Fluidifier ses transferts de données avec PostgreSQL

PostgreSQL User Group Genève

Cédric Villemain cedric@2ndQuadrant.com

11 Octobre 2018

Cédric Villemain

2ndQuadrant®
PostgreSQL

Expertise & Développement PostgreSQL
Formations
Support 24x7 & DBA à Distance

Sponsor Platine de PostgreSQL

- **9.1**, Synchronous replication
- **9.2**, Refactoring checkpoint group commit
- **9.3**, Event Triggers
- **9.4**, Slot de Replication
- **9.5**, Block Range INdex
- **9.6**, Amélioration performances datawarehouse
- **10**, CREATE STATISTICS, Réplication Logique
- **11**, Procedures, Partitionning

Au programme

COPY

Foreign Data Wrapper

Channels

Décodage Logique

Type de données

Protocole PostgreSQL

COPY

```
CREATE TABLE opendata (data jsonb);
COPY opendata
FROM  PROGRAM 'curl https://opendata/json'
FREEZE;
```

COPY

```
COPY (SELECT aid, bid, abalance from pgbench_accounts)
TO  '/export/pgbench_accounts.csv'
WITH (FORMAT 'csv');
```

FDW

```
CREATE EXTENSION file_fdw;
CREATE SERVER pglog FOREIGN DATA WRAPPER file_fdw;
```

FDW

```
CREATE FOREIGN TABLE pglog (
  log_time timestamp(3) with time zone,
  user_name text,
  database_name text,
  ....,
  location text,
  application_name text
) SERVER pglog
  OPTIONS ( FILENAME '/pgdata/10/main/log/pglog.csv'
          , FORMAT 'csv' );
```

LISTEN

```
LISTEN data;
```

NOTIFY

```
select pg_notify('data',
         (select jsonb_pretty(data)
           from opendata limit 1)::text);
```

Possible via trigger !

LISTEN

```
Notification asynchrone « data » reçue avec le contenu « {
    "code": "57da8f4c4d1c9",
    "tags": [
    ],
    "files": {
....
```

Init

```
SELECT *
FROM pg_create_logical_replication_slot(
        'demo', 'test_decoding');
```

Output

```
 SELECT * FROM pg_logical_slot_get_changes('demo', NULL, NULL)
 location   | xid     | data
------------+---------+------
 8/FD1AA7A8 | 8937361 | BEGIN 8937361
 8/FD1AA7A8 | 8937361 | table public.opendata: INSERT:
   data[jsonb]:'{"code": "57da8f4c4d1c9",
...
```

pglogical

Un plugin et un consommateur de données avancés
pour optimiser la réplication depuis PostgreSQL
vers PostgreSQL, json, Kafka, etc.

nombreuses fonctionnalités dont:
aggrégation de données, filtrage, modifications....

XML

```
SELECT xmlpi(name php, 'echo "hello world";');
            xmlpi
------------------------------
 <?php echo "hello world";?>
```

hstore

```
SELECT * FROM each('aaa=>bq, b=>NULL, ""=>1');
 key | value
-----+-------
     | 1
 b   |
 aaa | bq
```

hstore

```
SELECT 'aaa=>bq, b=>NULL, ""=>1'::hstore::jsonb;
                jsonb
-----------------------------------
 {"": "1", "b": null, "aaa": "bq"}
```

bytea ....

```
select armor(data::text::bytea) from opendata;
                          armor
-----------------------------------------------------
 -----BEGIN PGP MESSAGE-----

 eyxYzkiLCAidGFncyI6IFtdLCAiZmlsZXMiOiB7ImNzdiI6ICJo
...
```

WAL

intégrité des données

```
create unlogged table websession
   (cookie text, data bytea);
```

FrontEnd / BackEnd

```
copy opendata to stdout with (format 'binary');
PGCOPY
?
?{"code": "57da8f4c4d1c9
...
: []}??
```

Pas de conversion au format 'text', et plus compliqué à gérer....
Mais potentiellement plus performant

Des questions ?

C'est à vous !