MZ + PG: Source internals

Materialize

Hello

I’m Sean Loiselle.

Live in Washington Heights.

Started at “The View” in 2019.

Wrote a bunch of the SQL layer in the early days, now on the storage team.

1 What is MZ?

2 Inside MZ

3 PG+MZ

4 The Correctness Bug™

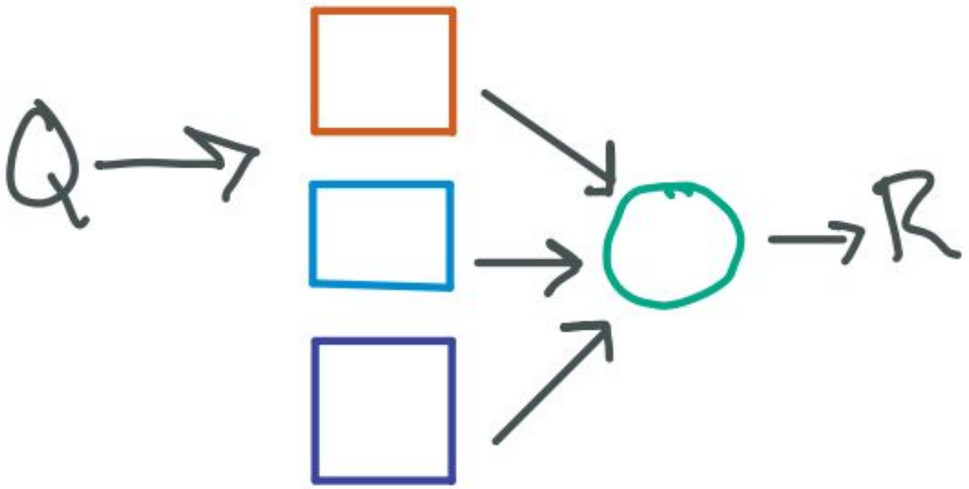
time allotting

5 Q&A

What is MZ?

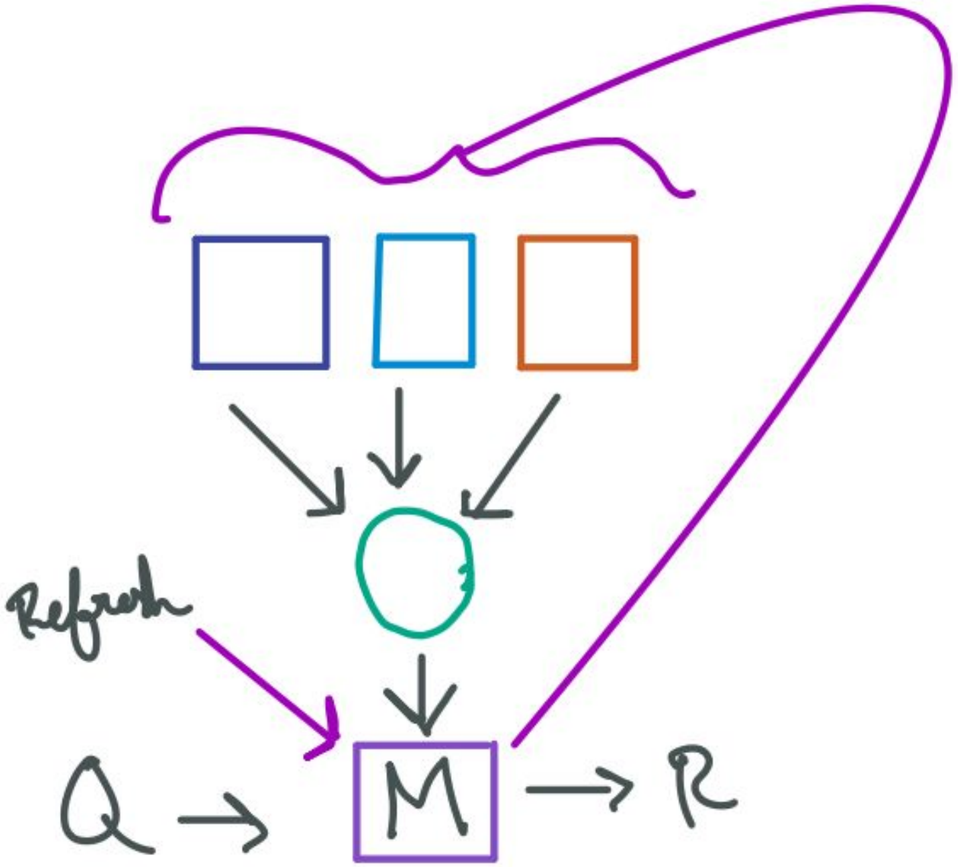
What are materialized views?

```
SELECT
    p.product_category,
    s.store_location,
    SUM(sales.amount) AS total_sales,
    AVG(sales.unit_price) AS average_price,
    COUNT(DISTINCT sales.customer_id) AS unique_customers
FROM
    sales
INNER JOIN products p ON sales.product_id = p.product_id
INNER JOIN stores s ON sales.store_id = s.store_id
WHERE
    sales.sale_date BETWEEN '2023-11-21' AND '2023-11-22'
GROUP BY
    p.product_category,
    s.store_location
ORDER BY
    total_sales DESC;
```



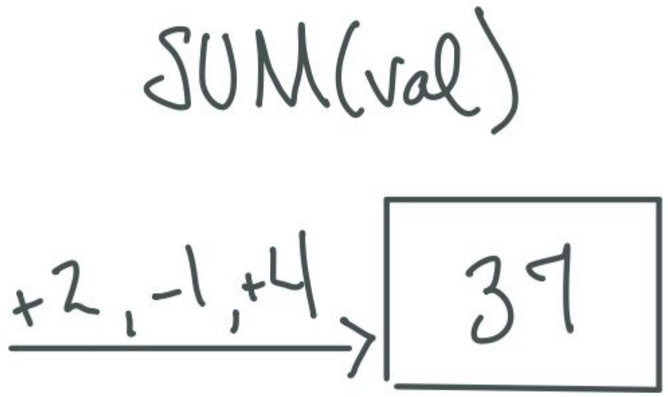
What are materialized views?

```
CREATE MATERIALIZED VIEW expensive_query AS SELECT
    p.product_category,
    s.store_location,
    SUM(sales.amount) AS total_sales,
    AVG(sales.unit_price) AS average_price,
    COUNT(DISTINCT sales.customer_id) AS unique_customers
FROM
    sales
INNER JOIN products p ON sales.product_id = p.product_id
INNER JOIN stores s ON sales.store_id = s.store_id
WHERE
    sales.sale_date BETWEEN '2023-11-21' AND '2023-11-22'
GROUP BY
    p.product_category,
    s.store_location
ORDER BY
    total_sales DESC;
```



What if they weren't stale?

What if they weren't stale?

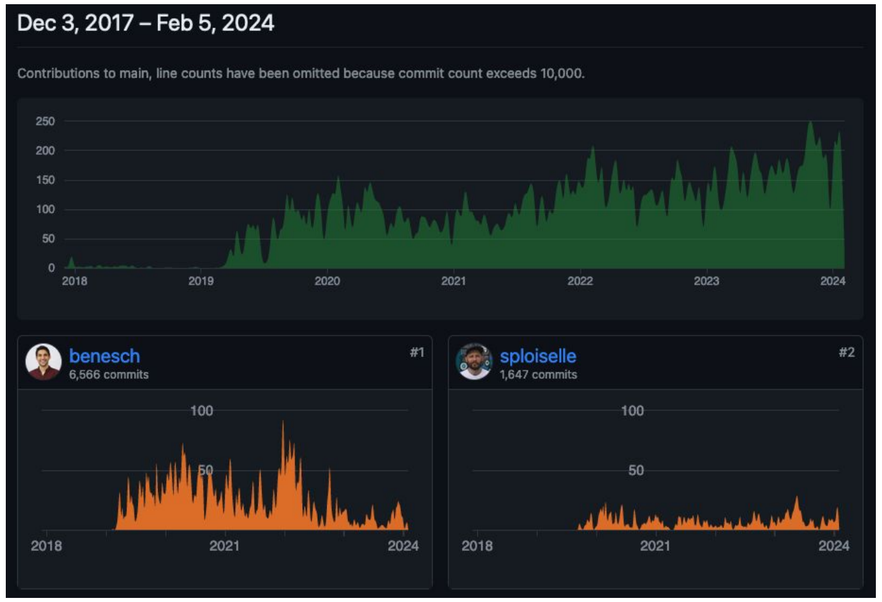


Materialize is born

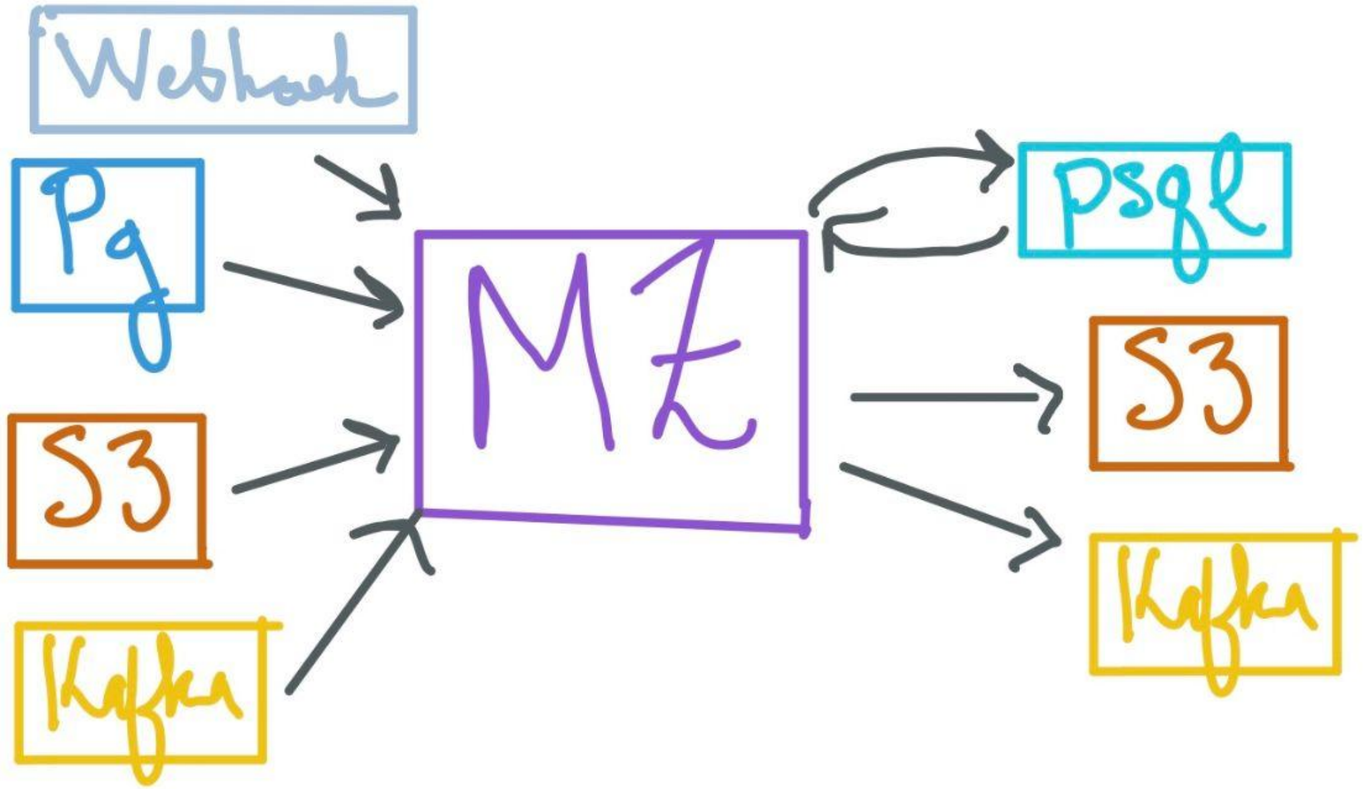
```
git ls-files | xargs wc -l
```

```
MZ: 335727
```

```
PostgreSQL: 917966
```

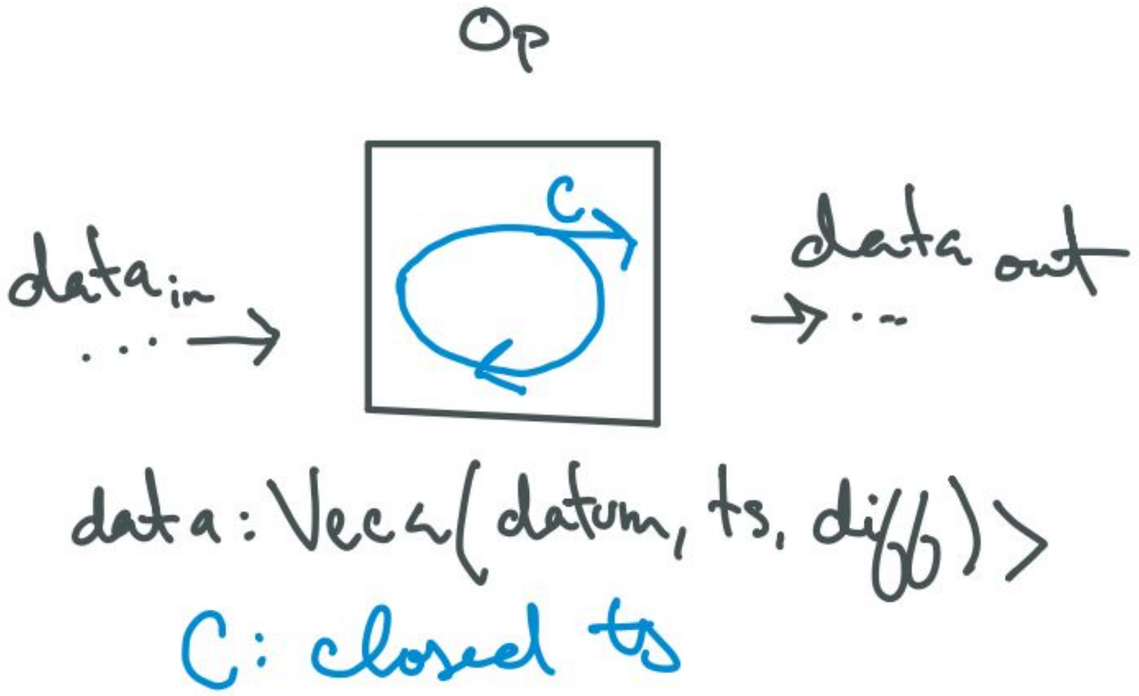


More than just views...

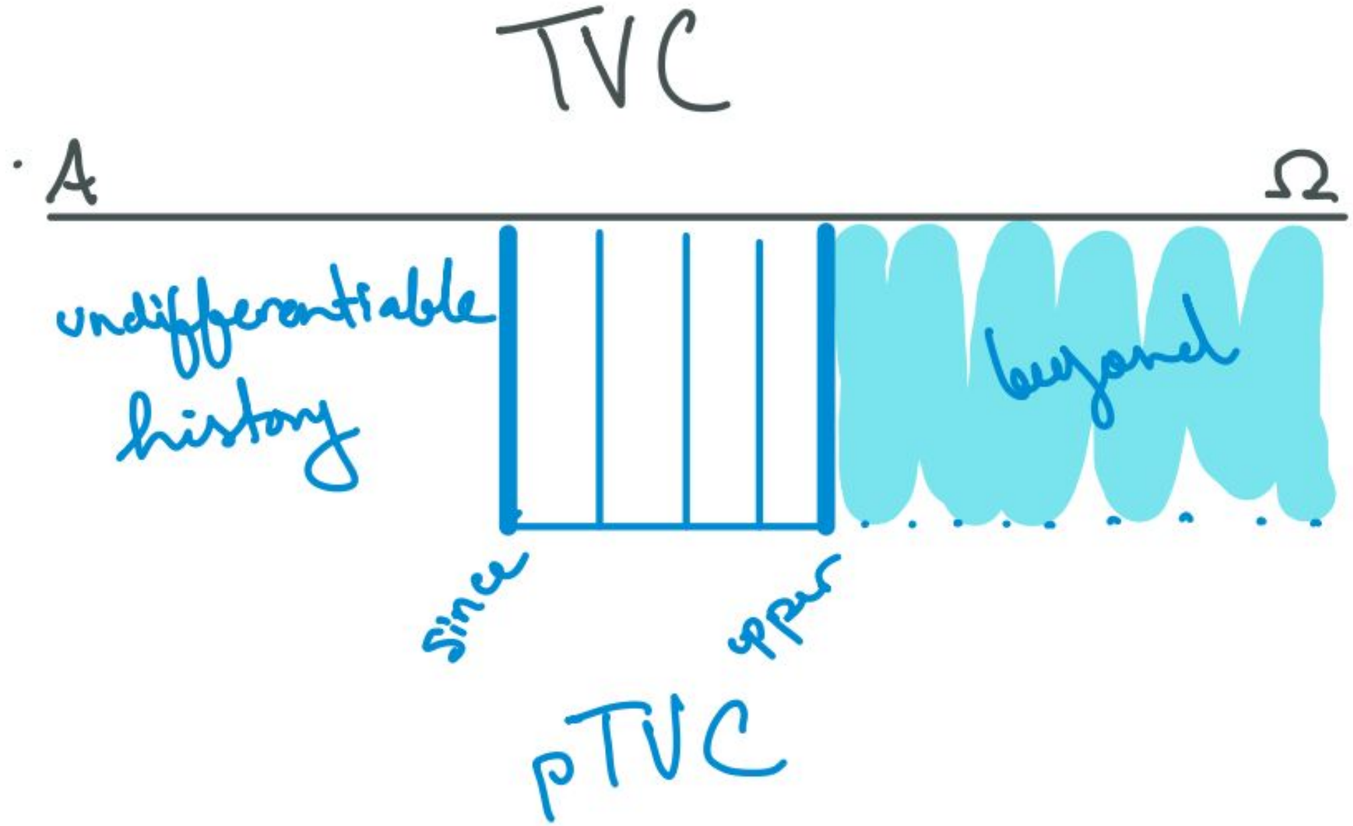


Inside MZ

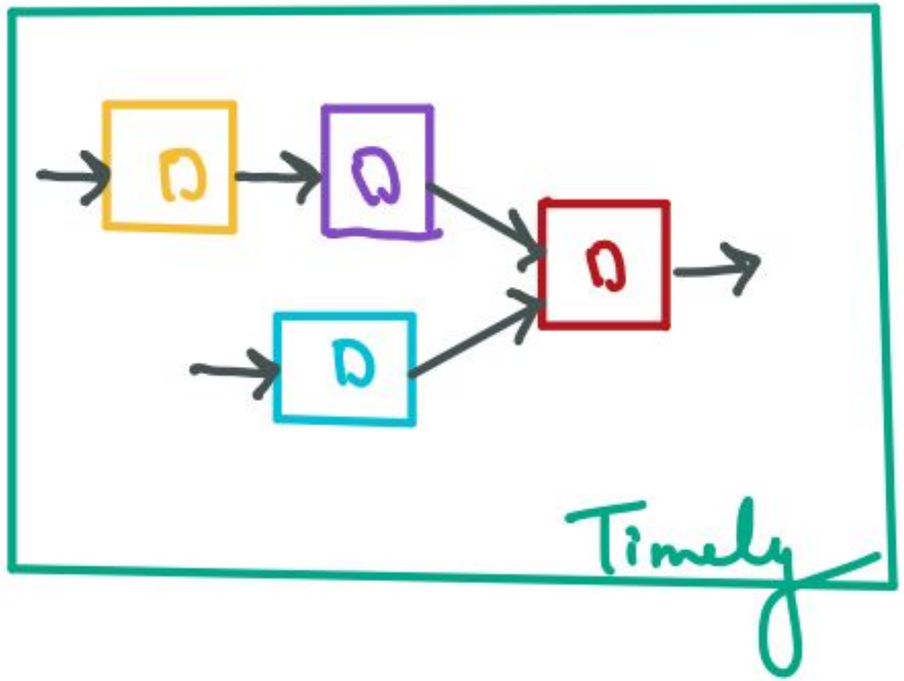
Timely and Differential



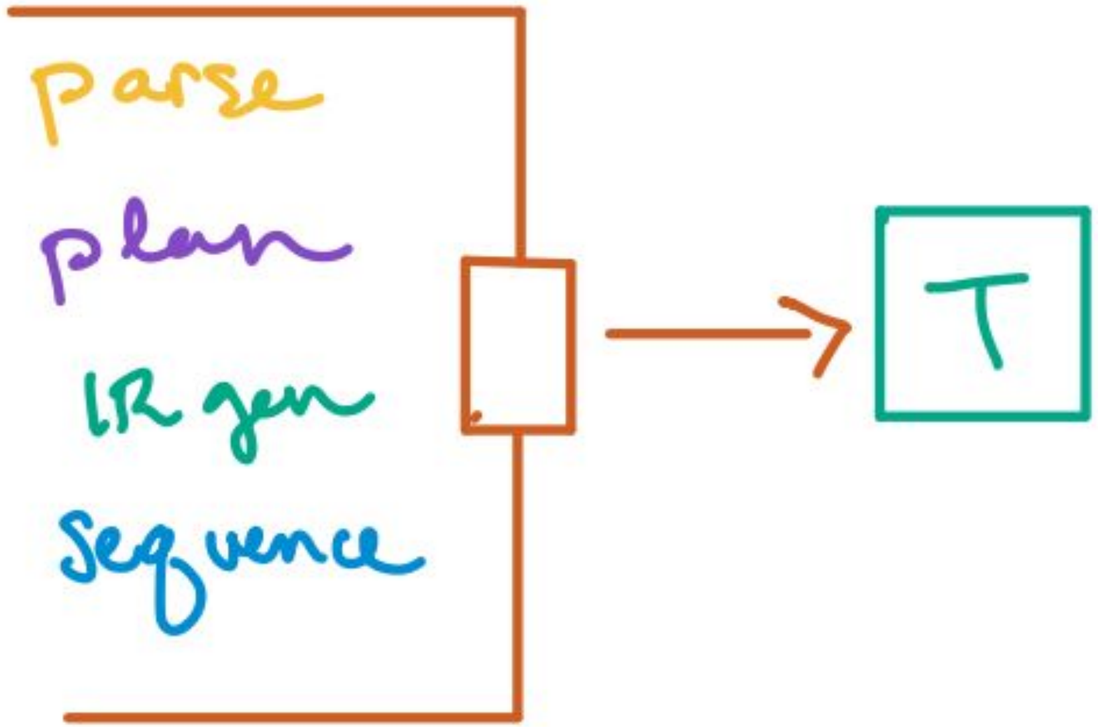
A little more about time



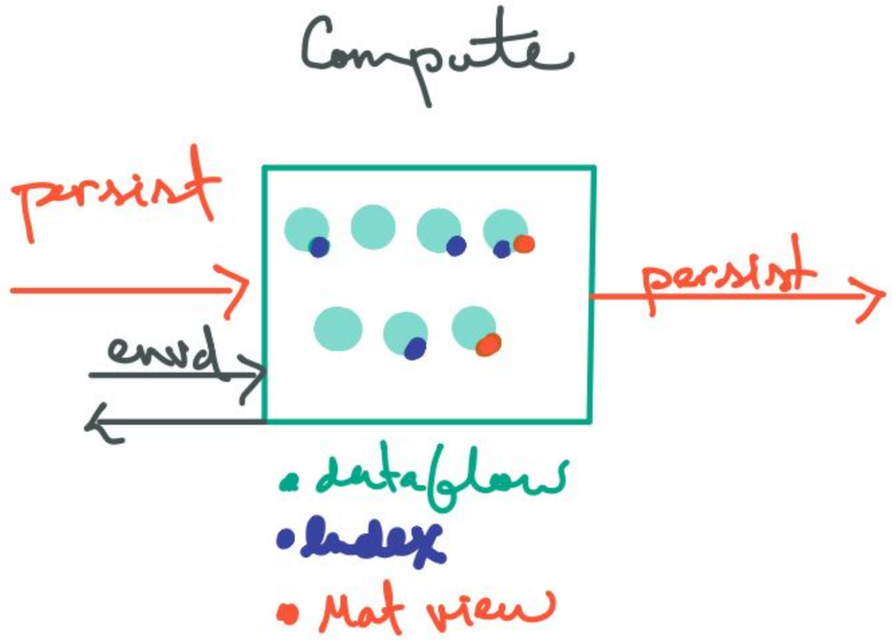
Timely and Differential



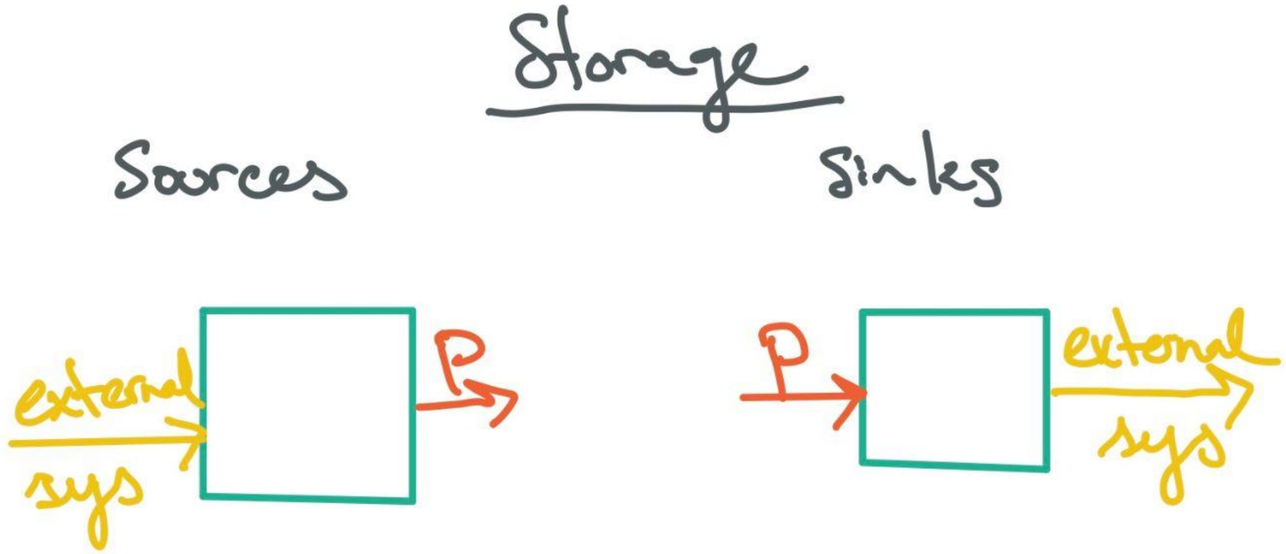
Executing dataflows



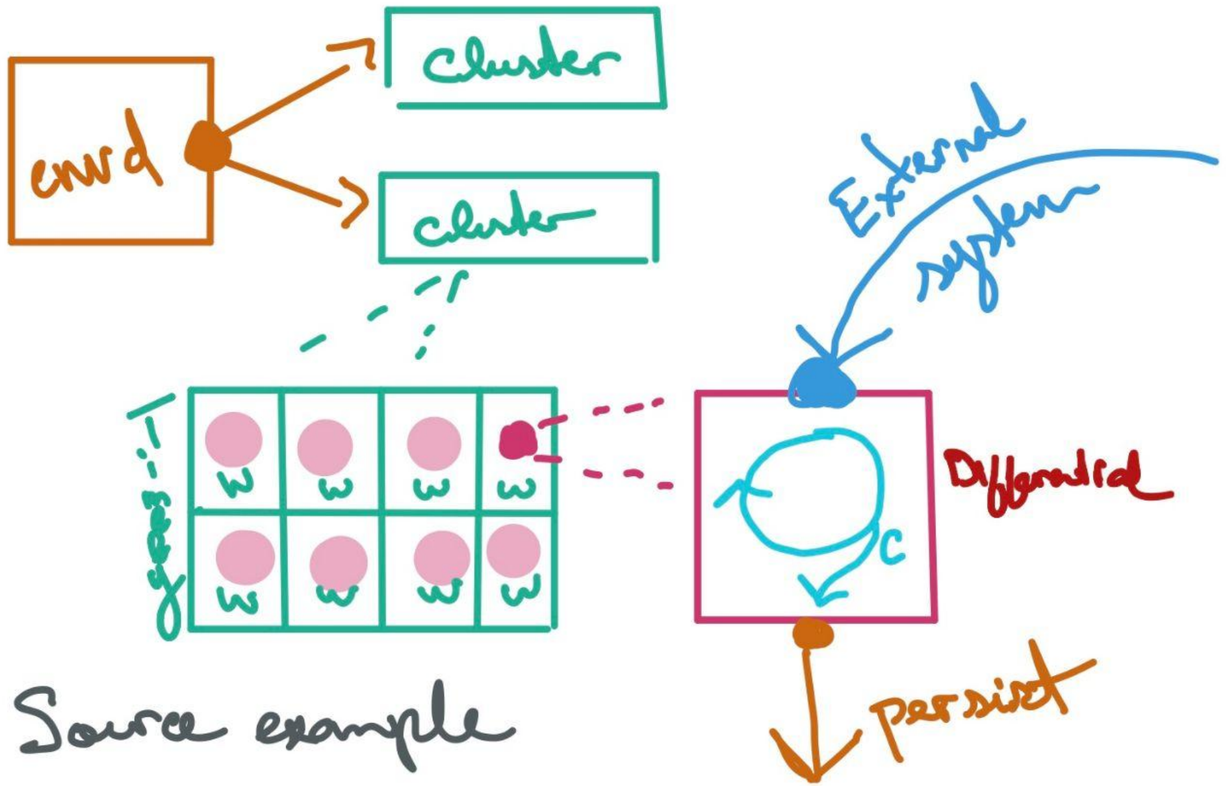
Executing dataflows



Executing dataflows

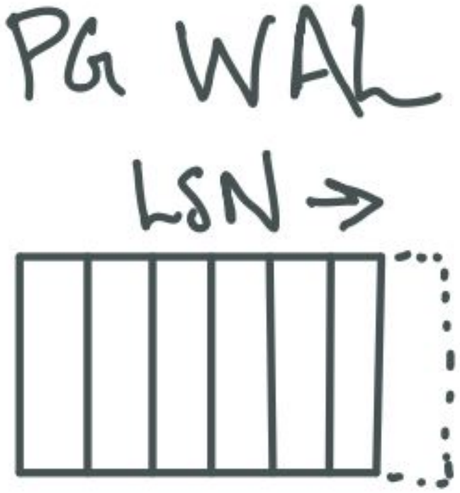


Diagram

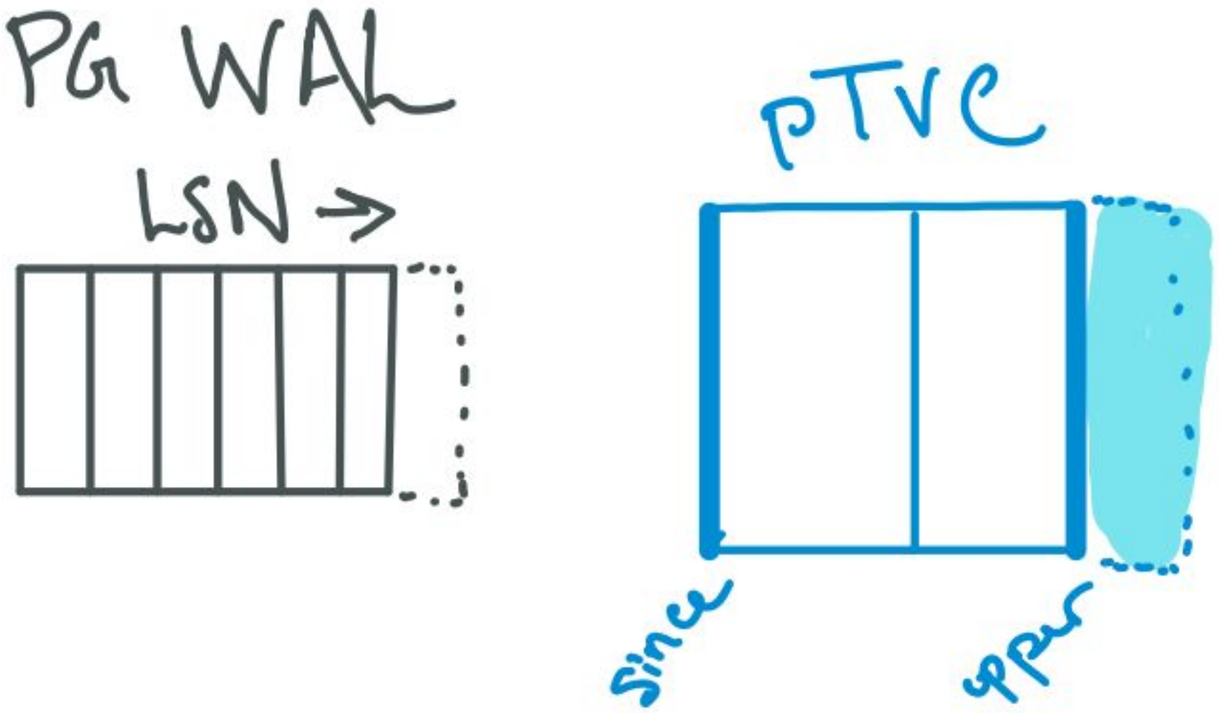

envd
cluster
cluster
External system
Differential
persist
Source example

PG + MZ

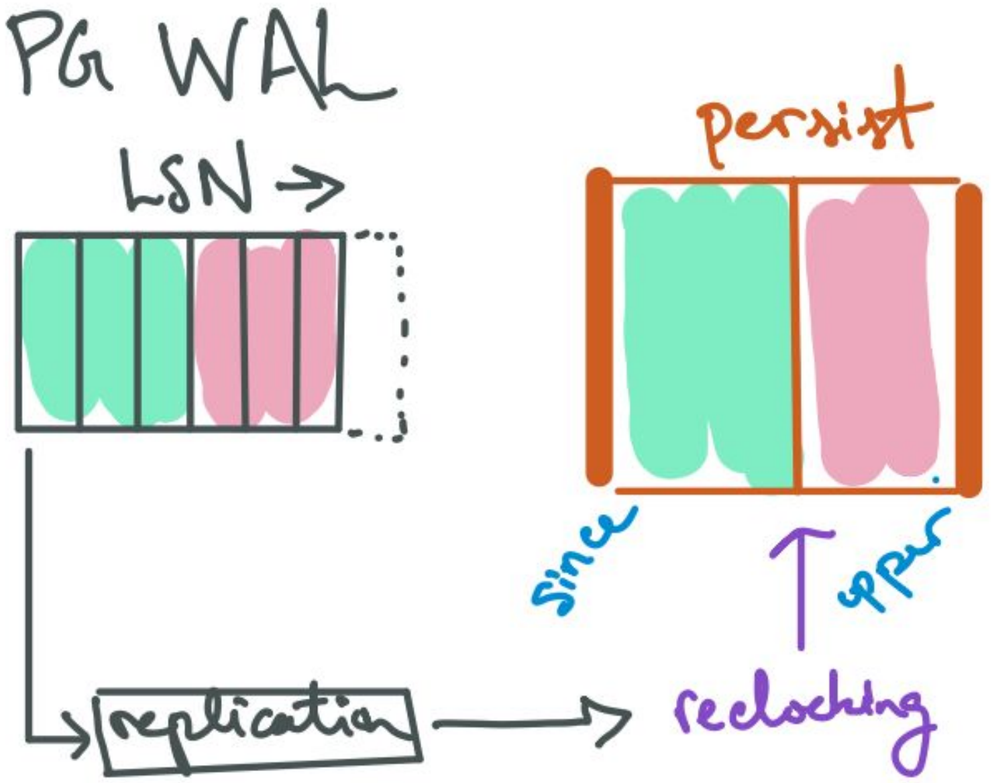
PostgreSQL replication



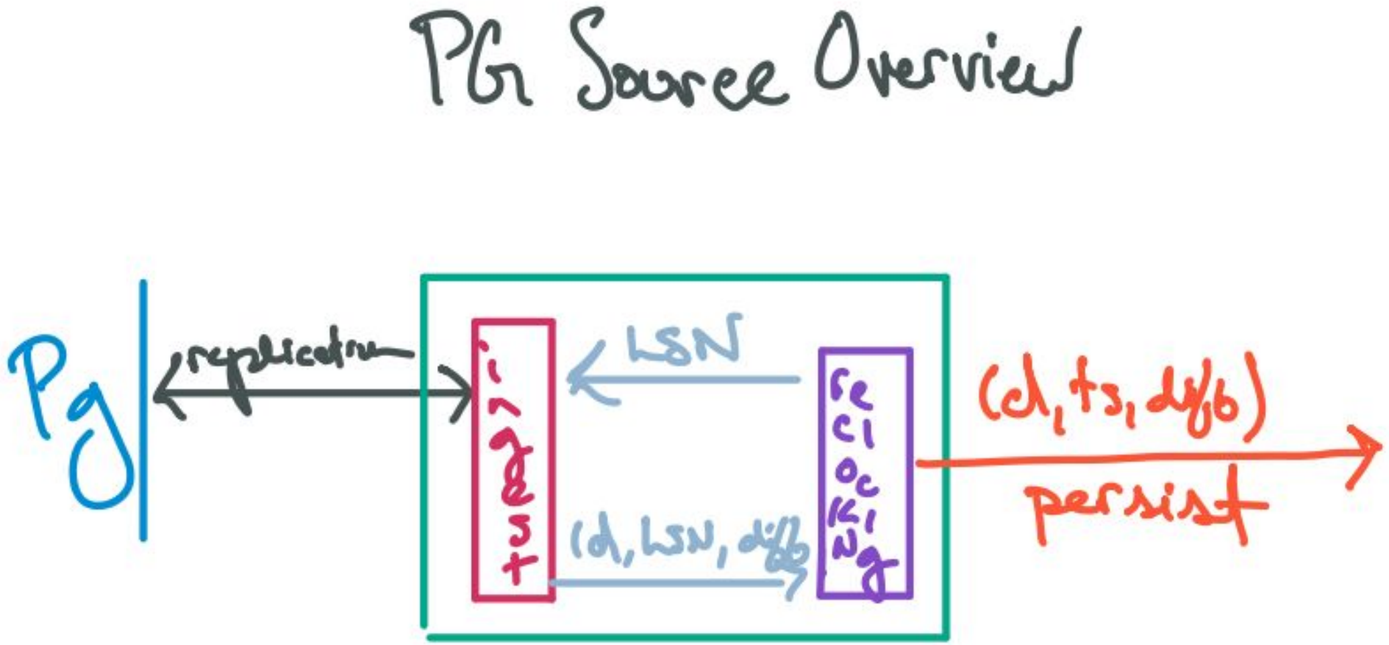
Ingesting data



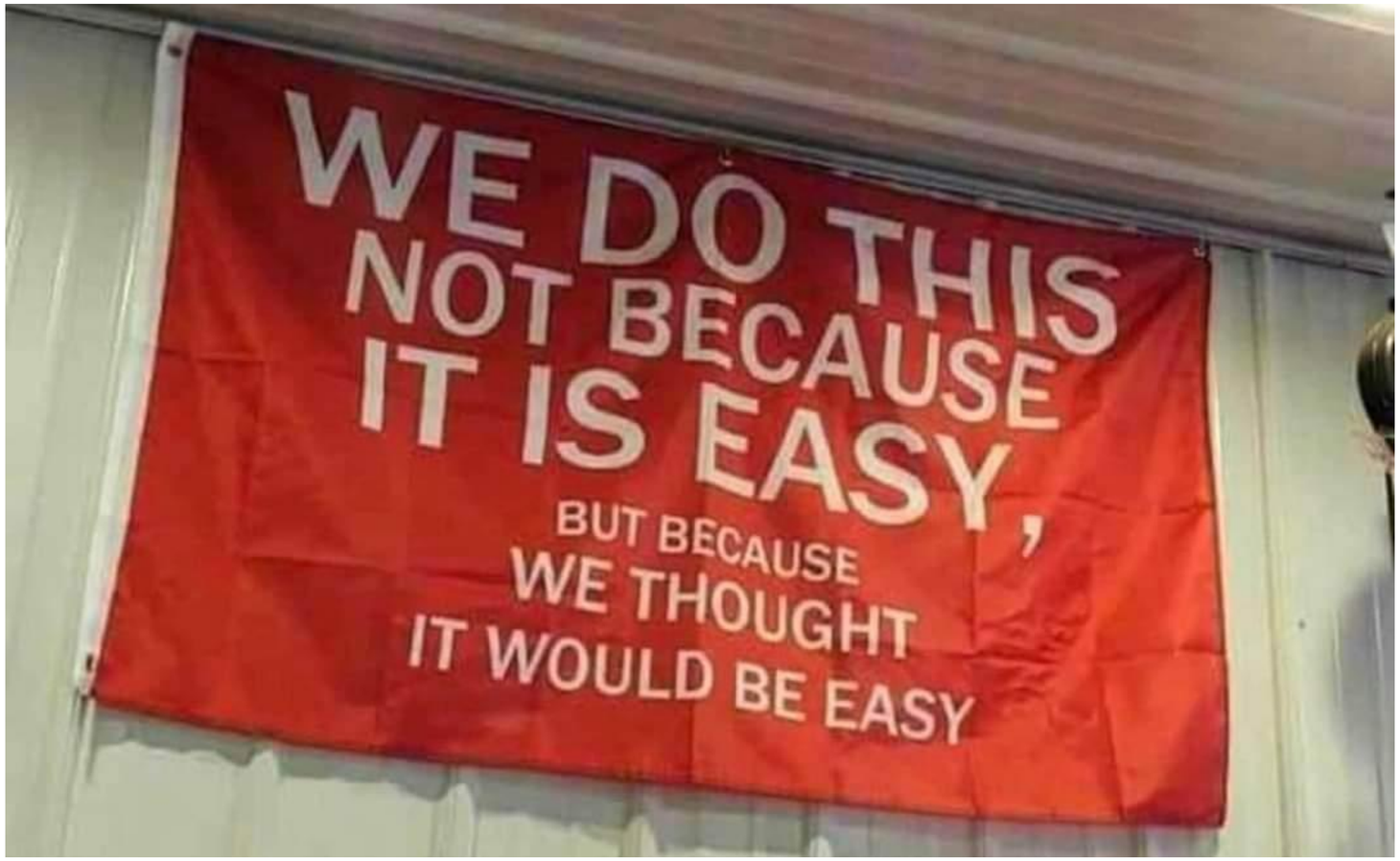
Ingesting data



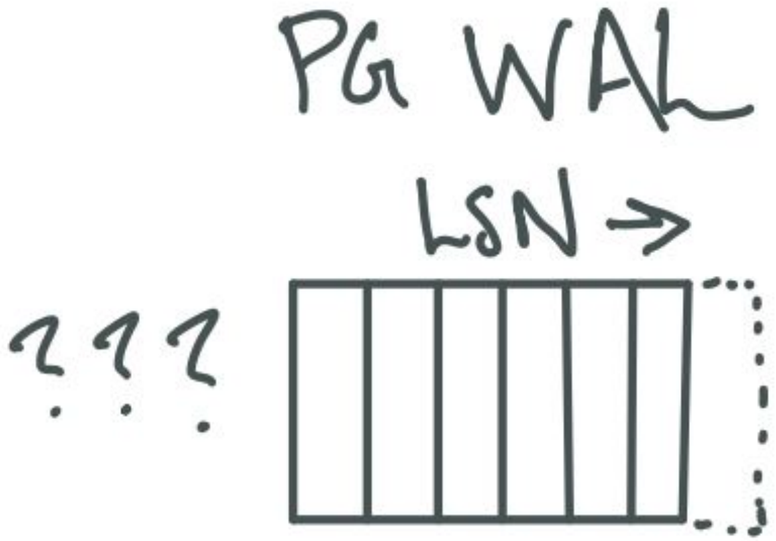
Ingesting data



Not so fast...


WE DO THIS
NOT BECAUSE
IT IS EASY,
BUT BECAUSE
WE THOUGHT
IT WOULD BE EASY

The WAL doesn’t contain everything

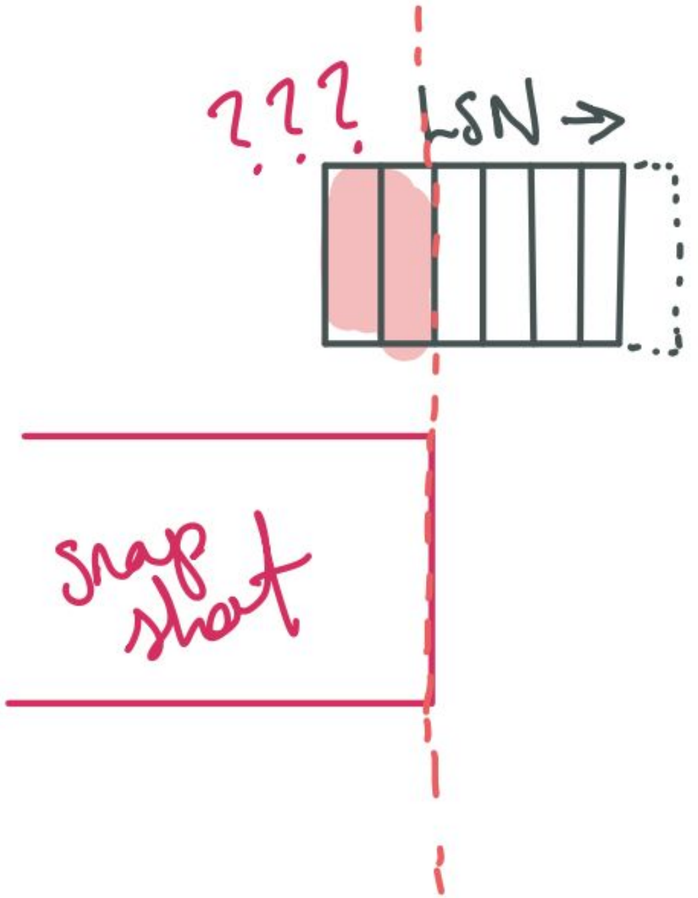


The WAL doesn't contain everything

```
--primary worker begins txn
BEGIN READ ONLY ISOLATION LEVEL REPEATABLE READ;
--replication slot returns consistent LSN
CREATE_REPLICATION_SLOT mz_slot123 TEMPORARY
LOGICAL "pgoutput" USE_SNAPSHOT;
--create exportable snapshot
SELECT pg_export_snapshot();
```

and then…

```
--secondary workers begin txns
BEGIN READ ONLY ISOLATION LEVEL REPEATABLE READ;
--enter exportable snapshot
SET TRANSACTION SNAPSHOT…;
```



Rewind requests

```
// A request to rewind a snapshot taken at `snapshot_lsn` to the initial LSN of the
replication

// slot. This is accomplished by emitting `(data, 0, -diff)` for all updates `(data, lsn,
diff)`

// whose `lsn <= snapshot_lsn`. By convention the snapshot is always emitted at LSN 0.

struct RewindRequest {

    /// The table OID that should be rewound.

    oid: u32,

    /// The LSN that the snapshot was taken at.

    snapshot_lsn: MzOffset,

}
```

REPLICA IDENTITY FUD

From PG 14:

If the table does not have any suitable key, then it can be set to replica identity “full”, which means the entire row becomes the key. This, however, is very inefficient and should only be used as a fallback if no other solution is possible.

Relation messages are inadequate

Relation

...

Int16

Number of columns.

  Next, the following message part appears for each column included in the publication (except generated columns):

String

Name of the column.

The truth is in the source code

```
Primary keepalive message (B)
Byte1('k')
Identifies the message as a sender keepalive.
Int64
The current end of WAL on the server.
...
```

The Correctness Bug™

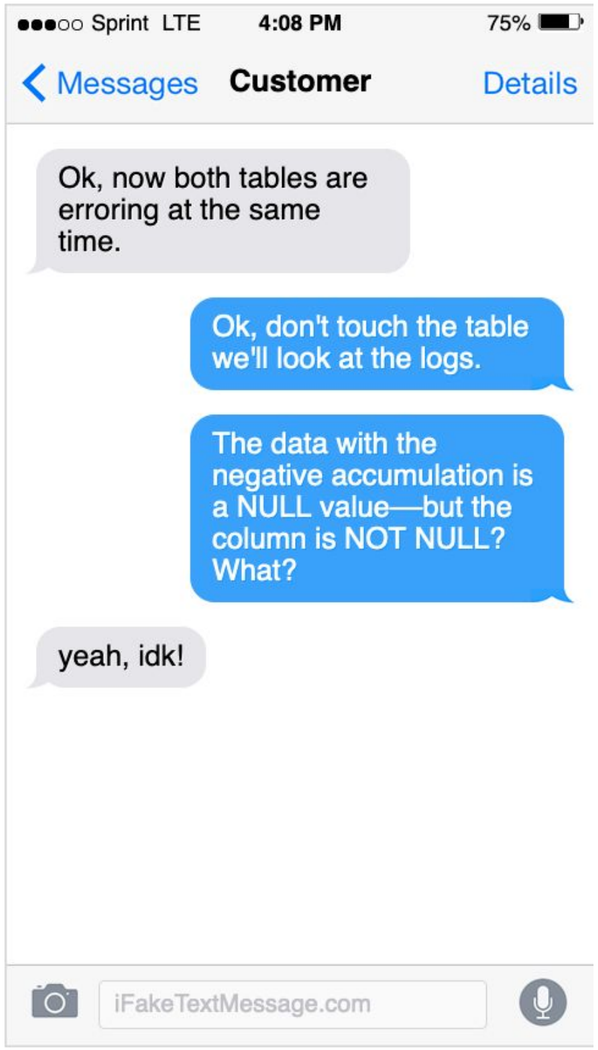
No negative multiplicities

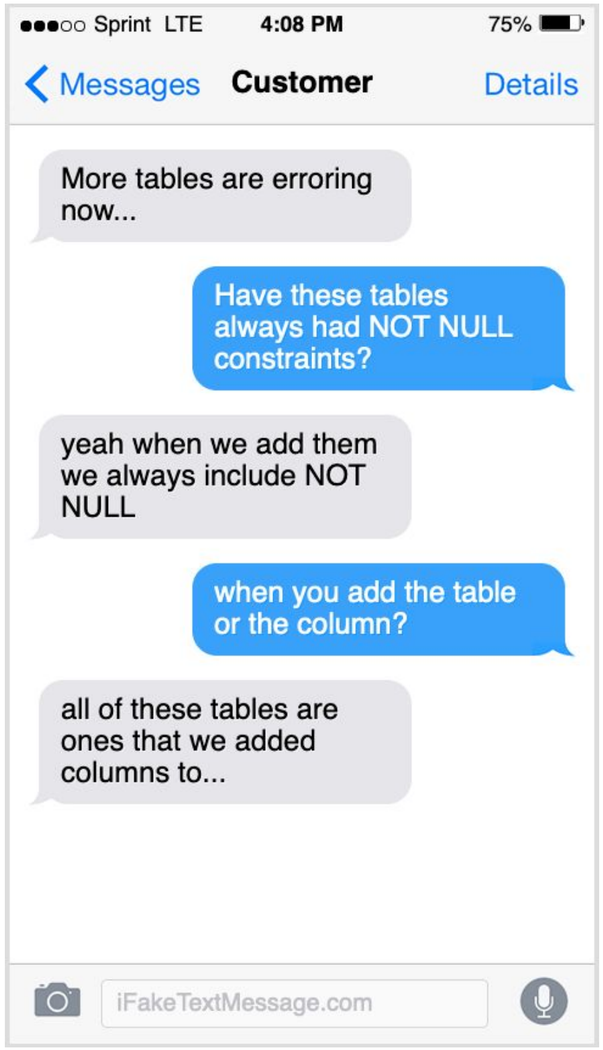
```
for (datum, time, diff) in data {
    // SQL is underpowered
    assert!(diff >= 0);
}
```

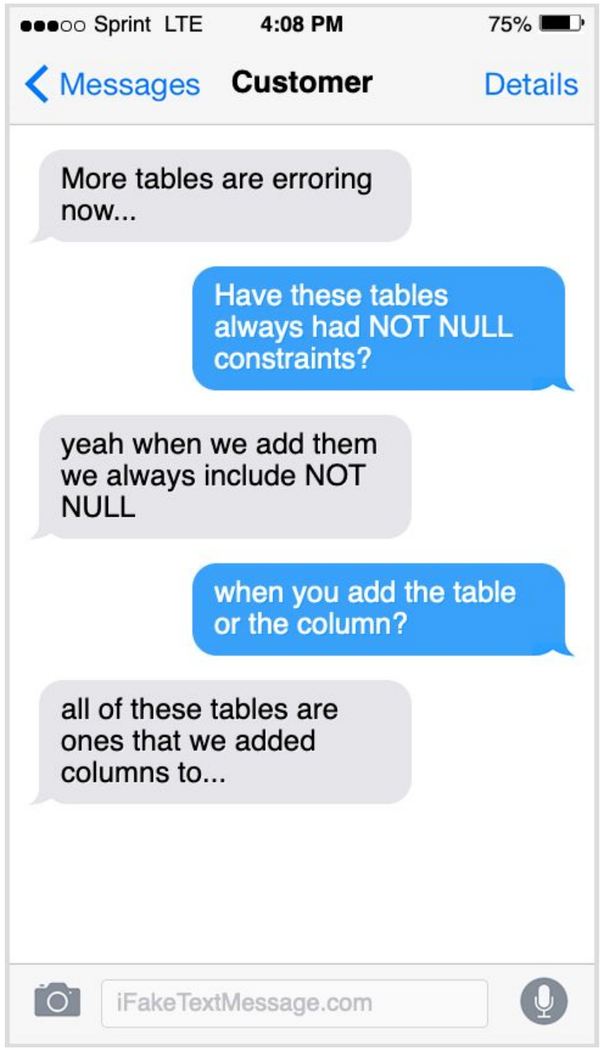

●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
Getting "Invalid data in source, saw negative accumulation" for table OID(123)
Weird
Ok, we recreated the source and the error went away
iFakeTextMessage.com


●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
We're back with the same error about negative accumulations...
Weird
Ok, we recreated the source and the error went away
iFakeTextMessage.com


●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
Ok, mow the error's showing up for another table--456
Weird. Is the old table erroring?
No. We recreated the source and the error went away
iFakeTextMessage.com


●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
Ok, now both tables are erroring at the same time.
Ok, don't touch the table we'll look at the logs.
The data with the negative accumulation is a NULL value—but the column is NOT NULL? What?
yeah, idk!
iFakeTextMessage.com


●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
More tables are erroring now...
Have these tables always had NOT NULL constraints?
yeah when we add them we always include NOT NULL
when you add the table or the column?
all of these tables are ones that we added columns to...
iFakeTextMessage.com


●●●○○ Sprint LTE
4:08 PM
75%
Messages
Customer
Details
More tables are erroring now...
Have these tables always had NOT NULL constraints?
yeah when we add them we always include NOT NULL
when you add the table or the column?
all of these tables are ones that we added columns to...
iFakeTextMessage.com

RE: Logical Replication - Should Destination Table Columns Be Defined With Default Value

I also thought that default value on the subscriber side are immaterial. However, with the case I showed without having default value on subscriber side it get null when the following occurs:

1. Table was created with two columns on publisher and subscriber side
2. Data inserted into this table
3. A third column is added to table with default value on publisher side, but without default value on subscriber side

    n.b. these are two ALTER commands

4. The default value column has value for existing rows on publisher, but null on the subscriber side.
5. Doing refresh publication etc. does not help and the column on subscriber side remains with nulls

...adding a column with a constant default value no longer means that each row of the table needs to be updated when the ALTER TABLE statement is executed. Instead, the default value will be returned the next time the row is accessed, and applied when the table is rewritten, making the ALTER TABLE very fast even on large tables.

However, we cannot reproduce our customer's behavior on PG 14...but it turns out that they were using PG 15....

```
CREATE TABLE t (a INT, b INT);

INSERT INTO t VALUES (1, 2), (3, 4);

--because of fast alter, this looks like (1, 2, null)...

ALTER TABLE t ADD COLUMN c INT NOT NULL DEFAULT 1;

--now set up the MZ source

UPDATE t SET c = a;

--the old value of C that gets propagated through logical replication is NULL
```

Correctness

And here is the bug fix Nikhil put together.

Q&A

Thank You