Query Optimization

Search Strategies: *Heuristic*

- **Heuristic Selection**
  - Approach
    - Make a sequence of choices for annotations based on heuristics.
    - E.g., A greedy algorithm for join ordering.
  - Only one plan is generated!

- This was the researchers's first idea.
- However, it did not work well.
- It is not *cost based*, and can pick expensive plans.

Query Optimization

Search Strategies: *Exhaustive*

- **Exhaustive Search**
  - Approach
    1. Price all possible physical query plans derivable from the logical query plan.
    2. Choose the least expensive.
  - Enumeration of plans (plan generator)
    - top-down
    - bottom-up

- This was the researchers's second idea.
- This is *cost based*, which is good.
- However, it is too expensive!
  The search space is much too large.

Query Optimization

Search Strategies: *Branch-and-Bound*

- **Branch-and-Bound**
  - Approach
    - Find an initial plan, usually by *heuristic selection*.
    - Consider different plans until
      - a plan cheaper than a threshold cost is found, or
      - time runs out.
  - Must be coupled with a plan generator.

- Dynamically prunes the search space.
- Was thought to be too expensive in the past.

Query Optimization

Search Strategies: *Hill Climbing*

- **Hill Climbing**
  - Approach
    - Find an initial plan, usually by *heuristic selection*.
    - Sequentially make local modifications to the plan that reduce the estimated cost.
    - When no cost reducing moves are left, halt.
  - Explores the plan space locally around the initial plan.

- Dynamically walks the search space.
- Was thought to be too expensive in the past.
- May be a reasonable approach now.

Query Optimization

Search Strategies: *Dynamic Programming*

- **Dynamic Programming**
  - Approach: A bottom-up enumeration
    - Price all possible plans for each initial sub-query.
      For each sub-query, keep the least expensive plan.
    - Move up the tree.
      Generate all possible plans for each sub-query, assuming the best plans already chosen for each of its sub-queries.
    - Finished once at the root of the tree.
  - Prunes the search space dynamically.

- Can control the search via dynamic programming.
- May need some compromises to prune the search space.

Query Optimization

Search Strategies: *System R*

- **Selinger-Style Optimization / System R**
  - Approach: Revision of *dynamic programming*
    - Price all possible plans for each initial sub-query.
      For each sub-query, keep several plans:
      - the least expensive plan, and
      - for each interesting order, the least expensive plan of those that preserve that *interesting order*.
    - Move up the tree.
    - Finished once at the root of the tree.
  - Prunes the search space dynamically.

- This is what made relational database systems feasible.
- Still is the basis of most commercial systems's optimizers.

Cost-based Optimizer

Buffer Pool Issues

- We have been very conscious of the impact of the number of buffer pool pages available for given operations (e.g., SMJ).
  How does the optimizer deal with this?

- **Assumption:** As many buffer pool pages as needed.
  - Each operation will be priced based on an assumption of a *reasonable*, given number of buffer pages.
  - There is not assumed a bound on the given number of buffer pages for the entire query.
  - The optimizer must vet plans so not “too many” buffer pages will be in concurrent use, which would lead to thrashing.

The Query Optimizer

Two Stage

Often, an optimizer is built as two-stage:

1. **The Rewrite Optimizer**
   - Produces a logical query plan from the SQL.
     logical query plan $=$ unannotated query tree
   - May *rewrite* this tree many times to arrive at the final.
   - Like the *heuristic selection* approach.
2. **The Cost-based Optimizer**
   - ***What we have been studying.***