21 – Query Optimizer Implementation

1. Bad query plan

    A. Common problem :
    incorrect join orderings because of inaccurate cardinality estimation

    B. Reason of plans getting worse than expectation :
    estimations are based on static summarizations
    can be solved by adaptive query optimization

2. Adaptive query optimization

    A. Modify the execution behavior of a query by generating multiple plans for it

    B. Use information collected during query execution.
    (affected by previous query runs)

    C. Approach

        i. Modify future invocation

        ii. Replan current invocations

        iii. Plan pivot points

3. Reversion-Based Plan Correction

    A. Check new plan to previously executed plan
    if new plan is not better than previous one, change query plan with previous one.

    B. Can recover from wrong estimated cost because comparison of query plans are done by actual cost.
    even if first estimation goes wrong, future invocation won't suffer by this estimation

4. Plan Stitching
    A. Using sub-plan of previously executed plan to build plan to compare with new plan
    B. Use heuristics to identify equivalent sub-plans
    C. Use OR-operators to indicate alternative paths through the plan
       AND-OR tree will be DAG structure
    D. Perform bottom-up search to select the cheapest sub-plan for each OR nodes.
    E. Redshift
       DBMS caches subplans(compiled one), combines these at runtime for new queries. (to avoid the compilation cost)
5. Quickstep
6. Parametric optimization
7. Proactive re-optimization