Incremental View Maintenance in DuckDB

Database Architectures

Kriti Kathuria, Ilaria Battiston, Peter Boncz - Centrum Wiskunde & Informatica, Amsterdam

Assumptions

Let q ∈ {σ,Π, ⋈, γ} be a query,
t be a table,
Δ⁺t be insertions into t. Then,
V: q(t) and V`: q(t+**Δ⁺**t)

Goal

Define IVM invariant q* ∈ {σ*,Π*, ⋈*, γ*} : q*(V, **Δ⁺**t) = q(t+**Δ⁺**t) = V`

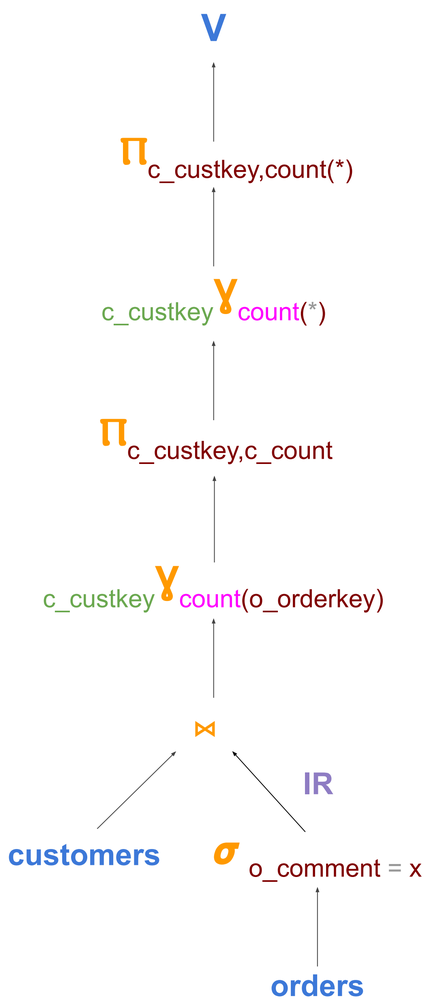
Additional property: {σ*,Π*, ⋈*, γ*} should be composable to allow chaining like with relational operators

ξ⁺: combine ΔV, V

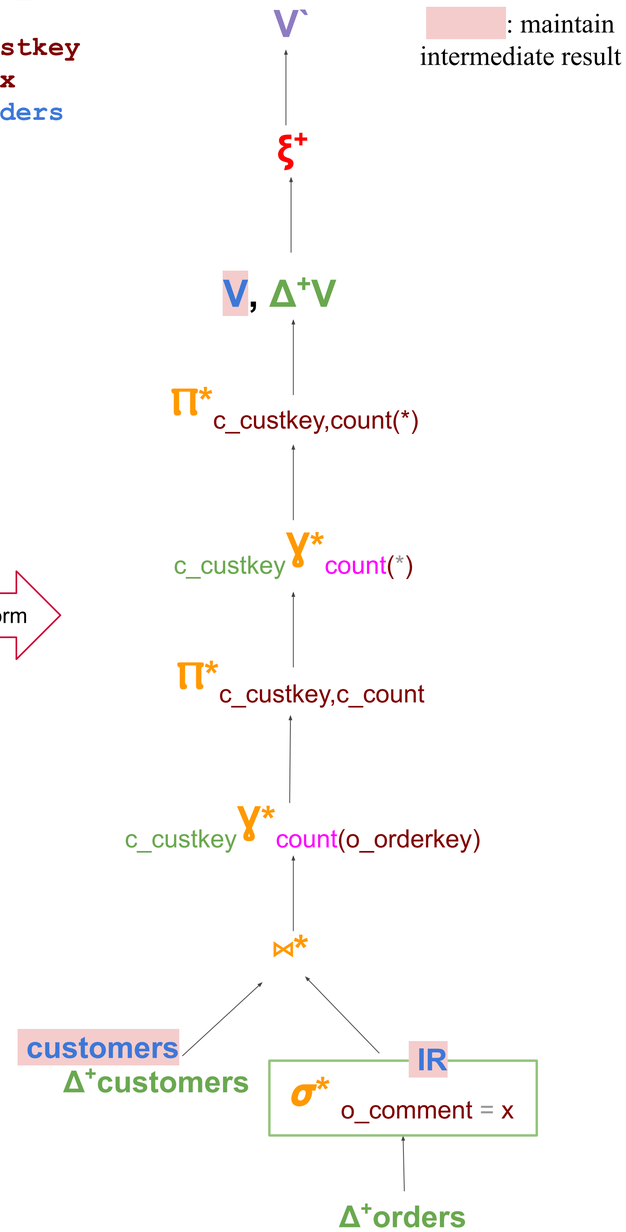
Define **ξ⁺**: **ξ⁺**(V, **Δ⁺**V) = V`

```
SELECT *
FROM    ( V UNION Δ⁺V )
OR
SELECT c_custkey,
       SUM(COUNT(*))
FROM    ( V UNION Δ⁺V )
GROUP BY c_custkey
```

```
SELECT c_count, COUNT(*)
FROM   (SELECT c_custkey,
               COUNT(o_orderkey) AS c_count
        FROM   customer JOIN orders
                 ON c_custkey = o_custkey
                    AND o_comment = x
        GROUP  BY c_custkey) AS c_orders
GROUP  BY c_count;
```



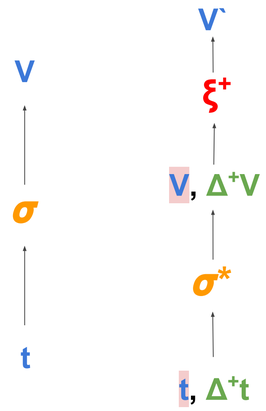
IVM transform



: maintain intermediate result

IVM Rules

- σ*, Π*, γ* remain the same as σ, Π, γ
- ⋈*: (Δl⋈Δr ∪ Δl⋈r ∪ l⋈Δr)
- Each IVM rule also consists of an upsert **ξ⁺** operator.



- Optimization: Depending on query structure, maintain specific intermediate results

Future Work

- Formal specification of IVM rules
- Performant maintenance of intermediate results for join and min/max processing
- Upsert op for deleted deltas
- Qualitative and quantitative evaluation

1 - https://github.com/duckdb/duckdb 2 - https://dbtoaster.github.io/ 3 - DBSP: https://dl.acm.org/doi/10.14778/3587136.3587137