Fallacies of the Cost Based Optimizer

Wolfgang Breitling
breitliw@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
```

Cost vs. Performance

Correlation between cost and performance?

Why not ?

Assumptions

- Uniform Distribution Assumption
  - Uniform Distribution over Blocks
  - Uniform Distribution over Rows
  - Uniform Distribution over Range of Values
- Predicate Independence Assumption
- Join Uniformity Assumption

Selectivity and Cardinality

$$\text{Selectivity} = FF = card_{est} \,/\, card_{base}$$

$$card_{est} = FF * card_{base}$$

The Makeup of Plan Costs

- The base table access cost is dependent on estimated # of blocks accessed which is - directly or indirectly - a function of the estimated row cardinality:
  - Table scan — nblks / k
  - 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

The Makeup of Plan Costs

Join cost is dependent on cardinality of row sources

- Nested Loop $\$_{outer} + card_{outer} * \$_{inner}$
- Sort-Merge $\$_{outer} + \$sort_{outer} + \$_{inner} + \$sort_{inner}$
- Hash $\$_{outer} + \$_{inner} + \$_{hash}$

Plan Costs Recap

Estimated cardinality = selectivity * base cardinality

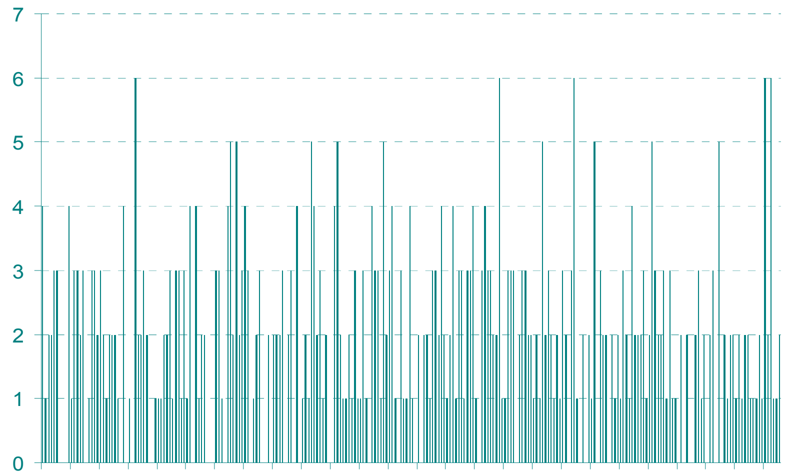
The cost of an access plan is a function of the estimated cardinalities of its components.

Incorrect estimates lead to incorrect plan component costs and sub-optimal or wrong access plans.

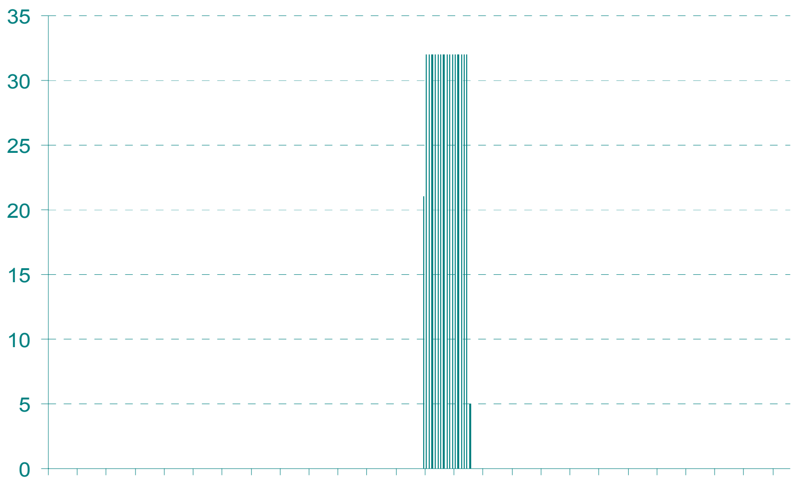
This is why accurate cardinality estimates are so important.

Distribution over blocks

uniform



clustered



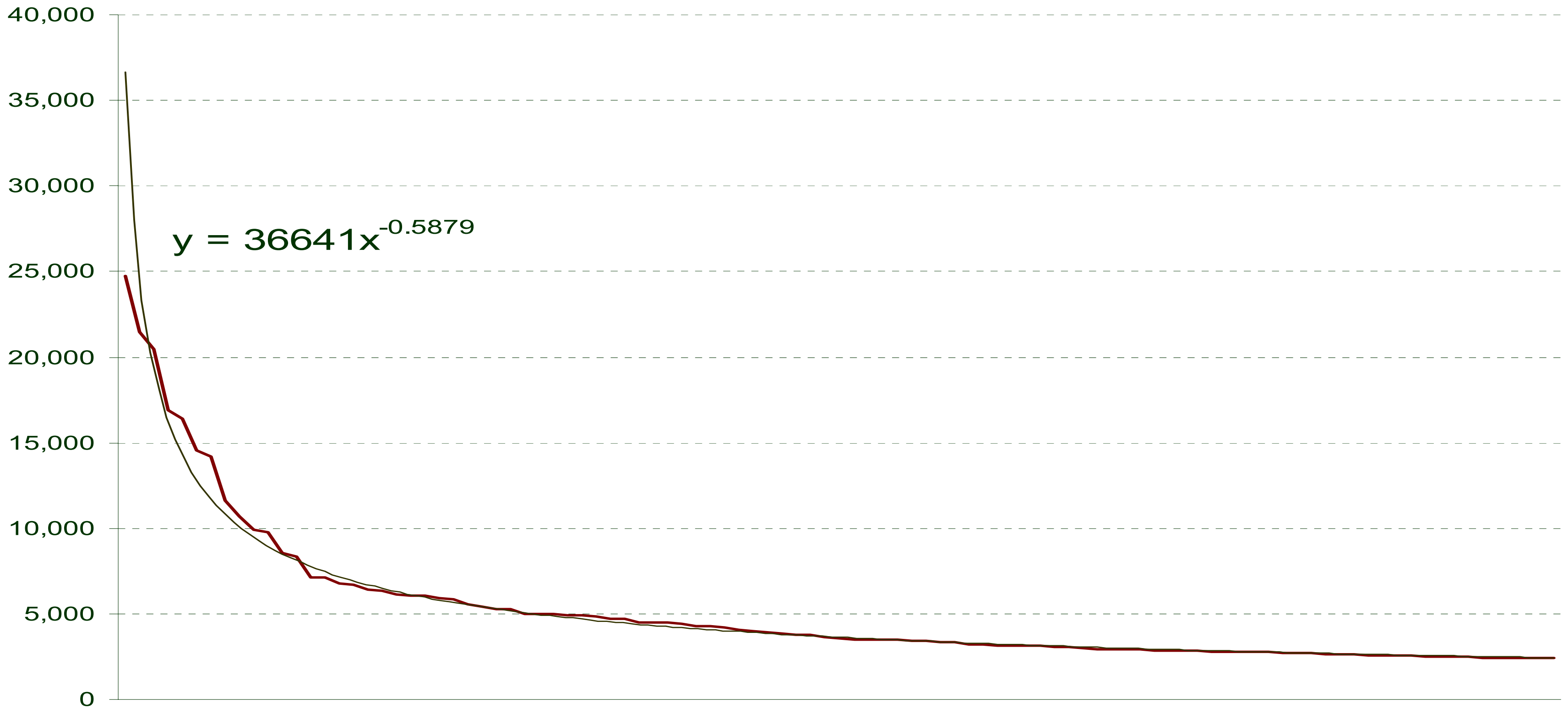
Distribution of Value Frequencies

“for an equality predicate (last_name = 'Smith') the selectivity is set to the reciprocal of the number of distinct values of last_name, because the query selects rows that all contain one out of N distinct values.“*

* Oracle 9*i* Performance Tuning Guide and Reference

Distribution of Value Frequencies

Power distribution



Distribution of Value Frequencies

```
column                NDV      density
-------------- ---------- ------------
EMPLID             10,000   1.0000E-04
...
COMPANY               200   5.0000E-03
```

```
Select emplid, jobcode, salary
from ps_job5 b where b.company = 'B01'
```

<u>explain plan</u>

```
card operation
---- ----------------------------------

  50 SELECT STATEMENT
  50   TABLE ACCESS BY INDEX ROWID PS_JOB5
  50     INDEX RANGE SCAN PSBJOB5
```

<u>execution plan</u>

```
  card operation
------ -------------------------------
   530 SELECT STATEMENT
   530  TABLE ACCESS BY INDEX ROWID PS_JOB5
   531    INDEX   GOAL: ANALYZED (RANGE SCAN) OF 'PSBJOB5' (NON-UNIQUE)
```

```
call       count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.47       0.47         21        359          5           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       37      0.48       0.47        420        567          0         530
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       39      0.95       0.94        441        926          5         530
```

Distribution of Value Frequencies

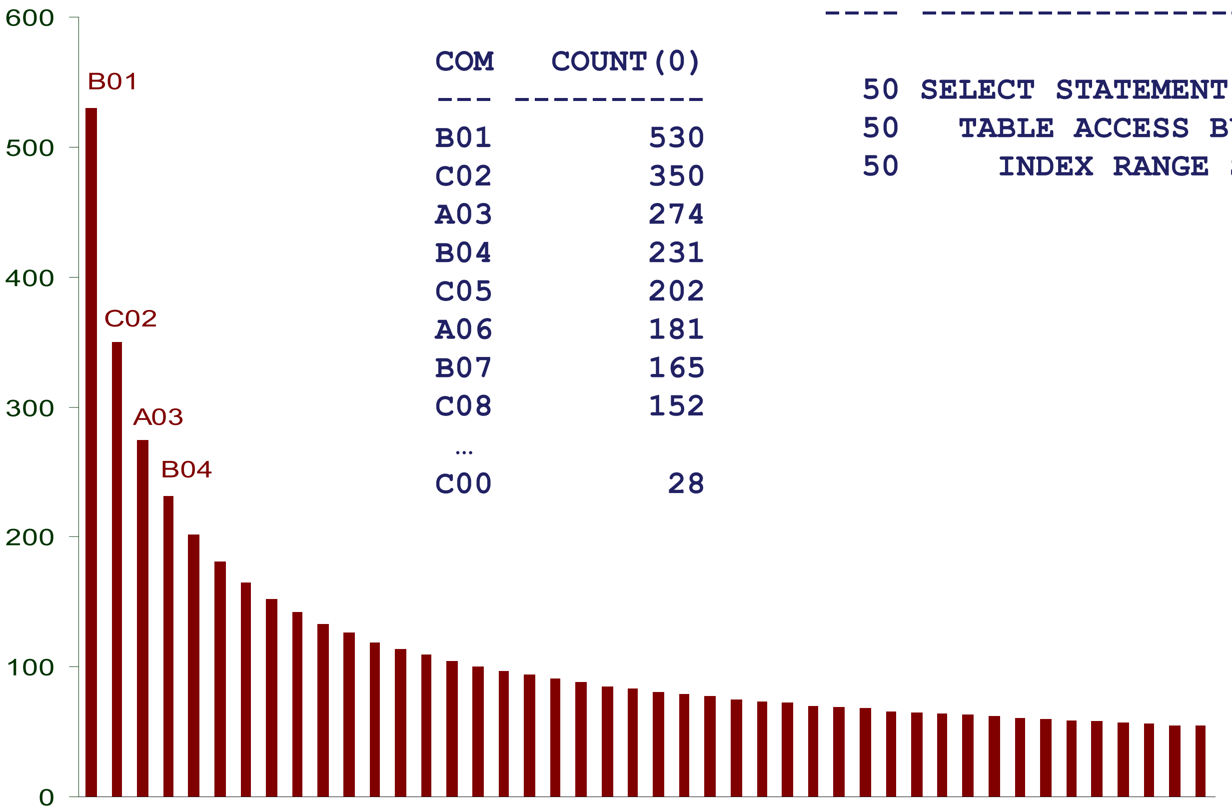
```
column                NDV       density
-------------- ---------- ------------
EMPLID             10,000   1.0000E-04
...
COMPANY               200   5.0000E-03
```

```
Select emplid, jobcode, salary
from ps_job5 b where b.company = 'B01'
```

explain plan

```
card operation
---- ---------------------------------

  50 SELECT STATEMENT
  50   TABLE ACCESS BY INDEX ROWID PS_JOB5
  50     INDEX RANGE SCAN PSBJOB5
```




```
COM   COUNT(0)
--- ----------
B01        530
C02        350
A03        274
B04        231
C05        202
A06        181
B07        165
C08        152
 …
C00         28
```

Distribution of Value Frequencies

With Histogram on company

```
Analyze table ps_job5 compute statistics for columns company [ size 75 ];
```

```
column                 NDV      density
-------------- ---------- ------------
EMPLID              10,000   1.0000E-04
...
COMPANY                200   6.0644E-03
```

```
Select emplid, jobcode, salary
from ps_job5 b where b.company = 'B01'
```

explain plan

```
card operation
---- ------------------------------------
 534 SELECT STATEMENT
 534   TABLE ACCESS FULL PS_JOB5
```

execution plan

```
  card operation
------ ------------------------------
   530  SELECT STATEMENT GOAL: CHOOSE
   530   TABLE ACCESS GOAL: ANALYZED (FULL) OF 'PS_JOB5'
```

```
call       count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.17       0.15         25        424          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       37      0.24       0.22        912        943         15         530
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       39      0.41       0.37        937       1367         15         530
```

Distribution of Value Frequencies

With Histogram and bind Variable on company

```
column                 NDV      density
-------------- ---------- ------------
EMPLID              10,000   1.0000E-04
...
COMPANY                200   6.0644E-03
```

```
Select emplid, jobcode, salary
from ps_job5 b where b.company = :b1
```

explain plan

```
card operation
---- ----------------------------------
  61 SELECT STATEMENT
  61   TABLE ACCESS BY INDEX ROWID PS_JOB5
  61     INDEX RANGE SCAN PSBJOB5
```

10,000 * 6.0644^{e-3} = 60.644 rounded up to 61.

Distribution of Value Frequencies

With Histogram and bind Variable on company

```
Analyze table ps_job5 compute statistics for columns company size 10;
```

```
column                 NDV      density
-------------- ---------- ------------
EMPLID              10,000   1.0000E-04
...
COMPANY                200   1.0870E-02
```

```
Select emplid, jobcode, salary
from ps_job5 b where b.company = :b1
```

<u>explain plan</u>

```
card operation
---- ------------------------------
 109 SELECT STATEMENT
 109   TABLE ACCESS FULL PS_JOB5
```

Column Statistics and Histograms

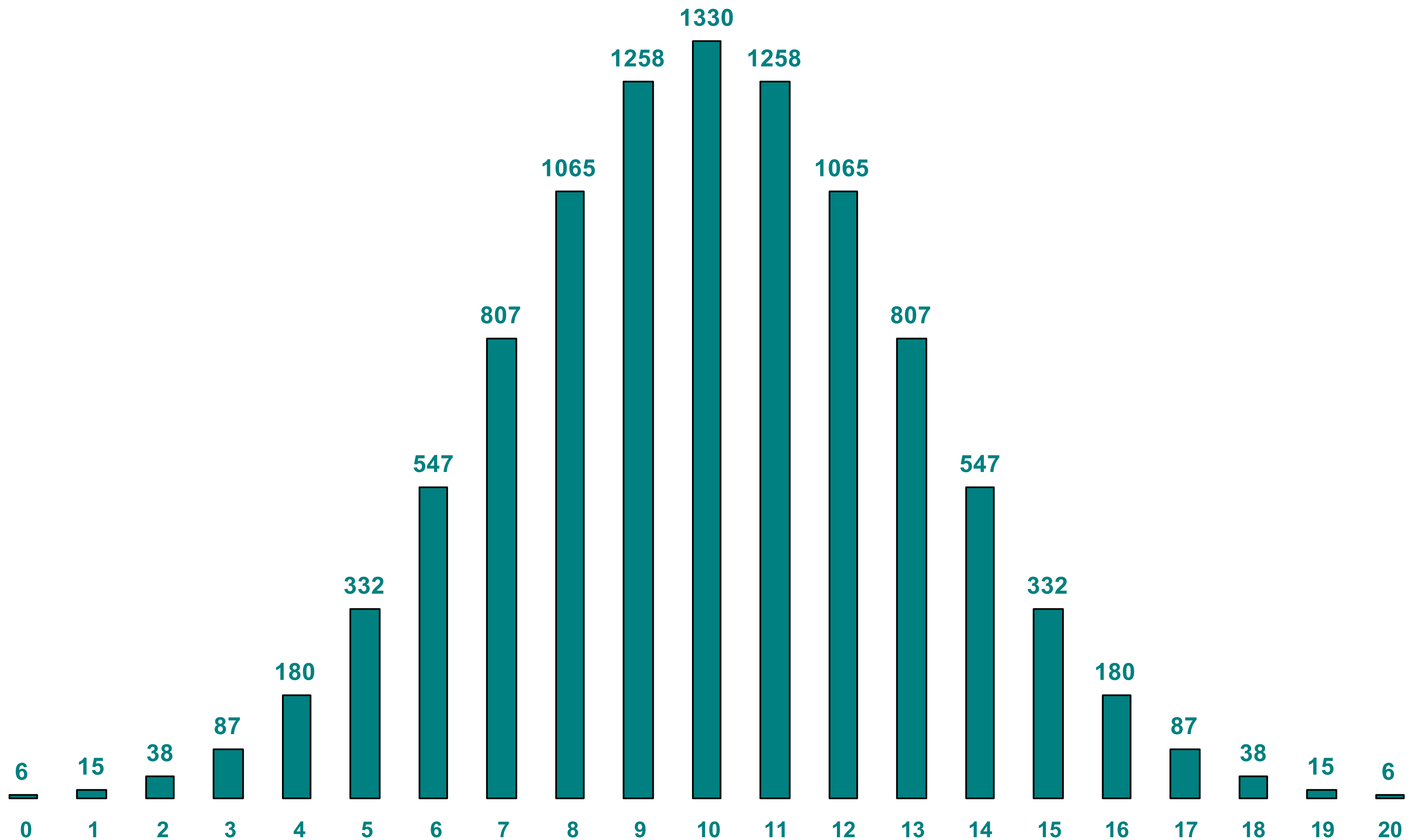
- Frequency Histogram

  # buckets = NDV

- Height Balanced Histogram

  # buckets < NDV

Histograms



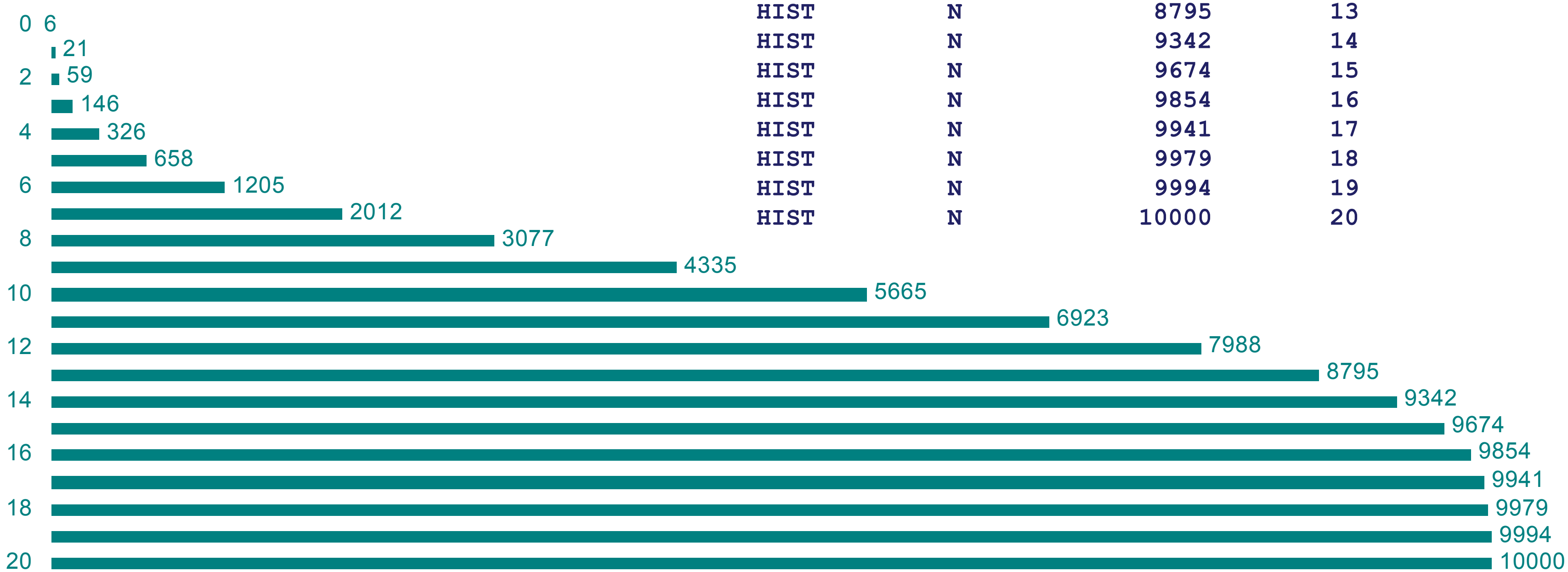
6
15
38
87
180
332
547
807
1065
1258
1330
1258
1065
807
547
332
180
87
38
15
6
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

Frequency Histogram

```
analyze table hist compute statistics for columns n [ size 75 ];
```

table	column	NDV	density	lo	hi	bkts
HIST	N	21	5.0000E-05	0	20	20

table	column	EP	value
HIST	N	6	0
HIST	N	21	1
HIST	N	59	2
HIST	N	146	3
HIST	N	326	4
HIST	N	658	5
HIST	N	1205	6
HIST	N	2012	7
HIST	N	3077	8
HIST	N	4335	9
HIST	N	5665	10
HIST	N	6923	11
HIST	N	7988	12
HIST	N	8795	13
HIST	N	9342	14
HIST	N	9674	15
HIST	N	9854	16
HIST	N	9941	17
HIST	N	9979	18
HIST	N	9994	19
HIST	N	10000	20



Frequency 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
HIST	N	1205	6
HIST	N	2012	7

```
N < 7       selectivity = 1205 / num_rows
N <= 7      selectivity = 2012 / num_rows
N = 7       selectivity = (2012-1205) / num_rows
```

Frequency Histogram

- Predicate does not match one of the values in the histogram

  Since this is a value base histogram that should mean there are no rows in the table with that value for the column and therefore the selectivity should be 0.
  However, the optimizer can not rely on the statistics being up-to-date and uses the density from the column statistics as selectivity.

- Bind Variable predicate

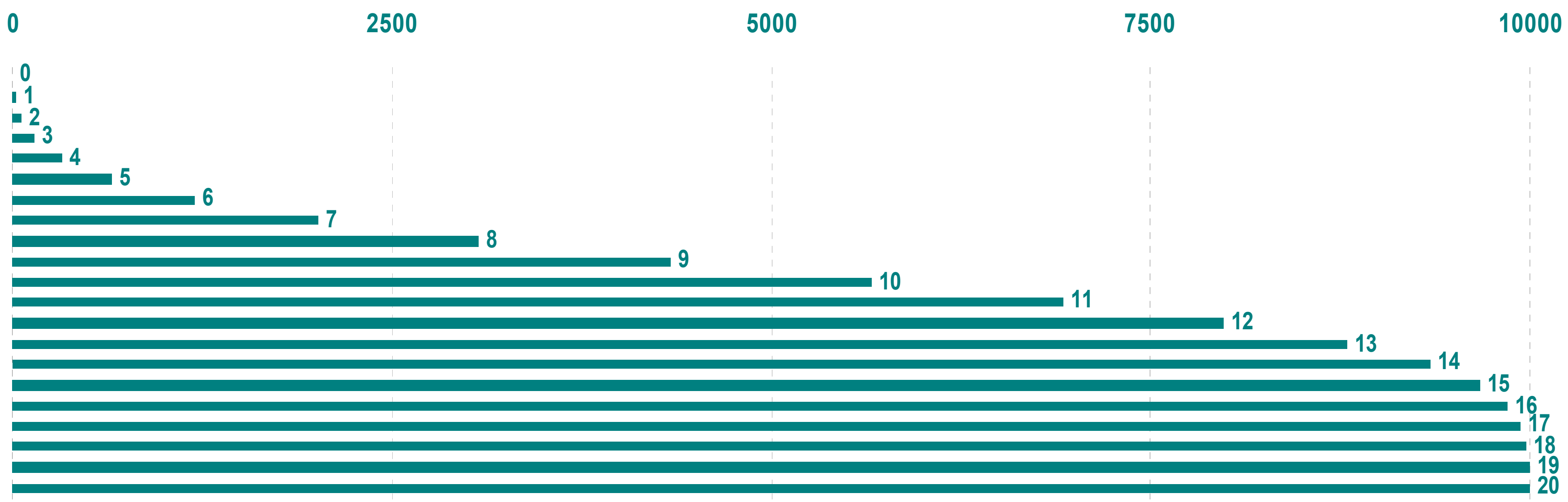
  The selectivity is taken as max(1/num_distinct, density), effectively ignoring the histogram.

Height Balanced Histogram

```
analyze table hist compute statistics for columns n size 4;
```

table	column	NDV	density	lo	hi	bkts
HIST	N	21	9.4128E-02	0	20	4

table	column	EP	value
HIST	N	0	0
HIST	N	1	8
HIST	N	2	10
HIST	N	3	12
HIST	N	4	20

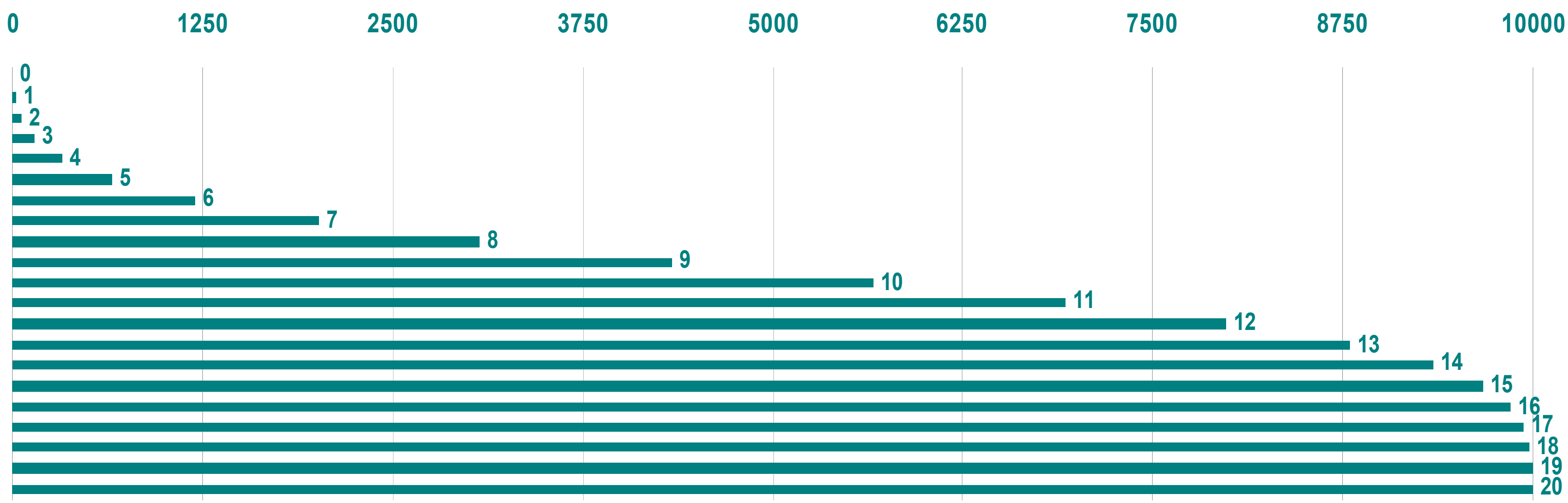


Height Balanced Histogram

```
analyze table hist compute statistics for columns n size 8;
```

table	column	NDV	density	lo	hi	bkts
HIST	N	21	6.2500E-02	0	20	8

table	column	EP	value
HIST	N	0	0
HIST	N	1	7
HIST	N	2	8
HIST	N	3	9
HIST	N	4	10
HIST	N	5	11
HIST	N	6	12
HIST	N	7	13
HIST	N	8	20

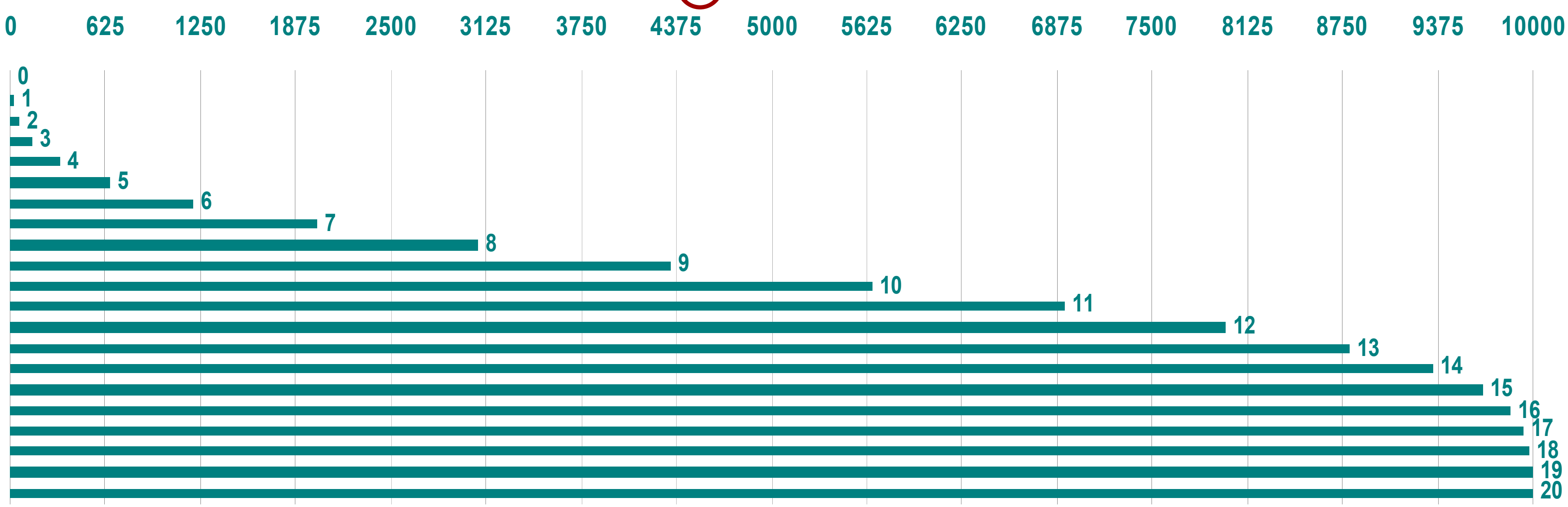


Height Balanced Histogram

```
analyze table hist compute statistics for columns n size 16
```

table	column	NDV	density	lo	hi	bkts
HIST	N	21	3.1250E-02	0	20	10

table	column	EP	value
HIST	N	0	0
HIST	N	1	5
HIST	N	3	7
HIST	N	4	8
HIST	N	6	9
HIST	N	9	10
HIST	N	11	11
HIST	N	12	12
HIST	N	14	13
HIST	N	15	15
HIST	N	16	20



Height Balanced Histogram

Predicate does not match any of the values in the histogram

	< | <=	=
selectivity =	EP of prior row / buckets + (value- $value_{low}$)/($value_{hi}$- $value_{low}$)/buckets	density

Example:

table	column	EP	value
HIST	N	0	0
HIST	N	1	5
HIST	N	3	7
HIST	N	4	8
HIST	N	6	9
HIST	N	9	10

table	column	EP	value
HIST	N	11	11
HIST	N	12	12
HIST	N	14	13
HIST	N	15	15
HIST	N	16	20

N < 17 selectivity = 15 / 16 + (17-15) / (20-15) / 16 = 9.6250E-01

N = 17 selectivity = density = 3.1250E-02

Height Balanced Histogram

Predicate matches a "non-popular" value in the histogram

	< | <=	=
selectivity =	EP of prior row / buckets	density

Example:

table	column	EP	value
HIST	N	0	0
HIST	N	1	5
HIST	N	3	7
HIST	N	4	8
HIST	N	6	9
HIST	N	9	10
HIST	N	11	11
HIST	N	12	12
HIST	N	14	13
HIST	N	15	15
HIST	N	16	20

N < 15 selectivity = 14 / 16 = 8.7500E-01

N = 15 selectivity = density = 3.1250E-02

Height Balanced Histogram

Predicate matches a "popular" value in the histogram

	< | <=	=
selectivity =	EP of prior row / buckets	range / buckets

Example:

table	column	EP	value
HIST	N	0	0
HIST	N	1	5
HIST	N	3	7
HIST	N	4	8
HIST	N	6	9
HIST	N	9	10
HIST	N	11	11
HIST	N	12	12
HIST	N	14	13
HIST	N	15	15
HIST	N	16	20

N < 13 selectivity = 12 / 16 = 7.5000E-01

N = 13 selectivity = 2 / 16 = 1.2500E-02

Histograms and Bind Variables

Density and cardinality estimate by # of buckets

```
Select emplid, jobcode, salary from ps_job5 b where b.company = :b1
```

buckets	density	card
10	1.0870E-02	109
25	8.5039E-03	86
50	7.4833E-03	75
75	6.0644E-03	61
90	5.5556E-03	56
100	5.0000E-03	50
150	3.3333E-03	50
199	2.5381E-03	50
200	5.0000E-05	50

Distribution over Range of Values

"The optimizer assumes that employee_id values are distributed evenly in the range between the lowest value and highest value."*

* Oracle 9*i* Performance Tuning Guide and Reference

Distribution over Range of Values

table	column	NDV	density	lo	hi
PS_LEDGER	ACCOUNTING_PERIOD	15	6.6667E-02	0	999

Period 0 holds opening balances, periods 1-12 hold the ledger entries for the months, and periods 998 and 999 are used for special processing.

Distribution over Range of Values

table	column	NDV	density	lo	hi
PS_LEDGER	ACCOUNTING_PERIOD	15	6.6667E-02	0	999

accounting_period = n [n ε {1 .. 12}]

$\Rightarrow$ selectivity = 1/ndv = 1/15 = 6.6667e^{-2}

accounting_period between 1 and 12

$\Rightarrow$ selectivity = 12/(999-0) + 1/15 = 7.8679e^{-2}

accounting_period < 12

$\Rightarrow$ selectivity = (12-0)/(999-0) = 1.2012e^{-2}

Distribution over Range of Values

Adjusting the high-value statistic

```
select sum(posted_total_amt) from ps_ledger
where accounting_period between 1 and 12

Column: ACCOUNTING  Col#: 11     Table: PS_LEDGER   Alias: PS_LEDGER
    NDV: 15         NULLS: 0          DENS: 6.6667e-002 LO:   0  HI: 999
  TABLE: PS_LEDGER      ORIG CDN: 745198  CMPTD CDN: 58632

Column: ACCOUNTING  Col#: 11     Table: PS_LEDGER   Alias: PS_LEDGER
    NDV: 15         NULLS: 0          DENS: 6.6667e-002 LO:   0  HI: 14
  TABLE: PS_LEDGER      ORIG CDN: 745198  CMPTD CDN: 684873

select sum(posted_total_amt) from ps_ledger
where accounting_period < 12

Column: ACCOUNTING  Col#: 11     Table: PS_LEDGER   Alias: PS_LEDGER
    NDV: 15         NULLS: 0          DENS: 6.6667e-002 LO:   0  HI: 999
  TABLE: PS_LEDGER      ORIG CDN: 745198  CMPTD CDN: 49680

Column: ACCOUNTING  Col#: 11     Table: PS_LEDGER   Alias: PS_LEDGER
    NDV: 15         NULLS: 0          DENS: 6.6667e-002 LO:   0  HI: 14
  TABLE: PS_LEDGER      ORIG CDN: 745198  CMPTD CDN: 638742
```

Predicate Independence Assumption

P1 AND P2	S(P1&P2) = S(P1) * S(P2)
P1 OR P2	S(P1|P2) = S(P1) + S(P2) -[S(P1) * S(P2)]

```
select emplid, jobcode, salary
from ps_job1 b
where b.company = 'CCC'
  and b.paygroup = 'FGH';
```

250 rows selected.

Explain Plan

```
 card operation
   251 SELECT STATEMENT
   251   TABLE ACCESS BY INDEX ROWID PS_JOB1
   251     INDEX RANGE SCAN PSBJOB1
```

```
select emplid, jobcode, salary
from ps_job2 b
where b.company = 'CCC'
  and b.paygroup = 'FGH';
```

2500 rows selected.

Explain Plan

```
 card operation
   251 SELECT STATEMENT
   251   TABLE ACCESS BY INDEX ROWID PS_JOB2
   251     INDEX RANGE SCAN PSBJOB2
```

Predicate Independence Assumption

table	rows	blks	empty	chain	avg rl
PS_JOB1	50,000	4,547	3	0	317

table	rows	blks	empty	chain	avg rl
PS_JOB2	50,000	4,547	3	0	317

table	column	NDV	density	bkts
PS_JOB1	EMPLID	10,000	1.0000E-04	1
PS_JOB1	JOBCODE	198	5.0505E-03	1
PS_JOB1	COMPANY	10	1.0000E-01	1
PS_JOB1	PAYGROUP	20	5.0000E-02	1
PS_JOB1	SALARY	49,597	2.0163E-05	1

table	column	NDV	density	bkts
PS_JOB2	EMPLID	10,000	1.0000E-04	1
PS_JOB2	JOBCODE	199	5.0251E-03	1
PS_JOB2	COMPANY	10	1.0000E-01	1
PS_JOB2	PAYGROUP	20	5.0000E-02	1
PS_JOB2	SALARY	49,848	2.0061E-05	1

$$card_{est} = card_{base} * sel\ (company\ AND\ paygroup)$$
$$= sel(company) * sel(paygroup)$$
$$= 50000 * 1.0000e^{-01} * 5.0000e^{-02} = 250$$

index	column	NDV	#LB
PSBJOB1		200	400
	COMPANY	10	
	PAYGROUP	20	

index	column	NDV	#LB
PSBJOB2		20	449
	COMPANY	10	
	PAYGROUP	20	

Join Uniformity Assumption

join cardinality = $card_A$ * $card_B$ * join selectivity

join selectivity = $1/\max(ndv_A, ndv_B)$

“principle of inclusion”, i.e. each value of the smaller domain has a match in the larger domain – which is frequently true for joins between foreign keys and primary keys.

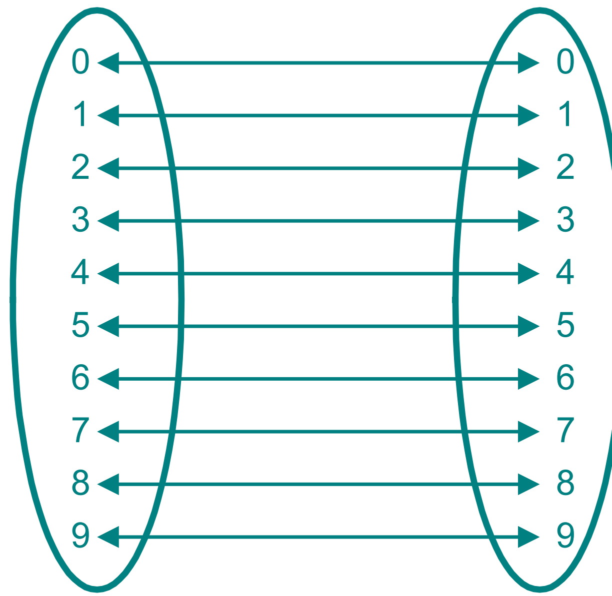
Join Uniformity Assumption

```
SQL> select 'A-'||a.n1, 'B-'||b.n1
  2  from t1 a, t1 b
  3  where a.n1 = b.n1;


   10 SELECT STATEMENT
   10   HASH JOIN
   10     TABLE ACCESS FULL T1
   10     TABLE ACCESS FULL T1
```

```
A-0  B-0
A-1  B-1
A-2  B-2
A-3  B-3
A-4  B-4
A-5  B-5
A-6  B-6
A-7  B-7
A-8  B-8
A-9  B-9

10 rows selected.
```



$$\text{Join cardinality} = card_A * card_B * \text{join selectivity}$$
$$= card_A * card_B * 1/\max(ndv_a, ndv_b)$$
$$= 10 * 10 * 1/\max(10, 10) = 10$$

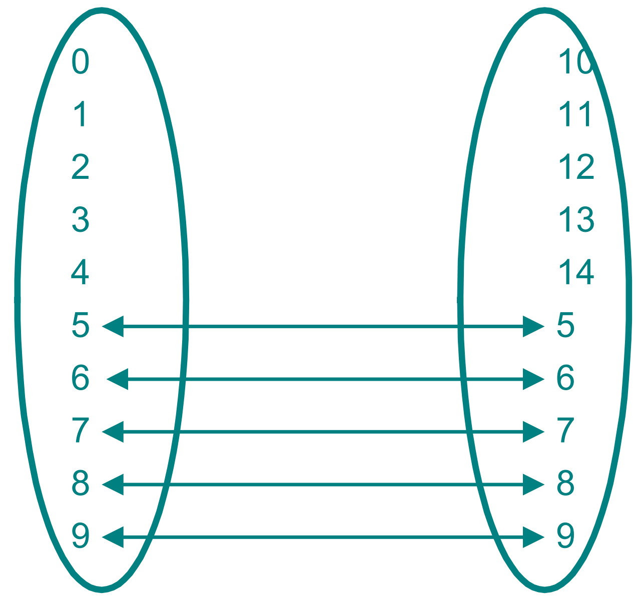
Join Uniformity Assumption

```
SQL> select 'A-'||a.n1, 'B-'||b.n1
  2  from t1 a, t2 b
  3  where a.n1 = b.n1;


    10 SELECT STATEMENT
    10   HASH JOIN
    10     TABLE ACCESS FULL
T2
    10     TABLE ACCESS FULL
T2
```

```
A-5   B-5
A-6   B-6
A-7   B-7
A-8   B-8
A-9   B-9

5 rows selected.
```



$$\text{Join cardinality} = card_A * card_B * \text{join selectivity}$$
$$= card_A * card_B * 1/\max(ndv_a, ndv_b)$$
$$= 10 * 10 * 1/\max(10, 10\ ) = 10$$

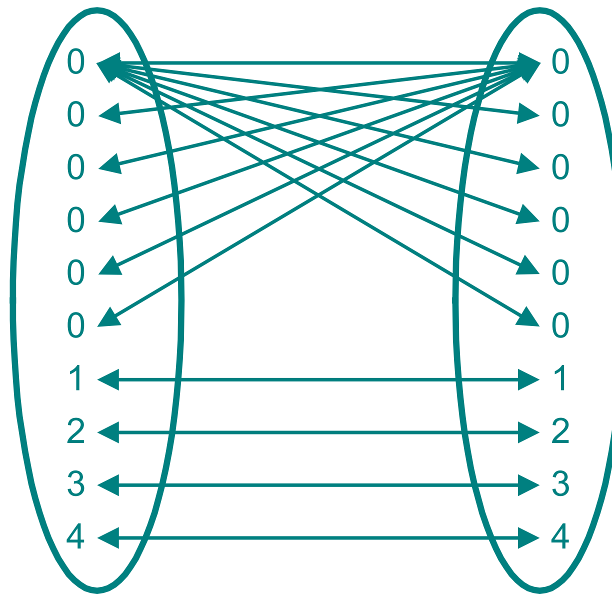
Join Uniformity Assumption

```
SQL> select 'A-'||a.n1, 'B-'||b.n1
  2  from t2 a, t2 b
  3  where a.n1 = b.n1;


    20 SELECT STATEMENT
    20   HASH JOIN
    10     TABLE ACCESS FULL T2
    10     TABLE ACCESS FULL T2
```

```
A-0   B-0
A-0   B-0
A-0   B-0
…
A-0   B-0
A-1   B-1
A-2   B-2
A-3   B-3
A-4   B-4

40 rows selected.
```



$$\text{Join cardinality} = card_A * card_B * \text{join selectivity}$$
$$= card_A * card_B * 1/\max(ndv_a, ndv_b)$$
$$= 10 * 10 * 1/\max(5, 5) = 20$$

Join Selectivity and Cardinality

```
insert into t1(n1,n2)
select mod(rownum,10),mod(rownum,5)
from dba_objects where rownum <= 50;
```

column	NDV	density
N1	10	1.0000E-01
N2	5	2.0000E-01

```
select 'A.'||A.n1||'-B.'||B.n1
from t1 a, t2 b
where a.n1 = b.n1;
```

Explain Plan

```
card operation
 250 SELECT STATEMENT
 250   HASH JOIN
  50     TABLE ACCESS FULL T1
  50     TABLE ACCESS FULL T2
```

Execution Plan

```
Rows    Execution Plan
     0  SELECT STATEMENT   GOAL: CHOOSE
   250   HASH JOIN
    50    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T1'
    50    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T2'
```

Join Selectivity and Cardinality

```
select 'A.'||A.n1||'-B.'||B.n1
from t1 a, t2 b
where a.n1 = b.n1
  and a.n2 = 5;
```

```
Explain Plan

    card operation
      50 SELECT STATEMENT
      50   HASH JOIN
      10     TABLE ACCESS FULL T1
      50     TABLE ACCESS FULL T2
```

```
Execution Plan

Rows     Execution Plan
      0  SELECT STATEMENT   GOAL: CHOOSE
     50   HASH JOIN
     10    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T1'
     50    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T2'
```

Join Selectivity and Cardinality

```
select 'A.'||A.n1||'-B.'||B.n1
from t1 a, t2 b
where a.n1 = b.n1
  and a.n1 = 5;
```

```
Explain Plan

   card operation
      5 SELECT STATEMENT
      5   HASH JOIN
      5      TABLE ACCESS FULL T1
      5      TABLE ACCESS FULL T2
```

```
Execution Plan

Rows     Execution Plan
      0  SELECT STATEMENT   GOAL: CHOOSE
     25   HASH JOIN
      5    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T1'
      5    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T2'
```

Join Selectivity and Cardinality

```
Table stats    Table: T2   Alias:  B
  TOTAL ::  CDN: 50  NBLKS:  1  TABLE_SCAN_CST: 1  AVG_ROW_LEN:  8

Table stats    Table: T1   Alias:  A
  TOTAL ::  CDN: 50  NBLKS:  1  TABLE_SCAN_CST: 1  AVG_ROW_LEN:  8

SINGLE TABLE ACCESS PATH
Column:         N1  Col#: 1      Table: T1   Alias:  A
    NDV: 10        NULLS: 0          DENS: 1.0000e-001 LO:  0  HI: 9
  TABLE: T1     ORIG CDN: 50  CMPTD CDN: 5

SINGLE TABLE ACCESS PATH
Column:         N1  Col#: 1      Table: T2   Alias:  B
    NDV: 10        NULLS: 0          DENS: 1.0000e-001 LO:  0  HI: 9
  TABLE: T2     ORIG CDN: 50  CMPTD CDN: 5

...

Join cardinality:  5 = outer (5) * inner (5) * sel (2.0000e-001)  [flag=0]
```

Transitive Closure

```
select
a.n1,a.n2,a.n3,b.n1,b.n2,b.n3,c.n1,c.n2,c.n3
from t4 a, t5 b, t6 c
where a.n1 = b.n1
  and b.n2 = c.n2
  and b.n1 = c.n1

    cost        card operation                    Rows    Execution Plan
-------- ----------- ---------------------------- ------- ----------------------------------------------
      23         198 SELECT STATEMENT                   0 SELECT STATEMENT   GOAL: CHOOSE
      23         198   HASH JOIN                      202  HASH JOIN
       6         100     HASH JOIN                    100   HASH JOIN
       1          20       TABLE ACCESS FULL T4        20    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T4'
       4         100       TABLE ACCESS FULL T5       100    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T5'
      16         500     TABLE ACCESS FULL T6         500   TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T6'

select
a.n1,a.n2,a.n3,b.n1,b.n2,b.n3,c.n1,c.n2,c.n3
from t4 a, t5 b, t6 c
where a.n1 = b.n1
  and b.n2 = c.n2
  and b.n1 = c.n1
  and a.n1 = c.n1

    cost        card operation                    Rows    Execution Plan
-------- ----------- ---------------------------- ------- ----------------------------------------------
      23          18 SELECT STATEMENT                   0 SELECT STATEMENT   GOAL: CHOOSE
      23          18   HASH JOIN                      202  HASH JOIN
       6         100     HASH JOIN                    100   HASH JOIN
       1          20       TABLE ACCESS FULL T4        20    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T4'
       4         100       TABLE ACCESS FULL T5       100    TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T5'
      16         500     TABLE ACCESS FULL T6         500   TABLE ACCESS   GOAL: ANALYZED (FULL) OF 'T6'
```

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
```

References

Oracle 9*i* Performance Tuning Guide and Reference

Note 10626.1	Cost Based Optimizer (CBO) Overview
Note 35934.1	Cost Based Optimizer - Common Misconceptions and Issues
Note 212809.1	Limitations of the Oracle Cost Based Optimizer
Note 46234.1	Interpreting Explain plan
Note 68992.1	Predicate Selectivity
Steve Adams	Ixora News - April 2001. www.ixora.com.au/newsletter/2001_04.htm
Cary Millsap	When to Use an Index. www.hotsos.com

References

G. Piatetsky-Shapiro, C. Connell: *Accurate Estimation of the Number of Tuples Satisfying a Condition*. Proceedings of the ACM SIGMOD International Conference on Management of Data. June 1984. p. 256-275.

C. A. Lynch: *Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distribution of Column Values*. Proceedings of the International Conference on Very Large Data Bases. August, 1988. p. 240-251.

Y. E. Ioannidis, S. Christodoulakis: *On the Propagation of Errors in the Size of Join Results*. Proceedings of the ACM SIGMOD International Conference on Management of Data. May, 1991. p. 268-277.

A. N. Swami, K. B. Schiefer: *On the Estimation of Join Result Sizes*. Proceedings of the International Conference on Extending Database Technology. March, 1994. p. 287-300.

M. Stillger, G. Lohman, V. Markl, M. Kandil: *LEO – DB2's LEarning Optimizer*. Proceedings of the International Conference on Very Large Data Bases. September 2001. Rome, Italy. p. 19-28

Wolfgang Breitling

Centrex Consulting Corporation

www.centrexcc.com