Query Optimization: Exercise

Session 6

Bernhard Radke

November 26, 2018
Lecture Evaluation

- Register for the course in TUMonline
- Evaluation will be done next week in the lecture on December 3
- Bring your laptop
Maximum Value Precedence (MVP) [1]
Weighted Directed Join Graph (WDJG)

Query graph to $WDJG = (V, E_p, E_v)$:
- nodes $V =$ joins
- physical edges $E_p$ between "adjacent" joins (share one relation)
- virtual edges $E_v$ everywhere else
- $\mathcal{R}(v)$: relations participating in join $v$
- Observation: every spanning tree in the WDJG leads to a join tree
Maximum Value Precedence (MVP)  Weighted Directed Join Graph (WDJG)

Query Optimization: Exercise

Session 6  November 26, 2018
Weights and Costs:

- **edge weights:**
  \[ w_{u,v} = \begin{cases} 
  |\mathcal{X}_u| \\
  1 
  \end{cases} \]
  \( |\mathcal{R}(u) \cap \mathcal{R}(v)| \) if \((u, v) \in E_p\)
  \( 1 \) if \((u, v) \in E_v\)

- **node costs:** \( C_{out} \) of the join
Effective Spanning Tree (EST)

Three conditions:

1. EST is binary
2. For every non-leaf node \( v_i \), for every edge \( v_j \rightarrow v_i \) there is a common base relation between \( v_i \) and the subtree with the root \( v_j \)
3. For every node \( v_i = R \ltimes S \) with two incoming edges \( v_k \rightarrow v_i \) and \( v_j \rightarrow v_i \)
   - \( R \) or \( S \) can be present at most in one of the subtrees \( v_k \) or \( v_j \)
   - unless the subtree \( v_j \) (or \( v_k \)) contains both \( R \) and \( S \)
MVP - informally

Construct an EST in two steps:

**Step 1 - Choose an edge to reduce the cost of an expensive operation**

- Start with the most expensive node
- Find the incoming edge that reduces the cost the most
- Add the edge to the EST and check the conditions
- Update the WDJG
- Repeat until
  - no edges can reduce costs anymore or
  - no further nodes to consider

**Step 2 - Find edges causing minimum increase to the result of joins**

- Similar to Step 1
- Start with the cheapest node
- Find the incoming edge that increases the cost the least
Example

We start with a graph without virtual edges.

Two cost lists:

- for the Step 1: $Q_1 = v_{14}, v_{34}, v_{12}, v_{23}$
- for the Step 2: $Q_2 = \emptyset$
Consider \( v_{14} \), select the edge \( v_{12} \rightarrow v_{14} \)

After \( v_{12} \) is executed, \( |R_1 \Join R_2| = 500 \)

Replace \( R_1 \) by \( R_1 \Join R_2 \) in \( v_{14} = R_1 \Join R_4 \): \( v_{14} = (R_1 \Join R_2) \Join R_4 \)

\[
\text{cost}(v_{14}) = 500 \times 100 \times 0.05 + 500 = 3000
\]
Move $v_{12} \rightarrow v_{14}$ to EST
Update WDJG, remove edges $v_{14} \rightarrow v_{12}$ and $v_{12} \rightarrow v_{23}$, add edge $v_{14} \rightarrow v_{23}$

$Q_1 = v_{14}, v_{34}, v_{12}, v_{23}; \ Q_2 = \emptyset$

Consider $v_{14}$, no more incoming edges with $w < 1$

$Q_1 = v_{34}, v_{12}, v_{23}; \ Q_2 = v_{14}$
Consider \( v_{34} \), select the edge \( v_{23} \rightarrow v_{34} \)
Recompute cost: \( \text{cost}(v_{34}) = 50 \times 100 \times 0.02 + 50 = 150 \)
Move to EST, Update WDJG

\[ Q_1 = v_{12}, v_{34}, v_{23}; \quad Q_2 = v_{14} \]
Maximum Value Precedence (MVP)

Example

$Q_1 = v_{12}, v_{34}, v_{23}; \ Q_2 = v_{14}$

Consider $v_{12}$, no edges

$v_{34}, v_{23}$: no incoming edges with $w < 1$

$Q_1 = \emptyset; \ Q_2 = v_{23}, v_{34}, v_{12}, v_{14}$

End of Step 1
Consider $v_{23}$, edge $v_{14} \rightarrow v_{23}$
Adding the edge would not violate EST conditions
Add edge to EST
Done.

$Q_2 = v_{23}, v_{34}, v_{12}, v_{14}$

Consider $v_{23}$, edge $v_{14} \rightarrow v_{23}$
Adding the edge would not violate EST conditions
Add edge to EST
Done.
Dynamic Programming
Overview

- generate optimal join trees bottom up
- start from optimal join trees of size one (relations)
- build larger join trees by (re-)using optimal solutions for smaller sizes
First Approach: DPsizeLinear [2]
- Enumerate increasing in size
- Generate linear trees by adding a single relation at a time

Modifications/Extensions:
- enumerate in integer order
- generate bushy trees by considering all pairs of subproblems
Example
Enumerate connected-subgraph-complement pairs
Query Simplification
Reordering constraints for non-inner joins
Lecture Evaluation

▶ Remember to bring your laptop for the lecture evaluation next week
★ Slides: db.in.tum.de/teaching/ws1819/queryopt
★ Exercise task: gitlab
★ Questions, Comments, etc:
  mattermost @ mattermost.db.in.tum.de/qo18
★ Exercise due: 9 AM next monday