

May 6-10, 2007

San Jose Convention Center

San Jose, California, USA

Session: G06

More Ways to Challenge the DB2 z/OS Optimizer

IDUG® 2007

North America

Terry Purcell
IBM Silicon Valley Lab

Tuesday May 8th, 2007 10:40 a.m. – 11:40 a.m.

Platform: DB2 z/OS



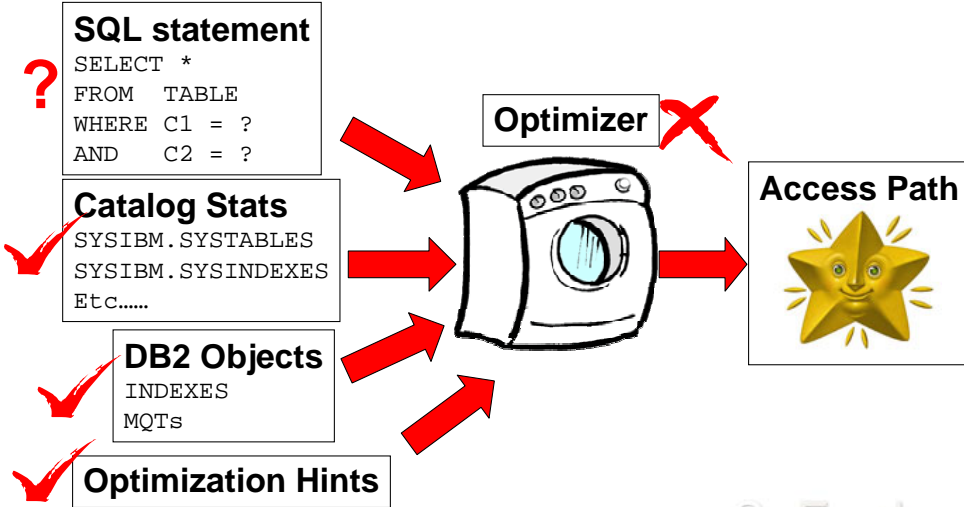
GoFurther



Agenda

- Introduction
- Process for validating the preferred access path
- Filter Factor Challenges
- Predicate Challenges
- Conclusion

What can you control?



Do you want to challenge the optimizer?

- Is the access path for your SQL obvious?
 - Is the optimizer able to apply the filtering early?
 - Are there indexes that support an efficient path?
 - Do statistics allow distinction between the choices?



- If no.....
 - You're more likely to have performance regressions
 - From minor changes in statistics
 - At release or maintenance upgrade times



- **So how do we challenge the optimizer less?**

GoFurther

Agenda

- Introduction
- Process for validating the preferred access path
- Filter Factor Challenges
- Predicate Challenges
- Conclusion

Verifying the optimizer's choice

- What access path would you like to see?
 - Run counts per object
 - Per table
 - Per index
 - Per predicate
 - To verify the optimal join sequence and index(es)
- Compare this to optimizer's chosen access path

SQL Statement

- To determine the preferred access path
 - Run counts to see what the data shows the best access path to be.....

```
SELECT *  
FROM CUSTOMERS C, ACCOUNTS A  
WHERE C.CUSTNO = A.CUSTNO  
      AND C.CUSTNO BETWEEN ? AND ?  
      AND C.STATUS = ?  
      AND C.GENDER = ?  
      AND A.OPENDATE > ?  
      AND A.CLOSEDATE < ?
```

7

GoFurther

Query Count

- Run a count of rows qualified from entire query
 - Requires knowledge of literal values

```
SELECT COUNT(*) = 6232 rows
FROM CUSTOMERS C, ACCOUNTS A
WHERE C.CUSTNO = A.CUSTNO
      AND C.CUSTNO BETWEEN 1 AND 100000
      AND C.STATUS = 'Y'
      AND C.GENDER = 'F'
      AND A.OPENDATE > '19600101'
      AND A.CLOSEDATE < '20070101'
```


Per Table Count

- Break apart the SQL per object
 - 1st level of granularity is per table

```
SELECT COUNT(*)
FROM CUSTOMERS C
WHERE C.CUSTNO BETWEEN 1 AND 100000
      AND C.STATUS = 'Y'
      AND C.GENDER = 'F'
```

Is this all the
single table
predicates???

```
SELECT COUNT(*)
FROM ACCOUNTS A
WHERE A.OPENDATE > '19600101'
      AND A.CLOSEDATE < '20070101'
```

,

GoFurther

Optimizer Generated Predicates

- To ensure the “per object” counts are correct
 - Add all optimizer generated predicates
 - Given.....

```
WHERE C.CUSTNO = A.CUSTNO  
      AND C.CUSTNO BETWEEN 1 AND 100000
```

- Optimizer will generate
AND A.CUSTNO BETWEEN 1 AND 100000
- Known as “Predicate Transitive Closure” (PTC)
 - Applies to =, <, <=, >, >=, LIKE, BETWEEN

Determining Generated Predicates

- DB2 V8 Visual Explain
 - Shows existence of predicate
 - (But not that it was generated)

Extract from Query Graph Detail

Stage 1 Predicates	
●	A.OPENDATE>'19600101'
●	A.CLOSEDATE<'20070101'
●	A.CUSTNO BETWEEN 1 AND 100000
<input checked="" type="checkbox"/> Show attribute explanation Views: Cost	
Name	
Input Cardinality	
Scanned Rows	
Stage 1 Predicates	
	A.OPENDATE>'19600101'
	A.CLOSEDATE<'20070101'
	A.CUSTNO BETWEEN 1 AND 100000

Extract from Predicate Report

Predicate Text
A.CUSTNO BETWEEN 1 AND 100000

Determining Generated Predicates

- DB2 9 OSC
 - Shows existence of predicate
 - (And that it was generated)

Stage1_Predicates

- @ A.OPENDATE > '19600101'
- @ A.CLOSEDATE < '20070101'
- @ A.CUSTNO BETWEEN 1 AND 100000

Stage 1 Predicates

- A.OPENDATE > '19600101'
- A.CLOSEDATE < '20070101'
- A.CUSTNO BETWEEN 1 AND 100000

Extract from Query
Graph Detail

Extract from
Predicate Report

TABLE	COLNAME	TYPE	BT	SI	JN	AJ	PTC
ADMFO01.ACCOUNTS	CUSTNO	BETWEEN	Y	Y	N	N	Y

Predicate Transitive Closure = 'Y'

GoFurther

OSC Predicate Report Notes

- Legend of OSC predicate report abbreviations

Predicate Report

Legend for column names that have been truncated	
FF	--Filter factor
BT	--Whether this predicate is a boolean term predicate
S1	--Whether the predicate is a stage 1 predicate
JN	--Whether this predicate is a join predicate
AJ	--Whether the predicate evaluation phase is after the join
PTC	--Whether the predicate is generated by transitive closure, which means that DB2 generates additional predicates to provide more information for access path selection when the set of predicates that belong to a query logically imply other predicates
MARKER	--Whether this predicate includes host variables, parameter markers, or special registers
PREDNO	--The predicate number

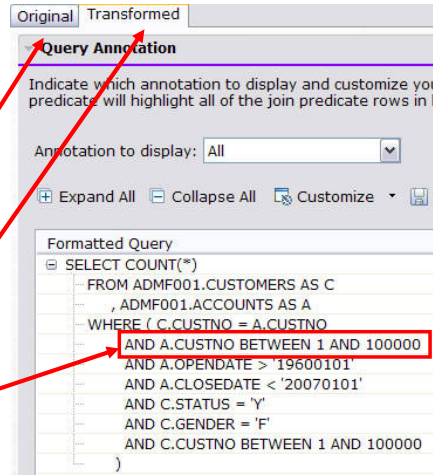


Determining Generated Predicates

- New for DB2 9 OSC
 - Query Annotation shows the transformed query

Click-on Original or Transformed

New predicate appears in transformed query



Per Table Count

- With generated predicates

```
SELECT COUNT(*) = 5308 rows
FROM CUSTOMERS C (cardf = 192960)
WHERE C.CUSTNO BETWEEN 1 AND 100000
      AND C.STATUS = 'Y'
      AND C.GENDER = 'F'
```

Preferred Table
as leading

```
SELECT COUNT(*) = 86959 rows
FROM ACCOUNTS A (cardf = 256663)
WHERE A.CUSTNO BETWEEN 1 AND 100000
      AND A.OPENDATE > '19600101'
      AND A.CLOSEDATE < '20070101'
```

Use of Table Counts

- Table counts can be directly compared to the optimizer's estimate from Table Report
 - 9 OSC QUALROWS field
 - V8 VE Qualified Rows field

Table Report

Legend for column names that have been truncated	
PARTS	-- Number of partitions in tablespace
QUALROWS	-- Optimizer's estimate for how many rows qualify if this table were the outer table
CLU	-- Is the index the clustering index?
UR	-- Unique rule
CR	-- Clusterratiof

Per Index Count – 1st Table

- Break-apart 1st table query further per object (index)
 - Repeat for 2nd table.

```
SELECT COUNT(*) = 100000 rows
FROM CUSTOMERS C (cardf = 192960)
WHERE C.CUSTNO BETWEEN 1 AND 100000
```

```
SELECT COUNT(*) = 22662 rows
FROM CUSTOMERS C (cardf = 192960)
WHERE C.STATUS = 'Y'
AND C.GENDER = 'F'
```

Indexes
CUSTIX1
 (CUSTNO)
CUSTIX2
 (STATUS
 ,GENDER)

Additional Index Comments

- It's not just about % of qualified rows from index
 - But % of qualified data pages dictates the data I/O

CUSTIX1 • Clustering index (Clusterratio = 100%)
(**CUSTNO**) • 51.82% of rows qualified from approx 52% of data pages*

CUSTIX2 • Non-clustering index (Clusterratio = 56.67%)
(**STATUS** • 11.74% of rows qualified, from 76% of data pages*
,GENDER)

- Combined filtering is 2.75%, but no single index support

** 2000/3860 getpages = 51.82%, 2944/3860 getpages = 76.27%

GoFurther

Per Predicate Count – 1st Table

- Break-apart 1st table query further per object (predicate)
 - Repeat for 2nd table.

```
SELECT COUNT(*) = 100000 rows
FROM CUSTOMERS C
WHERE C.CUSTNO BETWEEN 1 AND 100000
```

```
SELECT COUNT(*) = 183058 rows
FROM CUSTOMERS C
WHERE C.STATUS = 'Y'
```

```
SELECT COUNT(*) = 24118 rows
FROM CUSTOMERS C
WHERE C.GENDER = 'F'
```

Use of Predicate Counts

- Simple calculation to turn counts into “actual” FF
 - For comparison with Optimizer FF
 - From Predicate Report (V8 VE or 9 OSC)

Predicate	Count	Table Cardf	Actual FF (count/cardf)
CUSTNO BETWEEN 1 AND 100000	100000	192960	0.5182
STATUS = 'Y'	183058	192960	0.9487
GENDER = 'F'	24118	192960	0.1249

Agenda

- Introduction
- Process for validating the preferred access path
- Filter Factor Challenges
- Predicate Challenges
- Conclusion

Filter Factors

- Optimizer assigns a “Filter Factor” (FF) to each predicate or predicate combination
 - Number between 0 and 1 of estimated filtering
 - 0.25 = 25% of the rows are estimated to qualify

Combining Filter Factors

- Individual Filter Factors (FFs) are combined to determine the total filtering per object
 - AND predicate FFs are multiplied
 - OR predicate FFs are added
- Available statistics determine “degree” of multiplication

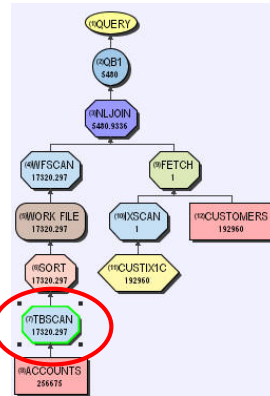
Assigning and combining Filter Factors

- Accuracy of FFs and how to combine them is important for
 - Index matching
 - Total index filtering
 - Total table level filtering
- Therefore.....for each object, the goal is:
 - To accurately assign the individual predicate FFs
 - To correctly combine the individual FFs
- **The more objects involved, the more important for optimizer to be able to distinguish between them**

Optimizer Calculations

- Optimizer will combine all FFs for each object
 - Optimizer chose tablespace scan of ACCOUNTS table 1st
 - Earlier counts show CUSTOMERS as preferred 1st table

Input Cardinality	256675
Scanned Rows	256675
Stage 1 Predicates	Filter Factor
A.OPENDATE>'19600101'	0.9973
A.CLOSEDATE<'20070101'	0.1306
A.CUSTNO BETWEEN 1 AND 100000	0.5182
Stage 1 Returned Rows	17320.297
Stage 2 Returned Rows	17320.297
Output Cardinality	17320.297



$256675 * 0.9973 * 0.1306 * 0.5182 \approx 17320 \text{ rows}$

Comparing Table estimates

- How does the optimizer table estimate compare to reality?

Table	Count	Estimate	
CUSTOMERS	5,308	31,644	← Overestimation
ACCOUNTS	86,959	17,320	← Underestimation

Comparing Predicate Estimates

- Table (and index) estimates are calculated from individual FF estimates
 - So how do these compare with reality?

Predicate	Actual FF	Estimate
C.CUSTNO BETWEEN 1 AND 100000	0.5182	0.5182
STATUS = 'Y'	0.9487	0.9487
GENDER = 'F'	0.1249	0.3333
A.CUSTNO BETWEEN 1 AND 100000	0.5001	0.5182
OPENDATE > '19600101'	1.0000	0.9973
CLOSEDATE < '20070101'	0.6940	0.1306

Reason for overestimation

Reason for underestimation

Equal Predicate Calculations

- Index on STATUS, GENDER
 - RUNSTATS collects leading column FREQVAL by default
 - Optimizer uses this in it's estimate

Predicate	Colcardf	Actual FF	Estimate	SYSCOLDIST CATALOG STATISTICS FOR STATUS
STATUS = 'Y'	2	0.9487	0.9487	FREQUENCY COLVALUE
GENDER = 'F'	3	0.1249	0.3333	-----

```

9.4868366500829E-01 X'00E8'
5.1316334991708E-02 X'00D5'
    
```

- No special RUNSTATS currently on GENDER
 - FF estimate uses $1/\text{colcardf} = 1/3$
 - Run RUNSTATS to collect skew information

Hex 'E8' = Y

```

RUNSTATS TABLESPACE TPTEST.TPTSTTS1
TABLE (ADMF001.CUSTOMERS)
COLGROUP (GENDER) FREQVAL COUNT 10
    
```

Range Predicate Calculations

- Simple calculation for INTEGER column
 - $(\text{Highvalue} - \text{Lowvalue}) / (\text{High2key} - \text{Low2key})$
 - $(100000 - 1) / (192959 - 2) = 0.5182$

Predicate	High2key	Low2key	Actual FF	Estimate
C.CUSTNO BETWEEN 1 AND 100000	192959	2	0.5182	0.5182
A.CUSTNO BETWEEN 1 AND 100000	192959	2	0.5001	0.5182

Range Predicate Calculations

- Challenges with “dates” stored as CHAR(8)
 - Invalid dates are “valid” as numerics or chars

Predicate	Colcardf	High2key	Low2key	Actual FF	Estimate
OPENDATE > '19600101'	372	20150630	19760225	1.0000	0.9973
CLOSEDATE < '20070101'	64	99991231	19940509	0.6940	0.1306

- High2key/Low2key spans a very wide range (in this example)
 - But no data exists between 2015 – 9998
- Collect Frequency stats to assist range estimate

```

RUNSTATS TABLESPACE TPTEST.TPTSTTS2
        TABLE (ADMFO01.ACCOUNTS)
        COLGROUP (CLOSEDATE) FREQVAL COUNT 60
  
```

Estimates after RUNSTATS

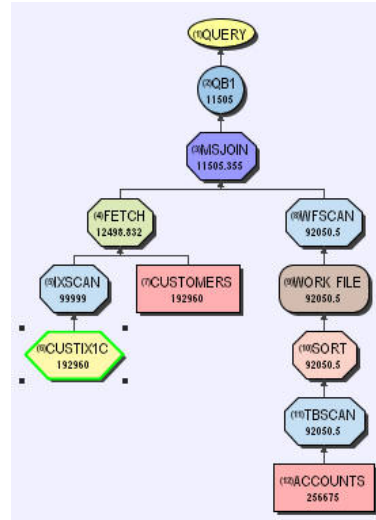
- All estimates now closer to reality

Predicate	Actual FF	Estimate
C.CUSTNO BETWEEN 1 AND 100000	0.5182	0.5182
STATUS = 'Y'	0.9487	0.9487
GENDER = 'F'	0.1249	0.1249
A.CUSTNO BETWEEN 1 AND 100000	0.5001	0.5182
OPENDATE > '19600101'	1.0000	0.9973
CLOSEDATE < '20070101'	0.6940	0.6939

Access Path after RUNSTATS

- Leading table is CUSTOMERS
 - Counts proved this table was preferred

- Literals required to exploit statistics
 - REOPT for host variables / parameter markers



What happens if I use host variables?

- FF estimates use default calculations
 - How do these compare?
 - **None of these are close to reality!!!**

Predicate	Colcardf	Actual FF	Estimate
C.CUSTNO BETWEEN ? AND ?	192960	0.5182	0.001
STATUS = ?	2	0.9487	0.5
GENDER = ?	3	0.1237	0.4126
A.CUSTNO BETWEEN ? AND ?	182272	0.5182	0.001
OPENDATE > ?	372	1.0000	0.1
CLOSEDATE < ?	64	0.6940	0.3333

$1/2 = 0.5$
 $1/2 * (192960 - 33739) = 0.4126$

Note 1: Range/Between predicates use default interpolation (next slide)

Note 2: GENDER excludes 33739 NULL rows from FF calculation

Range Predicate Estimates

- For host vars/parameter markers or default stats
 - Optimizer uses following formulas for range predicates*

COLCARDF	Factor for Op	Factor for LIKE/BETWEEN	Predicate	Colcardf	Estimate
>=100,000,000	1/10,000	3/100,000			
>=10,000,000	1/3,000	1/10,000			
>=1,000,000	1/1,000	3/10,000			
>=100,000	1/300	1/1,000			
>=10,000	1/100	3/1,000	C.CUSTNO BETWEEN ? AND ?	192960	0.001
>=1,000	1/30	1/100	A.CUSTNO BETWEEN ? AND ?	182272	0.001
>=100	1/10	3/100	OPENDATE > ?	372	0.1
>=2	1/3	1/10	CLOSEDATE < ?	64	0.3333
=1	1	1			
<=0	1/3	1/10			

Note: Op is one of these operators: <, <=, >, >=.

*Table documented in Admin Guide



What if I don't run RUNSTATS at all?

- Every object looks the same to optimizer

TABLE	CARDF	NPAGESF	TABNO	QUALROWS	
ADMF001.ACCOUNTS	-1	-1	2	111.111145	
INDEX	CLU	UR	NLEAF	NLEVEL	CR
ADMF001.ACCTIX1C	Y	U	-1	-1	0.0
TABLE	CARDF	NPAGESF	TABNO	QUALROWS	
ADMF001.CUSTOMERS	-1	-1	1	40.0	
INDEX	CLU	UR	NLEAF	NLEVEL	CR
ADMF001.CUSTIX1C	Y	U	-1	-1	0.0
ADMF001.CUSTIX2	N	D	-1	-1	0.0

Tables – Same # rows, pages

Indexes – Same # leaf, levels & clusterratio (also 1st & fullkeycard)

Predicate	Actual FF	Estimate
C.CUSTNO BETWEEN 1 AND 100000	0.5182	0.1
STATUS = 'Y'	0.9487	0.04
GENDER = 'F'	0.1237	0.04
A.CUSTNO BETWEEN 1 AND 100000	0.5182	0.1
OPENDATE > '19600101'	1.0000	0.3333
CLOSEDATE < '20070101'	0.6940	0.3333

Equal predicates – same FF

Range/Between predicates use default interpolation

Filter Factor Conclusions

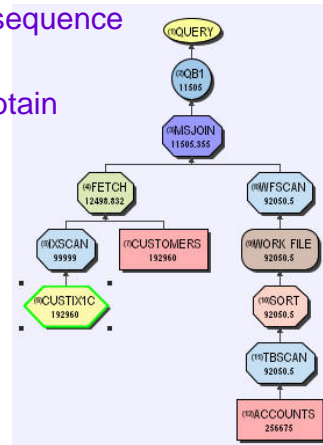
- Skew/Correlation important for costing each object
 - Without RUNSTATS
 - All predicates, tables, indexes, use defaults
 - Optimizer cannot distinguish where the filtering is
 - With host variables/parameter markers
 - Optimizer cannot take advantage of frequencies
 - Exception is NULL frequencies
 - Range predicates use optimistic defaults
 - Correlation is still utilized

And if I can't match FF estimate to reality?

- If you can't provide the perfect statistics
 - Must make the access path obvious another way
- For original query
 - No index to support the preferred join sequence
 - Account table as inner (CUSTNO)
 - Requires literals and special stats to obtain preferred join sequence

```

SELECT *
FROM CUSTOMERS C, ACCOUNTS A
WHERE C.CUSTNO = A.CUSTNO
      AND C.CUSTNO BETWEEN ? AND ?
      AND C.STATUS = ?
      AND C.GENDER = ?
      AND A.OPENDATE > ?
      AND A.CLOSEDATE < ?
    
```

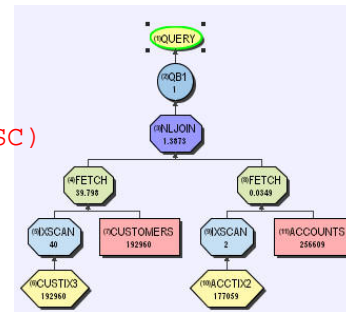


Indexing to overcome statistics limitations

- After creating indexes
 - On CUSTOMERS to support local filtering (as outer)
 - On ACCOUNTS to support the join predicate (as inner)
- Preferred join sequence now chosen with host variables

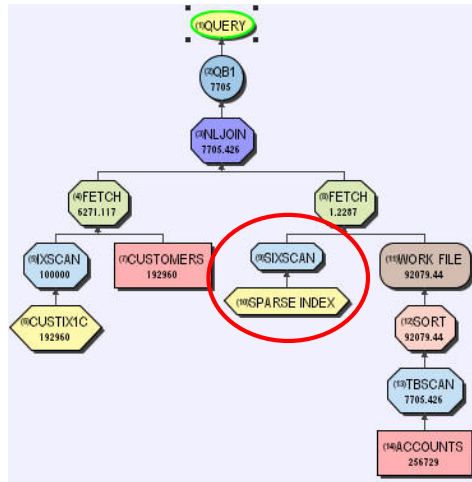
```
CREATE INDEX ADMF001.CUSTIX3
      ON ADMF001.CUSTOMERS
      (STATUS ASC, GENDER ASC, CUSTNO ASC)

CREATE INDEX ADMF001.ACCTIX2
      ON ADMF001.ACCOUNTS
      (CUSTNO ASC)
```



And if I don't create the indexes?

- DB2 9 can create a “sparse index” at runtime
 - To support join
- Previously only available for Star Join



What about skew over a range?

- For high cardinality columns with range skew
 - Too many FREQVAL statistics to collect
- Optimizer assumes data is evenly distributed between LOW2KEY & HIGH2KEY
 - Estimate does not match reality

```
SELECT COUNT(*) = 20131 rows
FROM CUSTOMERS
WHERE BIRTHDATE BETWEEN '1971-01-01' AND '1971-12-31'
```

Count	Cardf	Actual FF	Estimate
20131	192960	0.1043	0.0311

New Statistics in DB2 9

- Histogram statistics added in DB2 9
 - Also known as quantiles
 - Similar to frequencies
 - But each “frequency” specifies a range

```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
TABLE(ADMF001.CUSTOMERS) INDEX(ALL) KEYCARD
COLGROUP(BIRTHDATE) HISTOGRAM
```

- After histograms.....

Actual FF	Estimate
0.1043	0.1064

Stage 1 Predicates	Filter Factor
ADMF001.CUSTOMERS.BIRTHDATE BETWEEN '1971-01-01' AND...	0.1064

Improving FF estimates in DB2 V8

- What have we done in V8?
 - V8 Visual Explain provides optimizer estimates
 - Statistics Advisor (SA) identifies statistics conflicts and potentially missing statistics
 - COLGROUP keyword added to RUNSTATS
 - Simplifies collection of skew/correlation statistics
 - REOPT(ONCE) for dynamic SQL
 - Reopt at 1st execution using 1st set of literal values

Improving FF estimates in DB2 9

- What have we done in 9?
 - **OSC further extends VE functionality**
 - **Workload based OSC/SA**
 - Rather than single query V8 VE/SA
 - **More details**
 - Including transformed SQL text
 - Query annotation with statistics mapped to SQL
 - **Histograms added to RUNSTATS**
 - Provides distribution for higher cardinality columns
 - **REOPT(AUTO)**
 - Only reoptimizes when values change by x%

OSC Query Formatter/Annotation

Query text

There are several options to tune the selected query. Format or categorize selected query text, analyze the query, or use

Query Advisors Tools EXPLAIN timestamp:2007-02-20 16:14:43.429

EXPLAIN options: Run EXPLAIN again Use subsystem EXPLAIN information Use local EXPLAIN information

```
SELECT COUNT(*) FROM CUSTOMERS C, ACCOUNTS A WHERE C.CUSTNO = A.CUSTNO AND C.CUSTNO BETWEEN 1
AND 100000 AND C.STATUS = 'Y' AND C.GENDER = 'F' AND A.OPENDATE > '19600101' AND A.CLOSEDATE < '20070101'
```

- DB2 9 OSC will take an unformatted query
 - And format this, and add costing detail
 - Table cardf, Column cardinalities, FFs etc

Formatted Query	Annotation
SELECT COUNT(*)	
FROM ADMF001.CUSTOMERS AS C	CARDF=192,960 QUALIFIED_ROWS=6,271.117 NPAGESF=3,860
, ADMF001.ACCOUNTS AS A	CARDF=256,729 QUALIFIED_ROWS=92,079.44 NPAGESF=2,469
WHERE (C.CUSTNO = A.CUSTNO	COLCARDF=192,960/182,272 MAX_FREQ=/ FF=5.182420864002779E-6
AND A.CUSTNO BETWEEN 1 AND 100000	COLCARDF=182,272 MAX_FREQ= FF=0.5182397961616516 LOW2KEY=2
AND A.CLOSEDATE < '20070101'	COLCARDF=64 MAX_FREQ=21.35364528354802% FF=0.6939460039138794 LOW2KEY=2
AND A.OPENDATE > '19600101'	COLCARDF=372 MAX_FREQ= FF=0.9973117709159851 LOW2KEY=19766002
AND C.GENDER = 'F'	COLCARDF=3 MAX_FREQ=70.02746683250413% FF=0.12540936470031738
AND C.STATUS = 'Y'	COLCARDF=2 MAX_FREQ= FF=0.4999999403953552 LOW2KEY=N HI
AND C.CUSTNO BETWEEN 1 AND 100000	COLCARDF=192,960 MAX_FREQ= FF=0.5182397961616516 LOW2KEY=2
)	

Agenda

- Introduction
- Process for validating the preferred access path
- Filter Factor Challenges
- Predicate Challenges
- Conclusion

Can my predicates be applied early?

- Index Matching Predicates
 - Only indexable predicates will limit the index entries read
 - As matching or page range
- Stage 1 Predicates
 - Simple stage 1 and non-matching indexable predicates can be applied as index screening
 - To limit data rows accessed
- Stage 2 Predicates
 - Cannot be applied to the index before data access
 - But will be applied to index at stage 2 if index-only
- The stage of predicate application can affect the optimizer's choice of index or table join sequence/method.

Index matching/screening

- Index on
CITY,
CUSTNO,
STATE
- 1 matching
- 1 screening

```
SELECT *
FROM CUSTOMERS
WHERE CITY = ?
      AND STATE = ?
```

Name	Value
Input RIDs	192960
Index Leaf Pages	1678
Matching Predicates	Filter Factor
CUSTOMERS.CITY=(EXPR)	0.0385
Scanned Leaf Pages	65
Screening Predicates	Filter Factor
CUSTOMERS.STATE=(EXPR)	0.1111

Stage 1 predicates

- This WHERE clause predicate is stage 1

Name	Value
Input Cardinality	192960
Scanned Rows	192960
Stage 1 Predicates	Filter Factor
CUSTOMERS.BIRTHDATE BETWEEN '1971-01-01'...	0.0331

```

SELECT *
FROM CUSTOMERS
WHERE BIRTHDATE BETWEEN '1971-01-01' AND '1971-12-31'
    
```


Stage 2 predicates

- This WHERE clause predicate is stage 2

```
SELECT *
FROM CUSTOMERS
WHERE YEAR(BIRTHDATE) = 1971
```

Name	Value
Input Cardinality	192960
Scanned Rows	192960
Stage 1 Returned Rows	192960
Stage 2 Predicates	Filter Factor
YEAR(CUSTOMERS.BIRTHDATE)=1971	0.04

Note: This column expression uses a default FF (1/25)

Predicate Report – Stage 1 or 2

- From the V8 VE predicate report
 - “Is Sargable” value of “N” means stage 2

Is Sargable	Is Join Predicate	Is After Join Predicate	Has parameter marker	Predicate Text
N	N		N	YEAR (CUSTOMERS.BIRTHDATE) =1971

- From DB2 9 OSC predicate report
 - Y/N for “S1”

S1	--Whether the predicate is a stage 1 predicate
----	--

Improving Predicate Application

- Consider manual rewrite to promote predicate application
 - Stage 1 is more efficient than stage 2
 - Indexable is more efficient than stage 1

Stage 2	Stage 1	Indexable
*DEC_COL = :inhost		DEC_COL = DECIMAL(:inhost)
QTY * 2 = :dechost		QTY = :dechost / 2
YEAR(DCOL) = 2007		DCOL BETWEEN '2007-01-01' AND '2007-12-31'
:host BETWEEN C1 AND C2		:host >= C1 AND :host <= C2
DCOL + 10 YEARS < CURRENT DATE		DCOL < CURRENT DATE - 10 YEARS
LOCATE('P',LASTNAME) > 0	LASTNAME LIKE '%P%'	
	DCOL <> '9999-12-31'	DCOL < '9999-12-31'
	GENDER <> 'F'	GENDER = 'M'

* Most datatype/length mismatches resolved in V8

51

GoFurther

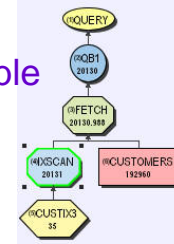
What if I can't rewrite my predicate?

- DB2 9 supports “index on expression”
 - Can turn a stage 2 predicate into indexable

```
SELECT *
FROM CUSTOMERS
WHERE YEAR(BIRTHDATE) = 1971
```

```
CREATE INDEX ADMF001.CUSTIX3
ON ADMF001.CUSTOMERS
(YEAR(BIRTHDATE) ASC)
```

Previous FF = 1/25
 Now, RUNSTATS
 collects frequencies.
 FF is more accurate



Name	Value
Input RIDs	192960
Index Leaf Pages	241
Matching Predicates	Filter Factor
ADMF001.CUSTOMERS.= CAST(1971 AS INTEGER)	0.1043
Scanned Leaf Pages	26
Output RIDs	20131
Total Filter Factor	0.1043
Matching Columns	1

Agenda

- Introduction
- Process for validating the preferred access path
- Filter Factor challenges
- Predicate Challenges
- Conclusion

What have we covered?

- Method for comparing optimizer estimates with reality
- Importance for optimizer to be able to distinguish a “good access path”
 - **And challenges in estimating FF when**
 - Host variables/parameter markers are used
 - RUNSTATS are not run
 - Point or range skew is unknown
 - **If you can't distinguish statistically**
 - Indexing should provide guidance to a good access path

What (else) have we covered?

- Features available to assist with query tuning
 - DB2 V8 Visual Explain
 - DB2 9 Optimization Service Center
- Enhancements to FF estimation
 - DB2 V8
 - COLGROUP keyword for RUNSTATS
 - REOPT(ONCE) for dynamic SQL
 - DB2 9
 - Histogram statistics in RUNSTATS
 - REOPT(AUTO) for dynamic SQL
 - RUNSTATS collects statistics on “Indexed expression”

Session: G06

More Ways to Challenge the DB2 z/OS Optimizer

Terry Purcell

IBM Silicon Valley Lab

tpurcel@us.ibm.com

