Flow-Centric Query Evaluation Pipelines

Victoria Dib, Andrew J. Mikalsen, Jaroslaw Zola, Oliver Kennedy
University at Buffalo, SUNY

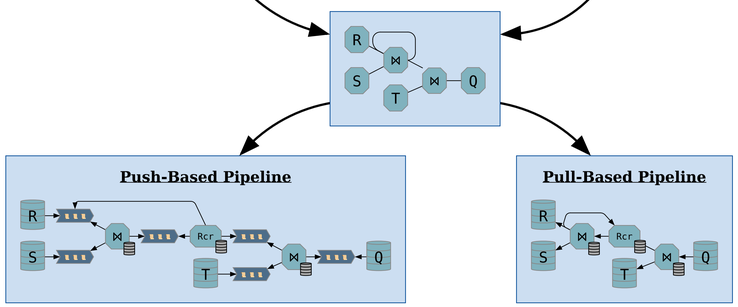
Query Evaluation Pipelines

SQL

```
WITH J AS
  SELECT R.A, S.B FROM R, S WHERE R.B = S.A UNION
  SELECT J.A, S.B FROM R, S WHERE J.B = S.A
SELECT * FROM J, T WHERE J.B = T.A
```

Datalog

```
J(A, B) :- R(A, C), S(C, B).
J(A, B) :- J(A, C), S(C, B).
Q(A, B, C) : J(A, C), T(C, B).
```



Pipeline Operator Implementation

Join (⋈)

```
def get_tuple():
  if lookup_r is None:
    lookup_r = build(R)
  for s in S:
    if r := lookup_r[s.A]:
      return (r, s)
```

```
def on_r(r):
  lookup_r[r.A] = r
  if s := lookup_s[r.A]:
    output(r, s)
def on_s(s):
  # Symmetric
```

Recursion (Rcr)

```
def get_tuple():
  for r in R:
    if not lookup_r[r]:
      lookup_r.insert(r)
      return r
```

```
def on_r(r):
  if not lookup_r[r]:
    lookup_r.insert(r)
    output(r)
```

Operator State Considered Harmful

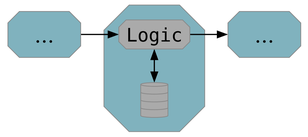
Many pipeline operators keep unbounded internal state!

- It's hard to re-use internal state across operators.
- It's hard to give the scheduler and planner insight into IO costs.
- It's hard to pipeline organizational work into upstream operators.
- It's hard to extend the system with new data structures.

We want to decouple relational state from the operator.

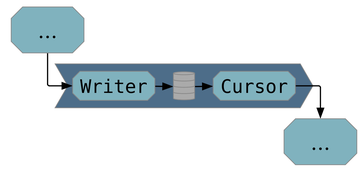
Decoupling Operators and State

Embedded State



Internal operator logic **both** writes to and reads from materialized relations

Decoupled State



Operators communicate through relation flows via writer/cursor abstractions

Cursor Properties

Operators declare the properties that they require of each input.

Property	Cursor	Data
Resettable	seek_head	-
Clustered[K]	seek_key	clustered on K
Sorted[K]	seek_key	sorted on K
Coalesced[K,⊕]		grouped by K
Diff		diff prev vers.

Examples

- **Join-Build**: Resettable, Clustered[K]
- **Join-Probe**: -
- **Print**: Coalesced[K]

Operator Properties

Operators declare the properties that they enforce in their outputs, or that they preserve from input to output.

Property	Data
Clustered[K]	clustered on K
Sorted[K]	sorted on K
Coalesced[K,⊕]	grouped by K

Examples

- **Base Rel**: Clustered, Coalesced
- **Join**: Probe Clustered → Clustered

Data Structures

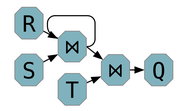
Each data structure provides support for a different set of properties.

Data Structure	Properties Enforced
Ring Buffer	-
Array	Resettable
Hash Table	Resettable, Clustered[K]
Agg. Hash Table	Resettable, Clustered[K], Coalesced[K, ⊕]
Multiversion B+ Tree	Resettable, Sorted[K], Diff

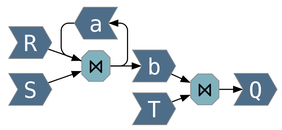
The planner picks data structures that enforce the union of all properties required by operators reading from the flow.

Compiling Flow-Centric Pipelines

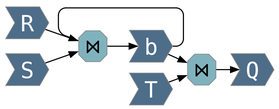
1. Logical Pipeline



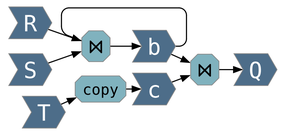
2. Instantiate Flows



3. Merge Flows



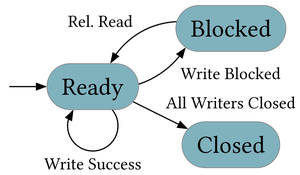
4. Inject Copies



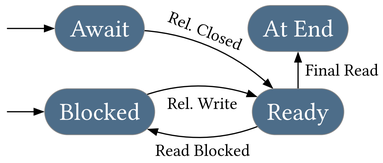
1. The planner starts with a logical pipeline as in operator-centric methods
2. Every edge and relation operator in the logical plan becomes a flow.
3. Flows that share sources and/or sinks, and who's sinks have compatible property requirements are merged.
4. Where necessary, copy operators are injected (e.g., to build indexes).
5. The planner selects a data structure for each internal flow. (not shown)

Running

Writer



Cursor



An operator is runnable iff all of its **writers** and **cursors** are Ready.

Draupnir

Draupnir is a work-in-progress **map-relational, out-of-core** datalog engine targeting large program analysis workloads (e.g., vulnerability detection in Chrome), and binary decompilation. **Draupnir enables *composable and scalable* program analyses on commodity hardware.**

The authors wish to acknowledge other contributors to Draupnir, including Arlen Cox, Andrew Hirsch, Ethan Canton, Krishna Sivakumar, Kamil Woskowiak, and Nick Brown.

https://git.odin.cse.buffalo.edu/Norn/Draupnir