Query Optimization

Exercise Session 6

Andrey Gubichev

November 24, 2014

DPsub

- Iterate over subsets in the integer order
- Before a join tree for S is generated, all the relevant subsets of S must be available

DPsub

DPsub(R)
Input: a set of relations $R = \{R_1, \dots, R_n\}$ to be joined
Output: an optimal bushy join tree
B = an empty DP table $2^R \rightarrow$ join tree
for each $R_i \in R$
  $B[\{R_i\}] = R_i$
for each $1 < i \leq 2^n - 1$ **ascending** $\{$
  $S = \{R_j \in R | (\lfloor i/2^{j-1} \rfloor \, mod \, 2) = 1\}$
  for each $S_1 \subset S, S_2 = S \setminus S_1$ $\{$
    if $\neg$cross products $\wedge \neg S_1$ connected to S_2 **continue**
    $p_1 = B[S_1], p_2 = B[S_2]$
    if $p_1 = \epsilon \vee p_2 = \epsilon$ **continue**
    P = CreateJoinTree(p_1, p_2);
    if $B[S] = \epsilon \vee C(B[S]) > C(P)$ $B[S] = P$
  $\}$
$\}$
return $B[\{R_1, \dots, R_n\}]$

Implementation: DPsize

- dbTable - the vector of lists of Problems, each Problem is either a relation or a join of Problems
- lookup (hashtable) - mapping the set of the relations to the best solution and its cost
- initialize dpTable[0] with the list of R1, ..., Rn
- set the size of dpTable to n

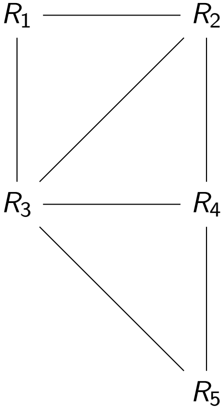
Implementation: DPsize

```
for (i = 1; i < dpTable.size(); i++)
    for (j=0; j < i; j++)
        for (leftRel in dpTable[j])
           for (rightRel in dpTable[i-j-1])
              can we join leftRel and rightRel?
              check lookup for solution and cost
              if the current is cheaper:
                 dpTable[i].add(leftRel join rightRel)
                 update lookup
```

DPccp

- Enumerate over all connected subgraphs
- For each subgraph enumerate all other connected subgraphs that are disjoint but connected to it

Connected Subgraph Enumeration



Connected Subgraph Enumeration

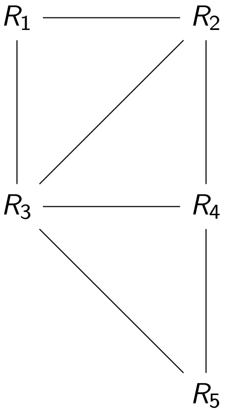
- Nodes in the query graph are ordered according to a BFS
- Start with the last node, all the nodes with smaller ID are forbidden
- At every step: compute neighborhood, get forbidden nodes, enumerate subsets of non-forbidden nodes N
- Recursive calls for subsets of N

Connected Subgraph Enumeration

```
EnumerateCsg(G)
for all i ∈ [n − 1, ..., 0] descending {
    emit {v_i};
    EnumerateCsgRec(G, {v_i}, B_i);
}

EnumerateCsgRec(G, S, X)
N = 𝒩(S) \ X;
for all S' ⊆ N, S' ≠ ∅, enumerate subsets first {
    emit (S ∪ S');
}
for all S' ⊆ N, S' ≠ ∅, enumerate subsets first {
    EnumerateCsgRec(G, (S ∪ S'), (X ∪ N));
}
```

Connected Subgraph Enumeration



Enumerating Complementary Subgraphs

EnumerateCmp(G,S_1)
$X = \mathcal{B}_{\min(S_1)} \cup S_1$;
$N = \mathcal{N}(S_1) \setminus X$;
for all ($v_i \in N$ by descending i) {
    emit $\{v_i\}$;
    EnumerateCsgRec(G, $\{v_i\}$, $X \cup (\mathcal{B}_i \cap N)$);
}

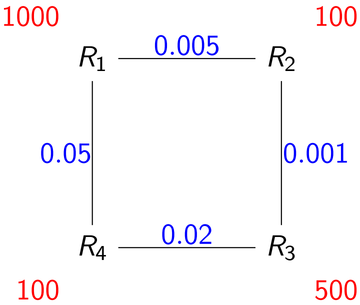
- EnumerateCsg+EnumerateCmp produce all ccp
- resulting algorithm DPccp considers exactly #ccp pairs
- which is the lower bound for all DP enumeration algorithms

Graph simplification

Sometimes the graph is too big, let's simplify it.

- GOO: choose the joins greedily (very hard, depends on all other joins)
- Simplification: choose the joins that must be avoided (we can start with 'obvious' decisions)

Graph simplification: Example



- $benefit(X \bowtie R_1, X \bowtie R_2) = \frac{C((X \bowtie R_1) \bowtie R_2)}{C((X \bowtie R_2) \bowtie R_1)}$
- $R_3 \bowtie R_2$ before $R_3 \bowtie R_4$. Remove $R_4 - R_3$
- $R_4 \bowtie (R_2 \bowtie R_3)$ before $R_4 \bowtie R_1$. Remove $R_1 - R_4$
- no more choices

$|R_1 \bowtie R_4| = 5000$, $|R_1 \bowtie R_2| = 500$, $|R_2 \bowtie R_3| = 50$, $|R_3 \bowtie R_4| = 1000$

More insights

- Guido Moerkotte, Thomas Neumann. Analysis of Two Existing and One New Dynamic Programming Algorithm. In VLDB'06
- Guido Moerkotte, Thomas Neumann. Dynamic Programming Strikes Back. In *SIGMOD'08*
- Thomas Neumann. Query Simplification: Graceful Degradation for Join-Order Optimization. In *SIGMOD'09*

Homework

- 3 problems (See website)
- You will need to revisit some lecture slides for it

Next programming task

- Due December 8
- Dynamic Programming (DPsize)
- We will compare the speed of different submissions :)

Info

- Exercises due: 9 AM, December 1, 2014