Estimating Join Result Size and Example of Logical Plan Optimization

Objective:

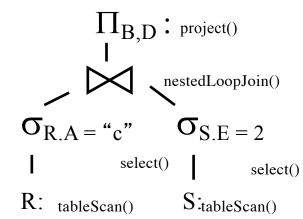
- Probability calculation of two tuples joining

Reading:

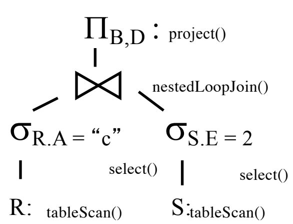
- Ch. 16.4

Adorning an Expression Tree

- Query expression trees
- Adorned with
  - choice of physical operator
  - structural data info. e.g. relation schema
  - statistical information concerning size and data distribution of the arguments
- …



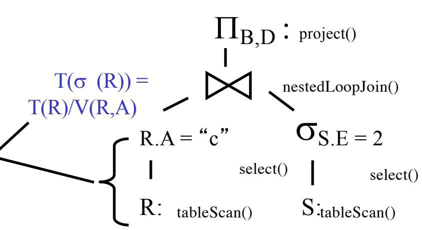
- Cost functions associated with operators connect the costs between nodes of the expression tree.
- Logically correct manipulation of those trees can result in query plans with different execution times.



What about interior operators? Joins in particular

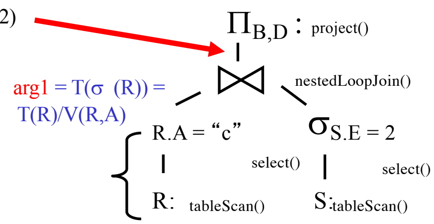
- Cost functions associated with operators connect the costs between nodes of the expression tree.

- Computed I/O cost
- Size of the result



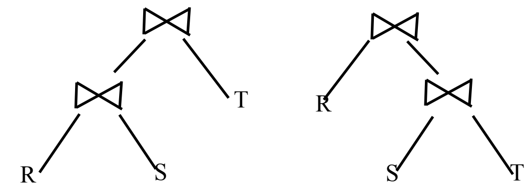
Joins: estimating size of the result

- How big, T(arg1, arg2)



Associativity of Join

- $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$



- Produce the same answer • (necessarily the same size answer)
- The cost of computing that answer can be drastically different
  • (difference must be in the intermediate result)

Join Selectivity

$$\text{join selectivity} = \frac{|R \bowtie S|}{|R \times S|}$$

Joins

$$R \bowtie S$$

Two ways to do this:

1. a lot of formulas and circumstances
2. formulate it as
   - T(R) x T(S) x Prob(r ∈R joins with s ∈ S)

Simplifying Assumptions

- Containment of value sets.
  - If a join attribute/argument, Y, appears in multiple relations, then, for the smallest relationship, S, every value of Y in S appears the other relations.
- Preservation of values
  - Values (columns) that are not subject to predicates retain their statistical values per the catalog

Estimating $R \bowtie_{Y} S$

Let V(R,Y) < V(S,Y)

Prob($r \in$ R joining with $s \in$ S) = 1/V(S,Y)

Estimated size of result =T(R)T(S)/V(S,Y)

// bigger domain in the denominator

This formulation makes determining join results sizes for complex predicates the same as for select predicates.

Example

R(a,b) |X| S(b,c) |X| T(c,d)

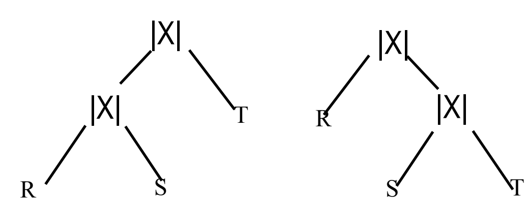
T(R) = T(S) = T(T) = 1000 rows // T of T sorry

V(R,b) = 200 V(S,b) = 100

V(S,c) = 500 V(T,c) = 20

Example

- Which is less expensive to compute?
  - measure will be total number of rows computed



Example

R(a,b) |X| S(b,c) |X| T(c,d)
T(R) = T(S) = T(T) = 1000 rows // T of T sorry

V(R,b) = 200 V(S,b) = 100
V(S,c) = 500 V(T,c) = 20

First, how many rows in, R(a,b) |X| S(b,c)?
or T(R(a,b) |X| S(b,c))

Example

R(a,b) |X| S(b,c) |X| T(c,d)
T(R) = T(S) = T(T) = 1000 rows // T of T sorry

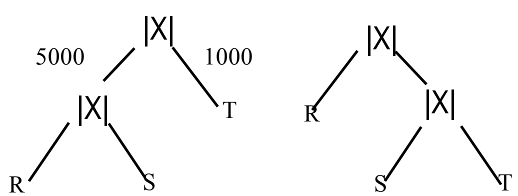
V(R,b) = 200 **V(S,b)** = 100
V(S,c) = 500 V(T,c) = 20

First, how many rows in, R(a,b) |X| S(b,c)?
or T(R(a,b) |X| S(b,c))
= T(R) * T(S) / Max(V(R,b), V(S,b))

= 1000^2 / 200 = 5000

Example

- Which is less expensive to compute?
  - measure will be total number of rows computed



Example

R(a,b) |X| S(b,c) |X| T(c,d)
T(R) = T(S) = T(T) = 1000 rows // T of T sorry

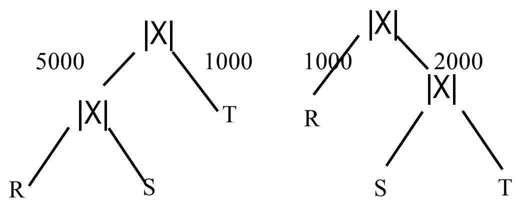
V(R,b) = 200 V(S,b) = 100
V(S,c) = 500 **V(T,c)** = 20

how many rows in, S(b,c) |X| T(c,d)?

= T(S) * T(T) / Max(V(S,c), V(T,c))

= 1000^2 / 500 = 2000

Example



Example

R(a,b) |X| S(b,c) |X| T(c,d)
T(R) = T(S) = T(T) = 1000 rows // T of T sorry

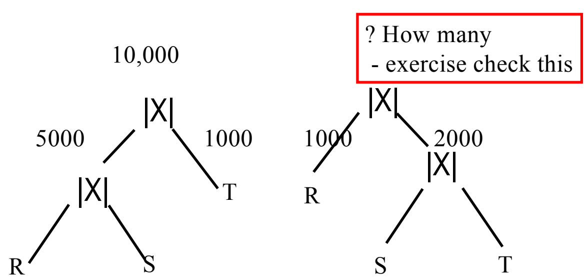
V(R,b) = 200 V(S,b) = 100
V(S,c) = 500 V(T,c) = 20

how many rows in, (R(a,b) |X| S(b,c)) |X| T(c,d) ?

= 5000 * T(T) / Max(V(S,c), V(T,c))

= 5000 * 1000 / 500 = 10,000

Example



Choose the Optimal Join Order

- Just determining the best logical join order in a skewed tree is NP-Hard

What if the join is on a foreign key?

- Let the foreign key in R, be a candidate key of S
- the result size = T(R) // a.k.a semantics-based optimization
- by formula,
  (T(R) * T(S))/ V(S,arg1) // why V(S, _)?
  – Correct, exact, answer (why?); not an estimate