Cost Based Plan Selection

CS157B
Chris Pollett
Mar. 23, 2005.

Outline

- From logical plan to physical plan
- Costs of Operations

Estimating Costs of Operations

- Last day we discussed how to transform a query into a logical query plan.
- From this we can a preferred logical query plan according to the heuristics discussed last day.
- Today, we consider ways of coming up with a physical plan from this logical plan.
- Typically, consider several plans generated from the logical plan, do an estimate of cost for each, and choose the plan of least cost. (*Least cost estimation*)

What to select for each plan we generate.

- An order and grouping on the associative and commutative operations like join, unions, and intersections.
- An algorithm for each operator in the logical plan. For example, need to choose between nested-loop join and hash-join.
- Additional operators - scanning, sorting, etc which are needed for the physical plan but are not present in the logical plan.
- The way in which arguments are passed from one operator to the next. (By intermediate results or by pipelining)

Estimating Sizes of Intermediate Relations

- Without computing the query itself, one cannot exactly determine the number of rows it will return.
- Nevertheless, we'd like to make an estimate of the number of rows which is --
  - Accurate
  - Easy to compute
  - Logically consistent -- the result should not depend on which way we calculate an intermediate result.
- Recall use B(R) -- number of blocks in R, T(R) -- number of tuples in R, V(R, a) -- number of distinct values for attribute a.

Estimating the Size of a Projection

- The number of rows returned by a projection will be the same as the original relation.
- Nevertheless, the space needed to store the relation could be less.
- For example, suppose the block size was 4096, where 96 bytes used for header info. Suppose had a 1,000,000 tuple relation. If tuples went from 40 bytes long to 20 bytes long after a projection. Then the number we could store in a block would go from 100 to 200 and the file would go from 10,000 blocks long to 5,000 blocks long.

Estimating the Size of a Selection

- For $S=\sigma_{A=c}(R)$. We estimate $T(S) = T(R)/V(R,a)$.
- For $S=\sigma_{A<c}(R)$. We estimate $T(S) = T(R)/3$. The somewhat bogus intuition being that people tend to write queries looking for something more selective than half of the tuples.
- For $S=\sigma_{A=/=c}(R)$. We estimate $T(S) = T(R)$. A more accurate but harder to calculate estimate is $T(S) = T(R)(V(R,a) -1)/V(R,a)$.

Estimating the Size of a Join

- Only consider natural joins, such as (R(X,Y) join S(Y,Z)), since other joins can be estimated from natural join and estimate for selections and projections.
- To simplify the task of estimating, we assume (not really true):
  - If V(R,Y) <= V(S,Y) then every Y value of R is a Y value of S.
  - If A is an attribute of R not involved in the join, then V(R join S, A) = V(R,A). Similarly for S.
- Under these assumption, every tuple t of R has a 1/V(S,Y) chance of joining with a tuple of S.
- So the expected number of tuple t will join with is T(S)/V(S,Y).
- So the total size of the join would be T(R)T(S)/V(S,Y).
- To make our estimate symmetric, we estimate:

  T(R join S) = T(R)T(S)/max(V(R,Y),V(S,Y))

Natural Joins with Mutliple Join Attributes

- Suppose we want to join R(x,y1,y2) with S(y1,y2,z).
- We can generalize our previous reasoning to obtain the estimate:

  T(R)T(S)/[max(V(R,y1), V(S,y1))*max(V(R,y2), V(S,y2))]

Joins of Many Relations

- Suppose now we have the join
  S= R1 join R2 join R2 … join Rn.
- To estimate the result we first multiply the size of the relations.
- Then we look at all the attributes A appearing at least twice, divide by all but the least of the V(R,A).
- For example, if had R(a,b,c) join S(b,c,d) join U(b,e), and T(R)=1000, T(S)=2000, and T(U)= 5000. Then would first compute 1000x 2000 x 5000. If V(R,b) =20, V(S,b) = 50, and V(U,b) =200, we would divide by V(S,b) and V(U,b). Finally, we divide by the larger of V(R,c) and V(S,c).