The Volcano Optimizer Generator: Extensibility and Efficient Search

Original Slides by:
Presentation: Alfred Pang
Discussion: Kati Radkhah
Modified by:
Rachel Pottinger

The Volcano Optimizer Generator

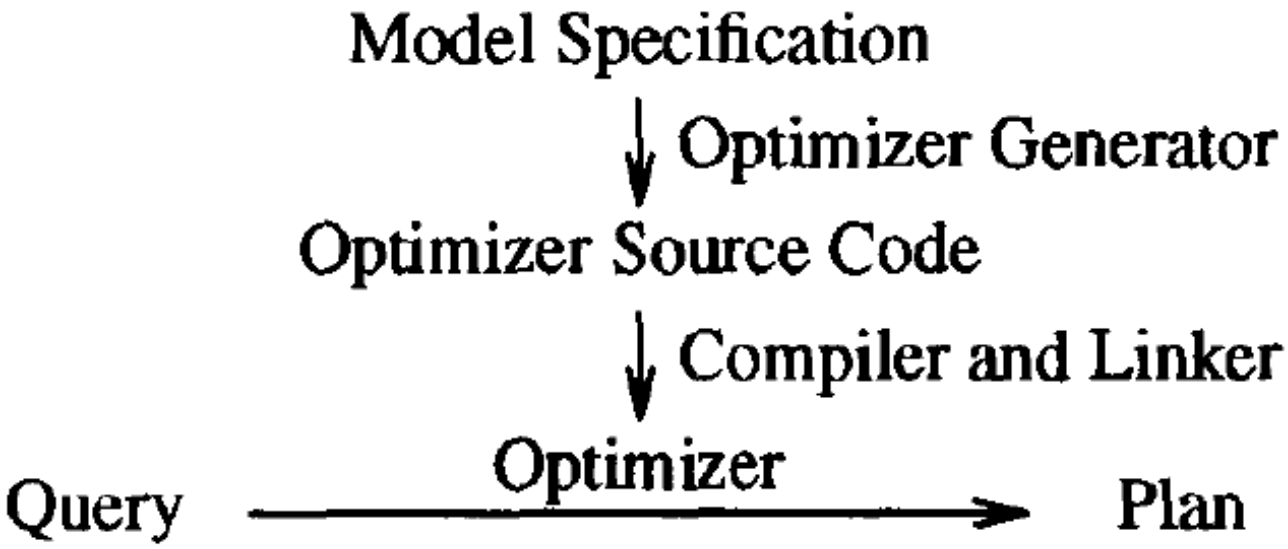
- Object-oriented and scientific database systems
- Allowing query optimization to be more tuned towards the application = higher performance
- (Expert) User optimize

Discussion:

- Many of you noticed that Volcano allowed for different data models other than just relational, but claimed to work for any. When writing a paper, you can either be generic w.r.t. established choices vs. trends or stick to one paradigm for the system/algorithm/whatever. What are the benefits and drawbacks of being more or less general?

- If you over generalize, you miss everything
- If you're too specific, the industry might move, and you miss
- 2% of the data is the most popular, so even if you generalize, it's good enough. You can have a more generic approach in part of the paper. It's better to say that you're going for one chunk, but go after it and have a good reason
- It depends on the maturity of the area. Start, generalized, then more specific
- Compare with commercial software. Has a ton of features that almost no one uses, but but can put a lot of resources to only one area, can be a lot of development in one area. The organization does the large piece, but develop in the small.
- Not really a choice. Based on the nature of the insights.

The Generator Paradigm



Optimizer Generator

- This is not the first time for this approach (EXODUS).
- Volcano improves on the work of EXODUS: ease of use, expressiveness.

Volcano Requirements

- Useable as standalone tool
- Efficient
- Support physical properties
- Expressive – heuristics, directed search, cost functions

Design Principles

- Relational algebra (logical and physical), especially to support OO
- Rules-based => modularization
- Map queries to same algebraic equiv as Volcano's input
- Rule compilation rather than interpretation
- Dynamic programming

General Optimizer: Input/Output

- Input: User Query => Logical algebra expression
- Output: Algorithms to access physical storage => Physical algebra expression

Volcano: Input/Output

- Input:
  - Set of logical operators
  - Algebraic transformation rules (logical -> physical)
  - Algorithms and enforcers
  - Implementation rules (operators to algorithms)
  - Cost functions
  - Applicability function for each algorithm and enforcer
  - Etc.
- Output: Generated optimizer

Volcano Plan Search Engine

- Search engine is same for all generated optimizers
- Directed dynamic programming; goal-oriented (driven by needs rather than by possibilities)
- Find costs of promising moves (transform, algorithm, or enforcer)

Volcano Plan Search Engine

- EXODUS did not consider logical expressions together with physical properties in optimization cost. (Volcano does)
- In OO systems, this can be used to more properly cost access of complex objects.
- Volcano algorithm is top-down (lower levels are explored only when warranted).

Comparison to Starburst

- Starburst has a hierarchy of intermediate levels; harder to see interactions. Volcano uses an algebraic approach which paper claims to be easier to understand.

Comparison to Starburst

- Query rewrites in Starburst do not include cost estimates. (Heuristic)
- Although paper is critical of this, Volcano does allow for heuristic transformations to be specified.

How good was it?

- Comparison between Volcano and EXODUS.
- Example used a small data model, consisting of relational select and join operators only.
- As similar data model descriptions as possible were specified for Volcano and EXODUS.

How good was it?

- Volcano took less time to optimize.
- EXODUS optimizer generator measurements were quite volatile and took a lot of memory.
- EXODUS's generated optimizer and search engine do not explore and exploit physical properties and interesting orderings.

Summary

- Tools not just relational databases, but also object-oriented and scientific databases.
- Extensibility using optimizer generator.
- Separation of logical and physical algebras.

Summary

- When and how to use heuristic transforms vs. cost-sensitive optimizations
- Physical properties considered throughout the optimization, rather than considered after all logical transforms.

Discussion

Nalin: What is considered Software Engineering, and what constitutes research? Some plumbing code is required for research, but what is the actual difference between developing features vs solving a research problem? Has the distinction become more hazy over the years?