PostgreSQL Logical Decoding

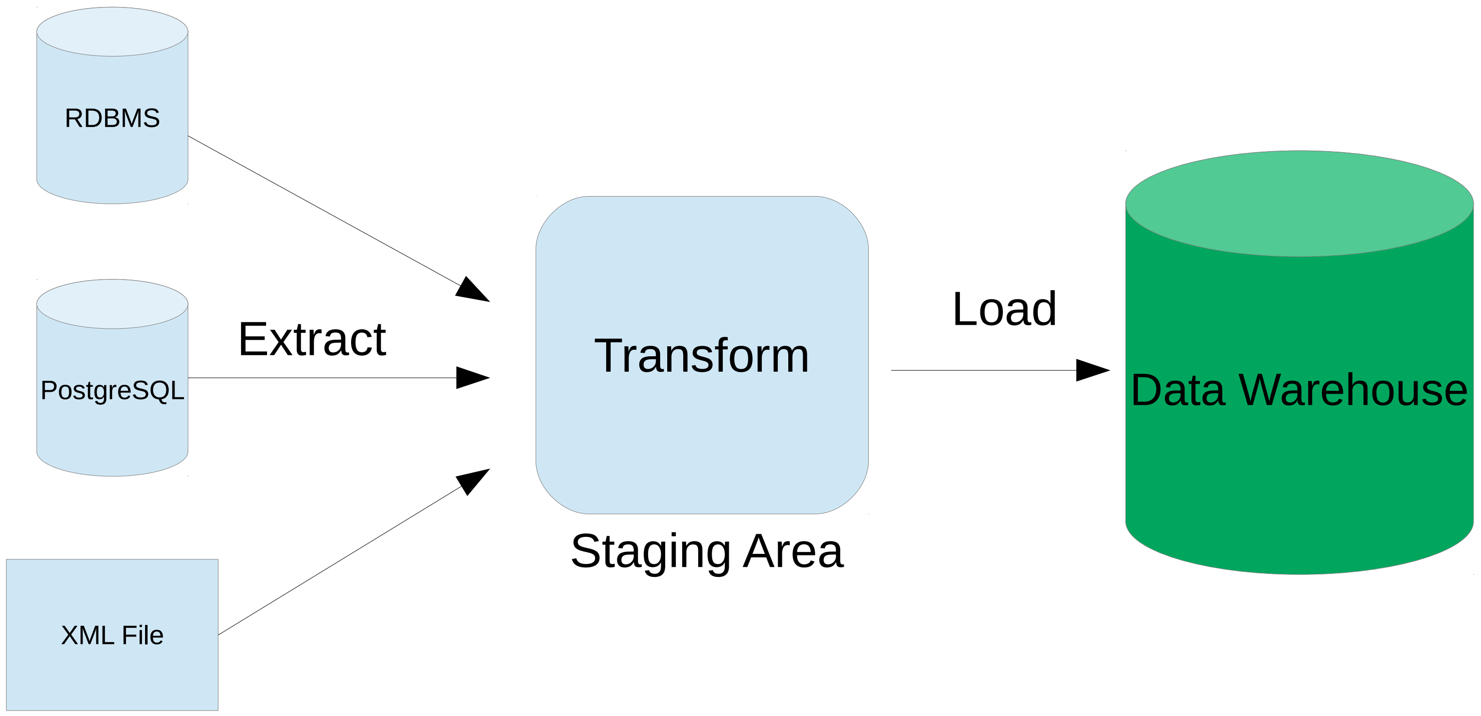
- Amit Khandekar

What is Logical Decoding

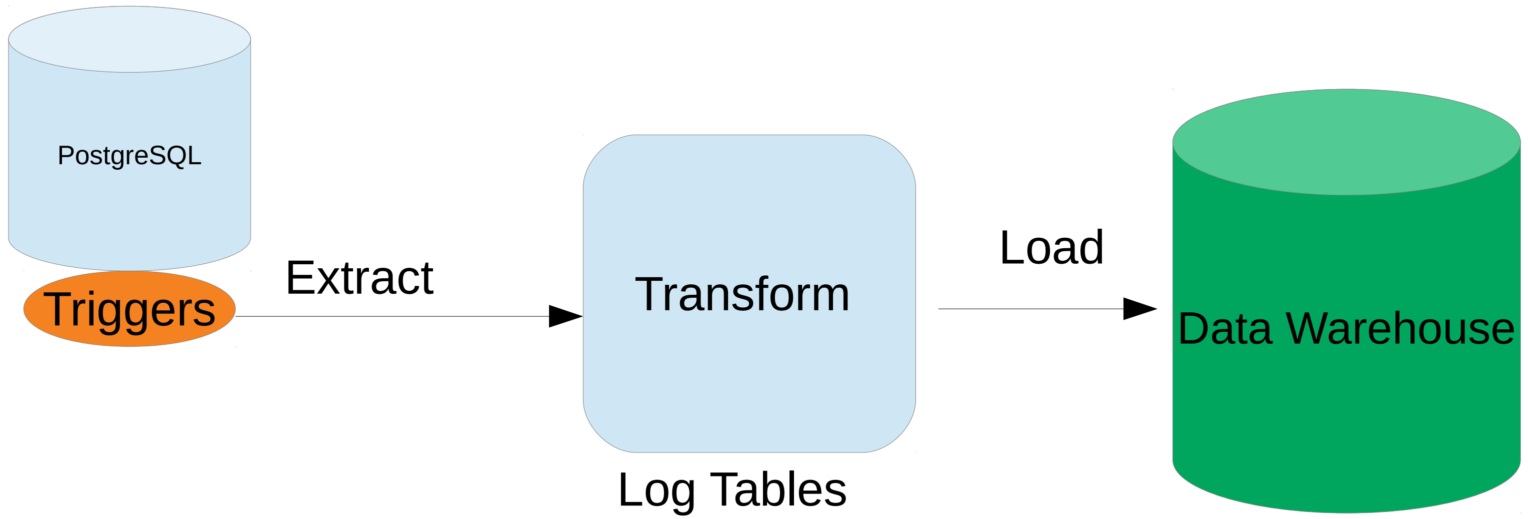
- Extracting db changes as they occur, in a simple format that can be interpreted by an external entity.

- Outside PostgreSQL, it is also called log-based change data capture (CDC)

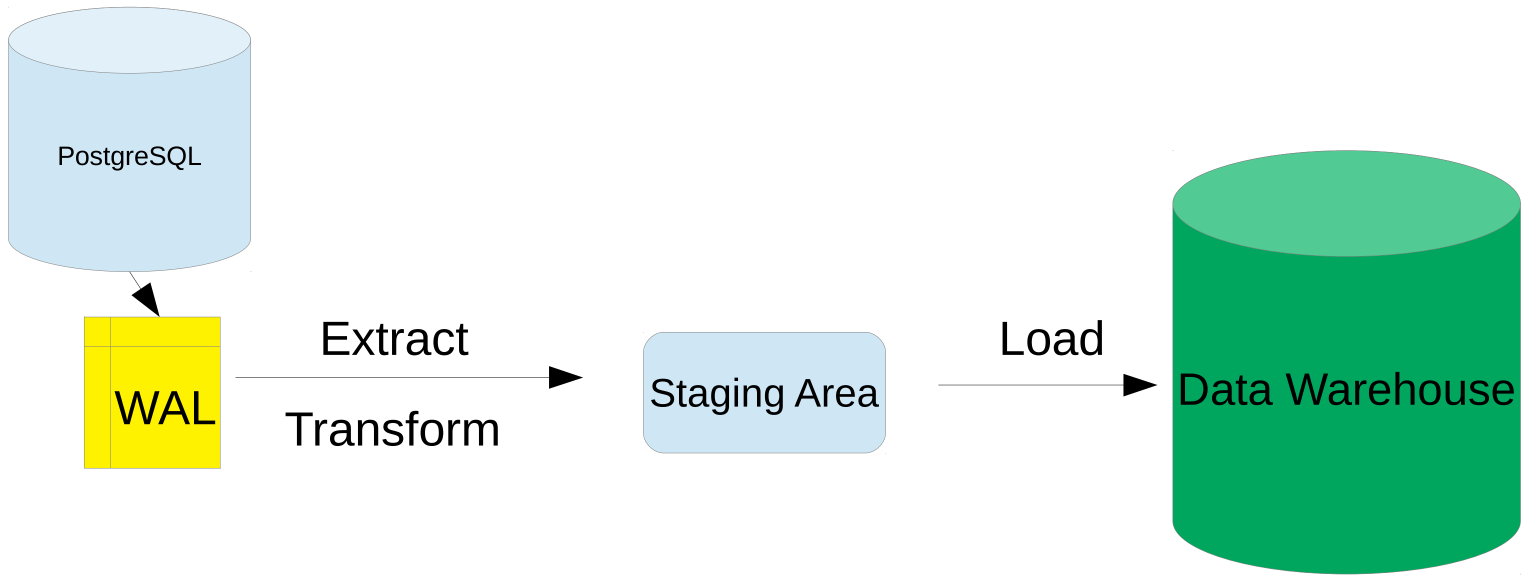
Change data capture for ETL



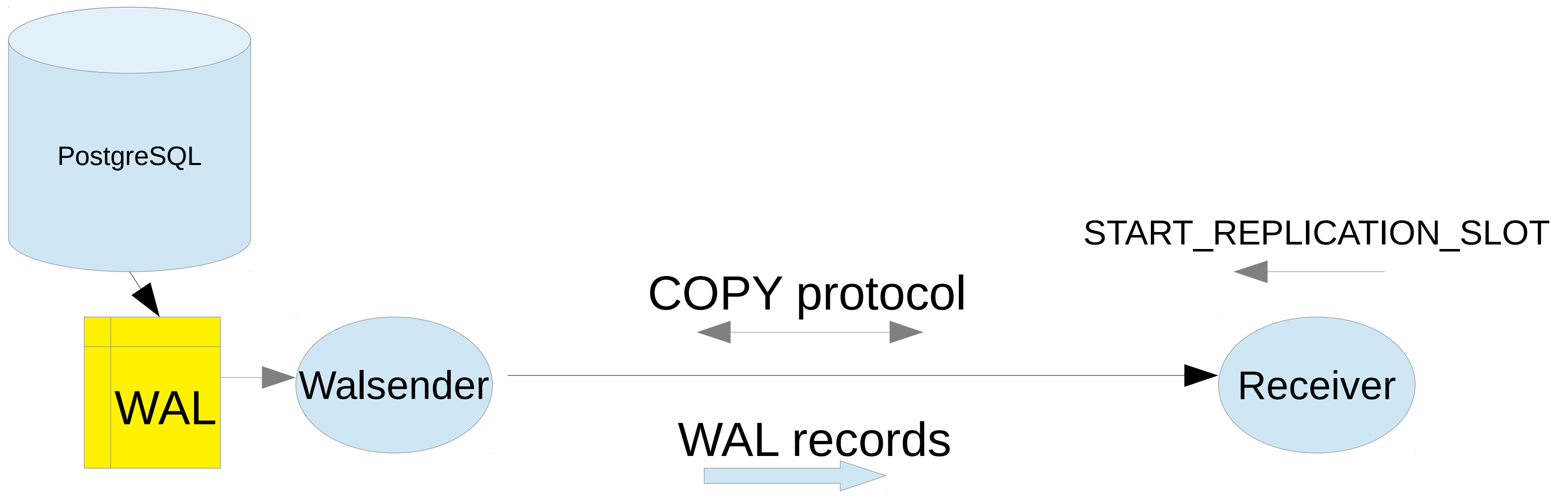
Triggers for change data capture



WAL for change data capture



Streaming Replication protocol



Requesting a logical log

- CREATE_REPLICATION_SLOT <slot_name> LOGICAL
- set wal_level = logical
- Replication slot is mandatory
- max_replication_slots should be at least 1

Output Plugin

- Client provides callback functions to server
- begin_cb(), commit_cb(), change_cb() are mandatory
- These functions are meant to emit the WAL log records from server.
- Create a shared library that has these function definitions
- Pass it while creating logical replication slot

  CREATE_REPLICATION_SLOT <slot_name> LOGICAL <output_plugin_file>.so
- Sample plugin provided for testing : test_decoding.so

Output Plugin (Cont)

```
void _PG_output_plugin_init(OutputPluginCallbacks *cb)
{
        .......
    cb->startup_cb = pg_decode_startup;
    cb->begin_cb = pg_decode_begin_txn;
    cb->change_cb = pg_decode_change;
    cb->truncate_cb = pg_decode_truncate;
    cb->commit_cb = pg_decode_commit_txn;
        .......
    cb->shutdown_cb = pg_decode_shutdown;
        .......
    cb->stream_change_cb = pg_decode_stream_change;
    cb->stream_truncate_cb = pg_decode_stream_truncate;
}
```

Replication slots

- Handle to the current position (LSN) in the WAL stream.
- Can also be used for physical streaming replication
- Can be thought of as a file pointer which advances at each read.
- Guarantees that required WAL is retained until consumed.
- Retains even after server restart.
- If not consumed, WAL log may eventually consume all disk space.
- Drop slot if not needed

Receiving logical records

Command line : pg_recvlogical

```
$ pg_recvlogical -d postgres --create-slot  -S myslot

postgres=# update tab set v = 'd', id1 = 10  where id1 = 11;
UPDATE 1

$ pg_recvlogical -d postgres -S myslot --start -f -
BEGIN 543
table public.tab: UPDATE: id[integer]:251 id1[integer]:11 v[character varying]:'d'
COMMIT 543
```

Receiving logical records (cont.)

SQL level API

```
postgres=# update tab set v = 'd', id1 = 11  where id1 = 10;
UPDATE 1

postgres=# SELECT * FROM pg_logical_slot_get_changes('myslot', NULL, NULL);
    lsn    | xid |                                   data
-----------+-----+-----------------------------------------------------------------------------
 0/16A7588 | 545 | BEGIN 545
 0/16A7588 | 545 | table public.tab: UPDATE: id[integer]:251 id1[integer]:11 v[character varying]:'d'
 0/16A7808 | 545 | COMMIT 545
```

Points to be noted

- Logical decoding skips DDL operations.
- Each replication slot decodes changes from one particular database.
- change_cb() does not get called until transaction commit.
  - It means, receiver receives changes only after they are committed.
- Upcoming PostgreSQL 14 added support to decode uncommitted changes using different set of callbacks.

Replica Identity

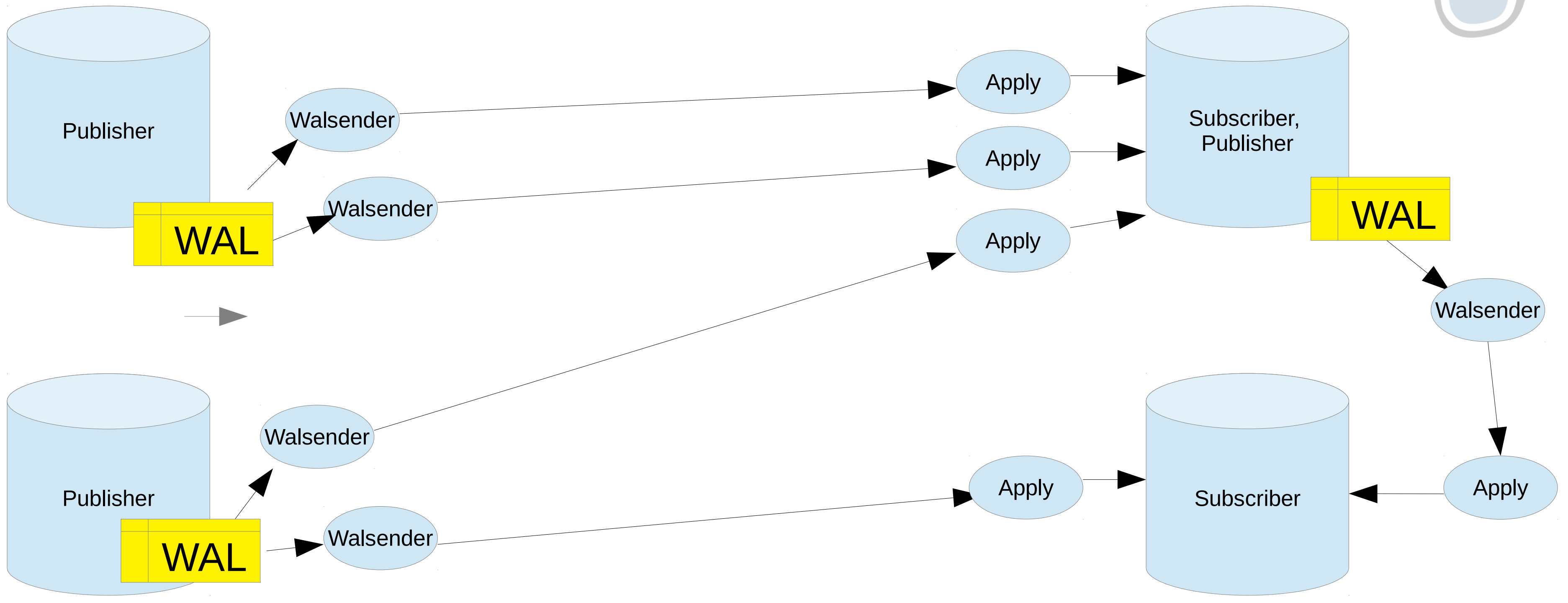
- ALTER TABLE <table_name> REPLICA IDENTITY
  [NOTHING | USING INDEX <name> | FULL]
- A way to specify which unique index column should be included in the OLD tuple for UPDATE and DELETE records.
- Used only for logical decoding.

Logical Replication

```
CREATE PUBLICATION insert_only_publication FOR TABLE mydata
    WITH (publish = 'insert');

CREATE SUBSCRIPTION mysub
         CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb'
         PUBLICATION mypublication, insert_only_publication;
```

Publisher-Subscriber model



Use Cases

- CDC tools like debezium use wal2json plugin and Postgres Decoderbufs
- Multi-master replication (Postgres-BDR, pglogical)
- Online upgrade
- Redundancy in sharded tables
- Aggregation : Plugin itself can output aggregated data
- Replicate to a foreign table using foreign data wrapper

Todos / Upcoming features

- Streaming large in-progress transactions (stream_commit_cb)
  - Will be available in upcoming Postgres 14
- Support for streaming transaction records for two-phase transactions.
  - Will be available in upcoming Postgres 14
- Replication slots are not synced to hot standbys - Work in progress
- Logical replication of a sequence is not supported - Work in progress
- Parallelism in Logical decoding (and also during apply)

Questions ?