A Brief Overview of Query Optimization

Wentao Wu

Microsoft Research

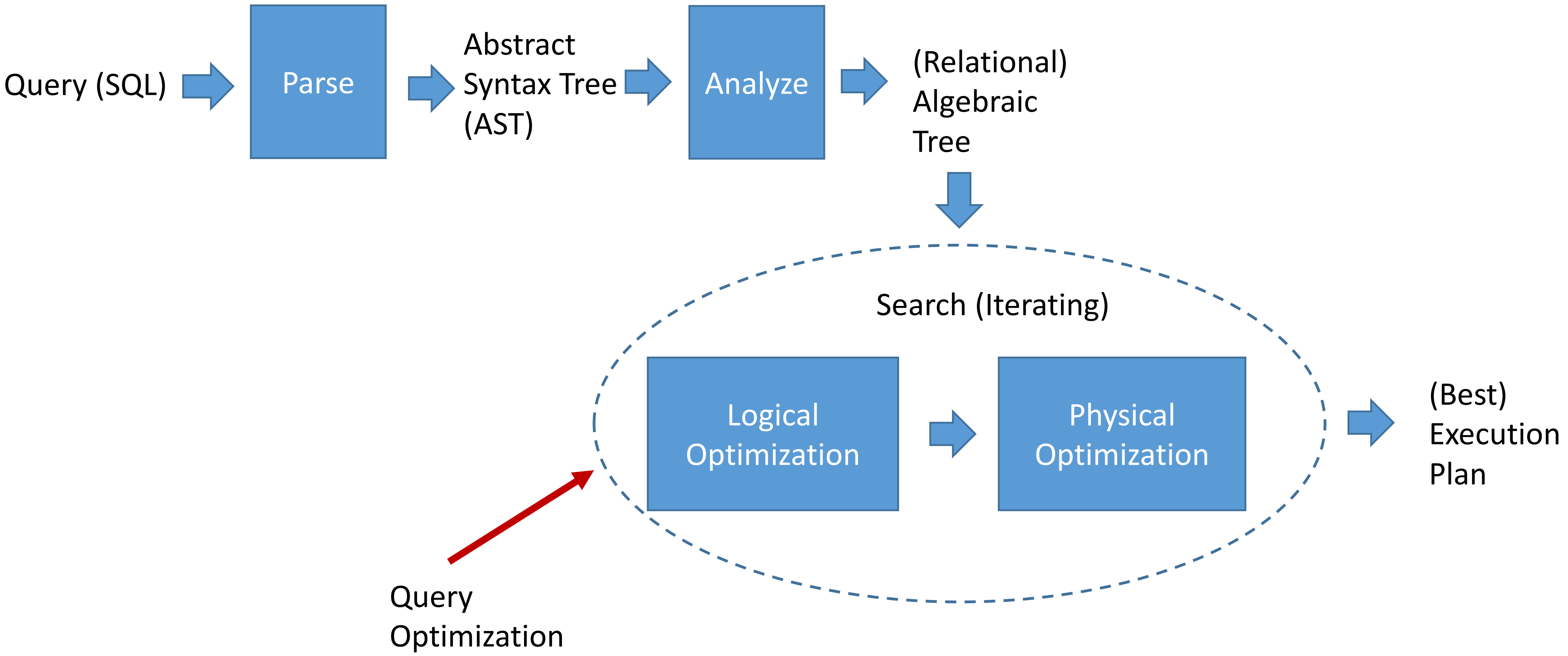
2/9/2018

Outline

- Architecture of Query Optimizer
- Cost Modeling
- Dynamic Query Optimization

Architecture of Query Optimizer

Conceptual View



Logical Optimization

- **Goal:** Produce logically equivalent (relational) algebraic trees.

- Common techniques
  - Push down selections/projections/aggregations.
  - Reorder joins (inner/outer/semi/anti joins).
  - Rewrite nested subqueries.

- References

[1] U. Dayal. Of nests and trees: A unified approach to processing queries that contain nested subqueries, aggregates, and quantifiers. (VLDB’87)

[2] Weipeng P. Yan, Per-Åke Larson: Eager Aggregation and Lazy Aggregation. (VLDB’95)

Physical Optimization

- **Goal:** Replace logical operators in the algebraic tree with physical operators.
  - E.g., join => hash/merge/nested-loop join
  - E.g., aggregation => sort-based/hash-based aggregation

- Common techniques
  - Rule-based: E.g., pattern matching in SparkSQL.
  - Cost-based: Use a cost model to estimate execution cost of a physical plan.

- References

[1] M. Armbrust et al. Spark SQL: Relational Data Processing in Spark. (SIGMOD'15)

[2] Leonard D. Shapiro: Join Processing in Database Systems with Large Main Memories. (ACM Trans. Database Syst., 1986)

Search Framework

- **Goal:** Search for the “best” execution plan (i.e., the plan with the lowest cost).

- Common techniques
  - Bottom-up: Dynamic programming (System R). Used by Oracle, IBM DB2, PostgreSQL.
  - Top-down: Volcano => Cascades. Used by Microsoft SQL Server, Greenplum (Pivotal).

- References

[1] P. Selinger et al.: Access Path Selection in a Relational Database Management System. (SIGMOD’79)

[2] Goetz Graefe: The Cascades Framework for Query Optimization. (IEEE Data Eng. Bull., 1995)

[3] M. A. Soliman et al.: Orca: A Modular Query Optimizer Architecture for Big Data. (SIGMOD’14)

Other References

- Surveys

  [1] Yannis E. Ioannidis: Query Optimization. (ACM Comput. Surv. 28:1, 1996)

  [2] Surajit Chaudhuri: An Overview of Query Optimization in Relational Systems. (PODS’98)

- Search frameworks

  [1] Guy M. Lohman: Grammar-like Functional Rules for Representing Query Optimization Alternatives. (SIGMOD’88)

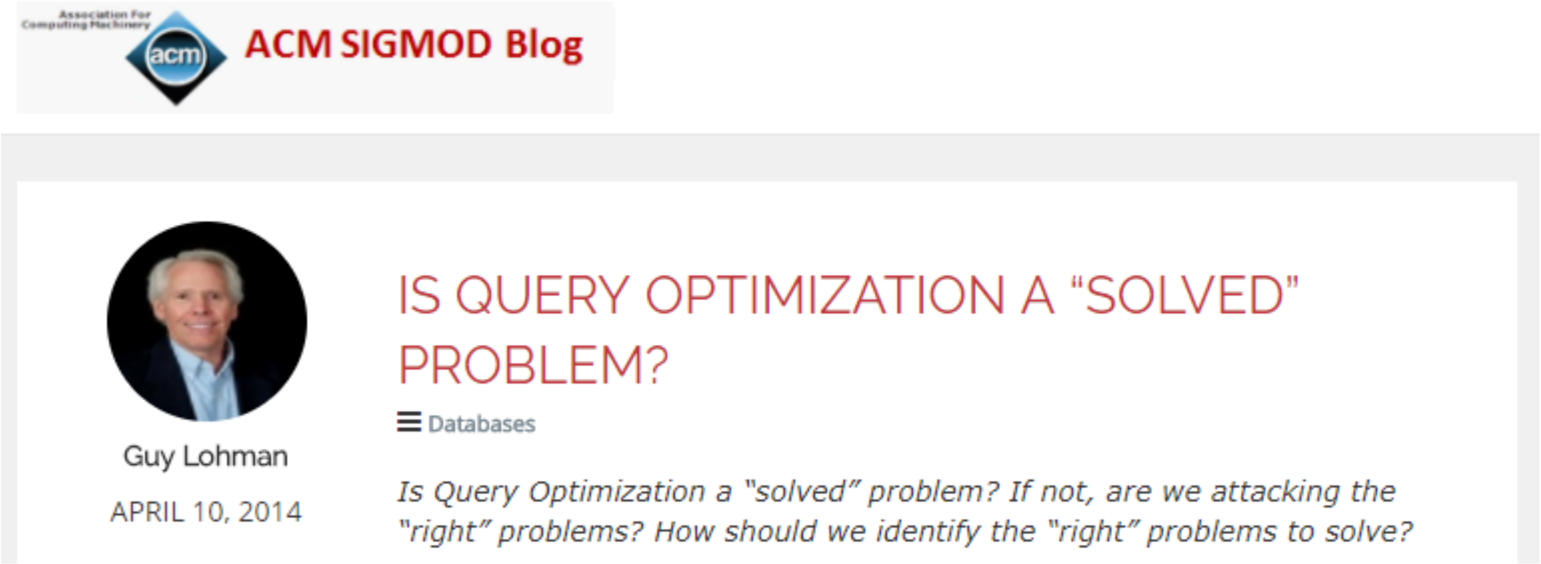
  [2] Laura M. Haas, Johann Christoph Freytag, Guy M. Lohman, Hamid Pirahesh: Extensible Query Processing in Starburst. (SIGMOD’89)

  [3] Goetz Graefe, William J. McKenna: The Volcano Optimizer Generator: Extensibility and Efficient Search. (ICDE’93)

  [4] Immanuel Trummer, Christoph Koch: Solving the Join Ordering Problem via Mixed Integer Linear Programming. (SIGMOD’17)

Is Query Optimization a “Solved” Problem?

- Well, it has been 40 years since the 1979 System-R paper …



What are the “right” problems that remain unsolved?

Cost Modeling: The Pain

Query Optimizer Needs Good Cost Models

- Unlike the other components in a query optimizer, cost modeling lacks a standard procedure and is case-by-case.
  - It is based on the "knowledge" from the database system developers about the relative execution overheads of different operators.

- Common techniques
  - Analytical modeling: Used by most (if not all) major database systems.
  - Machine learning: One of the hot research areas in recent years.

Analytic Modeling

- Basically develop cost formulas for different operators.
  - E.g. Cost = CPU cost + I/O cost + Network communication cost

- Cost models need validation and calibration.

- References

[1] Lothar F. Mackert, Guy M. Lohman: R* Optimizer Validation and Performance Evaluation for Local Queries. (SIGMOD'86)

[2] Lothar F. Mackert, Guy M. Lohman: R* Optimizer Validation and Performance Evaluation for Distributed Queries. (VLDB'86)

[3] W. Du, R. Krishnamurthy, and M.-C. Shan. Query optimization in a heterogeneous dbms. (VLDB'92)

[4] S. Mangegold et al., Generic database cost models for hierarchical memory systems. (VLDB'02)

Machine Learning

- Don’t trust the cost formulas made by optimizer developers.
  - Learn cost functions based on query execution data.

- References

[1] A. Ganapathi et al., Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. (ICDE’09)

[2] M. Akdere et al., Learning-based Query Performance Modeling and Prediction. (ICDE’12)

[3] J. Li et al., Robust Estimation of Resource Consumption for SQL Queries using Statistical Techniques. (PVLDB 5:11, 2012)

[4] W. Wu et al., Predicting Query Execution Time: Are Optimizer Cost Models Really Unusable? (ICDE’13)

([3] and [4] combine analytic modeling with machine learning.)

Cardinality Estimation: The Hardest Part

- No matter you use analytic modeling or machine learning, you need cardinality information (i.e., sizes of intermediate results produced by operators in query execution plans).
- Recent work shows that cardinality estimation may be the most (and often the only) important thing in cost modeling.

  [1] V. Leis et al., How Good Are Query Optimizers, Really? (PVLDB 9: 3, 2015)
- Common techniques
  - Use histograms: equi-width, equi-depth, multi-dimensional.
  - Use samples, sketches, statistical models, execution feedback, etc.

Single-column Histograms

- Histograms for a single column
  - Equi-width, equi-depth, V-optimal, …
  - **Attribute-Value-Independence (AVI) assumption**: Assume the independence between histograms (i.e., distributions) when estimate selectivity/cardinality for predicates involving more than one columns (e.g., X > 3 and Y < 8).
  - The estimation error can be exponential under AVI.

- References

[1] M. Muralikrishna and David J Dewitt., Equi-depth histograms for estimating selectivity factors for multidimensional queries. (SIGMOD'88)

[2] Yannis E. Ioannidis, Stavros Christodoulakis: On the Propagation of Errors in the Size of Join Results. (SIGMOD'91)

[3] V. Poosala et al., Improved histograms for selectivity estimation of range predicates. (SIGMOD'96)

[4] Yannis E. Ioannidis: The History of Histograms (abridged). (VLDB'03)

Multi-column Histograms

- Motivation: Overcome the AVI assumption.
  - Use histograms to capture the joint distribution across multiple columns.

- Drawback: The size increases exponentially w.r.t. the # of columns.
  - Workload-driven approaches: Only construct multi-column histograms for columns that appear in workload queries.

- References

[1] V. Poosala, Y. E. Ioannidis: Selectivity Estimation Without the Attribute Value Independence Assumption. (VLDB’97)

[2] N. Bruno et al., STHoles: A Multidimensional Workload-Aware Histogram. (SIGMOD’01)

[3] I. F. Ilyas et al., CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. (SIGMOD’04)

Other Approaches

- Sampling/Sketches (References)

[1] R. Lipton et al., Practical Selectivity Estimation through Adaptive Sampling. (SIGMOD'90)
[2] P. J. Haas et al., Selectivity and cost estimation for joins based on random sampling. (J. Comput. Syst. Sci., 52:3, 1996)
[3] S. Acharya et al., Join Synopses for Approximate Query Answering. (SIGMOD'99)
[4] Phillip B. Gibbons: Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. (VLDB'01)
[5] D. Vengerov et al., Join size estimation subject to filter conditions. (PVLDB 8:12, 2015)
[6] Yu Chen, Ke Yi: Two-Level Sampling for Join Size Estimation. (SIGMOD'17)

Theoretical results:

[1] S. Chaudhuri et al., On Random Sampling over Joins. (SIGMOD'99)
[2] M. Charikar et al., Towards Estimation Error Guarantees for Distinct Values. (PODS'00)
[3] M. Riondato et al., The VC-Dimension of SQL Queries and Selectivity Estimation through Sampling. (ECML/PKDD'11)

- Statistical models/Feedback (References)

[1] L. Getoor et al., Selectivity Estimation using Probabilistic Models. (SIGMOD'01)
[2] M. Stillger et al., LEO - DB2's LEarning Optimizer. (VLDB'01)
[3] L. Tzoumas et al., Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions. (PVLDB 4:11, 2011)

Dynamic Query Optimization

Motivation

- So far, we have been talking about “static query optimization”.
  - We assume that a query plan is ready and won’t be changed during execution.
  - The performance of the query plan is subject to cost modeling, which depends on the accuracy of cardinality estimation, an inherently hard problem.

- However, why should we stick with one single query plan?
  - We shouldn’t!

- Dynamic query optimization (a.k.a., interleave query optimization with query execution/processing)
  - Let’s prepare multiple plans and decide at runtime which one(s) to use.

Two Key Problems

- How to generate multiple query plans?
- When to switch to a different query plan?
- Common techniques
  - Parametric query optimization
  - Adaptive/robust query processing
  - Mid-query re-optimization

Parametric Query Optimization

- Mainly used for stored procedures (query templates).
  - Rather than use one plan for all parameter values, use different plans for different parameter values.
  - Multiple plans are generate during query compilation/optimization.
  - Pick one plan before execution depending on the parameter value observed.

- References

[1] Y. E. Ioannidis et al., Parametric Query Optimization. (VLDB'92)

[2] A. Hulgeri, S. Sudarshan, Parametric query optimization for linear and piecewise linear cost functions. (VLDB'02)

[3] N. Reddy, J. R. Haritsa: Analyzing Plan Diagrams of Database Query Optimizers. (VLDB'05)

[4] J. R. Haritsa, Query optimizer plan diagrams: Production, reduction and applications. (ICDE'11)

Adaptive/Robust Query Processing

- Generate multiple plans during query compilation.
  - Similar to parametric query optimization (PQO).
- Dynamically switch plans based on feedback from execution.
  - This is different from PQO, which does not switch after execution starts.
- References

[1] G. Graefe, K. Ward: Dynamic Query Evaluation Plans. (SIGMOD'89)
[2] G. Antoshenkov: Dynamic Query Optimization in Rdb/VMS. (ICDE'93)
[3] R. L. Cole, G. Graefe: Optimization of Dynamic Query Evaluation Plans. (SIGMOD'94)
[4] R. Avnur, J. M. Hellerstein: Eddies: Continuously Adaptive Query Processing. (SIGMOD'00)
[5] A. Dutt, J. R. Haritsa: Plan bouquets: query processing without selectivity estimation. (SIGMOD'14)

Mid-Query Re-Optimization

- Start from the plan generated by the optimizer.
  - So there is only one plan after query compilation/optimization stage. (This is different from PQO and adaptive/robust query optimization.)

- At runtime, keep monitor feedback from query execution.
  - If there is evidence that the current plan is sub-optimal (e.g., significant cardinality estimation error), stop execution and ask the optimizer to re-optimize the *remaining* part of the query based on execution feedback.

- References

[1] N. Kabra, D. J. DeWitt: Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. (SIGMOD'98)

[2] V. Markl et al., Robust Query Processing through Progressive Optimization. (SIGMOD'04)

[3] S. Babu et al., Proactive Re-optimization. (SIGMOD'05)

[4] W. Wu et al., Sampling-based query re-optimization. (SIGMOD'16)

Thank you!

The Optimizer of Microsoft SQL Server

- Reference for the example below:

  [1] Florian Waas, César A. Galindo-Legaria: Counting, Enumerating, and Sampling of Execution Plans in a Cost-Based Query Optimizer. (SIGMOD'00)

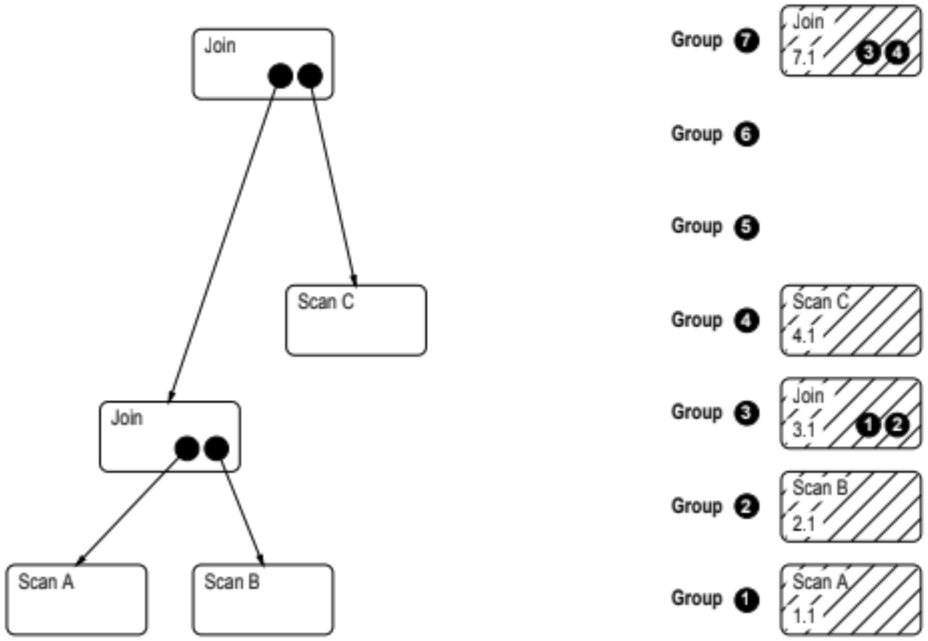


Figure 1: Copying the initial plan into the MEMO structure.

The Optimizer of Microsoft SQL Server (Cont.)

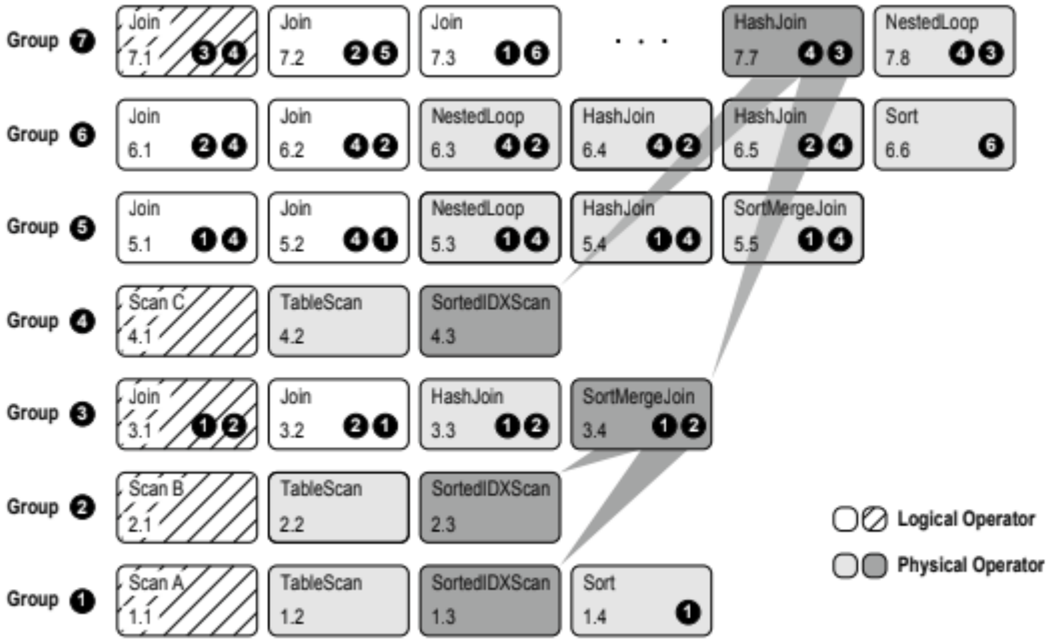


Figure 2: MEMO structure representing alternative solutions.

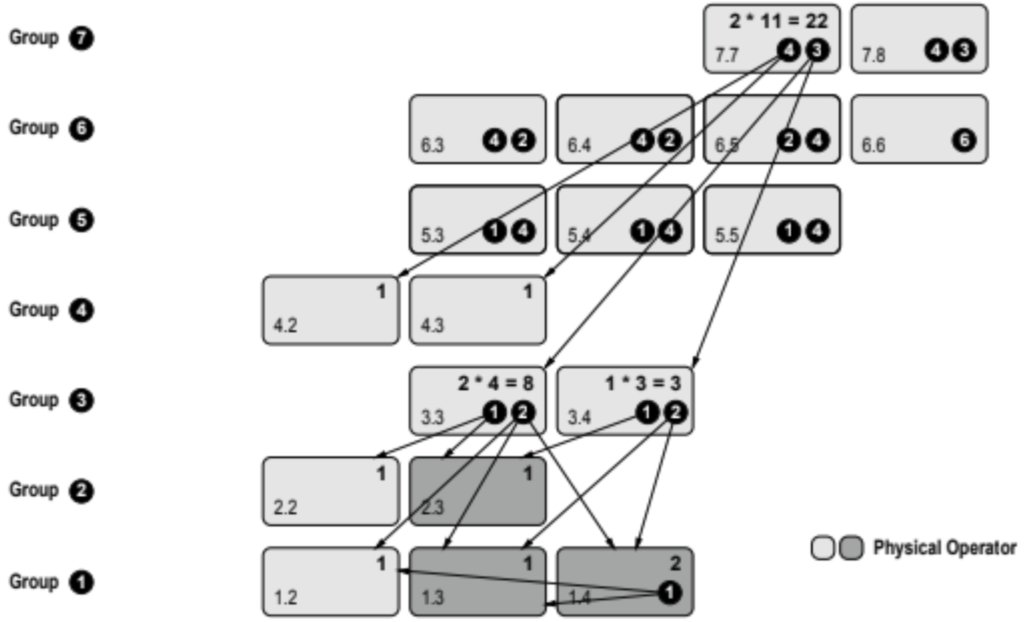


Figure 3: MEMO Structure with materialized links between operators and children, for possible plans rooted in operator 7.7.