

May 7-11, 2006

Tampa Convention Center

Tampa, Florida, USA

Why did the Optimizer Choose that Path? Part 2

IDUG® 2006

North America

Terry Purcell
IBM Silicon Valley Lab

GoFurther



IDUG® 2006 - North America

Agenda – Part 1 (session G04)

- Introduction
- Data skew
- Correlation and the effect of index screening
- Conclusion – part 1

Agenda – Part 2

- Predicate filtering stages
- Filter factor challenges
- Impact of Clustering
- Conclusion

Index Predicate Application

- Only indexable predicates can be index matching
- Simple stage 1 and non-matching indexable predicates can be applied as index screening
- 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.

3

Indexable, Stage 1, Stage 2?

- How do you determine if a predicate is indexable?
 - By checking MATCHCOLS in the PLAN_TABLE
 - What if you don't have an index to support matching?
- Is the predicate stage 1 or stage 2?
 - PLAN_TABLE doesn't identify these
- What predicates have been generated or rewritten by the optimizer?
 - PLAN_TABLE doesn't identify these

4

Visual Explain predicate details

- From access path graph
 - Highlight the index scan node for indexed predicates
 - Highlight the fetch node for non-indexed predicates

Detail will appear here

Name	Value
Input RIDs	192960
Index Leaf Pages	1670
Matching Predicates	Filter Factor
CUSTOMERS.CITY=(EXPR)	0.0385
Scanned Leaf Pages	65
Screening Predicates	Filter Factor
CUSTOMERS.STATE=(EXPR)	0.1111
Output RIDs	3711
Cumulative Total Cost	135.1853
Cumulative IO Cost	13.42
Cumulative CPU Cost	201100
Matching Filter Factor	0.0385

Index matching/screening

- Index on CITY, ACCTNO, 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 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

7

Predicate Report – Stage 2

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

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
1	NONCOL		EQUAL	VALUE		0.04

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

8

Stage 1 predicates

- Rewritten, the predicate becomes 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'
```

Predicate Report – Stage 1

- From the predicate report
 - “Is Sargable” value of “Y” means stage 1

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
1	BIRTHDATE	456	BETWEEN	VALUE		0.0331

Is Sargable	Is Join Predicate	Is After Join Predicate	Has parameter marker	Predicate Text
Y	N		N	CUSTOMERS.BIRTHDATE BETWEEN '1971-01-01' AND '1971-12-31'

Sargable and/or indexable

- Not all stage 1 (sargable) predicates are indexable
 - As a general rule, NOT predicates cannot be index matching. Exceptions include:
 - IS NOT NULL, which is converted to < high-values
 - NOT <, which is converted to >=
 - Most stage 1 predicates can be index screening

Sargable, but not index matching. Predicate is NOT BETWEEN

Is Sargable	Is Join Predicate	Is After Join Predicate	Has parameter marker	Predicate Text
Y	N		N	CUSTOMERS.BIRTHDATE NOT BETWEEN '1971-01-01' AND '1971-12-31'

11

Optimizer generated predicates

- Predicates generated by the optimizer through “transitive closure” will appear in the predicate summary

- If C.STATE = S.STATE
- And C.STATE BETWEEN ? AND ?
- S.STATE must also be BETWEEN ? AND ?

			Has parameter marker	Predicate Text
Y	Y		N	C.STATE=S.STATE
Y	N		Y	C.STATE BETWEEN (EXPR) AND (EXPR)
Y	N		Y	S.STATE BETWEEN (EXPR) AND (EXPR)

```
SELECT *
FROM SYSADM.CUSTOMERS C
, SYSADM.STATES S
WHERE C.STATE = S.STATE
AND C.STATE BETWEEN ? AND ?
```

Generated predicate

12

Optimizer rewritten predicates

- Predicates appear in their rewritten form (if applicable)
 - Simple OR predicates are rewritten to IN

```
SELECT *
FROM SYSADM.CUSTOMERS
WHERE ACCTNO = ?
      OR ACCTNO = ?
```

Predicate Number	Left-hand Side	Left-hand Side Column Cardinality	Predicate Type	Right-hand Side	Right-hand Side Column Cardinality	Filter Factor
1	ACCTNO	192960	IN	VALUE		1.036484263750026E-5

Is Sargable	Is Join Predicate	Is After Join Predicate	Has parameter marker	Predicate Text
Y	N		Y	CUSTOMERS.ACCTNO IN ((EXPR),(EXPR))

Rewritten predicate

Agenda

- Predicate filtering stages
- Filter factor challenges
- Impact of Clustering
- Conclusion

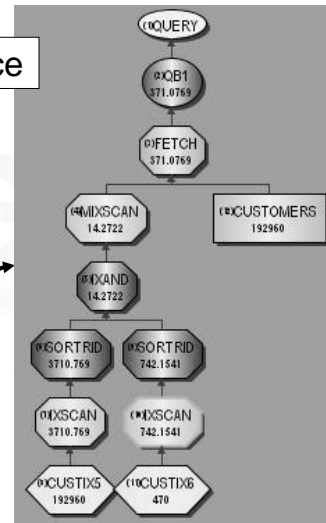
Query example

- User complain about performance

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

Multi-index
access path
via CUSTIX5
& CUSTIX6

15



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'
```

16

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)

17

Counts of qualified rows per index

- Per index
 - Count of rows qualified by CUSTIX5

```

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

- Count of rows qualified by CUSTIX6

```

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

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

18

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
 - 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
ZIPCODE = 30301	1,536	192,960	0.008	0.0192
BIRTHDATE < '9999-12-31'	192,960	192,960	1	0.1

ok
X
X

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

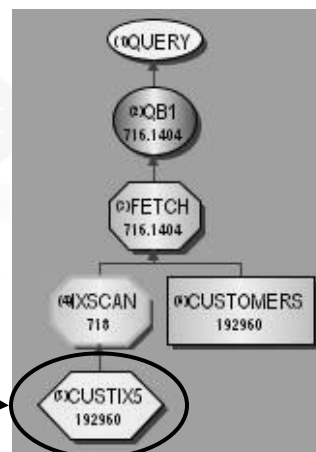
- Range predicate with parameter marker
 - Use default interpolation filter factor chart
 - COLCARDF 456 --> FF = 1/10
 - In reality, could qualify anywhere from all to no rows
 - Here's another sample predicate:
 - BIRTH_DATE <= ?
 - How many people in room born before parameter marker?
 - What if value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

23

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)
```



24

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 ✓

25

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

26

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

27

Possible Recommendations

- REOPT(ALWAYS) (a.k.a REOPT(VARS))
 - 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

28

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 STATE LIKE '%A%'
 - IN predicates with many duplicates
 - WHERE ACCTNO IN (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

29

Agenda

- Predicate filtering stages
- Filter factor challenges
- Impact of Clustering
- Conclusion

30

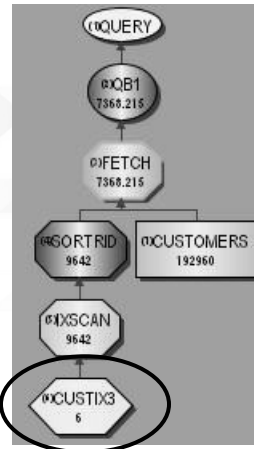
Impact of clustering

- Correct index is chosen
 - But is the index enough to ensure good performance?

```
SELECT *
FROM CUSTOMERS
WHERE COUNTRY = 'USA'
      AND GENDER = 'F'
      AND STATUS = 'N'
```

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

31



How many rows qualify?

- Count shows 1,121 rows qualifying
 - Given that there are 192,960 rows in the table
 - That's 0.6% of the table....excellent filtering!!!

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

32

Is indexing enough?

- 0.6% of the table qualifies.....but.....
 - It is 0.6% of the rows
 - How many data pages do these rows occur on?

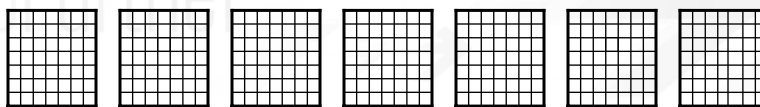
BP1	BPOOL	ACTIVITY	TOTAL

GETPAGES			969

969 data getpages to retrieve 1,121 rows

Current clustering effect

- Table is currently clustered by the unique key
 - ACCTNO
- When retrieving a large number of rows unrelated to ACCTNO, qualified rows are found on many pages
 - 969 getpages from 4020 pages



On average, 1.2 rows found on every 4th page

Data clustering vs scattering

- 1,121 rows qualified, and 48 rows per page
- Number of data pages accessed via index:
 - Min # pages (clustered) = $1121/48 = 24$
 - Max # pages = 1121
 - Pages may or may not be consecutive
 - Actual max getpages is $1121 * 2 = 2242$
 - If updated row is relocated because it does not fit on original page (NEARINDREF/FARINDREF)
- # of getpages
 - = # of qualified pages if list prefetch used
 - \leq # of qualified rows ($* 2$) for random I/O

35

After re-clustering

- After clustering by the data access:
 - STATUS, GENDER, ACCTNO
- Number of getpages 25
 - Compared with original 969

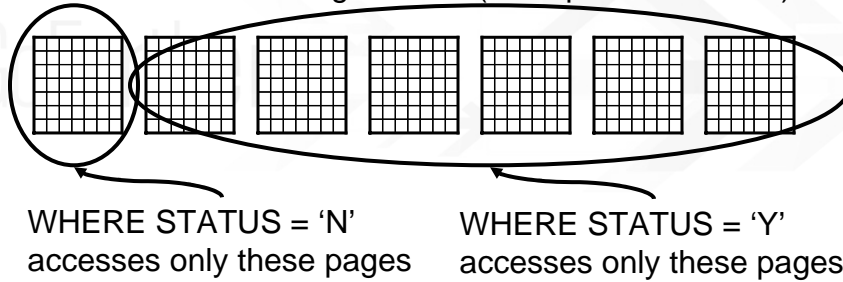
BP1	BPOOL	ACTIVITY	TOTAL
GETPAGES			25

- Note: ACCTNO was added to clustering index so that original clustering is partially maintained

36

New clustering effect - STATUS

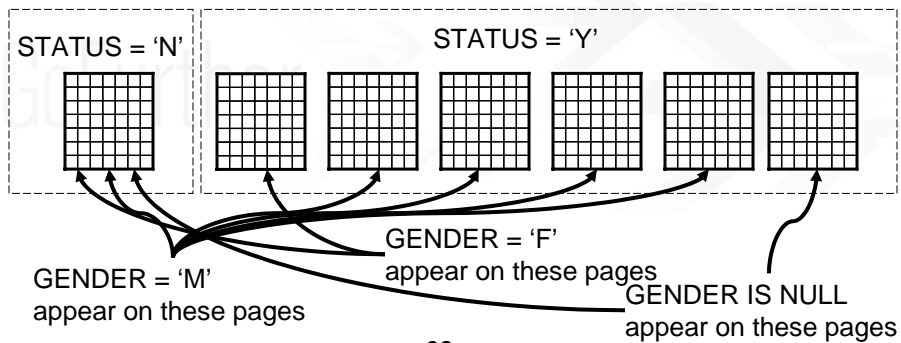
- Clustered by STATUS, GENDER, ACCTNO
 - Status = 95% Y, 5% N
 - The clustering effect is (example not to scale):



37

New clustering effect - GENDER

- Clustered by STATUS, GENDER, ACCTNO
 - Gender = 70% M, 17.5% NULL, 12.5% F
 - GENDER is clustered within each STATUS value
 - The clustering effect is (example not to scale):



38

New clustering effect - ACCTNO

- Clustered by STATUS, GENDER, ACCTNO
 - ACCTNO is clustered within each STATUS/GENDER
 - 6 unique combinations of STATUS/GENDER
 - Compared to original clustering sequence
 - A range of ACCTNOs may be found in 6 clustered pockets, rather than 1

STATUS	GENDER	Clustered ACCTNO range
N	M	1-192955
N	F	6-192959
N	NULL	9-192955
Y	M	2-192960
Y	F	7-192953
Y	NULL	8-192949

STATUS/GENDER values are distributed throughout ACCTNO

39

Clustering vs Clusterratio

- Clustering is the density of the data
- Clusterratio implies the sequential nature of the data
 - For the clustering index, clusterratio and clustering are equivalent
 - For the non-clustering index, high clusterratio generally implies the data is clustered

40

Clustering and partitioning

- Clustering is a logical grouping
 - Inserts attempt to maintain clustering sequence
- Partitioning is a physical grouping
 - Inserts guarantee that rows can only be inserted into the partition dictated by the partition limit keys
 - Can support efficient insert at end processing within partitions
 - Freespace search is limited to a partition
 - Partitioning pruning can be exploited without indexing
- Combine partitioning and clustering (and also UNION ALL in View) for multi-dimensional clustering

41

Conclusions

- When an index provides 10% filtering
 - Is that 10% of the data pages?
 - Or, 10% of the rows on 100% of the data pages?
- Filtering pages is just as important as filtering rows
 - Indexing filters the rows
 - Clustering/partitioning filters the pages
- Cluster/partition by data access, not by the primary key

42

Agenda

- Predicate filtering stages
- Filter factor challenges
- Impact of Clustering
- Conclusion

Re-cap on Filter Factors

- For each WHERE clause predicate
 - The optimizer determines the Filter Factor (FF) using available
 - Frequency statistics
 - Cardinality statistics
 - HIGH2KEY/LOW2KEY
 - Host variables cause optimizer to estimate
 - Column values being evenly distributed
 - Optimistic FFs for range predicates

Re-cap on filtering per object

- For each object (index and/or table)
 - The optimizer uses available correlation statistics to determine how individual FFs should be combined
- More accurately estimating individual FF and how to combine FFs is used to determine
 - Index matching filtering
 - Total index filtering
 - Total table filtering

45

Re-cap on final costing

- Index I/O cost is estimated based upon
 - Matching FF
 - NLEVELS
 - NLEAF
- Data I/O cost is estimated based upon
 - Total index filtering
 - Index clusterratio
 - NPAGESF
-and the lowest cost access path is the winner

46

Why did the optimizer choose that path? – Part 2

Thankyou for listening!!!

GoFurther

Terry Purcell

IBM Silicon Valley Lab
tpurcel@us.ibm.com

