DB2 for z/OS V8

Why did the optimizer choose that access path?

Terry Purcell
DB2 z/OS Optimizer Development
IBM Silicon Valley Lab

Agenda

- **Introduction**
- **Data skew**
- **Correlation and the effect of index screening**
- **Range predicate accuracy**
- **Conclusion**

Presentation Goal

- **When the optimizer makes a "poor access path choice"**
  - Demonstrate how DB2 for z/OS V8 makes it easier to:
    - Identify what went wrong
    - Resolve the problem with the "right" solution

Reaching the Goal

- **How to determine why the optimizer may choose a poor access path?**
  - Using some simple query examples……
    - Show how the DB2 z/OS optimizer "estimates" the number of qualified rows per object
    - Compare this with the "actual" qualified rows per object
  - 3 most common reasons for poor access path choice will be demonstrated:
    - Data skew
    - Correlation
    - Range predicates with host variables/parameter markers

Filter Factors

- **Optimizer assigns a "Filter Factor" (FF) to each predicate or predicate combination**
  - Number between 0 and 1 that provides the estimated filtering percentage
    - FF of 0.25 means 25% of the rows are estimated to qualify
  - Calculated using available statistics
    - Column cardinality (COLCARDF)
    - HIGH2KEY/LOW2KEY
    - Frequency statistics (FREQUENCYF in SYSCOLDIST)

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**

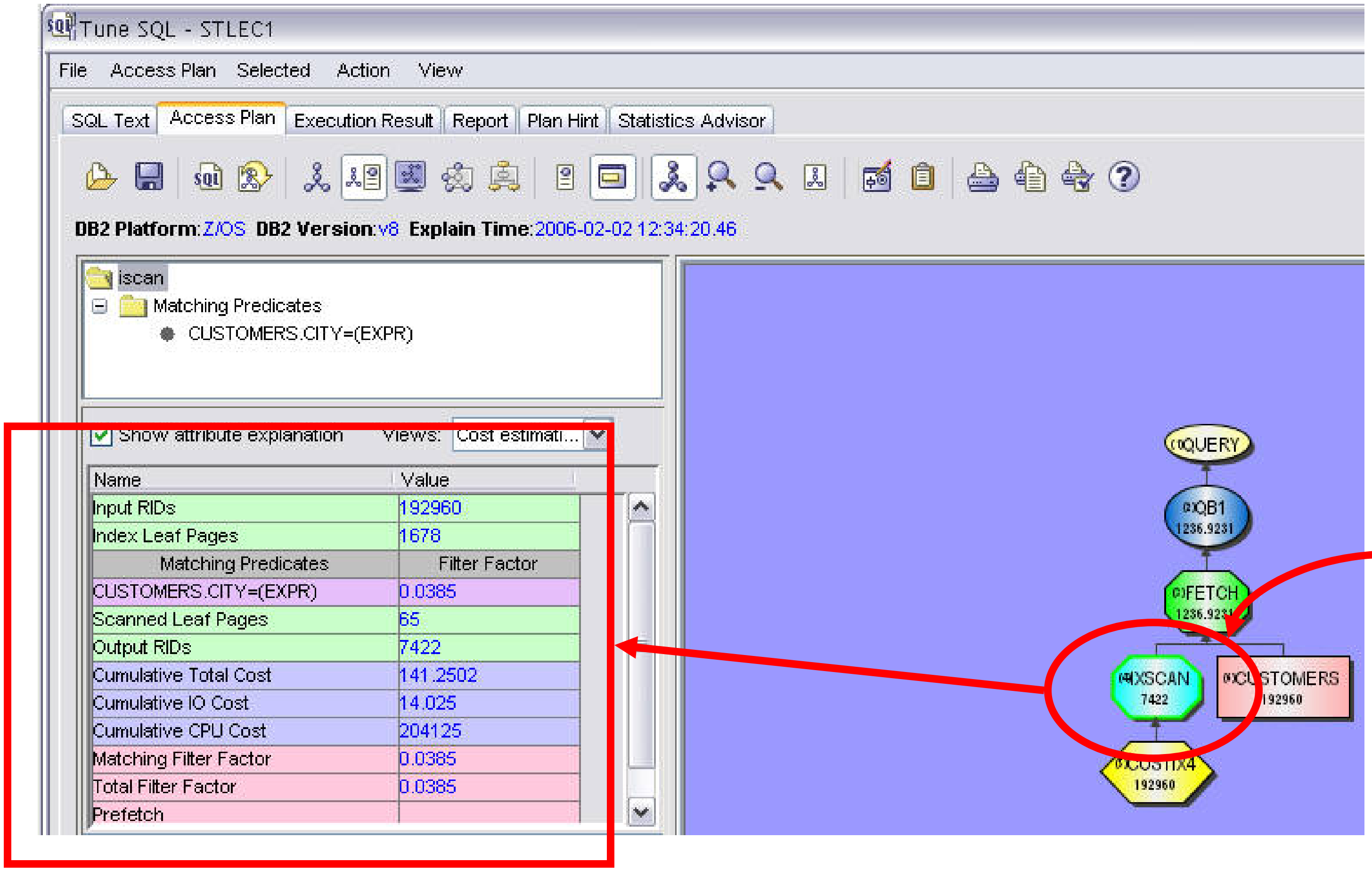
Assigning and combining Filter Factors

- **Accuracy of individual FFs and how to combine them is important for costing**
  - 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 these objects

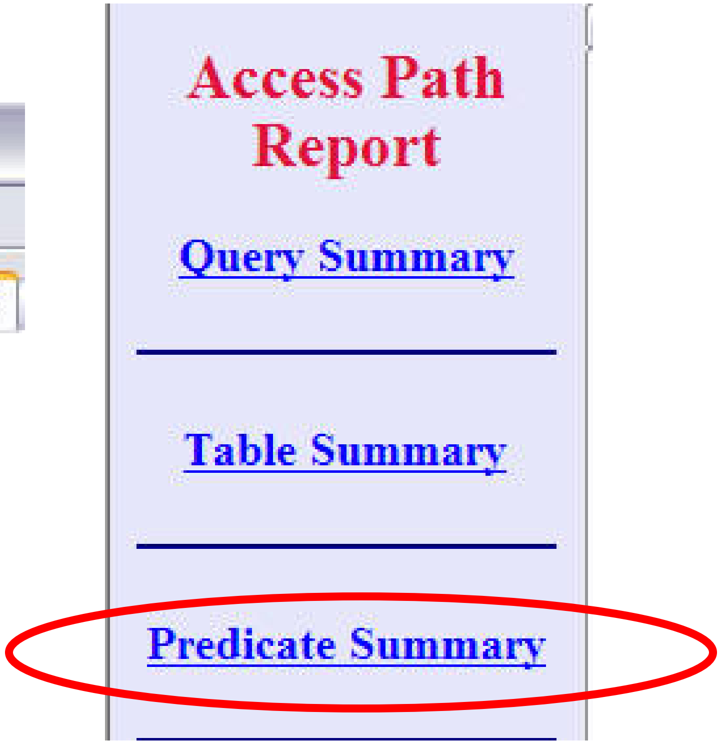
How to obtain Visual Explain detail



Click on a node of the graph to reveal detail

How to access Visual Explain reports

- **From the “Tune SQL” screen**
  – Choose the “Report” tab
  – Click “Generate Report”



  – Then choose either:
    - Query Summary
    - Table Summary
    - Predicate Summary

Agenda

- **Introduction**
- **Data skew**
- **Correlation and the effect of index screening**
- **Range predicate accuracy**
- **Conclusion**

Data Skew

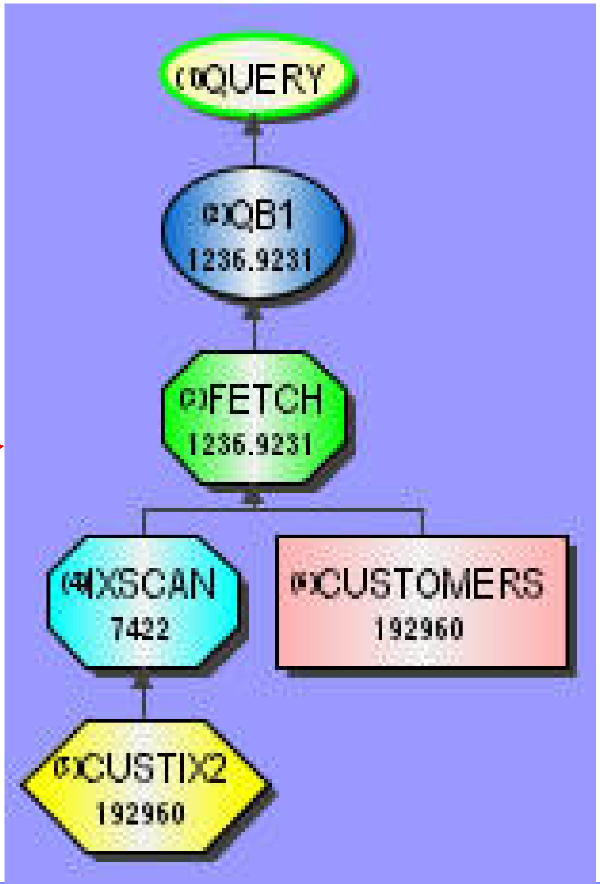
- **Data Skew (or skew)**
  - Describes situation where data is non-uniformly distributed
  - Data can be point-skewed on a value or skewed over a range
  - Eg. STATUS
    - Domain (Y, N)
    - 95% = Y, 5% = N

Query example

- **User is complaining about performance**

```
SELECT *
FROM CUSTOMERS
WHERE COUNTRY = ?
  AND CITY = ?
  AND GENDER = ?
  AND STATUS = ?
```

Access path
via index
CUSTIX2



Validate the user complaint

- **Start by determining where the problem is.**
  - Need actual data values to run a count of the query
    - Hint: Use data values that perform poorly

```
SELECT COUNT(*)      = 811 rows
FROM CUSTOMERS
WHERE COUNTRY = 'USA'
AND CITY = 'NEW YORK'
AND GENDER = 'F'
AND STATUS = 'N'
```

Identify available choices

- **Determine the available access path choices**
  - For multi-table joins
    - Many join sequences may be candidates
  - Within each table (or for single table queries)
    - Many indexes may be candidates
- **Break apart the query per object**
- **Our example is single table**
  - So what are the available indexes?

```
SELECT *
FROM  CUSTOMERS
WHERE COUNTRY = ?
  AND CITY = ?
  AND GENDER = ?
  AND STATUS = ?
```

Query breakdown per Object

- **Single table query**
  - Match query to available indexes

```
SELECT *
FROM CUSTOMERS
WHERE COUNTRY = ?
  AND CITY = ?
  AND GENDER = ?
  AND STATUS = ?
```

```
INDEX CUSTIX2
(COUNTRY ASC
,CITY ASC
,ACCTNO ASC)
```

```
INDEX CUSTIX3
(COUNTRY ASC
,STATUS ASC
,GENDER ASC)
```

Counts of qualified rows per object

- **Per object (index)**
  - Count of rows qualified by CUSTIX2

```
SELECT COUNT(*) = 147,456
FROM CUSTOMERS
WHERE COUNTRY = 'USA'
  AND CITY = 'NEW YORK'
```

Current Index choice

  - Count of rows qualified by CUSTIX3

```
SELECT COUNT(*) = 1,121
FROM CUSTOMERS
WHERE COUNTRY = 'USA'
  AND GENDER = 'F'
  AND STATUS = 'N'
```

Better choice

Counts of qualified rows per predicate

- **Further query breakdown per predicate**
  - Breakdown to the individual components that make up the “estimates per object”

```
SELECT COUNT(*) = 192,960
FROM CUSTOMERS
WHERE COUNTRY = 'USA'
```

```
SELECT COUNT(*) = 147,456
FROM CUSTOMERS
WHERE CITY = 'NEW YORK'
```

```
SELECT COUNT(*) = 24,393
FROM CUSTOMERS
WHERE GENDER = 'F'
```

```
SELECT COUNT(*) = 9,642
FROM CUSTOMERS
WHERE STATUS = 'N'
```

Optimizer Predicate Estimates

- **How did the optimizer determine these FFs?**

Predicate Summary

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor	
2	COUNTRY	1	EQUAL	VALUE		1	1/1 = 1
3	CITY	26	EQUAL	VALUE		0.0385	1/26 = 0.03846
4	GENDER	3	EQUAL	VALUE		0.3333	1/3 = 0.3333
5	STATUS	2	EQUAL	VALUE		0.5	1/2 = 0.5

FF estimates assume even distribution

Optimizer index estimate

- **How did optimizer use the FF estimates to calculate how many rows qualified from the index?**

Name	Value
Input RIDs	192960
Index Leaf Pages	1723
Matching Predicates	Filter Factor
CUSTOMERS.COUNTRY=(EXPR)	1
CUSTOMERS.CITY=(EXPR)	0.0385
Scanned Leaf Pages	67
Output RIDs	7422

1723 * 1/1 * 1/26 = 66.269 leaf pages

192960 * 1/1 * 1/26 = 7421.54 output RIDs

Outcome is dependent on the FFs.
If FFs are wrong, estimate is wrong.

Comparing estimates with reality

- **Calculate actual FF and compare with estimate**
  - Since optimizer estimates assume "even distribution"
    - Data must NOT be evenly distributed

Predicate	Count	Table cardf	Actual FF (count/cardf)	Optimizer FF	
COUNTRY = 'USA'	192,960	192,960	1	1	✓
CITY = 'NEW YORK'	147,456	192,960	0.764	0.0385	✗
GENDER = 'F'	24,393	192,960	0.125	0.3333	✗
STATUS = 'N'	9,642	192,960	0.05	0.5	✗

How do detect Uneven Distribution

- **Run the following count for the column**
  - Results provide proof that CITY is skewed

```
SELECT CITY, COUNT(*)
FROM CUSTOMERS
GROUP BY CITY
ORDER BY 2 DESC;
```

```
   +---------------------------+
   |        CITY   |           |
   +---------------------------+
1_| NEW YORK       |   147456 | 76.4%
2_| CHICAGO        |    21504 | 11.1%
3_| ATLANTA        |     7968 |  4.1%
4_| JACKSONVILLE   |     7872 |  4.1%
5_| LAS VEGAS      |     6144 |  3.2%
6_| PARIS          |       96 |  0.05%
```

Pct calculation
= count/cardf * 100
= 147456/192960 * 100 = 76.4

……only top 6 of 26 shown

Data Skew on other columns

- **Proof that STATUS and GENDER are also skewed**
  - If skew is not provided to optimizer, how can it know?

```
SELECT STATUS, COUNT(*)
FROM CUSTOMERS
GROUP BY STATUS
ORDER BY 2 DESC;
```

```
   +------------------+
   | STATUS |         |
   +------------------+
 1_| Y      |  183318 | 95%
 2_| N      |    9642 |  5%
   +------------------+
```

```
SELECT GENDER, COUNT(*)
FROM CUSTOMERS
GROUP BY GENDER
ORDER BY 2 DESC;
```

```
   +------------------+
   | GENDER |         |
   +------------------+
 1_| M      |  134845 | 70%
 2_| ?      |   33722 | 17.5%
 3_| F      |   24393 | 12.5%
   +------------------+
```

How to collect data skew statistics

- **DB2 V8 adds COLGROUP keyword**
  - For collection of non-uniform distribution statistics on non-indexed (or non-leading index) columns
  - V7/8 collects top 10 frequencies on leading index columns (by default)

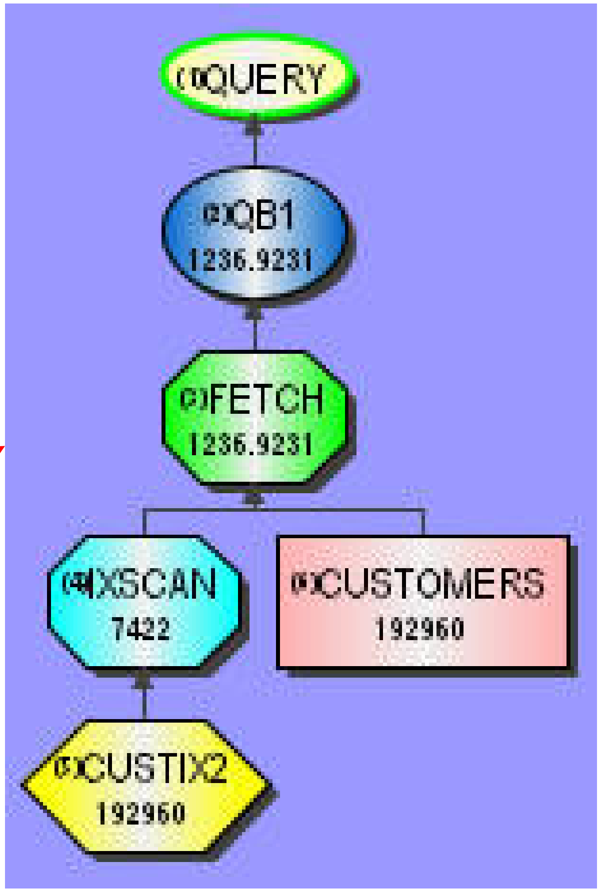
```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
     TABLE(SYSADM.CUSTOMERS)
     COLGROUP(STATUS) FREQVAL COUNT 10
     COLGROUP(CITY)   FREQVAL COUNT 10
     COLGROUP(GENDER) FREQVAL COUNT 10
```

Original query with new statistics

- **New statistics collected**
- **But….access path hasn't changed!**

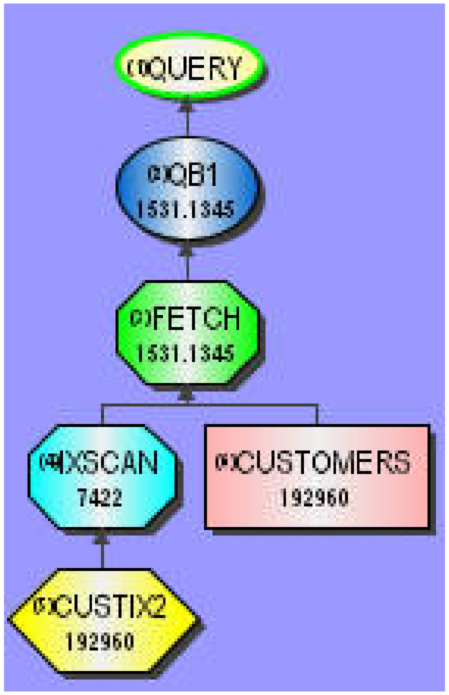
```
SELECT *
FROM CUSTOMERS
WHERE COUNTRY = ?
  AND CITY = ?
  AND GENDER = ?
  AND STATUS = ?
```

Access path via index CUSTIX2



Host variables and frequencies

- **Host variables cannot exploit frequencies**
  - Except GENDER = :HV cannot be NULL
  - Only this estimate has changed
- **Access path is original CUSTIX2**



Predicate Summary

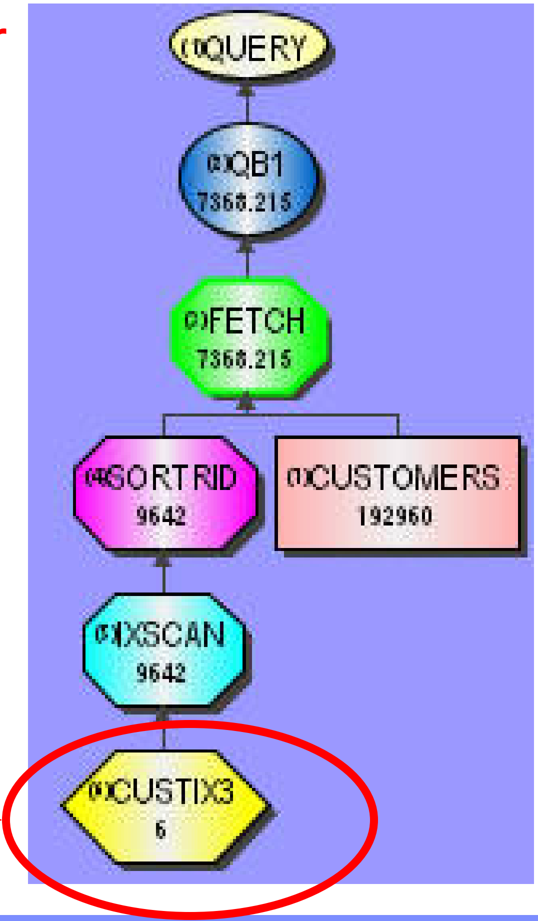
Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
2	COUNTRY	1	EQUAL	VALUE		1
3	CITY	26	EQUAL	VALUE		0.0385
4	GENDER	3	EQUAL	VALUE		0.4126
5	STATUS	2	EQUAL	VALUE		0.5

33722 NULLs
Thus FF = ½ of
(192960 – 33722)

Access path with literal values

- **After new statistics are collected**
- AND literals known to the optimizer either with REOPT(ALWAYS) or hard-coded
  – Optimizer now chooses the preferred index

```
INDEX CUSTIX3
(COUNTRY ASC
,STATUS ASC
,GENDER ASC)
```



Predicate report for literals

Predicate Summary

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
2	COUNTRY	1	EQUAL	VALUE		1
3	CITY	26	EQUAL	VALUE		0.7642
4	GENDER	3	EQUAL	VALUE		0.1264
5	STATUS	2	EQUAL	VALUE		0.05

Actual FF
1
0.764
0.125
0.05

- **Resultant FF estimates match reality**
  - Don't always expect perfection.
  - Objective is to have estimates close to reality.

And other values?

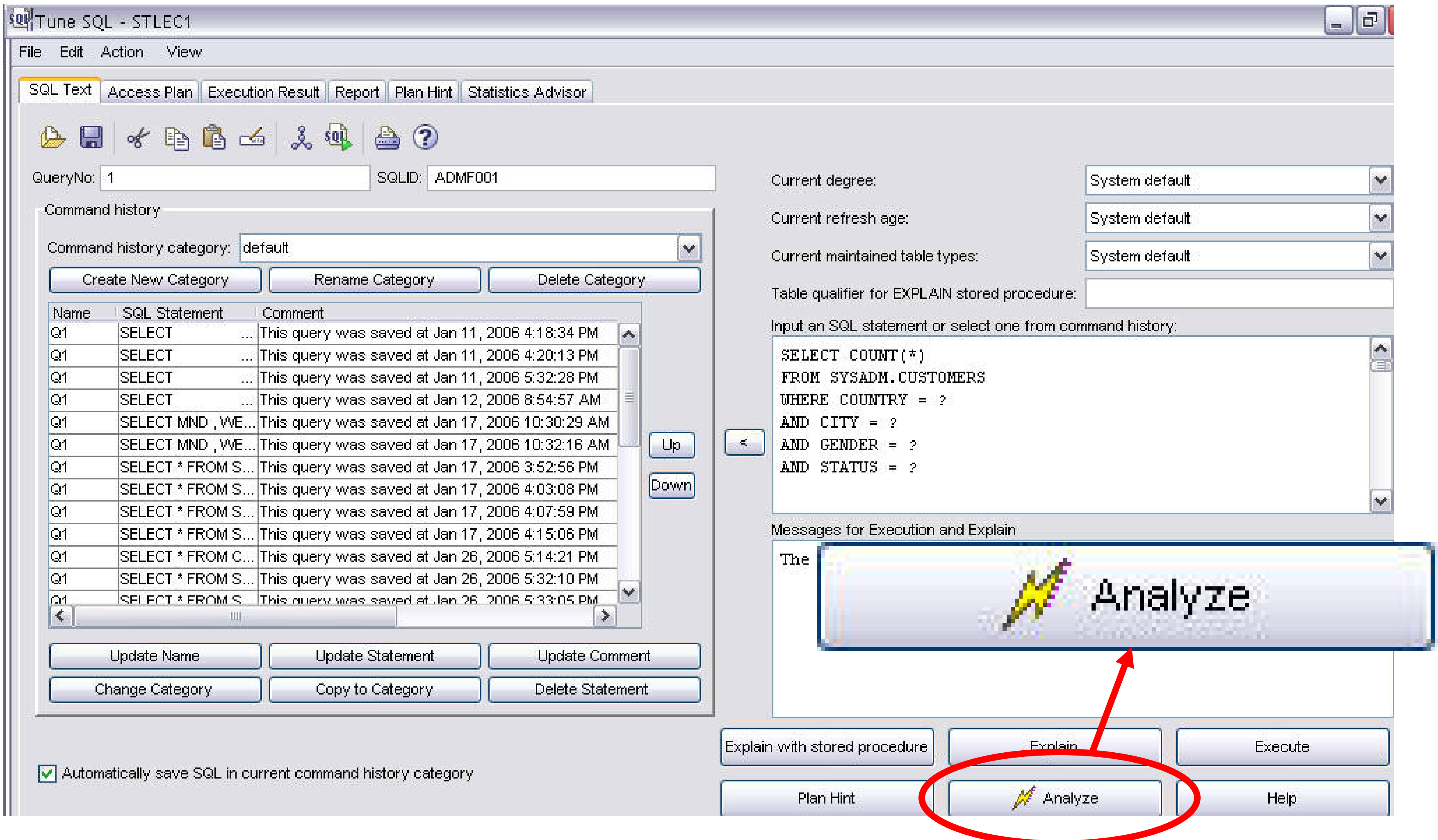
- **Index on CITY was a poor choice for “NEW YORK”**
  - But a good choice for values with lower frequencies
  - STATUS/GENDER index becomes a bad choice for higher frequency values of these columns
- **Different data values may call for a different index choice**

	CITY	
1_	NEW YORK	147456
2_	CHICAGO	21504
3_	ATLANTA	7968
4_	JACKSONVILLE	7872
5_	LAS VEGAS	6144
6_	PARIS	96

	STATUS	
1_	Y	183318
2_	N	9642

	GENDER	
1_	M	134845
2_	?	33722
3_	F	24393

Tuning shortcut - Visual Explain



Statistics Advisor Recommendations

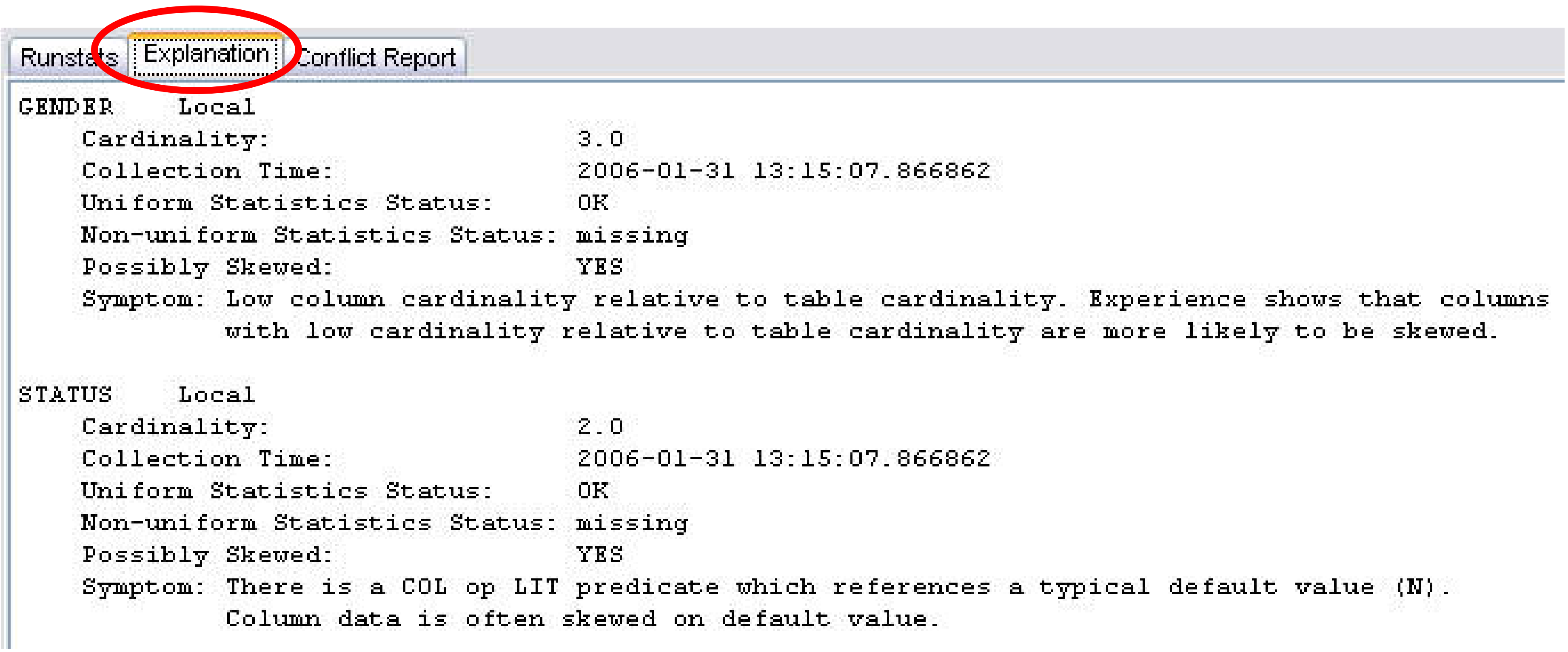
- **When literal values have been provided**
  - SA recommends the following RUNSTATS with FREQVAL option

Runstats | Explanation | Conflict Report

```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
         TABLE(SYSADM.CUSTOMERS)
         COLGROUP(STATUS) FREQVAL COUNT 1
         COLGROUP(CITY) FREQVAL COUNT 10
         COLGROUP(GENDER) FREQVAL COUNT 10
         SORTDEVT SYSDA
SHRLEVEL CHANGE REPORT YES
```

SA Explanations

- **Explanation tab provides reasons for recommendations**




```
GENDER     Local
    Cardinality:                   3.0
    Collection Time:               2006-01-31 13:15:07.866862
    Uniform Statistics Status:     OK
    Non-uniform Statistics Status: missing
    Possibly Skewed:               YES
    Symptom: Low column cardinality relative to table cardinality. Experience shows that columns
             with low cardinality relative to table cardinality are more likely to be skewed.

STATUS     Local
    Cardinality:                   2.0
    Collection Time:               2006-01-31 13:15:07.866862
    Uniform Statistics Status:     OK
    Non-uniform Statistics Status: missing
    Possibly Skewed:               YES
    Symptom: There is a COL op LIT predicate which references a typical default value (N).
             Column data is often skewed on default value.
```

Statistics Advisor recommendations

- **When host variables are specified**
  - REOPT is suggested

Runstats | Explanation | Conflict Report

```
Predicate Analysis Report:

==============================================================================================

The following predicates may contain host variables, parameter markers, or special registers =>

SYSADM.CUSTOMERS.COUNTRY = (MARKER) (COLCARD: 1.0, FF: 1.0)
SYSADM.CUSTOMERS.CITY = (MARKER) (COLCARD: 26.0, FF: 0.03846153989434242)
SYSADM.CUSTOMERS.GENDER = (MARKER) (COLCARD: 3.0, FF: 0.3333333134651184)
SYSADM.CUSTOMERS.STATUS = (MARKER) (COLCARD: 2.0, FF: 0.5)

Recommended action: use REOPT(VARS) or REOPT(ONCE) as bind option
```

Recommended action: use REOPT(VARS) or REOPT(ONCE) as bind option

Data Skew Conclusions

- **When column values are unevenly distributed (skewed)**
  - The optimizer is not aware unless:
    - Frequency statistics are collected
    - The literal values are provided or REOPT(ALWAYS or ONCE) is used
      - Exception: Knowledge of NULL frequencies will be utilized for host variables/parameter markers if predicate cannot be NULL

Possible Recommendations

1. **REOPT(ALWAYS) (a.k.a REOPT(VARS))**
2. **A single index that supports all combinations**
3. **If search is always for the same STATUS value**
   1. Hardcode the literal in the SQL.
   2. Ensure frequency statistics are collected for that value.
4. **Don’t index columns that are skewed and search is by high frequency value.**
   1. Unless literals are known to optimizer
5. **Or, separate SQLs for the skewed cases**

Agenda

- **Introduction**
- **Data skew**
- **Correlation and the effect of index screening**
- **Range predicate accuracy**
- **Conclusion**

Correlation

- **Correlation**
  - When two or more columns are NOT independent
    - Eg. CITY, STATE
      - Every city does NOT exist in every state.
    - Eg. Automobile MANUFACTURER, MODEL
      - Only TOYOTA makes a CAMRY
  - Determines degree that predicate FFs are multiplied
    - Applicable for literals, host vars and parameter markers

Query & Candidate Indexes

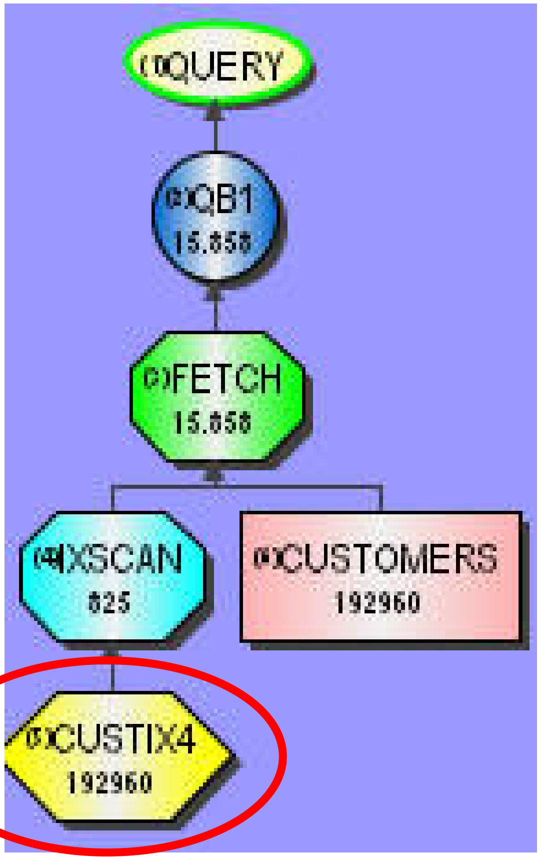
- **Customer complaint about performance**

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

Optimizer chose CUSTIX4

```
INDEX CUSTIX4
(CITY ASC
,ACCTNO ASC
,STATE ASC)
```

```
INDEX CUSTIX5
(ZIPCODE ASC
,ACCTNO ASC)
```



Validate the user complaint

- **Start by determining where the problem is.**
  - Need actual data values to run a count of the query
    - Using data values that perform poorly

```
SELECT COUNT(*)   = 3072
FROM CUSTOMERS
WHERE ZIPCODE = ‘60607’
  AND CITY    = ‘CHICAGO’
  AND STATE   = ‘IL’
```

Query breakdown per object

- **Single table query**
  - Match query to available indexes

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

```
INDEX CUSTIX5
(ZIPCODE ASC
,ACCTNO ASC)
```

```
INDEX CUSTIX4
(CITY ASC
,ACCTNO ASC
,STATE ASC)
```

Counts of qualified rows per index

- **Per index**
  - Count of rows qualified by CUSTIX4

```
SELECT COUNT(*) = 21,504
FROM CUSTOMERS
WHERE CITY     = 'CHICAGO'
  AND STATE    = 'IL'
```

Current Index choice

  - Count of rows qualified by CUSTIX5

```
SELECT COUNT(*) = 3,072
FROM CUSTOMERS
WHERE ZIPCODE = '60607'
```

Better choice

Counts of qualified rows per predicate

- **Breakup the query further**

```
SELECT COUNT(*) = 21,504
FROM CUSTOMERS
WHERE CITY = 'CHICAGO'
```

Equals CUSTIX4 total index count, thus STATE provides no further filtering

```
SELECT COUNT(*) = 21,792
FROM CUSTOMERS
WHERE STATE = 'IL'
```

```
SELECT COUNT(*) = 3,072
FROM CUSTOMERS
WHERE ZIPCODE = '60607'
```

Equals total table filtering, thus CITY/STATE provide no further filtering

Optimizer index estimate

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
Output RIDs	825
Cumulative Total Cost	141.2502
Cumulative IO Cost	14.025
Cumulative CPU Cost	204125
Matching Filter Factor	0.0385
Total Filter Factor	0.0043
Prefetch	
Matching Columns	1

Didn't we find STATE did not filter after CITY?

Why is total index FF 1/26 * 1/9 = 0.0043?

Optimizer considers predicates independent, unless statistics indicate otherwise

Detecting Correlation - Counts

- **Run Predicate counts**
  - Distinct occurrences of each column

```
SELECT COUNT(DISTINCT CITY)  =  26 CITIES
      ,COUNT(DISTINCT STATE) =   9 STATES
FROM CUSTOMERS
```

  - Distinct occurrences of the column group

```
SELECT COUNT(*)  =  31 Combinations of CITY, STATE
FROM
(SELECT DISTINCT CITY, STATE
 FROM CUSTOMERS) AS A
```

Detecting Correlation - Calculation

- **Calculation to detect correlation**
  - If the product of the individual counts > group count
    - Then columns are correlated
      - Product of counts = 26 * 9 = 234
      - Group count = 31
      - 234 > 31
    - Therefore, columns are correlated
- **Current index qualified row estimate**
  - 192960 * 1/26 * 1/9 = 824.6
  - Current statistics fail to show columns are correlated

Correlation - Theory

100% correlated | 100% independent

26 | 234

Reality | Current estimate

31 | 234

- **On the scale of "correlated" vs "independent"**
  - 26 (CITY colcardf) = columns are correlated
    - Largest COLCARDF of the 2 or more columns
    - Every city belongs in only 1 state
  - 234 (CITY colcardf * STATE colcardf) = columns are independent
    - Product of the 2 or more columns
    - Every city belongs in every state
  - Counts show that there are 31 combinations of city/state
    - Some cities exist in more than 1 state

Available correlation statistics

- **Optimizer uses available “multi-column cardinalities” (MCARDs) from**
  - Index KEYCARD
  - Index FULLKEYCARDF
  - COLGROUPs from other columns

- **Available cardinalities for CITY/STATE**
  - Index CUSTIX4 (CITY, ACCTNO, STATE)
    - Only index FULLKEYCARDF is available for this index
      - Fullkeycardf = 192960
    - Colcardf of ACCTNO = 192960 (unique)

What about KEYCARD?

- By default, RUNSTATS collects Firstkeycardf & Fullkeycardf
- KEYCARD collects all intermediate MCARDS:
  - **2ndkeycardf, 3rdkeycardf etc.**

```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
         INDEX(CUSTIX4) KEYCARD
```

```
CUSTIX4
(CITY        Firstkeycardf (CITY)
,ACCTNO      MCARD (CITY, ACCTNO) - from KEYCARD
,STATE)      Fullkeycardf (CITY, ACCTNO, STATE)
```

- **KEYCARD is irrelevant here because it does not contain both CITY & STATE**

Effect of screening predicates

- **Index CUSTIX4 (CITY, ACCTNO, STATE)**
  - Fullkeycardf = 192960 ← Contains CITY/STATE & ACCTNO
  - Colcardf of ACCTNO (unique) = 192960

- **To determine the filtering for index CUSTIX4**
  - Optimizer will multiply individual FFs for CITY/STATE
  - And compare with available MCARD (fullkeycardf = 192960)
    - 26 * 9 = 234 < 192960
    - Thus, columns are considered independent
      - CITY/STATE MCARD was destroyed by high colcardf column ACCTNO

Collecting correlation statistics

- **V8 simplifies MCARD collection for non-indexed columns or non-consecutive indexed columns**
  - Using the new COLGROUP option

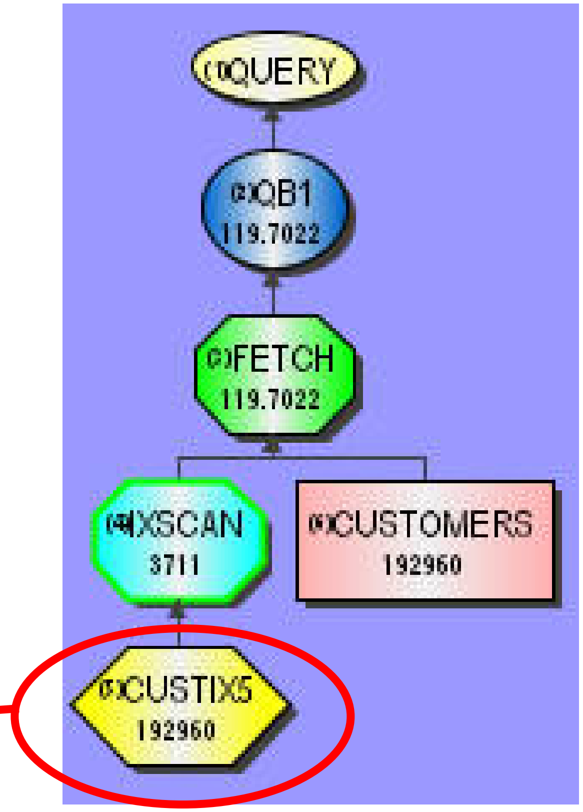
```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
         TABLE(SYSADM.CUSTOMERS)
         COLGROUP(CITY, STATE)
```

New Optimizer Choice

- **CUSTIX5 is now used**
  - ZIPCODE is considered more filtering than CITY/STATE

```
INDEX CUSTIX4
(CITY ASC
,ACCTNO ASC
,STATE ASC)

INDEX CUSTIX5
(ZIPCODE ASC
,ACCTNO ASC)
```



Optimizer index costing result

- **Visual explain index scan detail for new optimizer choice index CUSTIX5 (ZIPCODE)**

Name	Value
Input RIDs	192960
Index Leaf Pages	815
Matching Predicates	Filter Factor
CUSTOMERS.ZIPCODE=(EXPR)	0.0192
Scanned Leaf Pages	16
Output RIDs	3711
Cumulative Total Cost	39.7731
Cumulative IO Cost	3.95
Cumulative CPU Cost	55749.996
Matching Filter Factor	0.0192
Total Filter Factor	0.0192
Prefetch	
Matching Columns	1

No. of qualified rows from index = 3711

Optimizer data fetch costing result

Name	Value
Input Cardinality	3711
Scanned Rows	3711
Stage 1 Predicates	Filter Factor
CUSTOMERS.CITY=(EXPR)	0.0385
CUSTOMERS.STATE=(EXPR)	0.1111
Stage 1 Returned Rows	119.7022
Stage 2 Returned Rows	119.7022
Output Cardinality	119.7022
Cumulative Total Cost	207.5566
Cumulative IO Cost	20.2327
Cumulative CPU Cost	1067319
Stage 1 Columns	10
Page Range	
Prefetch	S

# of qualified rows from ZIPCODE index

Stage 1 predicate Filter Factors

3711 * 1/26 * 1/9 = 16
Thus we have used CITY/STATE MCARD.

But after ZIPCODE, CITY/STATE don't filter. So we need more correlation stats.

Correlation – Just to prove it (again)

- **Run counts for all predicate columns this time**
  - Distinct occurrences of each column

```
SELECT COUNT(DISTINCT CITY)    = 26 CITIES
      ,COUNT(DISTINCT STATE)   =  9 STATES
      ,COUNT(DISTINCT ZIPCODE) = 52 ZIPCODES
FROM CUSTOMERS
```

  - Distinct occurrences of the column group

```
SELECT COUNT(*) = 52 Combinations - CITY,STATE,ZIPCODE
FROM
(SELECT DISTINCT CITY, STATE, ZIPCODE
 FROM CUSTOMERS) AS A
```

Detecting Correlation - Calculation

- **Calculation to detect correlation**
  - If the product of the individual counts > group count
    - Then columns are correlated
      - Product of counts = 26 * 9 * 52 = 12,168
      - Group count = 52
      - 12,168 > 52
    - Therefore, columns are correlated

```
RUNSTATS TABLESPACE TPTEST.TPTSTTS1
         TABLE(SYSADM.CUSTOMERS)
         COLGROUP(CITY, STATE, ZIPCODE)
```

Optimizer data fetch final result

Name	Value
Input Cardinality	3711
Scanned Rows	3711
Stage 1 Predicates	Filter Factor
CUSTOMERS.CITY=(EXPR)	0.0385
CUSTOMERS.STATE=(EXPR)	0.1111
Stage 1 Returned Rows	3710.769
Stage 2 Returned Rows	3710.769
Output Cardinality	3710.769
Cumulative Total Cost	229.024
Cumulative IO Cost	20.2327
Cumulative CPU Cost	5448419
Stage 1 Columns	10
Page Range	
Prefetch	S
Correlated Subquery IO Cost	0

No of qualified rows from index = 3711

Stage 1 predicate Filter Factors

CITY, STATE predicates now have no effect on final count

Correlation Conclusions

- **Optimizer will consider columns to be independent unless statistics demonstrate otherwise**
  - Columns incorrectly assumed to be independent have a dramatic effect on the total filtering estimate
- **Correlation does not require knowledge of literal values**
- **While indexes can provide correlation information,**
  - Indexes should be designed for filtering
  - RUNSTATS should be used for correlation
    - Where possible

Possible Recommendations

- **Use KEYCARD option for all multi-column indexes**
- **Create indexes to support filtering**
- **Use COLGROUP option to collect correlation**
  - For matching + screening cases
  - And for total predicate filtering
  - Use indexing if RUNSTATS options are difficult to implement

Agenda

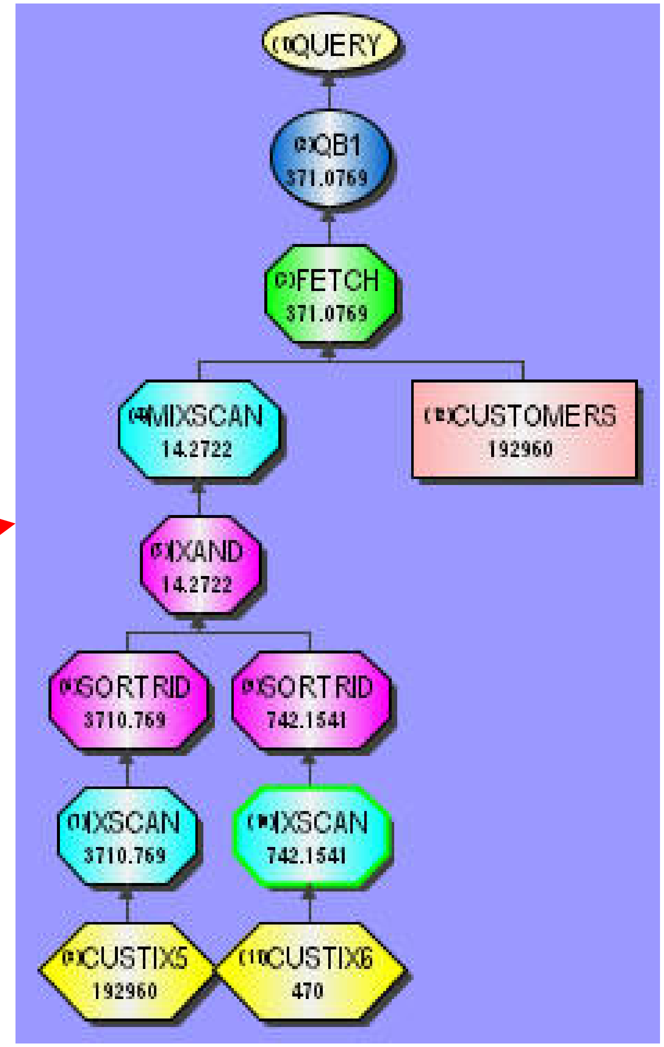
- **Introduction**
- **Data skew**
- **Correlation and the effect of index screening**
- **Range predicate accuracy**
- **Conclusion**

Query example

- **User complaint about performance**

```
SELECT *
FROM CUSTOMERS
WHERE ZIPCODE   = ?
  AND CITY      = ?
  AND BIRTHDATE < ?
```

Multi-index access path via CUSTIX5 & CUSTIX6



Validate the user complaint

- **Start by determining where the problem is.**
  - Need actual data values to run a count of the query

```
SELECT COUNT(*)        = 1536 rows
FROM CUSTOMERS
WHERE ZIPCODE   = 30301
  AND CITY      = 'ATLANTA'
  AND BIRTHDATE < '9999-12-31'
```

Query breakdown

- **Break apart the query to compare reality and estimates per object**
  - Match query to indexes

```
SELECT *
FROM CUSTOMERS
WHERE ZIPCODE   = ?
  AND CITY      = ?
  AND BIRTHDATE < ?
```

```
INDEX CUSTIX5
(ZIPCODE ASC
,ACCTNO  ASC)
```

```
INDEX CUSTIX6
(CITY      ASC
,BIRTHDATE ASC)
```

Counts of qualified rows per index

- **Per index**
  - Count of rows qualified by CUSTIX5

```
SELECT COUNT(*) = 1,536
FROM CUSTOMERS
WHERE ZIPCODE = 30301
```

Equals total table filtering, thus CITY/BIRTHDATE provide no further filtering

  - Count of rows qualified by CUSTIX6

```
SELECT COUNT(*) = 7,968
FROM CUSTOMERS
WHERE CITY      = 'ATLANTA'
  AND BIRTHDATE < '9999-12-31'
```

Counts of qualified rows per predicate

- **Breakup the query further**

```
SELECT COUNT(*) = 7,968
FROM CUSTOMERS
WHERE CITY   = 'ATLANTA'
```

```
SELECT COUNT(*) = 192,960
FROM CUSTOMERS
WHERE BIRTHDATE < '9999-12-31'
```

Equal to number of rows in table. Thus no filtering.

```
SELECT COUNT(*) = 1,536
FROM CUSTOMERS
WHERE ZIPCODE = 30301
```

Predicate Report with Host Vars

- **Obtain optimizer estimates from predicate report…**

Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
CITY	26	EQUAL	VALUE		0.0385
ZIPCODE	52	EQUAL	VALUE		0.0192
BIRTHDATE	456	RANGE	VALUE		0.1

Comparing estimates with reality

- **Calculate actual FF and compare with estimate**
  - We know CITY & ZIPCODE are skewed and correlated
  - But why is the BIRTHDATE estimate incorrect?

Predicate	Count	Table cardf	Actual FF (count/cardf)	Optimizer FF	
CITY = ‘ATLANTA’	**7,968**	**192,960**	0.0413	0.0385	ok
ZIPCODE = 30301	**1,536**	**192,960**	0.008	0.0192	✗
BIRTHDATE < ‘9999-12-31’	**192,960**	**192,960**	1	0.1	✗

Range Predicate Interpolation

Default filter factors for interpolation (from DB2 Admin Guide)

COLCARDF	Factor for Op	Factor for LIKE/BETWEEN
>=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
>=1,000	1/30	1/100
>=100	1/10	3/100
>=2	1/3	1/10
=1	1	1
<=0	1/3	1/10

1/10 = 0.1 filter factor for BIRTHDATE

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

COMMENT: This is DB2's documented guess for an impossible to estimate Filter factor. Used for host variables/parameter markers.

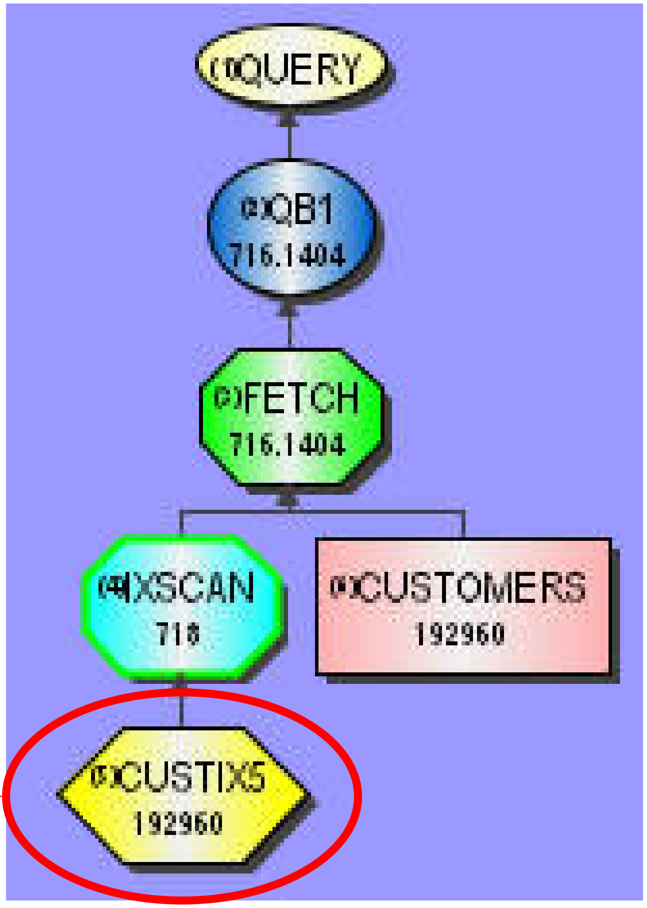
Range Predicate Interpolation

- **Range predicate with host var/parameter marker**
  - Use default interpolation filter factor chart
    - COLCARDF 456 --> FF = 1/10
  - In reality, could qualify anywhere from all to no rows
    - Here's another sample predicate:
      - BIRTH_DATE <= ?
    - How many people here were born before parameter marker?
      - What if value is '1920-01-01'?
      - What if value is '1990-01-01'?
      - Cannot accurately estimate without literal value

Access path with literal values

- Literals known to the optimizer either with REOPT(ALWAYS) or hard-coded
  - Optimizer chooses the preferred index

```
INDEX CUSTIX5
(ZIPCODE ASC
,ACCTNO  ASC)
```



Predicate report for literals

- **With literals or REOPT, how do estimates compare?**
  – CITY FF is correct
  – BIRTHDATE is very close
  – ZIPCODE is near enough

Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor	Actual FF
CITY	26	EQUAL	VALUE		0.0413	0.0413
ZIPCODE	52	EQUAL	VALUE		0.0037	0.008
BIRTHDATE	456	RANGE	VALUE		0.9978	1

Statistics Advisor Recommendations

- **For original query with host variables**
  - SA recommends using REOPT

Runstats | Explanation | Conflict Report

```
Predicate Analysis Report:

==========================================================================================

The following predicates may contain host variables, parameter markers, or special registers =>

SYSADM.CUSTOMERS.CITY = (MARKER) (COLCARD: 26.0, FF: 0.03846153989434242)
SYSADM.CUSTOMERS.ZIPCODE = (MARKER) (COLCARD: 52.0, FF: 0.019230768084526062)
SYSADM.CUSTOMERS.BIRTHDATE <op> (MARKER) (COLCARD: 456.0, FF: 0.10000002384185791)

Recommended action: use REOPT(VARS) or REOPT(ONCE) as bind option
```

use REOPT(VARS) or REOPT(ONCE) as bind option

Conclusions

- **Range predicates with parameter markers/host vars**
  - Optimizer calculates FFs using documented default interpolation formula:
    - Impossible for optimizer to always estimate correctly
    - Also used for special registers
      - WHERE BIRTHDATE < CURRENT DATE
    - Defaults are very optimistic
      - Which is problematic if the predicate provides poor filtering

Possible Recommendations

- **REOPT(ALWAYS)**
  - Or provide the literal value if regularly used
  - REOPT(ONCE) for dynamic SQL
- **Don’t index range predicate columns that are poorly filtering.**
  - If a correct FF estimate will be challenging for optimizer
  - Poorly estimated predicates can encourage an index to be chosen
    - If it must be indexed, then also add it to the preferred index

Other predicate challenges

- **Predicate examples that are difficult to estimate FF:**
  - Column expressions
    - WHERE SUBSTR(STATE,1,1) = 'A'
  - Non-column expressions
    - WHERE BIRTHDATE < DATE('2006-01-01') – 1 YEAR
  - LIKE with leading (or intermediate) wildcard
    - WHERE LASTNAME LIKE '%A%'
  - IN predicates with many duplicates
    - WHERE ACCTNO IN (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)

Agenda

- **Introduction**
- **Data skew**
- **Correlation and the effect of index screening**
- **Range predicate accuracy**
- **Conclusion**

Conclusion

- **We demonstrated the 2 main factors for the optimizer to accurately cost different objects**
  - The individual filter factors, and
  - How those filter factors are combined.
- **These impact estimates for**
  - Index matching
  - Total index filtering
  - Total table level filtering
- **And we highlighted how DB2 V8 simplifies the identification and resolution of poor optimizer estimates**

Why did the optimizer choose that access path?

Thank you for listening!!!

Terry Purcell

IBM Silicon Valley Lab

tpurcel@us.ibm.com