Distributed Data Processing Environments

José Orlando Pereira

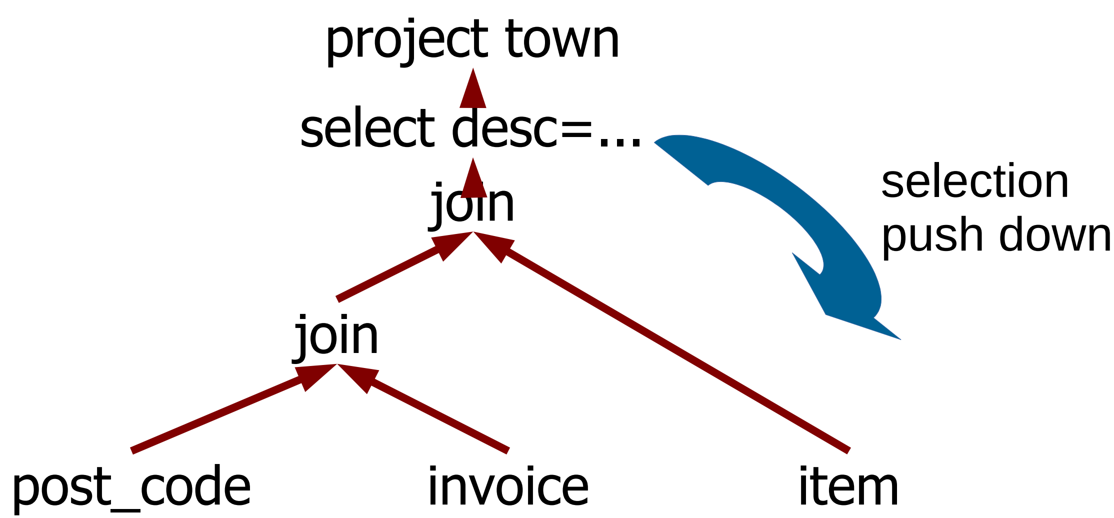
Departamento de Informática
Universidade do Minho

Roadmap

- What physical operators exist for each logical operation?
- How are physical operators implemented and composed?
- How is the best plan found?

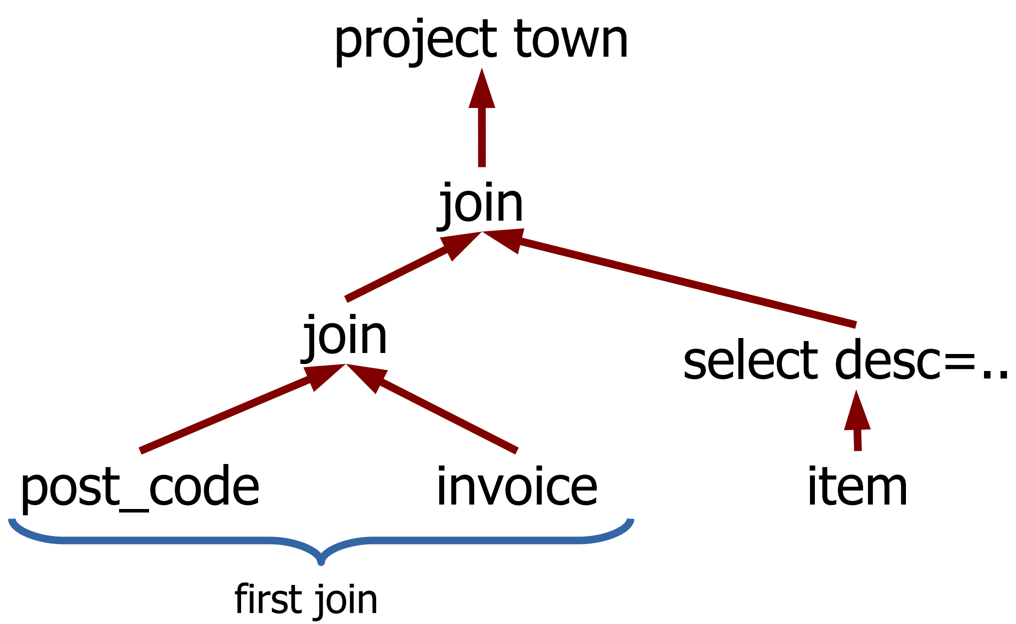
Optimization

select town from
post_code natural join invoice
natural join item where desc=...



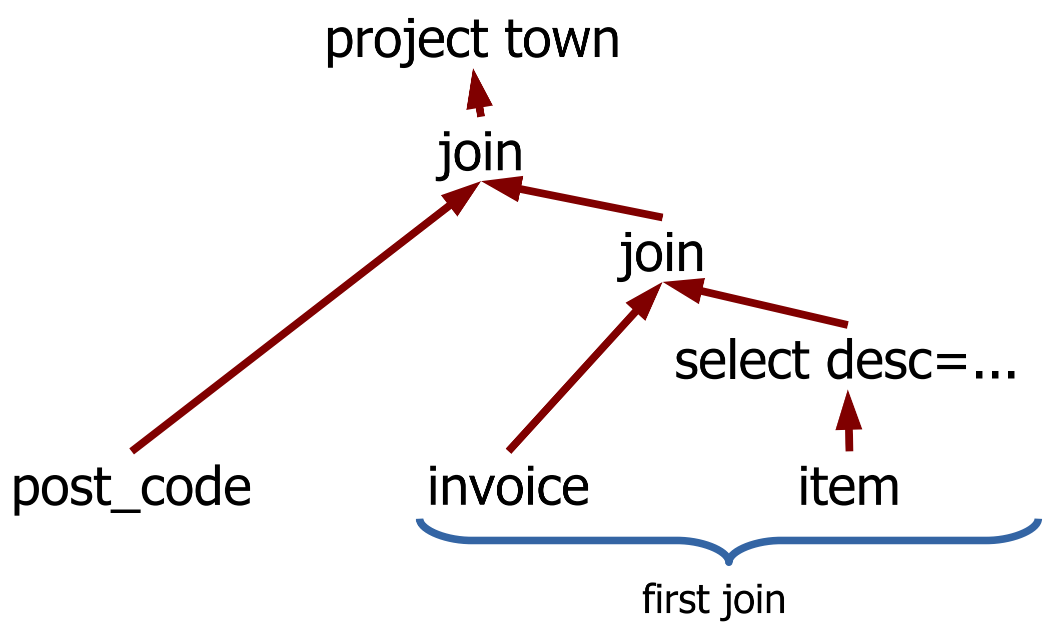
Optimization

select town from
post_code natural join invoice
natural join item where desc=...



Optimization

select town from
post_code natural join invoice
natural join item where desc=...



Roadmap

- How to estimate the cost of a plan?

- How to find alternative plans?

Cost estimation

- Tradeoff beween:
  - Actually executing the query and measuring what resources it consumes and how long it takes to execute

    vs
  - **Estimate that can be computed quickly**

    vs
  - Considering that all queries have the same cost
- We don’t want to know what is the actual cost
- We want to know which alternative costs less
  - Use a <u>model</u> that monotonously approximates real cost

Example: Simple sequential scan

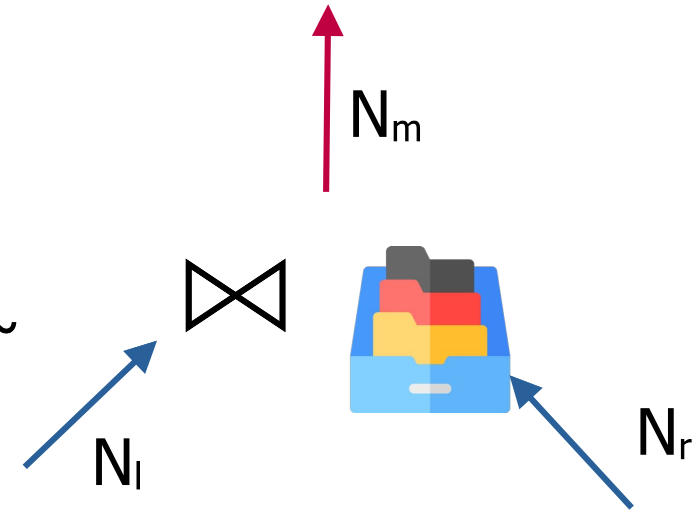
- Assumption of main cost factor:
  - Number of disk block I/O operations
- <u>Cost model</u> as a simple equation:
  - $C = C_0 + C_1 N_{blocks}$

Known from the size of the file

How to find them?

Example: Hash Join

- CPU cost:
  - Building hash table from inner table ~ N_r input rows
  - Checking each row in the outer table ~ N_l input rows
  - Creating the resulting <u>N_m rows</u>
- Very high cost if not enough memory to hold N_i rows in the hash table



How to estimate?

Icons by Flaticon.com.

Coefficients

- Cost coefficients depend on the actual hardware, software, and even workloads

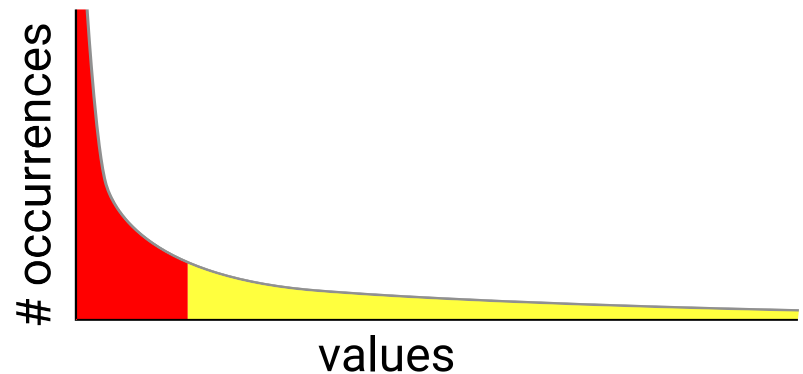
- Can be estimated by profiling simple workloads
- Can be tuned by the DBA

Cardinality - Selection

- Assumption of uniform distribution
- Know <u>#distinct</u>
  - Expected copies of each tuple:
    - #rows / #distinct

Cardinality - Selection

- Real data are usually not uniformly distributed:
  - 80/20 rule
  - Power law



- Know <u>most popular</u> and <u>#occurrences</u>
- Compute estimated #occurrences of others as uniformly distributed

Image source: Wikipedia

Cardinality - Selection

- Data often have complex multimodal distributions
- Know histogram:
  - % of occurrences in each interval or
  - interval for fixed % of occurrences
- Compute #occurrences as uniformly distributed within each interval

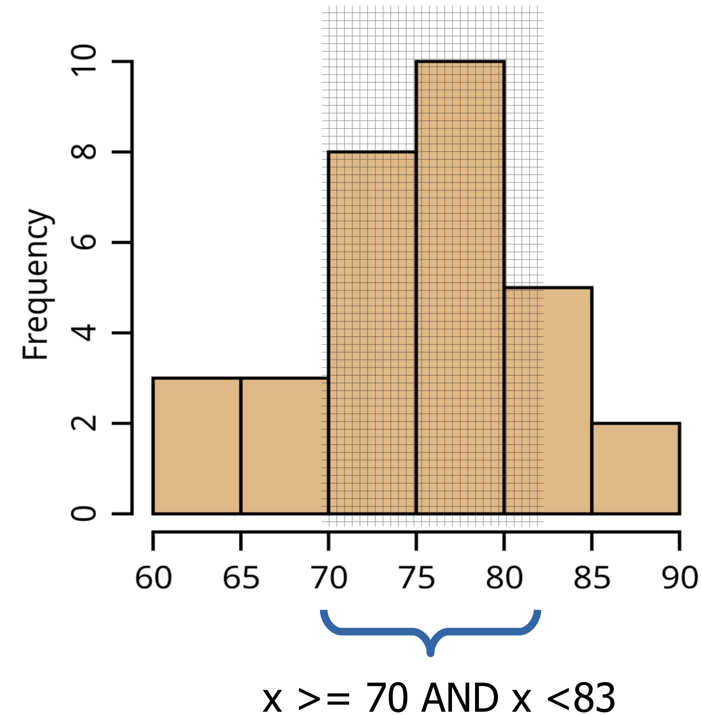


Image source: Wikipedia

Cardinality - Conjunction

- Filter conditions are often the conjunction of conditions on different columns
- Statistics on multiple columns are expensive:
  - Multidimensional
  - Many possible combinations
- Assume independently distributed values in different columns:
  - selectivity = #selected / #total rows
  - selectivity(a $\wedge$ b) = selectivity(a) * selectivity(b)
- Idem for disjunction

Cardinality - Join

- Cardinality for cross-product of A and B:
  - #rows from A × #rows from A
- Cardinality for join, first attempt:
  - Align buckets of histograms for A and B
  - Estimate the cardinality of each matching bucket as a cross-product:
    - Not good, as there are non-matching values
      - e.g. A has even numbers, B has odd numbers → no match!

Cardinality - Join

- Assume that one relation A has all values (containment)
- Cardinality for join, second attempt:
  - Align buckets of histograms for A and B
  - Estimate the cardinality of each matching bucket:
    - Each of #rows in B matches (#rows in A / #distinct in A)
      - Because A has all the values
    - Estimate as #rows B × #rows in A / #distinct in A
  - Generalize for the case where B has all values:
    - #rows in A × #rows in B / max(#distinct in A, #distinct in B)

Summary

- Tradeoff between complexity (data and computation) and accuracy
- Many other techniques:
  - More statistics
  - Heuristics
  - Hinting
  - Sampling of query
  - Machine learning
  - …

Roadmap

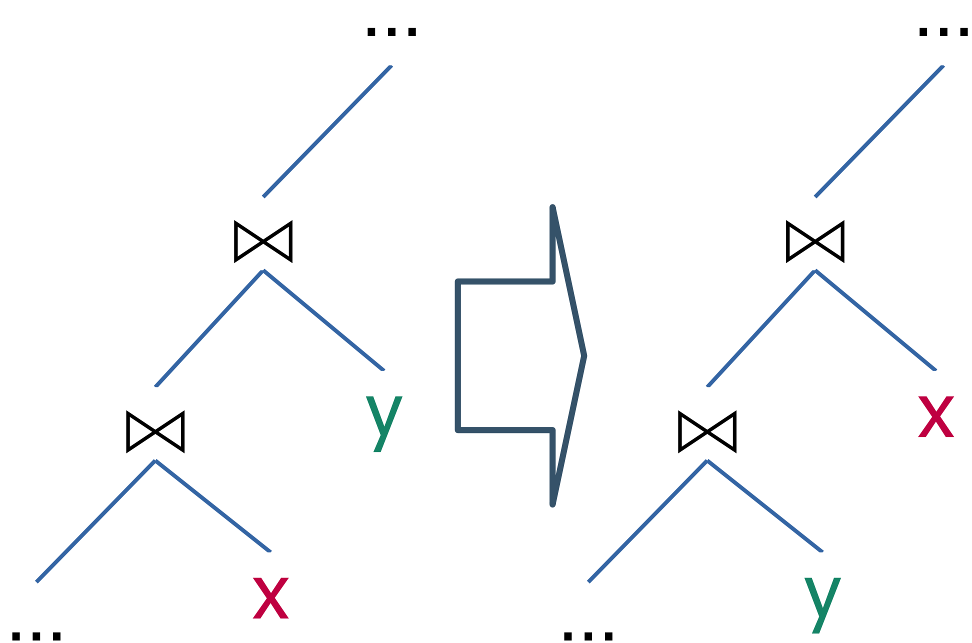
- How to estimate the cost of a plan?

- How to find alternative plans?

Search space

- The set of possible alternative plans (search space) is determined by a set of <u>rules</u>
  - Equivalent relational algebra expressions
  - Physical implementation of single operators or plan fragments
  - Enforcing physical properties
- The set of rules is the main configuration point for <u>extensible query optimizers</u>

A simple rule for Join order

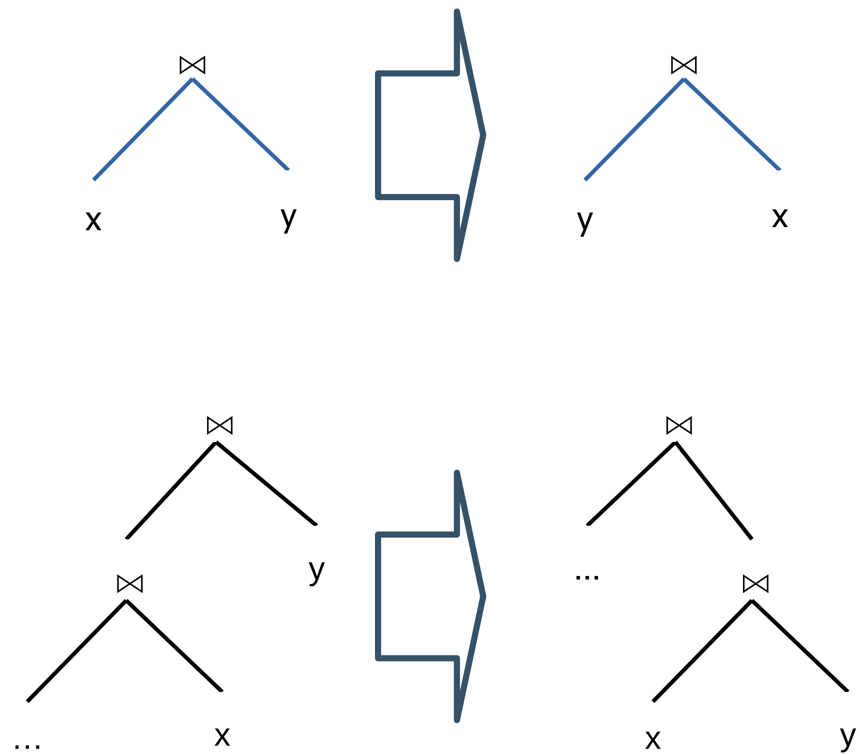


- Inner join is commutative and associative

  $(\ldots \bowtie \mathbf{x}) \bowtie \mathbf{y}$ =
  $\ldots \bowtie (x \bowtie y)$ =
  $\ldots \bowtie (y \bowtie x)$ =
  $(\ldots \bowtie \mathbf{y}) \bowtie \mathbf{x}$

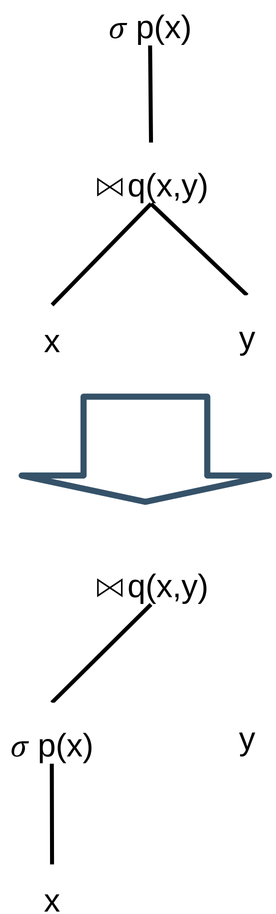
- This allows exploring all <u>left-deep</u> trees
  - n! permutations
- Does not consider bushy trees

Join commutativity and associativity



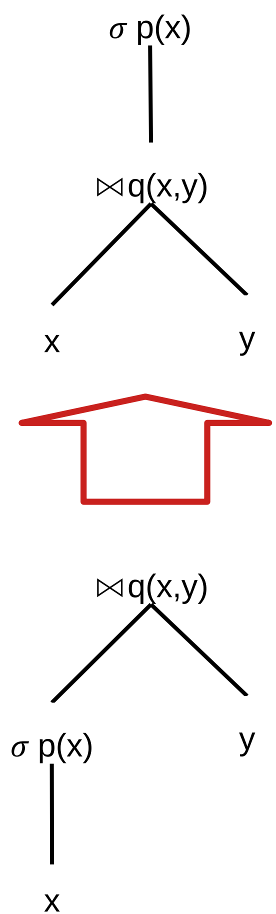
- Separately considering commutativity and associativity rules
- Allows exploring all possible join expressions:
  - Including bushy trees

Selection push-down



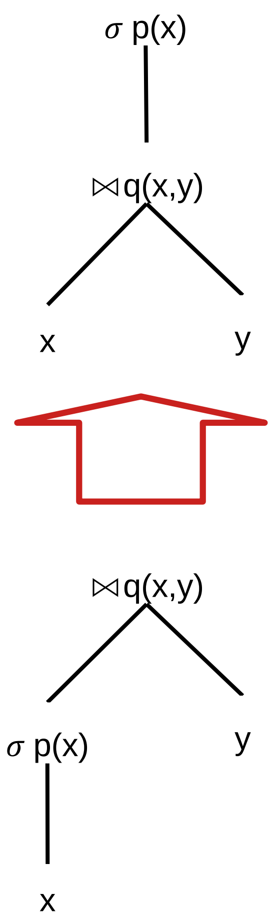
- Possible when the predicate involves only one of the branches
  - Otherwise, merge it in the join condition!
- If the predicate is highly selective, reduces the amount of work in join
- Selectivity is the key criteria for join ordering!!!

Selection pull-up



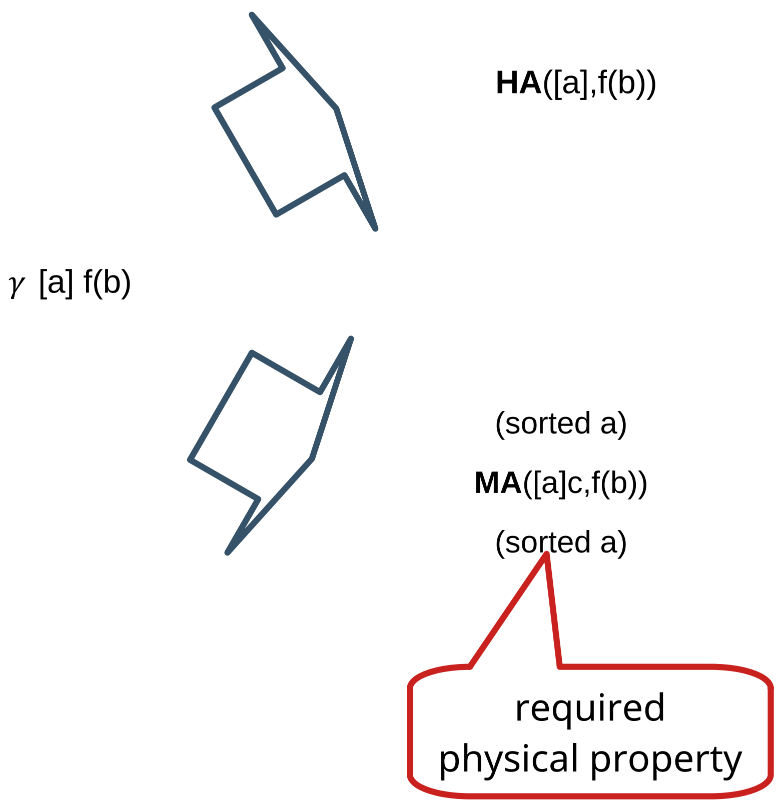
- Always possible, but…
- Does it ever make sense?!?!

Selection pull-up



- Assume that q is highly selective
- Assume that p is very <u>costly</u> to execute:
  - e.g. call into an LLM to check if "sentiment" on a textual column is "positive"
- Can be useful!

Grouping and aggregation implementation

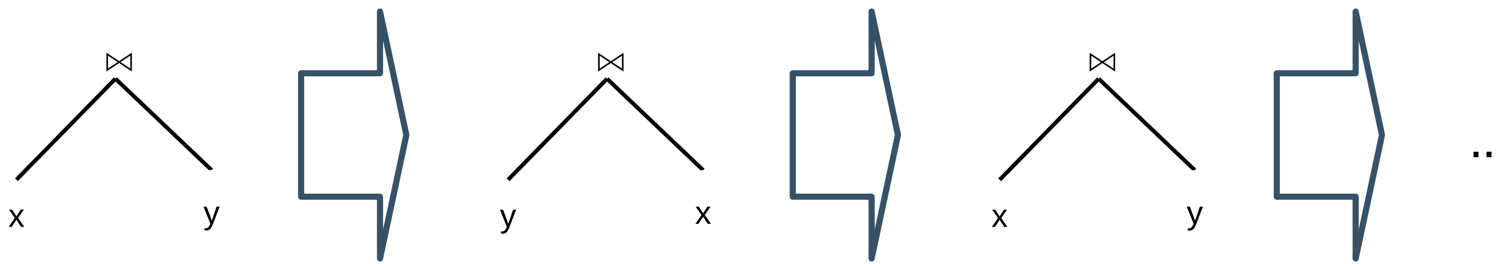


- Two-pass algorithms for grouping (and join) depend on sorted input
  - Expressed as a required "physical property"
- Sorting is preserved

Search space

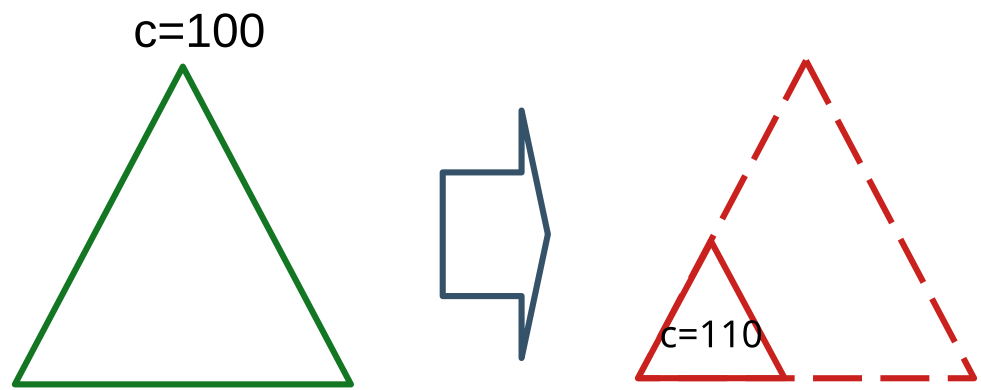
- Rule sets that produce a larger search space:
  - More likely to contain the optimum → faster execution
  - More work to evaluate all alternatives → faster planning
- Query execution is the sum of planning and execution
- Best overall performance:
  - Depends on workload
  - Found as a compromise of planning and execution

Search algorithm



- Repeated application of rules can result in an infinite loop
- Solution:
  - <u>Remember all plans</u> to check for repetitions

Search algorithm



- The cost of a sub-plan of an alternative being explored may be greater that the total cost of the best know alternative
- Solution:
  - <u>Prune</u> search based on current best estimate

Search algorithm

- Some sequences of rule applications tend to converge faster to the optimum plan
  - The sooner that we get a “good” estimate, the more alternatives that can be pruned
- Solution:
  - Order available rules by their heuristic <u>promise</u>

Conclusions

- The set of possible alternative execution plans for common analytical operations is extremely large
- There is no greedy algorithm that can find the optimum
- The optimum depends on current data, defeating manual optimization efforts

- The best option is to use a declarative data processing system such as a SQL query engine