A Development Environment for Query Optimizers

Olga Papaemmanouil, Brandeis University
Nga Tran, Brandeis University
Mitch Cherniack, Brandeis University
One-Design-D-N-F-A

- natural consequence of One-Size-D-N-F-A
- many new niche data management systems
- many new “redesigns” of DBMS components
  - exploiting HW Architecture (e.g., shared-nothing)
  - exploiting SW Architecture (e.g., column-stores)

- needed: Development Environment (DE) for
  - Rapid Prototyping
  - Evaluation
  - Refinement of alternative component designs
Project Overview

- Development Environment for Query Optimizers
  - Rapid Prototyping: Declarative Specifications, Generator Tools
  - Evaluation: Component Benchmarks
  - Refinement: Component Profiling Tools

- Proposal-Ware: |vision| >> |substance|

- Results for Join Plan Enumerator (JPE)
  - SQL Server: better plans with less optimizer work
  - Vertica: Part of version 2.0 Query Optimizer
Talk Outline

- Development Environment Overview
- Initial Results: JPEs
- Future Work: Other Components
Targeted Optimizer Components

Join Plan Selector

1. Chooses Join Order
2. Chooses Join Algorithms and Access Methods
3. Chooses Pre-Join Data Transfer (Distributed DBs)

Join Plan Enumerator
Physical Plan Mapper

JPE  PPM  Pruner
Targeted Optimizer Components

Parser

Join Plan Selector

Execution Engine

Legend

- Fixed
- Logical Plan
- Generated
Targeted Optimizer Components

Parser

Join Plan Selector

JPE

PPM

Pruner

Execution Engine

Legend

Join Graph

Physical Plan

Logical Plan

Fixed

Generated
Targeted Optimizer Components

Join Plan Selector

Legend
- X: Join Graph
- Y: Logical Plan
- Fixed
- Physical Plan
- Generated
Targeted Optimizer Components

Join Plan Selector

- Parser
- Join Graph
- Controller
- Physical Plan
- Execution Engine

Legend:
- × × Y
- Join Graph
- Logical Plan
- Physical Plan
- Generated
- Fixed

Plan:
- LJoin
- PNLJ
- Order
- Cost
- …
Targeted Optimizer Components

Controller

JPE

PPM

Pruner

LJoin

Plan

... 

Property

Order

...
Proposed Development Environment
Talk Outline

- Development Environment Overview
- Initial Results: JPEs
- Future Work: Other Components
Initial Results: JPEs

- Why not universal (exhaustive) enumeration?
  1. Scalability
     - most products include optimization-level “knob” (low = fast)
     - DBA Guide [Tow03]: high knob setting for simple queries only
     - issue of space (not just time)
  2. Relies on Brittle Cost Models
     - Cardinality Errors of Cost Models well-known
     - DBA Guide [Tow03]: low knob setting can improve plans
  3. Lightweight Query Optimizers
     - Useful for adaptive query optimization, automatic DB design

- Therefore, we assume targeted JPEs desirable
**JPEs: Generating**

- **Intuition:**
  - good plans usually perform certain joins before others
  - a good ‘early’ join might ...
    - produce small intermediate results
    - exploit ephemeral properties of inputs (e.g., order)

- **JPEG–Generated JPEs:**
  - specified by “Join ranking function”
    - higher rank = preferable to perform earlier in plan
  - only enumerate plans that perform joins in rank order
JPEs: Generating

- An Example JPEG Enumeration
JPEs: Generating

- Some Example Join Rankings
  - A. Cardinality: 1 = m::1, 2 = m::n
  - B. Order: 1 = Both, 2 = Larger, 3 = Smaller, 4 = Neither
  - C. Indexed: 1 = Either, 2 = Neither
  - D. Size: 1 = At Least One Very Small Table
    2 = Everything Else
    3 = At Least One Very Large Table
  - E. Distribution: 1 = “Both Join Partitioned/One Replicated
    2 = “Larger Partitioned”
    3 = “Smaller Partitioned”
    4 = “Neither Partitioned”
**Results: Replacing the SQL Server JPE**

<table>
<thead>
<tr>
<th>Query (# Tables)</th>
<th>SQL Server</th>
<th>SQL Server + JPEG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chosen Plan (Sec)</td>
<td>DJWUs</td>
</tr>
<tr>
<td>Q2 (5)</td>
<td>13.27</td>
<td>26</td>
</tr>
<tr>
<td>Q3 (3)</td>
<td>267.49</td>
<td>4</td>
</tr>
<tr>
<td>Q5 (6)</td>
<td>332.94</td>
<td>244</td>
</tr>
<tr>
<td>Q7 (6)</td>
<td>883.74</td>
<td>137</td>
</tr>
<tr>
<td>Q8 (6)</td>
<td>304.08</td>
<td>507</td>
</tr>
<tr>
<td>Q9 (6)</td>
<td>457.52</td>
<td>255</td>
</tr>
<tr>
<td>Q10 (4)</td>
<td>329.62</td>
<td>11</td>
</tr>
<tr>
<td>Q11 (3)</td>
<td>40.70</td>
<td>4</td>
</tr>
<tr>
<td>Q18 (3)</td>
<td>1392.86</td>
<td>4</td>
</tr>
<tr>
<td>Q21 (4)</td>
<td>259.47</td>
<td>11</td>
</tr>
</tbody>
</table>

10 Queries Based on TPC-H, Scale Factor 10

**DWJU = “Distinct Join Work Unit”**

Measure of Optimizer Work
(# of distinct logical joins that are costed)
**JPEs: Generating**

- **Results: Replacing the SQL Server JPE**

<table>
<thead>
<tr>
<th>Query (# Tables)</th>
<th>SQL Server</th>
<th>SQL Server + JPEG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chosen Plan (Sec)</td>
<td>DJWUs</td>
</tr>
<tr>
<td>Q2 (5)</td>
<td>13.27</td>
<td>26</td>
</tr>
<tr>
<td>Q3 (3)</td>
<td>267.49</td>
<td>4</td>
</tr>
<tr>
<td>Q5 (6)</td>
<td>332.94</td>
<td>244</td>
</tr>
<tr>
<td>Q7 (6)</td>
<td>883.74</td>
<td>137</td>
</tr>
<tr>
<td>Q8 (8)</td>
<td>304.08</td>
<td>507</td>
</tr>
<tr>
<td>Q9 (6)</td>
<td>457.52</td>
<td>255</td>
</tr>
<tr>
<td>Q10 (4)</td>
<td>329.62</td>
<td>11</td>
</tr>
<tr>
<td>Q11 (3)</td>
<td>40.70</td>
<td>4</td>
</tr>
<tr>
<td>Q18 (3)</td>
<td>1392.86</td>
<td>4</td>
</tr>
<tr>
<td>Q21 (4)</td>
<td>259.47</td>
<td>11</td>
</tr>
</tbody>
</table>

- Better Plan, Less Work
- Same Plan, Less Work
- Same Plan, Same Work
JPEs: Benchmarking

- How to benchmark JPE independently of cost model?
  - Best Runtime of Enumerated Plans
  - Worst Runtime of Enumerated Plans
  - Average Runtime of Enumerated Plans
  - Number of Enumerated Plans

- Issues:
  - Runtimes depend on execution engine
  - How to reconcile # of Plans vs Quality of Plans
JPEs: Profiling Tools

- Primary JPE Tuning Strategies:
  - Merging/Splitting Join Ranks
    - Generates more/fewer enumerated plans
  - Reorder Join Ranks
    - Generates different enumerated plans

- Profiling Tools to Guide Above:
  - Rank Prevalence: What % of joins have rank i?
  - Merge Impact: What is % increase in enumerated plans resulting from merging ranks i and j?
  - Reorder Impact: What is effect on benchmark metrics from swapping ranks i and j?
Talk Outline

- Development Environment Overview
- Initial Results: JPEs
- Future Work: Other Components
Pruner Tools: Early Thoughts

- Possible Specification:
  - $>\!\!\!: \text{Plan} \times \text{Plan} \to \text{Boolean}$
  - $\text{Ps}$: Set of “interesting” properties

- Possible Benchmarks:
  - $>\text{-quality}$: % of times predicted better plan is actually better
  - $P\text{-quality}$: % of plans identified as best that would have been pruned with property $P$ identified as interesting
Controller Tools: Early Thoughts

- Possible Specification
  - Top-Down vs Bottom-Up Traversal
  - Handoff Policy: conditions for transferring control from JPE to PPM/Pruner

- Possible Benchmarks ($C_{eager}$ vs $C_{lazy}$):
  - **Eager-Benefit**: % of subplans pruned earlier by $C_{eager}$ than by $C_{lazy}$ (assuming pruned by both)
  - **Eager-Penalty**: % of subplans pruned by $C_{eager}$ but not by $C_{lazy}$
Conclusions

- Early-Stage Project:
  Development Environment for Join Plan Selection Components of Query Optimizers

- Will Support Per Component:
  - Rapid Prototyping
  - Declarative Specification, Generator Tool
  - Evaluation Component Benchmarks
  - Refinement Profiling Tools

- Early Results for JPEs:
  - JPEG: Enumerate According to “Join Ranking Functions”
  - Used to build JPEs for SQL Server, Vertica
Questions