Overview of Query Optimization in Relational Systems

Original slides by
Presenter: Albert Wong
Discussion: Stephen Ingram
Modified by Rachel Pottinger

Overview of Query Optimization in Relational Systems

- SQL query provides a way for the user to interface with the DB
- SQL query must be translated into machine code that can efficiently find the desired information (*Query Evaluation)*
- Optimizing and executing the SQL query is key to performance and ensuring that relational databases are feasible
- Overview of the methods/problems involved in finding and choosing an execution path for a given SQL query

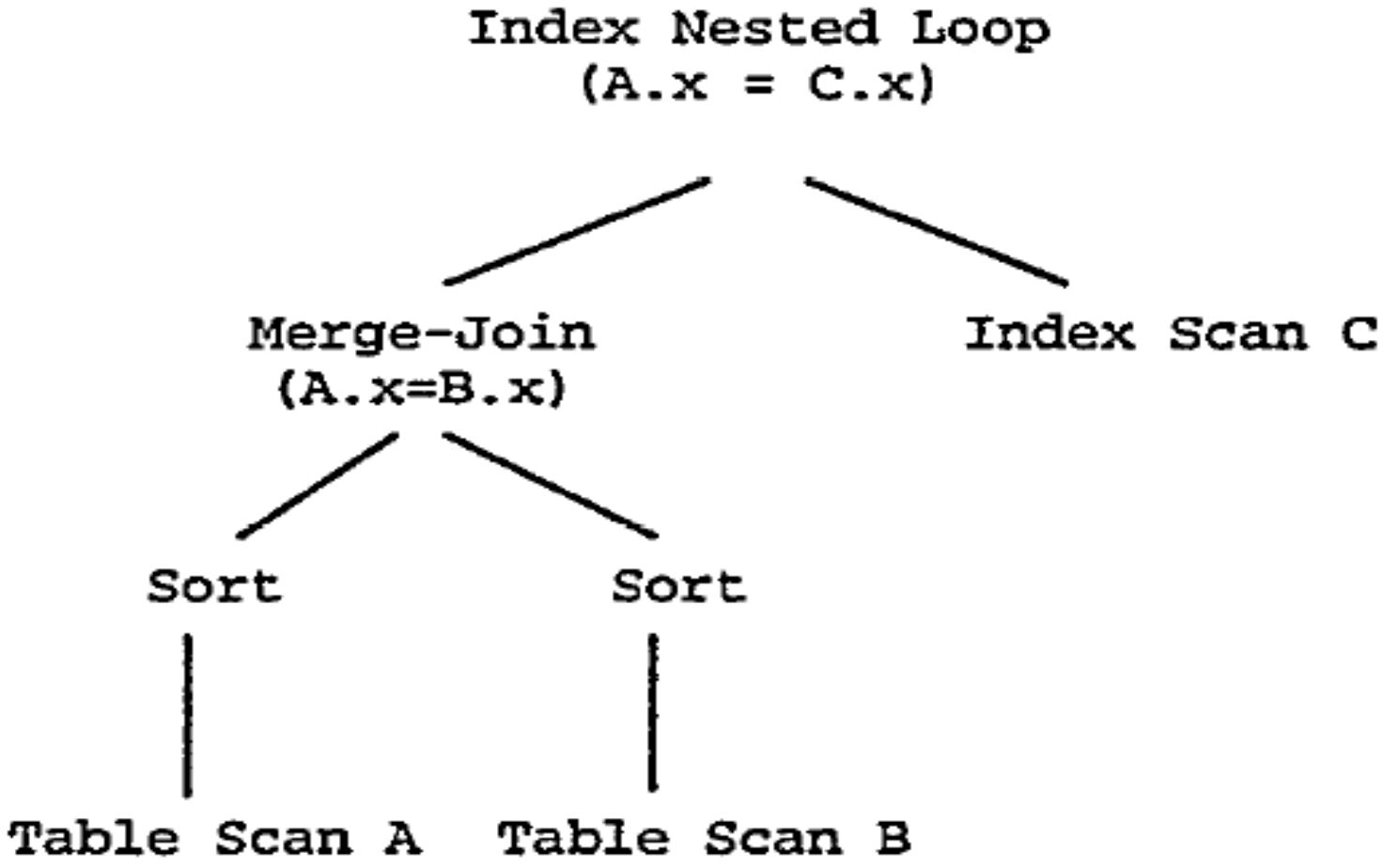
Introduction

- 2 key components for query evaluation in a SQL database system
  - Query optimizer
  - Query execution engine

Query Execution Engine

- Implements a set of physical operators
- A physical operator takes as input one or more data streams and produces an output data stream and propagate any physical properties (e.g. order)
  - Ex. (external) sort, sequential scan, index scan, nested loop join, sort-merge join
  - pieces of code used as building blocks to execute SQL queries
  - responsible for execution of operator tree (execution plan) that generates answers to the query

Example Operator Tree



Query Optimizer

- Input: parsed representation of SQL query
- Output: an efficient execution plan for the given SQL query from the space of possible execution plans
  - Input to Query Execution Engine

The Key Idea: Query Optimization as a Search Problem

- To solve problem, we need to provide:
  - Search space
  - Cost estimation technique to assign a cost to each plan in the search space
  - Enumeration algorithm to search through the execution space

- Search for the best (or not the worst) plan

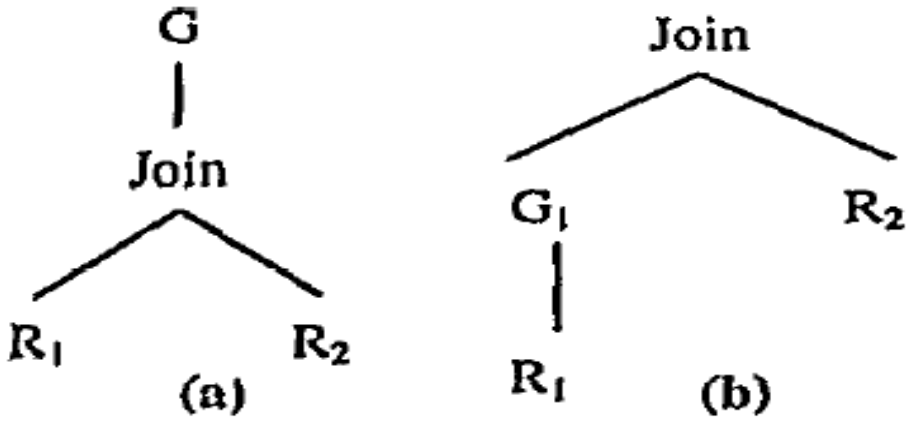
Search Space

- Depends on:
  - Equivalence performing algebraic transformations
  - Physical operators supported in an optimizer
- Transformations may *not* reduce cost and therefore must be applied in a cost-based manner to ensure a positive benefit

- Can’t explore *all* options

Commuting Between Operators

- Join operators are commutative and associative
- Join operators do not have to be linear, but result in bushy join trees
- Outer Join and Join
  - Join(R, S LOJ T) = Join(R, S) LOJ T
- Group-By and Join



A Hairy Discussion

- Does the formulation of a query affect the execution of that query? Can users optimize their queries' execution through better syntax?
- Why do you think most systems mainly focused on linear join sequences instead of bushy joins?

Multi-Block Query to Single-Block

- Merging Views
  - Q = Join(R,V)
  - View V = Join(S,T)
  - Q = Join(R,Join(S,T)
- Merging Nested Subqueries

```
SELECT Emp.Name
FROM Emp
WHERE Emp.Dept#  IN
       SELECT Dept.Dept#  FROM Dept
       WHERE Dept.Loc='Denver'
       AND Emp.Emp# = Dept.Mgr
```

```
SELECT E.Name
FROM Emp E, Dept D
WHERE E.Dept# = D.Dept#
AND D.Loc = 'Denver' AND E.Emp# = D.Mgr
```

Statistics and Cost Estimation

- Cost estimation must be accurate because optimization is only as good as its cost estimates
- Must be efficient as it is repeatedly invoked by the optimizer
- Basic estimation framework
  - collect statistical summaries of data stored
  - given an operator and statistical summaries of its input streams, determine
    - statistical summary of output data stream
    - estimated cost of executing the operation

Statistical Summaries of Data

- Ex.: # tuples in table, # physical pages used by table, statistical information on columns (e.g., histograms)
- Can use sampling to determine histograms that are accurate for a large class of queries
- Statistics must be propagated from base data to be useful
  - Can be difficult as assumptions must be made when propagating statistical summaries

Cost Computation

- Costs:
  - CPU
  - I/O
  - communication costs (parallel & distributed)
- Difficult to determine best cost estimator
- Statistical summary propagation and accurate cost estimation are difficult open issues in query optimization

A Cost Discussion

- Propagation of statistical information and cost estimation are crucial steps in query optimization where accuracy plays an important role. But inaccuracy can result in both steps. Why do you think it still works out?

Enumeration Architectures

- Enumeration algorithm explores search space to pick cheap execution plan
- Enumerators concentrate on *linear join sequences* rather than *bushy join sequences* due to the size of the search space including bushy join sequences

Extensible Optimizers

- Want enumerator to adapt to changes in search space
  - New transformations
  - Addition of new physical operators
  - Changes in cost estimation techniques
- Solutions:
  - Use generalized cost functions and physical properties with operator nodes
  - Use rule engine that allows transformations to modify the query expression or the operator trees
  - Expose “knobs” to tune behavior of system
  - Ex. Starburst and Volcano/Cascades (coming up)

Materialized Views

- Views cached by database system
- Query can take advantage of materialized views to reduce the cost of executing the query
- Problems
  - Reformulating query to take advantage of materialized views (general problem is undecidable)
  - Determining effective sufficient conditions is nontrivial

Summary of Chaudhuri's Paper

- Query optimization as a search problem whose solution requires:
  - a search space
  - cost estimation technique,
  - an enumeration algorithm
- Query optimization can be considered an art
- No one knows what the best execution plan for a given query is

Ending Discussions

- This overview gives us some perspective on what was kept from System-R and what wasn't. Are you surprised on how much was kept, or not? Do you expect much change in the years since printing of this paper?
- This all seems like a bit of a black art. And yet it largely works. Does this surprise you? Why or why not?
- What other areas of computer science is this most like? Could it benefit from ideas from other areas? How?

Ending Discussions

- This overview gives us some perspective on what was kept from System-R and what wasn't. Are you surprised on how much was kept, or not? Do you expect much change in the years since printing of this paper?
- This all seems like a bit of a black art. And yet it largely works. Does this surprise you? Why or why not?
- What other areas of computer science is this most like? Could it benefit from ideas from other areas? How?