Selinger Optimizer

6.814/6.830 Lecture 10
October 07, 2010

(Slides gently offered by Sam Madden)
The Problem

• How to order a series of N joins, e.g.,
  A.a = B.b AND A.c = D.d AND B.e = C.f

  N! ways to order joins (e.g., ABCD, ACBD, ....)
  (N-1)! plans per ordering (e.g., (((AB)C)D), ((AB)(CD), ...)
  Multiple implementations (e.g., hash, nested loops, etc)

• Naïve approach doesn’t scale, e.g., for 20-way join
  – 10! x 9! = 1.3 x 10 ^ 12
  – 20! x 19! = 2.9 x 10 ^ 35
Selinger Optimizations

- Left-deep only \(((AB)C)D\) (eliminate \((N-1)!!\))
- Push-down selections
- Don’t consider cross products
- Dynamic programming algorithm
Dynamic Programming

$R \leftarrow$ set of relations to join (e.g., ABCD)
For $\partial$ in $\{1 \ldots |R|\}$:
  
  for $S$ in $\{\text{all length } \partial \text{ subsets of } R\}$:
    
    $\text{optjoin}(S) = a \text{ join } (S-a),$
    
    where $a$ is the single relation that minimizes:
      
      cost($\text{optjoin}(S-a)$) +
      
      min. cost to join $(S-a)$ to $a$ +
      
      min. access cost for $a$

$\text{optjoin}(S-a)$ is cached from previous iteration
Example

optjoin(ABCD) – assume all joins are NL

\( d=1 \)

A = best way to access A
  (e.g., sequential scan, or predicate pushdown into index...)

B = best way to access B

C = best way to access C

D = best way to access D

<table>
<thead>
<tr>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subplan</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Total cost computations: \( choose(N,1) \), where N is number of relations
Example

optjoin(ABCD)

$\partial=2$

\{A,B\} = AB or BA
  (using previously computed best way to access A and B)

\{B,C\} = BC or CB
\{C,D\} = CD or DC
\{A,C\} = AC or CA
\{A,D\} = AD or DA
\{B,D\} = BD or DB

Total cost computations: $\text{choose}(N,2) \times 2$
Example

optjoin(ABCD)

∂=3

{A,B,C} = remove A, compare A({B,C}) to (B,C)A
remove B, compare B({A,C}) to (A,C)B
remove C, compare C({A,B}) to (A,B)C

{A,B,D} = remove A, compare A({B,D}) to (B,D)A

....

{A,C,D} = ...
{B,C,D} = ...

Already computed – lookup in cache

Total cost computations: choose(N,3) x 3 x 2

---

<table>
<thead>
<tr>
<th>Subplan</th>
<th>Best choice</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>index</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>seq scan</td>
<td>50</td>
</tr>
<tr>
<td>{A,B}</td>
<td>BA</td>
<td>156</td>
</tr>
<tr>
<td>{B,C}</td>
<td>BC</td>
<td>98</td>
</tr>
<tr>
<td>{A,B,C}</td>
<td>BCA</td>
<td>125</td>
</tr>
<tr>
<td>{B,C,D}</td>
<td>BCD</td>
<td>115</td>
</tr>
</tbody>
</table>
Example

optjoin(ABCD)

\( \delta=4 \)

\{A,B,C,D\} = remove A, compare A \{B,C,D\} to \{B,C,D\}A

remove B, compare B\{A,C,D\} to \{A,C,D\}B

remove C, compare C\{A,B,D\} to \{A,B,D\}C

remove D, compare D\{A,B,C\} to \{A,B,C\}D

Final answer is plan with minimum cost of these four

Total cost computations: \( \text{choose}(N,4) \times 4 \times 2 \)
Complexity

\[ \text{choose}(n,1) + \text{choose}(n,2) + \ldots + \text{choose}(n,n) \text{ total subsets considered} \]

All subsets of a size n set = power set of n = \(2^n\)

Equiv. to computing all binary strings of size n

000, 001, 010, 100, 011, 101, 110, 111

Each bit represents whether an item is in or out of set
Complexity (continued)

For each subset,
  
k ways to remove 1 join
  
k < n

m ways to join 1 relation with remainder

Total cost: \( O(nm2^n) \) plan evaluations
n = 20, m = 2
4.1 \( \times 10^7 \)
Interesting Orders

• Some queries need data in sorted order
  – Some plans produce sorted data (e.g., using an index scan or merge join

• May be non-optimal way to join data, but overall optimal plan
  – Avoids final sort

• In cache, maintain best overall plan, plus best plan for each interesting order

• At end, compare cost of
  
  best plan + sort into order
  
  to
  
  best in order plan

• Increases complexity by factor of k+1, where k is number of interesting orders
Example

SELECT A.f3, B.f2 FROM A,B where A.f3 = B.f4
ORDER BY A.f3

<table>
<thead>
<tr>
<th>Subplan</th>
<th>Best choice</th>
<th>Cost</th>
<th>Best in A.f3 order</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>index</td>
<td>100</td>
<td>index</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>seq scan</td>
<td>50</td>
<td>seqscan</td>
<td>50</td>
</tr>
<tr>
<td>{A,B}</td>
<td>BA hash</td>
<td>156</td>
<td>AB merge</td>
<td>180</td>
</tr>
</tbody>
</table>

compare:
  \[ \text{cost}(\text{sort(output)}) + 156 \]
to
  180