

# Connaissez-vous ces extensions ?

PGSession 14

Florent Jardin

(17 novembre 2021)

# PRENONS DEUX DÉVELOPPEURS

*Tu connais PostgreSQL ? Tu sais, le système de bases de données relationnelles le plus avancé au monde...*

*Dis comme ça, c'est prometteur. Okay, on part là-dessus !*

(Plusieurs versions du modèle de données  
et quelques procédures stockées plus tard...)

# Qualité

*Dis, c'est possible de faire des tests automatisés sur ton truc ?*

# PGTAP

- Stable depuis Février 2019 (v1.0.0) après 10 ans de chantier
- Test Anything Protocol (TAP) pour PostgreSQL écrit en PL/pgSQL

```
SELECT has_type('month_day');  
SELECT col_type_is('month_day', 'month', 'integer');  
SELECT col_type_is('month_day', 'day', 'integer');
```

```
~ pg_prove pgtap.sql  
pgtap.sql .. ok  
All tests successful.  
Files=1, Tests=34,  0 wallclock secs (0.03 CPU)  
Result: PASS
```

# PLPGSQL\_CHECK

- Extension maintenue depuis 2008
- Couteau suisse du développeur PL/pgSQL

```
SET search_path = "$user", plpgsqlcheck;  
SELECT functionid, lineno, message  
FROM plpgsql_check_function_tb('f1()');
```

```
-[ RECORD 1 ]-----  
functionid | f1  
lineno     | 6  
message    | record "r" has no field "c"
```

# Gestionnaire de tâches

*Ah mais, il n'y a pas d'orchestrateur interne dans PostgreSQL ? Tu sais, avec Oracle, il existe `DBMS_JOBS` ou `DBMS_SCHEDULER` ...*

# PG\_CRON

- Extension maintenue par Citus Data depuis 2016
- Syntaxe entièrement compatible avec `cron`
- Ne se déclenche que sur les instances primaires
- Disponible sur la plupart des fournisseurs Cloud

```
SELECT cron.schedule_in_database(  
  job_name => 'Purge events table',  
  database => 'app',  
  schedule => '30 3 * * 6',  
  command  => $$DELETE FROM events  
            WHERE event_time < now() - interval '1 week'$$  
);
```

# PG\_DBMS\_JOB

- Extension sortie en août 2021, proposée par MigOps Inc.
- Émule le composant `DBMS_JOB` d'Oracle
- Un daemon externe écrit en Perl
  - écoute les notifications
  - surveille les travaux à lancer

```
SELECT dbms_job.submit(  
  what          => 'ANALYZE',  
  next_date     => date_trunc('day', now()) + '1d'::interval,  
  job_interval  => $$date_trunc('day', now()) + '1d'::interval$$  
);
```

# Données sensibles

*Bon, on doit se conformer aux réglementations en vigueur. C'est possible de chiffrer les données facilement ?*

# PGCRYPTO

- Fonctions de hachage ( `md5` , `sha` , `hmac` , `xdes` , ... )
- Fonctions de chiffrement ( `pgp` , `bf` , `aes` )
- Fonctions de données aléatoires

```
SELECT lastname, pgp_pub_decrypt(creditcard, dearmor(  
  '-----BEGIN PGP PRIVATE KEY BLOCK-----  
  // ... //  
  -----END PGP PRIVATE KEY BLOCK-----'  
));
```

# POSTGRESQL\_ANONYMIZER

- Extension en bêta (0.9.0), proposée par Damien Clochard (Dalibo)
- Approche déclarative des règles d'anonymisation
  - Masquage statique (substitution permanente)
  - Masquage dynamique pour les rôles *MASKED*
  - Export anonymisé avec `pg_dump_anon`

```
SECURITY LABEL FOR anon ON ROLE pierre IS 'MASKED';  
SECURITY LABEL FOR anon ON COLUMN people.phone  
IS 'MASKED WITH FUNCTION anon.partial(phone,2,$$*****$$,2)';
```

# Performances

*Je comprends pas. Sur mon poste, c'est super rapide... Tu crois qu'il manque un index ?*

# PG\_STAT\_STATEMENTS

- Extension incluse avec PostgreSQL depuis la version 8.4
- Collecte de statistiques sur les planifications et exécutions de requêtes

```
—[ RECORD 1 ]—  
query          | UPDATE pgbench_accounts  
                |     SET abalance = abalance + $1  
                |     WHERE aid = $2  
calls          | 3000  
total_exec_time | 271.232977  
rows           | 3000  
hit_percent    | 98.85
```

# PG\_QUALSTATS

- Extension maintenue par l'équipe PoWA depuis 2016
- Collecte de statistiques sur les prédicats de recherches et de jointures
- Diagnostic des index manquants ou non optimaux

```
table | column | calls | exec_count | nbfiltered
-----+-----+-----+-----+-----
pgbench_accounts | aid | 1 | 100000 | 99999
```

```
-[ RECORD 1 ]-----  
indexes | "CREATE INDEX ON pgbench_accounts USING btree (aid)"
```

# HYPOPG

- Extension stable depuis 2016
- Émule la présence d'un index lors d'un `EXPLAIN`

```
SELECT hypopg_create_index('CREATE INDEX ON hypo (id)');  
EXPLAIN SELECT val FROM hypo WHERE id = 1;
```

## QUERY PLAN

---

```
Index Scan using <18284>btree_hypo_id on hypo  
(cost=0.04..8.06 rows=1 width=10)  
Index Cond: (id = 1)
```

# (Très) fortes volumétries

*Bon les gars, on augmente le nombre de serveurs pour absorber la charge, et il faut aussi scaler les bases de données PostgreSQL.*

# PG\_PARTMAN

- Extension maintenue par Crunchy Data depuis 2018
- Création automatique des partitions de type `range`
- De nombreux scripts et fonctions de maintenance

```
SELECT partman.create_parent (  
  p_parent_table => 'public.events',  
  p_control => 'at',  
  p_type => 'native',  
  p_interval => 'daily',  
  p_template_table => 'public.events_template'  
);
```

# FOREIGN DATA WRAPPERS (FDW)

- Famille d'extensions répondant à la norme SQL/MED
- Consulter les données d'une table hébergée sur un autre système
- `postgres_fdw` et `file_fdw` fournies avec PostgreSQL
- Compatibles avec le partitionnement

```
CREATE FOREIGN TABLE population_fridf
PARTITION OF population FOR VALUES IN ('FR-IDF')
SERVER server_fridf OPTIONS (table_name 'population');
```

# CITUS

- Extension proposée par Citus Data depuis 2016
- Distribution horizontale des données et des requêtes
- Quelques limitations SQL et de nombreuses fonctionnalités

```
SELECT citus_add_node('worker-101', 5432);  
SELECT citus_add_node('worker-102', 5432);  
SELECT create_distributed_table('companies', 'id');
```

# Questions / Réponses