Como funciona a replicação lógica no POSTGRESQL

João Vitor Foltran - DBA BRASIL - 06.2023

SOBRE MIM

25 anos
Blumenau, SC
Especialista PostgreSQL @ TOTVS

linkedin.com/in/joao-foltran-031b9312b

joao@foltrandba.com

- EnterpriseDB, iFood, GoldenGateBR
- Maluco por card games (FAB, MTG)
- Ex Jogador de Handball
- Amo automação

Diamante

Platina

Ouro

Prata

Apoio

O que é?

Também conhecida como *replicação transacional*, é um método de replicação de objetos de dados e suas mudanças, baseando-se na sua identidade de replicação (geralmente primary key).

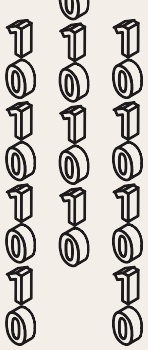
Funciona construindo um stream de modificações de dados lógicos entre clusters de PostgreSQL.

- **Implementado na versão 10**
- **Surgiu para atender limitações na replicação física**
  (selective replication, cross major versions, active standby, cross platform)

Usos

- Upgrades
- Migrações
- Consolidação de dados
- Compartilhar subsets de dados entre diversos clusters
- Replicação entre plataformas e major versions

Como funciona



1. Trabalha utilizando um método de *publisher* e *subscriber*.

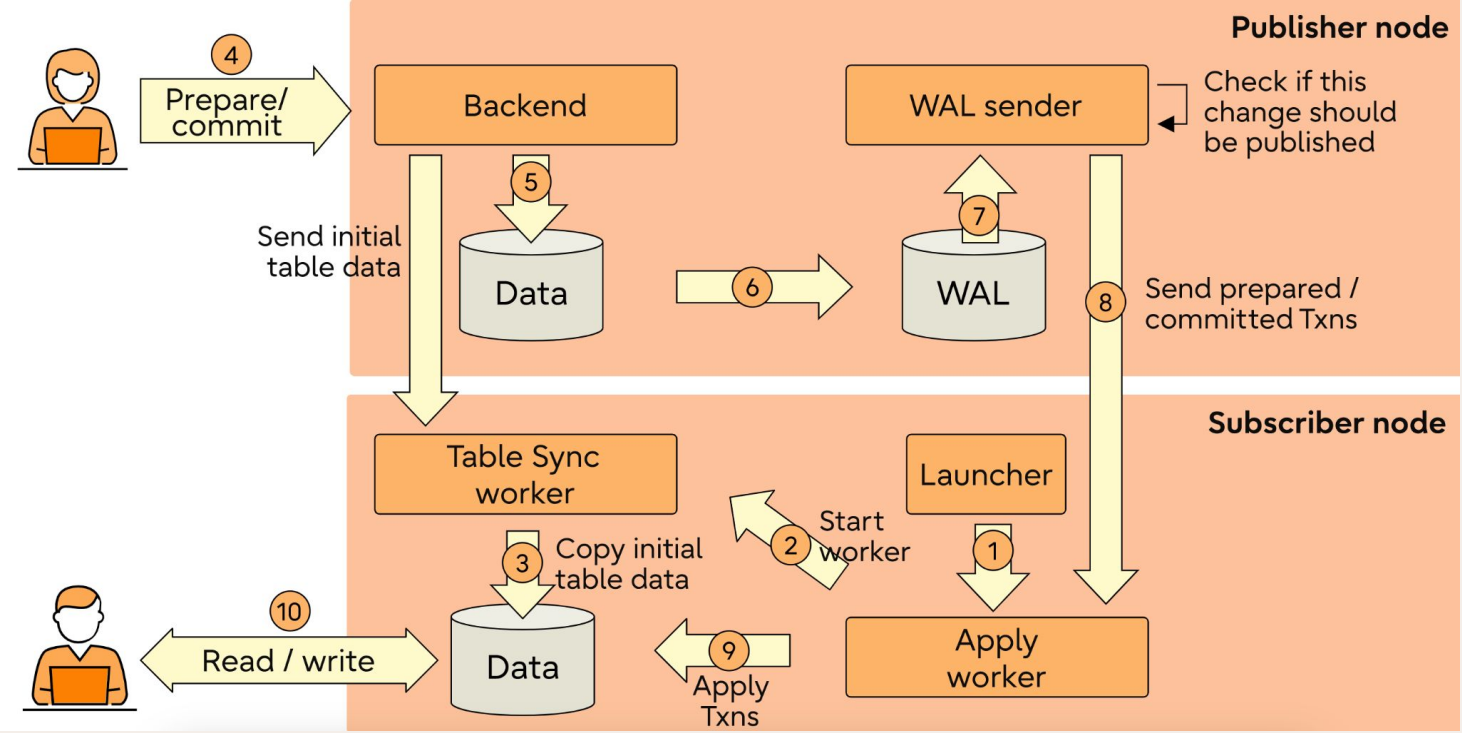
2. No node denominado como publisher, cria-se uma *publication*, que é um set de mudanças de uma tabela ou grupo de tabelas.

3. Já no node denominado de subscriber, cria-se uma *subscription*, que pode se subscrever a uma ou mais *publication*.

4. A replicação inicia copiando um snapshot das tabelas envolvidas na publication para seus subscribers.

5. Após sincronização inicial, as mudanças subsequentes no publisher são enviadas aos subscribers em real-time.

Arquitetura



https://www.postgresql.fastware.com/blog/inside-logical-replication-in-postgresql

Publication

Uma *publication* é uma combinação de mudanças originadas de uma tabela ou um grupo de tabelas, que existe dentro de um database. Também pode ser chamada de replication set ou change set.

- Possível limitar diversas combinações de mudanças (DML), similar a triggers
- Operações UPDATE/DELETE necessitam de uma *replica identity* (Primary Key, Unique Key)
- Possível adicionar e remover tabelas dinamicamente
- Não copia a estrutura das tabelas e operações DDL

```
CREATE PUBLICATION active_departments FOR TABLE departments WHERE (active IS TRUE);
```

```
CREATE PUBLICATION users_filtered FOR TABLE users (user_id, firstname);
```

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

https://www.postgresql.org/docs/current/logical-replication-publication.html

Subscription

Uma *subscription* define a conexão para a outra database e a publication ou set de publications que estará se subscrevendo.

- Possível se conectar a várias publications
- Cada subscription recebe as mudanças através de um replication slot
- Possível pausar e continuar a qualquer momento
- Tabelas com nomes diferentes do replication set não são suportadas

```
CREATE SUBSCRIPTION sub1
CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb'
PUBLICATION mypublication, insert_only;
```

```
CREATE SUBSCRIPTION sub1
CONNECTION 'host=192.168.1.50 port=5432 user=foo dbname=foodb'
PUBLICATION mypublication, insert_only WITH (disable_on_error = true);
```

https://www.postgresql.org/docs/current/logical-replication-subscription.html

Replication Slot

É um objeto importante para a replicação lógica. Cada subscriber recebe um slot de replicação no lado do publisher.

- Guarda em qual posição o subscriber está no apply de WALs
- Garante que WALs ainda necessários para o subscriber conectado não sejam apagados

Parâmetros

```
wal_level = logical
```

```
max_logical_replication_workers = subscribers + reserva
```

```
max_replication_slots = subscribers + reserva
```

```
max_wal_senders = max_replication_slots + replicas fisicas + reserva
```

```
max_worker_processes = default + max_logical_replication_workers + 1
```

https://www.postgresql.org/docs/current/logical-replication-config.html

Filtros

- Filtros de linha são aplicados antes da publicação das mudanças
- Possível aplicar filtros específicos para cada tabela em uma publication
- Atualmente somente expressões WHERE básicas são aceitas (sem funções)
- Para replicações UPDATE/DELETE, a cláusula WHERE precisa conter somente as colunas da replica identity

Updates

- Filtros são avaliados para a linha nova e antiga

Old row	New row	Transformation
no match	no match	don't replicate
no match	match	`INSERT`
match	no match	`DELETE`
match	match	`UPDATE`

https://www.postgresql.org/docs/current/logical-replication-row-filter.html

Conflitos

- Funciona como operações DML
- As operações são realizadas com privilégios do usuário da replicação
- Conflitos interrompem a replicação

```
ERROR:  duplicate key value violates unique constraint "test_pkey"
DETAIL:  Key (c)=(1) already exists.
CONTEXT:  processing remote data for replication origin "pg_16395" during "INSERT" for
replication target relation "public.test" in transaction 725 finished at 0/14C0378
```

- Possível pular transações

```
ALTER SUBSCRIPTION sub_alltables SKIP (lsn = '0/14C0378');
```

```
pg_replication_origin_advance()
```

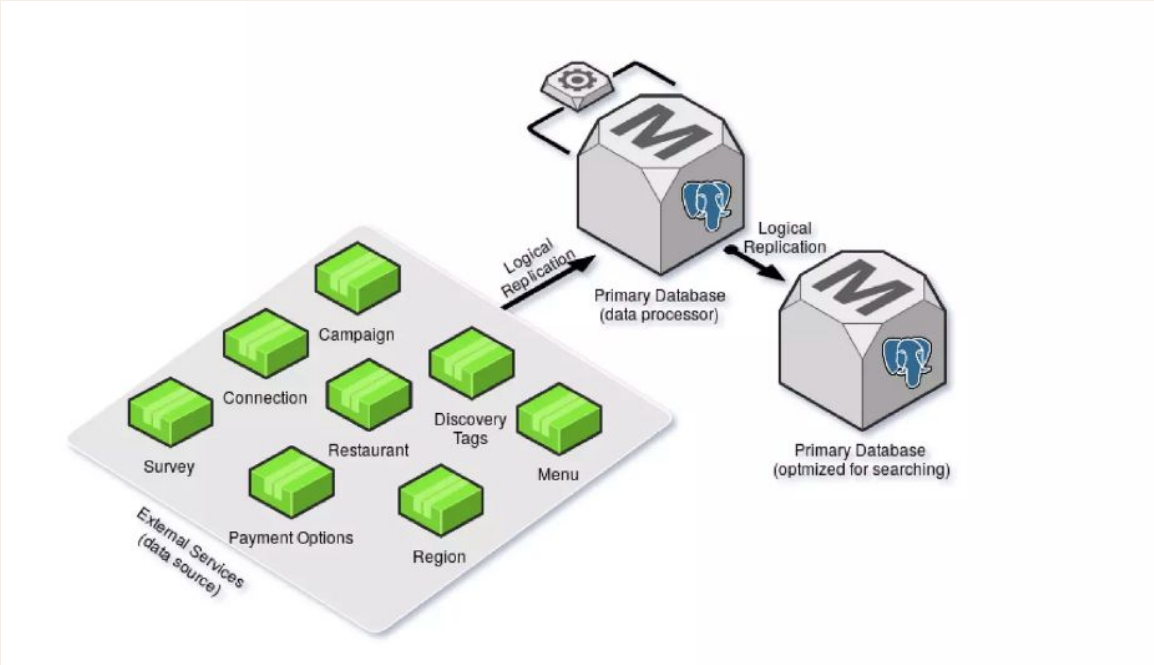
https://www.postgresql.org/docs/current/logical-replication-conflicts.html

Restrições

- O schema e comandos DDL não são replicados, tabelas no subscriber precisam existir antes da configuração
- Dados de sequences não são replicados, podendo afetar setups com intuito de realizar failover
- TRUNCATE funciona, porém é necessário cuidado em situações que envolvem FKs para tabelas que não estão no replication set
- LOBs não são replicados, sem possibilidade de workaround
- Somente tabelas podem ser replicadas. Views, MViews, Foreign Tables não são suportadas

https://www.postgresql.org/docs/current/logical-replication-restrictions.html

Exemplo iFood



Links úteis

- https://www.enterprisedb.com/postgres-tutorials/logical-replication-postgresql-explained
- https://www.postgresql.org/docs/current/logical-replication.html
- https://www.postgresql.fastware.com/blog/inside-logical-replication-in-postgresql
- https://www.crunchydata.com/blog/data-to-go-postgres-logical-replication

OBRIGADO