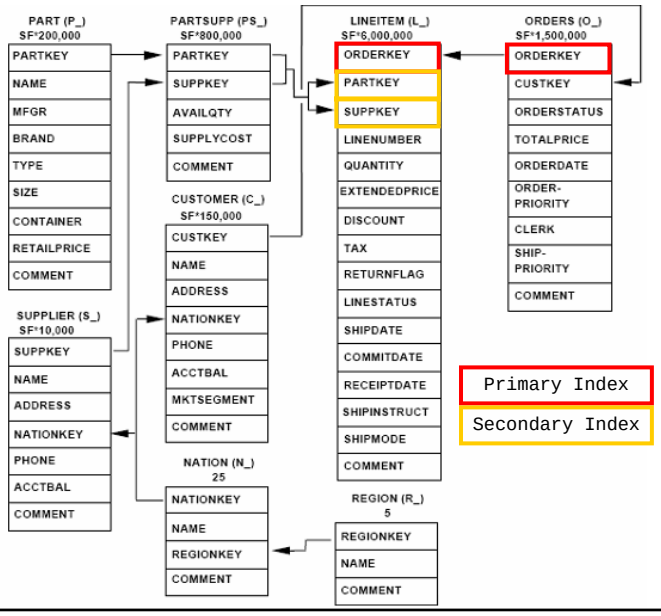
Outline

- SQL Server Optimizer
  - Enumeration architecture
  - Search space: flexibility/extensibility
  - Cost and statistics
- Automatic Physical Tuning
  - Database Tuning Advisor
  - New Directions

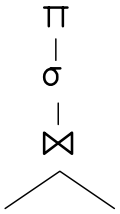
Running Example (TPC-H Database)



Running Example (Query)

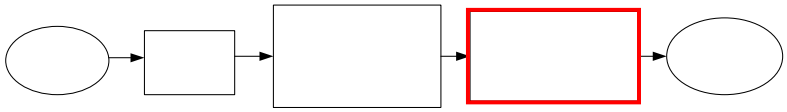
Sample query: Obtain information about certain ordered line-items that are filtered by suppliers and parts.

```
SELECT l_orderkey, l_linenumber, o_orderstatus
FROM   lineitem JOIN orders ON l_orderkey = o_orderkey
WHERE  l_suppkey < 2000 AND l_partkey < 2000
```



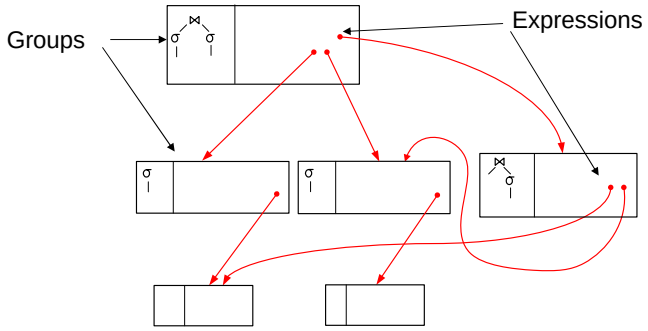
SQL Server Query Optimizer

- Based on Cascades Framework
  - Transformation-based, top-down approach
  - Optimization = Tasks + Memo
    (Programs = Algorithms + Data Structures)
- Fully cost-based
- Flexible and Extensible
  - Search space easy to change
  - New operators and rules easy to add

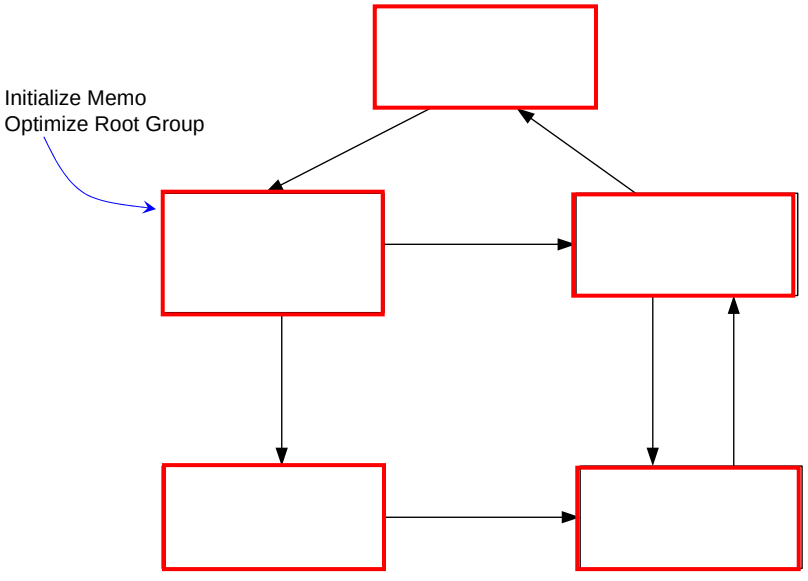


The Memo

- Search Space Memory
  - Compactly stores all explored alternatives (AND-OR graph)
  - Groups together equivalent operator trees and their plans
  - Provides memoization, duplicate detection, property and cost management, etc.



Optimization Tasks



1) SELECT (a<1C)
2) JOIN (x=y,)
3) ..

t>20

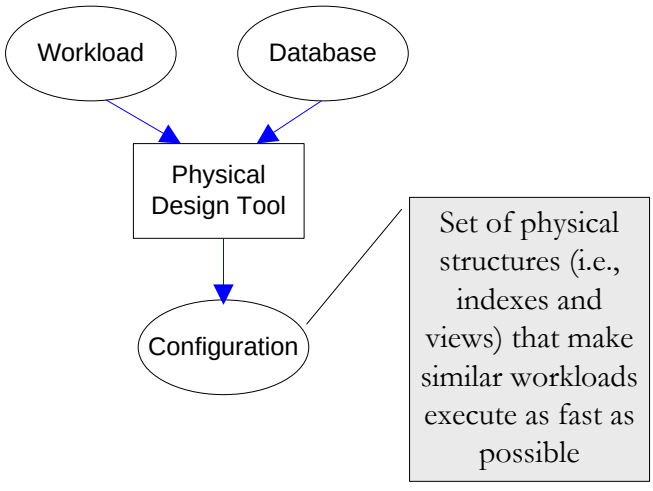
T (a<1C) S 1) SELECT

R) S 1) GET(S

SQL Server Optimizer: Summary

- Transformation-based, top-down approach
  - No need for bottom-up interesting orders
- Fully Cost-based
  - No separation into phases (heuristic+cost)
- Flexible and Extensible
  - New operators, rules, and strategies are simple to add
- Adaptive
  - Automatic statistics create and refresh
  - Automatic optimization levels
  - Physical Tuning

Problem Statement



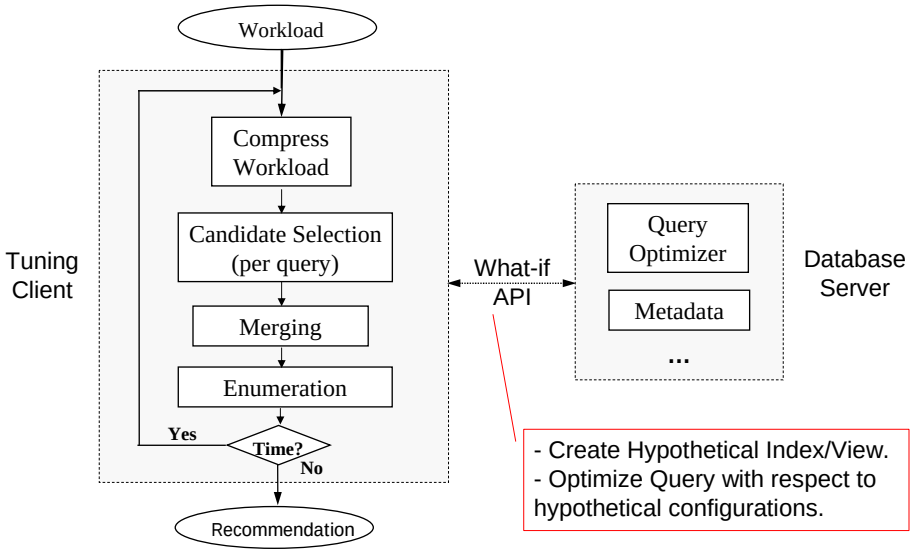
Challenges

- Recommend a variety of physical structures.
  - Indexes, indexed views, partitions, XML indexes, etc.
- Support space constraints, update queries.
- Exceptionally large search space, especially for materialized views and partitions.
  - Strong interaction among access paths.
  - Merging needed: Optimal solution with suboptimal parts.
  - Cannot implement and test alternatives!
- Industrial-strength quality (not trivial!)

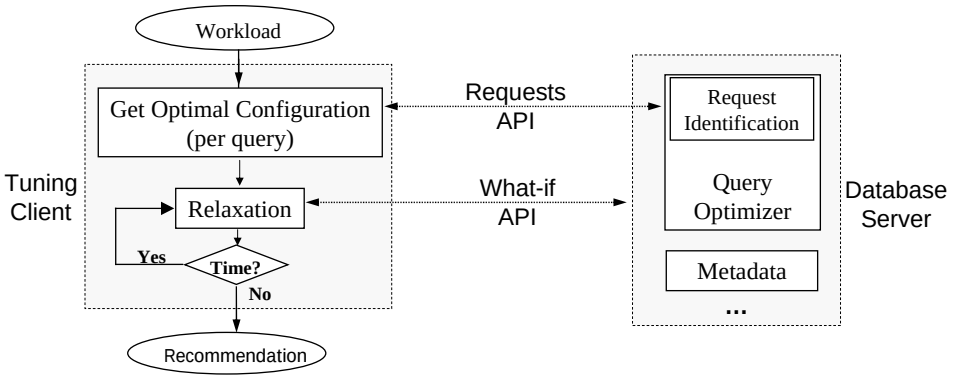
Design Principles

- Workload-driven: Take into account database *usage.*
- What-if API: Determine impact *without* actually materializing physical design.
- Don't second-guess the query optimizer!
  - An index is useful *only* if the query optimizer decides to use it.
  - Don't use external estimator of goodness, but the optimizer-estimated cost.
- Keep tool reasonably separated from today's optimizer (extensibility).

Database Tuning Advisor in Yukon



New Architecture



- Instrumenting the query optimizer.
- Search strategy based on relaxations.

Related Bibliography

- Surajit Chaudhuri
  An Overview of Query Optimization in Relational Systems.
  PODS 1998: 34-43
- Goetz Graefe
  The Cascades Framework for Query Optimization.
  IEEE Data Eng. Bull. 18(3): 19-29 (1995)
- Surajit Chaudhuri, Vivek R. Narasayya
  An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server.
  VLDB 1997: 146-155
- Sanjay Agrawal, Surajit Chaudhuri, Vivek R. Narasayya
  Automated Selection of Materialized Views and Indexes in SQL Databases.
  VLDB 2000: 496-505
- Nicolas Bruno, Surajit Chaudhuri
  Automatic Physical Database Tuning: A Relaxation-based Approach.
  SIGMOD 2005