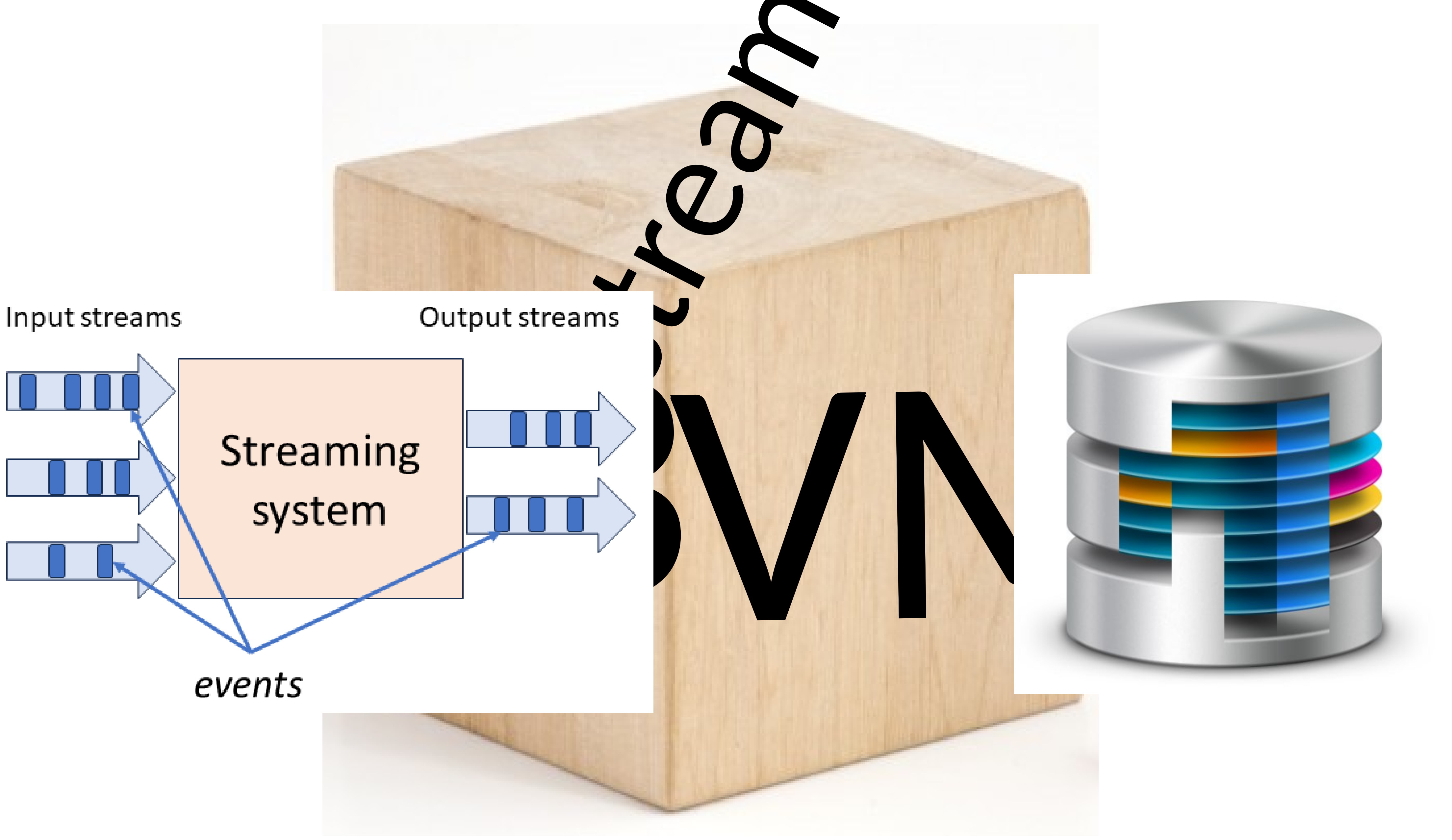
Streaming queries without compromise

Mihai Budiu, Leonid Ryzhyk – Feldera.com

April 24, 2024

Stream Processing Meetup

LinkedIn


Input streams
Output streams
Streaming system
events
tream
VN

Resources

[DBSP: Automatic Incremental View Maintenance for Rich Query Languages]

-
Mihai Budiu, Tej Chajed, Frank McSherry, Leonid Ryzhyk, Val Tannen

[DBSP: Incremental Computation on Streams and Its Applications to Databases]

`github.com/feldera/`

VLDB2023
vancouver

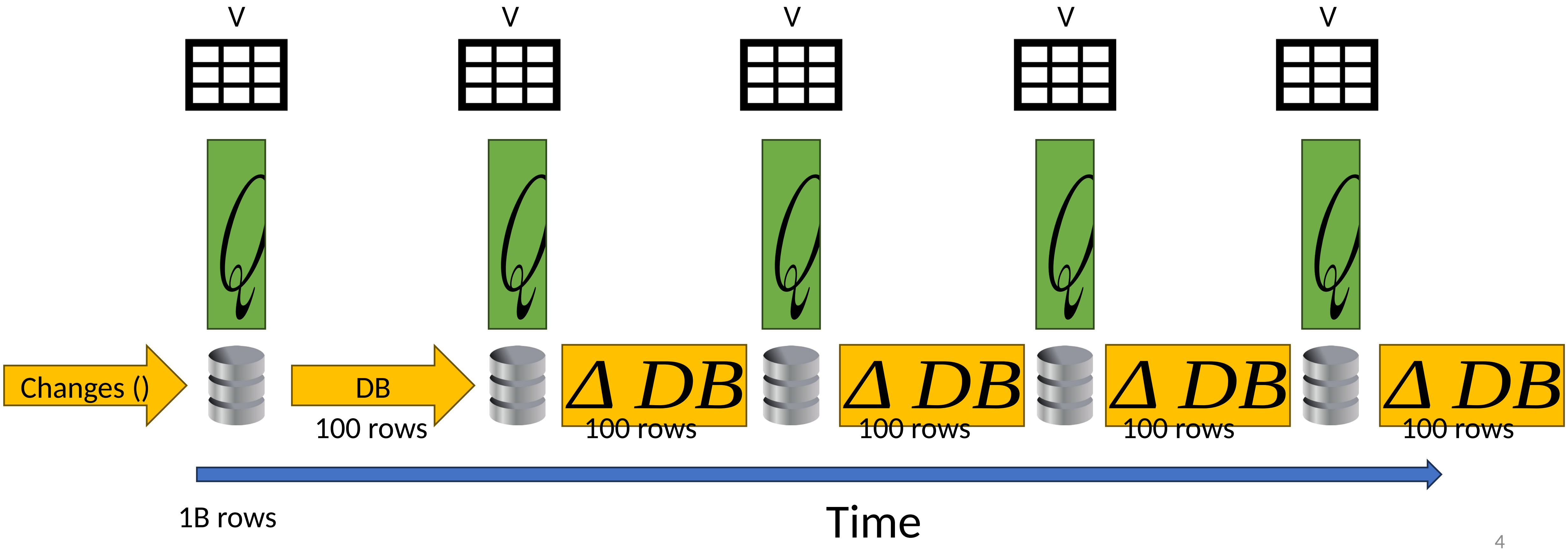
Best paper award

ACM SIGMOD 2024 Research Highlights Award

MIT
License

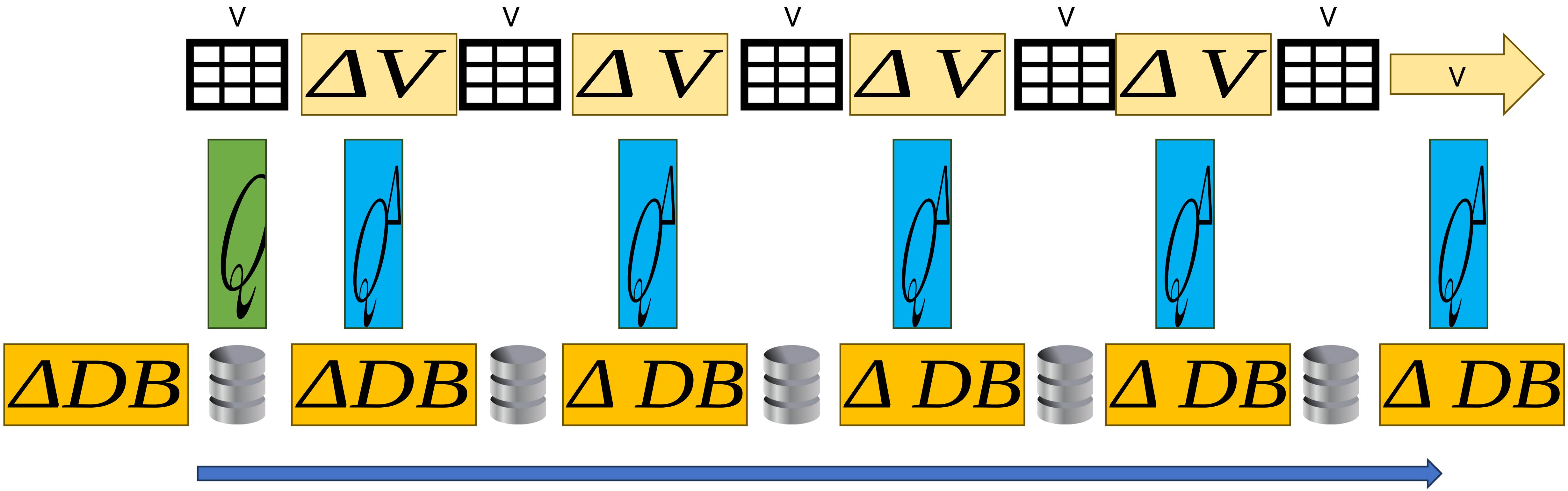
Periodic query evaluation

```
CREATE VIEW V AS SELECT … FROM …
```



Incremental View Maintenance

We want Work()



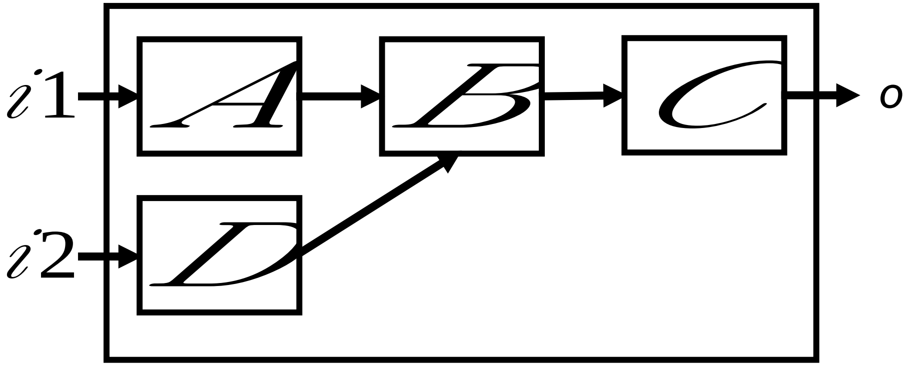
Time

Outline

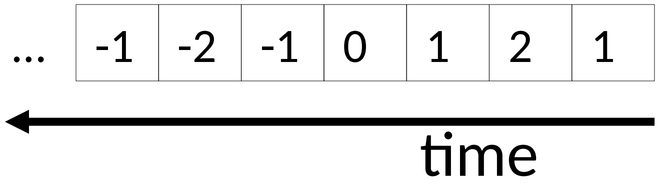
- Incremental View Maintenance
- **Stream computations**
- Databases as streaming systems
- Incremental computation on streams
- SQL in DBSP
- Demo

Dataflow graphs

- Boxes = computations (functions)
- Arrows = values
- (Query plans)

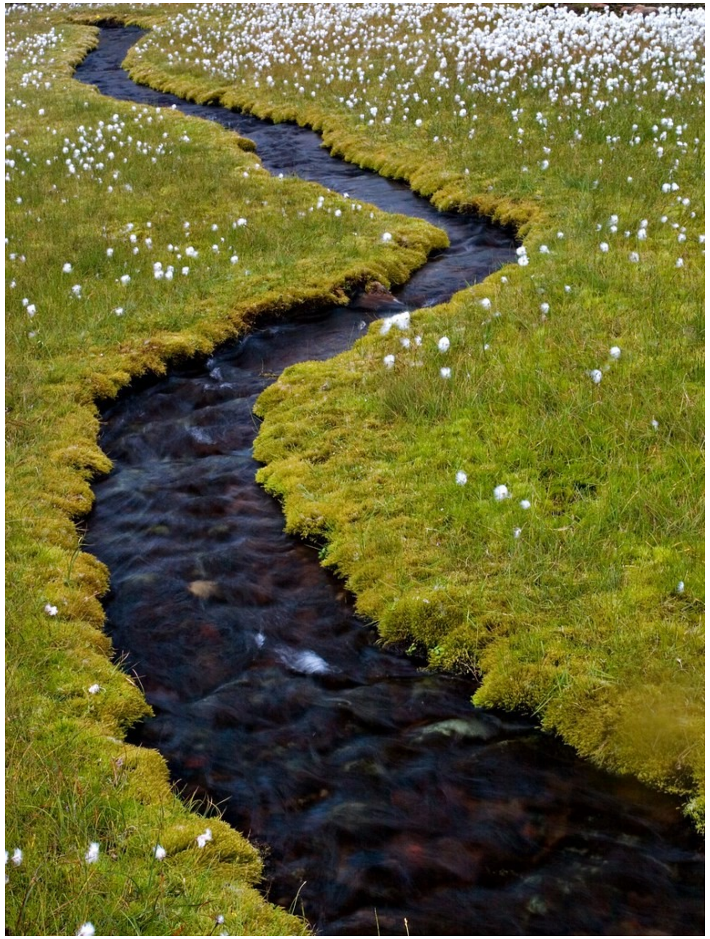


Streams

- Infinite vectors
- 

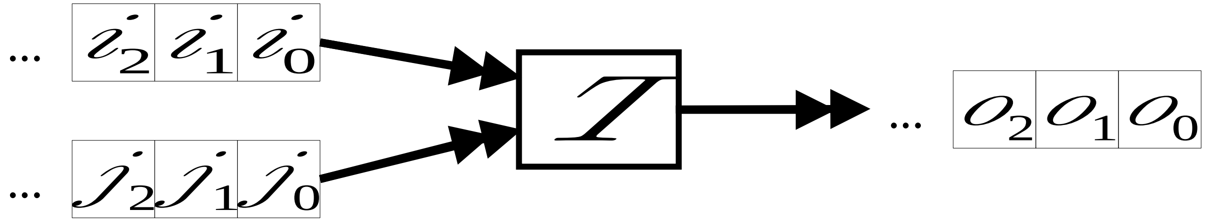
…	-1	-2	-1	0	1	2	1

- $s[0] = 1$
- = streams with elements of type
- We require to have
  - (a commutative group)



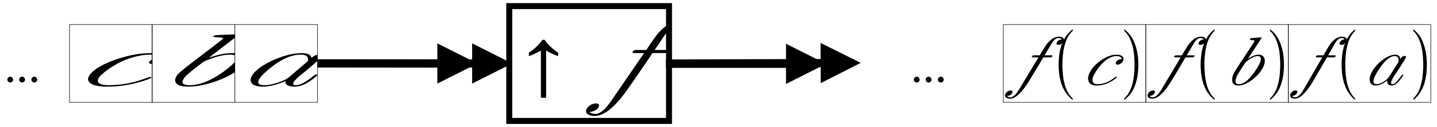
Stream operators

- Arrows with double head = streams
- Boxes = operators



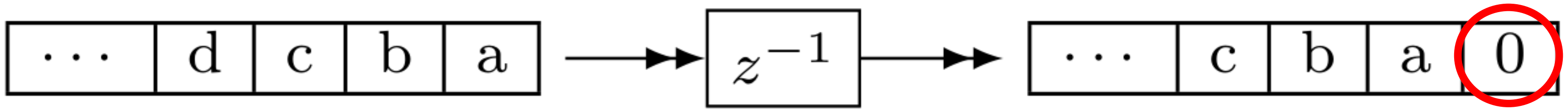
Lifting

Convert a function
into a stream operator



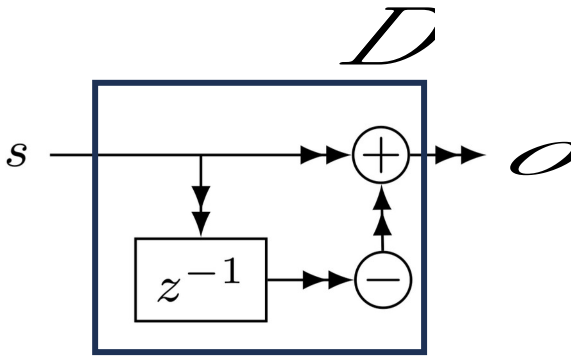
Delay ( )

- Output is input stream delayed by one step
- First value is 0

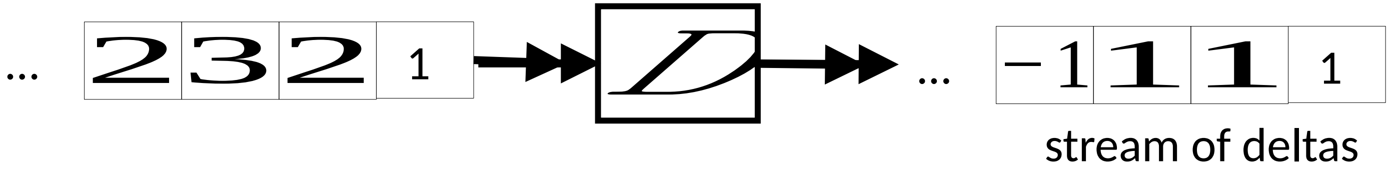


- Stores internal state (the only operator with state)

Computing changes (deltas) Differentiation

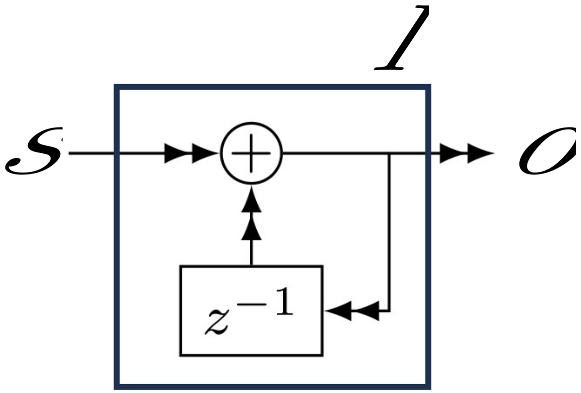


o is the *stream of changes* of *s*

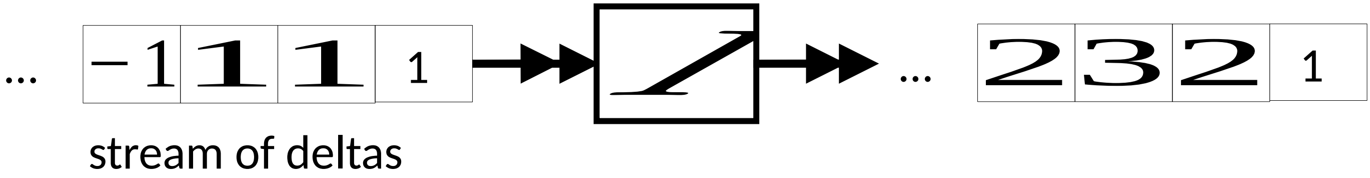


stream of deltas

Integration



- If *s* is a *stream of changes*…
- … then *o* is the original stream



and “cancel out”

$$s \twoheadrightarrow \boxed{\mathcal{I}} \twoheadrightarrow \boxed{\mathcal{D}} \twoheadrightarrow o \quad \cong \quad s \twoheadrightarrow o \quad \cong \quad s \twoheadrightarrow \boxed{\mathcal{D}} \twoheadrightarrow \boxed{\mathcal{I}} \twoheadrightarrow o$$

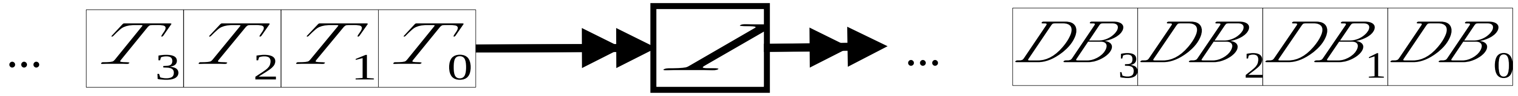
Outline

- Incremental View Maintenance
- Stream computations
- **Databases as streaming systems**
- Incremental computation on streams
- SQL in DBSP
- Demo

All databases are streaming databases!

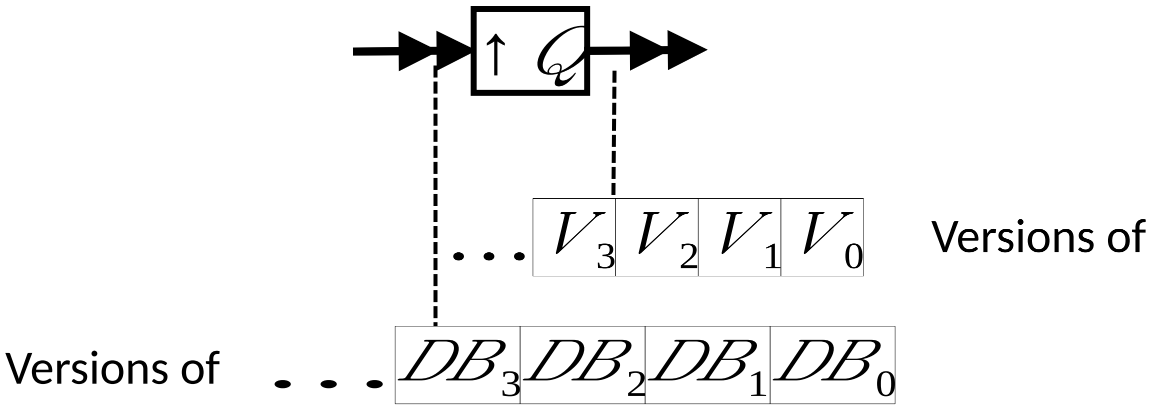
- Consider a database , a set of tables
- A committed transaction is a **change** to
- The set of linearized transactions define a **stream of changes** to
  - is the -th transaction
- is a **stream** of database snapshots
  - is the contents of the database after transactions have been committed
-]

A database (stream) is the integral of a transaction stream



Views are lifted queries

- Let be a query defining a view
- is a stream of view snapshots:



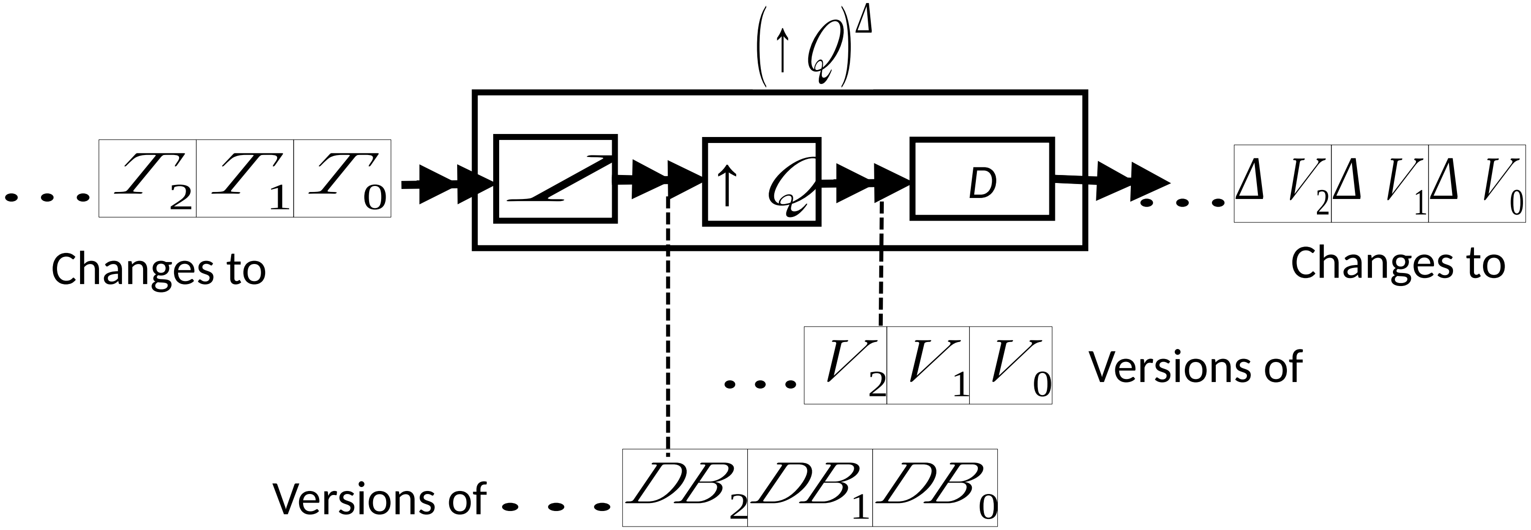
Versions of

Versions of

Outline

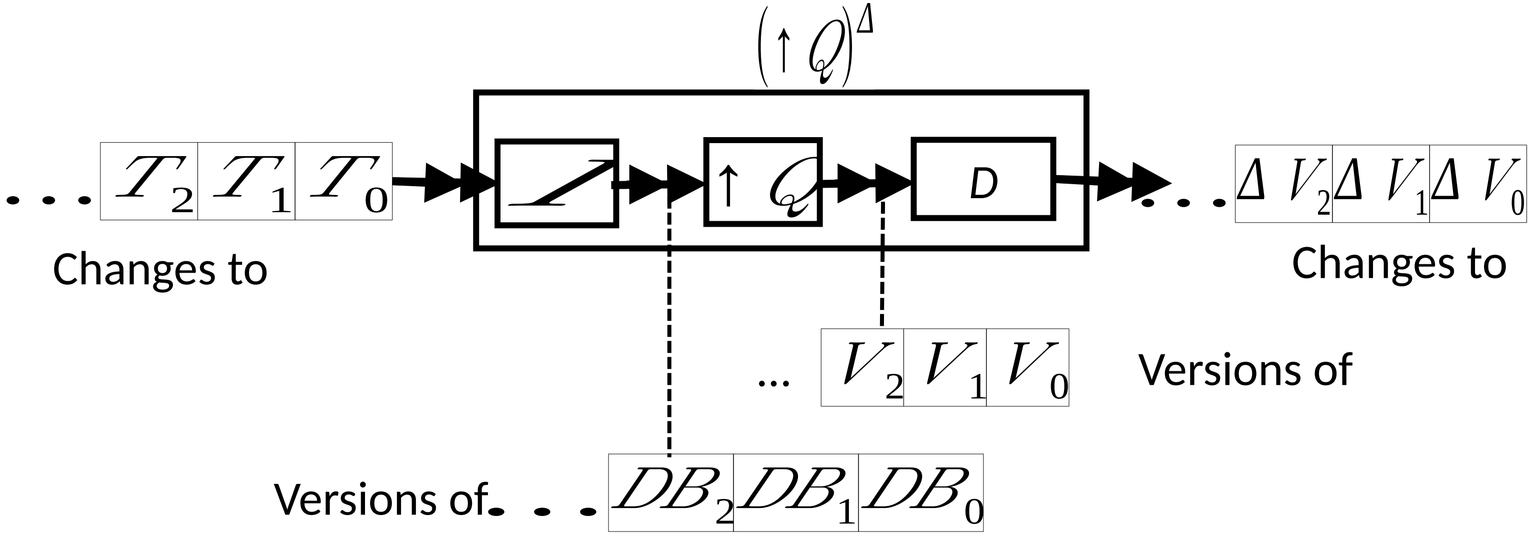
- Incremental View Maintenance
- Stream computations
- Databases as streaming systems
- **Incremental (DB) computations**
- SQL in DBSP
- Demo

Incremental view maintenance of view



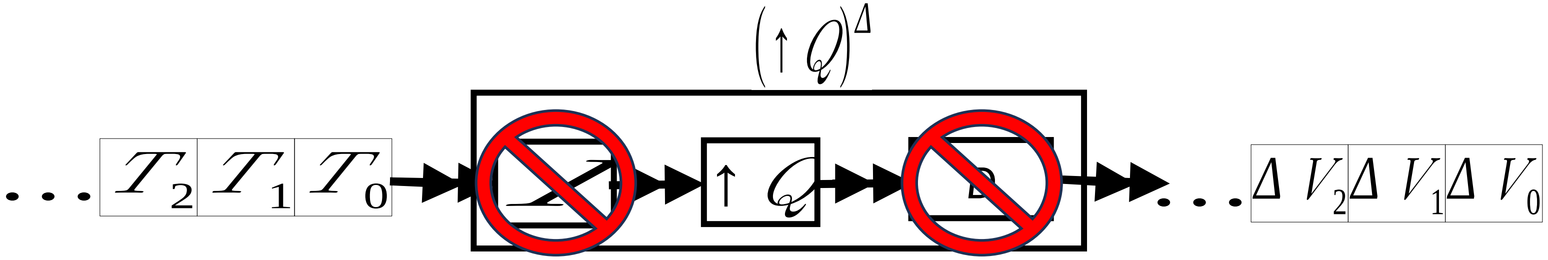
Incremental view maintenance

- This is our definition of IVM
- This definition is much better:
  - It is compositional
  - Inputs and outputs are both **deltas**
- This works, but is **inefficient**



Linear operators

- If is linear:

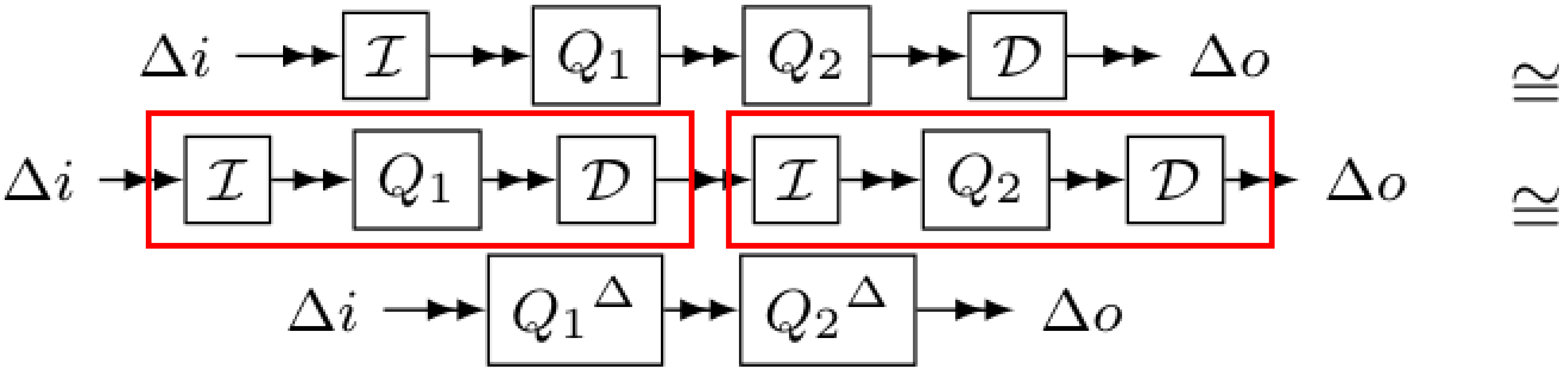


$(\uparrow Q)^{\Delta}$
$\ldots T_2 \; T_1 \; T_0$
$\uparrow Q$
$\ldots \Delta V_2 \; \Delta V_1 \; \Delta V_0$

- Most relational queries use linear operators!

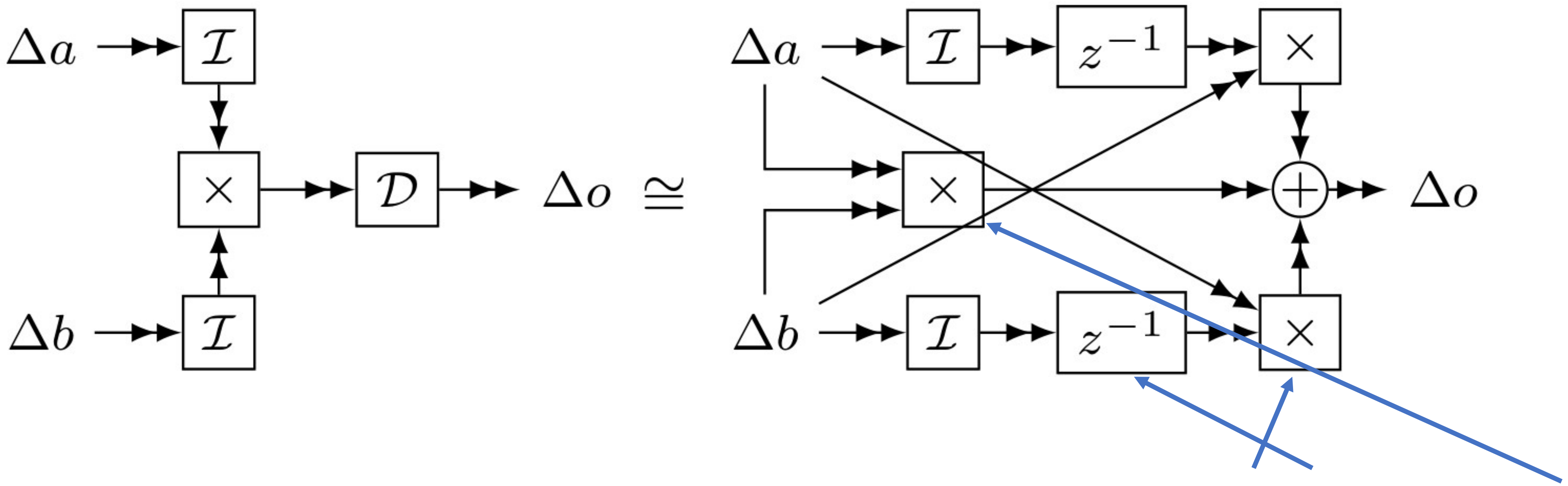
The chain rule

Proof by pictures:



Bilinear operators

- (Lifted) join, intersection, Cartesian product



Outline

- Incremental View Maintenance
- Stream computations
- Databases as streaming systems
- Incremental computation on streams
- **SQL in DBSP**
- Demo

Hold on! This is not well-defined!

- The definitions of , require streams over a commutative group
- However, tables (sets, multisets) are not groups
  - E.g., there is no table negation

-sets

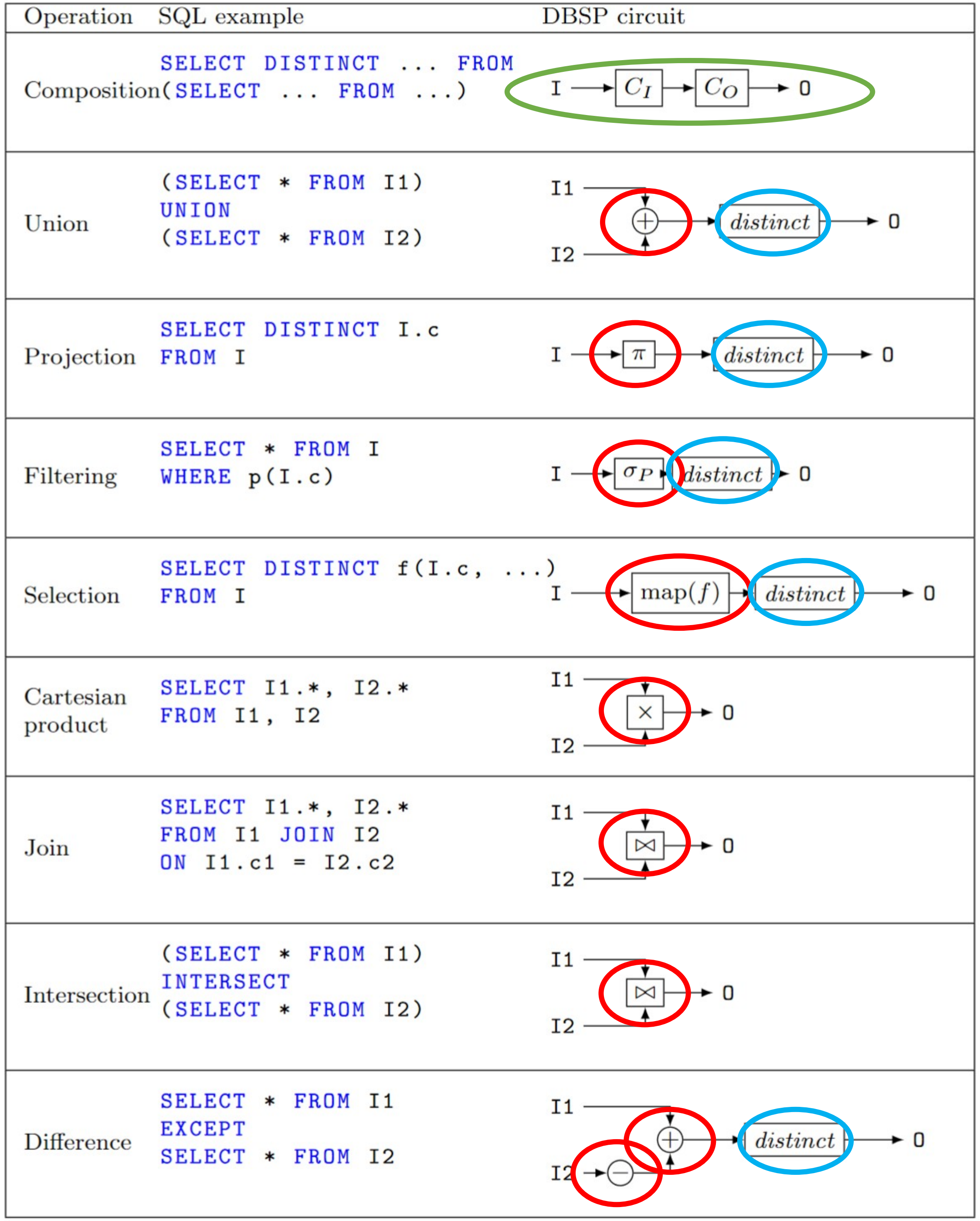
- Each row has an integer weight
- The weight can be positive, zero, or negative

Name	Age	Weight
Mike	10	1
John	12	3
Amy	8	-1
Chris	10	2

tuples

-sets are magic!

- Can represent **both** tables and **changes** to tables
  - Positive weights = elements added
  - Negative weights = elements removed
- Generalize sets and multisets
  - Classic DB table = -set where all weights are 1
- Form a commutative group (because is a group)



Operation	SQL example	DBSP circuit
Composition	SELECT DISTINCT ... FROM (SELECT ... FROM ...)	I → C_I → C_O → O
Union	(SELECT * FROM I1) UNION (SELECT * FROM I2)	I1, I2 → $\oplus$ → *distinct* → O

Projection	SELECT DISTINCT I.c
	FROM I
		I → π → *distinct* → O

Filtering	SELECT * FROM I
	WHERE p(I.c)
		I → σ_P → *distinct* → O

Selection	SELECT DISTINCT f(I.c, ...)
	FROM I
		I → map(f) → *distinct* → O

Cartesian product	SELECT I1.*, I2.*
	FROM I1, I2
		I1, I2 → × → O

Join	SELECT I1.*, I2.*
	FROM I1 JOIN I2
	ON I1.c1 = I2.c2
		I1, I2 → ⋈ → O

Intersection	(SELECT * FROM I1) INTERSECT (SELECT * FROM I2)	I1, I2 → ⋈ → O

Difference	SELECT * FROM I1
	EXCEPT
	SELECT * FROM I2
		I1, I2 → $\ominus$ → $\oplus$ → *distinct* → O

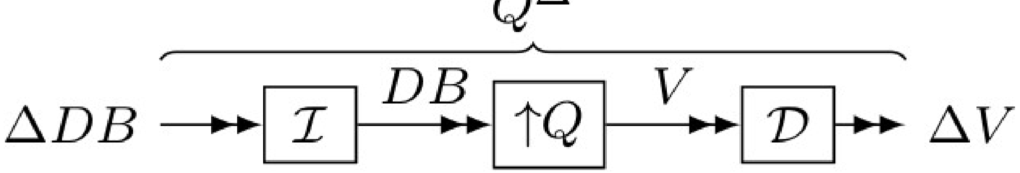
Relational algebra in - sets

- Recursively defined on query structure
- Many operations require a
  - (But some can be removed)
- The other operations are all linear or bilinear

- Can also model
  - Group-by
  - Unnest
  - Aggregation
  - Recursion

Algorithm for incremental view maintenance

1. Translate recursively a DB query into a circuit on -sets
2. Lift circuit to compute on streams:
3. Build incremental version of lifted circuit:

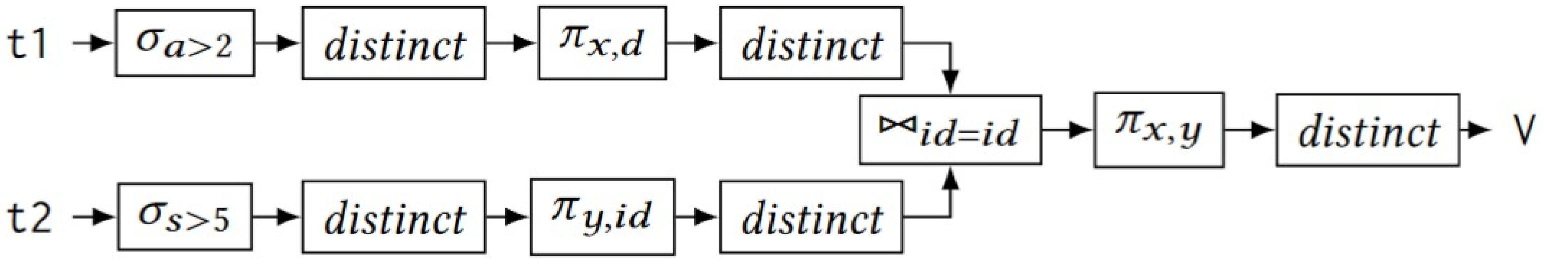


4. Optimize using chain rule:

This algorithm is deterministic. There are no heuristics.

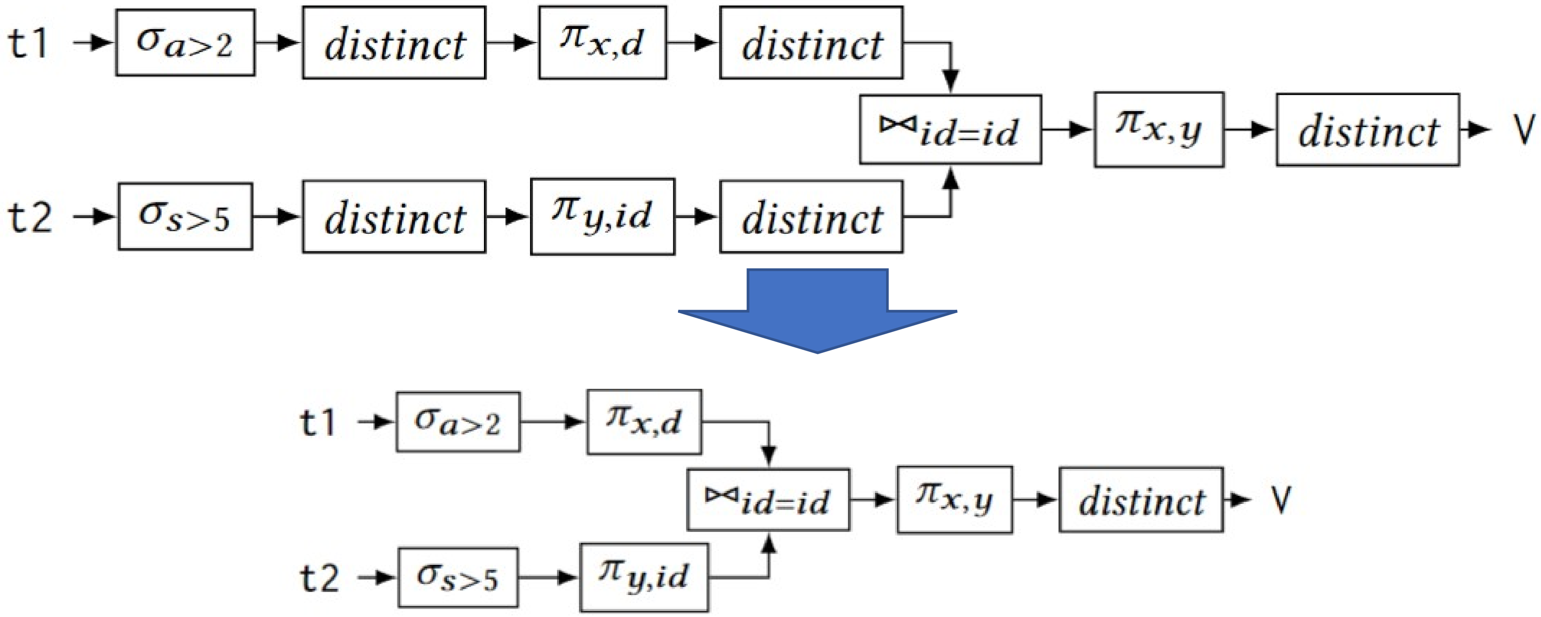
Example: (1) create circuit

```
CREATE VIEW V AS
SELECT DISTINCT a.x, b.y FROM (
  SELECT t1.x , t1.id
  FROM t1
  WHERE t1.a > 2
) a
JOIN (
 SELECT t2.id , t2.y
 FROM t2
 WHERE t2.s > 5
) b ON a.id = b.id
```

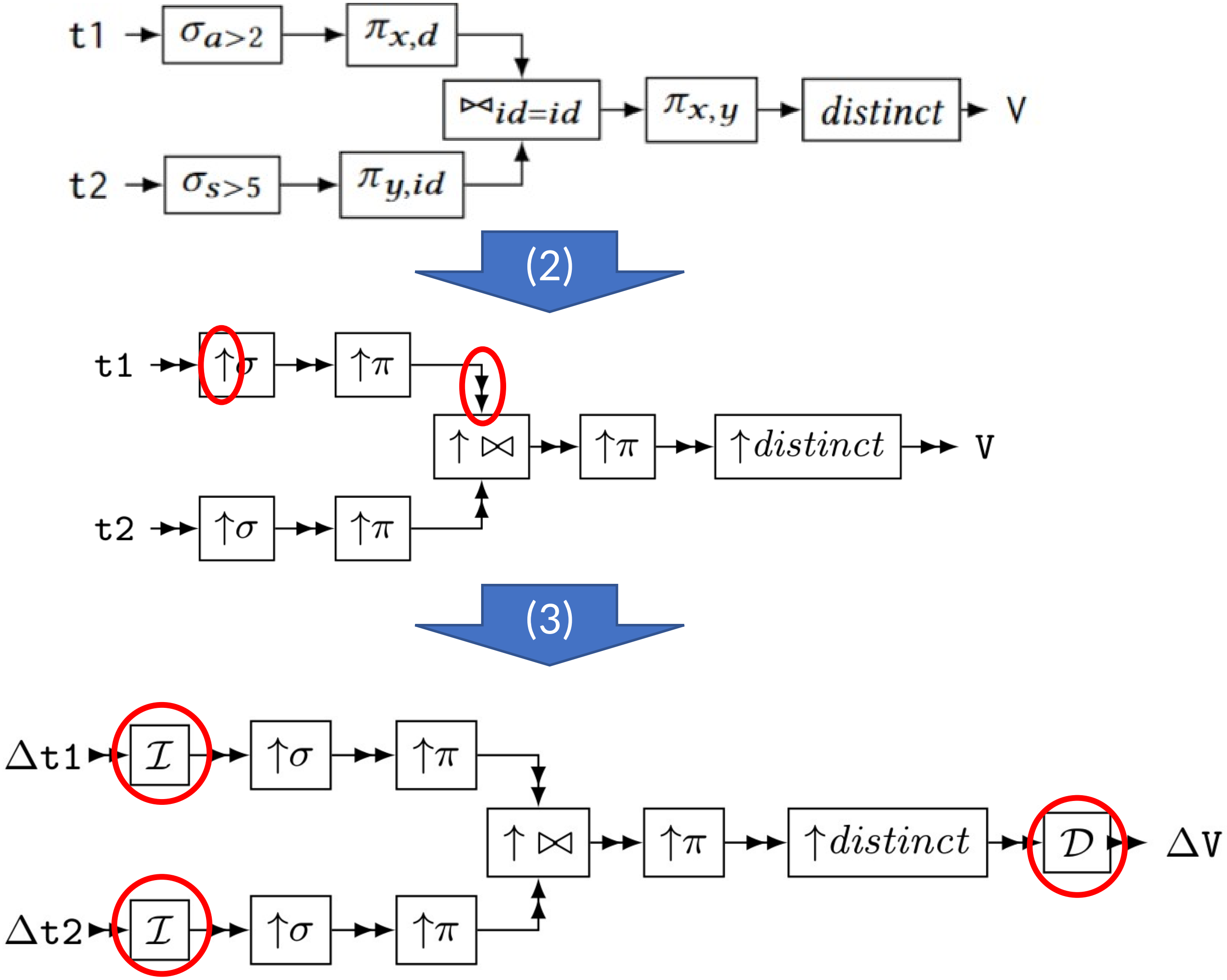


Inputs and outputs are collections, not streams

Remove `distinct` calls



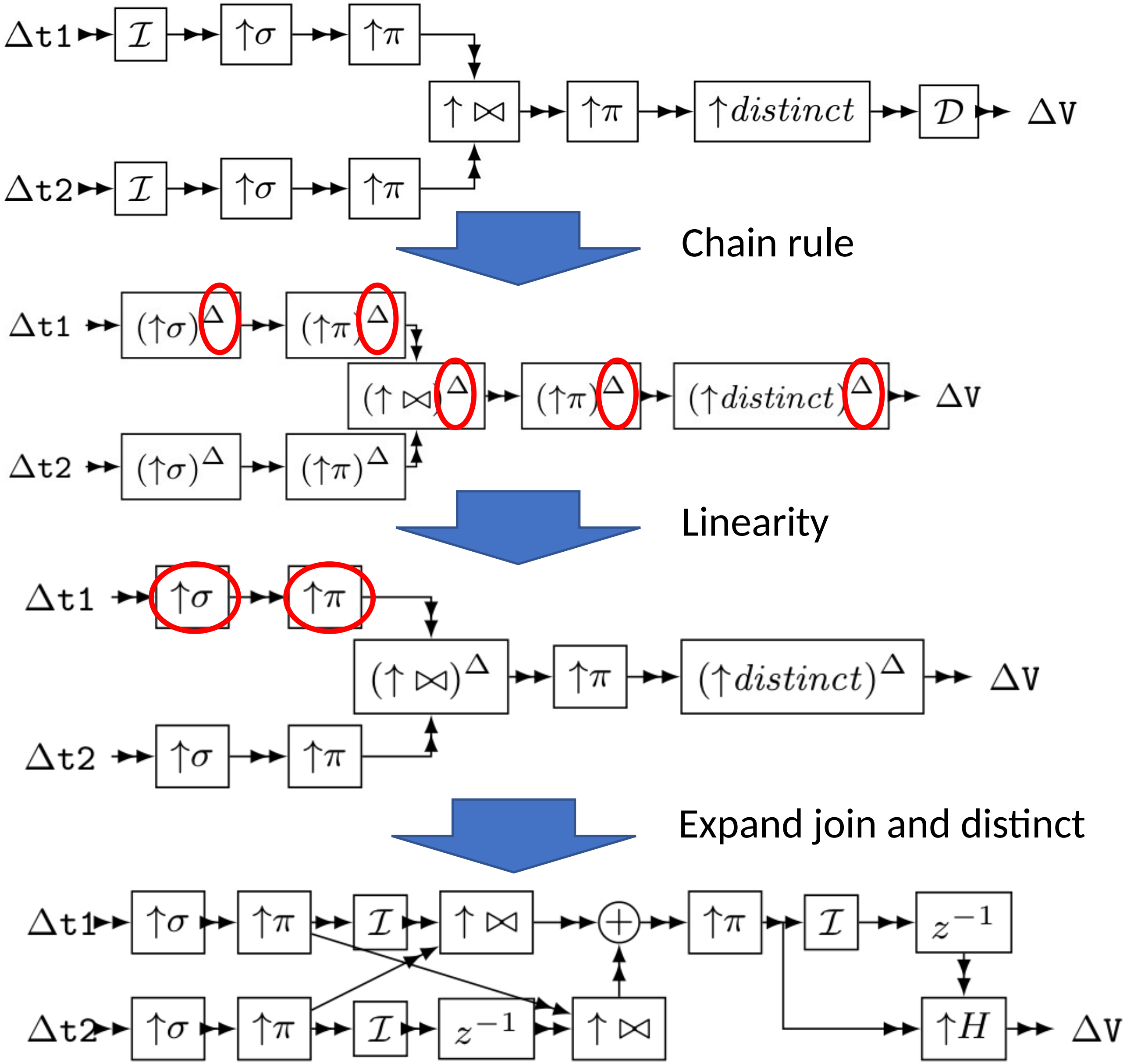
(2) Lift and (3) Incrementalize



Inputs and outputs
are streams

Incrementalize:
Inputs and outputs
are streams of changes

(4) optimize



Chain rule

Linearity

Expand join and distinct

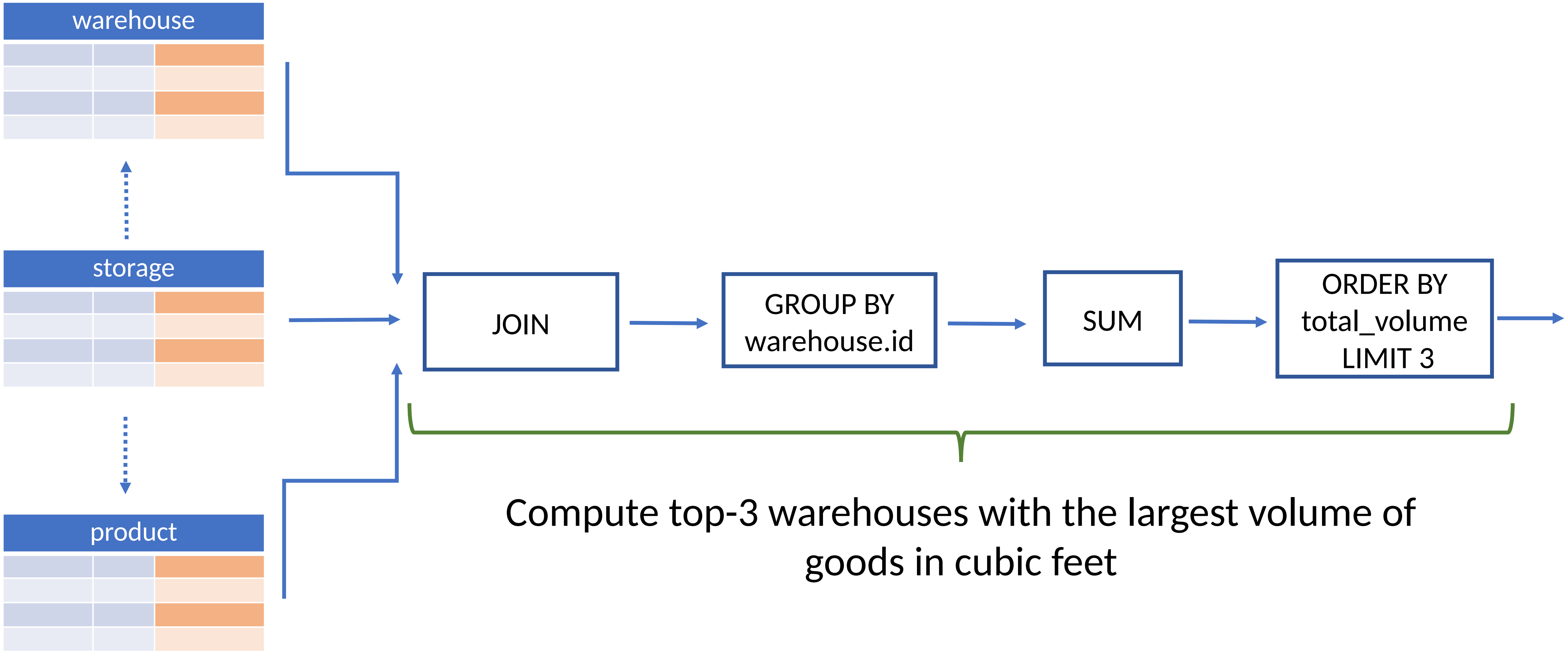
The main tricks

- Streams of **snapshots**
- IVM: from **changes** to changes
- -sets: model negative & positive changes

Outline

- Incremental View Maintenance
- Stream computations
- Databases as streaming systems
- Incremental computation on streams
- SQL in DBSP
- **Demo: Feldera implementation**

Product availability DB



Compute top-3 warehouses with the largest volume of goods in cubic feet

Real-time feature engineering

Credit card transactions

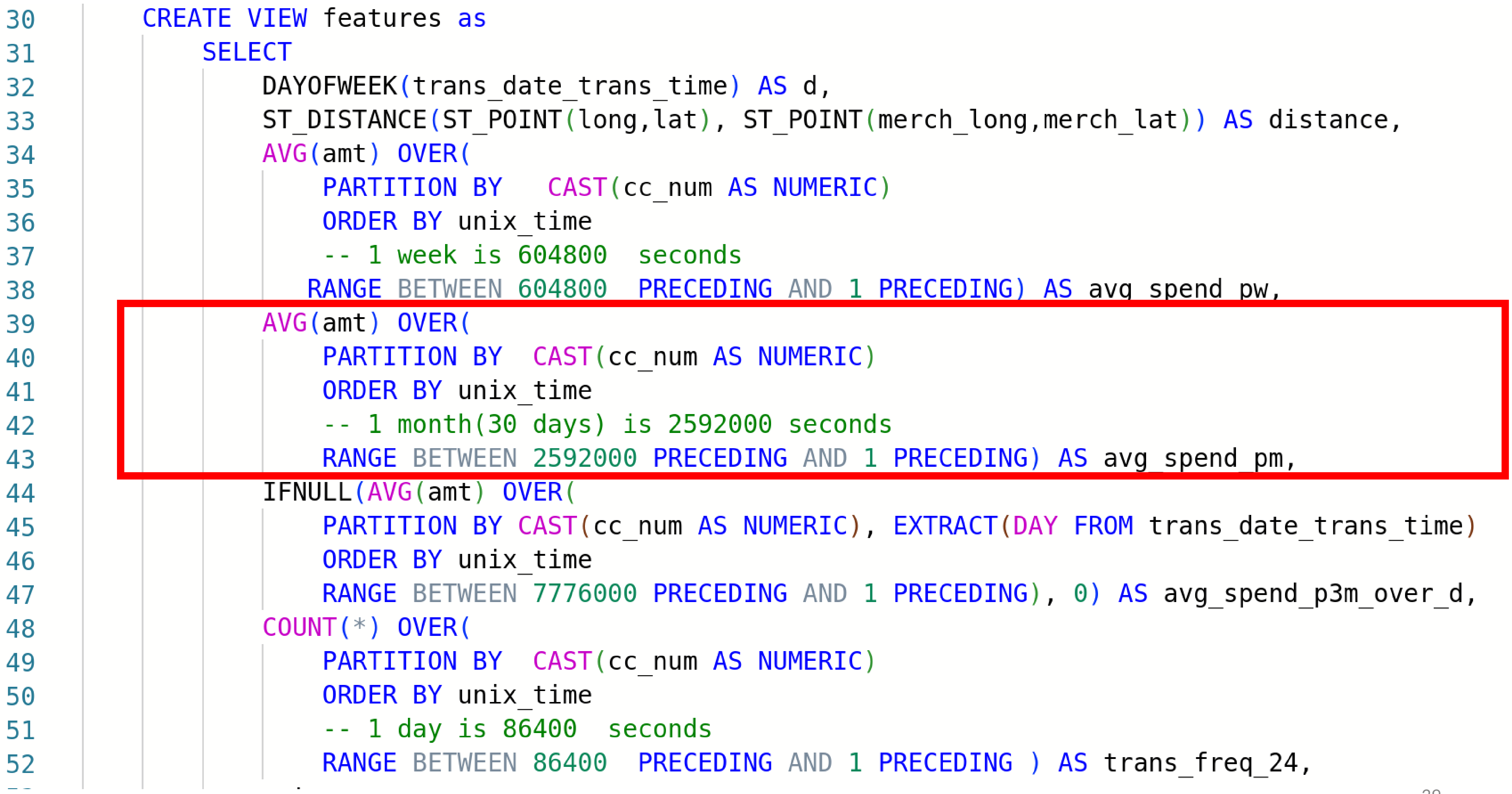
$125	Costco	Mar 1, 1pm
$60	Shell	Mar 3, 8am
$600	Hilton	Mar 8, 5pm
$380	Delta Air	Mar 8, 6pm
$40	Books Inc	Mar11, 1pm
$1840	Costco	Apr 1, 4pm
$15	7-eleven	Apr 3, 6pm
$65	Shell	Apr 7, 8am
$8	Starbucks	Apr 10, 9am
$12	Caltrain	Apr 11, 5pm
…	…	…

MAX/AVG/STDDEV over 30 days

Real-time feature engineering

			AVG	MAX
$125	Costco	Mar 1, 1pm	…	…
$60	Shell	Mar 3, 8am	…	…
$600	Hilton	Mar 8, 5pm	…	…
$380	Delta Air	Mar 8, 6pm	…	…
$40	Books Inc	Mar11, 1pm	…	…
$1840	Costco	Apr 1, 4pm	…	…
$15	7-eleven	Apr 3, 6pm	…	…
$65	Shell	Apr 7, 8am	…	…
$8	Starbucks	Apr 10, 9am	…	…
$12	Caltrain	Apr 11, 5pm	…	…
…	…	…	…	…

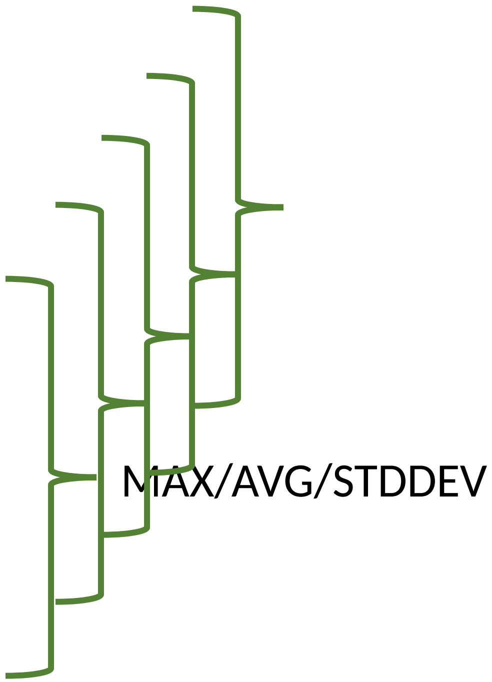
Model training and inference



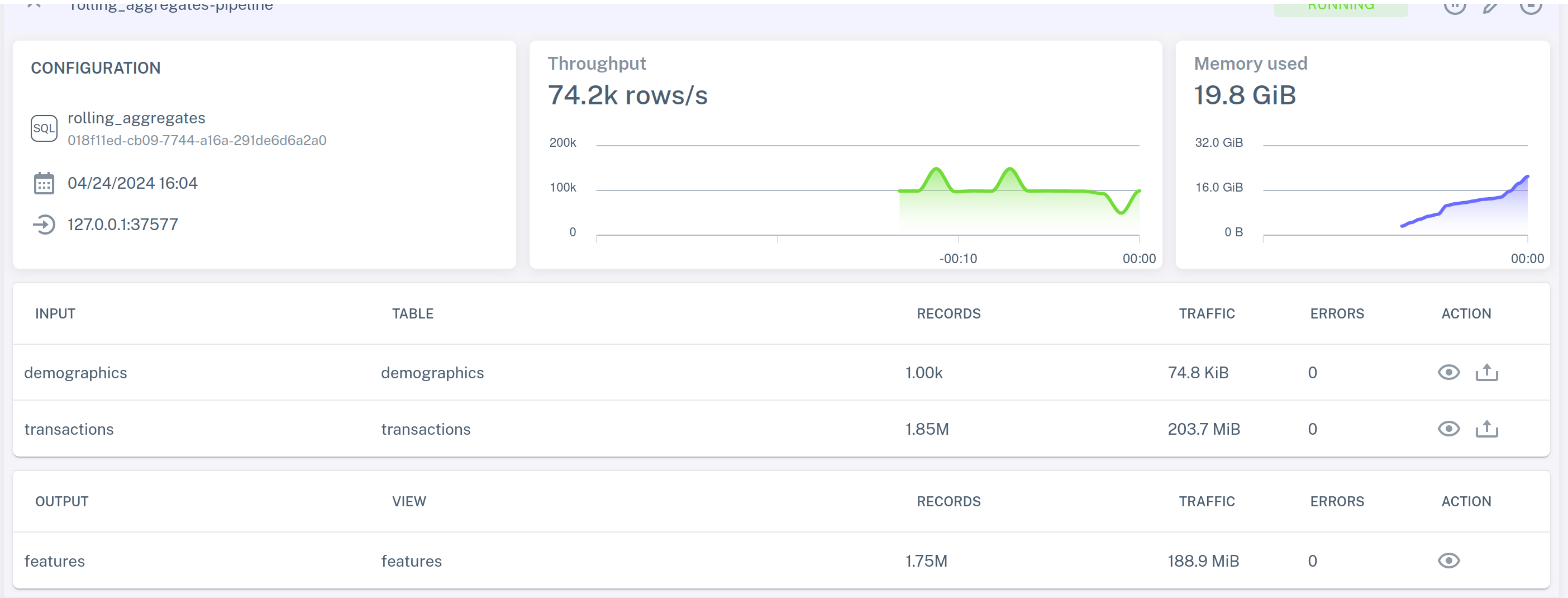
```
CREATE VIEW features as
    SELECT
        DAYOFWEEK(trans_date_trans_time) AS d,
        ST_DISTANCE(ST_POINT(long,lat), ST_POINT(merch_long,merch_lat)) AS distance,
        AVG(amt) OVER(
            PARTITION BY   CAST(cc_num AS NUMERIC)
            ORDER BY unix_time
            -- 1 week is 604800  seconds
           RANGE BETWEEN 604800  PRECEDING AND 1 PRECEDING) AS avg_spend_pw,
        AVG(amt) OVER(
            PARTITION BY  CAST(cc_num AS NUMERIC)
            ORDER BY unix_time
            -- 1 month(30 days) is 2592000 seconds
            RANGE BETWEEN 2592000 PRECEDING AND 1 PRECEDING) AS avg_spend_pm,
        IFNULL(AVG(amt) OVER(
            PARTITION BY CAST(cc_num AS NUMERIC), EXTRACT(DAY FROM trans_date_trans_time)
            ORDER BY unix_time
            RANGE BETWEEN 7776000 PRECEDING AND 1 PRECEDING), 0) AS avg_spend_p3m_over_d,
        COUNT(*) OVER(
            PARTITION BY  CAST(cc_num AS NUMERIC)
            ORDER BY unix_time
            -- 1 day is 86400  seconds
            RANGE BETWEEN 86400  PRECEDING AND 1 PRECEDING ) AS trans_freq_24,
```

Real-time feature engineering

$125	Costco	Mar 1, 1pm
$60	Shell	Mar 3, 8am
$600	Hilton	Mar 8, 5pm
$380	Delta Air	Mar 8, 6pm
$40	Books Inc	Mar11, 1pm
$1840	Costco	Apr 1, 4pm
$15	7-eleven	Apr 3, 6pm
$65	Shell	Apr 7, 8am
$8	Starbucks	Apr 10, 9am
$12	Caltrain	Apr 11, 5pm
…	…	…



MAX/AVG/STDDEV



rolling_aggregates-pipeline

RUNNING

CONFIGURATION

rolling_aggregates
018f11ed-cb09-7744-a16a-291de6d6a2a0

04/24/2024 16:04

127.0.0.1:37577

Throughput
74.2k rows/s

Memory used
19.8 GiB

INPUT	TABLE	RECORDS	TRAFFIC	ERRORS	ACTION
demographics	demographics	1.00k	74.8 KiB	0	👁 ⇪
transactions	transactions	1.85M	203.7 MiB	0	👁 ⇪

OUTPUT	VIEW	RECORDS	TRAFFIC	ERRORS	ACTION
features	features	1.75M	188.9 MiB	0	👁

try.feldera.com

github.com/feldera

www.feldera.com/community