PostgreSQL

Nouveautés de la version 11

< Gilles Darold @ Dalibo >

PgSession 10 - 2018

DALIBO
L'expertise PostgreSQL

PostgreSQL v11

L’agenda

- V11.0 sortie le 18 Octobre 2018
- V11.1 sortie le 08 Novembre 2018
- Prochaine version mineure prévue le 14 Février 2019

- V12.0 en développement depuis le 30 juin 2018
- Sortie prévue à l’automne 2019

Nouvelles fonctionnalités

- Partitionnement
- Performances
- WAL & Réplication
- SQL & PL/PgSQL

Partitionnement HASH

- Partitionnement déclaratif PostgreSQL v10 :
  - types de partitionnement :
    - liste (LIST)
    - intervalle (RANGE)
- Nouveau type de partitionnement par hachage: HASH
  - Permet de répartir les données équitablement sur les partitions sans classement particulier

Partitionnement HASH

```
CREATE TABLE kbln_info (
    id integer NOT NULL,
    blank_series varchar(50) NOT NULL
)
PARTITION BY RANGE (id);

CREATE TABLE kbln_p0 OF kbln
FOR VALUES FROM (MINVALUE) TO (500000)
PARTITION BY HASH (blank_series);

CREATE TABLE kbln_p0_1 OF kbln_p0
FOR VALUES WITH (MODULUS 2, REMAINDER 0);

CREATE TABLE kbln_p0_2 OF kbln_p0
FOR VALUES WITH (MODULUS 2, REMAINDER 1);
```

Partitionnement par défaut

Déplacement dans une table par défaut

lorsque la partition n’existe pas

```
CREATE TABLE catch_other_rows PARTITION OF main_table DEFAULT;
```

Update clé de partition

Mise à jour de la clé de partitionnement

- change la valeur dans la clé de partitionnement
- déplace les données dans la partition cible
- dans la partition par défaut si nécessaire

Index / contrainte sur table partitionnée

- Création/suppression d'index ou contrainte sur la table partitionnée :
  - propagé sur chaque partition
  - vrai aussi pour les clés primaires et les clés étrangères
  - pas pour les index uniques sans la clé de partition
  - les nouvelles partitions les reproduisent aussi

- Clé étrangère depuis une table partitionnée
  - mais pas vers une table partitionnée

INSERT … ON CONFLICT

S’applique maintenant aux tables partitionnées

Élagage de partition

- Le *partition pruning*
  - contrôlé par la directive « enable_partition_pruning = on»
  - activée par défaut
- « constraint_exclusion = partition »
  - réservé au partitionnement par héritage
- Élagage non seulement à la planification mais aussi à l’exécution

```
SELECT * FROM livres WHERE titre BETWEEN 'a' AND (SELECT 'c');
```

Jointures entre partitions

- enable_partitionwise_join = off(*)
  - jointures entre les partitions de tables qui ont les mêmes contraintes
  - puis jointure des résultats entre eux

- enable_partitionwise_aggregate = off(*)
  - Agrégation partielle entre les partitions puis agrégation finale

(*) Coût supplémentaire dans la planification

Performances - JIT

- JIT « Just In Time compilation » basé sur LLVM
  - transforme tout ou partie de la requête en un programme natif directement exécuté par le processeur
- Concernés par la compilation JIT :
  - Le décodage des enregistrements « tuples deforming »
  - Les évaluations d'expression, notamment les filtres des clauses WHERE
  - Les agrégats et les GROUP BY
  - Appel de fonctions « inlining »
- Désactivé par défaut : surcoût de planification

Performances – Index

- Colonnes embarquées à un index : INCLUDE
  - Utile pour les parcours d'index seul

- Parallélisme pour la création d'index BTREE
  - « max_parallel_maintenance_workers = 2 »

Performances - Parallélisme

- Nouvelles améliorations :
  - Noeuds Append (UNION ALL)
  - Jointure de type Hash
  - CREATE TABLE … AS SELECT …
  - CREATE MATERIALIZED VIEW
  - SELECT INTO

Performances - pg_prewarm

- Extension pg_prewarm : chargement manuel des données en cache.
- En V11 : mémorisation régulière des blocs dans les shared buffers (5 minutes par défaut)
  - shared_preload_libraries = 'pg_prewarm'
- Chargement automatique de ces blocs au démarrage
  - pg_prewarm.autoprewarm = true

WAL et Checkpoint

- Les WAL étaient conservés le temps de 2 checkpoints avec recyclage des WAL du premier
- Objectif : pouvoir revenir au précédent en cas de problème avec le deuxième
- Plus dangereux qu'utile
- Ce n'est plus le cas avec deux bénéfices directs :
  - Réduction de 33 % du nombre de checkpoint
  - Gain d'espace disque sur le stockage des WAL

Taille des WAL

- La taille du segment WAL est désormais configurable
- 16 MB par défaut, jusqu'à 1 GB
- initdb -D /data/pg11/ --wal-segsize=64
- utile sur des systèmes à très fort volume de génération de WAL

Réplication Logique

- Les ordres TRUNCATE sont à présent répliqués

- Montée de version majeure v10 → v11

Procédure stockées

- Avant : FUNCTION … RETURNS VOID
- CREATE PROCEDURE …
  - Conforme à la norme SQL
  - Appel avec CALL
  - Ne retourne rien
  - Permet un contrôle transactionnel

Exemple PROCEDURE

```
CREATE OR REPLACE PROCEDURE transaction_test1 ()
AS $$
BEGIN
    FOR i IN 0..5 LOOP
        INSERT INTO test1 (a)
            VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
END
$$
LANGUAGE plpgsql;
```

CALL proc_name() ;

ADD COLUMN ... DEFAULT

- ALTER TABLE ADD COLUMN ... DEFAULT ...
  - v10 : réécriture complète de la table !
  - v11 : valeur par défaut mémorisée, ajout instantané

- ... si le défaut n'est pas une fonction volatile

Améliorations JSONB

- Conversion de et vers du type jsonb
  - en SQL : booléen et nombre
  - en PL/Perl : tableau et hash (extension jsonb_plperl)
  - en PL/Python : dict et list (extension jsonb_plpython)

jsonb_to_tsvector()

Conversion JSON en tsvector pour le FTS

- *string* : les chaînes de caractères,
- *numeric* : les valeurs numériques,
- *boolean* : les booléens (true et false),
- *key* : pour inclure toutes les clés de la structure JSON,
- *all* : pour inclure tous les champs ci-dessus.

Fonctions de fenêtrage

- Finalisation du support de la norme SQL:2011
  - { RANGE | ROWS | GROUPS } frame_start [frame_exclusion]
  - { RANGE | ROWS | GROUPS } BETWEEN frame_start AND frame_end [frame_exclusion]
- frame_exclusion :EXCLUDE {CURRENT ROW| GROUP|TIES|NO OTHERS}

Merci !

Liste non exhaustive, de nombreuses autres améliorations ont été apportées.

\q || exit || quit

DALIBO
L'expertise PostgreSQL