

# Quick Wins on Improving SQL Performance

**Kent Milligan**  
Db2 for i Consultant  
**IBM Lab Services**  
*Power Systems Delivery Practice*  
kmill@us.ibm.com

© 2021 IBM Corporation

1



## Agenda

- Some better SQL
- Indexing

2

© 2021 IBM Corporation

2



## SQL Coding Considerations

- SQL is a very flexible language for data access
- The database does a good job of generalizing behavior across different coding styles
- But there are some general considerations for performance

3

© 2021 IBM Corporation

3



## Conditional Logic part 1

- CASE statements are great in SELECT lists but avoid CASE statements in predicates in the WHERE or ON clauses
- Ex:

```
SELECT ... FROM table
WHERE country =
    CASE
        WHEN :countryvar=' ' THEN country
        ELSE :countryvar
    END
```

- This CASE statement resolves to
  - country=:countryvar --OR--- country=country
- This basically disabled the use of the primary key index!
- Avoid conditional logic on any predicate!

4

© 2021 IBM Corporation

4



## Conditional Logic part 2

- Avoid conditional enabling logic in the WHERE or ON clauses

- Ex:

```
SELECT ... FROM table
WHERE (:hvc = 0 or country = :hvcountry)
      AND (:hvs = 0 or state = :hvstate)
      AND (:hvt = 0 or city = :hvtown)
...
```

- It may seem like a good way to conditionally enable predicates, but it comes at the cost of performance
- Use dynamic SQL to construct what is really 'active'

5

© 2021 IBM Corporation

5



## Transformational Logic

- Avoid UDF's on the WHERE and ON clauses

```
SELECT ...
FROM table1 INNER JOIN table2
      ON CYYMMDDtoDate(t1a) = YYYYMMDDtoDate(t2b)
```

- Be careful with transformational logic on join operations!
- BTW, this could be a good use case for a date dimension table

6

© 2021 IBM Corporation

6



## Writing Better SQL - Blank Checks

- Check for blanks this way:
  - WHERE character\_column = "
- Don't try to "help", trimming and padding is not helpful:
  - WHERE TRIM(TRAILING FROM character\_column) = "
  - WHERE LENGTH(RTRIM(character\_column)) = 0
  - WHERE character\_column = ' '
- Functions may be slower and can get in the way of the optimizer
- Keep it simple!
  - Let the database pad as it needs to
  - Easier to read
  - Better performance

7

© 2021 IBM Corporation

7



## Static versus Dynamic SQL

- People often think Static SQL is faster because Dynamic SQL statements must be parsed and prepared at run time
- In most environments, the performance difference is less than you think
- Dynamic is more flexible, letting you avoid conditional logic
- Static SQL is particularly problematic for "multitenancy" environments
  - Many instances of the same files in different libraries, using changing library list to resolve

8

© 2021 IBM Corporation

8

## Static and Dynamic compared



### Static SQL

```
EXEC SQL DECLARE static_csr1 CURSOR
FOR
  SELECT empno, lastname, salary
  FROM employee
  WHERE workdept =
    :v_workdept;
...

EXEC SQL OPEN static_csr1;
...

EXEC SQL FETCH static_csr1 INTO
  :v_empno, :v_lastname, :v_salary;
...
EXEC SQL CLOSE static_csr1;
```

### Dynamic SQL – Parameter Markers

```
EXEC SQL DECLARE dyn_csr1
CURSOR FOR dyn_stmt1;
...
dyn_stmt1=
'SELECT empno, lastname, salary
FROM employee
WHERE workdept = ?';
...

EXEC SQL PREPARE
  dyn_stmt1 FROM :dyn_stmt1;
...

EXEC SQL OPEN dyn_csr1
  USING :v_workdept;
...
EXEC SQL FETCH dyn_csr1 INTO
  :v_empno, :v_lastname, :v_salary;
...
EXEC SQL CLOSE dyn_csr1;
```

9

© 2021 IBM Corporation

9

# Indexing



## A Query Performance Critical Success Factor

10

© 2021 IBM Corporation

10

## Db2 for i



### Indexes?

- Two types of indexing technologies are supported
  - **Radix** Index. ← *The default unless you explicitly say otherwise*
  - **Encoded Vector** Index (EVI)
- Each type of index has specific uses and advantages
  - **Radix** is the best general purpose index
- Respective indexing technologies complement each other
  - EVI should be considered a more advanced topic

Why create indexes?  
To improve performance!

11

© 2021 IBM Corporation

11

## Creating Indexes



- CREATE INDEX SQL statement  
`CREATE INDEX MY_IX on MY_TABLE (KEY1, KEY2)`
- CREATE ENCODED VECTOR INDEX SQL statement  
`CREATE ENCODED VECTOR INDEX MY_EVI on MY_TABLE (KEY1)`
- IBM i ACS – Database graphical interface
- Underlying Primary Key, Foreign Key and Unique Key Constraints
  - CREATE TABLE
  - ALTER TABLE
  - ADDPFCST
- Also under the covers for keyed CRTPF and CRTLF files

12

© 2021 IBM Corporation

12

## Db2 for i



### Usage

- Indexes can be used by the database for implementation and statistics
- Indexes can provide both row numbers (RRNs) and data
- Indexes are probed or scanned
  - Probe can only occur on contiguous, leading key columns
  - Scan can occur on any key column
  - Probe and scan can be used together
- Indexes are used primarily for SQL to reduce the number of rows processed for the query (WHERE and ON clauses)
  - Secondary, lesser purpose is to order the data

Again, why create indexes?  
To improve performance!

13

© 2021 IBM Corporation

13

## Understanding Indexes



- An index contains data ordered by its key
  - Enables Index Probe – fast lookup
- An index remembers the underlying table's row for each key
  - Index *usually* used in concert with access to underlying table
- An index holds data
  - Sometimes data can be retrieved directly from the index, avoiding I/O on the underlying table
    - ✓ Known as index only access
- An index has entries for only active rows in the underlying table
  - Deleted table rows are not represented in an index
- One or more indexes can be used for each table in a given query
  - Even mixing and matching Radix and EVI is supported

14

© 2021 IBM Corporation

14

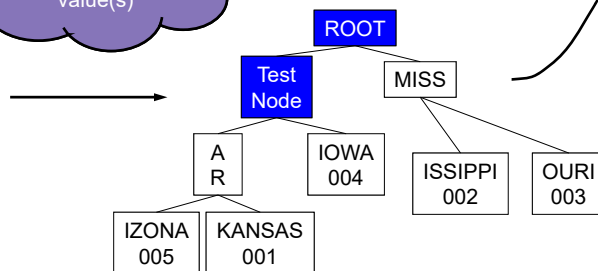
## The Money Case - Index Probe Example

Given an index on table Employees keyed on STATE column...

**SELECT COUNT(\*)**  
**FROM employees**  
**WHERE state = 'IOWA'**

Perform a probe into the range using the local selection value(s)

Index on STATE



Employees table

RRN	STATE
001	ARKANSAS
002	MISSISSIPPI
003	MISSOURI
<b>004</b>	<b>IOWA</b>
005	ARIZONA
006	MONTANA
007	IOWA
008	NEBRASKA
009	NEBRASKA
010	IOWA
011	KANSAS
012	WISCONSIN
013	MISSISSIPPI
014	WISCONSIN
015	WISCONSIN
016	ARKANSAS
017	IOWA

15

© 2021 IBM Corporation

15

## Indexing Strategy

### Radix Indexes

- Common local selection columns (WHERE clause)
- Join columns
- Local selection columns + join columns
- Local selection columns + grouping columns
- Local selection columns + ordering columns

} Minimum

**Note: Columns used with equal conditions are first in key list**

16

© 2021 IBM Corporation

16



## Indexing Strategy - Examples



```
-- Query 1
SELECT A.CUSTOMER_NO, A.ORDER_DATE, A.QUANTITY
FROM      ORDERS A
WHERE A.CUSTOMER_NO = 0112358
AND      A.ITEM_ID = 'ABC123YXZ';

CREATE INDEX ORDERS_IX1 ON ORDERS (CUSTOMER_NO, ITEM_ID);

-- Query 2
SELECT      A.CUSTOMER_NO, B.CUSTOMER, A.ORDER_DATE, A.QUANTITY
FROM      ORDERS A,
          CUSTOMERS B,
          ITEMS C
WHERE      A.CUSTKEY = B.CUSTKEY
AND        A.ITEMKEY = C.ITEMKEY
AND        A.CUSTOMER_NO = 0112358;

CREATE INDEX ORDERS_IX2a ON ORDERS (CUSTOMER_NO, CUSTKEY);
CREATE INDEX ORDERS_IX2b ON ORDERS (CUSTOMER_NO, ITEMKEY);
CREATE INDEX CUSTOMERS_IX2 ON CUSTOMERS (CUSTKEY);
CREATE INDEX ITEMS_IX2 ON ITEMS (ITEMKEY);
```

17

© 2021 IBM Corporation

17

## Extra Credit - Index with Derived Keys



- Creation of indexes with *derived* keys via SQL
 

```
CREATE INDEX ORDERPRIORITYUPPER ON T1
      (UPPER(ORDERPRIORITY) AS UORDERPRIORITY ASC);

CREATE INDEX YEARQTR ON T1
      (YEAR(ORDERDATE) AS ORDYEAR ASC, QUARTER(ORDERDATE)
      AS ORDQTR ASC);

CREATE INDEX TOTALEXTENDEDPRICE ON T1
      (QUANTITY * EXTENDEDPRICE AS TOTEXTPRICE ASC);
```
- Examples where derived indexes may be useful
  - Case insensitive searches – maybe use sort sequence!
  - Data extracted from a column (SUBSTR, YEAR, MONTH...)
  - Derive Common Grouping columns (YEAR(ORDERDATE)...)
    - Results of operations ( COL1+COL2 , QTY \* COST,...)
  - Might be useful to allow *index only access* in more cases

18

© 2021 IBM Corporation

18

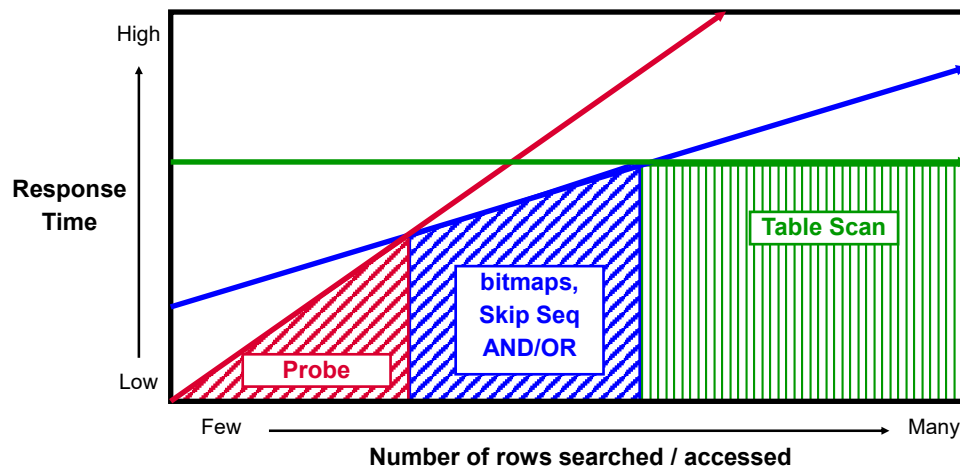
## Query Optimization (using indexes)

19

### Data Access Methods



Cost based optimization dictates that the fastest access method for a given table will vary based upon selectivity of the query

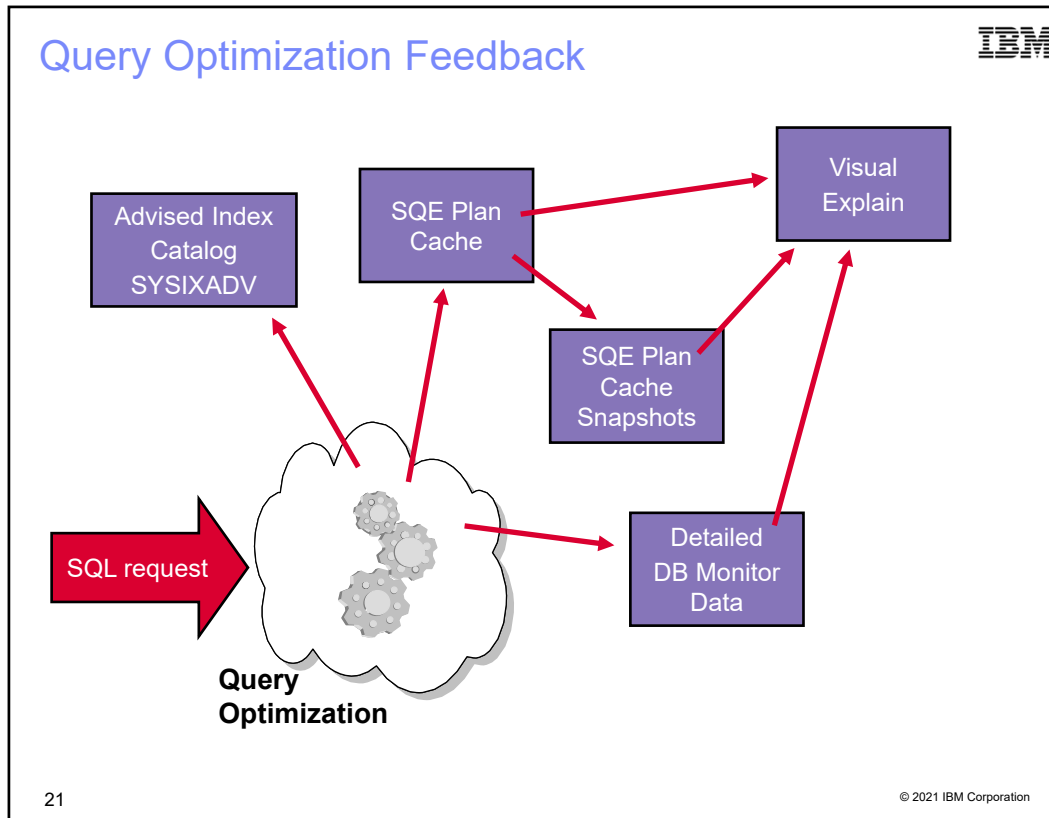


**\*\*Indexing White Paper:** [ibm.biz/db2Papers](http://ibm.biz/db2Papers)

20

© 2021 IBM Corporation

20



21

## Index Advice from the Query Optimizer

- Both Radix and EVI indexes advised (Radix advised more frequently)
  - Avoid EVI index creation until you understand their usage
- Advice based on all parts of the query (selection, join, grouping, ordering)
- Multiple indexes can be advised for the same query
- Advice is limited to columns (fields) only. No derivations/expressions
- Created Temporary Index creation (MTI) also provides some insight
- Advice can be 'Condensed' into a smaller, more general purpose set of indexes

22

© 2021 IBM Corporation

22



## Plan Cache, Scripts and Visual Explain

© 2021 IBM Corporation

23

## Where to start?



IBM i Access Client Solutions

SQL Performance Center - CTCV71.rchland.ibm.com

File View Actions Tools Help

Database: CTCV71

Plan Cache Performance Monitors Plan Cache Snapshots Plan Cache Event Monitors

Show Statements... Change Configuration... SQL Details for Jobs...

Properties

Description	Value	Value Unit
Time Of Summary	2018-10-06-11:23:57.13188	
Plan Cache Creation Time	2018-06-14-15:17:24.001252	

SQL Details for Jobs - CTCV71.rchland.ibm.com

Filters

Job name: All names Apply

Job user: All users Reset All Filters

Job number: All numbers

Current user: BESTGEN Browse...

Type	User	Number	Detailed Status	Subsystem	Cu
Interactive	BESTGEN	086927	Waiting for workstation I/O	QINTER	BES
Interactive	BESTGEN	086926	Waiting for workstation I/O	QINTER	BES
Batch	QUSER	086517	Waiting for time interval	QUSRWRK	BES

Untitled\* - Run SQL Scripts - CTCV71.rchland.ibm.com

File Edit View Run VisualExplain Monitor Options

```
1 select * from qsys2.systables
```

© 2021 IBM Corporation

24

24



SQL Plan Cache Statements - db2icoe2.rchland.ibm.com(Db2icoe2)

Filters to apply:

☐ Minimum runtime for the longest execution of the statement:  
1 Seconds

☐ Statements that ran on or after this date and time:  
July 18, 2017 3:49:36 PM

☐ Top 'n' most frequently run statements  
25

☐ Top 'n' statements with the largest total accumulated runtime:  
25

☐ Statements the following user has ever run:  
DBEPERF01

☐ Statements that are currently active

☐ Statements for which indexes have been advised

☐ Statements for which statistics have been advised

☐ Include statements initiated by the operating system

☐ Statements that reference the following objects:

Schema	Name

☐ Statements that contain the following text:

Reset All Filters Show

Empty

List is initially empty to allow for filtering

Example: filter on top 10 and current user

- ✓ Get the list of plans for the top 10 queries
- ✓ Get the list of plans that the current user has run
- ✓ "AND" the plan lists together
- ✓ Return those plans to the user

If the current user ran one or more of the top 10 plans, those particular plans will be returned.

If the user did not run any of the top 10 plans, nothing is returned.

The top 10 plans for current user is not necessarily returned.

© 2021 IBM Corporation

25

## SQL Plan Cache – Show Statements



SQL Plan Cache Statements - db2icoe2.rchland.ibm.com(Db2icoe2)

Filters to apply:

☐ Minimum runtime for the longest execution of the statement:  
1 Seconds

☐ Statements that ran on or after this date and time:  
July 18, 2017 3:49:36 PM

☐ Top 'n' most frequently run statements  
25

☐ Top 'n' statements with the largest total accumulated runtime:  
25

☒ Statements the following user has ever run:  
DBEPERF01

☐ Statements that are currently active

☐ Statements for which indexes have been advised

☐ Statements for which statistics have been advised

☐ Include statements initiated by the operating system

☐ Statements that reference the following objects:

Schema	Name

☐ Statements that contain the following text:

Reset All Filters Show

Shown at 3:54 PM (1)

Filters applied:  
● Statements that user 'DBEPERF01' has ever run

Last Time Run	Most Expensive Time (sec)	Total Processing Time (sec)	Total Times Run	Average Processing Time (sec)	Statement
2017-07-18 19:00:33...	0.4336	0.4336	1	0.4336	INSERT INTO TEST1B ( SELECT YEAR , QUARTER , SUM (
2017-07-18 19:00:34...	0.3485	0.3485	1	0.3485	INSERT INTO TEST2 ( SELECT YEAR , QUARTER , RETUR
2017-07-18 15:29:14...	0.0912	0.0912	1	0.0912	SELECT d.year, d.week, s.supplier, o.orderkey, o.quant
2017-07-18 19:00:33...	0.0636	0.0636	1	0.0636	INSERT INTO TEST1C ( SELECT YEAR , QUARTER , SUM (
2017-07-18 16:42:04...	0.0516	0.0519	2	0.0259	/* Lab - SQE Plan Cache and Show Current SQL */SELECT
2017-07-18 19:00:34...	0.0303	0.0303	1	0.0303	CREATE TABLE QTEMP , TEMP1 AS ( SELECT T . YEAR , T
2017-07-18 15:28:02...	0.0296	0.0296	1	0.0296	SELECT d.year, d.week, s.supplier, o.orderkey, o.quant
2017-07-18 19:00:32...	0.0218	0.0218	1	0.0218	INSERT INTO TEST1A ( SELECT YEAR , QUARTER , SUM (
2017-07-18 18:43:44...	0.0064	0.0172	12	0.0014	DECLARE X CURSOR FOR SELECT Partition_Name , REPL
2017-07-18 19:00:34...	0.0081	0.0081	1	0.0081	INSERT INTO TEST3 ( SELECT C . CUSTOMER , D . YEAR
2017-07-18 19:00:34...	0.0078	0.0078	1	0.0078	CR
2017-07-18 19:00:34...	0.0077	0.0077	1	0.0077	CR
2017-07-18 15:28:42...	0.0062	0.0062	1	0.0062	SE
2017-07-18 15:28:51...	0.0013	0.0037	3	0.0012	der
2017-07-18 22:25:16...	0.0024	0.0024	1	0.0024	DE
2017-07-18 10:32:05...	0.0022	0.0022	1	0.0022	SE
2017-07-18 15:24:33...	0.0010	0.0019	2	0.0009	de
2017-07-18 22:28:35...	0.0013	0.0013	1	0.0013	de
2017-07-18 22:28:35...	0.0013	0.0013	1	0.0013	deq
2017-07-18 16:37:52...	0.0011	0.0011	1	0.0011	de
2017-07-18 11:55:41...	0.0005	0.0010	2	0.0005	de
2017-07-18 15:29:21...	0.0010	0.0010	1	0.0010	de
2017-07-19 11:43:39...	0.0005	0.0010	2	0.0005	declare C41 cursor for SELECT * FROM QTEMP / QQ80Bv

Done: 76 rows retrieved.

Visual Explain Show Longest Runs Show Active Jobs Show User History Work with SQL Statement Work with SQL Statement and Variables Save to New... Plan

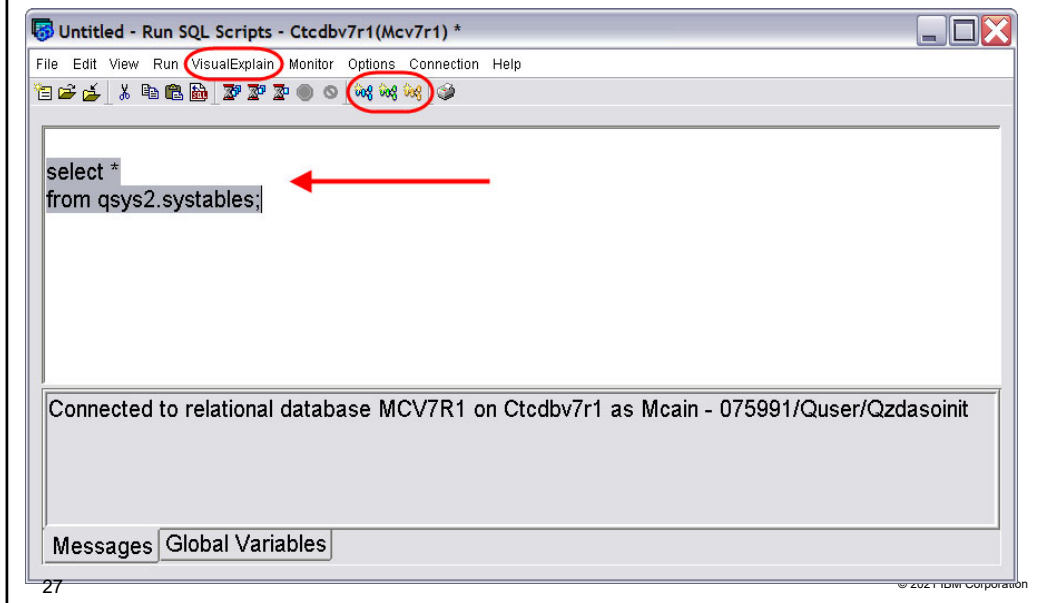
Columns... Save Results... Refresh

© 2021 IBM Corporation

26

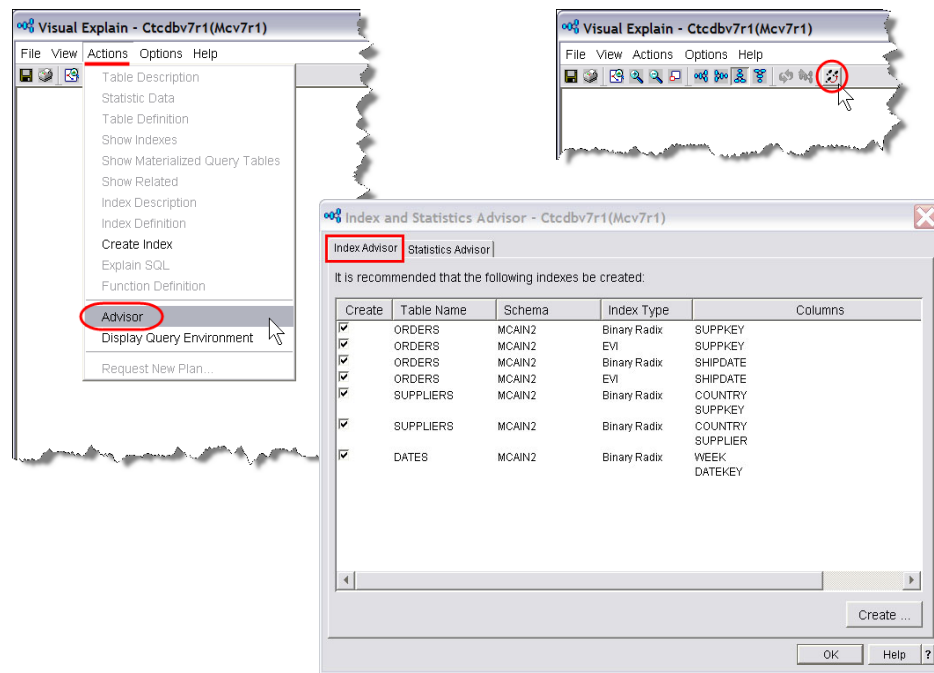
## ACS Run SQL Scripts

- JDBC client that supports dynamic SQL
- Direct performance feedback from Visual Explain



27

## Once in Visual Explain...



28

28

## Remember, it is Advice!



- The optimizer offers index advice
- Use your knowledge and common sense when creating indexes from advice
- Advice is not always perfect
  - But often very useful
- Ultimately you own the indexes that you create, not the optimizer

29

© 2021 IBM Corporation

29



General Index Advice

© 2021 IBM Corporation

30

## Index Advisor



General (not query-specific) advice available at multiple levels:

- System wide
- By schema (library)
- Table (file)

Schemas - DB2ICOE3.rchland.ibm.com

File Edit View Actions Tools

Databases

Table for Which Index was Advised	Schema	Partition	Keys Advised	Advised Index Type	Last Advised for Query Use	Times Advised for Query Use
BOTDIST	QWQREPOS	For all partitions	SCHEDULEID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	272
BOTSIT	QWQREPOS	For all partitions	SCHEDULEID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	269
BOTTASK	QWQREPOS	For all partitions	PACKETID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	269
QZG0000029	QGPL	For all partitions	QQRID, QQC21, QQJFLD, QQI5	Binary Radix	02/16/2021 08:59:22 AM	2

Change Query Attributes...  
Health Center  
Index Advisor  
Run SQL Scripts

31

© 2021 IBM Corporation

31

## Too Much Advice?



### Index Advisor - Condenser

#### Queries:

...WHERE YEAR = 2020 AND QUARTER = 1 AND COLOR = 'BLUE';  
 ...WHERE YEAR = 2020 AND QUARTER = 1;  
 ...WHERE YEAR = 2020;

#### Index advice by query:

YEAR, QUARTER, COLOR  
 YEAR, QUARTER  
 YEAR

#### Condensed advice:

YEAR, QUARTER, COLOR

Open in New Window	
Save List Contents...	Ctrl+S
Columns...	Ctrl+Shift+C
Include...	Ctrl+Shift+I
Refresh	F5
Generate SQL...	
Permissions	
Change Text...	
Index Advisor	Advised Indexes
Delete...	Delete
Rename...	Ctrl+Shift+R
New	Condense Advised Indexes
Properties	Prune Advised Indexes...
	Clear All Advised Indexes...

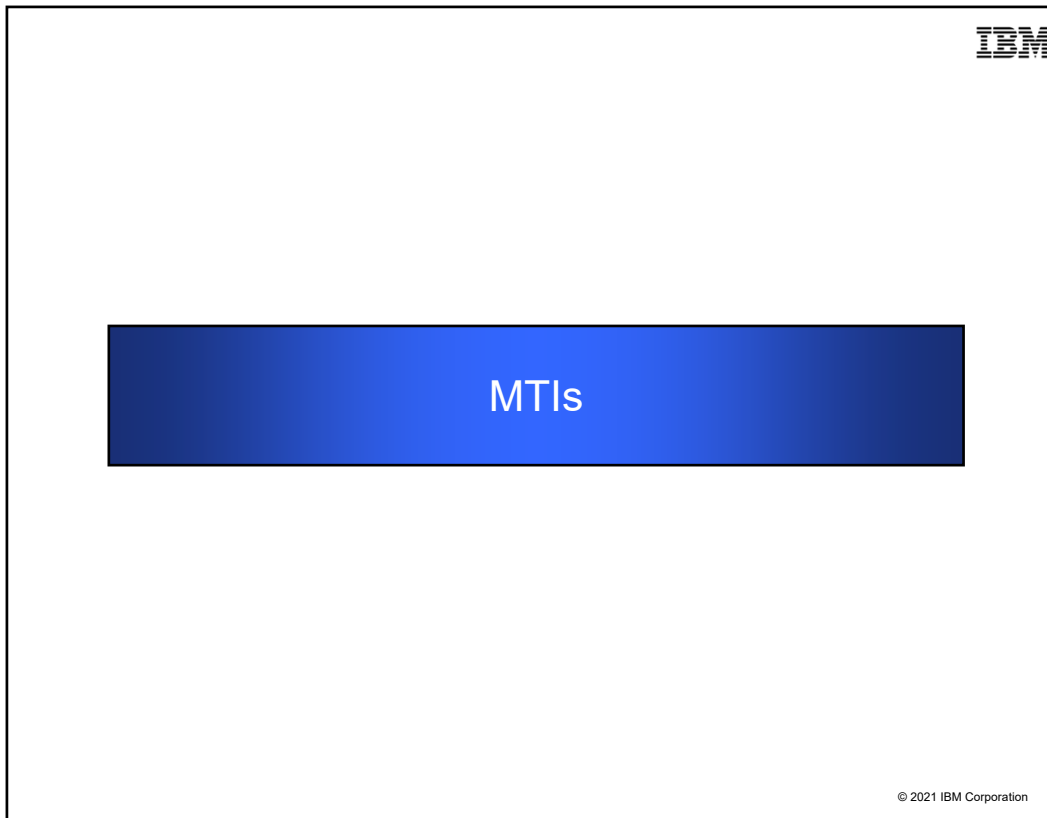
Can be used for all three queries

32

© 2021 IBM Corporation

32






33

### Autonomic (Temporary) Index Creation

- Optimizer may create a temporary index called an **MTI**
- Temporary indexes are *maintained*
- Temporary indexes are not permanent
- Temporary indexes are usually a good indicator that a permanent index should be created

34 

© 2021 IBM Corporation

34

Index Advisor & MTIs

IBM

Customize Columns

Column	Width	Visible
Table for Which Index was Advised	93	<input checked="" type="checkbox"/>
System Name	78	<input checked="" type="checkbox"/>

Move Up

Move Down

Table for Which Index was Advised	Schema	Partition	Keys Advised	Advised Index Type	Last Advised for Query Use	Times Advised for Query Use	MTI Used	MTI Created	MTI Last Used
BOTDIST	QWQREPOS	For all partitions	SCHEDULEID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	272	214	11 02/18/2021 08:30:00 PM	
BOTSIT	QWQREPOS	For all partitions	SCHEDULEID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	269	210	11 02/18/2021 08:30:00 PM	
BOTTASK	QWQREPOS	For all partitions	PACKETID, COUNTER	Binary Radix	02/18/2021 08:31:10 PM	269	210	11 02/18/2021 08:30:00 PM	
QZG0000029	QGFL	For all partitions	QQRID, QQCZ1, QQJFLD, QQIS	Binary Radix	02/16/2021 08:59:22 AM	2	2	1 02/16/2021 08:58:19 AM	

was Advised

Name

Estimated Index Creation Time

90

☒

QINAVMNTCMD

QINAV

Reason Advised

91

☒

UQA\_CACHECO...

UQA\_C

Logical Page Size Advised

81

☒

QINAVMNTRG

QINAV

Most Expensive Query Estimate

92

☒

BOTSIT

BOTSIT

Average of Query Estimates

130

☒

BOTSIT

BOTSIT

Rows in Table when Advised

109

☒

BOTSCHED

BOTSIT

NLSS Table Advised

81

☒

QA1ANET2

QA1A

NLSS Schema Advised

81

☒

QINAVMNTRG

QINAV

MTI Used

67

☒

MTI Created

82

☒

MTI Last Used

67

☒

MTI Used for Statistics

86

☒

MTI Last Used for Statistics

86

☒

EVI Distinct Values

78

☒

First Advised for Query Use

134

☒

Times Advised Dependent on Other Advice

142

☒

OK

Apply

Cancel

100 rows displayed (more)

Width (pixels): 93

Hide

Advised Index Type

ary Radix

ary Radix

ary Radix

ary Radix

ary Radix

ary Radix

ary Radix

© 2021 IBM Corporation

35

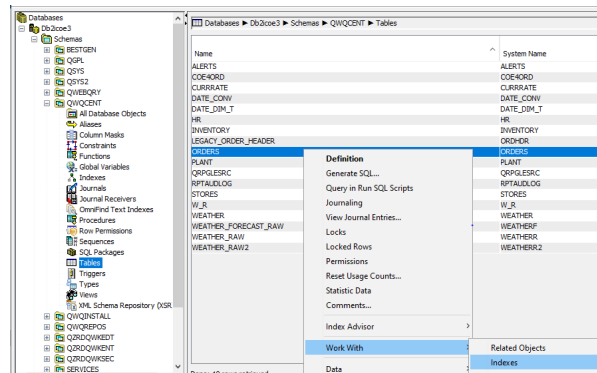
IBM

Existing Indexes

© 2021 IBM Corporation

36

## Index Evaluator (Work With->Indexes)



Meta data for  
Indexes...

File Edit View Actions

Indexes for QWQCENT.ORDERS

Name	Schema	Type	Date Created	Last Query Use	Last Query Statistics Use	Query Use Count	Query Statistics Use Count
"1 MAINTAINED TEMPORARY INDEXES"							
BODTEST_ORDERS	JIMBAINB	Index	06/13/2018 11:55:11 AM	06/13/2018 11:55:15 AM	09/09/2019 09:24:15 AM	0	0
Q_QWQCENT_ORDERS_PLANT_CODE_00001	QWQCENT	Foreign Key Constr...	06/18/2012 11:43:46 AM	12/15/2016 01:56:44 AM	03/09/2020 03:28:05 PM	1	1,757
Q_QWQCENT_ORDERS_PROD_NUM_00001	QWQCENT	Foreign Key Constr...	06/18/2012 11:43:46 AM	01/30/2020 03:00:29 PM	02/20/2020 07:03:21 PM	232	2,658
Q_QWQCENT_ORDERS_STORE_CODE_00001	QWQCENT	Foreign Key Constr...	06/18/2012 11:43:46 AM	02/20/2020 07:03:21 PM	02/20/2020 07:03:21 PM	940	1,987
RRINDEX	JIMBAINB	Index	03/27/2018 09:20:55 AM			0	0
TESTCAST1	JIMBAINB	Index	12/13/2017 03:47:45 PM	08/22/2018 05:07:50 PM	02/20/2020 09:43:31 AM	2	69

37

© 2021 IBM Corporation

37



Making Choices

© 2021 IBM Corporation

38



# What should you do?

Create all advised indexes?

Nothing, let the system handle it?

Monitor, analyze, and tune important tables and queries?

39

© 2021 IBM Corporation

39



## The Process of Identifying Indexes

### Proactive method

- Analyze the data model, application and SQL requests
- Add Database Primary Key, Unique Key and Referential Integrity constraints

### Reactive method

- Rely on optimizer feedback and actual implementation methods
- Rely on SQE's ability to auto tune using temporary indexes

### Understand the data being queried

- Column selectivity
- Column cardinality

40

© 2021 IBM Corporation

40



## Db2 for i Lab Services Offerings

<http://ibm.biz/db2ilabservices>

### Db2 for i Best Practices Consulting

Database Engineer (DBE) Enablement (tailored skills transfer)  
 SQL/Database Performance Assessment for Db2 for i  
 Planning for Growth (Very Large Database) Support  
 Planning for Db2 for i Database Modernization  
 Planning for Db2 for i Row and Column Access Control  
 Planning for Db2 for i Temporal Table Implementations

### Analytics

Query/400 Modernization Services  
 Db2 for i Analytics Discovery Workshop  
 Db2 Web Query Getting Started  
 Db2 for i Web Query DataMigrator ETL Getting Started

Contact Doug Mack at [mackd@us.ibm.com](mailto:mackd@us.ibm.com)

© 2021 IBM Corporation

41



42

© 2021 IBM Corporation

42