Z08

DB2 for z/OS Query Optimization Statistics

Patrick Bossman

Sept. 12-16, 2005 **Orlando, FL**

Importance of statistics

- **Accurate filter factor estimation is the cornerstone of cost based query optimization**
  - **Differentiate candidate indexes**
    - **Use indexes efficiently (prefetch?)**
  - **Choose most efficient join sequence**
  - **Choose efficient join method**
  - **Choose appropriate sorts**
  - **Avoid inefficient sorts**
- **In general, queries which are accurately costed perform more efficiently and have stable performance**

Filter factor issues

- **Filter factor accuracy important for...**
  - **Index matching**
    - **Accurately estimate index cost**
  - **Total index filtering**
    - **Estimate table access cost via index(es)**
    - **Choose how to use index (prefetch?)**
  - **Total table level filtering**
    - **Efficient join order**
    - **Efficient join method**
    - **Appropriate sorts**

Terminology

- **Correlation**
  - When data on two columns is not independent
  - Eg. CITY, STATE
    - Every city does not exist in every state.
- **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. Gender
    - Domain (M, F)
    - 35% = M, 65% = F

Terminology (cont.)

- **MFREQ: Multi-column frequency**
  - **Frequency on concatenated column group**
  - **MFREQ(C1,C2,C3)**
- **MCARD: Multi-column cardinality**
  - **Multi-column cardinality on a column group**
  - **MCARD(C1,C2,C3)**

Selectivity statistics

- **Single column**
  - **Cardinality**
  - **HIGH2KEY/LOW2KEY**
  - **Frequency**
- **Multi-column**
  - **Cardinality**
  - **Frequency**

Single column cardinality

- **Single column cardinality**
  - **Number of distinct values for a column**
  - **Assumes uniform distribution**
  - **Stored as**
    - **SYSCOLUMNS.COLCARDF**
    - **SYSINDEXES.FIRSTKEYCARDF**
  - **Used when better statistics can't be used...**
    - **Host variables, parameter markers, special registers**
    - **No other statistics available**

RUNSTATS column cardinality

- **How to collect**
  - **–RUNSTATS command:**

RUNSTATS TABLESPACE (DBNAME.TSNAME)
TABLE (ALL or PAT_TABLE)
COLUMN(ALL or <list of columns>)

  - **–Leading column of index when RUNSTATS on index performed**

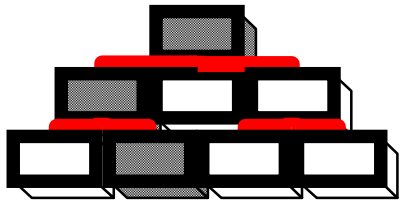
RUNSTATS INDEX (PAT_INDEX)

RUNSTATS TABLESPACE (DBNAME.TSNAME)
INDEX(ALL)

Single column cardinality

Select C2 from T1
Where C1 = ?
Index I1: C1,C2,C3

T1 CARDF = 100,000
C1 COLCARDF = 5
I1 NLEAF = 10,000
I1 NLEVELS = 3

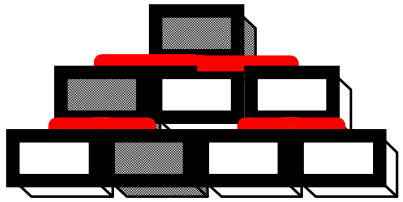


- For equals predicate, filter factor = 1/COLCARDF
  - –Index pages --> probe + matching FF * NLEAF
    - •3 + (1/5) * 10,000 = 2003 index pages
  - –Index record ids processed = CARDF * matching index filtering
    - •100,000 * (1/5) = 20,000
  - –Rows returned = CARDF * total filtering
    - •100,000 * (1/5) = 20,000

Single column cardinality

Select C3 from T1
Where C1 = ?
AND C2 = ?
Index I1: C1,C2,C3

C1 COLCARDF = 5
C2 COLCARDF = 10
FULLKEYCARDF 65,000

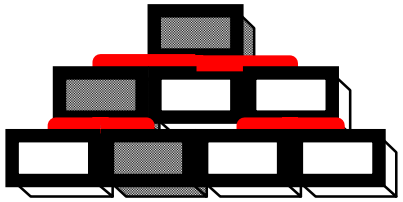


- Two matching predicates, multiply filter factors
  - –Index pages --> probe + matching FF * NLEAF
    - •3 + [(1/5) * (1/10)] * 10,000 = ~203 index pages
  - –Index record ids processed = CARDF * matching index filtering
    - •100,000 * [(1/5) * (1/10)] =~ 2,000
  - – qualified rows = CARDF * total filtering
    - •100,000 * [(1/5) * (1/10)] =~ 2,000 rows

Single column cardinality

Select C2 from T1
Where C1 > ?
Index I1: C1,C2,C3

C1 COLCARDF = 10,121



- Range predicate with parameter marker
  - Use default interpolation filter factor chart
    - COLCARDF 10,121 --> FF = 1/100
  - 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 '1930-01-01'?
      - What if value is '1980-01-01?
      - Cannot accurately estimate without literal value

Range predicate interpolation

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
>=0	1/3	1/10

Table 104. Default filter factors for interpolation

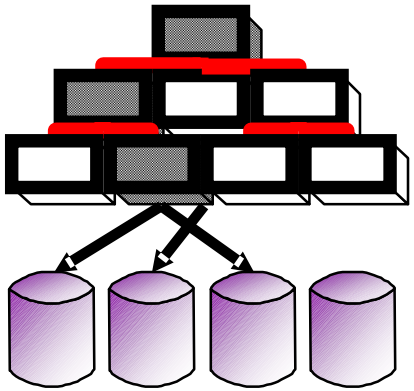
Note: Op is one of these operators: <, <=, >, >=.

COMMENT: This is DB2 s'documented guess for an impossible to estimate filter factor.

Single column cardinality

Select C4 from T1
Where C1 = ?
AND C3 > ?
Index I1: C1,C2,C3

CARDF = 1 million
NPAGES/F = 100,000
NLEAF = 10,000
C1 COLCARDF 10
C3 COLCARDF 10,121



clusterratiof 50

- Matching cost
  - –Index pages --> probe + matching FF * NLEAF
    - •probe + (1/10) * 10,000 =~ 1,003 pages
  - –Index rows processed = CARDF * Matching FF
    - •1,000,000 * (1/10) = 100,000 rows
- Screening
  - –Rows to access table for = CARDF * (Matching and screening FF)
    - •1,000,000 * [(1/10) * (1/100)] =~ 1,000 rows???

HIGH2KEY/LOW2KEY

- HIGH2KEY/LOW2KEY
  - Single column statistic
    - SYSCOLUMNS.HIGH2KEY
    - SYSCOLUMNS.LOW2KEY
  - When used?
    - Interpolation used to estimate range predicates
    - Like, between, <, <=, >, >=
    - Literal value must be known
    - As domain statistics when COLCARDF = 1 or 2
    - Can be used in combination with single column frequencies for more accurate estimate.

    - DB2 Interpolation: Technique to estimate the percentage of rows which qualify based on known high / low values.

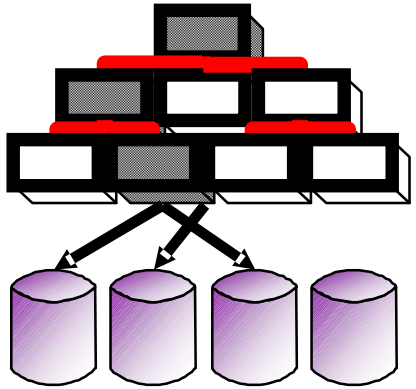
RUNSTATS HIGH2KEY / LOW2KEY

- How to collect
  - Whenever single column cardinality collected, HIGH2KEY / LOW2KEY also collected.
  - Reference RUNSTATS COLUMN CARDINALITY slide

Linear interpolation

Select C4 from T1
Where C1 = ?
AND C3 > 50
Index I1: C1,C2,C3

CARDF = 1 million
NPAGES/F = 100,000
C3 COLCARDF 10,241
LOW2KEY 0
HIGH2KEY 100



clusterratiof 50

- Matching cost - same as before
- Screening
  - –Rows to access table for = CARDF * (Matching + screening FF)
  - –1,000,000 * [(1/10) * (screening FF)] =~ ???
  - –Interpolation for C3 > 50
    - •(HIGH2KEY - LITVALUE) / (HIGH2KEY - LOW2KEY)
    - •(100 - 50) / (100 - 0) = 50/100 = 0.5
  - –1,000,000 * [(1/10) * (0.5)] =~ 50,000 rows (vs 1000 with def FF)

Single column frequencies

- Single column frequencies
  - SYSCOLDIST.FREQUENCYF
    - TYPE = 'F', NUMCOLUMNS = 1
  - Provides non-uniform distribution information
    - Data skew
  - When used?
    - Literal value must be known
    - Equals, is null, in
    - Like, between, <, <=, >, >=
    - Used in conjunction with other complementary statistics

RUNSTATS Frequency

- How to collect
  - V7 RUNSTATS only collectes on leading column of index

INDEX (PAT_INDEX) columns (C1,C2,C3)

RUNSTATS INDEX (PAT_INDEX)

Collect more than top 10:
RUNSTATS INDEX
(PAT_INDEX FREQVAL NUMCOLS(1) COUNT(20))

Eliminate existing frequencies:
(PAT_INDEX FREQVAL NUMCOLS(1) COUNT(0))

RUNSTATS Frequency

- How to collect
  - –V7 RUNSTATS only collects on leading column of index
    - •INDEX (PAT_INDEX) column (C1,C2,C3)
    - •Default collects top 10

  - –RUNSTATS defaults to top 10

RUNSTATS INDEX (PAT_INDEX)

  - –Top 20

RUNSTATS INDEX
(PAT_INDEX FREQVAL NUMCOLS(1) COUNT(20))

  - –Purge frequencies:

(PAT_INDEX FREQVAL NUMCOLS(1) COUNT(0))

RUNSTATS Frequency

- How to collect
  - V8 RUNSTATS allows collection on almost any column

```
RUNSTATS TABLESPACE DB1.TS1
TABLE (PAT_TABLE) COLUMNS(C1,C2,C3)
COLGROUP (C1) FREQVAL COUNT(1) MOST
COLGROUP (C2) FREQVAL COUNT(10) LEAST
COLGROUP (C3) FREQVAL COUNT(20) BOTH
```

  - Eliminate existing frequencies:

```
COLGROUP (C3) FREQVAL COUNT(0) MOST
```

DSTATS Frequency

- How to collect
  - –DSTATS works in DB2 V6 and DB2 V7. It WILL NOT work in V8
  - –DSTATS can collect on most columns
    - •DSTATS is program which runs SQL to generate some statistics which RUNSTATS can't currently collect
  - –Collect frequencies for top 20 occurring values of column C1

```
VALUES 20,0
SYSADM.PAT_TABLE.C1
```

  - –Collect top 10 values for all columns of table

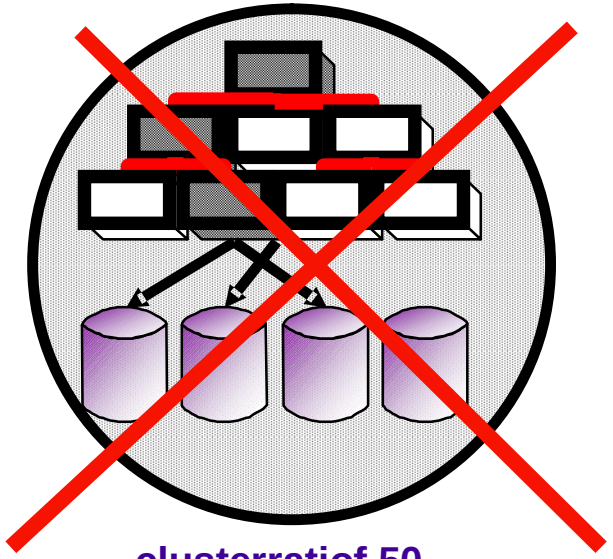
```
VALUES 10,0
SYSADM.PAT_TABLE.*
```

Single column frequency

Select C4 from T1
Where C1 = 'A'
Index I1: C1,C2,C3

T1 CARDF = 100,000
C1 COLCARDF = 5
I1 NLEAF = 10,000
I1 NLEVELS = 3

C1	FREQ
'A'	0.75
'B'	0.15
'C'	0.05
'D'	0.03
'E'	0.02



clusterratiof 50

- Value which exists a lot
  - –Index pages --> probe + matching FF * NLEAF
    - •3 + (0.75) * 10,000 =~ 7,503 index pages
  - –Index record ids processed = CARDF * matching index filtering
    - •100,000 * (0.75) =~ 75,000
  - –Rows returned = CARDF * total filtering
    - •100,000 * (0.75) =~ 75,000

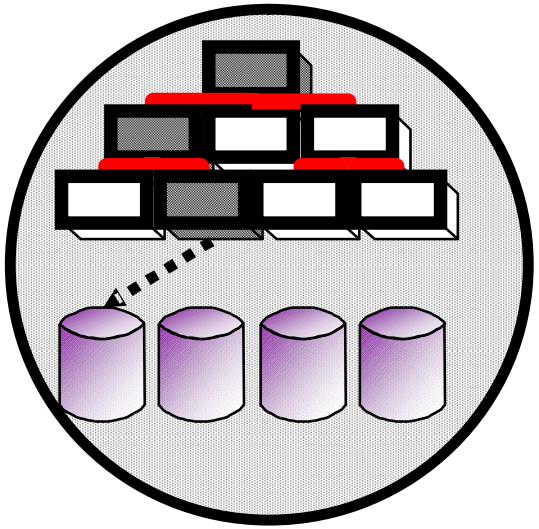
Single column frequency

Select C4 from T1
Where C1 = 'Z'
Index I1: C1,C2,C3

T1 CARDF = 100,000
C1 COLCARDF = 5
I1 NLEAF = 10,000
I1 NLEVELS = 3

?

C1	FREQ
'A'	0.75
'B'	0.15
'C'	0.05
'D'	0.03
'E'	0.02



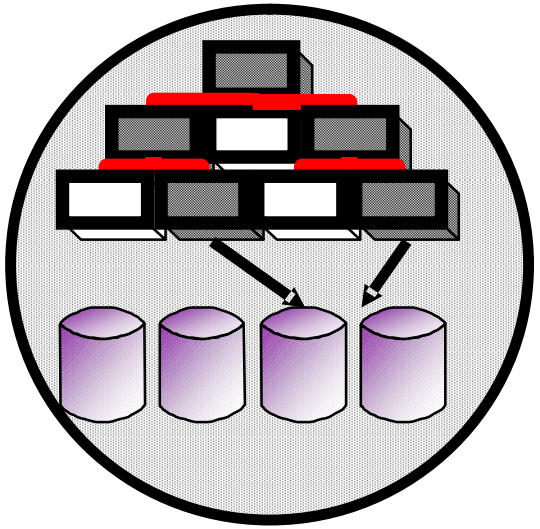
- Looking for value not in the domain.... **clusterratiof 0.50**
  - Index pages --> probe + matching FF * NLEAF
    - 3 + (~0) * 10,000 =~ 1 index page
  - Filter factor without the frequencies
    - 1/5 = 0.20

Single column frequency

Select C2 from T1
Where C1 in ('C', 'Z', 'E')
Index I1: C1,C2,C3

T1 CARDF = 100,000
C1 COLCARDF = 5
I1 NLEAF = 10,000
I1 NLEVELS = 3

C1	FREQ
'A'	0.75
'B'	0.15
'C'	0.05
'D'	0.03
'E'	0.02



- Some in domain, some not...
  - –Index pages --> probe + matching FF * NLEAF
    - •3 + (0.05 + 0.0 + 0.02) * 10,000 =~ 700 index page
  - –Rows returned = CARDF * total filtering
    - •100,000 * (0.05 + 0.0 + 0.02) =~ 7000
  - –Without frequencies filter factor = 3/5 = 0.60 versus 0.07

Single column recommendations

- Cardinality
  - Collect on all columns used in where clause
  - Used regardless of literal value known
- Interpolation (HIGH2KEY/LOW2KEY)
  - Collected with column statistics
  - Consider REOPT(VARS), V8 - REOPT(ONCE)
- Frequency
  - Literal values must be known
  - Used for most predicate types
  - Collect all values for low COLCARDF columns
  - Useful for indexed and non-indexed columns

Filter factor issues

- Filter factor accuracy important for...
  - Index matching
    - Estimate index cost
  - Total index filtering
    - Estimate table access cost via index(es)
    - Choose how to use index (prefetch?)
  - Table filtering
    - Efficient join order
    - Efficient join method
    - Appropriate sorts

Multi-column cardinalities

- Multi-column cardinalities (MCARD)
  - Stored in a few places...
    - SYSINDEXES.FULLKEYCARDF
    - SYSCOLDIST.CARDF
      - TYPE = 'C', NUMCOLUMNS > 1
  - Assumes uniform distribution
  - When used?
    - Primarily for indexes
    - Literal values not necessary
    - KEYCARD for partially matching indexes
      - Collect for all indexes with 3 or more columns
    - Collect to support multi-column frequencies
    - Collect for all multi-column join situations

RUNSTATS KEYCARD

- How to collect
  - V7 RUNSTATS only collects KEYCARD on leading column of index
  - By default, RUNSTATS only collects FIRST/FULLKEYCARDF

INDEX PAT_INDEX (C1,C2,C3,C4)

RUNSTATS INDEX(PAT_INDEX KEYCARD)

  - MCARD on leading concatenated column groups:
  - MCARD(C1,C2), MCARD(C1,C2,C3)

RUNSTATS COLGROUP

- DB2 V8 allows collection of MCARD on any column group

RUNSTATS TABLESPACE DB1.TS1
TABLE(PAT_TABLE) COLUMN(C1,C2,C3,C4)
COLGROUP(C1,C4)

Specifying COLGROUP with multiple columns collects multi-column cardinality on the group.

DSTATS MCARD

- How to collect
  - DSTATS only works for V6, V7.
  - V7 RUNSTATS only collects KEYCARD on leading column of index. DSTATS can be used for other columns
  - Collect column group cardinality on column group (C3,C4)
  - CARDINALITY ONLY
  - SYSADM.PAT_TABLE.C3,C4
  - DSTATS supports up to three columns.

Multi-column cardinality

- Useful for local predicates

```
SELECT    name, address, ...
FROM      CUST
WHERE     City = ?
AND       State = ?
AND       Last_name = ?
```

INDEX	Index column	1stkey	KEYCARD	FULLKEY
IX1	City, State, Zip	10,000	12,000	1,000,000
IX2	Last_name	20,000	N/A	20,000

COLCARDF STATE	50
IX1 Filter Factor without KEYCARD	1 / 500,000
IX1 Filter Factor WITH KEYCARD	1 / 12,000
Last_name	1 / 20,000

keycard!!!

Multi-column cardinality

- Useful for join predicates

```
SELECT    cols
FROM      T1, T2
WHERE     T1.C1 = T2.C1
AND       T1.C2 = T2.C2

INDEX I1 (C1,C2) on table T1
INDEX I2 (C1,C2,C3) on table T2
```

KEYCARD is necessary on index I2 to accurately estimate T1 à T2 join size.

Multi-column cardinality

- Matching + Screening

```
SELECT    cols
FROM      T1
WHERE     T1.C1 = ?
AND       T1.C3 = ?
AND       T1.C4 = ?
```

INDEX I2 (C1,C2,C3) on table T1

COLCARDF / FIRSTKEYCARDF for matching only

COLGROUP (C1,C3) ß matching + screen

COLGROUP (C1,C3,C4) ß table level filtering

Filter factor issues (reminder)

- Filter factor accuracy important for...
  - Index matching
    - Estimate index cost
  - Total index filtering
    - Estimate table access cost via index(es)
    - Choose how to use index (prefetch?)
  - Table filtering
    - Efficient join order
    - Efficient join method
    - Appropriate sorts

Multi-column frequency

- Multi-column frequencies
  - Very similar to single column frequencies
    - Distribution statistics concatenated column group values
    - Identifies multi-column skewed distributions
  - Stored in
    - SYSCOLDIST.FREQUENCYF
    - TYPE = F‘ ’
    - NUMCOLUMNS > 1

RUNSTATS INDEX

- How to collect
  - –V7 RUNSTATS only collects on leading concatenated column of index
  - –RUNSTATS does NOT collect multi-column frequencies by default.
  - –Must be explicitly requested.

INDEX (I1) columns (C1,C2,C3)

  - –Collect top 15 values for column group (C1,C2)

RUNSTATS INDEX (I1 FREQVAL NUMCOLS(2) COUNT(15))

  - –Eliminate frequencies on column group (C1,C2):

RUNSTATS INDEX (I1 FREQVAL NUMCOLS(2) COUNT(0))

RUNSTATS COLGROUP

- DB2 V8 allows collection of multi-column frequencies on almost any column group
  - Examples

```
RUNSTATS TABLESPACE DB1.TS1
TABLE (T1) COLUMN(C1,C2,C3)
COLGROUP(C1,C3) FREQVAL COUNT(10) MOST
COLGROUP(C2,C3) FREQVAL COUNT(1) LEAST
```

  - Eliminate frequencies on column group (C1,C3):

```
COLGROUP (C1,C3) FREQVAL COUNT(0)
```

DSTATS

- How to collect
  - DSTATS available for use in V6, and V7 only
  - DSTATS can collect on most column groups
    - Maximum of 3 columns
    - Not all data types supported
  - Top 20 occurring values of column C4,C5

VALUES 20,0
SYSADM.PAT_TABLE.C4,C5

Comment: DSTATS performs R-scan or non-matching index scan for EACH column. DSTATS automates the statistics generation and formatting required for SYSIBM.SYSCOLDIST

Multi-column frequency

- Multi-column frequencies
  - Limited use
    - Boolean equal predicates only
    - Always collect supporting multi-column cardinality
  - Collect single column frequencies for
    - Range predicates
    - In-lists
    - Single column predicates
    - other non-equal predicates

Multi-column frequency

Gender	COLCARDF = 2
Category	COLCARDF = 4
(Gender,Category)	MCARD = 8

Category	Gender	FREQUENCYF	1/MCARD
Women's Health	F	0.2375	0.125
Women's Health	M	0.0125	0.125
Men's Health	M	0.2375	0.125
Men's Health	F	0.0125	0.125
Hockey	M	0.15	0.125
Hockey	F	0.10	0.125
Soccer	F	0.15	0.125
Soccer	M	0.10	0.125

Multi-column considerations

- Cardinality
  - Collect KEYCARD for all indexes with 3 or more columns
  - Literal values not required
- Frequencies
  - Not useful when literal values aren't known
  - Dynamic SQL prepared with parameter markers
  - Static SQL with hostvars (consider REOPT)
  - Special registers (consider REOPT)
  - Collect for specific cases
- Pay special attention to...
  - Low cardinality column groups
  - Volatile data

References

- Hint and Tip document available on web
  - How to collect cardinality and frequency statistics

  http://www.ibm.com/software/data/db2/os390/support.html
    - Search DB2 online support documentation for hints and tips
    - Search for DSTATS
- DSTATS functionality has been integrated with RUNSTATS in DB2 V8
  - Improved performance, statistics management issues eliminated
  - DSTATS program will cease to exist in V8

One more time

- Filter factor accuracy important for...
  - Index matching
    - Estimate index cost
  - Total index filtering
    - Estimate table access cost via index(es)
    - Choose how to use index (prefetch?)
  - Table filtering
    - Efficient join order
    - Efficient join method
    - Appropriate sorts

Statistics Advisor (SA)

- **Statistics Advisor**
  - **Tool to automate the determination of statistics for a specific query**
  - **Input SQL statement**
  - **Output RUNSTATS commands**
- **Session Z09 contains specifics**
  - **A guide to Visual Explain**
  - **Section showing how to run and configure SA**

DB2 for z/OS Query Optimization Statistics
Session: Z08

Patrick Bossman

IBM

bossman@us.ibm.com