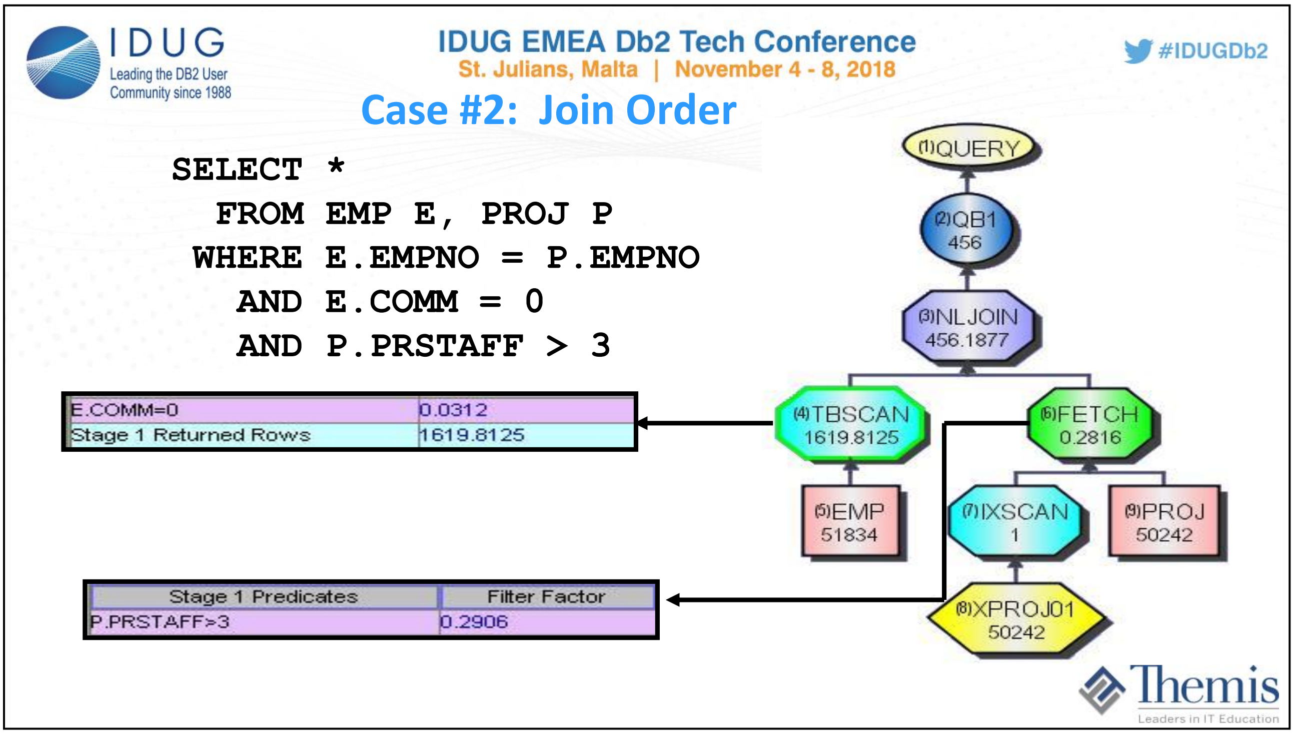
Result of Adding Frequency Stats



Name	Value
Index Only	N
Type	I
Matching Columns	1
Cost Information	
Input RIDs	51,834
Index Leaf Pages	87
Scanned Leaf Pages	1
Scanned RIDs	3
Scanned Rows	3
Output RIDs	3
Matching Filter Factor	5.7877056E-5
Total Filter Factor	5.7877056E-5

Better Index Chosen

With the additional statistics in place, the optimizer chooses XEMP02 to accomplish the filtering instead of XEMP03 resulting in better performance. More rows are eliminated from consideration earlier in the process using this access path.



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Case #2: Join Order

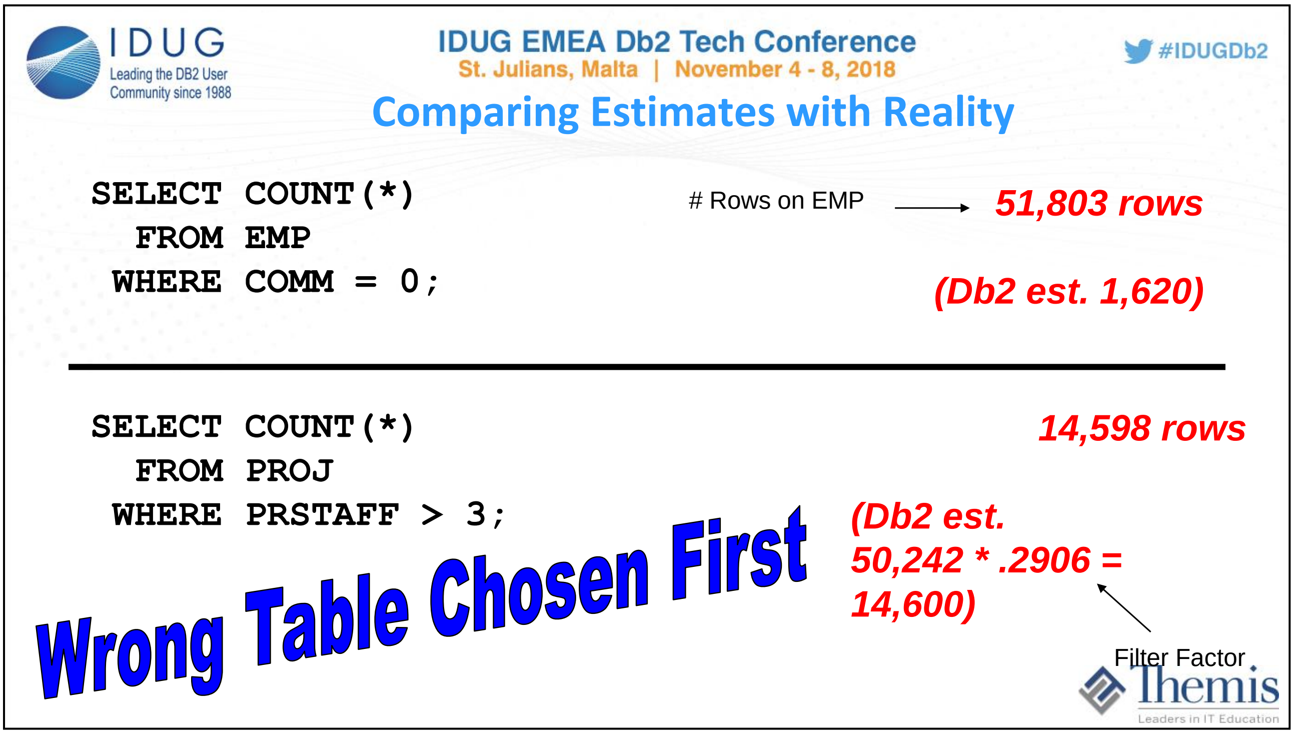
```
SELECT *
  FROM EMP E, PROJ P
 WHERE E.EMPNO = P.EMPNO
   AND E.COMM = 0
   AND P.PRSTAFF > 3
```

E.COMM=0	0.0312
Stage 1 Returned Rows	1619.8125

Stage 1 Predicates	Filter Factor
P.PRSTAFF>3	0.2906

A similar problem can occur when joining tables together even when the predicates involved are not indexed. In this example, the EMP table is being joined to the PROJ table. Local predicates exist on both tables, so Db2 must make what appears to be a narrow decision on which table should be accessed first.

There exist an index on PRSTAFF with some frequency values statistics.



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Comparing Estimates with Reality

```
SELECT COUNT(*)
  FROM EMP
 WHERE COMM = 0;
```

Rows on EMP → **51,803 rows**

(Db2 est. 1,620)

```
SELECT COUNT(*)
  FROM PROJ
 WHERE PRSTAFF > 3;
```

14,598 rows

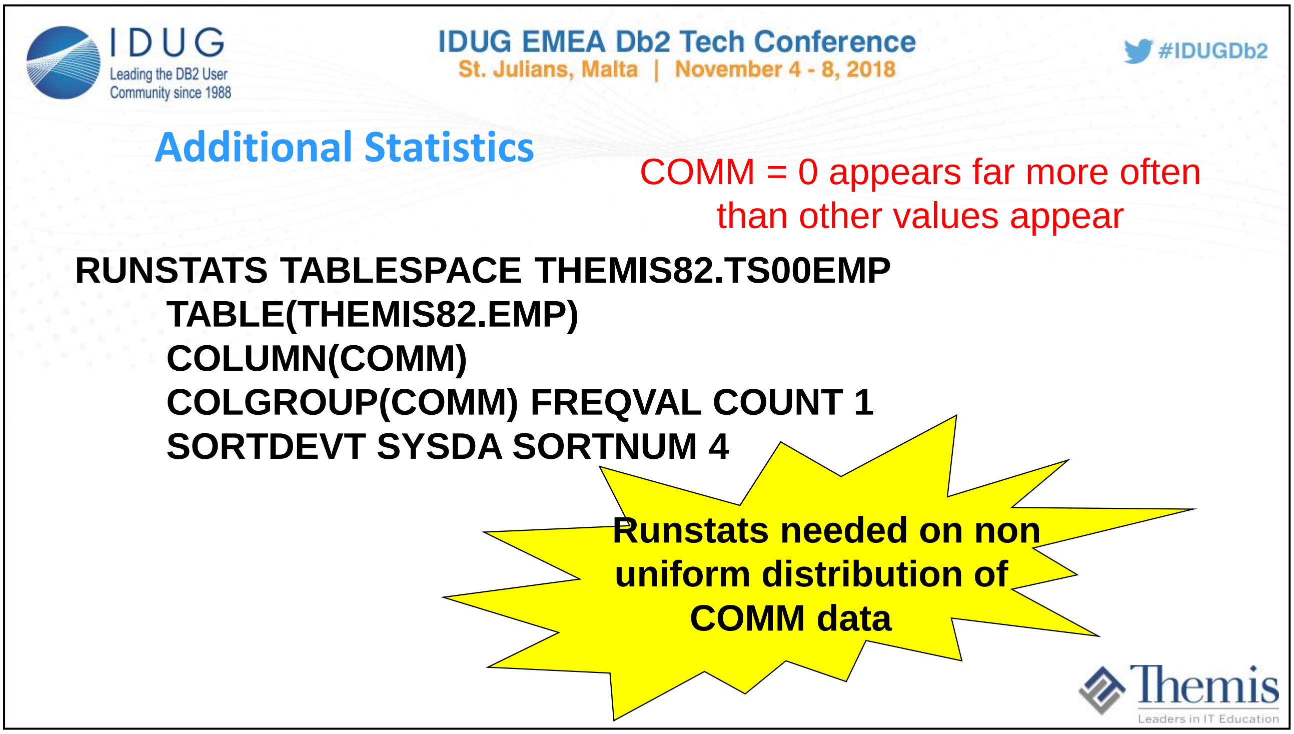
(Db2 est. 50,242 * .2906 = 14,600)

Filter Factor

Wrong Table Chosen First

Themis
Leaders in IT Education

The optimizer calculates that 1620 rows will qualify in the EMP table using formulas for uniform distribution. Actual counts reveal that 51,803 rows actually exist that meet the requested condition. Zero appears to be a default value for COMM that occurs much more often than any other value. This is a fairly common condition. Many columns have default or null values that occur much more often than anything else.



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Additional Statistics

COMM = 0 appears far more often than other values appear

```
RUNSTATS TABLESPACE THEMIS82.TS00EMP
         TABLE(THEMIS82.EMP)
         COLUMN(COMM)
         COLGROUP(COMM) FREQVAL COUNT 1
         SORTDEVT SYSDA SORTNUM 4
```

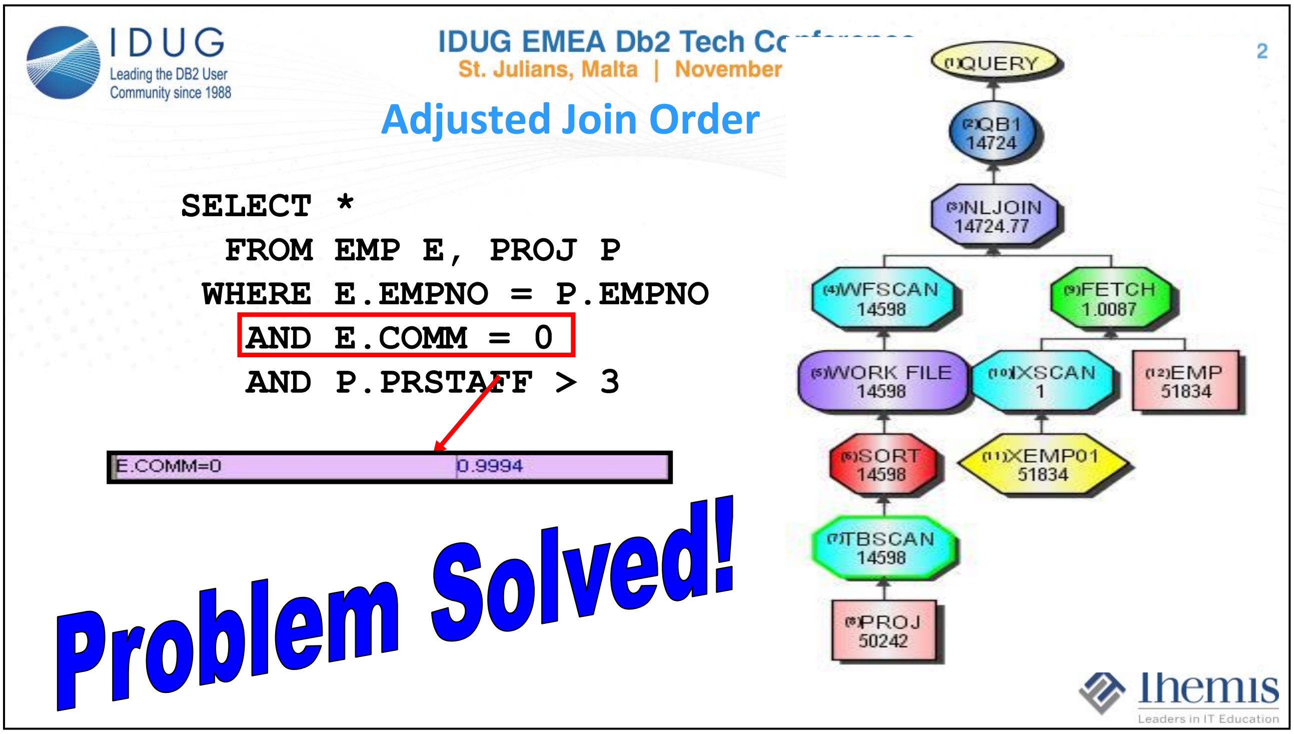
Runstats needed on non uniform distribution of COMM data

Themis
Leaders in IT Education

The syntax presented here will gather column level statistics for the COMM column and also gather the most frequently occurring value for COMM (COLCOUNT 1). Note that runstats must sort the data to obtain this information so sort parameters may need to be included in the syntax.

Also notice that only a count of 1 is needed since there is only 1 value that contains 99% of the data. For developers in coding predicate logic using this column, they would then need 2 different queries:

```
If :COMM-HV = 0  then
 SELECT …….
 FROM EMP
 WHERE …….
   AND COMM = 0;
ELSE
 SELECT …….
 FROM EMP
 WHERE …….
   AND COMM = :HV-COMM
   AND COMM <> 0;
END IF;
```



IDUG EMEA Db2 Tech Co

St. Julians, Malta | November

Adjusted Join Order

```
SELECT *
  FROM EMP E, PROJ P
 WHERE E.EMPNO = P.EMPNO
   AND E.COMM = 0
   AND P.PRSTAFF > 3
```

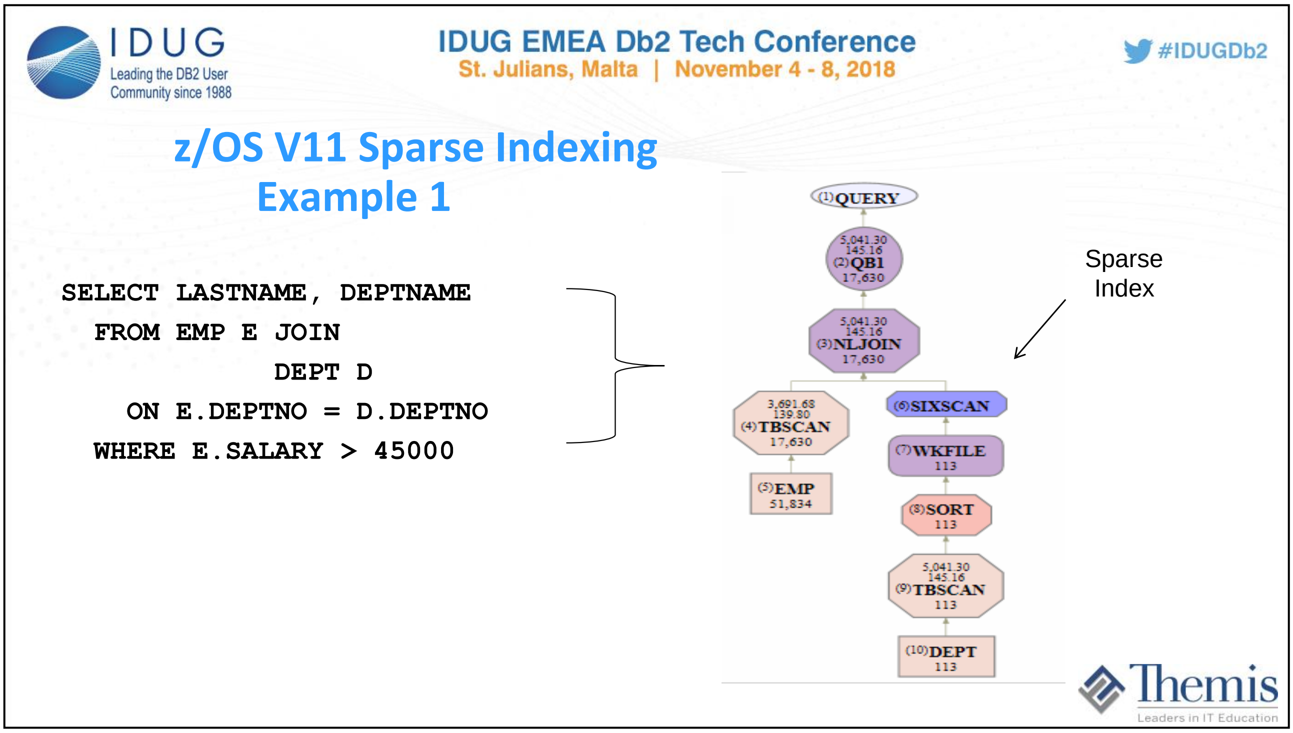
E.COMM=0	0.9994

Problem Solved!

When the additional statistics are provided, the optimizer understands the true nature of the data on the EMP table and chooses the PROJ table to be the first accessed since it provides much better filtering.

Again, if different values come in for COMM and they are put into a host variable, then whenever the value is > 0, the following would need to be executed:

```
SELECT …….
 FROM EMP
 WHERE …….
   AND COMM = :HV-COMM
   AND COMM <> 0;
```



IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS V11 Sparse Indexing Example 1

```
SELECT LASTNAME, DEPTNAME
  FROM EMP E JOIN
            DEPT D
    ON E.DEPTNO = D.DEPTNO
  WHERE E.SALARY > 45000
```

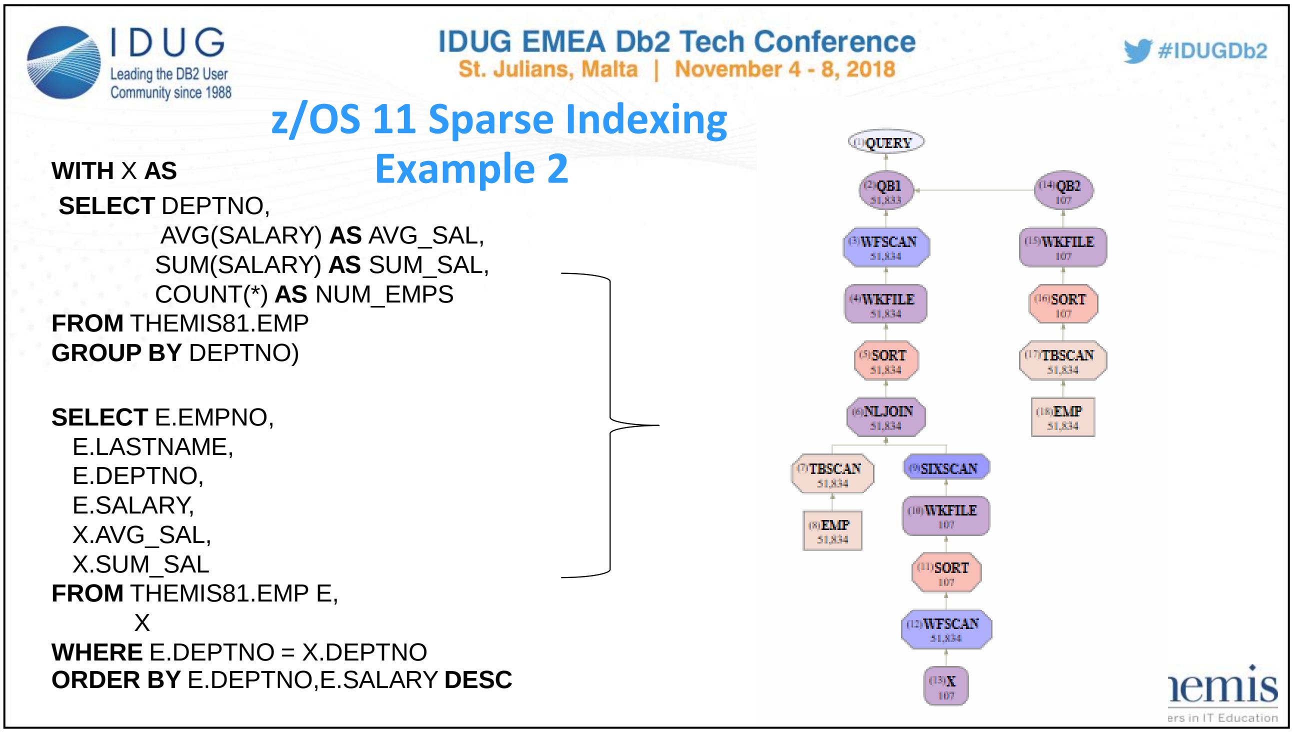
Sparse Index

z/OS sparse index processing is similar to hash joining on other platforms (like DB2 LUW). This is usually a good thing that the optimizer chooses.

The index is built in memory (called In-Memory-Data-Cache). Could overflow to a work file if the entries in the sparse index are too many.

Sparse index/hash join can beat out nested loop if the inner table (after any local predicates) can be contained in-memory (default for MXDTCACH is 20MB), and the join has enough rows from the outer to inner to "pay back" the build cost of the sparse index/hash.

The benefit of sparse index processing is that each lookup to the inner (using the hash of the key) happens in RDS (stage 2). Probing the inner table index (even if index only) has higher overhead than probing the hashed sparse index.



IDUG Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

z/OS 11 Sparse Indexing Example 2

```
WITH X AS
 SELECT DEPTNO,
             AVG(SALARY) AS AVG_SAL,
            SUM(SALARY) AS SUM_SAL,
            COUNT(*) AS NUM_EMPS
FROM THEMIS81.EMP
GROUP BY DEPTNO)

SELECT E.EMPNO,
   E.LASTNAME,
   E.DEPTNO,
   E.SALARY,
   X.AVG_SAL,
   X.SUM_SAL
FROM THEMIS81.EMP E,
         X
WHERE E.DEPTNO = X.DEPTNO
ORDER BY E.DEPTNO,E.SALARY DESC
```

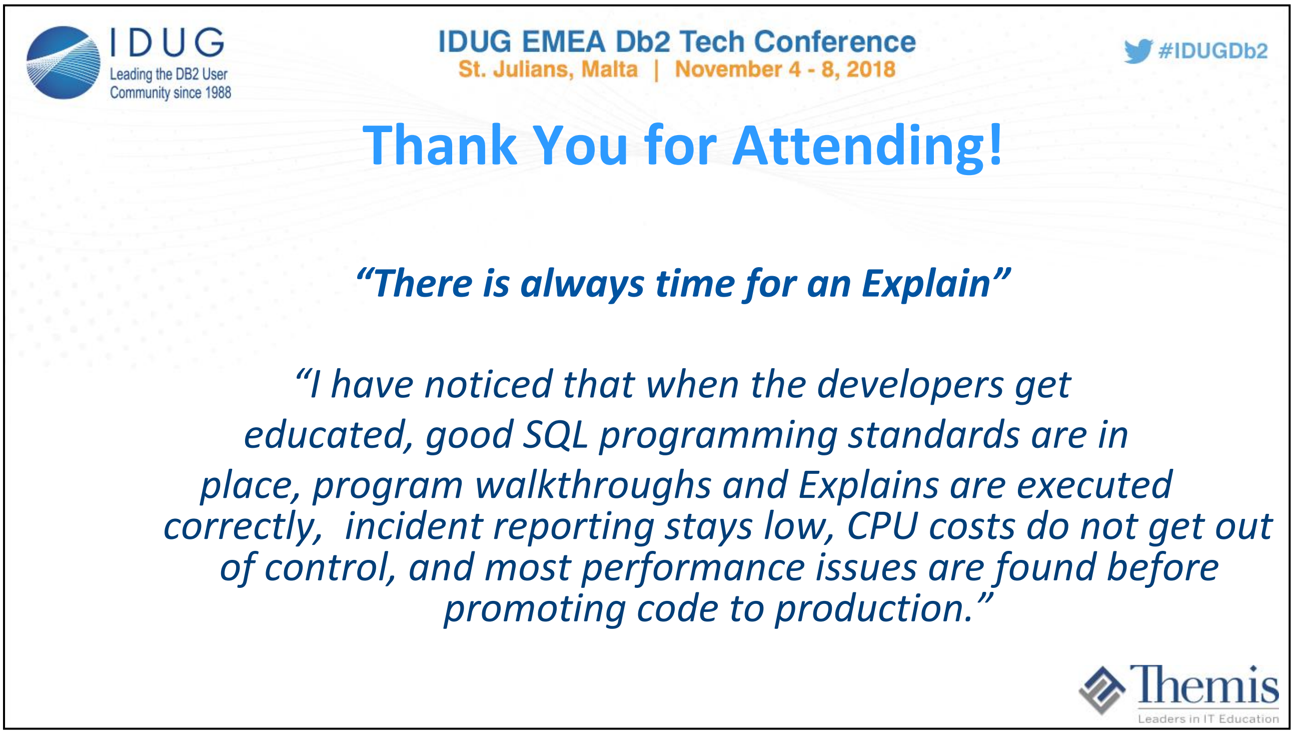
ıemis
ers in IT Education

z/OS sparse index processing is similar to hash joining on other platforms (like DB2 LUW). This is usually a good thing that the optimizer chooses.

The index is built in memory (called In-Memory-Data-Cache). Could overflow to a work file if the entries in the sparse index are too many.

Sparse index/hash join can beat out nested loop if the inner table (after any local predicates) can be contained in-memory (default for MXDTCACH is 20MB), and the join has enough rows from the outer to inner to "pay back" the build cost of the sparse index/hash.

The benefit of sparse index processing is that each lookup to the inner (using the hash of the key) happens in RDS (stage 2). Probing the inner table index (even if index only) has higher overhead than probing the hashed sparse index.



IDUG
Leading the DB2 User Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

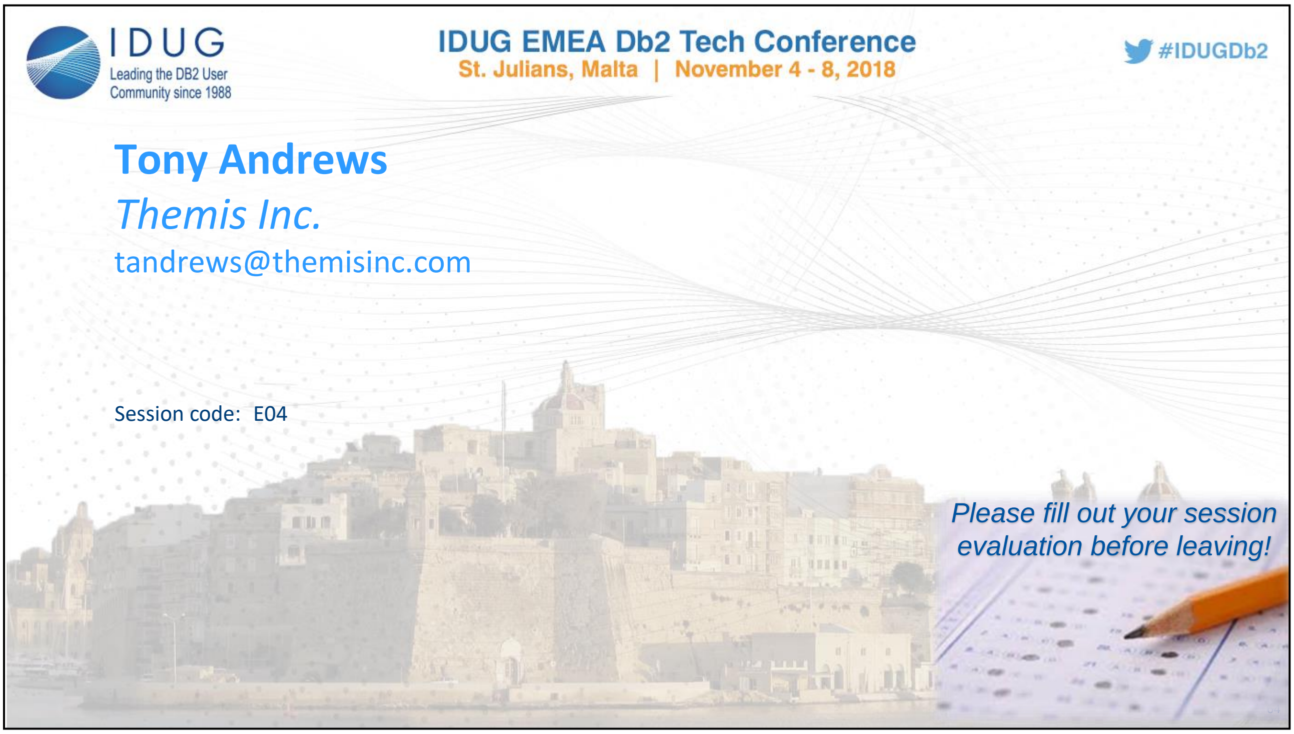
#IDUGDb2

Thank You for Attending!

"There is always time for an Explain"

"I have noticed that when the developers get educated, good SQL programming standards are in place, program walkthroughs and Explains are executed correctly, incident reporting stays low, CPU costs do not get out of control, and most performance issues are found before promoting code to production."

Themis
Leaders in IT Education



IDUG
Leading the DB2 User
Community since 1988

IDUG EMEA Db2 Tech Conference
St. Julians, Malta | November 4 - 8, 2018

#IDUGDb2

Tony Andrews

Themis Inc.

tandrews@themisinc.com

Session code: E04

Please fill out your session evaluation before leaving!