Query Optimization Discussion

Murat Kantarcioglu

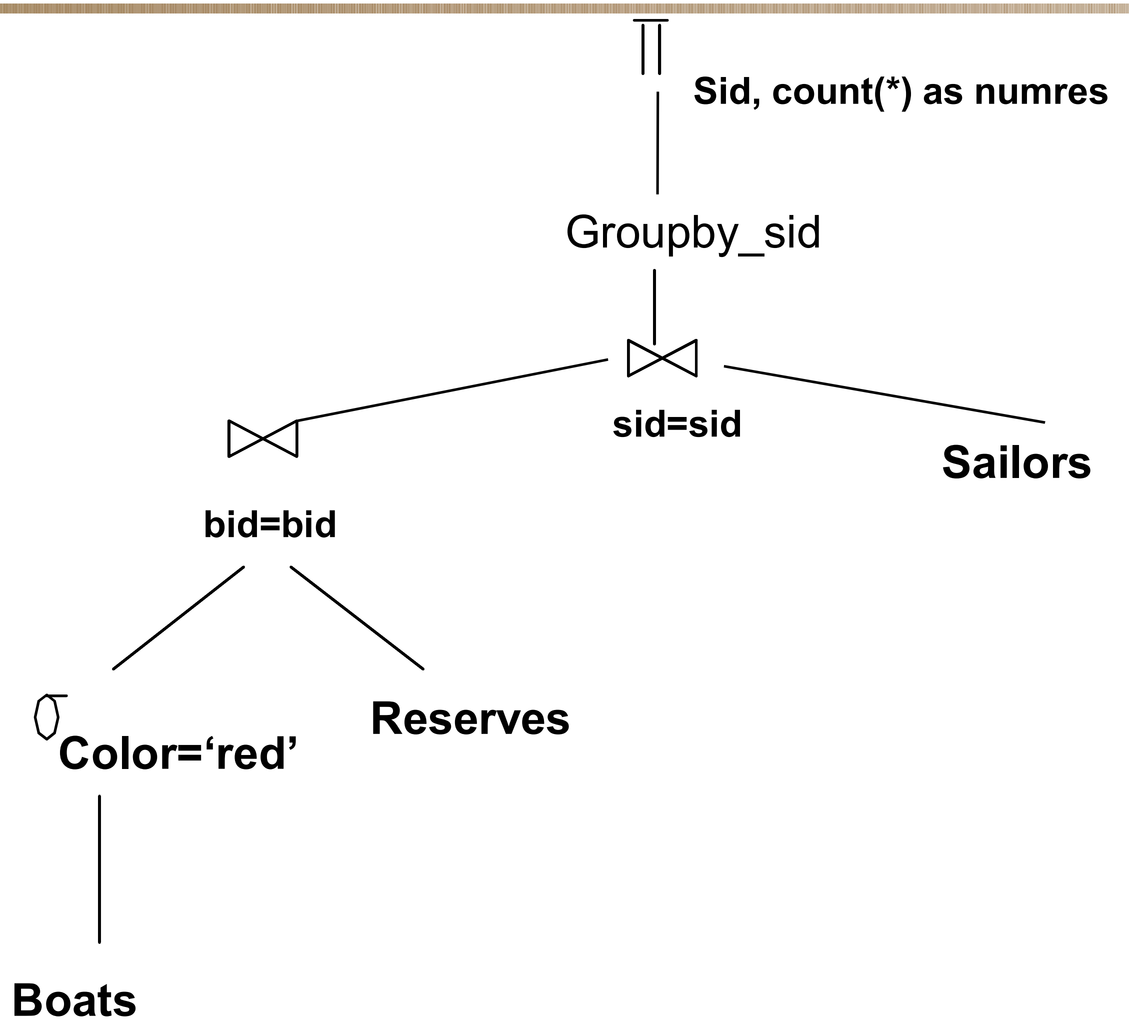
Join Order Optimization Algorithm (General Case)

```
procedure findbestplan(S)
    if (bestplan[S].cost $\neq \infty$)
        return bestplan[S]
    // else bestplan[S] has not been computed earlier, compute it now
    if (S contains only 1 relation)
        set bestplan[S].plan and bestplan[S].cost based on the best way
        of accessing S  /* Using selections on S and indices on S */
    else for each non-empty subset S1 of S such that S1 $\neq$ S
        P1= findbestplan(S1)
        P2= findbestplan(S - S1)
        A = best algorithm for joining results of P1 and P2
        cost = P1.cost + P2.cost + cost of A
        if cost < bestplan[S].cost
                bestplan[S].cost = cost
                bestplan[S].plan = “execute P1.plan; execute P2.plan;
                                        join results of P1 and P2 using A”
    return bestplan[S]
```

Example

- Select S.bid, Count(*) As numres
  From Boats B, Reserves R, Sailors S
  Where R.sid=S.sid and B.bid=R.bid and
  B.color=‘red’
  Group by sid
- Suppose (these are chosen to make discussion easier)
  - Reserves have B+ tree on sid, clustered B+ on bid
  - Sailors B+ tree and hash index on sid
  - Boats B+ tree and hash index on color

Example


Sid, count(*) as numres
Groupby_sid
sid=sid
Sailors
bid=bid
Color=‘red’
Reserves
Boats

Pass 1

- For reserves and sailors best option is file scan
- We can use Hash index on boats to get boats with matching color.

Pass 2 and Pass 3

- Pass 2: Consider each pair of joins and the every join method available and all the access paths for the inner.
- Pass 3: For each pair of tables considered in Pass 2, consider the remaining one as the inner one.
- Keep interesting ones such as sorted orders for the group by.
- If the results found after pass 3 is not sorted, add sorting cost.

Heuristic Optimization

- Cost-based optimization is expensive, even with dynamic programming. Systems may use *heuristics* to reduce the number of choices that must be made in a cost-based fashion.
- Heuristic optimization transforms the query-tree by using a set of rules that typically (but not in all cases) improve execution performance:
  - Perform selection early
  - Perform projection early
  - Perform most restrictive selection and join operations (i.e. with smallest result size) before other similar operations.

Structure of Query Optimizers

- Many optimizers considers only left-deep join orders.
  - Plus heuristics to push selections and projections down the query tree
  - Reduces optimization complexity and generates plans amenable to pipelined evaluation.
- Heuristic optimization used in some versions of Oracle:
  - Repeatedly pick "best" relation to join next
    - Starting from each of n starting points. Pick best among these
- Intricacies of SQL complicate query optimization
  - E.g. nested subqueries

Structure of Query Optimizers (Cont.)

- Some query optimizers integrate heuristic selection and the generation of alternative access plans.
  - Frequently used approach
    - heuristic rewriting of nested block structure and aggregation
    - followed by cost-based join-order optimization for each block
  - Some optimizers (e.g. SQL Server) apply transformations to entire query and do not depend on block structure
- Even with the use of heuristics, cost-based query optimization imposes a substantial overhead.
  - But is worth it for expensive queries
  - Optimizers often use simple heuristics for very cheap queries, and perform exhaustive enumeration for more expensive queries