Introduction to Query Optimization

Chapter 13

Overview of Query Optimization

- **Plan**: Tree of R.A. ops, with choice of alg for each op.
  - Each operator typically implemented using a 'pull' interface; when an operator is 'pulled' for the next output tuples, it 'pulls' on its inputs and computes them.
- **Two main issues**:
  - For a given query, what plans are considered?
  - Algorithm to search plan space for cheapest (estimated) plan.
- **Ideally**: Want to find best plan. Practically: Avoid worst plans!
- We will study the System R approach.

Highlights of System R Optimizer

- **Impact**:
  - Most widely used currently; works well for < 10 joins.
- **Cost estimation**: Approximate art at best.
  - Statistics, maintained in system catalogs, used to estimate cost of operations and result sizes.
  - Considers combination of CPU and I/O costs.
- **Plan Space**: Too large, must be pruned.
  - Only the space of left-deep plans is considered.
  - Left-deep plans allow output of each operator to be pipelined into the next operator without storing it in a temporary variable.
- Cartesian products avoided.

Motivating Example

- **RA Tree**:
  - $\pi_{name}$ (On-the-fly)
  - $\pi_{bid=10}$ (On-the-fly)
  - $\pi_{rating>5}$ (On-the-fly)
  - $\sigma_{bid=100} \land rating>5$ (On-the-fly)
  - $\sigma_{bid=100} \land rating>5$ (On-the-fly)

- Cost: 500+500*100 1/Os
- By no means the worst plan!
- Misses several opportunities: selections could have been pushed earlier, no use is made of any available indexes, etc.
- **Goal of optimization**: To find more efficient plans that compute the same answer.

Schema for Examples

- Reserves (sid: integer, name: string, rating: integer, age, real)
- Sailors (sid: integer, bid: integer, day: dates, name: string)

- Similar to old schema; **name** added for variations

- Reserves:
  - Each tuple is 40 bytes long. 100 tuples per page, 1000 pages.
- Sailors:
  - Each tuple is 50 bytes long. 80 tuples per page, 500 pages.

Alternative Plans 1 (No Indexes)

- **Main difference**: with sorts.
- With 5 buffers, cost of plan:
  - Scan Reserves (1000) + write temp T1 (10 pages, if we have 10 buffers, uniform distribution).
  - Scan Sailors (1000) + write temp T2 (20 pages, if we have 10 buffers).
  - Sort T1 (2*10), sort T2 (2*20), merge (10+20)
  - Total: 300 i/Os per page.
- If we used BNL join, join cost = 10+4*250, total cost = 2770.
- If we 'push' projections T1 has only sid, T2 only sid and name.
  - T1 has in 3 pages, cost of BNL drops to under 250 pages, total < 300.
**Alternative Plans 2**
**With Indexes**
- With clustered index on sid of Reserves, we get 100,000 tuples on 100/100 = 10 pages.
- JNL with peeking (outer is not materialized).
- -Projecting out unnecessary fields from outer doesn’t help.
- Join column sid is a key for Sailors.
  - At most one matching tuple, unclustered index on sid OK.
- Decision not to push 
  - Cost: Selection of Reserves tuples (10 I/Os); for each, must get matching Sailors tuple (100 I/Os); total 1200 I/Os.

**Statistics and Catalogs**
- Need information about the relations and indexes involved. Catalogs typically contain at least:
  - # tuples (Ntuples) and # pages (Npages) for each relation.
  - # distinct key values (NKeys) and NPages for each index.
  - Index height, low/high key values (Low/High) for each tree index.
- Catalogs updated periodically.
  - Updating whenever data changes is too expensive; lots of approximation anyway, so slight inconsistency OK.
- More detailed information (e.g., histograms of the values in some field) are sometimes stored.

**Cost Estimation**
- For each plan considered, must estimate cost:
  - Must estimate cost of each operation in plan tree.
  - Depends on input cardinalities.
  - We’ve already discussed how to estimate the cost of operations (sequential scan, index scan, join, etc.)
  - Must estimate size of result for each operation in tree!
  - Use information about the input relations.
  - For selections and joins, assume independence of predicates.
- We’ll discuss the System R cost estimation approach.
  - Very inexact, but works ok in practice.
  - More sophisticated techniques known now.

**Size Estimation and Reduction Factors**

**Summary**
- Query optimization is an important task in a relational DBMS.
- Must understand optimization in order to understand the performance impact of a given database design (relations, indexes) on a workload (set of queries).
- Two parts to optimizing a query:
  - Consider a set of alternative plans.
    - Must prune search space; typically, left-deep plans only.
    - Must estimate cost of each plan that is considered.
    - Must estimate size of result and cost for each plan node.
    - Key: access Statistics, indexes, operator implementations.