Demystifying Postgres Logical Replication

Emanuel Calvo
April 2017
Percona Live 2017 - Santa Clara

Who am I?

- Data Engineer / 3manuek.com
- Currently working as a Remote DBA at Percona.
- Worked at PalominoDB, 2ndQuadrant, 8kdata, iMedicare, Pythian among others.
- Being oscillating between MySQL, Postgres, Oracle and other data solutions.

Agenda

- The path of replication in Postgres
- Streaming Replication and Logical Replication
- Replication flow for MySQL DBAs
- Feature capabilities
- Postgres Logical Replication for MySQL DBAs
- What can be done with this
- Limitations
- Elements
- Conflicts
- Monitoring

The path of replication in Postgres

- Streaming replication incorporated in 9.0.
- Cascading streaming replication introduced in 9.2.
- Switch timeline added in 9.3.
- Logical Decoding added in 9.4.
- More support to LD added in 9.6.
- Postgres 10 Logical replication natively supported.

Streaming and logical replication

- Streaming replication is a byte-by-byte replication, the whole instance (all databases) are replicated.
- Logical replication is supported through pglogical for +9.4
- Natively supported in the next Postgres release.

Replication flow for MySQL DBAs

- MySQL
  - Engine log + Binlog -> byte encoded -> bingo stream -> binlog apply
  - Cross-engine Events are append to the binlog (unless skipped sql_log_bin)
  - Slaves filter using do%
  - Row_format: Replicates the change or the complete statement
- Postgres
  - WAL -> Logical Decoding/output_plugin -> logical log -> sender -> receiver & apply
  - Filtering is done at publisher
  - Closer to row based replication

Feature capabilities

- LR replicates data objects based upon their replication identity (generally a primary key).
- Destination server is writable. Different indexes and security definition.
- Cross-version support
- Event-based filtering
- Less write amplification than streaming replication
- Publications can have several subscriptions

What can be achieved with LR?

- Storage flexibility through replicating smaller sets (even partitioned tables)
- Flexible topology
- Minimum server load compared with trigger based solutions
- Allows parallel streaming across publishers
- Migrations and upgrades
- Multi source replication for consolidation
- Data distribution
- Flexible replication chains
- Data transformation

Limitations

- Can’t stream over to the same host (subscription will get locked).
- Tables must have the same full qualified name between publication and subscription.
- Subscriptions can have more columns or different order but the types and column names must match between P/S.
- Database superuser is needed for P/S creation.

Elements

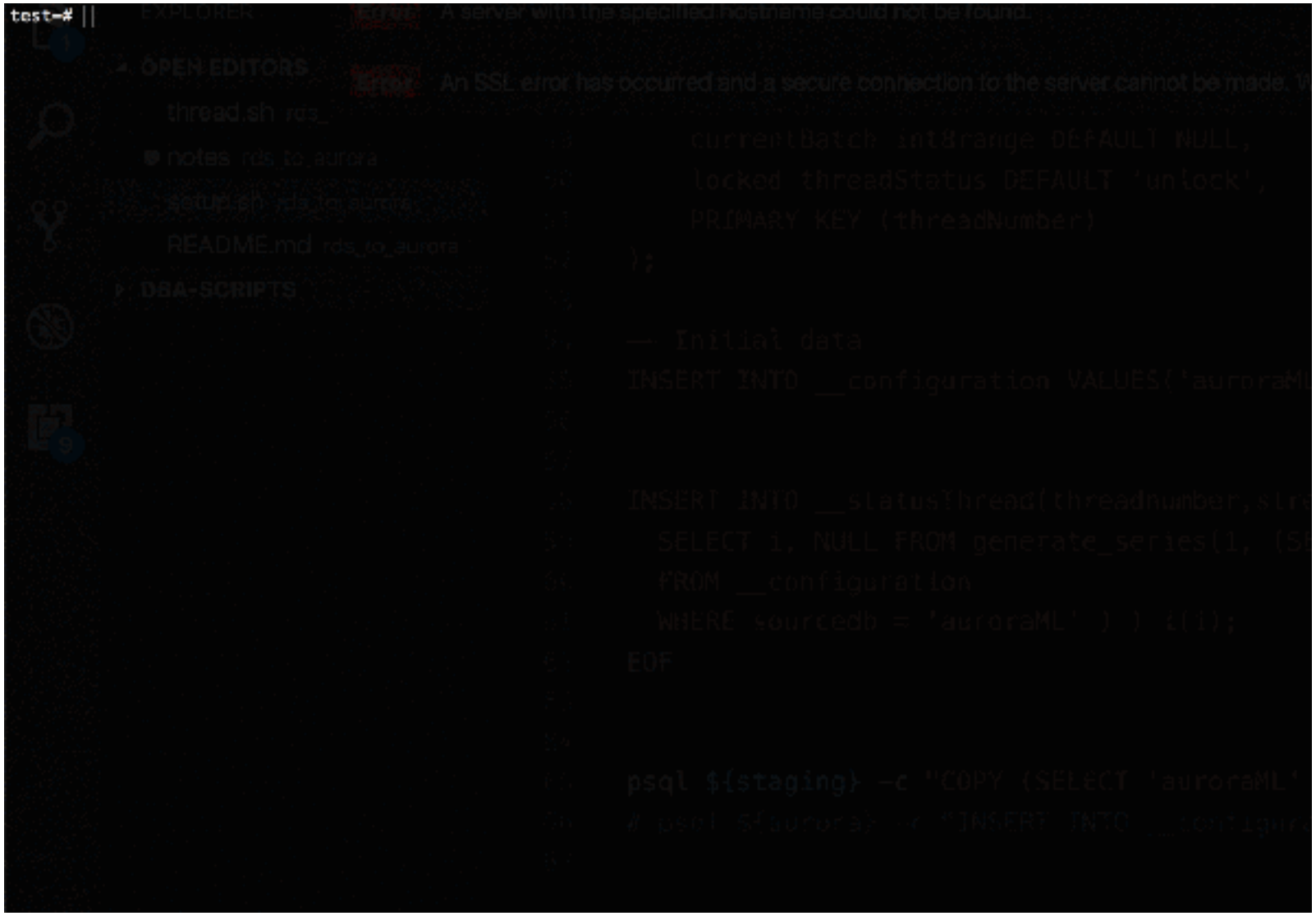
- Logical Decoding
  - Replication Slots
  - Output plugin
- Exported Snapshot
- Publication
- Subscription

[Logical] Replication Slots

- Keep track of the replication.
- Each replica stream has one in the origin for tracking consuming changes.
- Locations are explicitly in LSN (log sequence number).
- catalog_xmin is the transaction number
- Slots are placed in the origin.

slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn
s_data3	pgoutput	logical	16384	percona	f	t	5089		568	0/15FFD78	0/15FFE90
s_data	pgoutput	logical	16384	percona	f	t	1948		568	0/15FFE58	0/15FFE90

Example [Logical] Replication Slots



Output Plugin

- Converts WAL records entries into custom output
- Internal plugin name is `pgoutput`.
- For testing Logical Decoding capabilities, test_decoding.

Exported snapshot

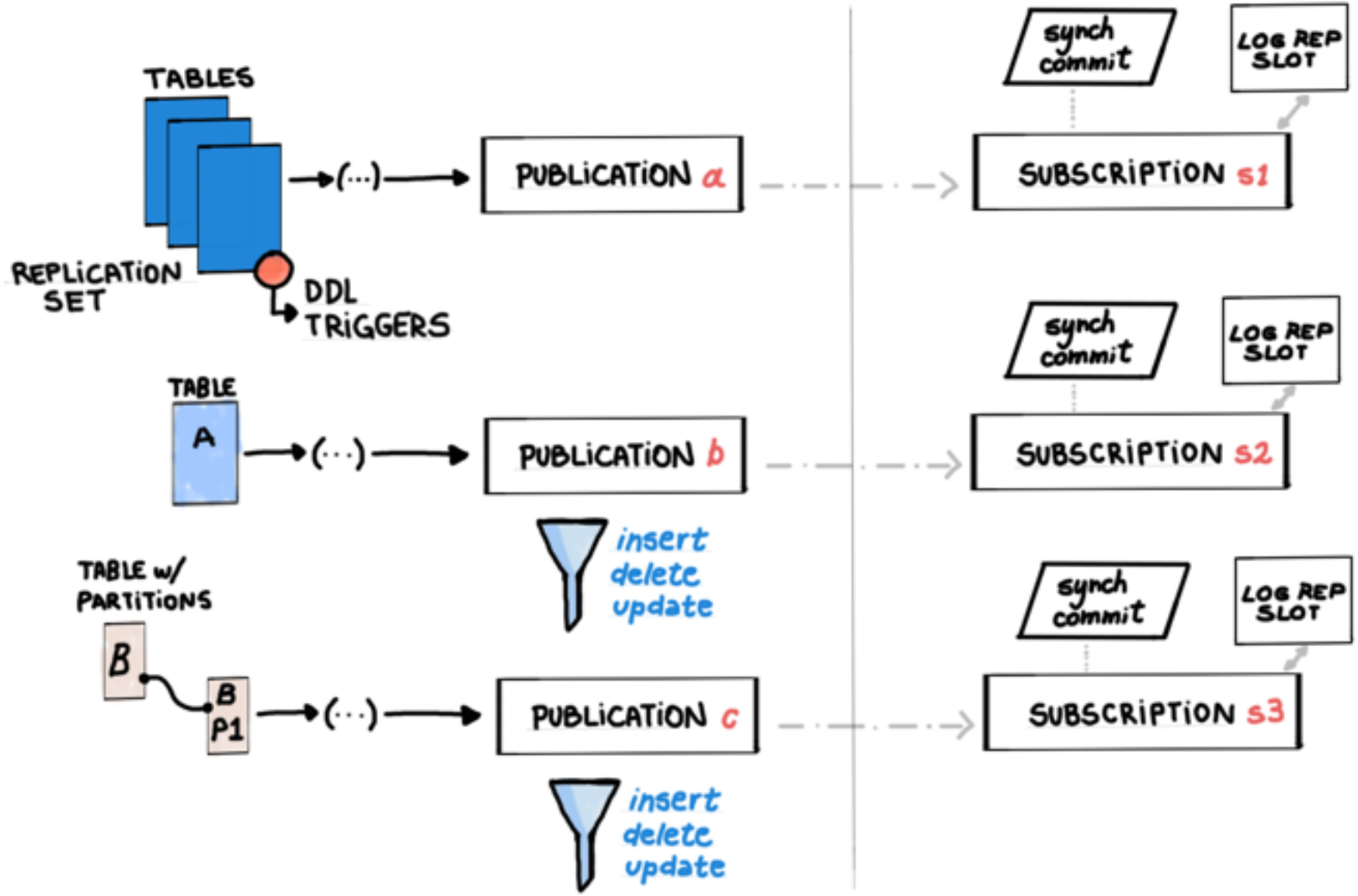
- Sharing visibility between transactions by exporting the current snapshot of the transaction.
- This is used for the initial COPY.
- Can be used to query outside a transaction but sharing its visibility.

Publication

- Publications can have more than one subscriber.
- Tables added to the publication must be declared with REPLICA IDENTITY. Otherwise subsequent operations will fail.

Subscription

- Subscriptions receive changes through replication slots.
- More than one replication slot may needed for the initial data copy.
- Initial COPY data is done by pg_dump.
- The session_replication_role is set to replica in order to manage triggers on tables as expected.
- DDL of replicated tables must previously exist.
- If creating a replication slot, it will use the name of the subscriber, so beware as slots are in the origin (you will need to specify different subscription names across subscribers).

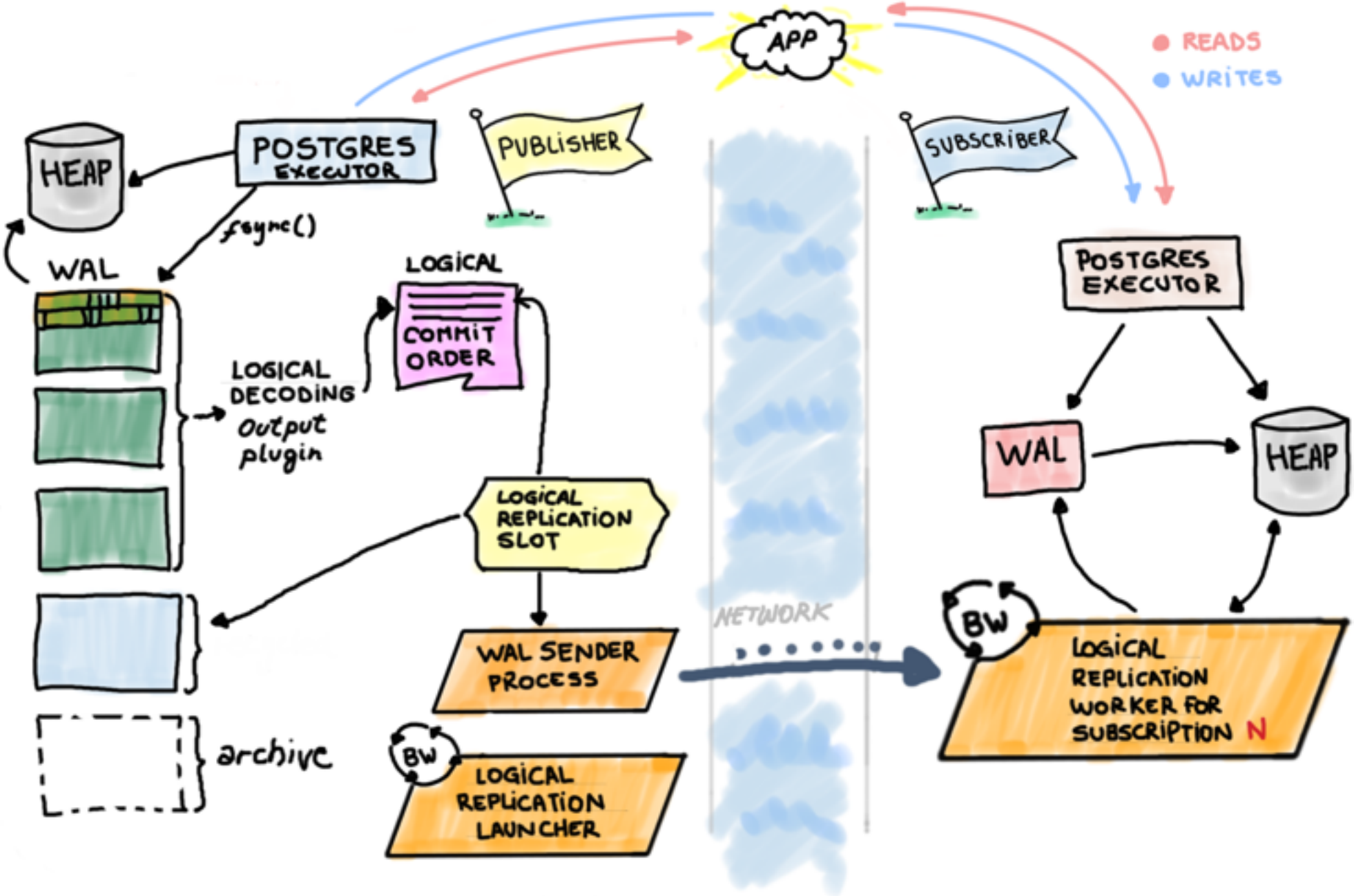

TABLES
(...)
PUBLICATION a
synch commit
LOG REP SLOT
SUBSCRIPTION s1
REPLICATION SET
DDL TRIGGERS
TABLE
A
(...)
PUBLICATION b
synch commit
LOG REP SLOT
SUBSCRIPTION s2
insert
delete
update
TABLE w/ PARTITIONS
B
B P1
(...)
PUBLICATION c
synch commit
LOG REP SLOT
SUBSCRIPTION s3
insert
delete
update

Basic definition

```
CREATE PUBLICATION P_main_P0 FOR TABLE main_shard0
WITH (NOPUBLISH DELETE);

CREATE SUBSCRIPTION S_main_P0
  CONNECTION 'port=7777 user=postgres dbname=master'
  PUBLICATION P_main_P0 WITH (CREATE SLOT, COPY DATA);
```

NOTE: Slot name will be the subscription name in the publisher


APP
READS
WRITES
HEAP
POSTGRES EXECUTOR
PUBLISHER
SUBSCRIBER
fsync()
WAL
LOGICAL
COMMIT ORDER
LOGICAL DECODING
output plugin
LOGICAL REPLICATION SLOT
archive
WAL SENDER PROCESS
BW
LOGICAL REPLICATION LAUNCHER
NETWORK
POSTGRES EXECUTOR
WAL
HEAP
BW
LOGICAL REPLICATION WORKER FOR SUBSCRIPTION N

Conflicts

- Any violation in constraints stops replication.
- UPDATE and DELETE operations on missing data will be skipped.
- Transaction can be omitted using pg_replication_origin_advance(subscriber_name + position). sql_skip_counter *cough*.
- Current position can be seen at pg_replication_origin_status at subscriber.

- Showcase LR conflict
- Showcase publication with many subscribers

REPLICA IDENTITY

- Which identity is used for conflict resolution:

```
    REPLICA IDENTITY { DEFAULT | USING INDEX
index_name | FULL | NOTHING }
```

Monitoring

- Publisher:

```
select * from pg_replication_slots;
```

- Subscribers:

```
postgres=# select * from pg_replication_origin;
 roident |  roname
---------+----------
       1 | pg_16394
(1 row)

postgres=# select * from pg_replication_origin_status;
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16394    | 0/0        | 0/15E0E80
```

Monitoring — cont.

- Subscribers:

```
select * from pg_stat_subscription where subname = 's_queue';"
percona
-[ RECORD 1 ]---------+-----------------------------
subid                 | 16418
subname               | s_queue
pid                   | 5293
relid                 |
received_lsn          | 0/1678E98
last_msg_send_time    | 2017-04-25 19:25:15.858439+00
last_msg_receipt_time | 2017-04-25 19:25:15.858475+00
latest_end_lsn        | 0/1678E98
latest_end_time       | 2017-04-25 19:25:15.858439+00
```

Minimun configuration

```
wal_level = logical #minimal, replica, or
logical
Max_wal_senders = 10
Wal_keep_segments # don't use it if slots
Max_replication_slots =10
#max_worker_processes = 8

# Subscribers
#max_logical_replication_workers = 4
# taken from max_worker_processes
#max_sync_workers_per_subscription = 2
# taken from max_logical_replication_workers
```

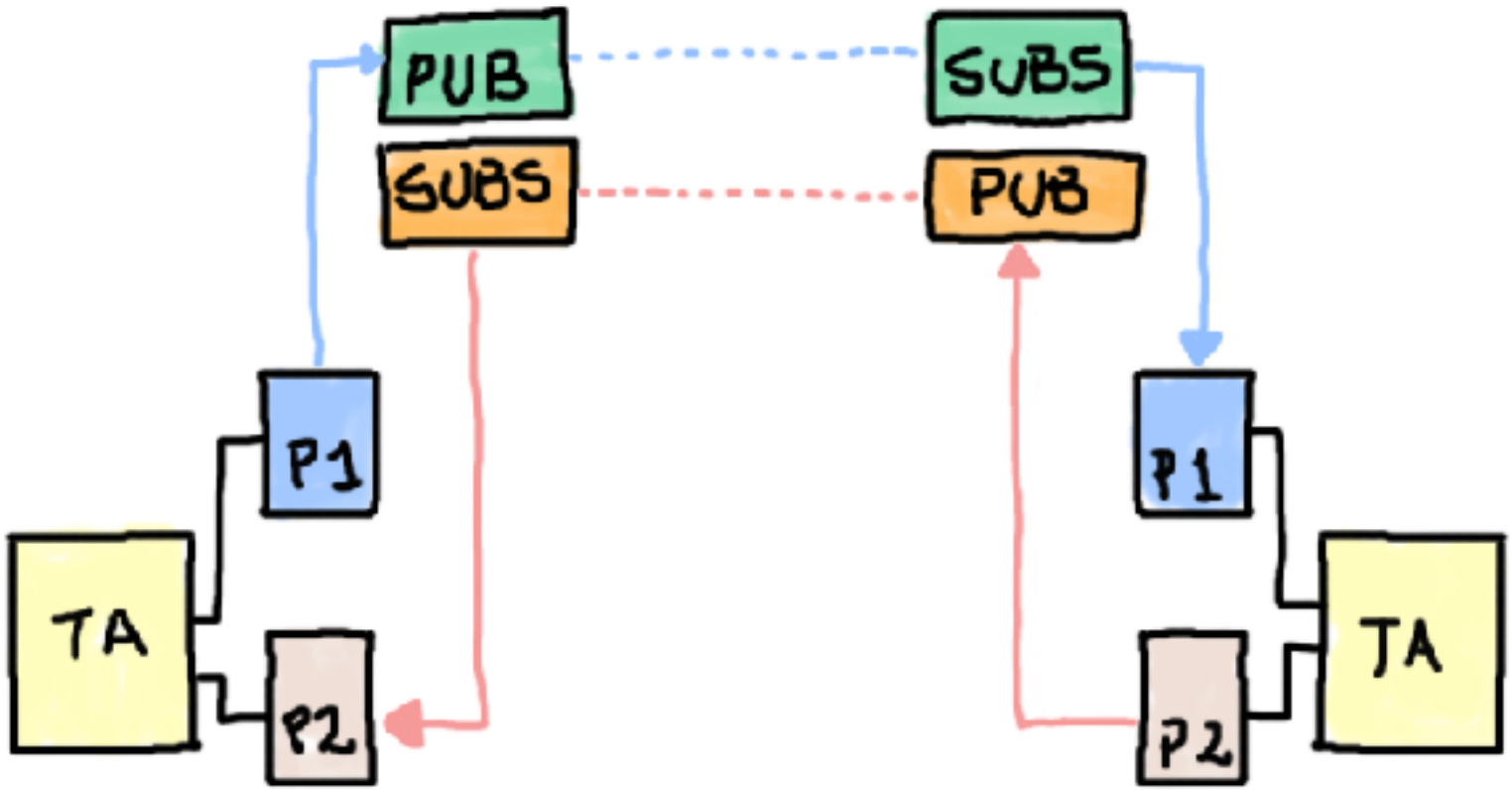
Functions related

- pg_create_logical_replication_slot
- pg_drop_replication_slot

Consuming (get) /Seeing(peek) changes (will fail with pgoutput, but this works with other logical decoding plugins, probably a bug):

- pg_logical_slot_*peek*_changes
- pg_logical_slot_*get*_changes
- pg_logical_slot_*get*_binary_changes
- pg_logical_slot_*peek*_binary_changes

Partitions and Logical Replication



pglogical

- Extension, providing similar capabilities as the future native implementation
- Additional flexibility, by allowing row filtering
- Manageable through functions
- It allows define Replication Sets
- Supports Synchronous commit
- Logical Decoding over WAL
- Stream is in commit order
- For versions over 9.4
- On subscriber it executes triggers as ENABLE REPLICA (basic transformation).

BDR

- Bi-directional replication.
- Currently is a fork, intended to be an extension on 9.6
- Allows master-master replication up to 48 nodes (or more).
- Conflict detection
- Selective replication

RDS test_decoding support

- A basic and premature implementation is on RDS by using test_decoding
- Not much documented in RDS documentation, but functional.

Reference links

- Upcoming postgres 10 features by Robert Hass
- Logical Replication and Partitioning features by me
- First insights by Robert Hass
- RDS test_decoding support

QA

Welcome to

PERCONA LIVE

www.percona.com/live