PGSession 14

Connaissez-vous ces extensions ?

17 novembre 2021

Florent Jardin

Connaissez-vous ces extensions ?

PGSession 14

TITRE : Connaissez-vous ces extensions ?
SOUS-TITRE : PGSession 14

DATE: 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…)

Connaissez-vous ces extensions ?

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
```

Auteur : David E. Wheeler (pgenv) Projet : https://github.com/theory/pgtap

- stable après plus de 10 ans de chantier
- tests au format SQL dans un script ou dans un schéma dédié

Article https://fljd.in/2020/05/14/ecrire-ses-tests-unitaires-en-sql

---

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"
```

Auteur : Pavel Stehule (orafce) Projet : https://github.com/okbob/plpgsql_check

- Version 2.0.0 sortie en sept. 2021
- Successeur de `plpgsql_lint` (2008 à 2013)
- Nombreuses fonctionnalités
  - qualité du code
  - dépendances
  - profiler

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  => $$DELETE FROM events
     WHERE event_time < now() - interval '1 week'$$
);
```

Auteur : Marco Slot (Citus Data) Projet : https://github.com/citusdata/pg_cron

- Paramètres `shared_preload_libraries` et `cron.database_name`
- Un *background worker* se réveille pour lire la table des travaux
- pg_cron ouvre une nouvelle connexion locale pour chaque exécution

---

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 => $$date_trunc('day', now()) + '1d'::interval$$
);
```

Auteur : Gilles Darold (MigOps Inc.) Projet : https://github.com/MigOpsRepos/pg_dbms_job

- Déclenchement asynchrone à l'aide de `pg_notify`
- Surveillance des travaux asynchrone toutes les 100 ms
- Déclenchement d'un travail planifié toutes les 5 minutes

Article https://www.migops.com/blog/2021/08/27/announcing-pg_dbms_job-in-postgresql-for-oracle-dbms_job-compatibility/

Connaissez-vous ces extensions ?

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-----'
));
```

Documentation : https://www.postgresql.org/docs/current/pgcrypto.html

- Extension inclue avec PostgreSQL depuis la version 8.3
- Jusqu'en version 13, l'extension fournissait gen_random_uuid() version 4. La méthode est inclue dans PostgreSQL depuis la version 13.
- Privilégier la méthode `crypt` plutôt que `md5` pour bénéficier d'un salage
  - utile pour le stockage des mots de passe
  - pas de besoin de déchiffrement
- Privilégier le chiffrement PGP plutôt que les fonctions de chiffrement brut
  - pas de vérification d'intégrité de la donnée chiffrée
  - le type `text` n'est pas supporté
  - chiffrement symétrique (une clé) et asymétrique (deux clés)

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)';
```

Auteur : Damien Clochard (Dalibo) Projet : https://gitlab.com/dalibo/postgresql_anonymizer
Documentation : https://postgresql-anonymizer.readthedocs.io/en/latest/

- Paramètre `session_preload_libraries`
- S'appuie sur les *Security labels* et des vues intermédiaires dans un schéma `mask`
- Statique
  - Fonctions `anonymize_database()` et `anonymize_table()`
- Dynamique
  - Fonctions `start_dynamic_masking()` et `stop_dynamic_masking()`
  - Limiter à un seul schéma dans une base
  - Performance dégradé sur les jointures de données masquées
- Export anonymisé
  - Wrapper écrit en Go qui repose sur `psql` et `pg_dump`
- Un atelier complet pour se familiariser à l'outil https://dalibo.gitlab.io/postgresql_anonymizer/how-to.handout.pdf

Connaissez-vous ces extensions ?

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
```

Documentation : https://www.postgresql.org/docs/current/pgstatstatements.html

- Paramètres `shared_preload_libraries`, `pg_stat_statements.max` (5000)
- Deux vues `pg_stat_statements` et `pg_stat_statements_info`
- `query_id` (normalisation des requêtes) globalement disponible depuis la version 14 de PG avec `compute_query_id`

Article : https://yhuelf.github.io/2021/09/30/pg_stat_statements_bottleneck.html

---

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)"
```

Projet : https://github.com/powa-team/pg_qualstats

- Analyser les prédicats (*quals*) les plus fréquemment exécutés
- Identifier les colonnes corrélées

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)
```

Auteur : Julien Rouhaud Projet : https://github.com/HypoPG/hypopg Documentation : https://hypopg.readthedocs.io

- Index hypothétique stockée dans l'espace mémoire de la session
  - valable uniquement dans la même session
  - aucune incidence pour les autres sessions concurrentes

Connaissez-vous ces extensions ?

(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'
);
```

Auteur : Keith Fiske (Crunchy Data) Projet : https://github.com/pgpartman/pg_partman

- Paramètre `shared_preload_libraries`
- Configuration d'un *background worker*
- Plusieurs fonctions de maintenance
  - `run_maintenance()` et `run_maintenance_proc()` (optimisée sur PG11+)
  - `partition_gap_fill()` pour compléter les trous de partitions
  - `dump_partitioned_table_definition()`
  - `partition_data_proc()` pour transformer une table en partitions
- Plusieurs scripts de maintenance
  - `vacuum_maintenance.py`
  - `dump_partition.py`

> As of version 4.1.0, pg_partman no longer requires a superuser to run for native partitioning.

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');
```

- SQL/MED = *Management of External Data*
- Architecture en constante amélioration depuis PostgreSQL 8.4
- Nouveau nœud `Async Foreign Scan` avec PostgreSQL 14

Wiki https://wiki.postgresql.org/wiki/Foreign_data_wrappers Article https://fljd.in/2021/07/16/parlons-un-peu-des-donnees-externes/

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');
```

Projet : https://github.com/citusdata/citus

- CitusDB devient une extension en 2016 avec la sortie de la version 5.0
- Paramètre `shared_preload_libraries`
- Privilégier les fonctions pour les traitements transactionnels
- Certains types de requêtes ne sont pas compatibles
  - Recursive CTEs
  - SELECT … FOR UPDATE
  - Grouping sets
- Base de données managée (Hyperscale) de Microsoft Azure https://docs.microsoft.com/en-us/azure/postgresql/hyperscale-overview
- Citus 10 sorti en février 2021

Connaissez-vous ces extensions ?

- Méthode d'accès `columnar` avec PostgreSQL 12 et supérieures
  - Pas de support `UPDATE` et `DELETE`
  - Pas de support de la réplication logique
  - Pas de TOAST
- *Shard rebalancer* disponible en version communautaire
  - verrou d'écriture durant l'opération > For non-blocking reads and writes during rebalancing, try Citus > Enterprise edition.

QUESTIONS / RÉPONSES