DataTalks: Streaming Processing in SQL with RisingWave

Noel Kwan, RisingWave Labs, March 2024

About Me

- Database Kernel Engineer at RisingWave, building a cost-effective and scalable database solution designed to streamline data processing and query serving.
- Focus on Performance & Reliability, e.g. SqlSmith and Backfill
- Previously studied at NUS, focus area in Programming Languages.

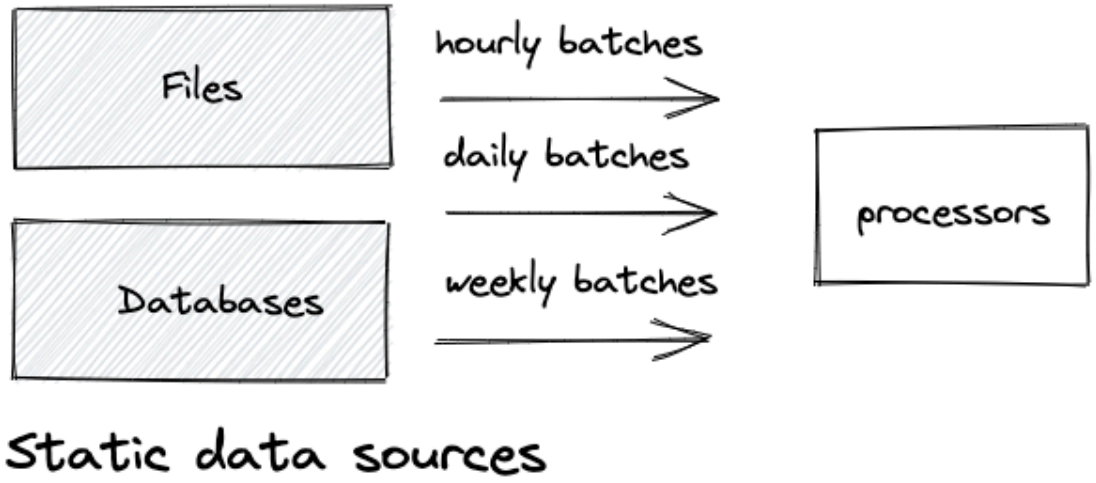
Agenda

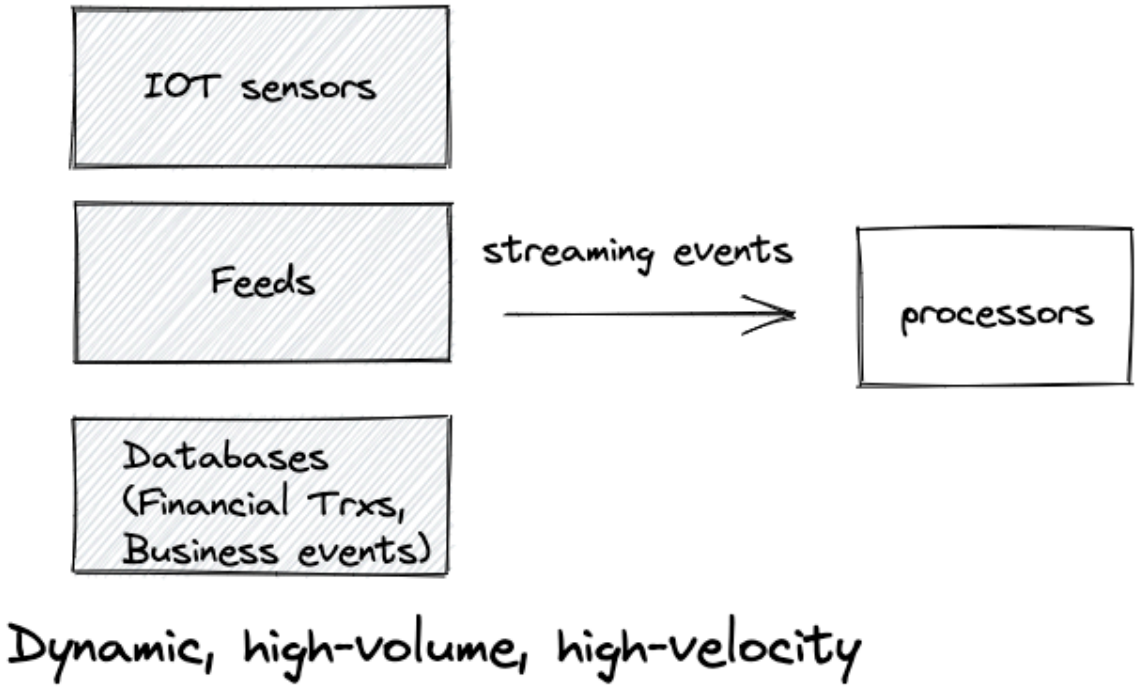
1. Why Stream Processing?
2. What is RisingWave?
3. Data Ingestion and Delivery
4. Stateless / Stateful Computation
5. Hands-on Project

Agenda

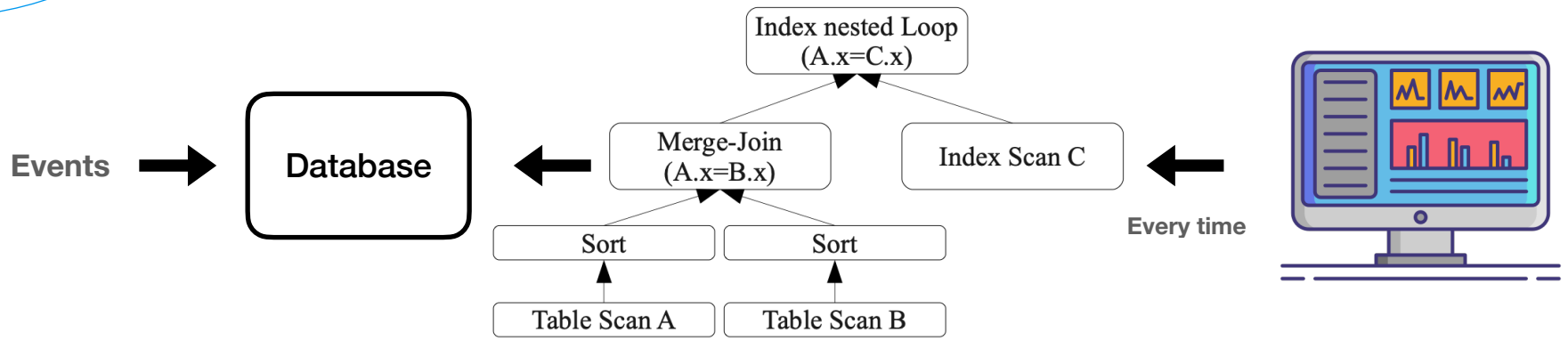
1. Why Stream Processing?

Batch vs. Stream (Data Ingestion)

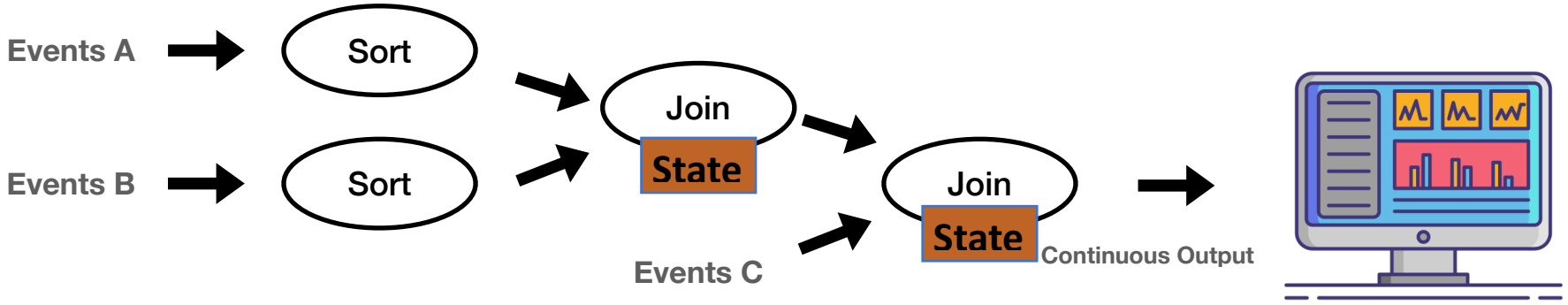




Batch vs. Stream (Data Processing Paradigm)




```
SELECT * FROM A JOIN B ON A.x = B.x JOIN C ON C.x = A.x;
```



Batch or Stream

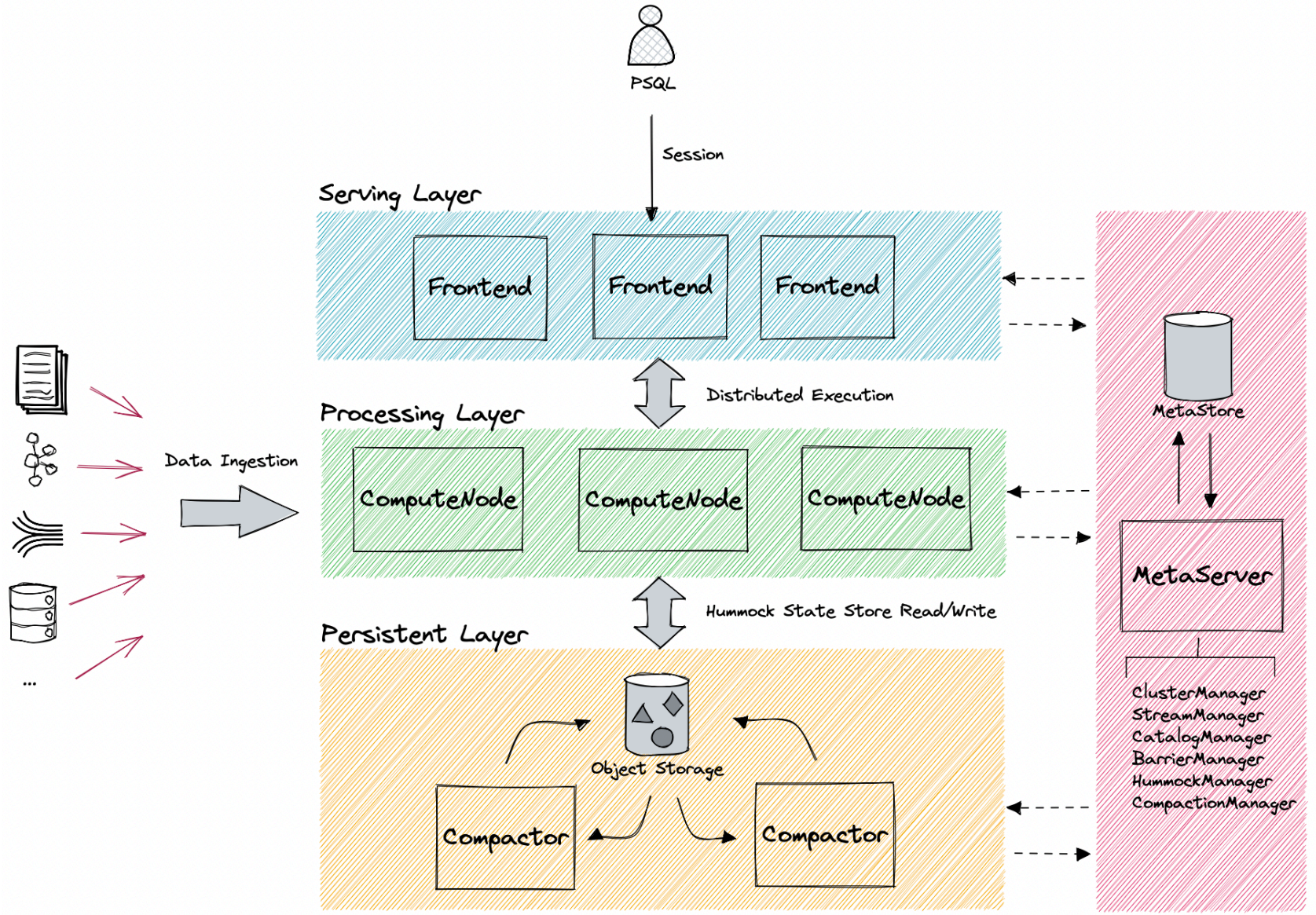
- How would you analyze social media data for sentiment analysis?
- How would you monitor the performance of a fleet of IoT devices?

Agenda

1. Why Stream Processing?
2. What is RisingWave?
3. Data Ingestion and Delivery
4. Stateless / Stateful Computation
5. Hands-on Project

2 What is RisingWave?

… But Purposely Built for Data Movement



- **Frontend**: Parsing, validation, optimization, and answering the results of each individual query.
- **ComputeNode**: Executing the optimized query plan.
- **Compactor**: Executing the compaction tasks for our storage engine.
- **MetaServer**: The central metadata management service. There are multiple sub-components running in MetaServer. Key thing to highlight is that the MetaServer is responsible for the scheduling and monitoring of streaming jobs.

Streaming Processing Engines Properties

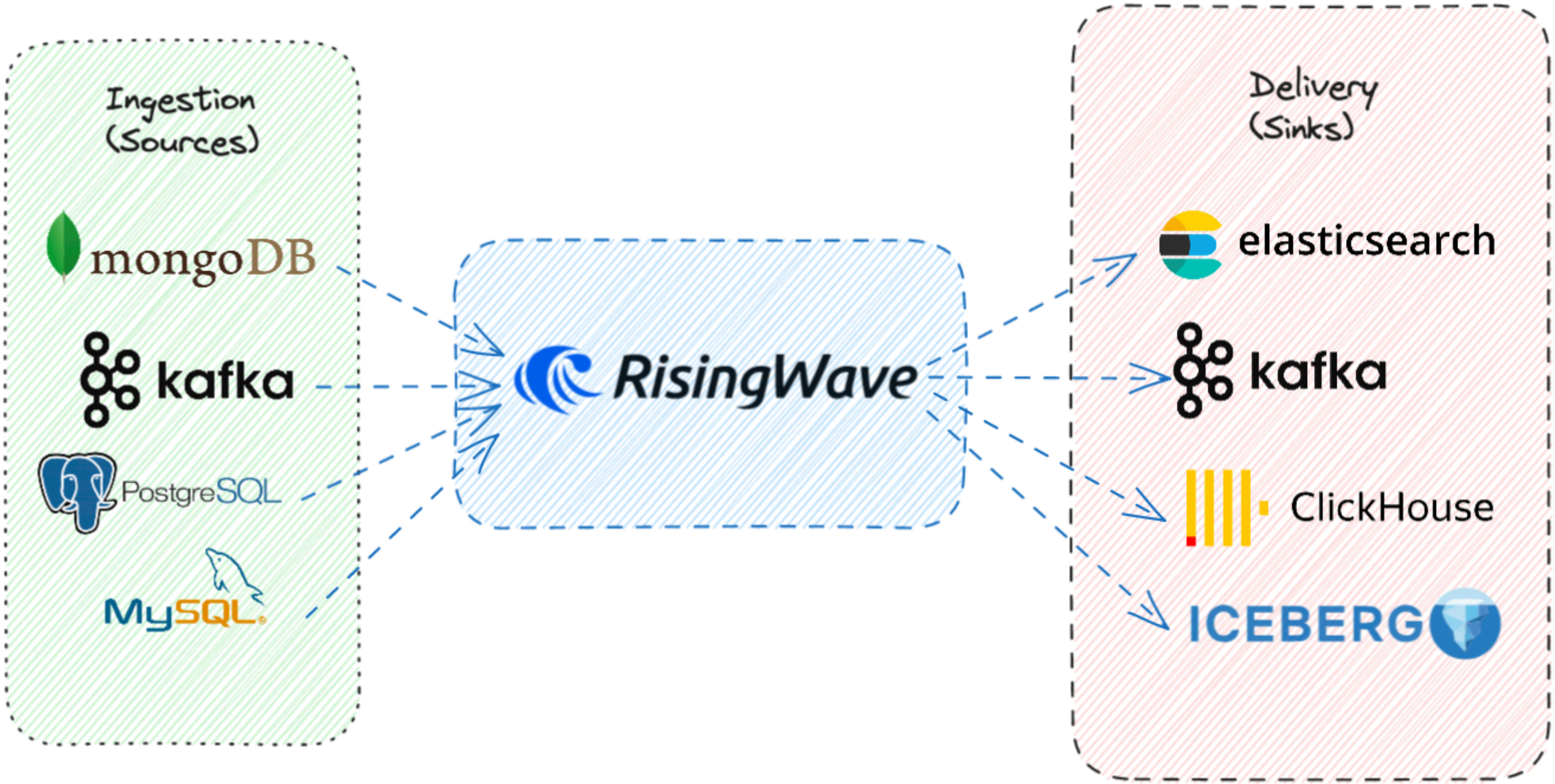
Criteria	RisingWave
Processing model	Both Stream processing and Batch processing
Latency	Low (seconds - milliseconds)
Throughput	High
Fault tolerance	Exactly-once semantics with checkpoints and snapshots
API support	SQL only
Storage system	Internal storage system based on Hummock tiered storage.
State management	Advanced (supports stateful queries with incremental view maintenance and materialized views)

Agenda

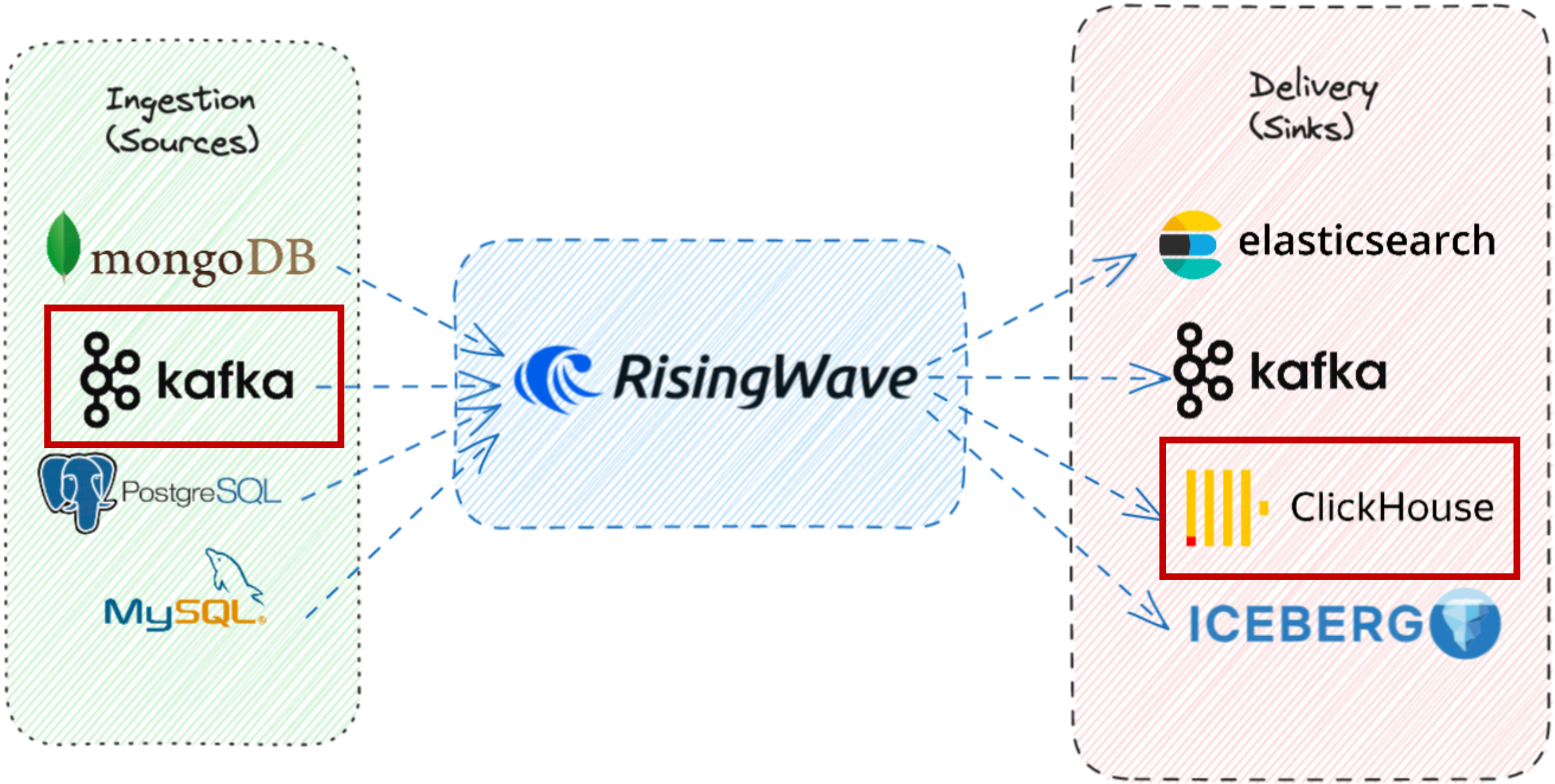
1. Why Stream Processing?
2. What is RisingWave?
3. Data Ingestion and Delivery
4. Stateless / Stateful Computation
5. Hands-on Project

3 Data Ingestion and Delivery

Data Ingestion and Delivery



Data Ingestion and Delivery



Agenda

1. Why Stream Processing?
2. What is RisingWave?
3. Data Ingestion and Delivery
4. Stateless / Stateful Computation
5. Hands-on Project

Agenda

4 Stateless / Stateful Computation

Stateless Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m1 AS SELECT v1 + 1 FROM t WHERE v1 > 0;
```

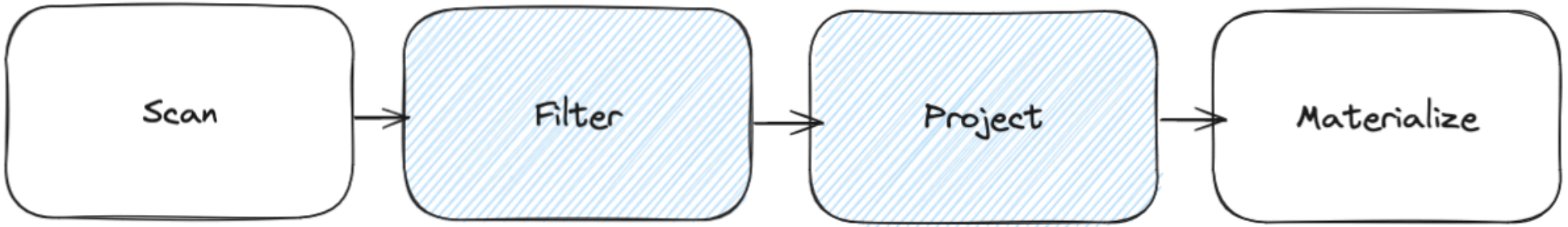
Stateless Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m1 AS SELECT v1 + 1 FROM t WHERE v1 > 0;
```

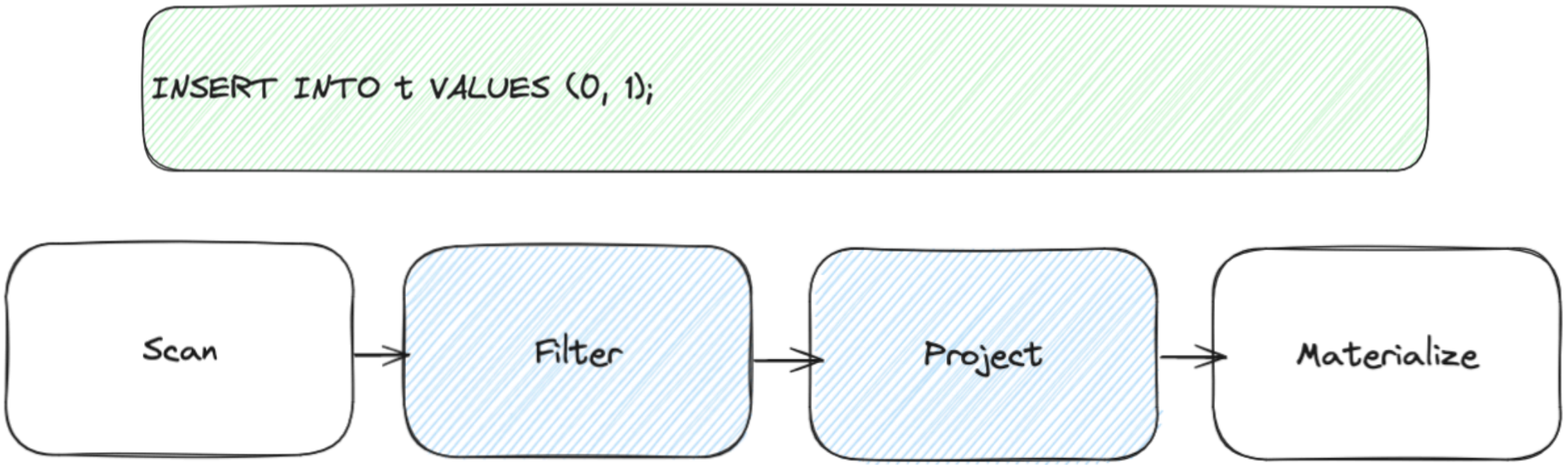
```
                                    QUERY PLAN
------------------------------------------------------------------------------------------
 StreamMaterialize {
     columns: [v1, t._row_id(hidden)],
     stream_key: [t._row_id],
     pk_columns: [t._row_id],
     pk_conflict: NoCheck
 }
  └─StreamProject { exprs: [(t.v1 + 1:Int32) as $expr1, t._row_id] }
    └─StreamFilter { predicate: (t.v1 > 0:Int32) }
      └─StreamTableScan { table: t, columns: [v1, _row_id] }
(4 rows)
```

Stateless Computation

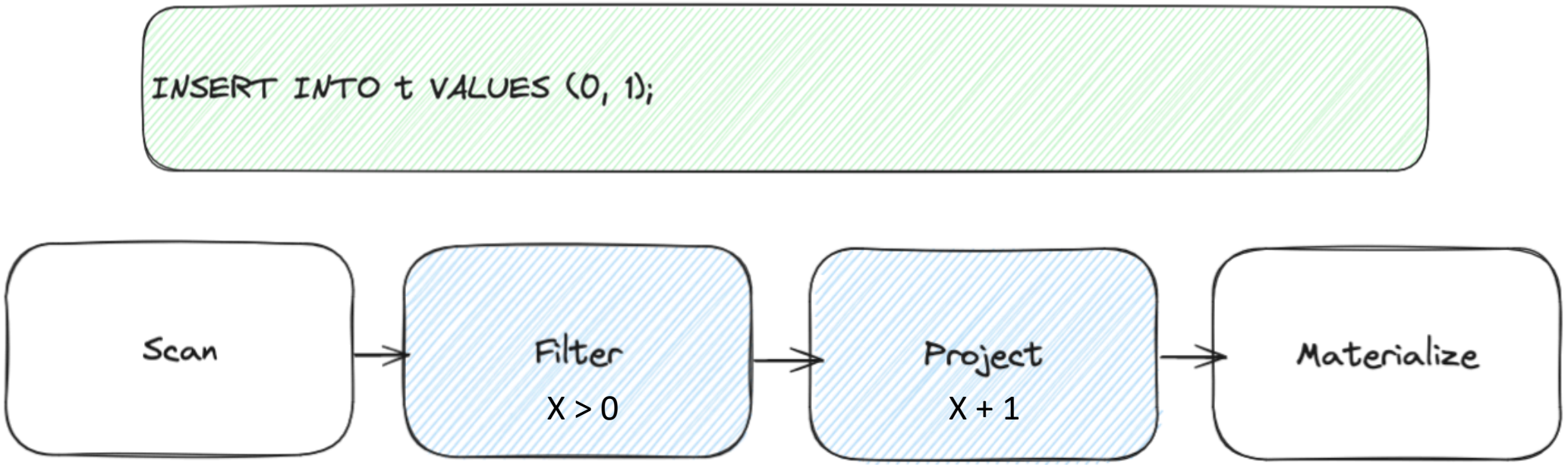
```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m1 AS SELECT v1 + 1 FROM t WHERE v1 > 0;
```



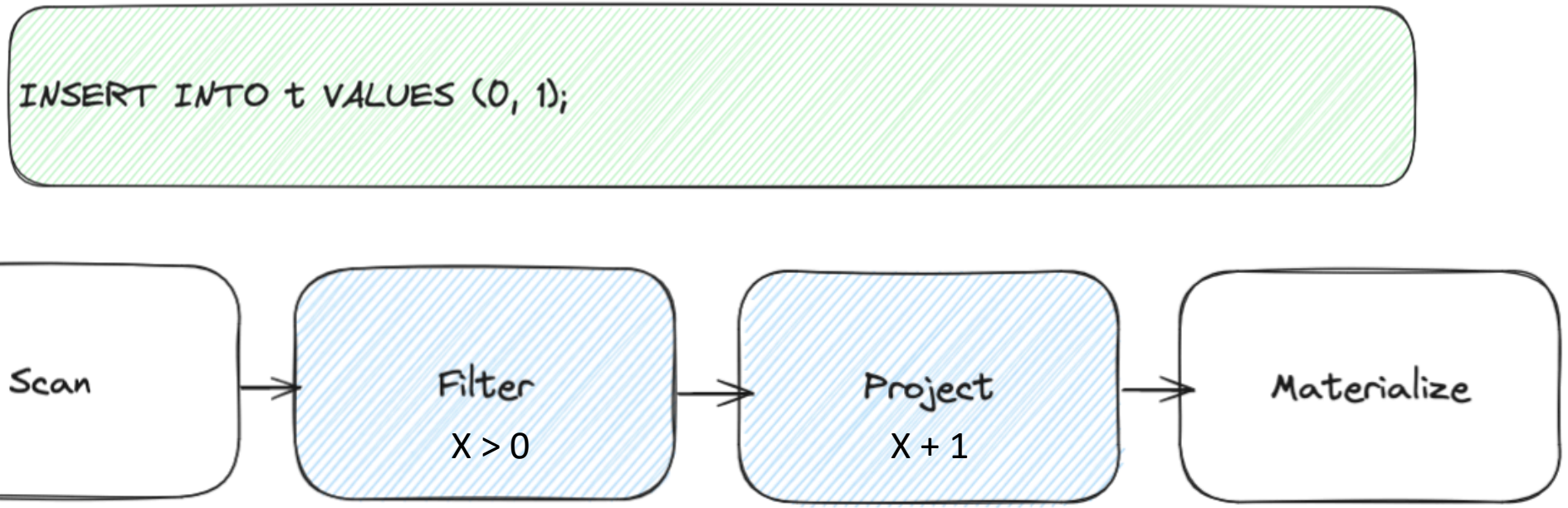
Stateless Computation



Stateless Computation

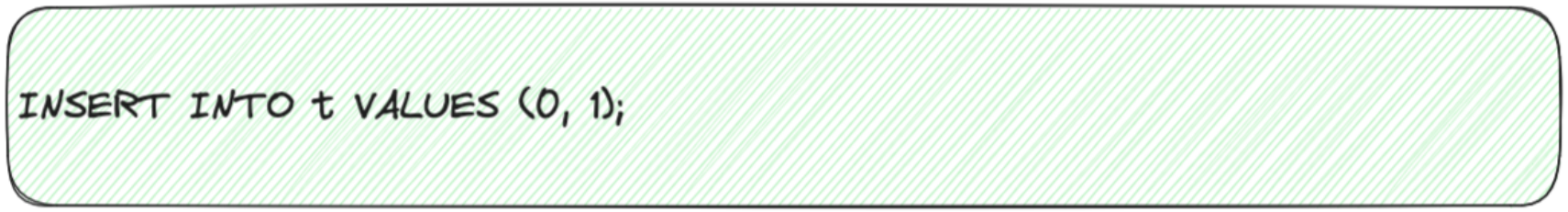


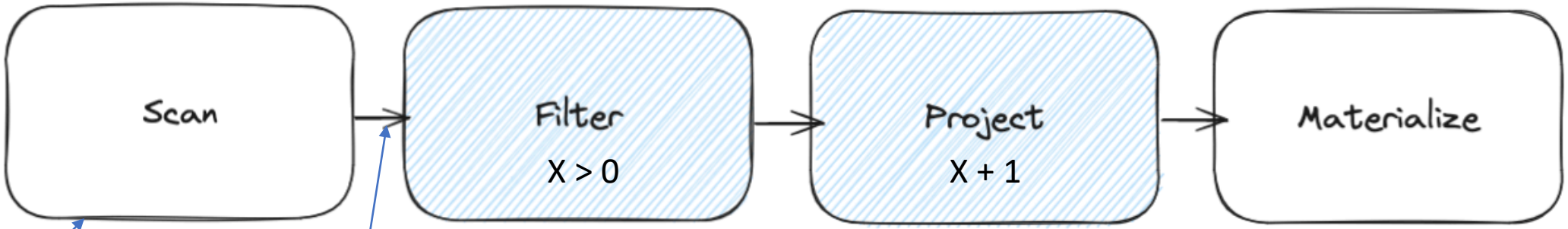
Stateless Computation



Op	V1
+	0
+	1

Stateless Computation

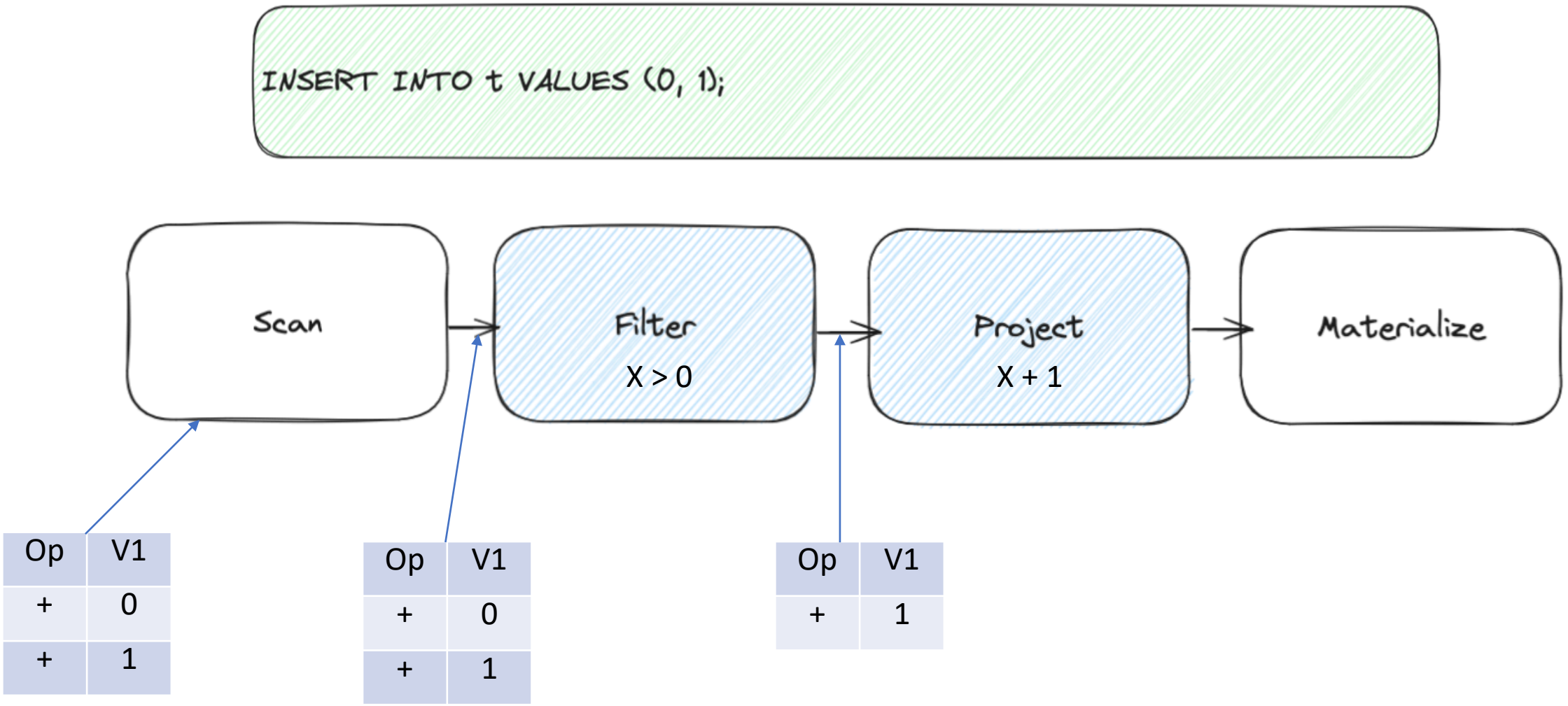




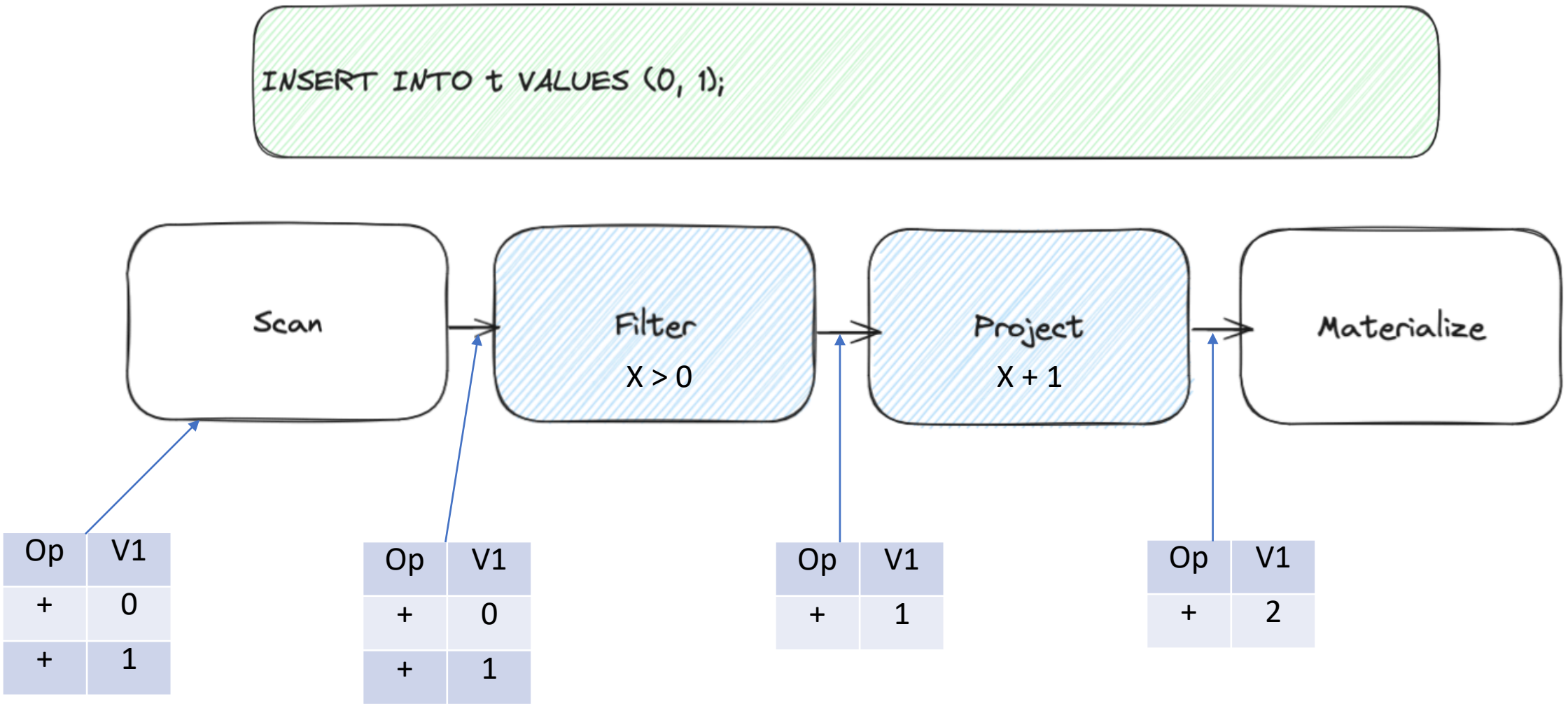
Op	V1
+	0
+	1

Op	V1
+	0
+	1

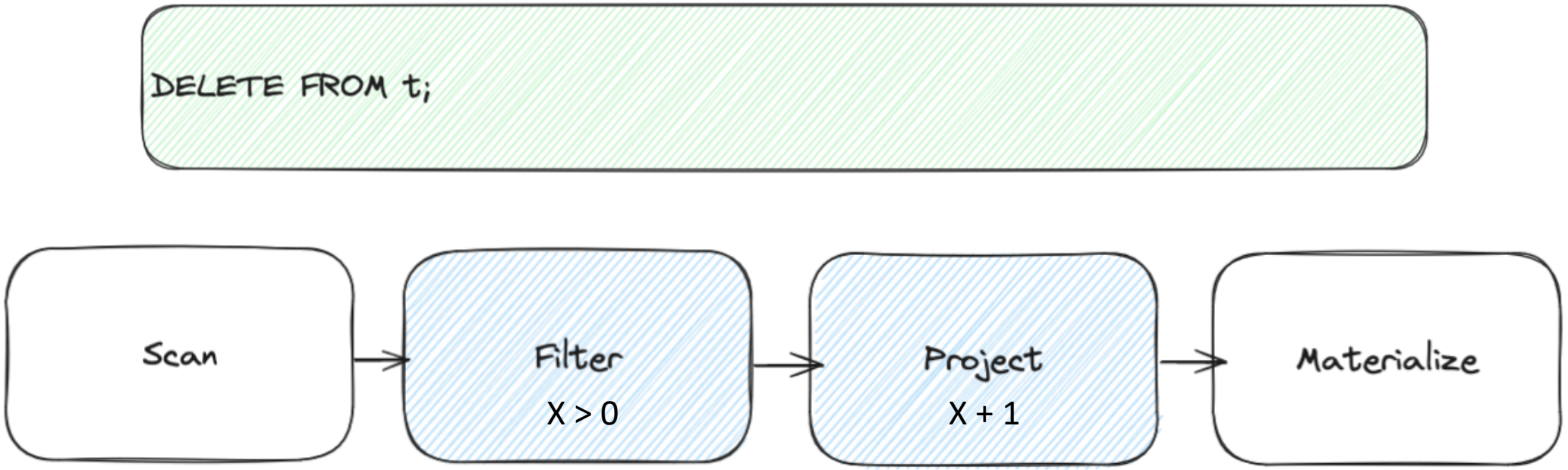
Stateless Computation



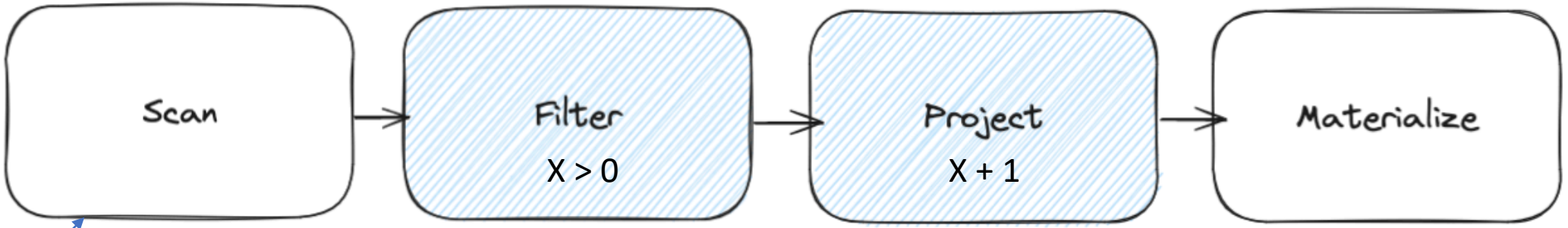
Stateless Computation



Stateless Computation

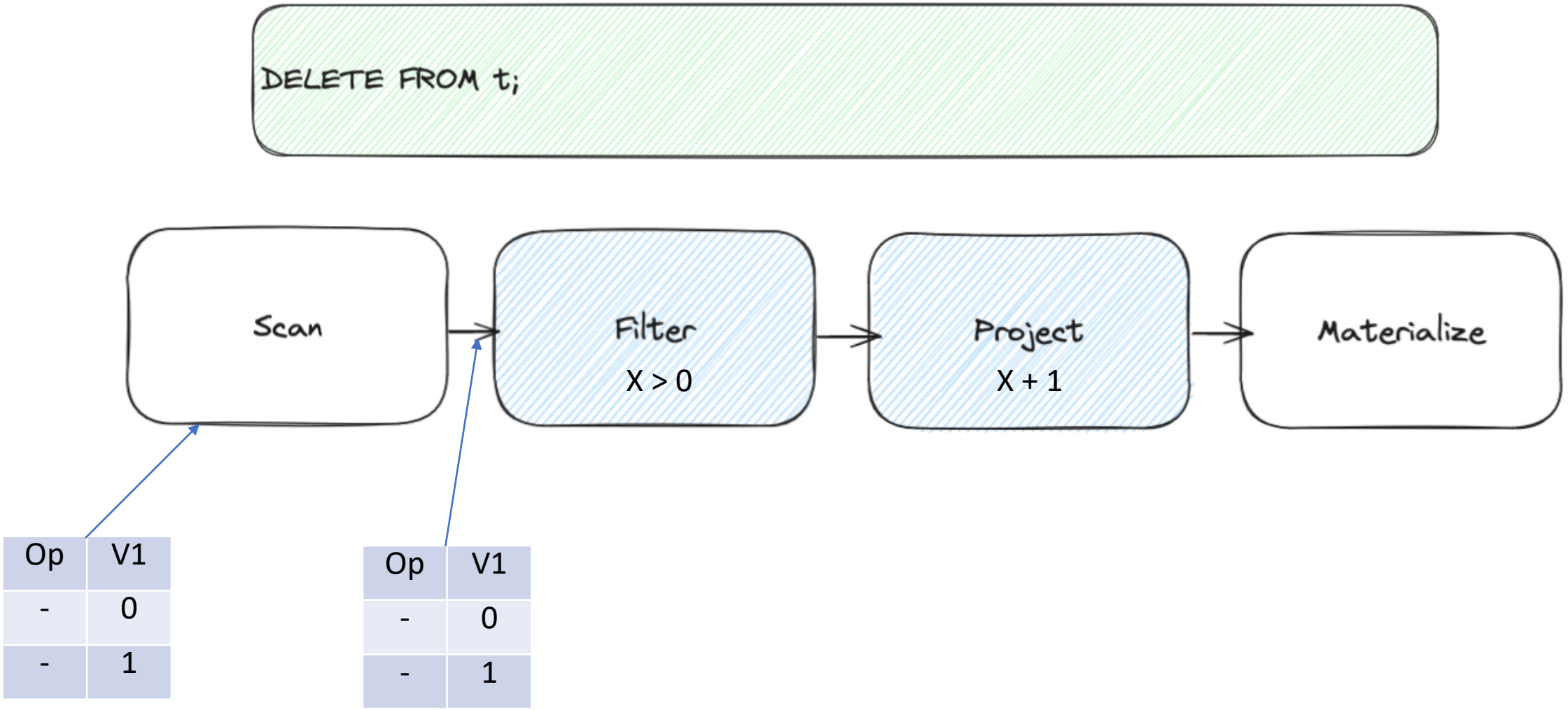


Stateless Computation

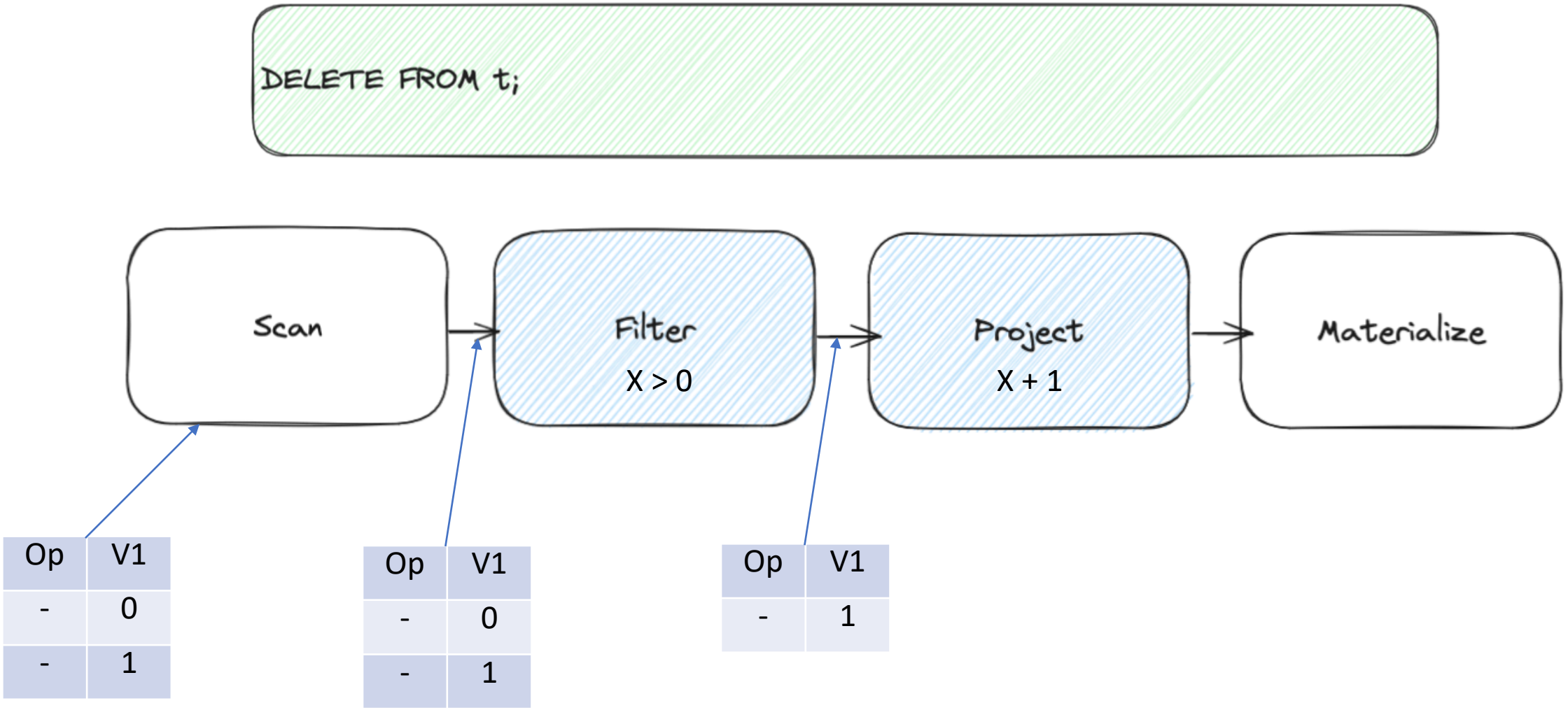


Op	V1
-	0
-	1

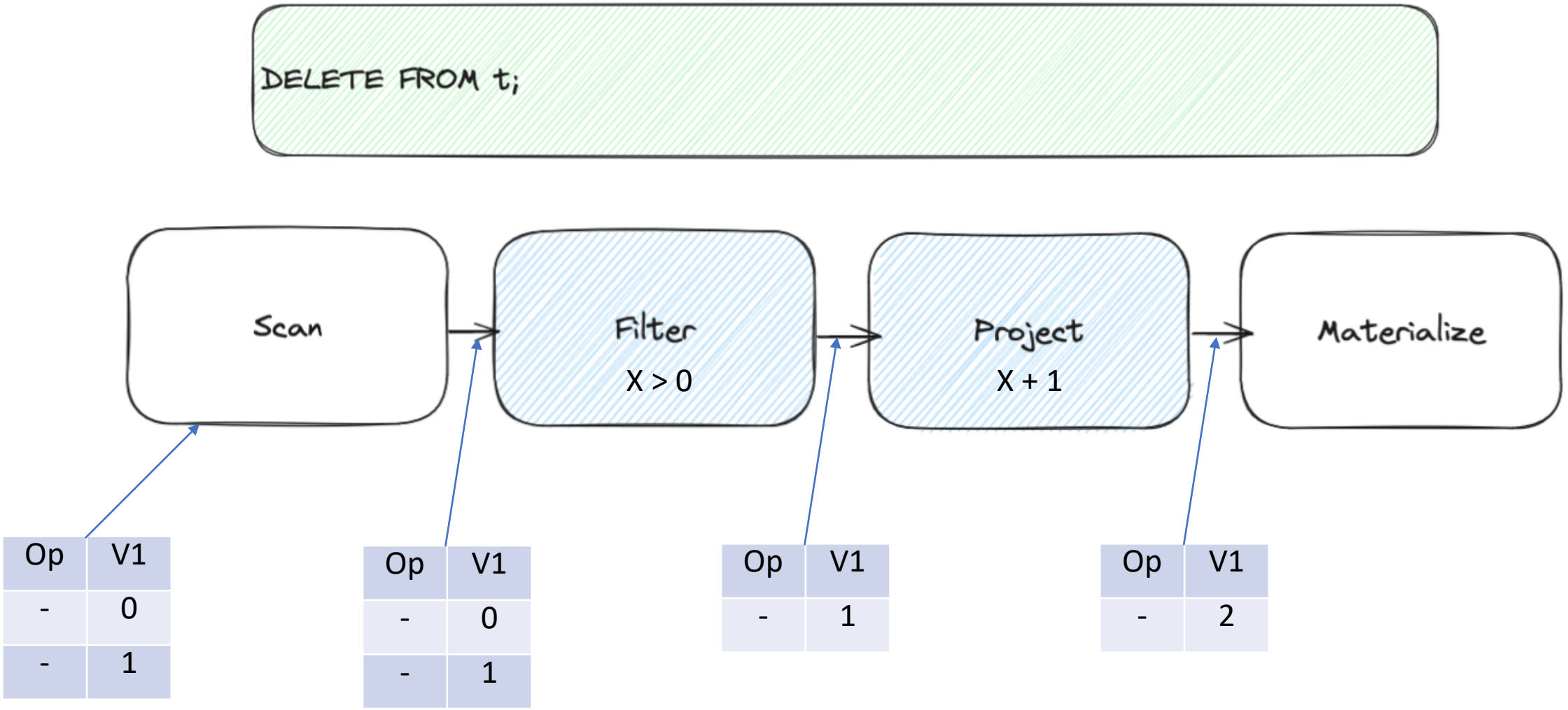
Stateless Computation



Stateless Computation

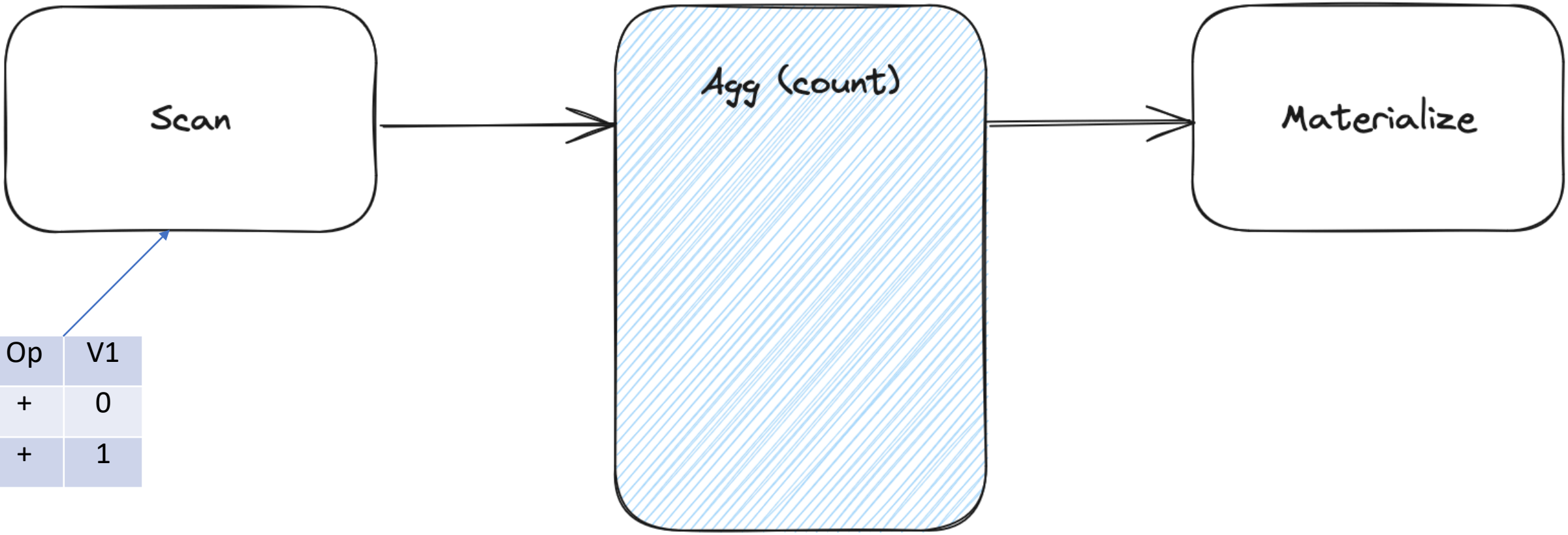


Stateless Computation



Stateful Computation

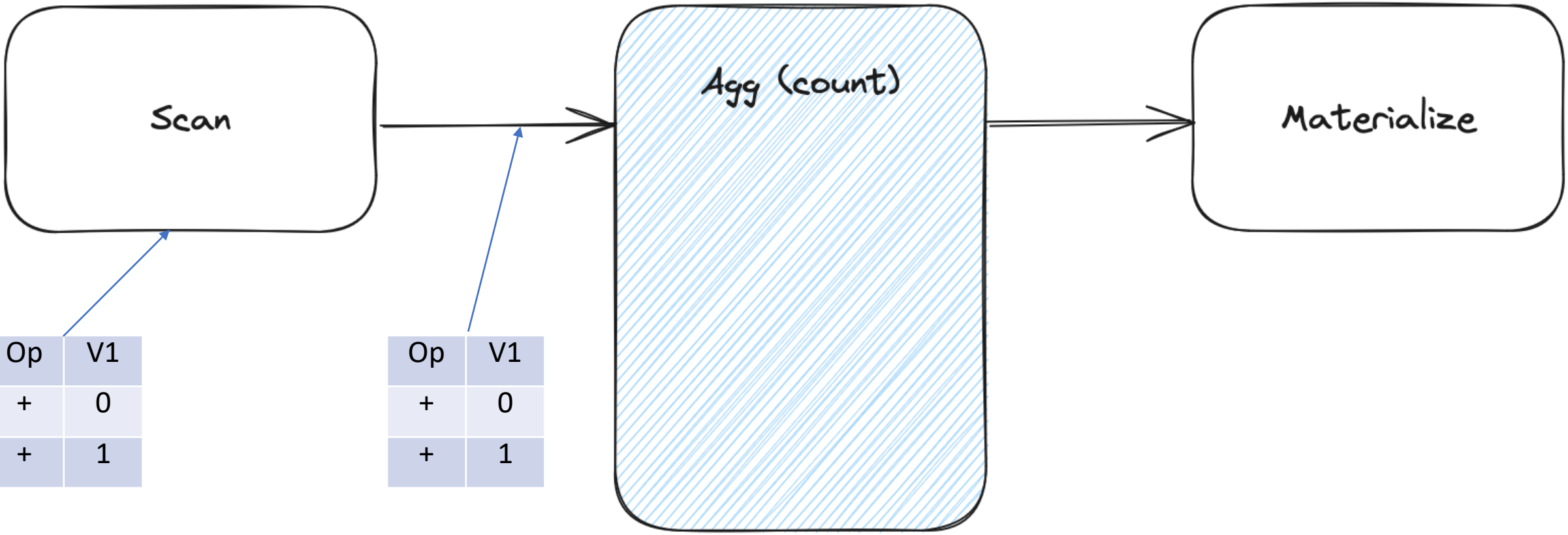
```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m2 AS SELECT count(*) FROM t;
```



Op	V1
+	0
+	1

Stateful Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m2 AS SELECT count(*) FROM t;
```

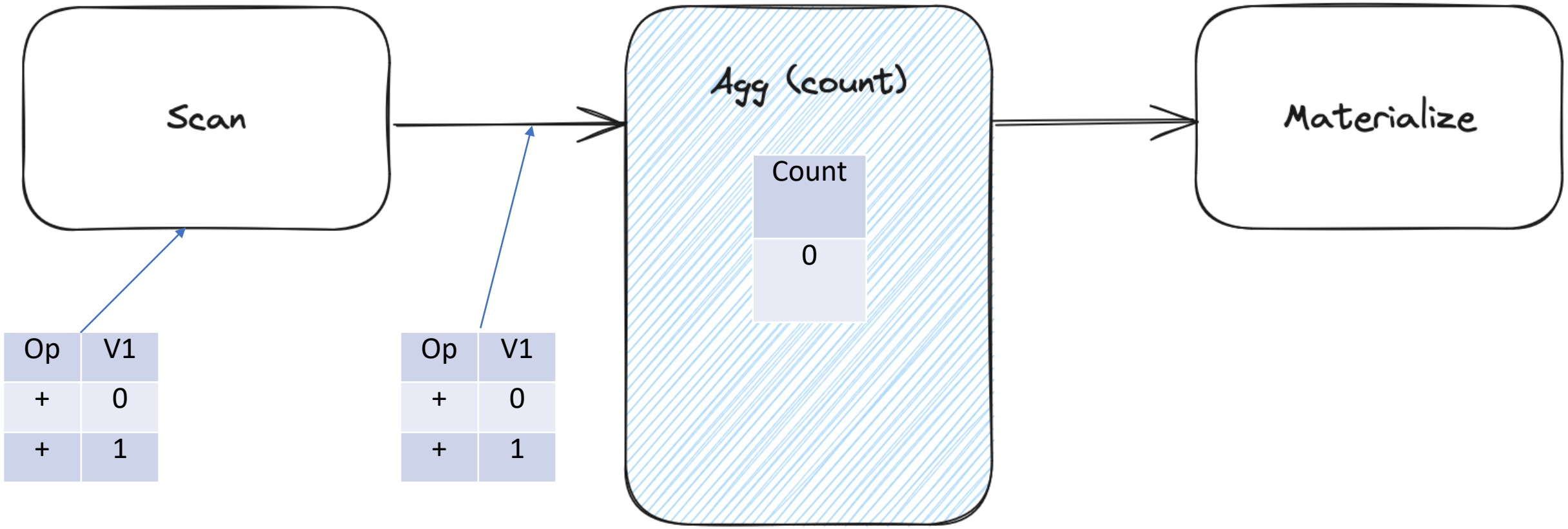


Op	V1
+	0
+	1

Op	V1
+	0
+	1

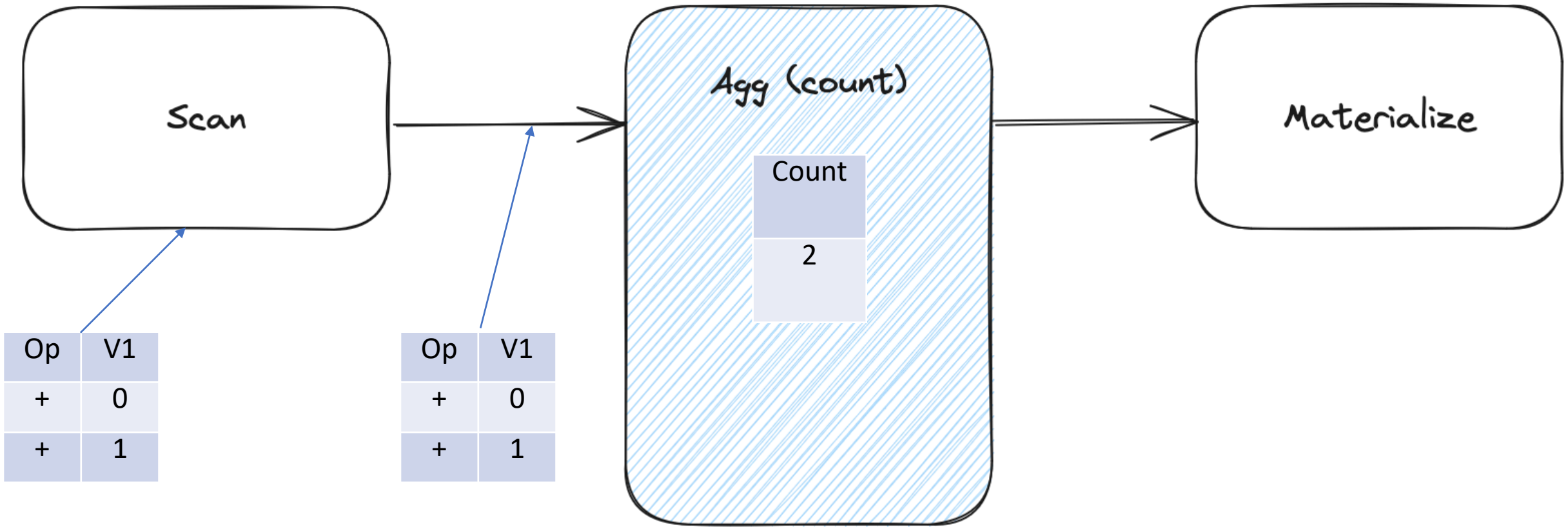
Stateful Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m2 AS SELECT count(*) FROM t;
```



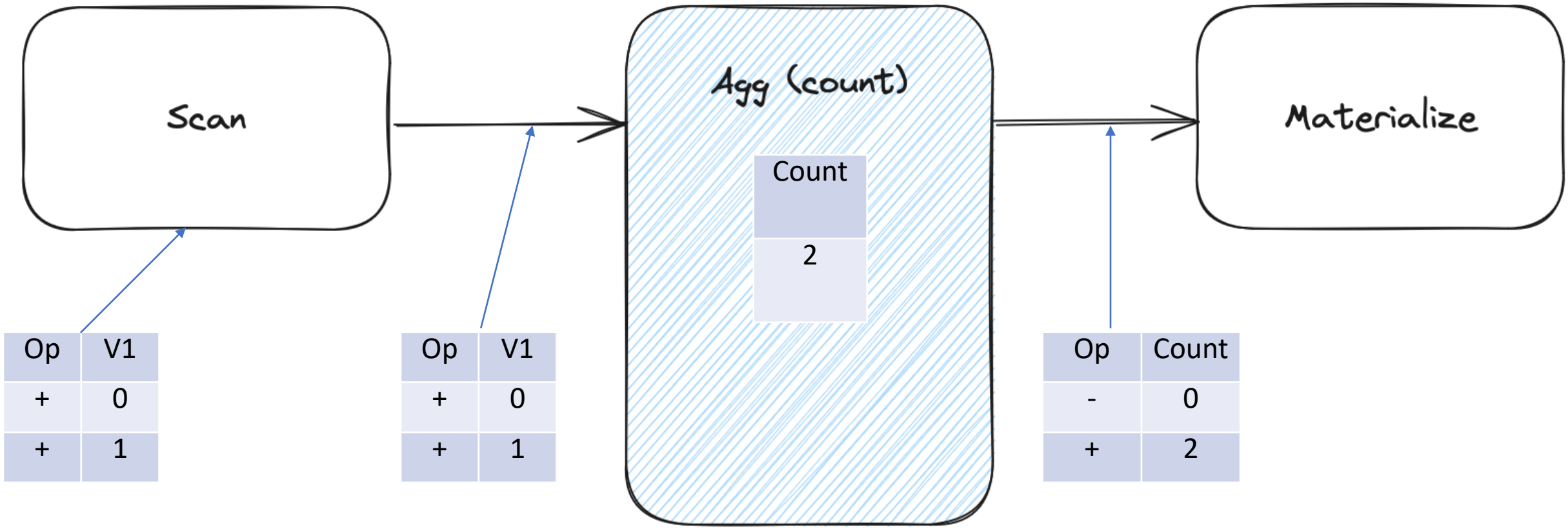
Stateful Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m2 AS SELECT count(*) FROM t;
```



Stateful Computation

```
CREATE TABLE t (v1 int);
CREATE MATERIALIZED VIEW m2 AS SELECT count(*) FROM t;
```

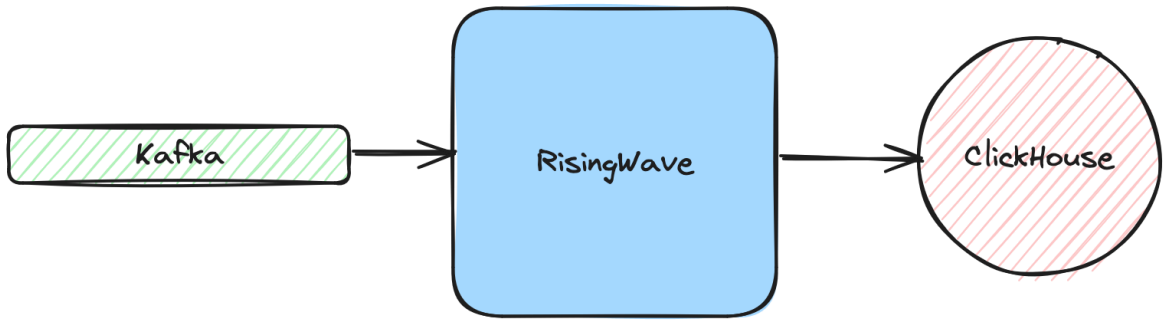


Agenda

1. Why Stream Processing?
2. What is RisingWave?
3. Data Ingestion and Delivery
4. Stateless / Stateful Computation
5. Hands-on Project

5 Hands-on Project

Project



https://github.com/risingwavelabs/risingwave-data-talks-workshop-2024-03-04

Try out RisingWave!

Install RisingWave for macOS and Linux

```
curl https://risingwave.com/sh | sh
```

See the quick start guide for next steps or other ways to run RisingWave.

Thank you for joining the workshop!

My Socials
LinkedIn: https://www.linkedin.com/in/noelkwan/
GitHub: @kwannoel

GitHub

https://github.com/risingwavelabs/risingwave

slack

https://risingwave-labs.com/slack

RisingWave Cloud

https://cloud.risingwave.com/auth/signin