The Effect of Correlation and Distribution Statistics on the DB2 Optimizer

Tom Moulder
TREX Associates, Inc.

May 7, 2007 1:40 p.m. – 2:40 p.m.

Platform: DB2 for z/OS
Presentation’s Key Objectives

- The attendee will learn how the various inputs to the Optimizer determine the selected access path.
- The attendee will learn what distribution statistics are and how they influence access path selection in the DB2 Optimizer.
- The attendee will learn what correlation statistics are and how they influence access path selection in the DB2 Optimizer.
- The attendee will learn of changes in the DB2 Optimizer in Version 8 and how these changes can be exploited to improve performance during Version migration.
- The attendee will learn how to capture distribution and correlation statistics and update the catalog so that the DB2 Optimizer can use these values to improve access paths.
Agenda

Introductory Comments
Distribution Statistics
Correlation Statistics
Collecting these Statistics
Summary
Introductory Comments

DB2 Optimizer choices

- **Static SQL**
  - Bind/Rebind Time
  - Automatic Rebind Option

- **Dynamic SQL**
  - Mini-Bind at Execution
  - Effect of Dynamic SQL Cache
SQL Access Paths

Bind Time Access Path Choices

- Direct Row access using ROWID
- One Fetch Index Scan using Min, MAX
- Unique Matching Index Scan using a predicate value
- Matching Index Scan Only
- Non-Matching Index Scan Only
- Matching Index Cluster Scan
- Matching Random Index Scan
SQL Access Paths

Bind Time Access Path Choices Continued

- Multiple Matching Index Scan using AND and OR
- Non Matching Cluster Index Scan
- Segmented Table Space Scan
- Non Segmented Table Space Scan (in parallel or sequential)
- Non-matching Random Index Scan

Listed in order of the Optimizer Preference
Optimizer Input Affecting APS

Statistics
- Cardinality
- Distribution
- Correlation

Predicates
- Matching
- Non-matching
Optimizer Input Affecting APS

Directives
- Optimize for “n” Rows
- Fetch First “n” Rows Only

DSNZPARM
- MAX_OPT_CPU
- MAX_OPT_ELAP

Externals
- Buffer Pools
- Processors
SQL Access Paths

Access Path Choices Documented

- **Explain**
  - Bind Time Option with a value of YES
    - Actually changes what DB2 will do
  - Anytime as a DB2 command
    - Makes no changes within DB2
    - Reporting is the only function

- **Visual Explain**
  - Free Download
  - Reports through Windows Applet (JAVA)
  - Requires DB2 Connect
  - Personal Version is provided for free
SQL Access Paths

Key to all of these choices?
- Filter Factors

Why?
- Used to Estimate I/O count at execution
- More I/O increases estimated costs
- Less I/O reduces estimated costs

Key word on this slide
- ESTIMATED!
SELECT *
FROM GROUP G INNER JOIN PEOPLE P
ON G.ID_NAME = P.ID_NAME
WHERE
  G.NOTICE = 'NOW'
  AND G.STATUS_NOTICE = 'AD'
  AND P.PEOPLE_GROUP = 'ST'
  AND P.COUNTRY = 'AM'
  AND P.STATUS = 'BEAT'
  AND P.CARE_STATE = 'M'

Example SQL to examine
Join of Two Tables
Sequence of Join is Important
Join Sequence
- “P” Table is First
- “G” Table then is Second

Is this Good?
- Determine Selected Rows for Both Tables
- Based on SQL Predicate Analysis
- Notice the Math based on “AND” Usage
- “OR” is not present in the Sample
Distribution Statistics

Visual Explain – or Some Other Explain Tool
- Find the SQL Statement
- Locate Cardinality Estimates by Table

No Tool or not on DB2 V8?
- Decompose the SQL
- Execute to Determine Row Counts
- See Next Slide for an Example
Distribution Statistics

SELECT COUNT(*)
FROM GROUP G
WHERE
  G.NOTICE = 'NOW'
  AND G.STATUS_NOTICE = 'AD'

SELECT COUNT(*)
FROM PEOPLE P
WHERE
  P.PEOPLE_GROUP = 'ST'
  AND P.COUNTRY = 'AM'
  AND P.STATUS = 'BETT'
  AND P.CARE_STATE = 'M'

Column Names replaced by Count(*)
Predicate is Unchanged
Result Set Contains Actual Cardinality – Not Estimate
Distribution Statistics

<table>
<thead>
<tr>
<th>Column</th>
<th>Estimated Distribution</th>
<th>Actual Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON_GROUP</td>
<td>11.11%</td>
<td>86.43%</td>
</tr>
<tr>
<td>COUNTRY</td>
<td>0.44%</td>
<td>90.40%</td>
</tr>
<tr>
<td>STATUS</td>
<td>3.45%</td>
<td>95.75%</td>
</tr>
<tr>
<td>CARE_STATE</td>
<td>8.33%</td>
<td>83.06%</td>
</tr>
</tbody>
</table>

Notice the Disparity between the Estimate and Actual
Why? – Lack of Distribution Statistics in the DB2 Catalog
Fact – DB2 assumes Equal Distribution if no Statistics Exist
Distribution Statistics

Total Filter Factor Calculation by Optimizer -
- Original Estimate
  - \((0.1111 \times 0.0044 \times 0.0345 \times 0.0833) = 0.00014\)
- Actual Count Filter Factor
  - \((0.8643 \times 0.9040 \times 0.9675 \times 0.8306) = 0.62787\)

Filter Factor is Multiplied by Table Row Count
Result Determines Estimated I/O

- Original Estimate
  - \(1,000,000 \times 0.00014 = 140\) Rows
- Actual Count Filter Factor
  - \(1,000,000 \times 0.62787 = 627870\) Rows
Distribution Statistics

A Few Points

- Predicates in Example were all Equal
- Statistics Collected for “n” values
- Non-Matching Values still have a more accurate Estimate
- Each Column Filter Factor Multiplied to Calculate Total Filter Factor for Table
- Row Count Translated to Estimated I/O Count
Distribution Statistics

What if Predicates are something other than Equals?

How about an Example with Range Predicates

- One Column to Make it Simple
- Possible Values are 0 - 999

<table>
<thead>
<tr>
<th>Values for Column_A</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 99</td>
<td>20%</td>
</tr>
<tr>
<td>100 – 199</td>
<td>10%</td>
</tr>
<tr>
<td>300 – 399</td>
<td>25%</td>
</tr>
<tr>
<td>600 – 699</td>
<td>40%</td>
</tr>
<tr>
<td>900 – 999</td>
<td>5%</td>
</tr>
</tbody>
</table>
Distribution Statistics

“Where Column_A Between 800 and 899”
- No Distribution Statistics
  - Filter Factor of .1
- Distribution Statistics from Previous Slide
  - Filter Factor of 0

“Where Column_A Between 600 and 699”
- No Distribution Statistics
  - Filter Factor of .1
- Distribution Statistics from Previous Slide
  - Filter Factor of .4
Distribution Statistics

A Few Points

- One Column in the Predicate Simplifies
- Multiple Columns Calculated like the First Example
- Ranges here are sums of actual values
- Any Range would use Distribution Statistics
Correlation Statistics

Another Example

- Table Named Texas
  - Columns named Zip and City

What we all know

- A Zip Code can only exist in One City

Therefore

- A Correlation Exists between Zip and City
Example SQL

```
SELECT *
FROM TEXAS
WHERE CITY = 'CLEBURNE'
AND ZIPCODE = '76031'
```
Correlation Statistics

What if the Correlation was not Inherent?
Counting Distinct Values Will Tell.
Selects for the Current Example

```
SELECT COUNT(DISTINCT CITY) FROM TEXAS;

SELECT COUNT(DISTINCT ZIPCODE) FROM TEXAS;

SELECT COUNT(*) FROM (SELECT DISTINCT CITY, ZIPCODE FROM TEXAS) AS A;
```
Correlation Statistics

What to do with the Counts?

- Multiply Column 1 Count by Column 2 Count
- Is the Count of the Two Columns Together Lower?
  - No – Then no Correlation Exists
  - Yes – Then Correlation Exists
- Correlation Varies and can be Significant
- Collecting Statistics will Assist DB2
Correlation Statistics

The TEXAS Table Example
- Cities in Texas = 1,463
- Zip Codes in Texas = 2,692

Multiply the two Columns Together
- 3938396

Distinct Values for the Two Columns Together
- 2692

Highly Correlated
Correlation Statistics

Now for the Effect on Filter Factors - Estimates

- **City**
  - \( \frac{1}{1463} = 0.00068 \)

- **Zip**
  - \( \frac{1}{2692} = 0.00037 \)

- **Total Filter Factor**
  - \( 0.00068 \times 0.00037 = 0.0000002 \)
Correlation Statistics

Now for the Effect on Filter Factors - Reality

- City
  - \( \frac{1}{1463} = 0.00068 \)

- Zip
  - \( \frac{1}{2692} = 0.00037 \)

- Total Filter Factor
  - 0.00068
  - Because of Correlation

- 3400 Times Different

- Estimated I/O is 3400 times less than Reality
Collecting these Statistics

Let's talk about Correlation first

- Example Control Cards Below
- COLGROUP is the Key

RUNSTATS TABLESPACE
DSNSTATE.DSNTEXAS
TABLE(STATES.TEXAS)
COLGROUP(CITY,ZIP) FREQVAL COUNT 10 BOTH
Collecting these Statistics

Statistics Stored in DB2 Catalog

- SYSCOOLDIST and SYSCOOLDISTSTATS are the Catalog Table Names for Storing
- Correlation Statistics Represent
  - Unique Values based on the Combination of Columns Specified in COLGROUP
  - COUNT Specifies the number of Values Stored
  - BOTH Specifies to Collect LEAST and MOST Frequently Occurring Values
Collecting these Statistics

Let's talk about Distribution Now
- Example Control Cards Below
- COLUMN is the Key

RUNSTATS TABLESPACE DSNSTATE.DSNTEXAS TABLE(STATES.TEXAS) COLUMN(CITY,ZIP) FREQVAL COUNT 10 BOTH
Collecting these Statistics

Statistics Stored in DB2 Catalog

- SYSCOLDIST and SYSCOLDISTSTATS are the Catalog Table Names for Storing

- Distribution Statistics Represent
  - Unique Values based for each Column Specified in COLUMN
  - COUNT Specifies the number of Values Stored
  - BOTH Specifies to Collect LEAST and MOST Frequently Occurring Values
Collecting these Statistics

How Often Do You Collect These Statistics?
- How Often Does the Data Change?
- Static Data
  - One Time Collection
- Highly Variable Data
  - Collect When Data Changes by 10%
Summary

Steps to Follow

- Collect Statistics
- Bind/Re-Bind Plans or Packages
  - Make sure to use Explain(Yes) Parameter
- Use Explain to Check Access Path Changes
- Use Monitor to Determine Resource Savings
- Important Point to Think About
  - NO SQL CHANGES REQUIRED
Questions?
Contact TREX Associates, Inc.

Tom Moulder
9728 Delmonic o Dr.
Keller, TX.  76248-9559
+1 817-741-5549 Office
+1 682 558-6527 Mobile
Tom.moulder@trexassociates.com
www.t-rex-associates.com
Session: A03

The Effects of Correlation and Distribution Statistics on the DB2 Optimizer

Tom Moulder
TREX Associates, Inc.
tom.moulder@trexassociates.com