

How an Access Path gets selected

A presentation to the Tridex DB2 User Group March 2007

Cost Based Optimization

- All “possible” Access Paths are evaluated and placed in a list
- The most cost effective is selected
- CPU Cost and I/O Costs are weighted
- The “best” Access Path in the list is selected
- This is what is entered in the PLAN_TABLE, STATEMENT_TABLE and FUNCTION_TABLE

What is Reported

- The PLAN_TABLE has the Access Path
- The DSN_STATEMNT_TABLE has the estimated costs
- The DSN_FUNCTION_TABLE has the function information up to V8 where only UDFs are reported
- Remember this information is only for the Access Path selected not the others

Access Path Preference

- 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

Access Path Preference

- 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

Creating Access Path List

- The list is created by an algorithm
- The most likely to work are first
- Based on that list what might be better is tried
- If MAX_OPT_CPU is exceeded the list is stopped
- If MAX_OPT_ELAP is exceeded the list is stopped

Some Examples and Warnings

- These are 13 Examples of changes and their effects
- **Do NOT put a PLAN_TABLE in a Unicode Tablespace**
- BIND will work (There is a PTF that stops BIND from working) and Explain will not
- The PLAN_TABLE is still EBCDIC, as are all the other EXPLAIN tables
- **Do NOT create a V8 version PLAN_TABLE until you are in NFM unless you don't intend to build indexes on PLAN_TABLE**

What was done

- Created a test program that selected data from SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS
- 2 SQL statements
- Bind and EXPLAIN on DB2 V6, V7, and V8
- Made copies of SYSTABLES and SYSCOLUMNS for test cases 6 to 13
- The same programs were run on V5, V6, V7 and V8
- We changed SQL, Statistics, Indexes and some DSNZPARMs

First SQL

- SELECT NAME, TBNAME, COLNO, COLTYPE, LENGTH FROM SYSIBM.SYSCOLUMNS WHERE TBNAME = :SYSCOLUMNS.SYSCOL-TBNAME

Second SQL

- DECLARE SELECT-1 CURSOR FOR SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH FROM SYSCOLUMNS A INNER JOIN SYSTABLES B ON A.TBNAME = B.NAME AND A.TBCREATOR = B.CREATOR WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR

Case 1 SQL Change Where Clause

- old SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT NAME, TBNAME, COLNO, COLTYPE, LENGTH
- FROM SYSIBM.SYSCOLUMNS
- WHERE TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- new SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT NAME, TBNAME, COLNO, COLTYPE, LENGTH
- FROM SYSIBM.SYSCOLUMNS
- WHERE TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR

Case 1 Version 6 Before & After

- STATEMENT 327 THIS IS THE OLD ACCESS PATH.
- STATEMENT# 327 DECLARE SELECT-1 CURSOR FOR SELECT NAME , TBNAME , COLNO , C
- OLTYPE , LENGTH FROM SYSIBM . SYSCOLUMNS WHERE TBNAME = : H
- STEP 1 ACCESSES TABLE SYSIBM .SYSCOLUMNS
- USING INDEX SYSIBM .DSNDCX01 USING 0 COLUMN(S).
- THE ACCESS WILL USE SEQUENTIAL PREFETCH
- THE INTENT LOCK FOR THE TABLE IS IS
- THE TIMESTAMP FOR THIS EXPLAIN IS 2004040514113054
- THE TABLE HAS 1,692 ROWS OF 981 BYTES.
- THE UNIQUE INDEX HAS 1,692 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 84
- DB2 ESTIMATES THIS QUERY WILL USE 23 MILLISECONDS OF CPU TIME
- DB2 ESTIMATES THIS QUERY WILL USE 23 SERVICE UNITS
- STATEMENT 327 THIS IS THE NEW ACCESS PATH.
- STATEMENT# 327 DECLARE SELECT-1 CURSOR FOR SELECT NAME , TBNAME , COLNO , C
- OLTYPE , LENGTH FROM SYSIBM . SYSCOLUMNS WHERE TBNAME = : H
- AND TBCREATOR = : H
- STEP 1 ACCESSES TABLE SYSIBM .SYSCOLUMNS
- USING INDEX SYSIBM .DSNDCX01 USING 2 COLUMN(S).
- THE INTENT LOCK FOR THE TABLE IS IS
- THE TIMESTAMP FOR THIS EXPLAIN IS 2004040514143610
- THE TABLE HAS 1,692 ROWS OF 981 BYTES.
- THE UNIQUE INDEX HAS 1,692 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 84
- DB2 ESTIMATES THIS QUERY WILL USE 1 MILLISECONDS OF CPU TIME
- DB2 ESTIMATES THIS QUERY WILL USE 1 SERVICE UNITS
- BND206I COMPARE COMPLETE FOR PROGRAM = TEST01 VERSION = 2004-02-25-16.02.30.638796
- PREVIOUS VERSION = 2004-02-18-22.14.56.638599

Summary for TEST01

	V6	V7	V8
Before MSU's	23	43	9
After MSU's	1	1	2
Access change	Mc=2 No PF	Mc=2 No PF	Mc=2 No PF

Case 1 Results

- SQL changes the index from match columns = 0 to match columns=2 – noticeable improvement in estimated MSU's
- All versions turn off sequential prefetch

Case 2 SQL Change Select to Join

- old SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT NAME, TBNAME, COLNO, COLTYPE, LENGTH
- FROM SYSIBM.SYSCOLUMNS
- WHERE TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- new SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
- FROM SYSIBM.SYSCOLUMNS A
- INNER JOIN
- SYSIBM.SYSTABLES B
- ON A.TBNAME = B.NAME
- AND A.TBCREATOR = B.CREATOR
- WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME

Case 2 Version 6 Before & After

- STATEMENT# 327 DECLARE SELECT-1 CURSOR FOR SELECT A . NAME , A . TBNAME , A
- . COLNO , A . COLTYPE , A . LENGTH FROM SYSIBM . SYSCOLUMNS
- A INNER JOIN SYSIBM . SYSTABLES B ON A . TBNAME = B . NAME
- AND A . TBCREATOR = B . CREATOR WHERE A . TBNAME = : H WITH
- RR
- STEP 1 ACCESSES TABLE SYSIBM .SYSTABLES B
- USING INDEX SYSIBM .DSNDTX02 USING 0 COLUMN(S).
- THE INTENT LOCK FOR THE TABLE IS S
- THE TIMESTAMP FOR THIS EXPLAIN IS 2004040514143724
- THE TABLE HAS 91 ROWS OF 549 BYTES.
- THE UNIQUE INDEX HAS 91 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 75
- STEP 2 ACCESSES TABLE SYSIBM .SYSCOLUMNS A
- USING INDEX SYSIBM .DSNDCX01 USING 2 COLUMN(S).
- THE JOIN METHOD IS NESTED LOOP
- THE INTENT LOCK FOR THE TABLE IS IS
- THE TIMESTAMP FOR THIS EXPLAIN IS 2004040514143724
- THE TABLE HAS 1,692 ROWS OF 981 BYTES.
- THE UNIQUE INDEX HAS 1,692 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 84
- DB2 ESTIMATES THIS QUERY WILL USE 3 MILLISECONDS OF CPU TIME
- DB2 ESTIMATES THIS QUERY WILL USE 3 SERVICE UNITS
- STATEMENT 327

- STATEMENT# 327 DECLARE SELECT-1 CURSOR FOR SELECT NAME , TBNAME , COLNO , C
- OLTYPE , LENGTH FROM SYSIBM . SYSCOLUMNS WHERE TBNAME = : H

- STEP 1 ACCESSES TABLE SYSIBM .SYSCOLUMNS
- USING INDEX SYSIBM .DSNDCX01 USING 0 COLUMN(S).
- THE ACCESS WILL USE SEQUENTIAL PREFETCH
- THE INTENT LOCK FOR THE TABLE IS IS
- THE TIMESTAMP FOR THIS EXPLAIN IS 2004040514114224
- THE TABLE HAS 1,692 ROWS OF 981 BYTES.
- THE UNIQUE INDEX HAS 1,692 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 84
- DB2 ESTIMATES THIS QUERY WILL USE 23 MILLISECONDS OF CPU TIME
- DB2 ESTIMATES THIS QUERY WILL USE 23 SERVICE UNITS
- BND206I COMPARE COMPLETE FOR PROGRAM = TEST02 VERSION = 2004-02-25-19.40.30.985067
- PREVIOUS VERSION = 2004-02-25-19.39.32.241563

Summary for TEST02

	V6	V7	V8
Before MSU's	3	4	2
After MSU's	23	43	9
Access change	NLJ No PF	NLJ No PF	NLJ No PF

Case 2 Results

- All versions stop using sequential prefetch
- Cost estimation are very different between releases
- Because the new statement is a join, the SQL doesn't match so it shows deleted and added SQL which means the new access path is shown first then the deleted

Case 3 SQL Change Change Join Criteria

- old SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
- FROM SYSIBM.SYSCOLUMNS A
- INNER JOIN
- SYSIBM.SYSTABLES B
- ON A.TBNAME = B.NAME
- AND A.TBCREATOR = B.CREATOR
- WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR
- new SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
- FROM SYSIBM.SYSCOLUMNS A
- INNER JOIN
- SYSIBM.SYSTABLES B
- ON A.TBNAME = B.NAME
- WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR

Summary for TEST03

	V6	V7	V8
Before MSU's	3	4	2
After MSU's	23	43	9
Access change	TS Scan Cartesian Join	Same	Same

Case 3 Results

- V6 V7 and V8 are the same index and then nested loop
- V6 and V7 estimates increase substantially, while V8 estimate increases somewhat
- The index is tbcreator, tbname so the match columns goes from 2 to 0 and will give a Cartesian product of all the columns with all the tables of that name

Case 4 Change Data List

- Add column from SYSTABLES (B.DBID)
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH ,B.DBID
- FROM SYSIBM.SYSCOLUMNS A INNER JOIN SYSIBM.SYSTABLES B
- ON A.TBNAME = B.NAME
- AND A.TBCREATOR = B.CREATOR
- WHERE B.NAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND B.CREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR

Case 4 Change Data List

- V6 does will not use VARCHAR columns in the index.
- V7 needs to have a DSNZPARM set (SPRMVCFK) to allow data use of VARCHAR from an Index
- Be careful if using VARCHAR access in V7 to update as result is padded with blanks and most programmers do not rewrite as VARCHAR

Summary for TEST04

	V6	V7	V8
Before MSU's	1	1	2
After MSU's	1	2	2
Access change	No change	Data access	Data access

Case 4 Results

- V6 has no change because it can not use VARCHAR
- V7 needed VCFK set to get index only on VARCHAR fields in index - V8 doesn't care
- The access changes from index only to data+index on V7 and V8
- VARCHAR index access is the default on V8, but V8 does not pad if index is defined as NOPAD
- **If programs re-write with the padded length you will enlarge the data**

Case 5 SQL Change Join to Subselect

- old SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
- FROM SYSIBM.SYSCOLUMNS A
- INNER JOIN
- SYSIBM.SYSTABLES B
- ON A.TBNAME = B.NAME
- AND A.TBCREATOR = B.CREATOR
- WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR
- new SQL
- DECLARE SELECT-1 CURSOR FOR
- SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
- FROM SYSIBM.SYSCOLUMNS A
- WHERE EXISTS
- (SELECT B.NAME
- FROM SYSIBM.SYSTABLES B
- WHERE A.TBNAME = B.NAME
- AND A.TBCREATOR = B.CREATOR)
- AND A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
- AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR

Summary for TEST05

	V6	V7	V8
Before MSU's	2	nc	nc
After MSU's	3	nc	nc
Access change	NLJ to IX only	No change	No change

Case 5 Results

- V6 index – nested loop to index only
- V7 and V8 have no change
- Change join to Subselect - this won't change the access path
- Since the old join did not retrieve data, it can be replaced with a Subselect and have the same access path and return the same data

Notes on Cases 6 through 13

- Expected path for the join would be an index read of SYSTABLES and a data read from SYSCOLUMNS
- The Optimizer must estimate from the statistics which table has the least cardinality and would be best to read first
- V8 appears to be more susceptible than V7

Notes on Cases 6 through 13

- Old DBRM goes against SYSCOLUMNS and SYSTABLES in the Catalog
- New DBRM goes against a copy of SYSCOLUMNS and SYSTABLES
- This is meant to simulate a movement from one subsystem to another

Statistic Changes

- All the examples use the same SQL with statistic changes
 - DECLARE SELECT-1 CURSOR FOR
 - SELECT A.NAME, A.TBNAME, A.COLNO, A.COLTYPE, A.LENGTH
 - FROM SYSCOLUMNS A
 - INNER JOIN
 - SYSTABLES B
 - ON A.TBNAME = B.NAME
 - AND A.TBCREATOR = B.CREATOR
 - WHERE A.TBNAME = :SYSCOL-SYSCOLUMNS.SYSCOL-TBNAME
 - AND A.TBCREATOR = :SYSCOL-SYSCOLUMNS.SYSCOL-TBCREATOR
 - and
 - DECLARE SELECT-2 CURSOR FOR
 - SELECT NAME, CREATOR, TYPE, DBNAME, TSNAME
 - FROM SYSTABLES
 - WHERE CREATOR = :SYSTAB-SYSTABLES.SYSTAB-CREATOR
- www.hlstechnologies.com

Case 6 No RUNSTATS

- All statistics show -1.
- Results
- All access SYSCOLUMNNS first
- V6 index list prefetch – merge scan - to list prefetch
- V7 & V8 index only – nested loop - to list prefetch
- DB2 chose list prefetch over the index scan chosen with stats accessed SYSCOLUMNNS first

Case 7 Tables are emptied

- RUNSTATS done on empty tables.
- Stats show 0 not -1
- RESULTS
- V6, V7 and V8 are the same
- DB2 chooses index access even though stats show table with 0 rows – Tablespace Scan was expected
- Single table select did not use list prefetch

Case 8

- Cluster ratio for SYSCOLUMNS changed to 50%
- UPDATE SYSIBM.SYSINDEXES
- SET CLUSTERRATIO = 50
- ,CLUSTERRATIOF = +0.050000000000000000E+01
- ,FIRSTKEYCARD = 300
- ,FULLKEYCARD = 300
- ,FIRSTKEYCARDF = +0.300000000000000000E+03
- ,FULLKEYCARDF = +0.300000000000000000E+03
- WHERE CREATOR = 'PUBLIC8'
- AND NAME = 'DSNDCX01'

Case 8 Results

- V6 change both Join and single table cursor
- V7 change only join
- V8 change only single table cursor
- The estimates for V8 are much less

Case 9

- Change host variable to not match the table for the join where clause
- A.TBCREATOR = :WS-PARM-VALUEC
- This is the first column in the both indexes
- RESULTS
- V6 accidental change to single table cursor
- V6 match columns = 0 index
- V7 match columns = 0 index
- V8 match columns = 2 index
- V8 can still use the index even with data type mismatch

Case 10

- Create new index and expect switch between Index 1 and Index 2
- CREATE INDEX PUBLIC10.DSNDCXX1
- ON SYSCOLUMNS (TBCREATOR ,TBNAME)
- RESULTS
- V6 change single table to TS scan and change join sequence
- V7 change the join sequence and add prefetch
- V8 only change single table cursor to TS scan
- V8 Estimates are lower than V7

Case 11

- Dropped a column from the index
- Removed TBNAME from SYSCOLUMNS index - change match columns from 2 to 1
- CREATE INDEX PUBLIC11.DSNDCX01
- ON SYSCOLUMNS (TBCREATOR)
- RESULTS
- V6 change single table to TS scan and change join sequence
- V7 change the join sequence and add prefetch
- V8 only change single table cursor to TS scan
- V8 Estimates are lower than V7

Case 12

- Drop index on SYSCOLUMNNS
- Expect Tablespace Scan
- RESULTS
- V6 changed both single table cursor and join
- V7 only changed join
- V8 changed both single table cursor and join
- Join access does TS scan on SYSCOLUMNNS first then join to SYSTABLES
- V8 estimates are lower than V7

Case 13

- Change Column Sequence in the index
- CREATE INDEX PUBLIC13.DSNDCX01
- ON SYSCOLUMNS (TBCREATOR ,NAME ,TBNAME)
- Access to SYSCOLUMNS now match columns = 1
- RESULTS
- V6 change both single table cursor and join
- V7 only change join
- V8 change both single table cursor and join
- V6 changed from match columns 2 to 1 because column name was moved in front of tname in the index - DB2 still does non-matching scan using the index data but not part of match columns.
- V7 and V8 changed the join sequence to scan SYSCOLUMNS first then join to SYSTABLES

Conclusions

- DB2 optimizer is becoming more dependent on statistics
- What statistics to collect becomes more important
- The default was `RUNSTATS TABLESPACE dbname.tsname
TABLE INDEX`
- V6 forward will most likely produce more Access Path Changes than V7 to V8
- Some Access Paths will change between V7 and V8
- The V8 Optimizer comes up with lower estimates than V7 or V6

Reorg the tables – V8

- REORG TABLESPACE TEST16.TABLESP1 REUSE LOG NO SORTDATA SORTKEYS SHRLEVEL NONE Because the initial tables were built from unload and not sorted
- INDEX HAS 6,361 ENTRIES IN 2 LEVELS WITH CLUSTER RATIO 81
- REORG and rerun RUNSTATS
- RESULTS
- No changes

REORG the tables – V8 and change RUNSTATS request

- REORG TABLESPACE TEST16.TABLESP1 REUSE LOG NO
SORTDATA SORTKEYS SHRLEVEL NONE
- RUNSTATS TABLESPACE TEST17.TABLESP1
TABLE (ALL) INDEX (ALL KEYCARD FREQVAL NUMCOLS 2
COUNT 10) REPORT YES
- INDEX HAS 6,361 ENTRIES IN 2 LEVELS WITH
CLUSTER RATIO 100
- REORG and rerun RUNSTATS for SYSCOLUMNMS
- RESULTS
- Join always used SYSTABLES first and then SYSCOLUMNMS

RUNSTATS case Results

- The index DSNDCX01 has 3 columns
- Default RUNSTATS TABLESPACE only collects first key cardinality and full key cardinality
- The JOIN specified only 2 of the columns in the index
- Because DB2 didn't have cardinality for the 2 column combination, it tried to estimate the value from the cardinality of each column
- This value was sufficiently wrong that the wrong table was chosen as the first table of the join
- INDEX (ALL KEYCARD FREQVAL NUMCOLS 2 COUNT 10) collects cardinality for the 2 columns in combination
- FREQVAL is all that is required for host variable access

New in V8

- Index matching with 'WRONG' datatypes without using functions to change datatype
- Will use the index for matching access and JOIN's
- Does have a CPU cost at run time
- No indication for PLAN_TABLE or access path

New in V8

- CREATE TABLE ____ VOLATILE
- Specifies that index access should be used on this table whenever possible for SQL operations. However, be aware that list prefetch and certain other optimization techniques are disabled when VOLATILE is used.
- Like OPTIMIZE for 1 ROWS without SQL changes

New in V8

- Unpadded indexes
- HOWEVER – existing indexes will stay padded until ALTER INDEX PADDED
- When an index with at least one varying-length column is changed from PADDED to NOT PADDED, or vice versa, the index is placed in restricted REBUILD-pending status (RBDP). The index cannot be accessed until it is rebuilt from the table

New in V8

- READ INDEX backward
- V7 could SELECT MAX from ascending index
- V8 can use index forwards or backwards but must match all columns ascending and descending

Gerald Hodge

HLS Technologies

ghodge@hlstechnologies.com

WWW.HLSTECHNOLOGIES.COM

Has copies of JCL and DBRM's to recreate these
test cases and the longer version of this
presentation

www.hlstechnologies.com