Replicación Lógica

Una introducción a esta nueva característica en Postgres 10

Emanuel Calvo

Sr. Technical Services Engineer - Remote DBA

PgDay 2017 - Santa Fe

PERCONA

Guien so'?

- 3manuek.com
- Actualmente Remote DBA en Percona. MySQL, XtraDB Cluster.
- root at ayres.io: Pipelines/Data Eng in/out Postgres, Clickhouse, RethinkDB, Elastic Search, Kafka. iMedicare, Eloquentix, YumeApp.
- Posiciones anteriores: PalominoDB, 2ndQuadrant, iMedicare, 8kData, Pythian, Globant.
- Purista de la filosofía *Bachmanity*.

El camino de la replicación en Postgres

- Streaming replication incorporada en 9.0.
- Replicación streaming en cascada en 9.2.
- Cambio de timeline agregada en 9.3.
- Logical Decoding agregada en 9.4. Mejorada en 9.6.
- Postgres 10, replicación lógica soportada de manera nativa.

Streaming / logical replication

- Streaming replication es una replicación *byte-by-byte*. Todas las bases del cluster son replicadas.
- Replicación lógica es soportada en >9.4 a través de pglogical.

Replicación comparada con MySQL

- MySQL
  - Engine log + Binlog -> byte encoded -> binlog 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

Capacidades de la nueva característica

- La RL replica objetos en base a su *replica identity (generalmente la PK).*
- El servidor destino es RW, permitiendo tener índices diferentes, definición de seguridad distinta e incluso columnas añadidas.
- Soporte *Cross-version*.
- Filtrado basado en eventos.
- Menor amplificación de escritura en red que SR.
- Una subscripción puede consumir desde varias publicaciones(multi destino).
- Una publicación puede servir para más de 1 subscripción (ALERT).

Que se puede lograr con RL?

- Flexibilidad de almacenado a través de replicación de conjunto de datos más pequeños (incluso tablas particiona).
- Topología flexible
- Sobrecarga mínima comparada con soluciones basadas en disparadores.
- Permite streaming paralelo a través de varias publicaciones.
- Migraciones y Upgrades.
- *Multi source replication* para consolidación de datos.
- Distribución de datos.
- Cadenas de replicación flexibles.
- *Data transformation*

Limitaciones

- No se puede hacer stream al mismo host (la subscripción queda bloqueada al crearse).
- Las tablas deben tener el mismo nombre (FQN) entre ambas partes.
- Las subscripciones pueden tener más columnas o un orden distinto, pero los tipos de las columnas y nombres deben ser iguales.
- Para la creación de P/S es necesario privilegios de *superuser*.
- No se replica DDL automágicamente.

Elementos

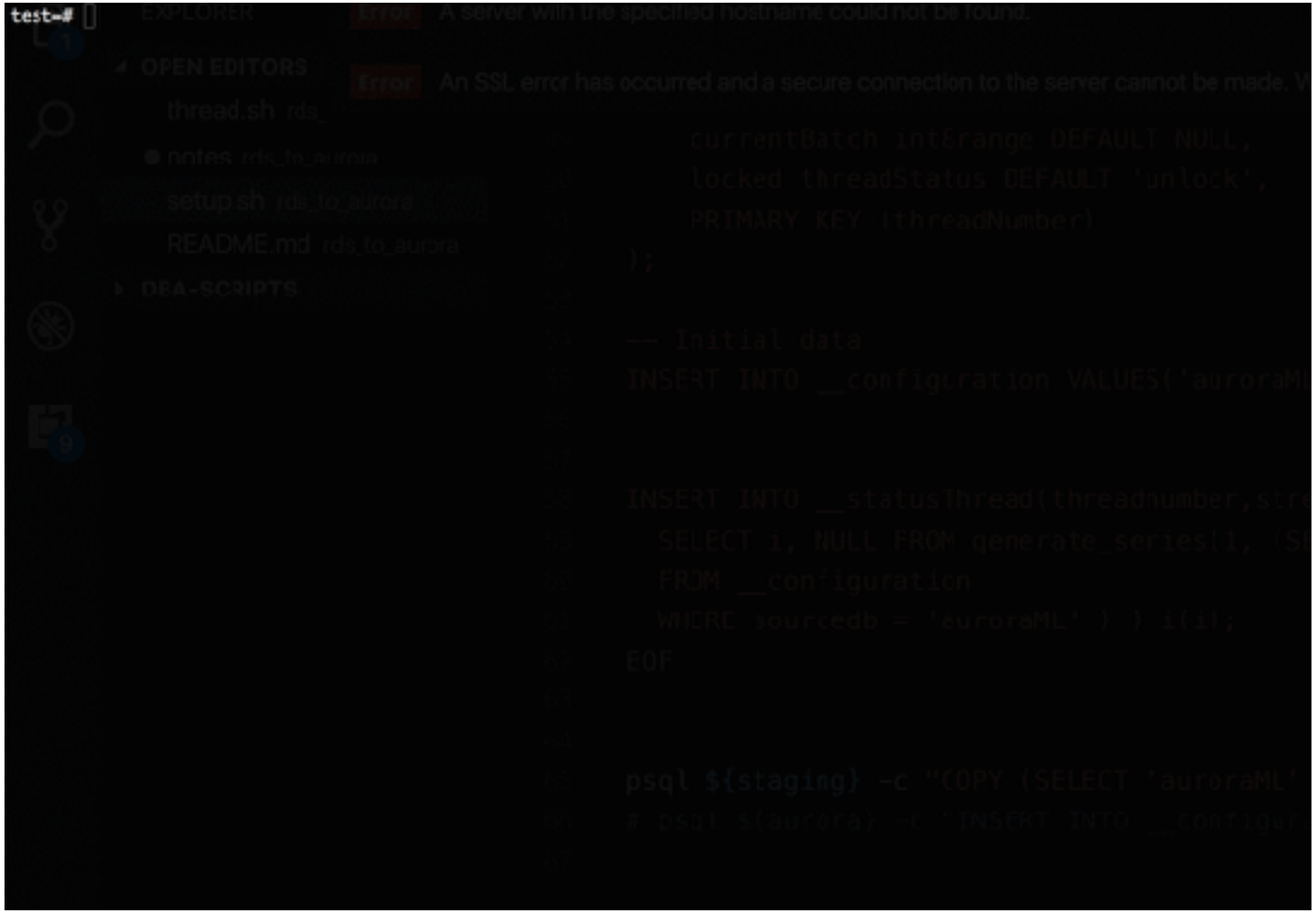
- Logical Decoding
  - Replication Slots
  - Output plugin
- Exported Snapshot
- Publication (Origin)
- Subscription (Destination)

[Logical] Replication slots

- Control de la replicación y traceo.
- Cada replica tiene un slot en el *origin* para consumir los eventos.
- Las ubicaciones son explícitamente en LSN (log sequence number).
- catalog_xmin es el numero de transacción.
- Los slots se ubican en el 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

Ejemplo de *log. replication slots*



Output Plugin

- Convierte los *WAL records* en un *custom output*.
- El nombre interno del plugin es `pgoutput`.
- Para testing de Logical Decoding, se puede usar test_decoding.

Exported snapshot

- Visibilidad compartida entre transacciones mediante la exportación del actual snapshot de la transacción. Útil para debuguear transacciones.
- Esto es usado para la copia inicial de los datos, aunque es transparente.

Publication

- Las *Publications* pueden tener más de un *subscriber*.
- Las tablas agregadas en la publicación deben estar declaradas con REPLICA IDENTITY. De otro modo las operaciones siguientes van a fallar.

publication_parameter

publish (string)

'insert, update, delete' is the default (all events).

Subscription

- Las *Subscriptions* reciben los cambios a través de un subscription worker, que a su vez utiliza un slot para saber que cambios consumir.
- Puede que más de 1 replication slot sea necesario para la copia inicial.
- La copia inicial es hecho con *pg_dump*.
- Los disparadores no son ejecutados en el destino gracias a que session_replication_role es establecido a *replica.*
- La estructura de las tablas debe ser preexistente en el destino.
- No se puede tener distintos destinos con nombres de subscripción idénticos si no se especifica un slot previo, ya que al momento de crearse los slots toman por defecto el nombre de la subscription. Esto es si ambos destinos comparten el mismo *origin*.

Subscription —cont

- Las subscripciones pueden recibir datos de varias publicaciones.
- Se pueden re sincronizar los cambios a través de REFRESH.

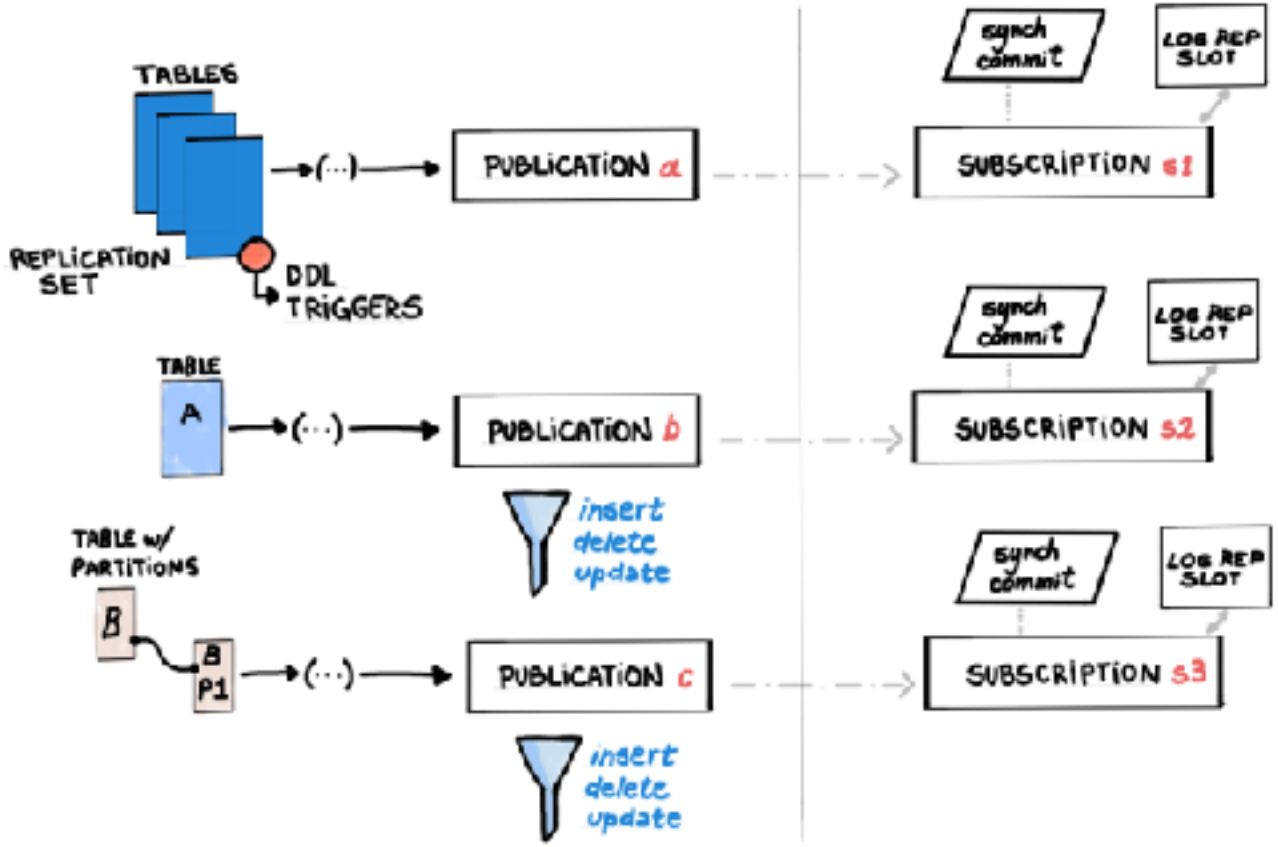
```
ALTER SUBSCRIPTION name SET PUBLICATION publication_name [, ...] { REFRESH [ WITH
( refresh_option value [, ... ] ) ] | SKIP REFRESH }
ALTER SUBSCRIPTION name REFRESH PUBLICATION [ WITH ( refresh_option value [, ... ] ) ]
```

```
refresh_option
  copy_data (boolean)
```

subscription_parameter

copy_data
create_slot
enabled
slot_name
synchronous_commit
connect (afecta copy_data, create_slot y enabled)

Ejemplos de configuración de *Publication*



PERCONA

Definición básica

```
CREATE PUBLICATION P_main_P0 FOR TABLE main_shard0 WITH
(publish = 'insert, update');  -- no delete

CREATE SUBSCRIPTION S_main_P0
  CONNECTION 'port=7777 user=postgres dbname=master'
  PUBLICATION P_main_P0 WITH (create_slot =true, copy_data
=true);
```

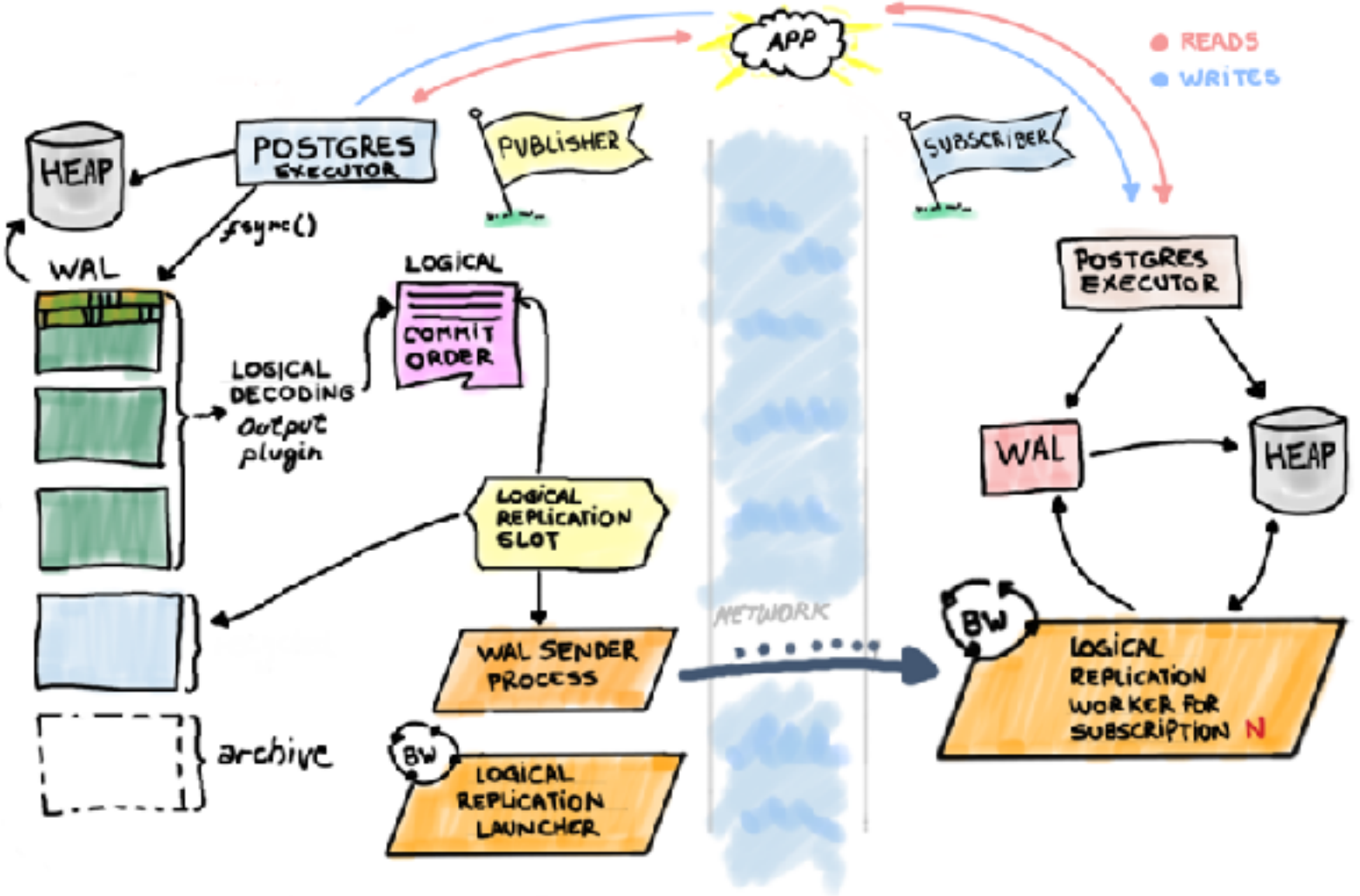
NOTE: Slot name will be the subscription name in the publisher

Agregando publicaciones en subscripciones

```
CREATE PUBLICATION P_queue_test FOR TABLE queue WITH (publish =
'insert, update,delete');
CREATE PUBLICATION P_queue2_test FOR TABLE queue2 WITH (publish =
'insert, update,delete');
CREATE SUBSCRIPTION S_queue_test
  CONNECTION 'port=8888 user=postgres dbname=percona'
  PUBLICATION P_queue_test WITH (create_slot =true, copy_data =true);

ALTER SUBSCRIPTION S_queue_test SET PUBLICATION P_queue_test,
P_queue2_test REFRESH WITH (copy_data = true);
ALTER SUBSCRIPTION S_queue_test REFRESH PUBLICATION WITH (copy_data =
true);
```

Flow



PERCONA

Conflictos

- Cualquier violación de consistencia para la replicación (constraints)
- Las operaciones de UPDATE y DELETE en datos ausentes serán omitidos y no generan error.
- La transacción puede ser saltada usando *pg_replication_origin_advance(subscriber_name + position)*.
  - aka sql_skip_counter en MySQL **cough**.

- La actual posición puede ser vista en pg_replication_origin_status en el subscriber.

Replica Identity

- Que identidad será usada para corroboración de consistencia:

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

Monitoring

- Publisher:

```
select * from pg_replication_slots;
```

- Subscribers:

```
percona=# select pg_replication_origin_progress(r.r, true) FROM (select roname from
pg_replication_origin where roident = 1) r(r);
 pg_replication_origin_progress  | 0/17024E0

postgres=# select * from pg_replication_origin;
 roident |  roname
---------+----------
       1 | pg_16394

percona=# select * from pg_replication_origin_status ;
 local_id | external_id | remote_lsn | local_lsn
----------+-------------+------------+-----------
        1 | pg_16503    | 0/17024E0  | 0/16DEE30
```

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
```

Minimum 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
```

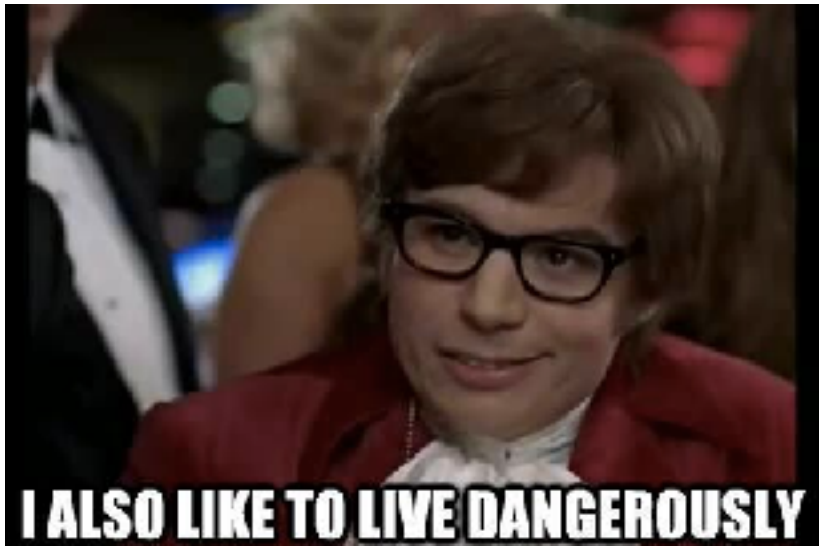
Funciones relacionadas

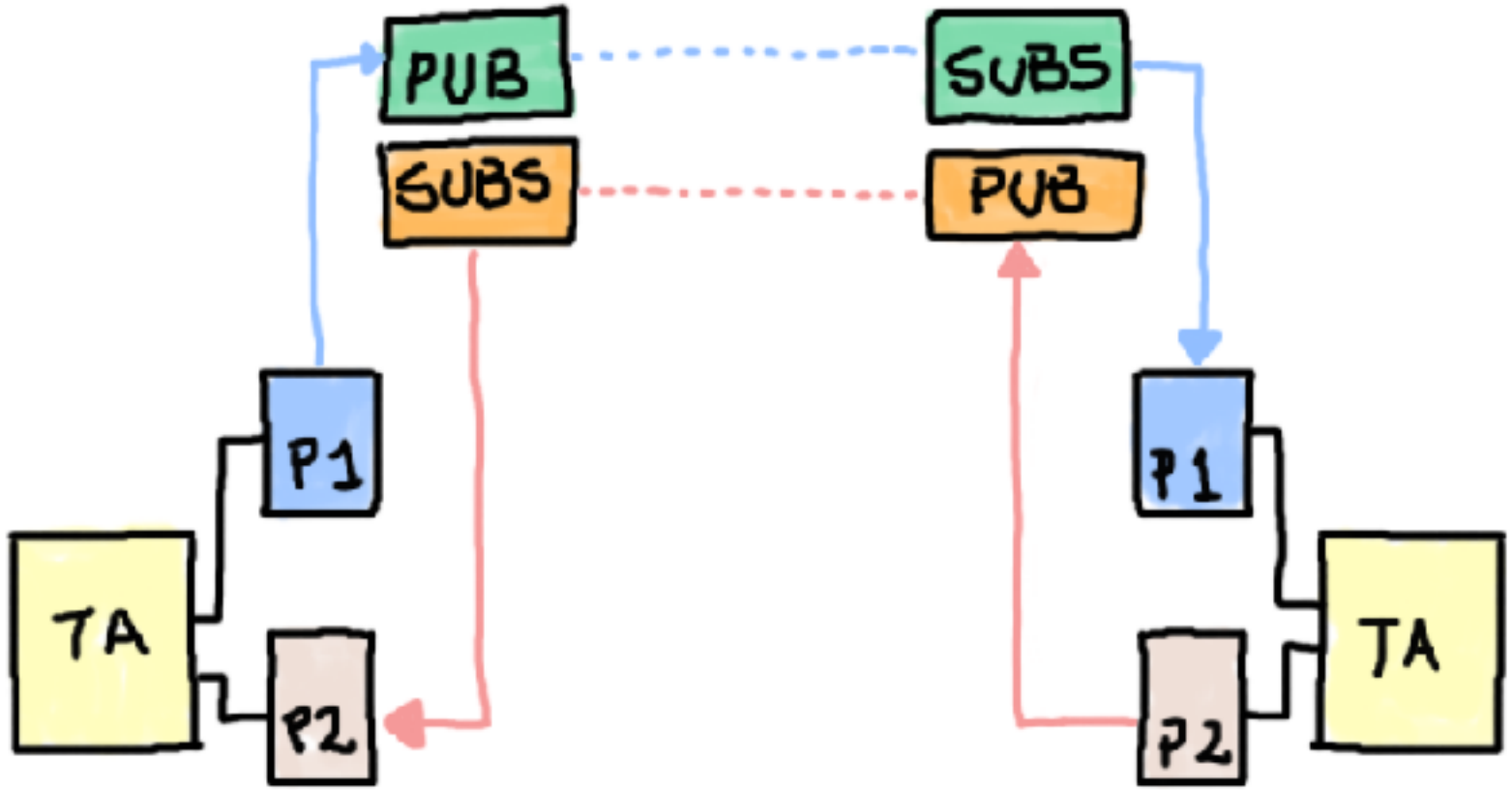
- pg_create_logical_replication_slot
- pg_drop_replication_slot

Consuming (get) /Seeing(peek) (falla con *pgoutput*, pero funciona con los plugins de logical decoding):

- 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





PERCONA

pglogical

- Extensión, provee características similares
- Flexibilidad adicional a través de filtros por valores de columnas.
- Administrable a través de funciones.
- Permite definir conjuntos de réplica.
- Soporta Synchronous commit
- Logical Decoding sobre WAL
- EI Stream en en orden de commit
- >9.4
- En el subscriber ejecuta disparadores como ENABLE REPLICA (basic transformation).

BDR

- Replicación asíncrona bi-direcccional.
- Actualmente es un fork, posiblemente una extensión para 9.6
- Permite replicación master-master hasta 48 nodos (o más).
- Detección de conflictos
- Replicación selectiva

RDS test_decoding support

- RDS soporta replicación lógica a través de test_decoding
- No posee mucha documentación, pero es funcional.

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

Showcase

Como se ve trabajar con LR.

Cases

- Showcase LR conflict
- Showcase publication with many subscribers
- Bug?

Database Performance Matters