CSE 444: Database Internals

Lecture 11
Query Optimization (part 2)

Query Optimizer Overview

- **Input**: A logical query plan
- **Output**: A good physical query plan
- **Basic query optimization algorithm**
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
    - Compute number of I/Os
    - Optionally take into account other resources
  - Choose plan with lowest cost
  - This is called cost-based optimization

Two Types of Plan Enumeration Algorithms

- Dynamic programming (in class)
  - Based on System R (aka Selinger) style optimizer[1979]
  - Limited to joins: *join reordering algorithm*
  - Bottom-up
- Rule-based algorithm (will not discuss)
  - Database of rules (=algebraic laws)
  - Usually: dynamic programming
  - Usually: top-down

Dynamic Programming

Originally proposed in System R [1979]

- Only handles single block queries:

```
SELECT list
FROM   R1, …, Rn
WHERE cond1 AND cond2 AND . . . AND condk
```

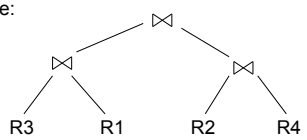
- Some heuristics for search space enumeration:
  - Selections down
  - Projections up
  - Avoid cartesian products

Dynamic Programming

- Search space = join trees
- Algebraic laws = commutativity, associativity
- Algorithm = dynamic programming ☺

Join Trees

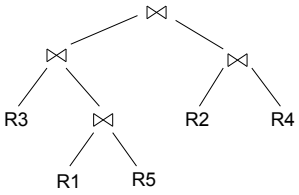
- R1 ⋈ R2 ⋈ …. ⋈ Rn
- Join tree:



- A plan = a join tree
- A partial plan = a subtree of a join tree

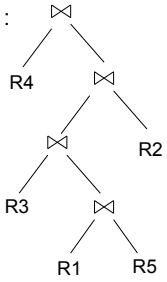
Types of Join Trees

- Bushy:



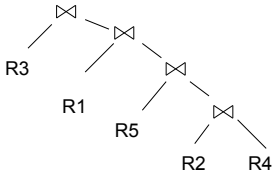
Types of Join Trees

- Linear :



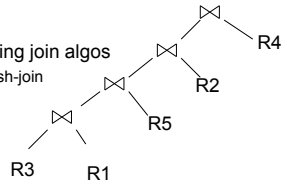
Types of Join Trees

- Right deep:



Types of Join Trees

- Left deep:
  - Work well with existing join algos
    - Nested-loop and hash-join
  - Facilitate pipelining



  - Dynamic programming can be used with all trees

Reducing the Search Space

- Restriction 1: only left linear trees (no bushy)

Why?

- Restriction 2: no trees with cartesian product

R(A,B) ⋈ S(B,C) ⋈ T(C,D)

Plan: (R(A,B)⋈T(C,D)) ⋈ S(B,C)
has a cartesian product.
Most query optimizers will not consider it

Selinger Algorithm

Selinger enumeration algorithm considers

- Different logical and physical plans *at the same time*
- Cost of a plan is IO + CPU
- Concept of *interesting order* during plan enumeration
  - Same order as that requested by ORDER BY or GROUP GY
  - Attributes that appear in equi-join predicates
    - They can speed-up a sort-merge join later

More about the Selinger Algorithm

- Step 1: Enumerate all access paths for a single relation
  - File scan or index scan
  - Keep the cheapest for each *interesting order*
- Step 2: Consider all ways to join two relations
  - Use result from step 1 as the outer relation
  - Consider every other possible relation as inner relation
  - Estimate cost when using sort-merge or nested-loop join
  - Keep the cheapest for each *interesting order*
- Steps 3 and later: Repeat for three relations, etc.

Selinger Algorithm Example

- On the white board
- This example is in the paper