Query Optimization
Exercise Session 5

Bernhard Radke

Dezember 05, 2016
Example: Bushy with cross product

\[ R_1 \quad \text{0.5} \quad R_2 \quad \text{0.5} \quad R_3 \]

\[ R_4 \quad \text{0.001} \]

\[ R_1 \quad \text{2} \quad \text{1000} \quad \text{2} \quad \text{500} \]
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 $R = \{R_1, \ldots, R_n\}$ to be joined

Output: an optimal bushy join tree $B = \text{an empty DP table } 2^R \rightarrow \text{join tree}$

for each $R_i \in R$

$B[\{R_i\}] = R_i$

for each $1 < i \leq 2^n - 1$ ascending {

$S = \{R_j \in R | ([i/2^{j-1}] \mod 2) = 1\}$

for each $S_1 \subset S$, $S_2 = S \setminus S_1$

if $\neg\text{cross products } \land \neg S_1 \text{ connected to } S_2$ continue

$p_1 = B[S_1], p_2 = B[S_2]$

if $p_1 = \epsilon \lor p_2 = \epsilon$ continue

$P = \text{CreateJoinTree}(p_1, p_2)$;

if $B[S] = \epsilon \lor C(B[S]) > C(P)$ $B[S] = P$

}

return $B[\{R_1, \ldots, R_n\}]$
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
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

\begin{align*}
\text{EnumerateCmp}(G, S_1) \\
X &= B_{\text{min}}(S_1) \cup S_1; \\
N &= \mathcal{N}(S_1) \setminus X; \\
\text{for all } (v_i \in N \text{ by descending } i) \{ \\
&\quad \text{emit } \{v_i\}; \\
&\quad \text{EnumerateCsgRec}(G, \{v_i\}, X \cup (B_i \cap N)); \\
\}
\end{align*}

- \text{EnumerateCsg} + \text{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

\[
\begin{align*}
\text{benefit}(X \times R_1, X \times R_2) &= \frac{C((X \times R_1) \times R_2)}{C((X \times R_2) \times R_1)} \\
R_3 \times R_2 \text{ before } R_3 \times R_4. \\
\text{Remove } R_4 - R_3 \\
R_4 \times (R_2 \times R_3) \text{ before } R_4 \times R_1. \text{ Remove } R_1 - R_4 \\
\text{no more choices}
\end{align*}
\]

\[|R_1 \times R_4| = 5000, |R_1 \times R_2| = 500, |R_2 \times R_3| = 50, |R_3 \times R_4| = 1000\]
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: Task 1 (10 points)

Create the DP table (manually) for the relations $A$, $B$, $C$ with cardinalities $|A| = 10$, $|B| = 20$, $|C| = 100$ and selectivities $f_{AB} = 0.5$, $f_{BC} = 0.1$ (cost function $C_{out}$). Mark the final table entries. Enumerate subsets in the integer order. Consider cross products.
Homework: Task 2 & 3 (20 points)

- Using the program from the last exercise as basis, implement Greedy Operator Ordering. Print the partial steps together with their costs (e.g., \( P = R_1 \bowtie R_2 \bowtie 200 \), \( Q = P \bowtie R_3 \bowtie 400 \)), as well as the final join tree.

- Load the TPC H data set. (You can use our snapshot of the data set, the loadtpch-* script loads the data). Then, execute the following SQL query using the program implemented above:

```sql
select *
  from lineitem l, orders o, customers c
where l.l_orderkey=o.o_orderkey
  and o.o_custkey=c.c_custkey
  and c.c_name='Customer#000014993'.
```
Exercises due: 9 AM, December 12, 2016