Introduction to Database Systems

CSE 444

Lecture #16
March 5, 2001

Query Optimization

Required Reading: 7.2, 7.4, 7.5, 7.6

Query Optimization: Phases

- Parsing phase
  - Produces a parse tree
- Query-Rewrite phase
  - Produces a logical tree
- Physical Query plan generation
  - Produces executable (physical) plan

3

Query Optimization

- Algebraic laws provide alternative execution plans
- Estimate costs of alternative modes of execution
- Efficiently search the space of alternatives
  - Simplify search by applying heuristics (without costing)
    - apply laws that *seem* to result in cheaper plans

4

Converting from SQL to Logical Plans

Select a1, …, an
From R1, …, Rk
Where C

$\Pi_{a1,\dots,an}(\sigma_C(R1 \bowtie R2 \bowtie \dots \bowtie Rk))$

5

Converting from SQL to Logical Plans

Select a1, …, an
From R1, …, Rk
Where C
Group by b1, …, bl

$\Pi_{a1,\dots,an}(\gamma_{b1, \dots, bm, aggs}(\sigma_C(R1 \bowtie R2 \bowtie \dots\dots \bowtie Rk)))$

6

Algebraic Laws

- Commutative and Associative Laws
  - $R \cup S = S \cup R$, $R \cup (S \cup T) = (R \cup S) \cup T$
  - $R \cap S = S \cap R$, $R \cap (S \cap T) = (R \cap S) \cap T$
  - $R \bowtie S = S \bowtie R$, $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
- Distributive Laws
  - $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

7

Algebraic Laws: Selection

- Laws involving selection:
  - $\sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$
  - $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$
  - $\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$
    - When C involves only attributes of R
  - $\sigma_C(R - S) = \sigma_C(R) - S$
  - $\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$
  - $\sigma_C(R \cap S) = \sigma_C(R) \cap S$

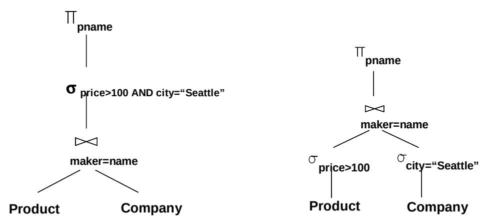
8

Algebraic Laws: Selection

- Example: R(A, B, C, D), S(E, F, G)
  - $\sigma_{F=3}(R \bowtie_{D=E} S) =$?
  - $\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) =$?

9

Heuristic: Predicate Pushdown



The earlier we process selections, less tuples we need to manipulate higher up in the tree (but may cause us to lose an important ordering of the tuples).

10

Algebraic Laws: Projection

- Laws involving projections
  - $\Pi_M(R \bowtie S) = \Pi_N(\Pi_P(R) \bowtie \Pi_Q(S))$
    - Where N, P, Q are appropriate subsets of attributes of M
  - $\Pi_M(\Pi_N(R)) = \Pi_{M,N}(R)$
- Example R(A,B,C,D), S(E, F, G)
  - $\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_?(\Pi_?(R) \bowtie_{D=E} \Pi_?(S))$

11

Other Algebraic Laws

- Duplicate Elimination
  - $\delta(R \bowtie S) = \delta(R) \bowtie \delta(S)$, …
- Grouping
  - $\delta(\gamma_L(R)) = \gamma_{LL}(R)$, …
  - Many transformations depend on aggregate
    - MAX, SUM etc.

12

Cost Estimation

- For a given logical plan, there may be many possible physical plans
- We want to choose physical plan with lowest *execution cost*
- Goal: For a given physical plan, estimate cost **without** executing the query

13

Cost Estimation

- Ideally should be…
  - Accurate
  - Easy to compute
  - Consistent
    - E.g. cardinality should not depend on join order
- Reality …
  - ?

14

Estimating Size of Selection

- How to estimate size of S = $\sigma_{A=20}(R)$?
- Approach 1: Guess!
  - Surprisingly popular method e.g. T(R)/10
- Approach 2: Use *statistics*
  - T(S) = T(R)/V(R,A)

  Where V(R,A) = number of distinct values of A in R
- How about S = $\sigma_{A\leq 20}(R)$?
  - Guess: T(R)/3
  - Statistics: Use *histogram* if available (more later)

15

Estimating Size of Projection

- Projection does not change number of tuples
- Size estimate depends on length of columns
- Example: R(a,b,c): a, b are integers, c string of 100 bytes. Tuple header = 12 bytes, Block size = 1024
- $\pi_{a,b,c}(R)$ = ? $\pi_{a,b}(R)$ = ?
- What if c is variable length string?

16

Estimating Size of Join

- R(a,b), S(b,c), estimate T(R⋈S)
- Problem: Don’t know how values of R.b and S.b are related
  - May be disjoint sets of values => T (R⋈S) = 0
  - S.b may be *key* of S and R.b may be *foreign key* => T(R⋈S) = T(R)
- Estimate for T(R⋈S)
  - T(R)T(S)/max(V(R,b), V(S,b))

17

Estimating Size of Join

- Example: T(R)=1000, T(S)=2000, V(R,b) = 20, V(S,b) = 50
- T(R⋈S) = ?

18

Estimating Size of Join

- What happens if query has multiple join attributes?
  - Example: R(a,b,c), S(b,c,d)
  - Estimate = ?
- What happens if query has joins of many relations?
  - Example: R(a,b), S(b,c), U(b,e)
  - Estimate = ?

19

Estimating Size of Other Operators

- Union (R,S)
  - Bag Union: T(R) + T(S)
  - Set Union: Max(T(R),T(S)) + Min(T(R),T(S))/2
- Intersection (R,S)
  - Min(T(R),T(S))/2
- Difference (R,S)
  - T(R) - T(S)/2
- Duplicate Elimination

20

Cost Based Plan Selection

- Estimates for size parameters
  - Use statistics, e.g. histograms
- Enumerating physical plans

21

Histograms

- Popular in commercial DBMSs
- Can give much more accurate cost estimates
- Many types of histograms
  - Equal-width
  - Equal-depth
  - Frequent values
  - …

22

Equal-width Histogram

- Each bucket in histogram has same width
- Example: Values = {2,5,23,25,29,31}

Range	Count
1-10	2
11-20	0
21-30	3
31-40	1

- $T(\sigma_{A\leq 20}(R)) = 2$

23

Equi-depth Histogram

- Each bucket in histogram has same number of values
- Example: {2,5,33,35,39,41}
  - Bucket Boundary
  - 5
  - 35
  - 41

24

Frequent Values

- Keep exact counts of frequent values
- Total count of all other (non-frequent) values
- Example: Values = {1,3,4,4,4,4,4,9}
- Histogram: 4: 5, Others: 3

25

Using Histogram for Size Estimation

- Example: R(a,b) S(b,c).
- Histograms:
  - R.b: 1:200, 0:150, 5:100, Others:550
  - S.b: 0:100, 1:80, 2:70, Others:250
- Size of join = ?

26

Creating and Maintaining Statistics in a DBMS

- For large tables, creating/refreshing statistics can be expensive
- Alternatives:
  - Refresh statistics only after many changes to data
  - Incremental updating
  - Sampling – need to be careful…

27

Enumerating Physical Plans

- Exhaustive – Consider all possible:
  - Join Orders
  - Algorithms for each operator
- Heuristic Search
  - E.g. Greedy approach
  - Pick next relation such that join size is smallest

28

Enumerating Physical Plans

- Branch-and-Bound Enumeration
  - Find a good starting plan (having cost C)
  - In subsequent search, eliminate any subquery with cost > C
- Hill Climbing
  - Start with heuristically selected plan
  - Explore plans in the "neighborhood"
    - E.g. replace Nested-Loops join with Hash-Join

29

Enumerating Physical Plans

- Dynamic Programming
  - Bottom-up strategy
  - For each subexpression, only keep plan with the least cost
  - Consider possible implementations of each node assuming
  - Extension: also consider *interesting orders*
    - E.g., when subexpression is sorted on a sort attribute at the node
  - More later

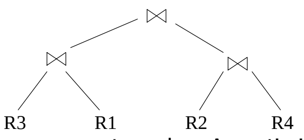
30

Determining Join Order

- Select-project-join
- Push selections down, pull projections up
- Hence: we need to choose the join order
- This is the main focus of an optimizer

Determining Join Order: Join Trees

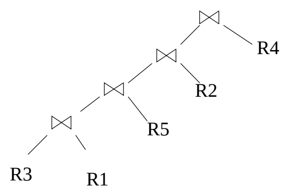
- R1 ⋈ R2 ⋈ …. ⋈ Rn
- Join tree:



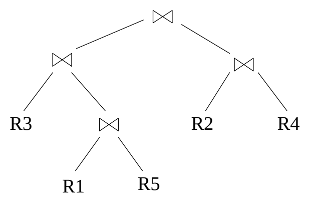
- A join tree represents a plan. An optimizer needs to inspect many (all ?) join trees

Linear Join Trees

- Left deep:



Bushy Join Trees



Join Ordering Problem

- Given: a query R1 ⋈ R2 ⋈ … ⋈ Rn
- Assume we have a function cost() that gives us the cost of every join tree
- Find the best linear join tree for the query

Dynamic Programming

- For each subquery Q□{R1, …, Rn} compute the following:
  - Size(Q)
  - A best plan for Q: Plan(Q)
  - The cost of that plan: Cost(Q)

Dynamic Programming

- Step 1: For each {Ri} do:
  - Size({Ri}) = B(Ri)
  - Plan({Ri}) = Ri
  - Cost({Ri}) = (cost of scanning Ri)

37

Dynamic Programming

- Step i: For each Q□{R1, …, Rn} of cardinality i do:
  - Compute Size(Q)
  - For every pair of subqueries Q', Q"
    s.t. Q = Q' U Q"
    compute cost(Plan(Q') ⋈ Plan(Q"))
  - Cost(Q) = the smallest such cost
  - Plan(Q) = the corresponding plan

38

Dynamic Programming

- Return Plan({R1, …, Rn})

39

Dynamic Programming

To illustrate, we will make the following simplifications:

- Cost(P1 ⋈ P2) = Cost(P1) + Cost(P2) +
  size(intermediate result(s))
- Intermediate results:
  - If P1 = a join, then the size of the intermediate result is size(P1), otherwise the size is 0
  - Similarly for P2
- Cost of a scan = 0

40

Dynamic Programming

- Example:
- Cost(R5 ⋈ R7) = 0 (no intermediate results)
- Cost((R2 ⋈ R1) ⋈ R7)
  = Cost(R2 ⋈ R1) + Cost(R7) + size(R2 ⋈ R1)
  = size(R2 ⋈ R1)

41

Dynamic Programming

- We used naïve size/cost estimations
- In practice:
  - More realistic size/cost estimations
  - 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"

42