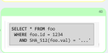
16-optimizer2

- 概念
  - Object-oriented implementation of the Volcano query optimizer.
  - Simplistic expression re-writing can be through a direct mapping function rather than an exhaustive search.
  - Optimization tasks as data structures.
  - Rules to place property enforcers.
  - Ordering of moves by promise.
  - Predicates as logical/physical operators.
  - expression
    - A ***expression*** is an operator with zero or more input expressions.
    - Logical Expression:
      - $(A \bowtie B) \bowtie C$
    - Physical Expression:
      - $(A_x \bowtie_{alg1} B_y) \bowtie_{alg2} C_z$
  - group
    - A ***group*** is a set of logically equivalent logical and physical expressions that produce the same output
      - All logical forms of an expression
      - All physical expressions that can be derived from selecting the allowable physical operators for the corresponding logical forms
      -
      -
  - multi-expression
    - Instead of explicitly instantiating all possible expressions in a group, the optimizer implicitly represents redundant expressions in a group as a ***multi-expression***.
      - This reduces the number of transformations, storage overhead, and repeated cost estimations.
      -
  - rule
    - A ***rule*** is a transformation of an expression to a logically equivalent expression.
      - ***Transformation Rule***: Logical to Logical
      - ***Implementation Rule***: Logical to Physical
    - Each rule is represented as a pair of attributes
      - ***Pattern***: Defines the structure of the logical expression that can be applied to the rule.
      - ***Substitute***: Defines the structure of the result after applying the rule.
      -
  - Memo table
    - Stores all previously explored alternatives in a compact graph structure.
    - Equivalent operator trees and their corresponding plans are stored together in groups.
    - Provides memoization, duplicate detection, and property + cost management.
    -
- Cascades / Columbia
- Principle of Optimality
  - Every sub-plan of an optimal plan is itself optimal.
  - This allows the optimizer to restrict the search space to a smaller set of expressions
    - The optimizer never has to consider a plan containing sub-plan **P1** that has a greater cost than equivalent plan **P2** with the same physical properties.
- Search Termination
  - Approach #1: Wall-clock Time
    - Stop after the optimizer runs for some length of time.
  - Approach #2: Cost Threshold
    - Stop when the optimizer finds a plan that has a lower cost than some threshold.
  - Approach #3: Transformation Exhaustion
    - Stop when there are no more ways to transform the target plan. Usually done per group.
- 实现
  - Standalone
    - Wisconsin OPT++ (1990s)
    - Portland State Columbia (1990s)
    - Pivotal Orca (2010s)
    - Apache Calcite (2010s)
  - Integrated:
    - Microsoft SQL Server (1990s)
    - Tandem NonStop SQL (1990s)
    - Clustrix (2000s)
    - CMU Peloton (2010s)

- Predicate
  - Predicates are defined as part of each operator
    - These are typically represented as an AST.
    - Postgres implements them as flatten lists.
  - The same logical operator can be represented in multiple physical operators using variations of the same expression.
  - Predicate Pushdown
    - Approach #1: Logical Transformation
      - Like any other transformation rule in Cascades
      - Can use cost-model to determine benefit.
    - Approach #2: Rewrite Phase
      - Perform pushdown before starting search using an initial rewrite phase. Tricky to support complex predicates
    - Approach #3: Late Binding
      - Perform pushdown ***after*** generating optimal plan in Cascades. Will likely produce a bad plan.
  - Predicate Migration
    - : Not all predicates cost the same to evaluate on tuples.
    - 

    - The optimizer should consider selectivity and computation cost when determining the evaluation order of predicates.
- Orca Optimizer
  - Pivotal Orca
    - Standalone Cascades implementation.
      - Originally written for Greenplum.
      - Extended to support HAWQ.
    - A DBMS can use Orca by implementing API to send catalog + stats + logical plans and then retrieve physical plans.
    - Supports multi-threaded search.
  - Issue #1: Remote Debugging
    - Automatically dump the state of the optimizer (with inputs) whenever an error occurs
    - The dump is enough to put the optimizer back in the exact same state later on for further debugging.
  - Issue #2: Optimizer Accuracy
    - Automatically check whether the ordering of the estimate cost of two plans matches their actual execution cost.
- MemSQL Optimizer
  - Rewriter
    - Logical-to-logical transformations with access to the cost-model.
  - Enumerator
    - Logical-to-physical transformations.
    - Mostly join ordering.
  - Planner
    - Convert physical plans back to SQL.
    - Contains MemSQL-specific commands for moving data.
  -