

FluidB: Adaptive storage layout using reversible relational operators

<Subtitle>

Christos Perivolaropoulos

University of Edinburgh

January 1, 1980

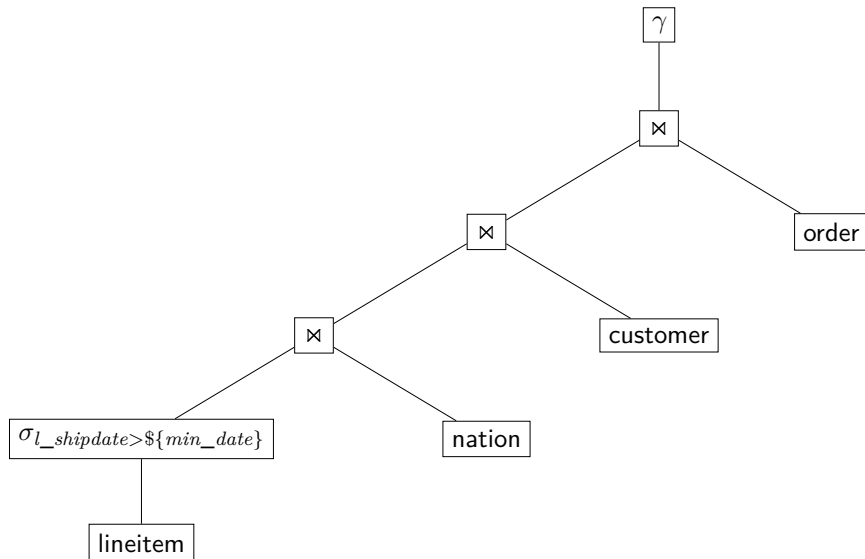
FluiDB at a glance

- FluiDB is an in-memory RDBMS
- Radical approach to IR recycling: adapt data layout to the workload
 - ▶ enable efficient plans
 - ▶ constrained (quality) budget
- The main novelty relates to the introduction of reversible relational operations which affords a new perspective on query planning and view selection.
- FluiDB materializes all intermediate results and deletes garbage collects when she runs out of space.

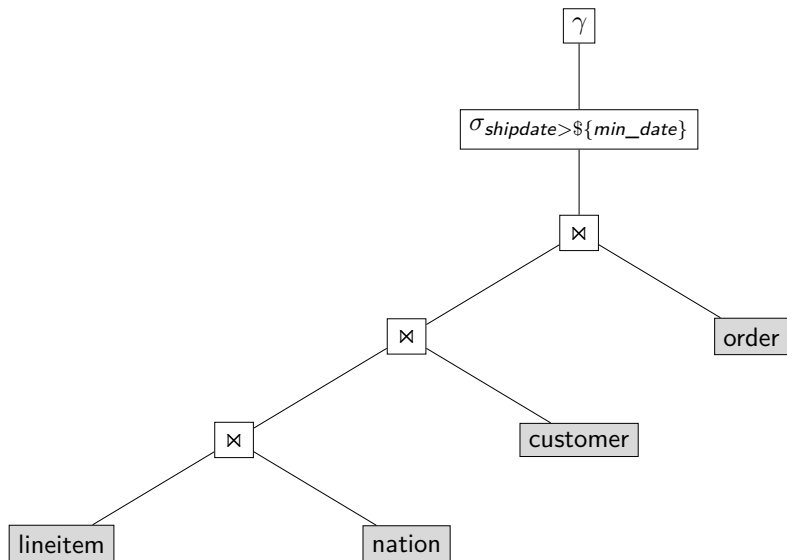
Example 1: Template query

```
select      n_name, avg(l_discount)
from        lineitem, customer, nation, order
where       l_orderkey = o_orderkey
and        c_custkey = o_custkey
and        c_nationkey = n_nationkey
and        l_shipdate > ${min_date}
group by   n_name
```

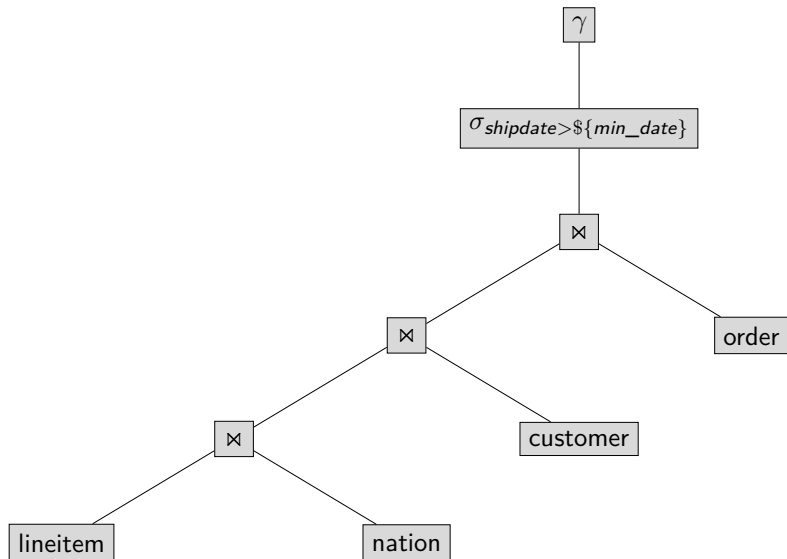
Example 1: Traditionally greedy plan



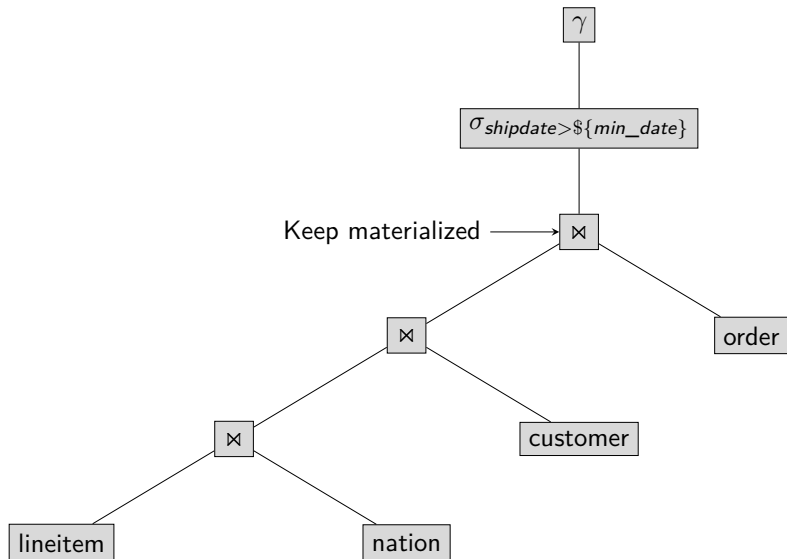
Example 1: Adapt to workload



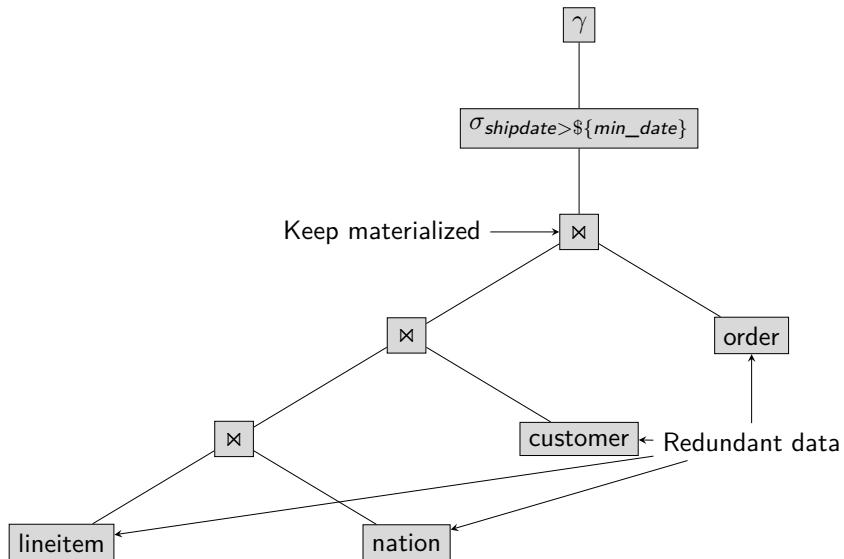
Example 1: Adapt to workload



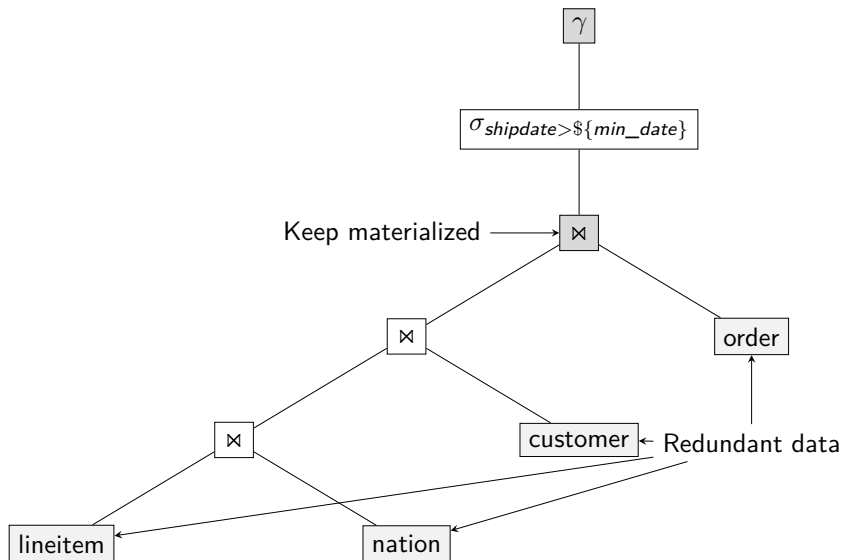
Example 1: Adapt to workload



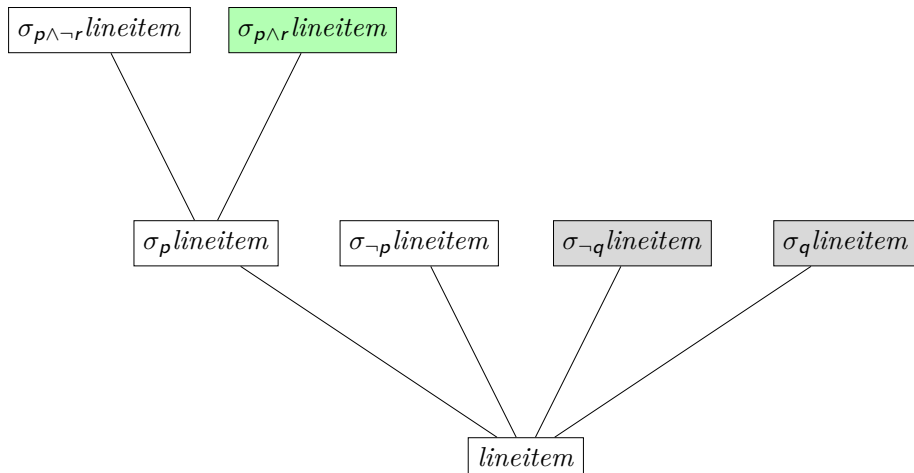
Example 1: Adapt to workload



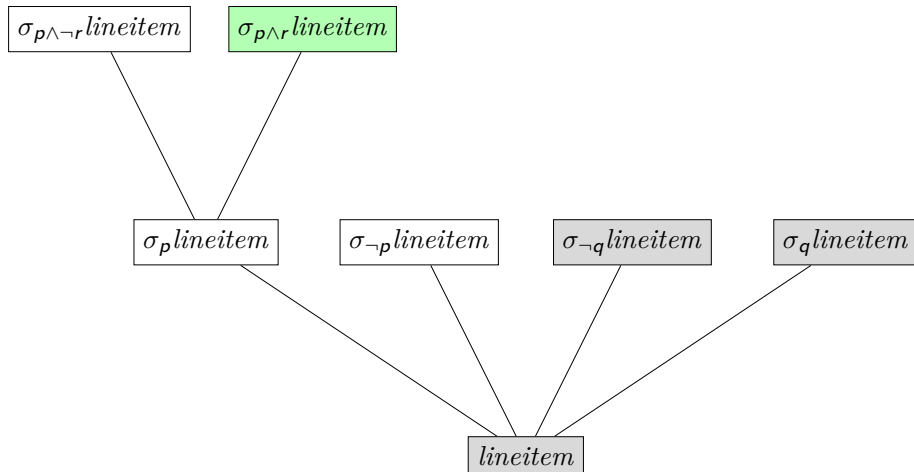
Example 1: Adapt to workload



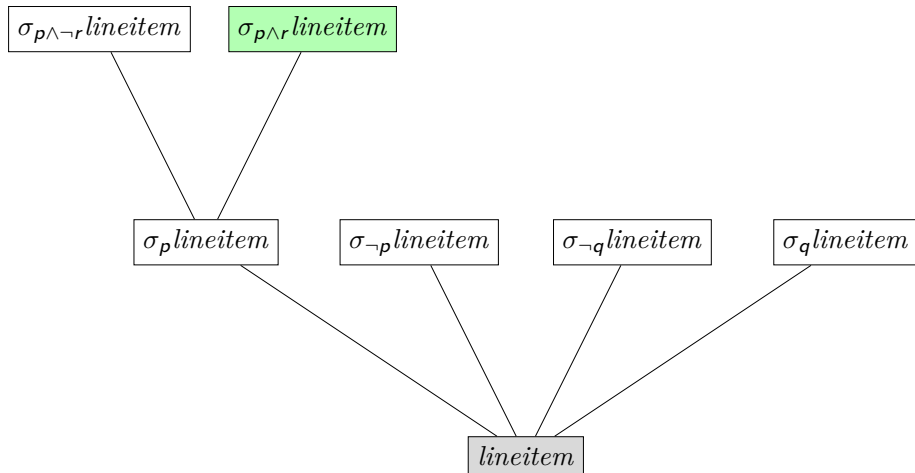
Example 2: Plan with missing primary data



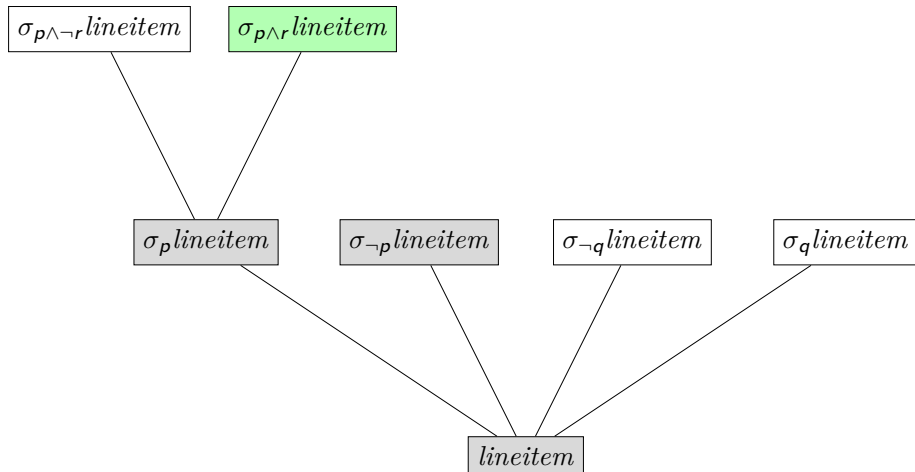
Example 2: Plan with missing primary data



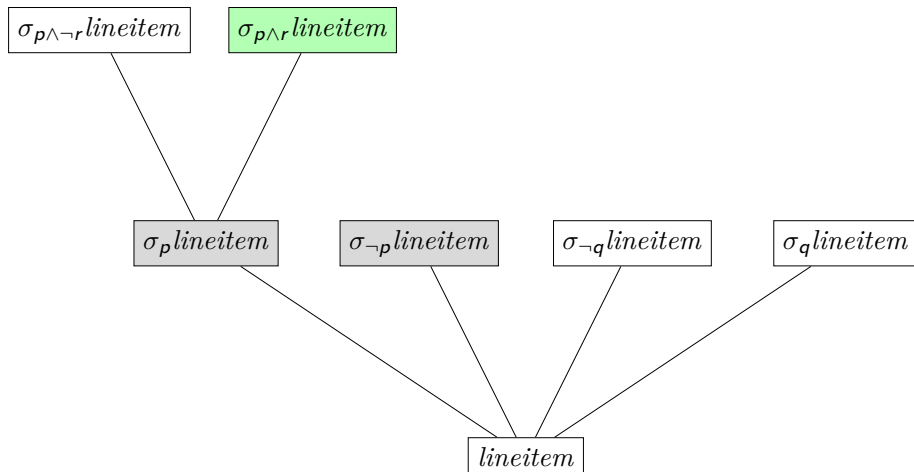
Example 2: Plan with missing primary data



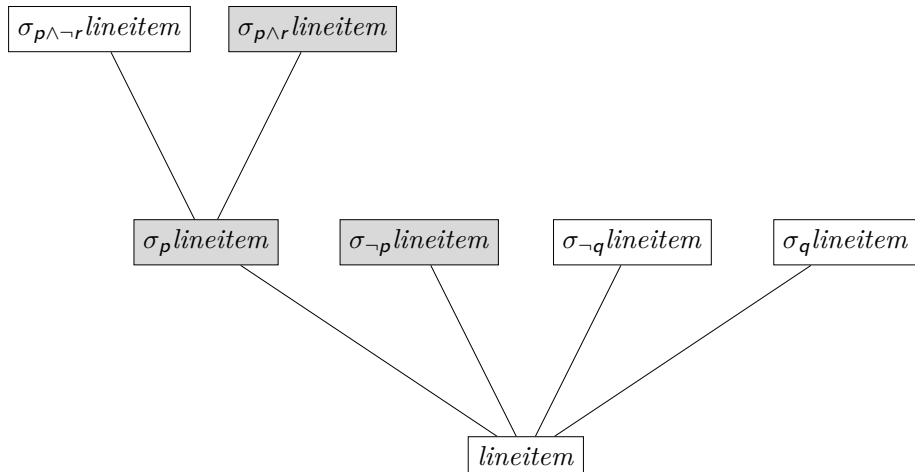
Example 2: Plan with missing primary data



Example 2: Plan with missing primary data



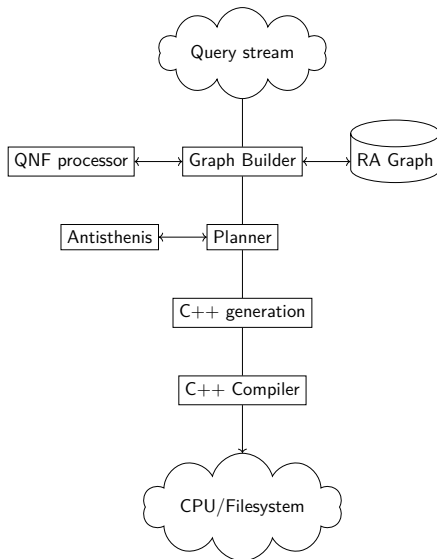
Example 2: Plan with missing primary data



The interesting components

- Graph management and query normal form representation
- Logical planning infrastructure
- Antisthenis: An incremental numeric evaluation system for cost estimation.
- Logical planning algorithm and garbage collector
- Code generation system.

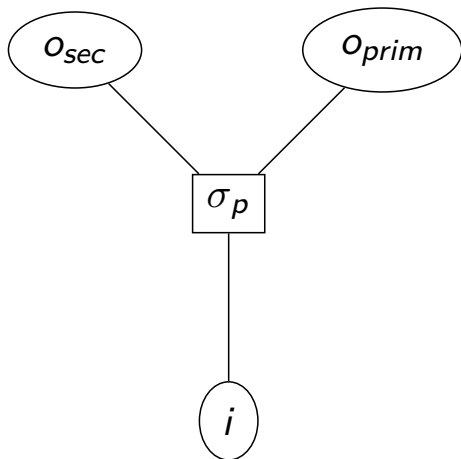
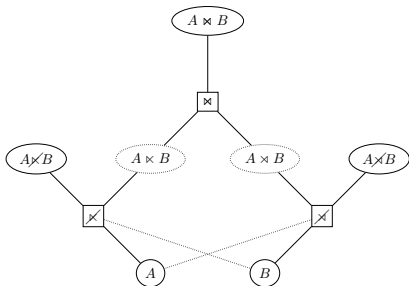
Architecture



Logical planning

- Bipartite query graph – RA operations/relations unified for all queries
- Join ordering enumeration
- QNF – $\pi\sigma(Q_1 \times Q_2 \times \dots)$ or $\gamma\sigma(Q_1 \times Q_2 \times \dots)$
- Relation shape propagation – cardinality, columns/types, unique subtuples

Reversible operators



Reversible operations



Physical planning

HCntT logic monad

Logic framework for “fair” traversal of the plan search space. Intrinsics:

- `a </> b`: Try the rest of the computation with `a` and if it fails try `b`.
- `once c`: try the continuation with values from `c` until one works and stick with that one.
- `halt n`: yield to a scheduler and assign priority `n` to the continuation.

Physical planning

Business logic

```
materialize n = unless (materialized n) $ do
  op ← inputOps n
  outputs ← possibleOutputs n op
  let inputs = inputsOf op
  -- Assuming we materialized the output, what is the cost of the
  -- outputs
  once (gc outputs)
  histCost ← withMaterialized outputs $ historicalCosts
  -- Stop and schedule this branch according to its cost
  halt (cost op + histCost + anticipatedCost inputs)
  -- Recursively materialize the input relations
  mapM materialize inputs
  registerPlan op
  mapM (setState Materialized) output
```

Antisthenis

Dynamically scheduled incremental computation

Materializability and cost inference are numerical operations:

- Input is mostly the same between runs: **incremental**.
- **Order of computation** highly affects the performance (eg absorbing elements, min).
- Self referential computations may appear earlier than the absorbing element.

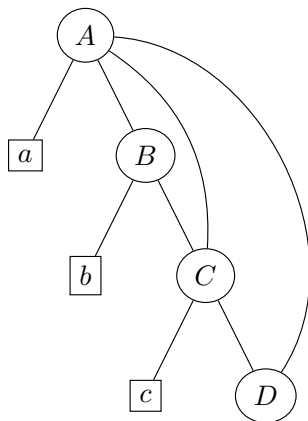
Antisthenis: Expression graphs

$$A = a + B + C + D$$

$$B = C \times b$$

$$C = D + c$$

$$D = 0$$



Antisthenis: Absorbing element

$$A = B \times C \times D$$

$$B = \sum_i i$$

$$C = 10 - 10$$

$$D = \sum_i i$$

Antisthenis: Early stopping – recursive expressions

While expressions may be self-referential, we can sometimes still evaluate them.

$$A = \min(B, C, D)$$

$$B = b_1 + b_2 \cdot D$$

$$C = c_1 + c_2 \cdot A$$

$$D = d_1 + d_2 \cdot B$$

$$b_1 = b_2 = d_1 = d_2 = 1$$

$$c_1 = 3$$

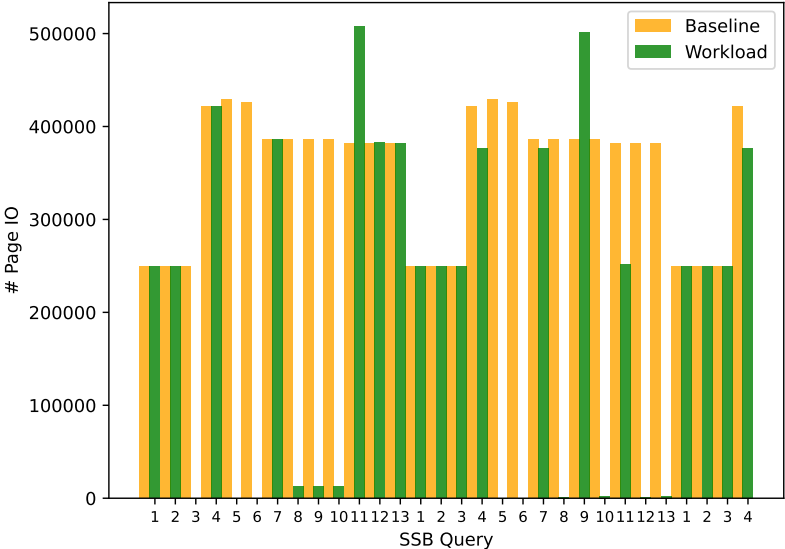
$$c_2 = 0$$

Data layout

Code generation

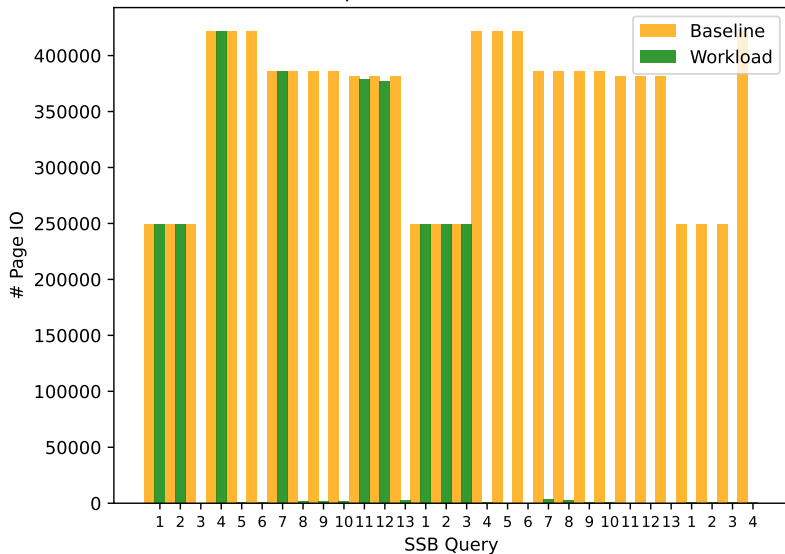
Evaluation: 23K pages budget

FluidDB performance on SSB TPC-H



Evaluation: 65K pages budget

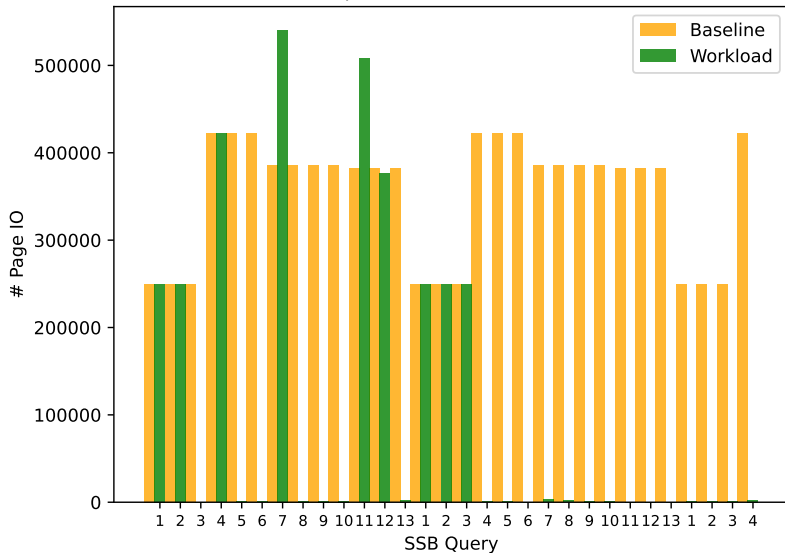
FluidDB performance on SSB TPC-H



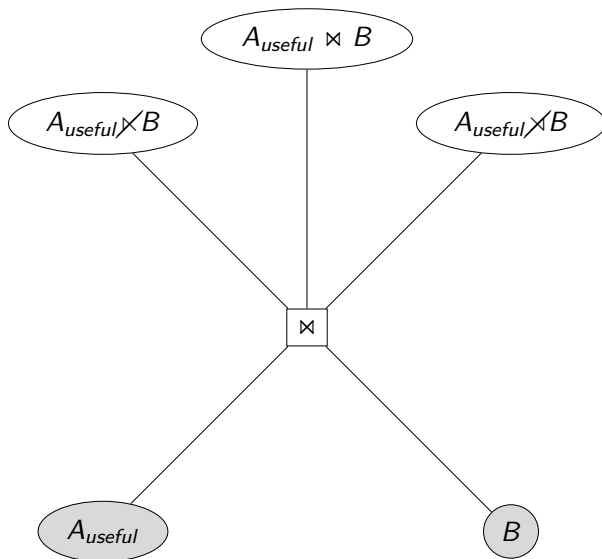
Evaluation: But ... 61K pages budget

lineorder is deleted at 6 because all join outputs were materialized

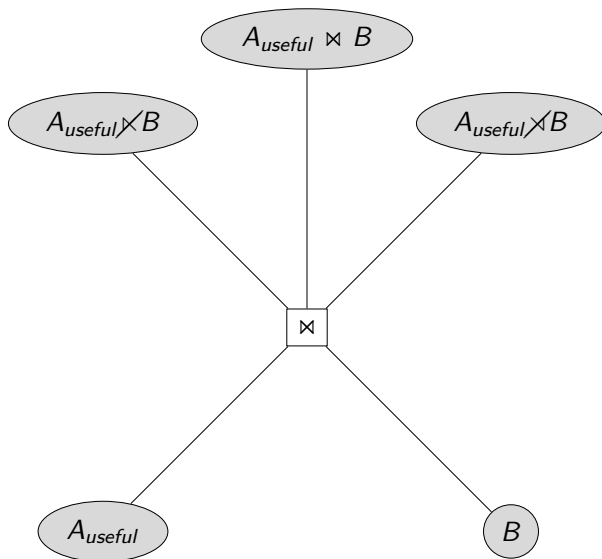
FluidDB performance on SSB TPC-H



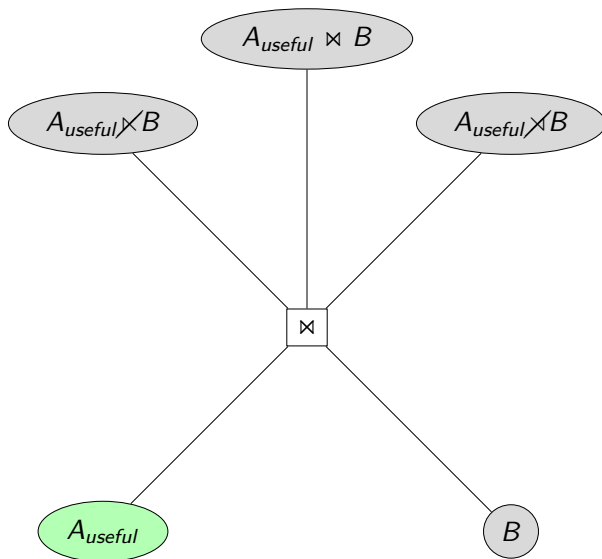
Plenty of memory



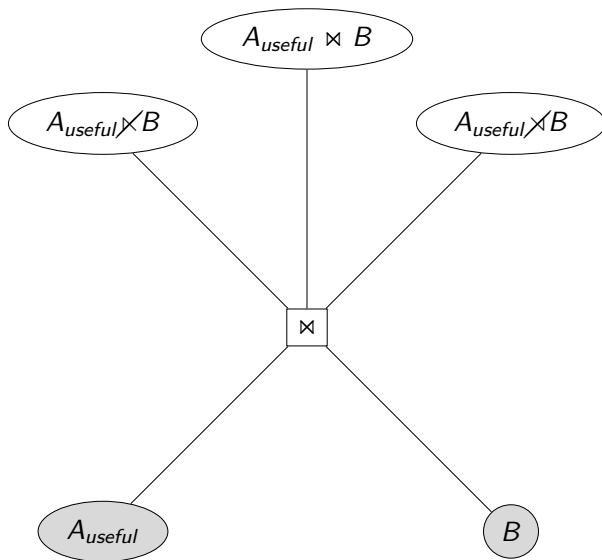
Plenty of memory



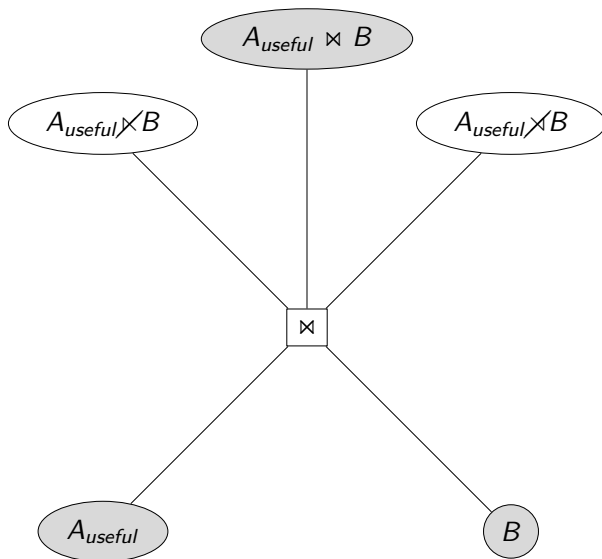
Plenty of memory



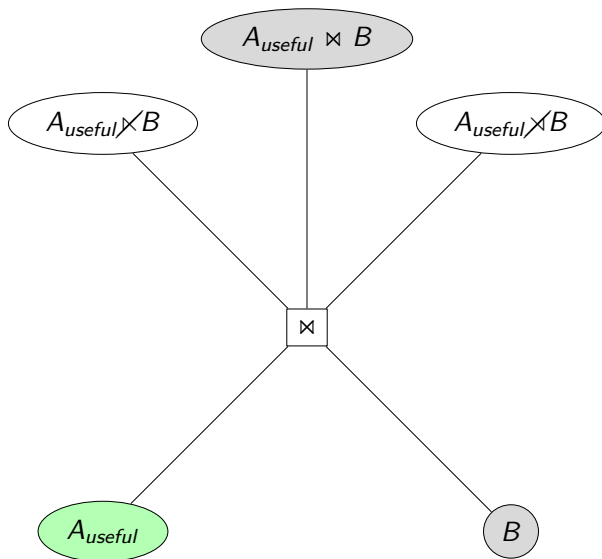
Being on a budget



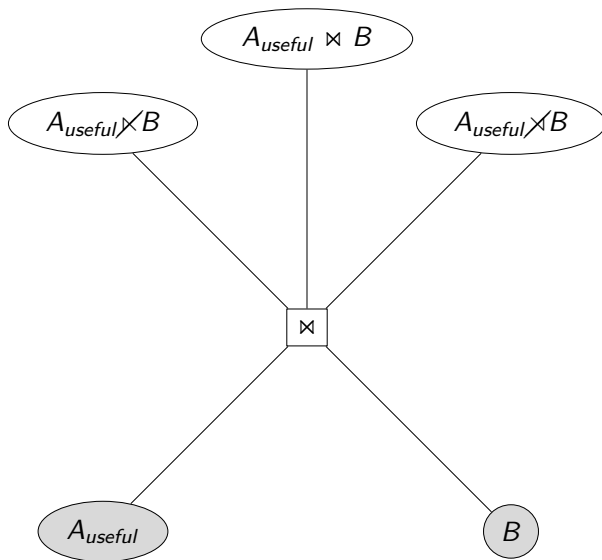
Being on a budget



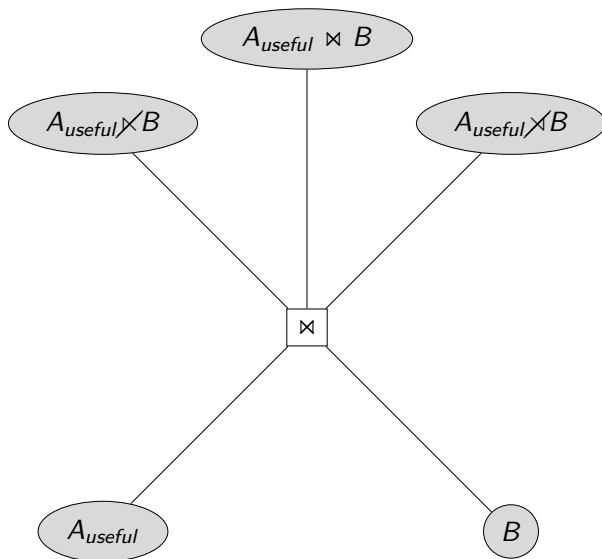
Being on a budget



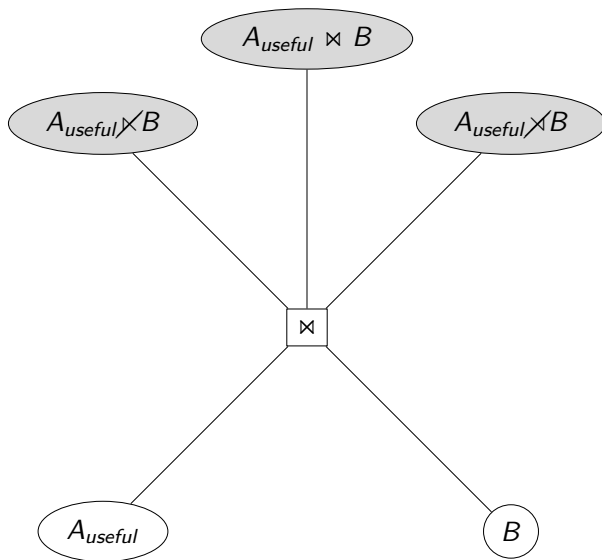
Having just enough rope



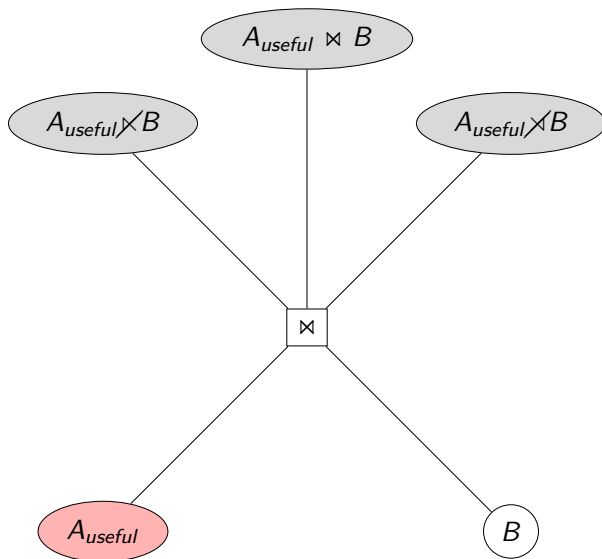
Having just enough rope



Having just enough rope



Having just enough rope



Cocclusions and future perspectives

- FluidB can efficiently use memory budget to store useful intermediate results.
- It would be interesting to:
 - ▶ Cardinality estimation is a major pain point for FluidB, the architecture is accomodating to propagation of statistics
 - ▶ Parallel query processing
 - ▶ Support updates
 - ▶ extend the algebra with index-building operators.
 - ▶ Drop the C++ compiler.