Shipping Data from Postgres to ClickHouse

Murat Kabilov, Adjust GmbH

Who am I

Shipping from Postgres to ClickHouse

- psql -c "copy … to stdout" | clickhouse-client --query "INSERT INTO …"
- clickhouse FDW
- trigger-based solutions, pgq
- via Kafka

or you can use logical replication

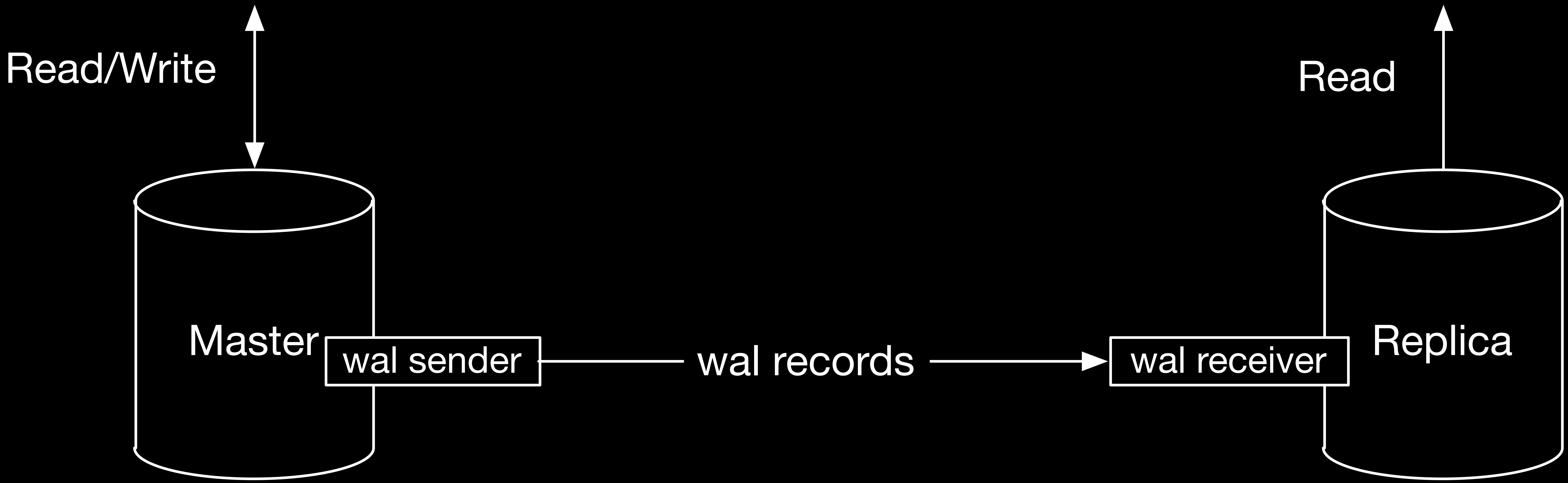
Replication in Postgres

- WAL: write-ahead log contains binary changes of the data files

- LSN: log sequence number, 64-bit integer representing a byte position in the WAL stream

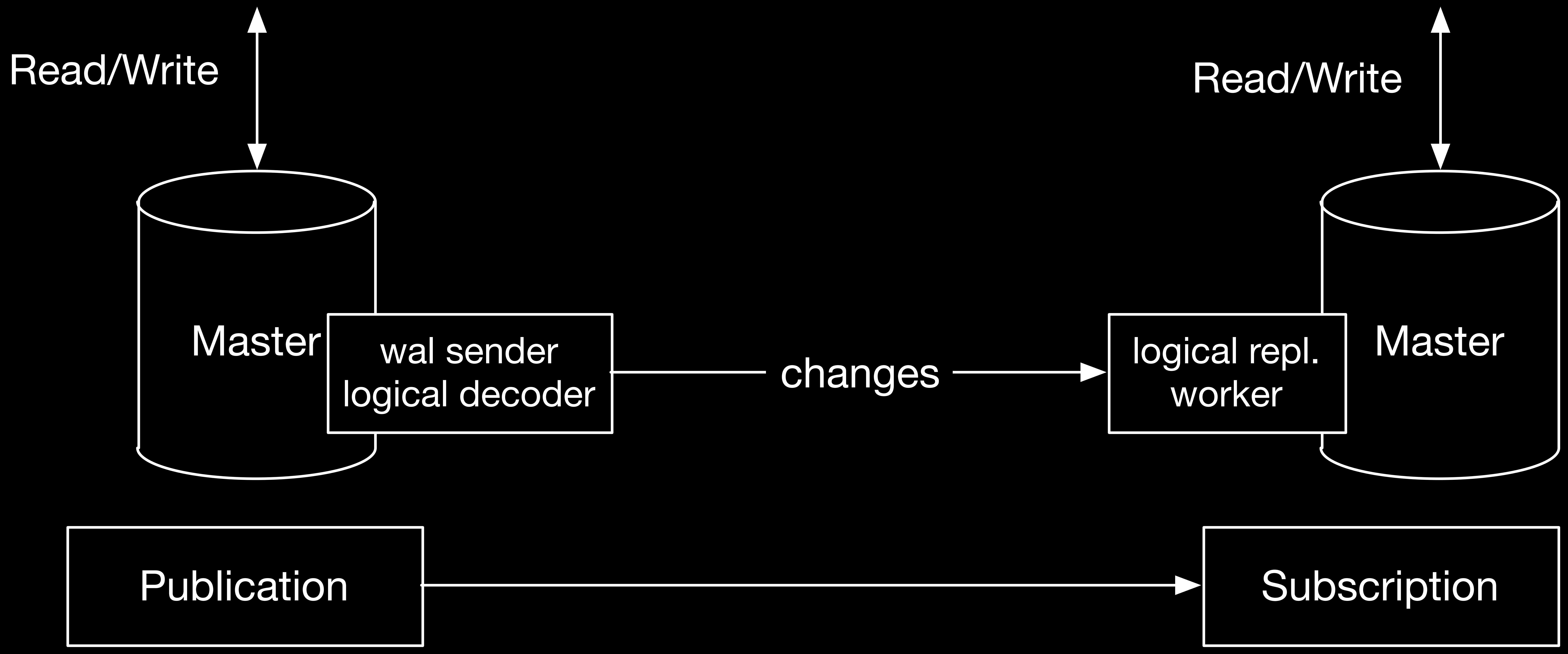
$$\begin{tikzcd}
 & \text{Replication} \arrow[dl] \arrow[dr] & \\
\text{Physical} & & \text{Logical}
\end{tikzcd}$$

Physical replication



byte-to-byte, the whole instance is replicated. replica is read-only

Logical replication



postgres >=10; only DML commands are replicated

Output plugins

```
- built-in one: pgoutput
- decoderbufs (https://github.com/debezium/postgres-decoderbufs)
- wal2json (https://github.com/eulerto/wal2json):
      "change": [
    {
       "kind": "insert",
       "schema": "public",
       "table": "table_with_pk",
       "columnnames": ["a", "b", "c"],
       "columntypes": ["int4", "varchar", "timestamp"],
       "columnvalues": [1, "Backup and Restore", "2015-08-27 16:46:35.818038"]
    }]

- decoding-json (https://github.com/leptonix/decoding-json):
  {"type":"transaction.begin","xid":"2010561","committed":"2015-04-22
  19:23:35.714443+00"}
  {"type":"table","name":"abc","change":"INSERT","data":{"a":6,"b":7,"c":42}}
  {"type":"table","name":"abc","change":"UPDATE","key":{"a":6,"b":7},"data":
  {"a":6,"b":7,"c":13}}
```

Logical replication

- Publisher/Subscriber model
- DML commands to replicate can be specified: `insert, update, delete, truncate`
- Data is streamed only when transaction is committed
- Uses built-in `pgoutput` output plugin

Publication

```
CREATE PUBLICATION name
    [ FOR TABLE [ ONLY ] table_name [ * ] [, …] | FOR ALL TABLES ]
    [ WITH ( publication_parameter [= value] [, ... ] ) ]
```

```
e.g.
CREATE PUBLICATION my_pub FOR ALL TABLES WITH (publish='insert');
```

How?

- INSERT
- TRUNCATE (starting from pg 11)
- UPDATE/DELETE
  - we need to somehow identify old version of the row

Replica identity

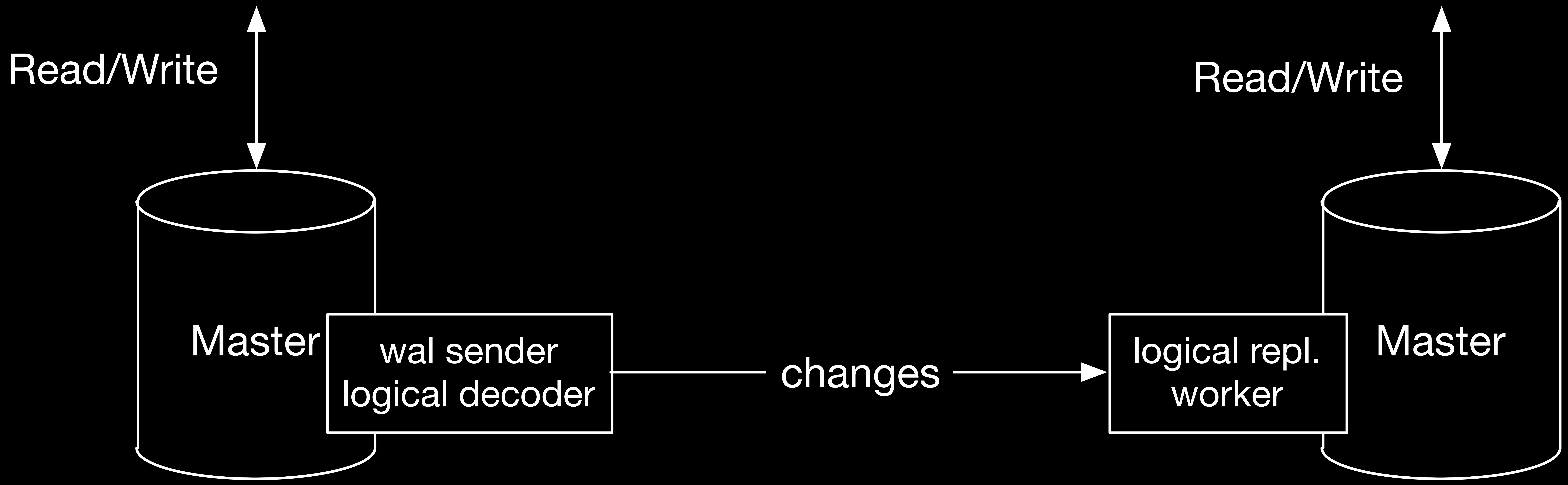
```
ALTER TABLE … REPLICA IDENTITY …;
```

- Default: uses Primary Key
- Using index: uses unique index
- Full: uses all the columns of the row
  old values of all the columns are sent
- Nothing

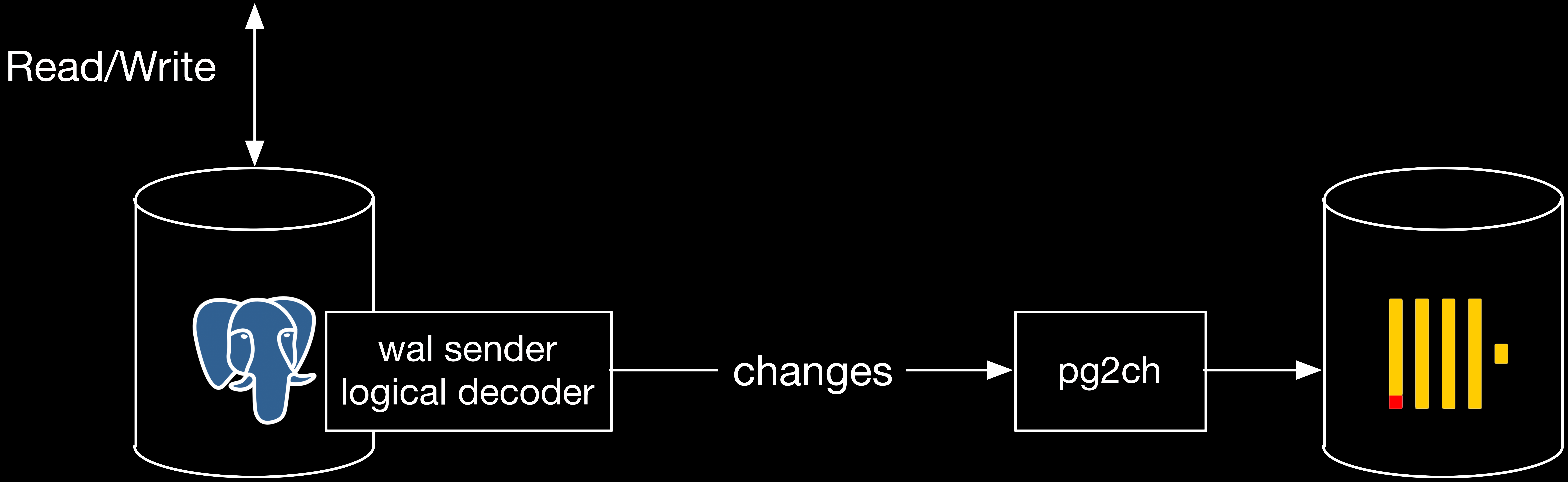
pgoutput

- **Begin:** FinalLSN:0/2384C888 Timestamp:2019-03-15T13:00:41Z XID:870035
  - **Relation:** OID:16414 Name:pgbench_accounts Replica Identity:full Columns:[…]
  - **Update:** Relation OID:16414 newValues:[…] oldValues:[…]
  - **Relation:** OID:16408 Name:pgbench_history Replica Identity:full Columns:[…]
  - **Insert:** Relation OID:16408 values:[…]
  - **Delete:** Relation OID:16414 values:[…]
- **Commit:** LSN:0/2384C888 Timestamp:2019-03-15T13:00:41Z TxEndLSN: 0/2384C8B8

Logical replication



pg2ch



pg2ch

- written in Go
- can create initial copy and keeps the position of the changes
- uses vanilla postgres (ver ≥10), no plugins/extensions required
- uses internal buffer to accumulate the data
- can use intermediate buffer table on the ClickHouse side

pg2ch

```
tables:
    pgbench_accounts:
        main_table: ch_accounts
        engine: CollapsingMergeTree
        sign_column: sign
        max_buffer_length: 1000

...
```

```
clickhouse:
    host: localhost
    database: default
    username: default

pg:
    host: localhost
    database: pg2ch
    user: postgres
    replication_slot_name: my_slot
    publication_name: my_pub

lsn_state_filepath: state.yaml
inactivity_flush_timeout: ‘30s'
```

pg2ch

- currently supports *MergeTree, ReplacingMergeTree and CollapsingMergeTree* table engines

CollapsingMergeTree

- requires `sign` column in the table on the ClickHouse side
- requires FULL Replica Identity for the replicating table
- on UPDATE inserts two rows:
  - with -1 in the `sign` column to “cancel” row (thanks to FULL replica identity)
  - with 1 to “state” row
- on DELETE only “cancel” row is inserted

CollapsingMergeTree

user_id	name	surname	sign
42	John	Doe	1

↓

user_id	name	surname	sign
42	John	Doe	1
42	John	Doe	-1
42	Richard	Doe	1

ReplacingMergeTree

- requires `version` column in the table on the ClickHouse side
- LSN (UInt64) is used as a version
- What to do with DELETES?

ReplacingMergeTree

user_id	name	surname	ver
1	John	Doe	1000

↓

user_id	name	surname	ver
1	John	Doe	1000
1	Richard	Doe	1003

MergeTree

- only INSERTS operations are replicated
- DELETE/UPDATES are discarded

Thank you!
Questions?

Links

- https://github.com/mkabilov/pg2ch
- https://www.postgresql.org/docs/current/logical-replication.html
- https://wiki.postgresql.org/wiki/Logical_Decoding_Plugins
- https://www.postgresql.org/docs/current/protocol-logicalrep-message-formats.html
- https://clickhouse.yandex/docs/en/operations/table_engines/
- https://github.com/Percona-Lab/clickhousedb_fdw