Notation

We use the following parameters

- $B(R)$: The number of (disk) blocks needed for the tuples in R
- $T(R)$: The number of tuples in R; if we know the size of a tuple, we can easily convert from $T(R)$ to $B(R)$
- $V(R, a)$: The *value count* for attribute a in R. The value count is the number of distinct values relation R has for attribute a
- Similarly, $V(a_1, \ldots, a_k)$ is the number of distinct values the tuple of attributes a_1, $\ldots$, a_k has in R

Example

R:

A	B	C
1	2	3
2	2	3
3	2	3
3	4	3
4	5	3
5	5	3

- $V(R, A) = 5$
- $V(R, B) = 3$
- $V(R, C) = 1$
- $V(R, [A, B]) = 6$
- $V(R, [A, C]) = 5$
- $V(R, [B, C]) = 3$
- $V(R, [A, B, C]) = T(\delta(R))$

Estimating parameters

- Usually, we do not have exact statistics for relations, but only estimates
- The better the estimates, the better our query plan will be
- If the estimates are very bad, we will select less efficient plans, but we will still get the right answer
- Estimates can be maintained, and changed whenever the database is updated
- If we have an index, we can estimate $T(R)$ by counting the number of blocks in the index (assuming, e.g., that each block is 75% full)
- If a is a key (or `UNIQUE`), $V(R, a) = T(R)$
- $V(R, a)$ can also be estimated by sampling

Estimating sizes of results

- If we have a complex algebraic expression, we need to estimate the size of the results (inputs to other expressions) as well
- The rules we use are also *estimates*. Even if the parameters are known precisely, our estimates of the results may not be accurate
- We use very simple rules. In practice, knowlege of the domain, sampling, remembering parameters from previous queries etc., can be used to improve the estimates

Projection

- Can be estimated precisely
- Projection (bag projection) reduces the size of each tuple without changing the number
- $T(\pi_A(R)) = T(R)$
- $V(\pi_A(A), A) = V(R, A)$
- $B(\pi_A(R))$ requires knowlege of the sizes of tuples and blocks but is a fixed proportion
- We'll see a few examples, but we don't usually estimate costs of projections

$T(R) = 10.000$

Tuples

- $R(a, b, c)$
- a, b integers (4 bytes)
- c string of 100 bytes
- Tuple header: 12 bytes
- So each tuple takes 120 bytes

Blocks

- Block: 1024 bytes
- Block header: 24 bytes
- So: 8 tuples per block
- $B(R) = 1.250$

Example: $S = \pi_{a+b,c}(R)$

- Tuples now need 116 bytes
- Even so, only 8 fit in a block
- $T(S) = 10.000$, $B(S) = 1.250$

$U = \pi_{a,b}(R)$

- Tuples are now only 20 bytes long
- 50 tuples now fit in a block
- $T(U) = 10.000$, $B(U) = 200$

Selection

$S = \sigma_{A=c}(R)$

- We use the rule that $T(S) = \frac{T(R)}{V(R,A)}$
- This assumes a uniform distribution of values for A
- Domain knowledge or other techniques could be used to get better estimates
- A similar rule for $B(S)$
- We assume that, for other attributes, $V(S, B) = V(R, B)$ (and $V(S, A)$ is of course equal to 1)

Range queries

What about $S = \sigma_{A<10}(R)$

- Natural assumption: On average, half the tuples
- However, people tend to write queries looking for exceptions
- So in practice, usually less than half
- We use $T(S) = \frac{T(R)}{3}$

$S = \sigma_{a \neq 10}(R)$

- $T(S) = T(R)$ is probably good enough
- But we could use

$$T(S) = \frac{T(R)(V(R,a) - 1)}{V(R,a)}$$

`AND`: Iterate using our rules

- Example: $R(a, b, c)$, $T(R) = 10.000$, $V(R, a) = 50$
- $S = \sigma_{a=10 \text{ AND } b<20}(R)$
- $T(S) = \frac{T(R)}{50 \times 3}$
- Note that the order does not matter
- What are $V(R, a)$, $V(R, b)$ and $V(R, c)$?

$S = \sigma_{C1 \texttt{ OR } C2}(R)$

- One approximation: Sum of $T(\sigma_{C1}(R))$ and $T(\sigma_{C2}(R))$
- If a lot of tuples satisfy both, this could be much too large
- Another approach: take the smaller
- Better, but more complicated: Take the sum, and subtract an estimate of the conjunction (assuming the two conditions are independent)

Example

- $R(a, b)$, $T(R) = 10.000$, $V(R, a) = 50$
- $S = \sigma_{a=10 \texttt{ OR } b<20}(R)$
- First condition: 200 tuples
- Second condition: 3.333 tuples
- Sum: 3533
- Estimate for conjunction: 66 tuples
- Better estimate for S: $3533 - 66 = 3467$

Other operations

- Union $R \cup S$
  - Bag: Sum of the two arguments $T(R) + T(S)$
  - Set: Between $T(R) + T(S)$ and $\max(T(R), T(S))$. We use the average of these
- Intersection: Between 0 and $\min(T(R), T(S))$. We use the average
- Difference $R - S$. Between $T(R)$ and $T(R) - T(S)$. We take

$$T(R) - \frac{T(S)}{2}$$

- Duplicate elimination $\delta(R(a, b, c))$
  - Cannot be greater than $V(R, a)V(R, b)V(R, c)$
  - We take $\min(\frac{T(R)}{2}, V(R, a)V(R, b)V(R, c))$
- Aggregation. Same idea as δ, but only use the attributes that are left in the result

Join

- We focus on natural join
- Our techniques can be extended to theta-joins
- We consider two relations, $R(X, Y) \bowtie S(Y, Z)$
- For now, assume that Y is a single attribute
- We start with the case where Y is a foreign key from R to S
- In this case $T(R \bowtie S) = T(R)$

Now, assume that Y is a reference from R to S, but is not a key in S

- Every value of Y in R is present in at least one tuple of S
- Therefore $V(R,Y) \leq V(S,Y)$ (we shall use this later)
- Each tuple in R joins with, on average, $\frac{T(S)}{V(S,Y)}$ tuples of S
- So

$$T(R \bowtie S) = \frac{T(R)T(S)}{V(S,Y)}$$

Now, let Y is a reference from S to R. By the same reasoning

- Therefore $V(S,Y) \leq V(R,Y)$
- And

$$T(R \bowtie S) = \frac{T(R)T(S)}{V(R,Y)}$$

Putting this together

$$T(R \bowtie S) = \frac{T(R)T(S)}{\max(V(R,Y), V(S,Y))}$$

In practice, we usually join two relations, only when there is a reason to do so, which often means a relation like the one we have just studied

Formally, we assume that the following two conditions hold

- *Containment of value sets*
  - If Y is in several relations, the values of Y always are a prefix of a fixed string of values y_1, y_2, ...
  - This implies that if $V(R, Y) \leq V(S, Y)$, then every value of Y in R is also a value of Y in S
- *Preservation of value sets*
  - If we join R with S, and A is not a join attribute, then the list of possible values for A does not change
  - This implies that $V(R \bowtie S, A) = V(R, A)$

Under these assumptions, we can show that our formula is valid

$$T(R \bowtie S) = \frac{T(R)T(S)}{\max(V(R, Y), V(S, Y))}$$

Example

Three relations, $R(a, b)$, $S(b, c)$, and $U(c, d)$

- $T(R) = 1.000$, $V(R, b) = 20$
- $T(S) = 2.000$, $V(S, b) = 50$, $V(S, c) = 100$
- $T(U) = 5.000$, $V(U, c) = 500$

Compute $R \bowtie S \bowtie U$. We have to pick a pair to join first

We start with $(R \bowtie S) \bowtie U$

- $T(R \bowtie S) = \frac{1.000 \times 2.000}{\max(50,20)} = 40.000$
- $V(R \bowtie S, c) = V(S, c) = 100$
- $T((R \bowtie S) \bowtie U = \frac{40.000 \times 5.000}{\max(100,500)} = 400.000$

Now look at $R \bowtie (S \bowtie U)$

- $T(S \bowtie U) = \frac{2.000 \times 5.000}{\max(100,500)} = 20.000$
- $V(S \bowtie U, b) = V(S, b) = 50$
- $T(R \bowtie (S \bowtie U) = \frac{20.000 \times 1.000}{\max(20,50)} = 400.000$

The estimates are the same, and this can be shown to always hold with our formula

But note that the first approach computed an intermediate relation with 40.000 tuples, while the second needed only 20.000

So the second approach will be more efficient

More than one attribute

Assume that $Y = \{y_1, y_2\}$

- Take tuples r and s. What is the probability that they agree on y_1?
  - If $V(R, y_1) \geq V(S, y_1)$, by containment of value sets this is $\frac{1}{V(R,y_1)}$
  - If $V(R, y_1) \leq V(S, y_1)$, by containment of value sets this is $\frac{1}{V(S,y_1)}$
  - So the probability they agree on y_1 is $\frac{1}{\max(V(R,y_1),V(S,y_1))}$
- The same argument shows that the probability they agree on y_2 is

$$\frac{1}{\max(V(R, y_2), V(S, y_2))}$$

- Therefore of all $T(R)T(S)$ pairs of tuples, the expected number of that match is

$$\frac{T(R)T(S)}{\max(V(R, y_1), V(S, y_1)) \max(V(R, y_2), V(S, y_2))}$$

This generalizes easily to more attributes and to joins of more than 2 relations

Going back to our last example, there is a third way to evaluate the join

- Take $(R \bowtie U) \bowtie S$
- R and U have no attributes in common, so this is a cartesian product
- $T(R \times U) = 1.000 \times 5.000 = 5.000.000$ (so the intermediate result is much bigger)
- $V(R \times U, b) = V(R, b) = 20$, $V(R \times U, c) = V(U, c) = 500$

$$T(R \bowtie U) \bowtie S) = \frac{2.000 \times 5.000.000}{\max(20, 50) \max(100, 500)} = 400.000$$

If we are not interested in the sizes of intermediate relations, we could compute it directly as

$$T(R \bowtie S \bowtie U) = \frac{1.000 \times 2.000 \times 5.000}{\max(20, 50) \max(100, 500)} = 400.000$$

Another example

$$U = R(a, b, c) \bowtie_{R.b=S.d \text{ AND } R.c=S.e} S(d, e, f)$$

- $T(R) = 1.000$, $V(R, b) = 20$, $V(R, c) = 100$
- $T(S) = 2.000$, $V(S, d) = 50$, $V(S, e) = 50$

$$T(U) = \frac{1.000 \times 2.000}{\max(20, 50) \max(100, 50)} = 400$$

Another example

$$W = R(a, b, c) \bowtie S(b, c, d) \bowtie U(b, e)$$

Parameters

- $T(R) = 1.000$, $V(R, a) = 100$, $V(R, b) = 20$, $V(R, c) = 200$
- $T(S) = 2.000$, $V(S, b) = 50$, $V(S, c) = 100$, $V(S, d) = 400$
- $T(U) = 5.000$, $V(U, b) = 200$, $V(U, e) = 500$

$$T(R \bowtie S) = \frac{1.000 \times 2.000}{\max(50, 20) \max(100, 200)}$$

So

$$T(W) = \frac{1.000 \times 2.000 \times 5.000}{\max(50, 20) \max(100, 200) \max(50, 200)} = 5.000$$

Furthermore

- $V(W, a) = 100$
- $V(W, b) = 20$
- $V(W, c) = 100$
- $V(W, d) = 400$
- $V(W, e) = 500$

Better join estimates

Instead of the parameters we have used, we could gather more detailed information stored as *histograms*

For small $V(R, a)$ we could store the number of occurences of each attribute

When $V(R, a)$ is larger, we could use

- Equal width histograms (we'll see an example later)
- Equal height histograms
- Detailed information only for the most frequent values

Most frequent value histograms

$R(a, b) \bowtie S(b, c)$

Suppose we have the following information about the three most frequent values

- $R.b$: 1 (200 tuples), 0 (150), 5 (100), others (550)
- $S.b$: 0 (100), 1 (80), 2 (70), others (250)
- $V(R, b) = 14$, $V(S, b) = 13$

So $T(R) = 1.000$ and $T(S) = 500$

Join size estimate: $\frac{1.000 \times 500}{14} = 35.714$

Using the histograms

Value of attribute b

- 0. 150 tuples in R join with 100 tuples in S: 15.000 tuples
- 1. 200 tuples in R join with 80 tuples in S: 16.000 tuples
- 2. 70 tuples in S. How many in R?
  - Since $V(R, b) < V(S, b)$, 2 must be a value of R
  - It must be one of the "others"
  - These 550 tuples are divided among $V(R, b) - 3 = 11$ values
  - On average, 50 of them have $b = 2$
  - The result then has $50 \times 70 = 3.500$
- 5. 100 tuples in R. As one of the most frequent values, we assume it is present in S. "others" for S has 250 tuples, divided among 10 values, so about 25 with value 5. The join therefore has $100 \times 25 = 2.500$

- Other values.
  - 9 b-values are left that are in both relations
  - They must all be among "others"
  - We have already worked out that each b-value from R in "others" appears 50 times and each such value from S appears 25 times
  - This means $9 \times 50 \times 25 = 9 \times 1.250$
- Our new join estimate is then $15.000 + 16.000 + 3.500 + 2.500 + 9 \times 1.250 = 48.250$
- Compare with 35.714

Equal-width partition

- Two relations, `Jan(day,temp)` and `July(day,temp)`
- Find pairs of days in January and July with the same temperature

```
SELECT Jan.day, July.day
FROM Jan,July
WHERE Jan.temp = July.temp
```

- Assume temperatures (Farenheit) between 0-100, i.e.,

$$V(\texttt{Jan}, \texttt{Temp}) = V(\texttt{Jul}, \texttt{Temp}) = 100$$

- Assume each relation has 245 tuples
- Then the join has approx

$$\frac{245 * 245}{100} = 600$$

tuples

Histograms

Each band: 10 values for `temp`

Range	Jan	July
0-9	40	0
10-19	60	0
20-29	80	0
30-39	50	0
40-49	10	5
50-59	5	20
60-69	0	50
70-79	0	100
80-89	0	60
90-99	0	10

- If a band has T_1 tuples (January) and T_2 tuples (July), and there are V (10) values per band, the join has $\frac{T_1 T_2}{V}$ tuple on average
- Most bands have 0 tuples in one of the relations
- Our estimate is therefore

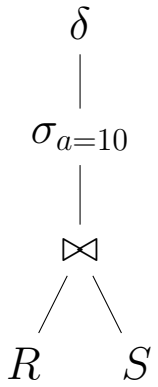
$$\frac{10 \times 5}{10} + \frac{5 \times 20}{10} = 15$$

- Compare with 600
- If we knew that January temperatures were always between 0 and 50 (50 values) and July between 40 and 99 (60) we could have got a got a better estimate of

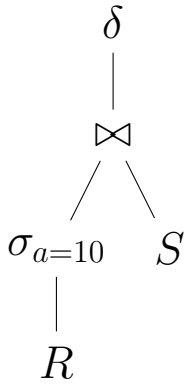
$$\frac{245 * 245}{\max(50, 60)} = 1.000$$

Algebraic optimization

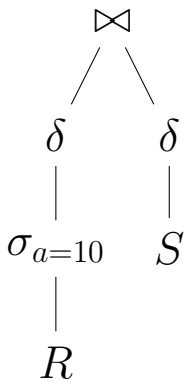
Given the query tree

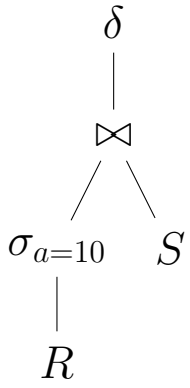


We push down the selection:



We can (but don't have to) push down the δ. We get two possible algebraic query plans

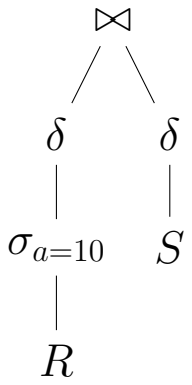




Which is better?

We need some statistical information

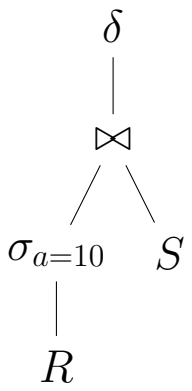
- $T(R) = 5.000$, $V(R, a) = 50$, $V(R, b) = 100$
- $T(S) = 2.000$, $V(S, b) = 200$, $V(S, c) = 100$



Estimates:

- $\sigma_{a=10}(R)$: $\frac{5000}{50} = 100$
- $\delta(\sigma_{a=10}(R))$: $\frac{100}{2} = 50$
- $\delta(S)$: $\frac{2000}{2} = 1000$
- Final join: $\frac{50 \times 1.000}{\max(100,200)} = 250$

Second plan



- $T(\sigma_{a=10}(R)) = \frac{5.000}{50} = 100$
- $T(\sigma_{a=10}(R) \bowtie S) = \frac{100 \times 2.000}{\max(100,200)} = 1.000$
- Final result: $\frac{1.000}{2} = 500$

Which is better?

- First approach has 250 tuples, the second 500
- Does that mean the first is better?
- No! The result of a query is fixed. If this estimates are very different, this just means that at least one of our estimates is not very good
- This is an indication that we should try to get better estimates for the future
- But for the current query, we have to make do with what we have
- What matter is not the final result (fixed) but how much work we have to do to compute it
- We do this by estimating how many tuples we have to construct for the *intermediate results*
- We don't count the input relations (fixed) or the output (also fixed, even if we don't know the value)

Intermediate relations constructed with the first plan

- $\sigma_{a=10}(R)$: 100
- $\delta(\sigma_{a=10}(R))$: 50
- $\delta(S)$: 1000
- Total: 1.150 tuples

Second plan:

- $T(\sigma_{a=10}(R))$: 100
- $T(\sigma_{a=10}(R) \bowtie S)$: 1.000
- Total: 1.100

So the second plan is (slightly) better

Let's try with smaller $V(R, b) = V(S, b) = 10$

First plan remains 1.150, as we never use this parameter

The second plan is now:

- $T(\sigma_{a=10}(R))$: 100
- $T(\sigma_{a=10}(R) \bowtie S)$: $\frac{100 \times 2.000}{10} = 20.000$
- Total: 20.100

Here the first plan is much better

- Why are the estimates of the total size so different?
- In this particular case, we can answer this
- Assume that each relation has two copies of each tuple
- In this case δ reduces the size by a factor of two, exactly like our estimate
- But then $R \times S$ has *four* copies of each tuple, so that δ really reduces the size by a factor of 4
- While we do not study this, one could try to fix this problem by making the selectivity of δ depends on the number of joins beneath it in the tree

Join order

- We show how to select the best join order
- At this point, we consider all possible orders. We shall later see how (and why) we might want to restrict the ones we consider
- We have seen how to pick the best way to join 3 relations
- For $\bowtie (R, S, U)$ we have to chose between 3 possibilities
- For $\bowtie (R, S, U, V)$ we have many more, and the number of choices grows rapidly
- We show how to enumerate all choices, using a technique called *dynamic programming*
- Given m relations, we find, for each i from 1 to m, the best way to evaluate a join of any subset of size i

Example: Four relations

We want to evaluate the join $\bowtie (R(a,b), S(b,c), T(c,d), U(d,a))$. We assume that each has 1.000 tuples

- $V(R,a) = 100$, $V(R,b) = 200$
- $V(S,b) = 100$, $V(S,c) = 500$
- $V(T,c) = 20$, $V(T,d) = 50$
- $V(U,d) = 1.000$, $V(U,a) = 50$

We start with "joins" of single relations. In the tables, we list all combinations of i relations, the size of the results, the cost (of intermediate results) and the best plan. For $i = 1$ this is trivial

	$\{R\}$	$\{S\}$	$\{T\}$	$\{U\}$
Size	1.000	1.000	1.000	1.000
Cost	0	0	0	0
Best Plan	R	S	T	U

Two relations

	$\{R, S\}$	$\{R, T\}$	$\{R, U\}$	$\{S, T\}$	$\{S, U\}$	$\{T, U\}$
Size	5.000	1.000.000	10.000	2.000	1.000.000	1.000
Cost	0	0	0	0	0	0
Best Plan	$R \bowtie S$	$R \bowtie T$	$R \bowtie U$	$S \bowtie T$	$S \bowtie U$	$T \bowtie U$

- Cost is zero as there are no intermediate results
- Best plan is trivial, as there is only one possibility

Three relations

	$\{R,S,T\}$	$\{R,S,U\}$	$\{R,T,U\}$	$\{S,T,U\}$
Size	10.000	50.000	10.000	2.000
Cost	2.000	5.000	1.000	1.000
Best Plan	$(S \bowtie T) \bowtie R$	$(R \bowtie S) \bowtie U$	$(T \bowtie U) \bowtie R$	$(T \bowtie U) \bowtie S$

- Possibilities for $\bowtie (R,S,T)$:
  - $(R \bowtie S) \bowtie T$: intermediate cost $5.000 + 0$
  - $R \bowtie (S \bowtie T)$: intermediate cost $2.000 + 0$
  - $(R \bowtie T) \bowtie S)$: intermediate cost $1.000.000 + 0$
- Intermediate cost is the sum of the size of the intermediate results and its cost
- Best is the second

We now want to join $\bowtie(R, S, T, U)$. This can be done as

- $(\bowtie(R, S, T)) \bowtie U$
- $(\bowtie(R, S, U)) \bowtie T$
- $(\bowtie(R, T, U)) \bowtie S$
- $(\bowtie(S, T, U)) \bowtie R$
- $(\bowtie(R, S)) \bowtie (\bowtie(T, U))$
- $(\bowtie(R, U)) \bowtie (\bowtie(S, T))$

We consider two cases

- $(\bowtie (R, S, T)) \bowtie U$
  - From the previous table $\bowtie (R, S, T)$ has size 10.000, cost 2.000 and best plan $(S \bowtie T) \bowtie R$
  - So our current expression has cost $10.000 + 2.000 = 12.000$ and best plan $((S \bowtie T) \bowtie R) \bowtie U$
- $(\bowtie (R, S)) \bowtie (\bowtie (T, U))$.
  - The plans for the subexpressions are trivial (in the table for 2 relations)
  - The sizes are 5.000 and 1.000, so the cost is 6.000

In a similar way, we get

Grouping	Cost
$((S \bowtie T) \bowtie R) \bowtie U$	12.000
$((R \bowtie S) \bowtie U) \bowtie T$	55.000
$((T \bowtie U) \bowtie R) \bowtie S$	11.000
$((T \bowtie U) \bowtie S) \bowtie R$	3.000
$(T \bowtie U) \bowtie (R \bowtie S)$	6.000
$(R \bowtie T) \bowtie (S \bowtie U)$	2.000.000
$(S \bowtie T) \bowtie (R \bowtie U)$	12.000

So the best plan is $((T \bowtie U) \bowtie S) \bowtie R$ with cost 3.000