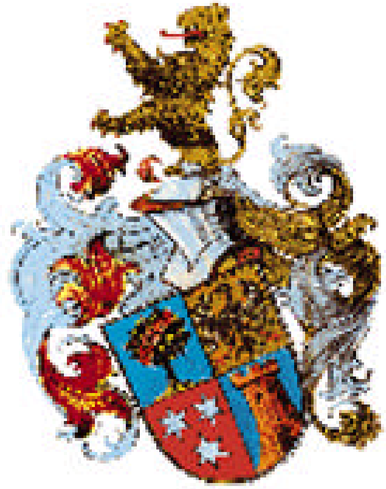
# Fallacies of the Cost Based Optimizer

Wolfgang Breitling
breitliw@centrexcc.com



# Who am I

- Independent consultant since 1996
  - specializing in Oracle and Peoplesoft setup, administration, and performance tuning
- 25+ years in database management
  - DL/1, IMS, ADABAS, SQL/DS, DB2, Oracle
- OCP certified DBA - 7, 8, 8*i*, 9*i*
- Oracle since 1993 (7.0.12)
- Mathematics major at University of Stuttgart

# Who are YOU ?

DBA

Developer

Management

Oracle 9 R1 / R2

Oracle 8

Oracle 7 ??

Oracle 10 ??

# 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

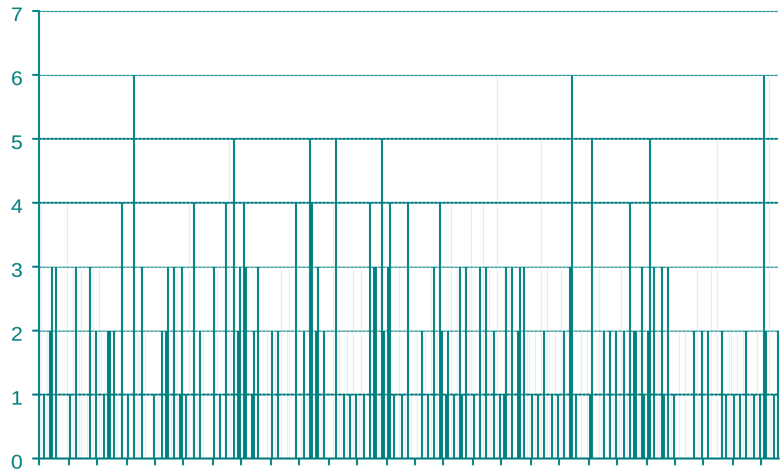
The cost of an access plan is a function of the estimated cardinalities of its components.

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

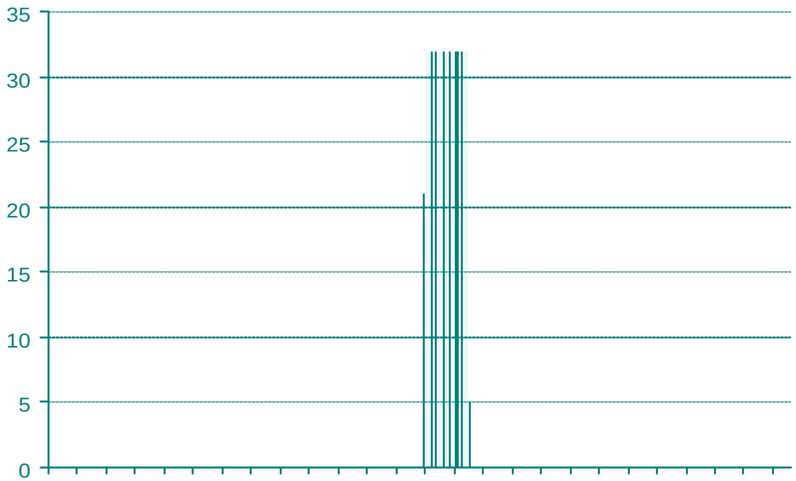
This is why accurate cardinality estimates are so important.

# Distribution over blocks

uniform




clustered



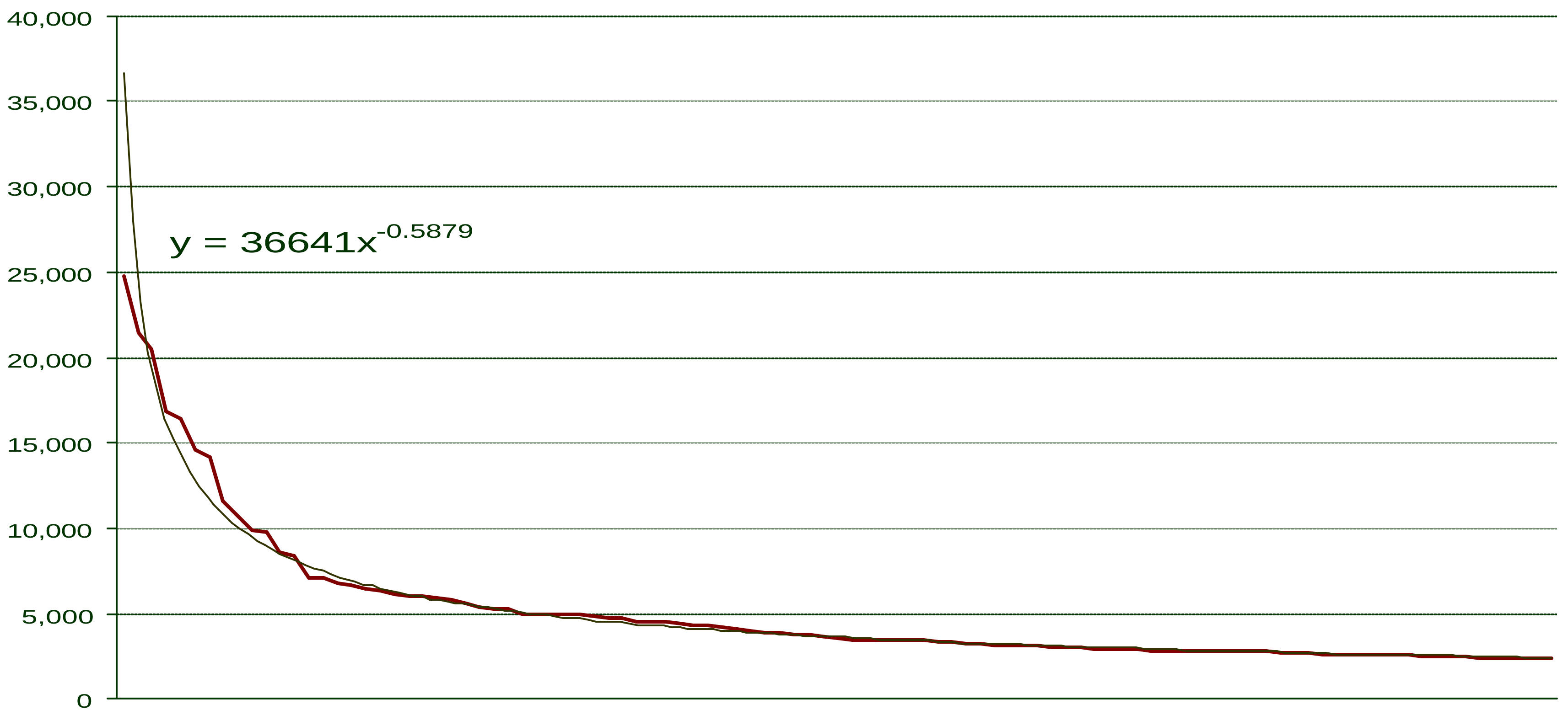
# Distribution of Value Frequencies

“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'
```

explain plan

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

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

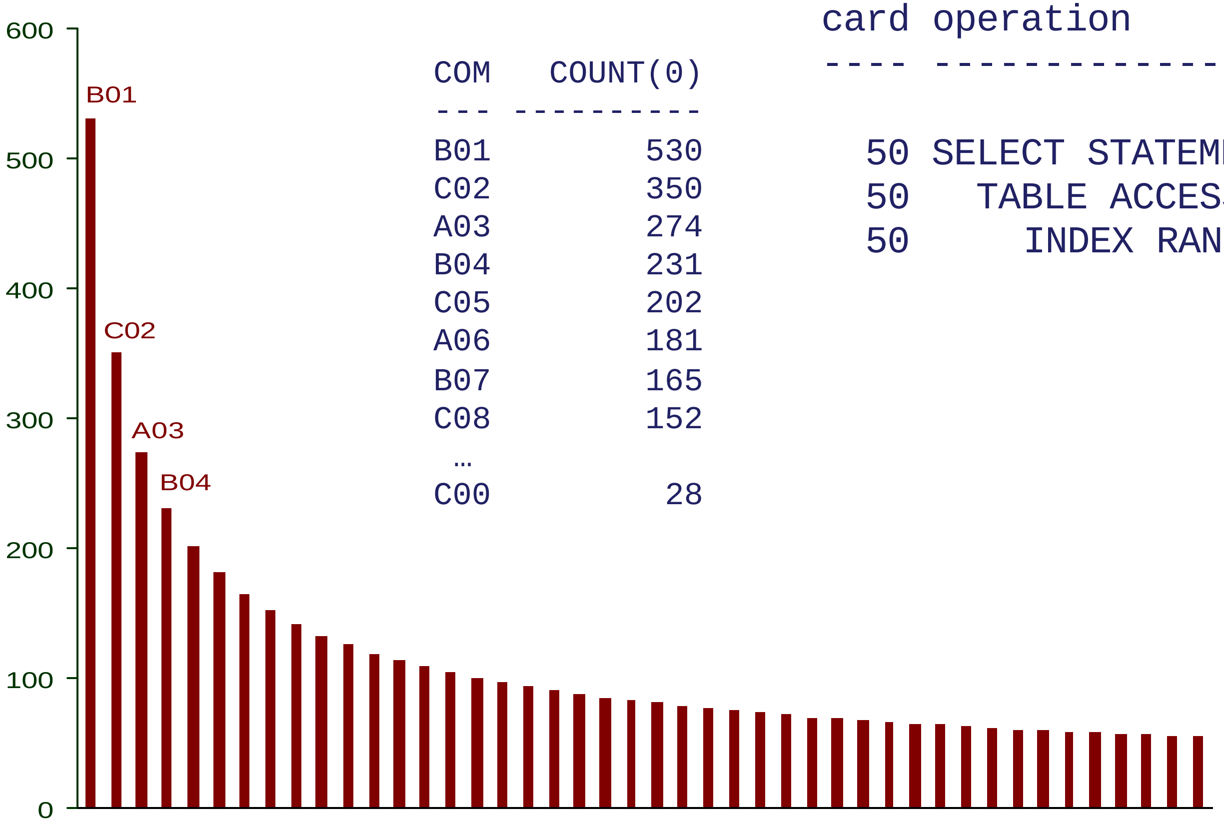
execution plan

```
  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 |




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

```
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
```

# 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
```

explain plan

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

# 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.

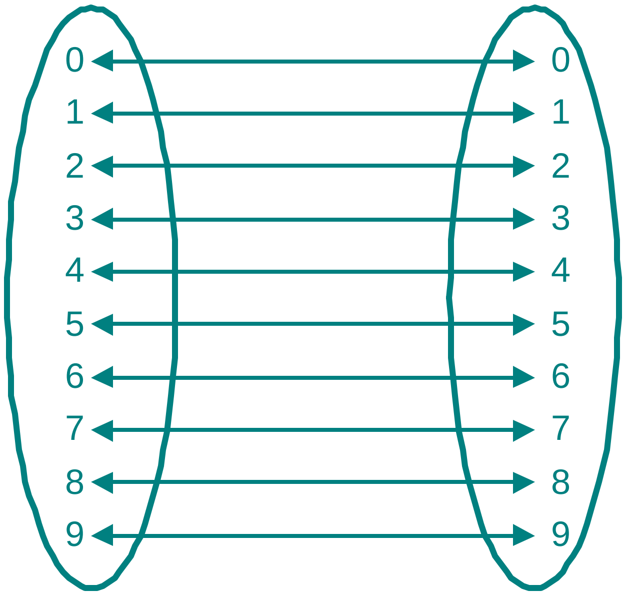
# Join Uniformity Assumption

```
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.
```




$$\text{Join cardinality} = \text{card}_A * \text{card}_B * \text{join selectivity}$$
$$= \text{card}_A * \text{card}_B * 1/\max(\text{ndv}_a, \text{ndv}_b)$$
$$= 10 * 10 * 1/\max(10, 10\ ) = 10$$

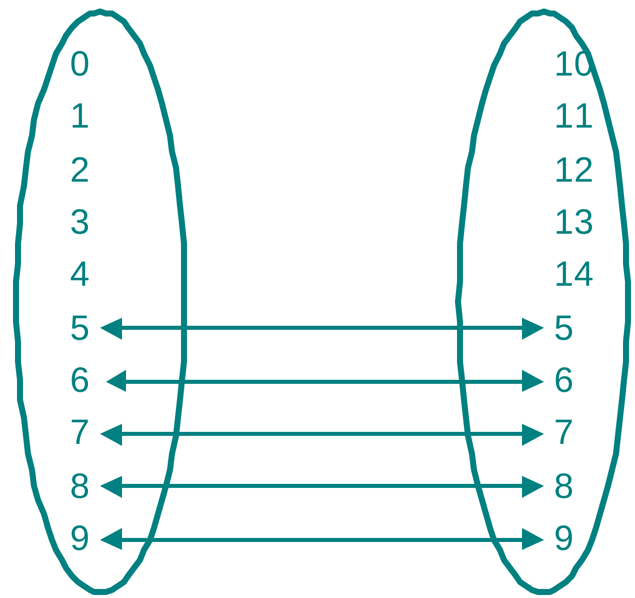
# Join Uniformity Assumption

```
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.
```




$$\text{Join cardinality} = \text{card}_A * \text{card}_B * \text{join selectivity}$$
$$= \text{card}_A * \text{card}_B * 1/\max(\text{ndv}_a, \text{ndv}_b)$$
$$= 10 * 10 * 1/\max(10, 10\ ) = 10$$

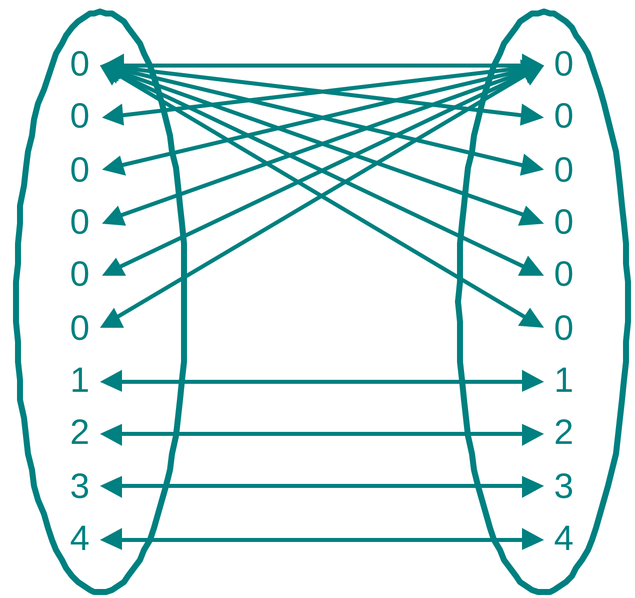
# Join Uniformity Assumption

```
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

## Oracle 9i (9.2.0.4) and 10g

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

 Card Plan                                                     9i & 10g
   50   SELECT STATEMENT (all_rows)                            (Cost 5)
   50     HASH JOIN                                            (Cost 5)
   10       TABLE ACCESS (analyzed) TABLE SCOTT T1 (full)      (Cost 2)
   50       TABLE ACCESS (analyzed) TABLE SCOTT T2 (full)      (Cost 2)
```

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

 Card Plan                                                     9i         10g
   25   SELECT STATEMENT (all_rows)                            (Cost 12) (Cost 4)
   25     MERGE JOIN    (cartesian)                            (Cost 12) (Cost 4)
    5       TABLE ACCESS (analyzed) TABLE SCOTT T1 (full)      (Cost 2)  (Cost 2)
    5       BUFFER    (sort)                                   (Cost 10) (Cost 2)
    5         TABLE ACCESS (analyzed) TABLE SCOTT T2 (full)    (Cost 2)  (Cost 0)
```

# 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

breitliw@centrexcc.com

www.centrexcc.com