Overview of Query Optimization in Relational Systems

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

Overview of Query Optimization in Relational Systems

- An overview of current SQL query optimization techniques in relational database systems.
- Gives fundamentals of SQL query optimization

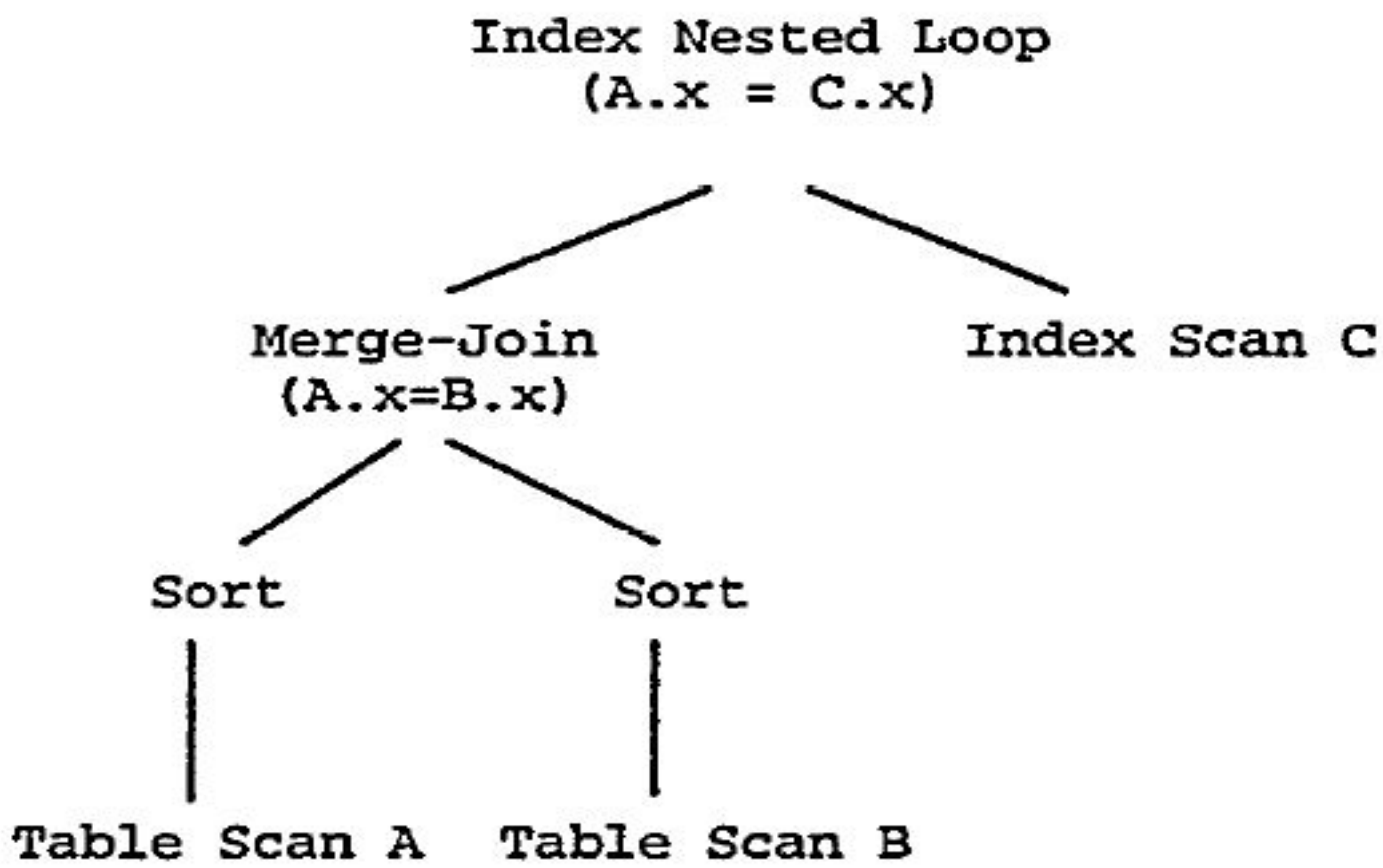
Introduction

- 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,..
  - 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 space of possible execution plan can be huge:
  - Many logically algebraic transformations.
  - Many operator trees for a given representation.
  - Throughput varies widely with each plan.

The Key Idea: Query Optimization as a Search Problem

- To solve problem, we need to provide:
  - Search space (low cost plans desirable)
  - Cost estimation technique to assign a cost to each plan in the search space (accuracy desired)
  - Enumeration algorithm to search through the execution space (efficiency desired)

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

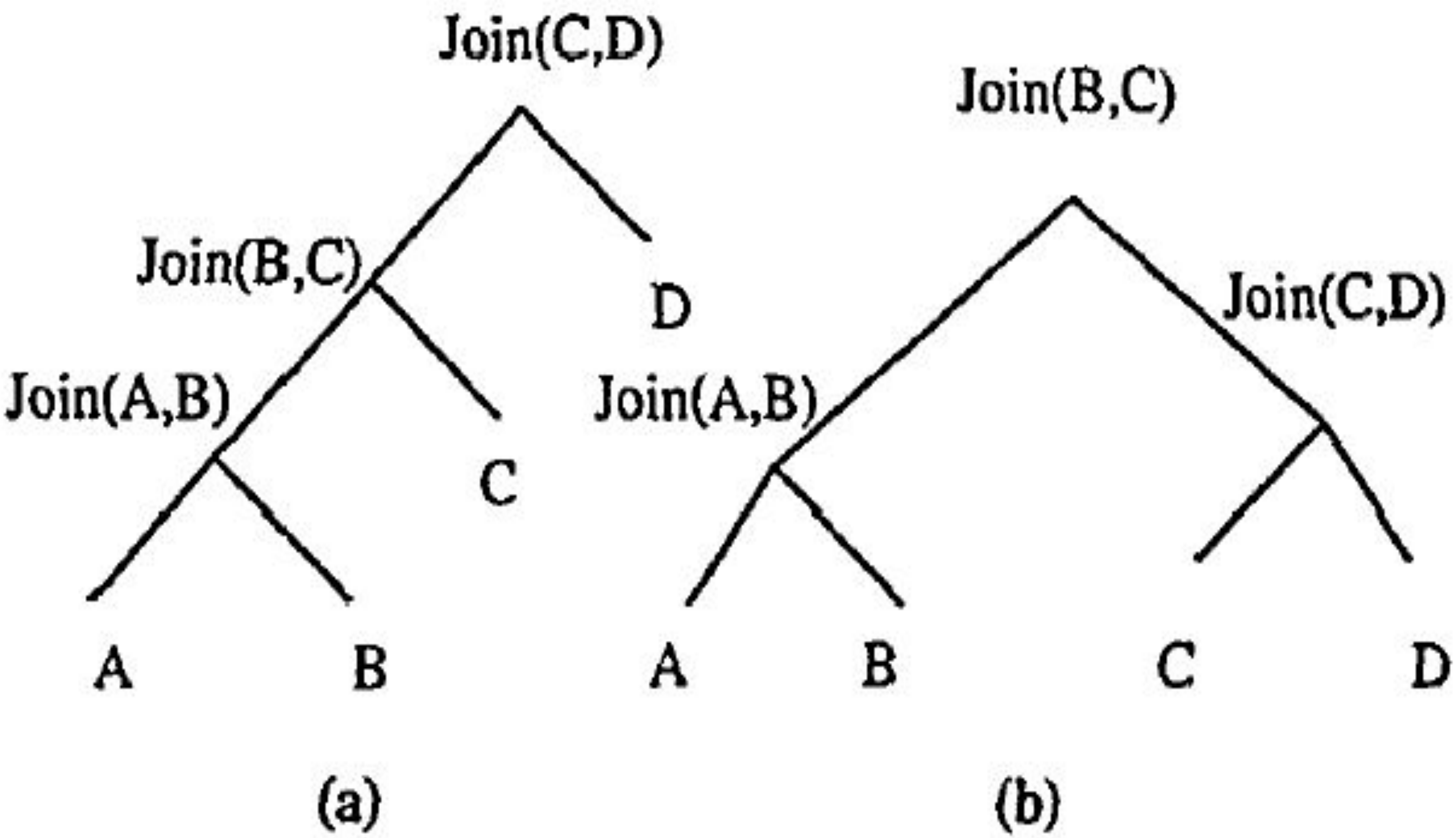
Search Space

- Depends on:
  - Equivalence among 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

Commuting Between Operators

- Generalized Join Sequencing
  - Linear join most common.
  - Bushy join (materialization, cheaper query plan, expensive enumeration)
- Outer Join and Join
  - Join(R, S LOJ T) = Join(R, S) LOJ T
  - Still need to account for cost.
- Group-By and Join
  - In some cases, performing Group-By first may reduce the cost of join.
  - Inexpensive with index.

Linear and 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
  - Uncorrelated.
  - Correlated.
  - Complexity depends on structure.

```
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
```

Discussion

How do you feel about using a declarative language and optimizer to query data? Do you think the users at the time would have been quick to adopt this kind of system? Why or why not?

Statistics and Cost Estimation

- Deciding which operator tree consumes least resources (CPU, I/O, memory,..)
- 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, min or max, second lowest and second highest)
- Can use sampling to build 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

Discussion

What effect do you think all of this tuning and maintenance has on: How database systems are deployed? How they are managed? How they are used?

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

Extensible Optimizers

- Want enumerator to adapt to changes in search space
  - New transformations
  - Addition of new physical operators
  - Changes in cost estimation techniques
- Solution:
  - infrastructure for evolution of optimizer design.
  - Trade off between generality in the architecture and efficiency in enumeration
  - 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
  - effective and correct SQL transformations
  - robust cost metric
  - extensible architecture

Ending Discussions

- What value does this paper contribute to the community and why do you think this paper was accepted?