19 – Query Optimizier Overview

1. Optimizing

   A. Finding correct query that has lowest model

   B. Proven to be NP-Complete
      So, Optimizer does not always give optimal solution

2. Background

   A. Logical Plan VS. Physical Plan
      Optimizer generates a mapping of a logical algebra expression to the optimal equivalent physical algebra expression
      ex) access table -> access table with B+Tree index……,
          choice of join operators…

   B. Search ARGument ABLE
      DBMS can take advantage of an index to speed up the execution of the query.
      usually, OLTP queries are sargable. Because they doesn't have table access that much.

   C. Cost estimation
      generate an estimate of the cost of executing a plan for the current state fo the DB

      i. Interactions with other work in DBMS

      ii. Size of result set, data properties

      iii. Choice of algorithms, access methods

      iv. Resource utilization

3. Implementation Design Decisions

   A. Optimization granularity

      i. Single query
         small search space, no reuse result across queries
         just consider what is currently running

      ii. Multiple queries
         large search space, useful for data/intermediate result sharing
         efficient if there are many similar queries

   B. Optimization timing

      i. Static optimization
         optimize before execution

      ii. Dynamic optimization
         optimize subsequently with execution

      iii. Adaptive optimization
         optimize before execution,
         if performance is lower than threshold, re-optimize query plan and re-execute query.

   C. Prepared statement (re executing same/similar query)

      i. Reuse last plan

      ii. Re-optimize

      iii. Multiple plans

      iv. Average plan

D. Plan stability

i. Hints
DBA can provide hints to the optimizer.(some constraints on optimizing)

ii. Fixed optimizer versions
set the optimizer version number and migrate queries one-by-one to the new optimizer

iii. Backwards compatible plans
save query plan from old version and provide it to the new DBMS

E. Search termination

i. Wall-clock time
stop optimizer at some length of time

ii. Cost threshold
stop optimizer when it finds a plan that has lower cost than some threshold

iii. Exhaustion
stop when there are no mor enumerations of the target plan.

4. Optimizer Search Strategies

A. Heuristics

i. rules

1. Perform most restrictive selection early

2. Perform all selections before joins

3. Join ordering based on cardinality

ii. Pros & cons

1. Easy to implement and debug, works reasonably well.

2. If operators have complex inter-dependencies, it'll not work.

B. Heuristics + cost-based join order search

   i. Rules

      1. Use static rules to perform initial optimization

      2. Use dynamic programming to determine the best join order

   ii. Pros & cons

      1. Can find a reasonable plan without an exhaustive search

      2. Same problem with heuristic only approach

      3. Left-deep join trees are not always optimal

C. Randomized algorithms

   i. Rules

      1. Select random query plan until cost threshold is reached

      2. Simulated annealing : start with initial query plan which is generated by heuristic only approach, do random change of operator and accept when cost is reduced.

   ii. Pros & Cons

      1. can avoid local minimum trap

      2. low memory overhead

      3. difficult to determine why the DBMS may have chosen a plan

D. Stratified search

   i. Rules

      1. Rewrite the logical query plan using transformation rules

      2. Then perform a cost-based search to map the logical plan to a physical plan

   ii. Pros & cons

      1. Works well in practice

      2. Difficult to assign priorities to transformations

      3. Rules maintenance is big pain

E. Unified search

   i. Rules

      1. Unify the notion of logical-logical and logical-physical transformation

      2. Use memoization to reduce redundant work
      since it generates many transformations