Query Planning: Part 2 (Joins)

Overview

- Query usually divided in to SELECT query blocks for optimization (there are exceptions)
- For each block we consider:
  - each relation in FROM clause: all available access paths
  - [a hopefully good subset of] all join trees (considering all permutations and applicable join methods)

Enumeration of Alternative Plans

- Some strategy is needed to reduce the search space
- Exhaustive search possible for small number of relations
  - Heuristics are needed to reduce search space
- Estimate the cost of each plan
  - estimate cost of each operator
    - depends on input cardinalities, sorting
  - estimate the size of result

Join Plans - Heuristics

- System R (IBM) approach
  - Think of join plans as operator trees
  - Consider only “left-deep” plans
    - each join has at least one “base table”
    - joining one relation at a time in any order
- Avoid cross (cartesian) products

Example

- SELECT *
  FROM t0, t1, t2, t3
  WHERE t0.pk = t1.fk AND t1.pk = t2.fk1 AND t2.fk2 = t3.pk
  AND t0.c = 100 AND t3.e = 200;

- t0 is 10K rows, t0.c = 100 has 1/100 selectivity=>t0r is 100 rows
- t1 is 1M rows
- t2 is 10M rows
- t3 is 100K rows, t3.e = 200 has 1/200 selectivity => t3r is 500 rows

Pass 1

- t0 is 10K rows, t0.c = 100 has 1/100 selectivity=>t0r is 100 rows.
  - form t0r with best available access path
    - compare Index access (if available) to table Scan
- t1 is 1M rows
- t2 is 10M rows
- t3 is 100K rows, t3.e = 200 has 1/200 selectivity => t3r is 500 rows
  - form t3r with best available access path
    - compare Index access (if available) to table scan

Pass 2 - 2 table joins

- Six combinations - commit to the best way of joining pair
- t0 join t1 result size = 10000; cost = t0r + t1 = 1,000,100
- t0 join t2 result size = 1000000000; cost = t0r + t2 = 10,000,100
- t0 join t3 result size = 50000; cost = t0r + t3r = 600
- t1 join t2 result size = 10000000; cost = t1 + t2 = 1,1000,000
- t1 join t3 result size = 500000000; cost = t1 + t3r = 1,000,500
- t2 join t3 result size = 50000; cost = t2+ t3r = 10,000,500

Pass 3 - 3 table joins

- t0 join t1 join t2 result rows = 100000; cost = t0 join t1 cost + read result + read t2 cost = 11,010,100
  - t0 join t2 join t1 result rows = 100000; cost 1011000100
  - t1 join t2 join t0 result rows = 100000; cost 21000100
- t0 join t1 join t3 result rows = 5000000; cost = t0 join t1 cost + read result + read t3 cost = 1,010,600
  - t0 join t3 join t1 result rows = 5000000; cost 1050600
  - t1 join t3 join t0 result rows = 5000000; cost 501000600
- t0 join t3 join t2 result rows = 5000000; cost = t0 join t3 cost + read result + read t2 cost = 10,050,600
  - t2 join t3 join t0 result rows = 5000000; cost 10050600
  - t0 join t2 join t3 result rows = 5000000; cost 1010000600
- t2 join t3 join t1 result rows = 50000; cost = t2 join t3 cost + read result + read t1 cost = 11,050,500
  - t1 join t2 join t3 result rows = 50000; cost 21000500
  - t1 join t3 join t2 result rows = 50000; cost 511000500

4th Pass - 500 result rows

- t0 join t1 join t2 (100K) join t3
  - cost = 11,010,100 + read result 100K + read t3 (500) = 11,110,600
- t0 join t1 join t3 (5M) join t2
  - cost = 1,010,600 + read result 5M + read t2 (10M) = 16,010,600
- t0 join t3 join t2 (5M) join t1
  - cost = 10,050,600 + read result 5M + read t1 (1M) = 16,050,600
- t2 join t3 join t1 (50K) join t0
  - cost = 11,050,500 + read result 50K + read t0 (100) = 11,100,600

System R - left deep plans

- Enumerated using N passes (if N relations joined):
  - Pass 1: Find best 1-relation plan for each relation.
  - Pass 2: Find best way to join result of each 1-relation plan to another relation. (All 2-relation plans.)
  - Pass N: Find best way to join result of a (N-1)-relation plan to the N'th relation. (All N-relation plans.)
- For each subset of relations, retain only:
  - Cheapest plan overall, plus
  - Cheapest plan for each *interesting order* of the tuples.
    - sort-order information is useful for group-by, order-by and even joins
- Cartesian (cross) products considered at the end

Another Approach

- Evaluate (exhaustively) all N-way joins for the least costly join
  - If there are M tables: $\binom{M}{N}$ such combinations
    - Choose the least expensive of them
- Commit the least-costly join of the least costly N-way join.
- Replace the two source relations used in the join with their resulting joined relation.
- Loop through the process until only one relation remains.

all 3 way joins

- t0 join t1 join t2 result rows = 100000; cost = t0 join t1 cost + read result + read t2 cost = 11,010,100
  - t0 join t2 join t1 result rows = 100000; cost 1011000100
  - t1 join t2 join t0 result rows = 100000; cost 21000100
- t0 join t1 join t3 result rows = 5000000; cost = t0 join t1 cost + read result + read t3 cost = 1,010,600
  - t0 join t3 join t1 result rows = 5000000; cost 1050600
  - t1 join t3 join t0 result rows = 5000000; cost 501000600
- t0 join t3 join t2 result rows = 5000000; cost = t0 join t3 cost + read result + read t2 cost = 10,050,600
  - t2 join t3 join t0 result rows = 5000000; cost 10050600
  - t0 join t2 join t3 result rows = 5000000; cost 1010000600
- t2 join t3 join t1 result rows = 50000; cost = t2 join t3 cost + read result + read t1 cost = 11,050,500
  - t1 join t2 join t3 result rows = 50000; cost 21000500
  - t1 join t3 join t2 result rows = 50000; cost 511000500

Next Iter:(t0 join t1) join t2 join t3

- ((t0 join t1) join t2) join t3
  - cost 11,110,600
- (t0 join t1) join (t2 join t3)
  - cost 11,060,600
- ((t0 join t1) join t3) join t2
  - cost 16,010,600