Lecture 27: Optimizations

Wednesday, November 29, 2000

Outline

- Finish dynamic programming (7.6)
- Estimating the Size of Operations (7.4)
- Histograms (7.5.1)
- Completing the physical-query-plan selection (7.7)

Dynamic Programming

- Recall: computes optimal plans for subqueries:
  - Step 1: {R1}, {R2}, …, {Rn}
  - Step 2: {R1, R2}, {R1, R3}, …, {Rn-1, Rn}
  - …
  - Step n: {R1, …, Rn}
- We used naïve size/cost estimations
- In practice:
  - more realistic size/cost estimations (next)
  - heuristics for Reducing the Search Space
    - Restrict to left linear trees
    - Restrict to trees "without cartesian product"
  - need more than just one plan for each subquery:
    - "interesting orders"

Estimating Sizes

- Need size in order to estimate cost
- Example:
  - Cost of partitioned hash-join E1 ⋈ E2 is 3B(E1) + 3B(E2)
  - B(E1) = T(E1)/ block size
  - B(E2) = T(E2)/ block size
  - So, we need to estimate T(E1), T(E2)

Estimating Sizes

Estimating the size of a projection

- Easy: $T(\Pi_L(R)) = T(R)$
- This is because a projection doesn't eliminate duplicates

Estimating Sizes

Estimating the size of a selection

- $S = \sigma_{A=c}(R)$
  - T(S) san be anything from 0 to T(R) – V(R,A) + 1
  - Mean value: T(S) = T(R)/V(R,A)
- $S = \sigma_{A<c}(R)$
  - T(S) can be anything from 0 to T(R)
  - Heuristics: T(S) = T(R)/3

Estimating Sizes

Estimating the size of a natural join, $R \bowtie_A S$

- When the set of A values are disjoint, then $T(R \bowtie_A S) = 0$
- When A is a key in S and a foreign key in R, then $T(R \bowtie_A S) = T(R)$
- When A has a unique value, the same in R and S, then $T(R \bowtie_A S) = T(R)\ T(S)$

Estimating Sizes

Assumptions:

- *Containment of values*: if V(R,A) <= V(S,A), then the set of A values of R is included in the set of A values of S
  - Note: this indeed holds when A is a foreign key in R, and a key in S
- *Preservation of values*: for any other attribute B, $V(R \bowtie_A S, B) = V(R, B)$ (or V(S, B))

Estimating Sizes

Assume V(R,A) <= V(S,A)

- Then each tuple t in R joins *some* tuple(s) in S
  - How many ?
  - On average S/V(S,A)
  - t will contribute S/V(S,A) tuples in $R \bowtie_A S$
- Hence $T(R \bowtie_A S) = T(R)\ T(S) / V(S,A)$

In general: $T(R \bowtie_A S) = T(R)\ T(S) / \max(V(R,A),V(S,A))$

Estimating Sizes

Example:

- T(R) = 10000, T(S) = 20000
- V(R,A) = 100, V(S,A) = 200
- How large is $R \bowtie_A S$?

Answer: $T(R \bowtie_A S) = 10000\ 20000/200 = 1M$

Estimating Sizes

Joins on more than one attribute:

- $T(R \bowtie_{A,B} S) =$

  $T(R)\ T(S)/\max(V(R,A),V(S,A))\max(V(R,B),V(S,B))$

Histograms

- Statistics on data maintained by the RDBMS
- Makes size estimation much more accurate (hence, cost estimations are more accurate)

Histograms

Employee(ssn, name, salary, phone)

- Maintain a histogram on salary:

Salary:	0..20k	20k..40k	40k..60k	60k..80k	80k..100k	> 100k
Tuples	200	800	5000	12000	6500	500

- T(Employee) = 25000, but now we know the distribution

Histograms

Ranks(rankName, salary)

- Estimate the size of Employee $\bowtie_{Salary}$ Ranks

Employee	0..20k	20k..40k	40k..60k	60k..80k	80k..100k	> 100k
	200	800	5000	12000	6500	500

Ranks	0..20k	20k..40k	40k..60k	60k..80k	80k..100k	> 100k
	8	20	40	80	100	2

Histograms

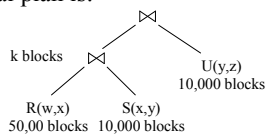
- Assume:
  - V(Employee, Salary) = 200
  - V(Ranks, Salary) = 250
- Then T(Employee $\bowtie_{Salary}$ Ranks) =
  $= \Sigma_{i=1,6} T_i T_i' / 250$
  $= (200x8 + 800x20 + 5000x40 + 12000x80 + 6500x100 + 500x2)/250$
  $= \ldots.$

Completing the Physical Query Plan

- Choose algorithm to implement each operator
  - Need to account for more than cost:
    - How much memory do we have ?
    - Are the input operand(s) sorted ?
- Decide for each intermediate result:
  - To materialize
  - To pipeline

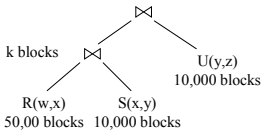
Example 7.38

- Logical plan is:



- Main memory M = 101 buffers

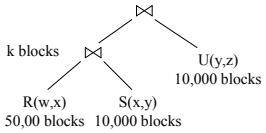
Example 7.38



Naïve evaluation:

- 2 partitioned hash-joins
- Cost 3B(R) + 3B(S) + 4k + 3B(U) = 75000 + 4k

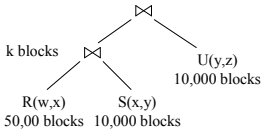
Example 7.38



Smarter:

- Step 1: hash R on x into 100 buckets, each of 50 blocks; to disk
- Step 2: hash S on x into 100 buckets; to disk
- Step 3: read each Ri in memory (50 buffer) join with Si (1 buffer); hash result on y into 50 buckets (50 buffers) -- here we *pipeline*
- Cost so far: 3B(R) + 3B(S)

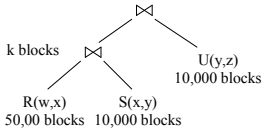
Example 7.38



Continuing:

- How large are the 50 buckets on y ? Answer: k/50.
- If k <= 50 then keep all 50 buckets in Step 3 in memory, then:
- Step 4: read U from disk, hash on y and join with memory
- Total cost: 3B(R) + 3B(S) + B(U) = 55,000

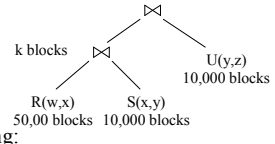
Example 7.38



Continuing:

- If 50 < k <= 5000 then send the 50 buckets in Step 3 to disk
  - Each bucket has size k/50 <= 100
- Step 4: partition U into 50 buckets
- Step 5: read each partition and join in memory
- Total cost: 3B(R) + 3B(S) + 2k + 3B(U) = 75,000 + 2k

Example 7.38



Continuing:

- If k > 5000 then materialize instead of pipeline
- 2 partitioned hash-joins
- Cost 3B(R) + 3B(S) + 4k + 3B(U) = 75000 + 4k

Example 7.38

Summary:

- If k <= 50, cost = 55,000
- If 50 < k <=5000, cost = 75,000 + 2k
- If k > 5000, cost = 75,000 + 4k