A Look Under the Hood of CBO
The 10053 Event

Wolfgang Breitling
breitliw@centrexcc.com

Agenda

- Overview
- Trace contents
- Table, Index and Column Statistics
- Cost Calculations
  - Single Table Access Costs
  - Join Costs

Event 10053

Event 10053 details the choices made by the CBO in evaluating the execution path for a query

Event 10053 externalizes the information that the optimizer uses in generating a plan for a query

Setting Event 10053

for your own session

on:

```
alter session set events
'10053 trace name context forever[, level {1|2}]'
```

off:

```
alter session set events
'10053 trace name context off'
```

Setting Event 10053

for another session

on:

```
sys.dbms_system.set_ev
(<sid>, <serial#>, 10053, {1|2}, '')
```

off:

```
sys.dbms_system.set_ev
(<sid>, <serial#>, 10053, 0, '')
```

Trace Generation

When the statement is parsed by the CBO

1. the statement is parsed

   and

2. the statement is parsed by the CBO

Trace Contents

- Query
- Parameters used by the optimizer
- Base Statistical Information
- Base Table Access Cost
- Join Order and Method Computations
- Recosting for special features

Query

```
QUERY

select dname, ename from emp, dept

where emp.deptno = dept.deptno

and ename = :b1
```

Parameters used by the Optimizer

OPTIMIZER_FEATURES_ENABLE

OPTIMIZER_INDEX_CACHING

OPTIMIZER_INDEX_COST_ADJ

OPTIMIZER_PERCENT_PARALLEL

OPTIMIZER_DYNAMIC_SAMPLING = 1

HASH_MULTIBLOCK_IO_COUNT

DB_FILE_MULTIBLOCK_READ_COUNT

HASH_AREA_SIZE

SORT_AREA_SIZE

Base Statistical Information

```
BASE STATISTICAL INFORMATION
**********************
Table stats    Table: EMP   Alias: EMP
  TOTAL ::  CDN: 72130  NBLKS:  900  AVG_ROW_LEN:  42
Column:     DEPTNO  Col#: 8      Table: EMP   Alias: EMP
    NDV: 12        NULLS: 0         DENS: 3.1935e-05
    FREQUENCY HISTOGRAM: #BKT: 339 #VAL: 12
-- Index stats
  INDEX NAME: EMP_1  COL#: 1
    TOTAL ::  LVLS: 1   #LB: 283  #DK: 73227  LB/K: 1  DB/K: 1  CLUF: 5392
  INDEX NAME: EMP_2  COL#: 2
    TOTAL ::  LVLS: 2   #LB: 588  #DK: 42  LB/K: 14  DB/K: 380  CLUF: 15978
  INDEX NAME: EMP_3  COL#: 8
    TOTAL ::  LVLS: 2   #LB: 483  #DK: 12  LB/K: 40  DB/K: 389  CLUF: 4673
```

Table Statistics

trace:

```
Table stats    Table: EMP   Alias: EMP
  TOTAL ::  CDN: 72130  NBLKS:  903  AVG_ROW_LEN:  39
```

dba_tables:

```
NUM_ROWS                       : 72130
BLOCKS                         : 903
EMPTY_BLOCKS                   : 0
AVG_SPACE                      : 0
AVG_ROW_LEN                    : 39
```

Index Statistics

trace:

```
-- Index stats
  INDEX NAME: EMP_1  COL#: 1
    TOTAL ::  LVLS: 1   #LB: 308  #DK: 72130  LB/K: 1  DB/K: 1  CLUF: 4922
  INDEX NAME: EMP_2  COL#: 2
    TOTAL ::  LVLS: 2   #LB: 352  #DK: 42  LB/K: 8  DB/K: 378  CLUF: 15883
  INDEX NAME: EMP_3  COL#: 8
    TOTAL ::  LVLS: 2   #LB: 290  #DK: 12  LB/K: 24  DB/K: 367  CLUF: 4407
```

dba_indexes:

```
BLEVEL                          : 2
LEAF_BLOCKS                     : 352
DISTINCT_KEYS                   : 42
AVG_LEAF_BLOCKS_PER_KEY         : 8
AVG_DATA_BLOCKS_PER_KEY         : 378
CLUSTERING_FACTOR               : 15883
```

Column Statistics

trace:

```
Column:       ENAME  Col#: 2      Table: EMP   Alias: EMP
    NDV: 42          NULLS: 0          DENS: 2.3810e-02
    NO HISTOGRAM: #BKT: 1 #VAL: 2
```

dba_tab_columns:

```
NUM_DISTINCT                    : 42
LOW_VALUE                       : 4144414D53
HIGH_VALUE                      : 77617264
DENSITY                         : .023809523809523 8
NUM_NULLS                       : 0
NUM_BUCKETS                     : 1
```

Single Table Access Path

```
SINGLE TABLE ACCESS PATH
TABLE: EMP     ORIG CDN: 72130  ROUNDED CDN: 1717  CMPTD CDN: 1717
  Access path: tsc  Resc:  88  Resp:  88
  Access path: index (equal)
      Index: EMP_2
  TABLE: EMP
      RSC_CPU: 0   RSC_IO: 397
  IX_SEL:  0.0000e+00  TB_SEL:  2.3810e-02
  Access path: index (equal)
      Index: EMP_2
  TABLE: EMP
      RSC_CPU: 0   RSC_IO: 16
  IX_SEL:  2.3810e-02  TB_SEL:  2.3810e-02
  Access path: index (no sta/stp keys)
      Index: EMP_3
  TABLE: EMP
      RSC_CPU: 0   RSC_IO: 485
  IX_SEL:  1.0000e+00  TB_SEL:  1.0000e+00
```

Cardinality Estimate

```
TABLE: EMP
     ORIG CDN:       72130
     ROUNDED CDN:     1717
     CMPTD CDN:       1717


TB_SEL:  2.3810e-02
Column: ENAME … NDV: 42 … DENS: 2.3810e-02

          ORIG CDN * TB_SEL = CMPTD CDN
          72130 * 2.3810e-02 = 1717.415
```

Base Access Plans

2 Table Scan

3 Index Unique

4 Index Range

5 Index And-Equal

23 index fast full scan

Cardinality Estimate

Column: ENAME Col#: 2 Table: EMP Alias: EMP
NDV: 14 NULLS: 0 DENS: 1.6667e-001

TABLE: EMP ORIG CDN: 855
CMPTD CDN: 143

BEST_CST: 1.00 PATH: 2 Degree: 1

$$142.5 = 855 * 1.6667e^{-001}$$

CMPTD CDN = ORIG CDN * FF

Table Scan Cost

Table stats Table: ■■■ Alias: D
TOTAL :: CDN: 115630 NBLKS: 4339
SCAN_CST: 265

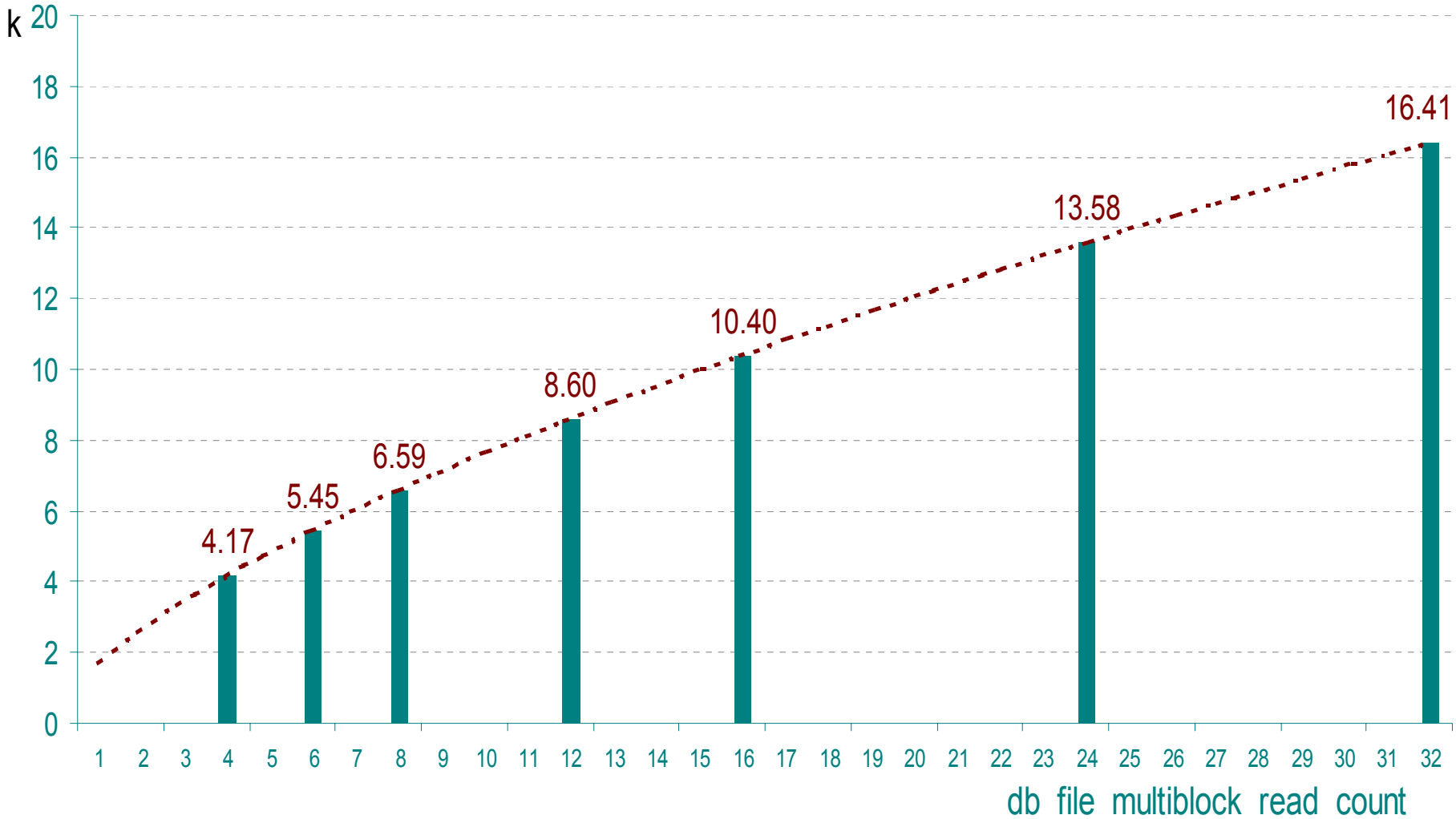
4339 / 265 = 16.373

Table stats Table: ■■■ Alias: A
TOTAL :: CDN: 454503 NBLKS: 8975
SCAN_CST: 548

8975 / 548 = 16.377

SCAN_CST = NBLKS / k

Table Scan Cost and multi_block_read_count



Predicates and Filter Factors
without Bind Variables

Predicate	filter factor
c1 = value	1/c1.num_distinct
c1 like value	1/c1.num_distinct
c1 > value	(Hi - value) / (Hi - Lo)
c1 < value	(value - Lo) / (Hi - Lo)

Predicates and Filter Factors without Bind Variables

Predicate	filter factor
c1 >= value	(Hi - value) / (Hi - Lo) + c1.num_distinct
c1 <= value	(value - Lo) / (Hi - Lo) + c1.num_distinct
c1 between	(upper - lower) / (Hi - Lo) + 2* c1.num_distinct or the selectivity of c1 <= upper if that is smaller

Predicates and Filter Factors with Bind Variables

Predicate	filter factor
c1 = :b1	c1.density
c1 like :b1	{ $5.0e^{-02}$ | c1.density }
c1 { > | >= | < | <= } :b1	$5.0e^{-02}$
c1 between :b1 and :b2	$2.5e^{-03}$ ($5.0e^{-02} * 5.0e^{-02}$)

Predicates and Filter Factors
Combining Predicates

Predicate	filter factor
P1 AND P2	FF1 * FF2
P1 OR P2	FF1 + FF2 - FF1 * FF2
NOT P1	1 - FF1

Column Statistics and Histograms

- Value Based Histogram

  # buckets = NDV

- Height Based Histogram

  # buckets < NDV

Value Based Histogram

- Predicate matches one of the values in the histogram:

	<	<=	=
selectivity =	EP of prior row /num_rows	EP of matching row /num_rows	difference /num_rows

Example:

table	column	EP	value
PSTREESELECT06	SELECTOR_NUM	10001	651
PSTREESELECT06	SELECTOR_NUM	10242	664

selector_num < 664 selectivity = 10001 / num_rows
selector_num <= 664 selectivity = 10242 / num_rows
selector_num = 664 selectivity = (10242-10001) / num_rows

Value Based Histogram

- Predicate does not match one of the values in the histogram
  Since this is a value base histogram that means there are no rows in the table with that value for the column and therefore the selectivity should be 0. However, the optimizer uses the density from the column statistics as selectivity.

- Bind Variable predicate:
  The selectivity is taken as 1/num_distinct, effectively ignoring the histogram.

Height Based Histogram

- the

Index Access Costs

Unique scan	blevel + 1
Fast full scan	leaf_blocks / k
Index-only	blevel + FF*leaf_blocks
Range scan	blevel + FF*leaf_blocks + FF*clustering_factor

Index Access Costs

INDEX#	Col#	LVLS	#LB	#DK	CLUF
8417	27, 1	1	13100	66500	1469200
8418	1, 12, 7	2	19000	74700	1176500
8419	3, 1, 4, 2	2	31000	49700	118000
15755	1, 12, 8	1	12600	18800	1890275

Col#: 1 NDV: 10 DENS: 1.0000e-001
Col#: 12 NDV: 8 DENS: 1.2500e-001
Col#: 8 NDV: 33 DENS: 3.0303e-001

$$\begin{aligned} & & 2 & & 2 \\ & + 19000 * 1.0000e^{-1} * 1.2500e^{-1} & & & 237.5 \\ & + 1176500 * 1.0000e^{-1} * 1.2500e^{-1} & & & \underline{14706.25} \\ & & & & 14945.75 \end{aligned}$$

Access path: index (scan) INDEX#: 8418 CST: 14947

Index Access Costs

INDEX#	Col#	LVLS	#LB	#DK	CLUF
~~8417~~	~~27, 1~~	~~1~~	~~13100~~	~~66500~~	~~1469200~~
8418	1, 12, 7	2	19000	74700	1176500
~~8419~~	~~3, 1, 4, 2~~	~~2~~	~~31000~~	~~49700~~	~~118000~~
15755	1, 12, 8	1	12600	18800	1890275

Col#: 1 NDV: 10 DENS: 1.0000e-001
Col#: 12 NDV: 8 DENS: 1.2500e-001
Col#: 8 NDV: 33 DENS: 3.0303e-001

$$\begin{array}{rr} & 1 \quad 1 \\ + 12600 * 1.0000e^{-1} * 1.2500e^{-1} * 3.0303e^{-1} & 47.73 \\ + 1890275 * 1.0000e^{-1} * 1.2500e^{-1} * 3.0303e^{-1} & \underline{7160.13} \\ & 7208.86 \end{array}$$

Access path: index (equal) INDEX#: 15755 CST: 7209

Default Index Statistics

INDEX#: 23574 COL#: 1
TOTAL :: LVLS: 1 #LB: 25 #DK: 100 LB/K: 1 DB/K: 1
CLUF: 800

INDEX#: 23575 COL#: 2
TOTAL :: LVLS: 1 #LB: 25 #DK: 100 LB/K: 1 DB/K: 1
CLUF: 800

INDEX#: 23576 COL#: 8
TOTAL :: LVLS: 1 #LB: 25 #DK: 100 LB/K: 1 DB/K: 1
CLUF: 800

Default Table Statistics

Table stats Table: EMP Alias: EMP

TOTAL :: (NOT ANALYZED) CDN: 2240 NBLKS: 55
SCAN_CST: 4 AVG_ROW_LEN: 100

Table stats Table: EMP Alias: EMP

TOTAL :: CDN: 4457 NBLKS: 55
SCAN_CST: 4 AVG_ROW_LEN: 36

CDN = NBLKS * (db_block_size – 24) / 100

Default Column Statistics

Column: ENAME Col#: 2 Table: EMP Alias: E
NO STATISTICS (using defaults)
NDV: 70 NULLS: 0 DENS: 1.4286e-002

Column: HIREDATE Col#: 5 Table: EMP Alias: E
NO STATISTICS (using defaults)
NDV: 70 NULLS: 0 DENS: 1.4286e-002

DENS = NBLKS * m

Join Costs

1. **NL Join**

   join cost = cost of accessing outer table
   + (cardinality of outer table * cost of accessing inner table)

2. **SM Join**

   join cost = (cost of accessing outer table + outer sort cost)
   + (cost of accessing inner table + inner sort cost)

3. **HA Join**

   join cost = (cost of accessing outer table)
   + (cost of building hash table)
   + (cost of accessing inner table)

NL Join

join cost = cost of outer table access

+ (cardinality of outer table * cost of inner table access)

Outer table: cost: 1 cdn: 4 rcz: 11 resp: 1

Inner table: EMP

Access path: tsc Resc: 4

Join resc: 17 Resp: 17 *[17 = 1 + 4 * 4]*

Join Cardinality

Join cardinality: 36 = outer (4) * inner (107) *
sel (8.3333e-002) [flag=0]

join selectivity = 1/max[NDV(t1.c1), NDV(t2.c2)]
* [(card t1 - # t1.c1 NULLs) / card t1]
* [(card t2 - # t2.c2 NULLs) / card t2]

SM Join

join cost = (cost of accessing outer table + outer sort cost)
+ (cost of accessing inner table + inner sort cost)

Outer table:

resc: 1 cdn: 4 rcz: 11 deg: 1 resp: 1

Inner table: EMP

resc: 4 cdn: 107 rcz: 13 deg: 1 resp: 4

SM Join

SORT resource Sort statistics
Sort width: 3 Area size: 43008 Degree: 1
Blocks to Sort: 1 Row size: 23 Rows: 4
Initial runs: 1 Merge passes: 1 Cost / pass: 2
Total sort cost: 2

SORT resource Sort statistics
Sort width: 3 Area size: 43008 Degree: 1
Blocks to Sort: 1 Row size: 25 Rows: 107
Initial runs: 1 Merge passes: 1 Cost / pass: 2
Total sort cost: 2

Merge join Cost: 8 Resp: 8 *[(1 + 2) + (4 + 2)]*

HA Join

join cost = (cost of outer table access)
+ (cost of building hash table) + (cost of inner table access)

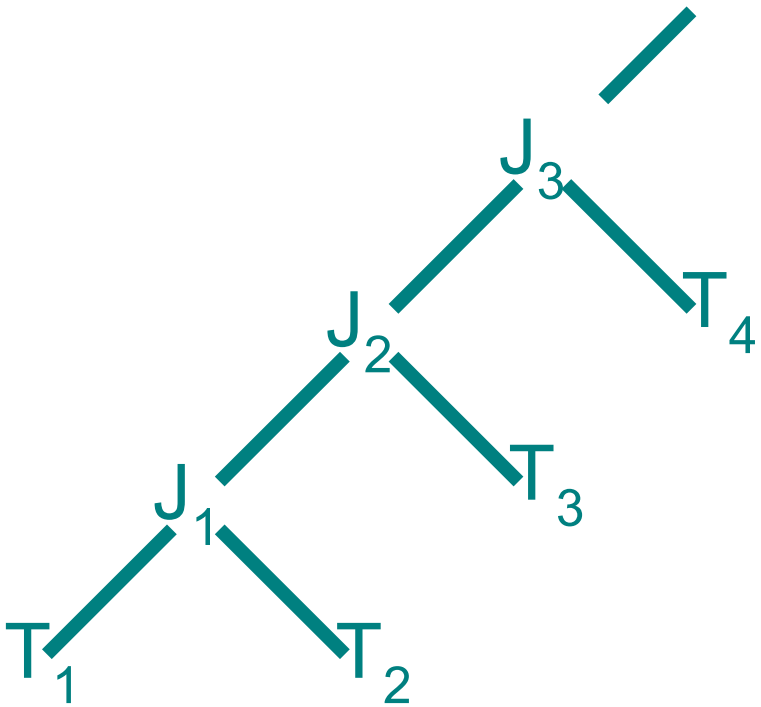
Outer table: resc: 1 cdn: 4 rcz: 11 deg: 1 resp: 1

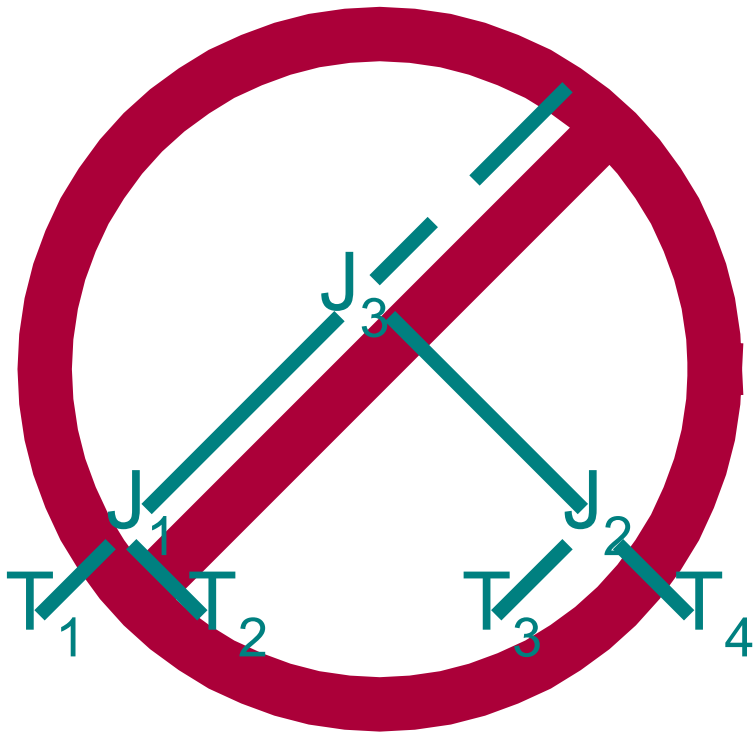
Inner table: EMP
resc: 4 cdn: 107 rcz: 13 deg: 1 resp: 4

Hash join one ptn: 1 Deg: 1
hash_area: 32 buildfrag: 33 probefrag: 1 ppasses: 2

Hash join Resc: 6 Resp: 6 *[1 + 4 + 1]*

Multi-table Joins





Multi-table Joins

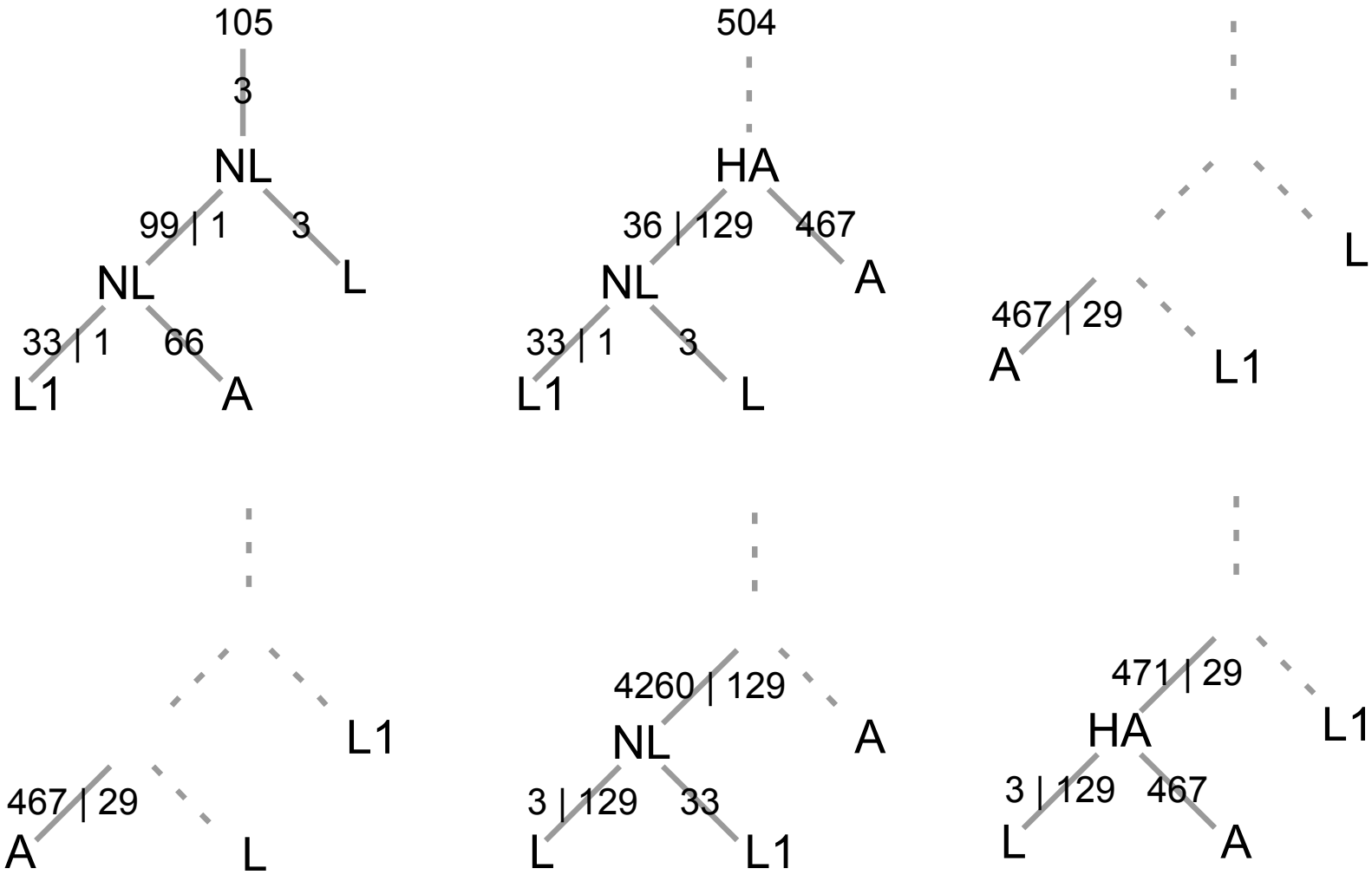
SINGLE TABLE ACCESS PATH

TABLE: A ORIG CDN: 683620 CMPTD CDN: 29
BEST_CST: 467.00 PATH: 4 Degree: 1

TABLE : L1 ORIG CDN: 125263 CMPTD CDN: 1
BEST_CST: 33.00 PATH: 2 Degree: 1

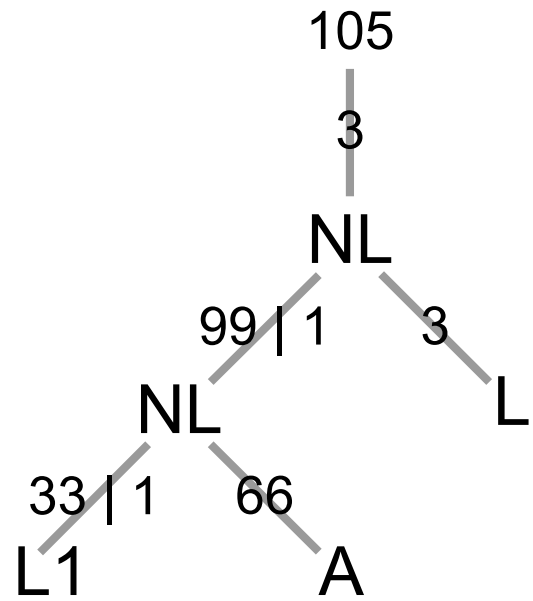
TABLE : L ORIG CDN: 238504 CMPTD CDN: 129
BEST_CST: 3.00 PATH: 4 Degree: 1

Multi-table Joins



Multi-table Joins

TABLE: A CMPTD CDN: 29
TABLE: L1 CMPTD CDN: 1
TABLE: L CMPTD CDN: 129




```
   cost      card operation
    105         1 SELECT STATEMENT
    105         1   SORT GROUP BY
    102         1     NESTED LOOPS
     99         1       NESTED LOOPS
     33         1         TABLE ACCESS FULL L1
     66        29         TABLE ACCESS BY LOCAL INDEX ROWID A:6-6
      2        29           INDEX RANGE SCAN A_ACC:6-6
      3       129       INDEX RANGE SCAN L
```

Metalink Notes

40656.1	Supposedly a note about event 10053. Not externally available (yet).
75713.1	Important Customer Information about numeric EVENTS
35934.1	Cost Based Optimizer - Common Misconceptions and Issues
66030.1	Relationship between optimizer_max_permutations and optimizer_search_limit
32895.1	SQL Parsing Flow Diagram
68992.1	Predicate Selectivity
104817.1	Discussion on Oracle Joins - Costs - Algorithms & Hints
67522.1	Why is my index not used?

More Metalink Notes

62364.1	Hints and Subqueries
46234.1	Interpreting Explain plan
33089.1	Troubleshooting Guide: SQL Tuning
1031826.6	Histograms: An Overview
72539.1	Interpreting Histogram Information
77228.1	How to Tell if a Table has been analyzed
70075.1	Use of bind variables in queries
31412.1	Select to show Optimizer Statistics for CBO
43214.1	Autotrace Option in 7.3

Resources

Oracle University - Course ID: 65340

Oracle8i: Everything You Always Wanted to Know about the Optimizer

asktom.oracle.com	(Thomas Kyte)
www.ixora.com.au	(Steve Adams)
www.hotsos.com	(Cary Millsap)
www.orapub.com	(Craig Shallahamer)
www.jlcomp.demon.co.uk	(Jonathan Lewis)
www.oraperf.com	(Anjo Kolk)
www.evdbt.com	(Tim Gorman)

Wolfgang Breitling

Centrex Consulting Corporation

www.centrexcc.com

Which Plan is better?

a)

cost	card	operation
2,979	446	SELECT STATEMENT
2,979	446	SORT ORDER BY
		FILTER
2,955	446	HASH JOIN
10	13,679	TABLE ACCESS FULL E
2,901	49,755	HASH JOIN
737	8,629	HASH JOIN
5	45	HASH JOIN
3	6	TABLE ACCESS FULL A
1	15	TABLE ACCESS FULL D
731	316,380	TABLE ACCESS FULL B
1,953	239,142	TABLE ACCESS FULL C

b)

cost	card	operation
792	1	SELECT STATEMENT
792	1	SORT ORDER BY
		FILTER
790	1	HASH JOIN
760	83	HASH JOIN
758	11	NESTED LOOPS
749	1	HASH JOIN
3	6	TABLE ACCESS FULL A
731	28,762	TABLE ACCESS FULL B
9	239,142	TABLE ACCESS BY INDEX ROWID C
4	239,142	INDEX RANGE SCAN C_IX0
1	15	TABLE ACCESS FULL D
10	13,679	TABLE ACCESS FULL E

Analysis of the Explain Plan

cost	card	operation
792	1	SELECT STATEMENT
792	1	SORT ORDER BY
		FILTER
790	1	HASH JOIN
760	83	HASH JOIN
758	11	NESTED LOOPS
749	1	HASH JOIN
3	6	TABLE ACCESS FULL A
731	28,762	TABLE ACCESS FULL B
9	239,142	TABLE ACCESS BY INDEX ROWID C
4	239,142	INDEX RANGE SCAN C_IX0
1	15	TABLE ACCESS FULL D
10	13,679	TABLE ACCESS FULL E

Quiz

- When is a statement parsed by the rule based optimizer rather than the cost based optimizer?
- When is a statement parsed by the cost based optimizer rather than the rule based optimizer?
- How do you guarantee that a SQL statement gets parsed in order to generate a 10053 trace but avoid that it actually gets executed?

Value Based Histogram

table	column	EP	value
PSTREESELECT06	SELECTOR_NUM	9278	218
PSTREESELECT06	SELECTOR_NUM	9519	588
PSTREESELECT06	SELECTOR_NUM	9760	638
PSTREESELECT06	SELECTOR_NUM	10001	651
PSTREESELECT06	SELECTOR_NUM	10242	664
PSTREESELECT06	SELECTOR_NUM	18164	715
PSTREESELECT06	SELECTOR_NUM	26086	720
PSTREESELECT06	SELECTOR_NUM	26338	1508
PSTREESELECT06	SELECTOR_NUM	26590	1696
PSTREESELECT06	SELECTOR_NUM	26854	2564
...			
PSTREESELECT06	SELECTOR_NUM	87490	10088
PSTREESELECT06	SELECTOR_NUM	97545	10112
PSTREESELECT06	SELECTOR_NUM	107600	10114
PSTREESELECT06	SELECTOR_NUM	117655	10115
PSTREESELECT06	SELECTOR_NUM	127710	10116
PSTREESELECT06	SELECTOR_NUM	143163	10127

Height Based Histogram

table	column	EP	value
PSTREESELECT06	SELECTOR_NUM	2	218
PSTREESELECT06	SELECTOR_NUM	4	715
PSTREESELECT06	SELECTOR_NUM	5	720
PSTREESELECT06	SELECTOR_NUM	6	2564
PSTREESELECT06	SELECTOR_NUM	8	3449
PSTREESELECT06	SELECTOR_NUM	10	4235
PSTREESELECT06	SELECTOR_NUM	11	6904
PSTREESELECT06	SELECTOR_NUM	13	7294
PSTREESELECT06	SELECTOR_NUM	15	8206
PSTREESELECT06	SELECTOR_NUM	17	8274
PSTREESELECT06	SELECTOR_NUM	18	9343
PSTREESELECT06	SELECTOR_NUM	19	9879
PSTREESELECT06	SELECTOR_NUM	21	10112
PSTREESELECT06	SELECTOR_NUM	24	10114
PSTREESELECT06	SELECTOR_NUM	26	10115
PSTREESELECT06	SELECTOR_NUM	28	10116
PSTREESELECT06	SELECTOR_NUM	32	10127

Wolfgang Breitling

Centrex Consulting Corporation

www.centrexcc.com