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

- An overview of current query optimization techniques
- Gives fundamentals of query optimization

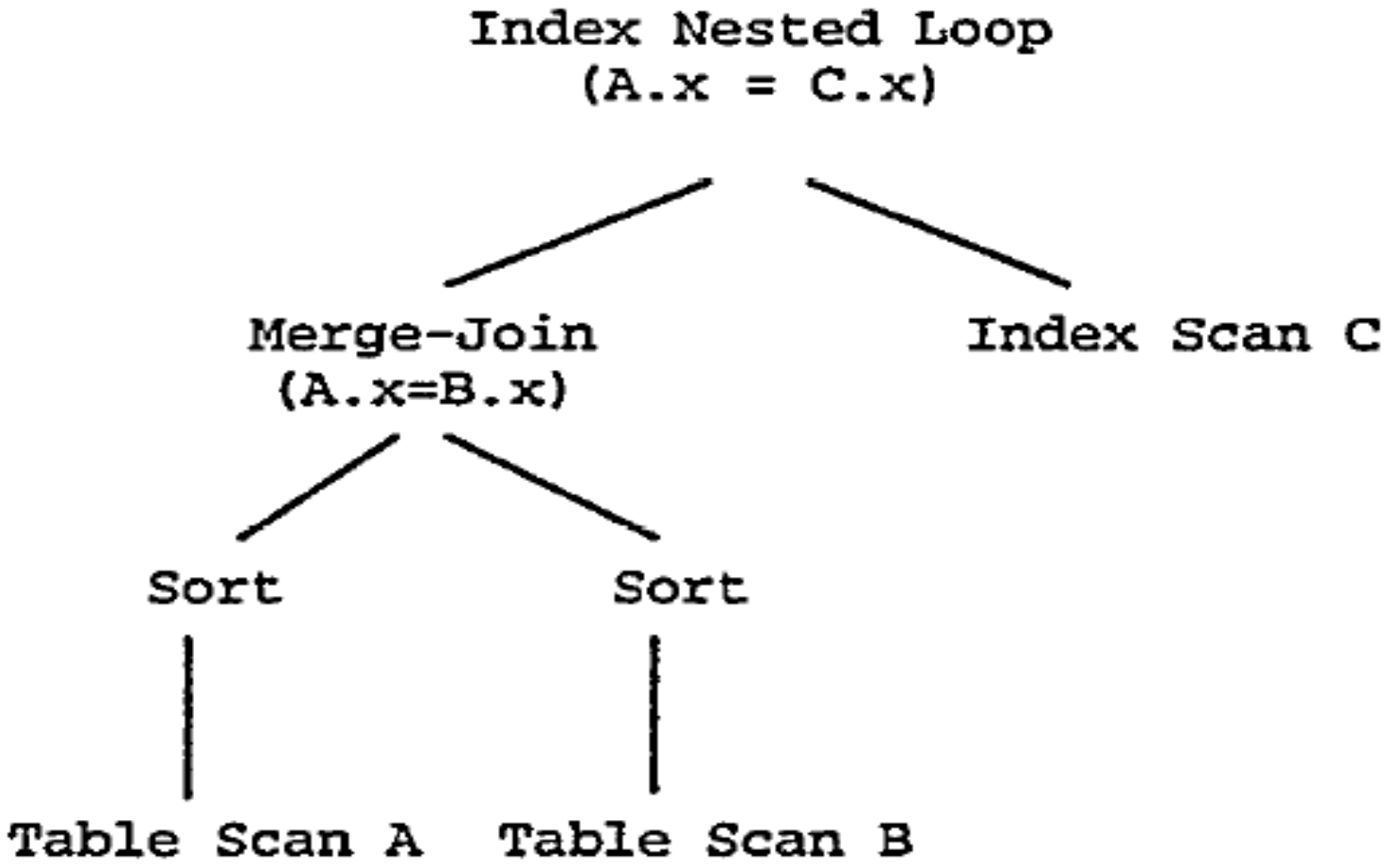
Introduction

- 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
  - 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

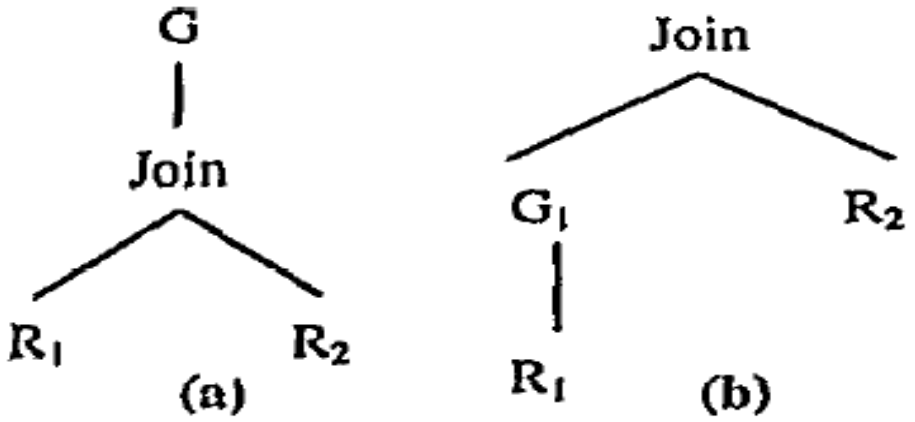
Search Space

- 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

- Generalized Join Sequencing
- Outer Join and Join
  - Join(R, S LOJ T) = Join(R, S) LOJ T
- Group-By and Join



Quick Discussion

This all seems like a bit of a black art. And yet it largely works. Does this surprise you? Why or why not?

Pair and then come back together.

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
  - estimating distinct values is provably error prone
- Statistics must be propagated from base data to be useful
  - Can be difficult as assumptions must be made when propagating statistical summaries

A Statistical Discussion

- Some estimated statistics are provably erroneous. Is it then worth estimating? If so, what sort of strategy should we adopt when using estimates with known problems?

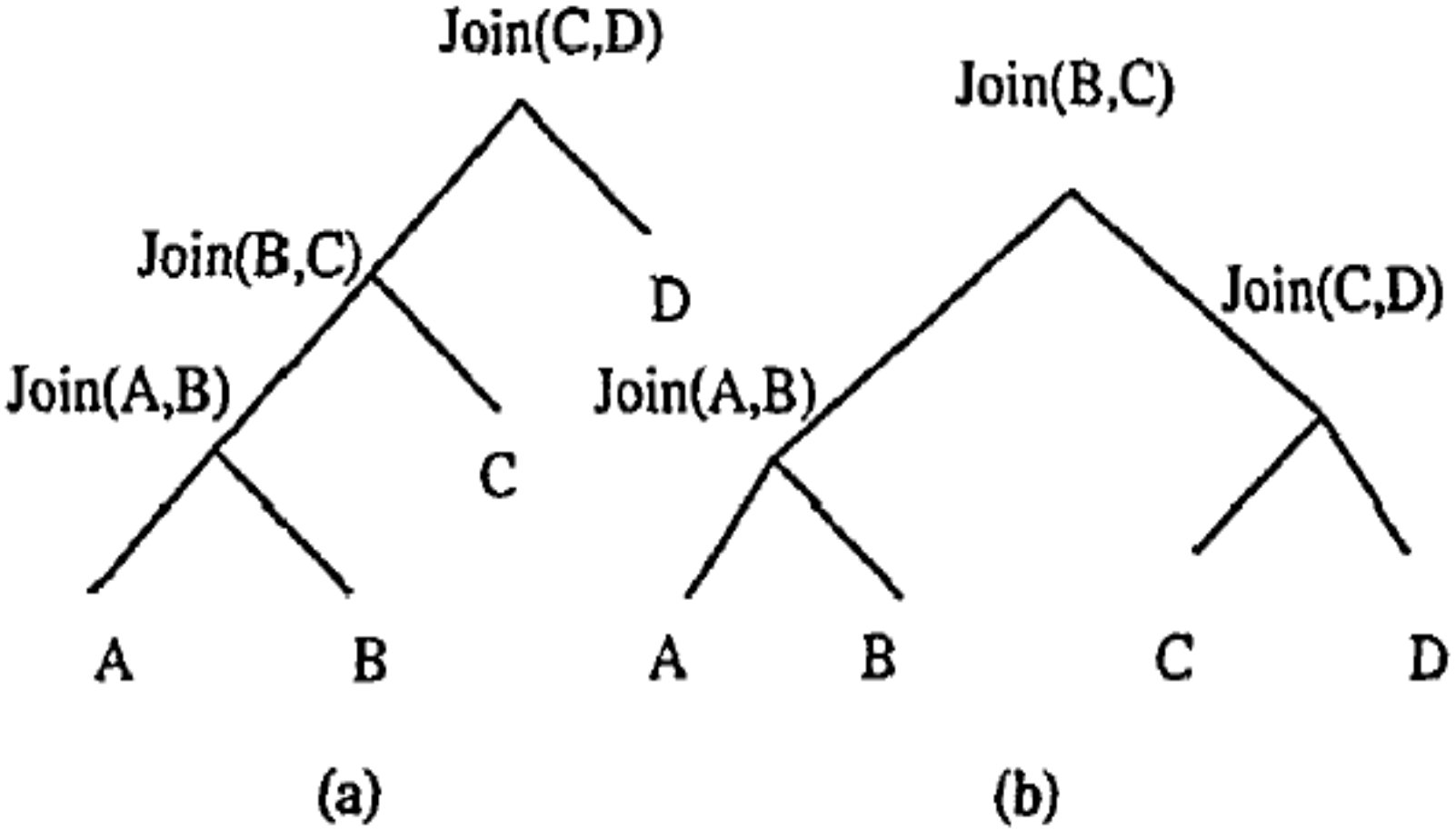
Cost Computation

- 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

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

Linear and Bushy Joins



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

The most asked question (paraphrased)

Couldn’t we just do all this with ML?

- Get into 4 groups
- Divide those who have had ML experience and those have not
- Talk over:
  - What about query optimization would lend itself well to being solved with ML
  - What about query optimization would lend itself poorly?