PostgreSQL 9.5

Nouveautés et Améliorations

Cycle de Vie

Sortie de PostgreSQL 9.5 en janvier 2016

1 version majeure par an

entre 3 et 6 versions mineures par an : actuellement 9.5.1

Chaque version majeure est supportée 5 ans

Les grandes lignes

UPSERT

Row Level Security

Big Data

UPSERT

```
# CREATE TABLE test (x INTEGER PRIMARY KEY);

# INSERT INTO test VALUES (1);

# INSERT INTO test VALUES (1);

ERROR: duplicate key value violates unique constraint "test_pkey"

# INSERT INTO test VALUES (1) ON CONFLICT DO NOTHING;

INSERT 0 0
```

UPSERT

```
# INSERT INTO test VALUES (2);

# INSERT INTO test VALUES (2)

    ON CONFLICT (x) DO UPDATE SET x = 3;

INSERT 0 1

# INSERT INTO test  VALUES (2)

   ON CONFLICT (x) DO UPDATE SET status = 'conflict';
```

Row Level Security

Nouvelles règles d’accès pour SELECT, INSERT, UPDATE, DELETE

Permissions basées sur le contenu des lignes existantes

Même si l’application a une faille de séurité,

les données sensibles restent invisibles

Row Level Security

SELECT * FROM ventes;

```
id   |  product | user_magasin | ventes
-----+----------+--------------|----------
1    | foo      | lille        |   100
2    | bar      | rennes       |    30
3    | foo      | rennes       |     5
```

Row Level Security

```
CREATE POLICY p ON ventes
  FOR ALL
  TO PUBLIC
  USING (user_magasin = CURRENT_USER );
```

Row Level Security

```
SET SESSION AUTHORIZATION Lille;

SELECT * FROM ventes;

id   |  product | user_magasin | ventes
-----+------------+--------------------|----------
1    | foo          | lille                 |   100
```

Row Level Security

```
SET SESSION AUTHORIZATION rennes;
```

```
id   |  product | user_magasin | ventes
-----+------------+--------------------|----------
2    | bar          | rennes             |    30
3    | foo          | rennes             |     5
```

Row Level Security

```
CREATE POLICY p_region ON ventes
USING ( user_magasin IN
   ( WITH RECURSIVE t AS
       ( SELECT id FROM magasins WHERE id_magasin = CURRENT_USER
    UNION ALL
        SELECT id FROM magasins m INNER JOIN t ON t.region=m.region
       ) SELECT id FROM t
    )
)
```

Big Data / Index Brin

Index très petit pour tables volumineuses

Stocke les valeurs min/max values pour une “tranche” de données (128 pages)

Permet de “sauter” des sections entières de la table

Idéal pour les données inséré chronologiquement

Simple à mettre à jour

Moins rapide qu’un index classique mais beaucoup plus petit

Big Data / Index Brin

```
CREATE TABLE exemple AS SELECT generate_series(1,100000000) AS id;
CREATE INDEX btree_index ON exemple(id);
CREATE INDEX brin_index ON exemple USING brin(id);
```

```
relname          | pg_size_pretty
-----------------+---------------
brin_exemple | 3457 MB
btree_index    | 2142 MB
brin_index       | 104 kB
```

Big Data / OLAP

GROUPING SETS, CUBE and ROLLUP

TABLESAMPLE

Mais aussi....

SQL MED / Foreign Data Wrappers

Encore plus d'opérateurs JSON

Réplication

Optimisations

Skip Locked

checkpoint_segments

SQL-MED

IMPORT FOREIGN SCHEMA

Foreign Table Inheritance

~~Join Push Down~~

JSON

jsonb_set() : remplacement / ajout de données dans un document

```
# SELECT '{"nom": "Damien", "age": "37"}'::jsonb - ‘age’;
  {"name": "Damien"}

SELECT '{"nom": "Damien",}'::jsonb || '{"age": "37"}'::jsonb;
  {"name": "Damien", "age": “37” }
```

json_pretty : affiche un document de manière plus lisible

Optimisations

VACUUM Parallèle

Tris sur texte plus rapides

Lock scalability

Empreinte mémoire réduite

Index Scan Only sur les Index GIST

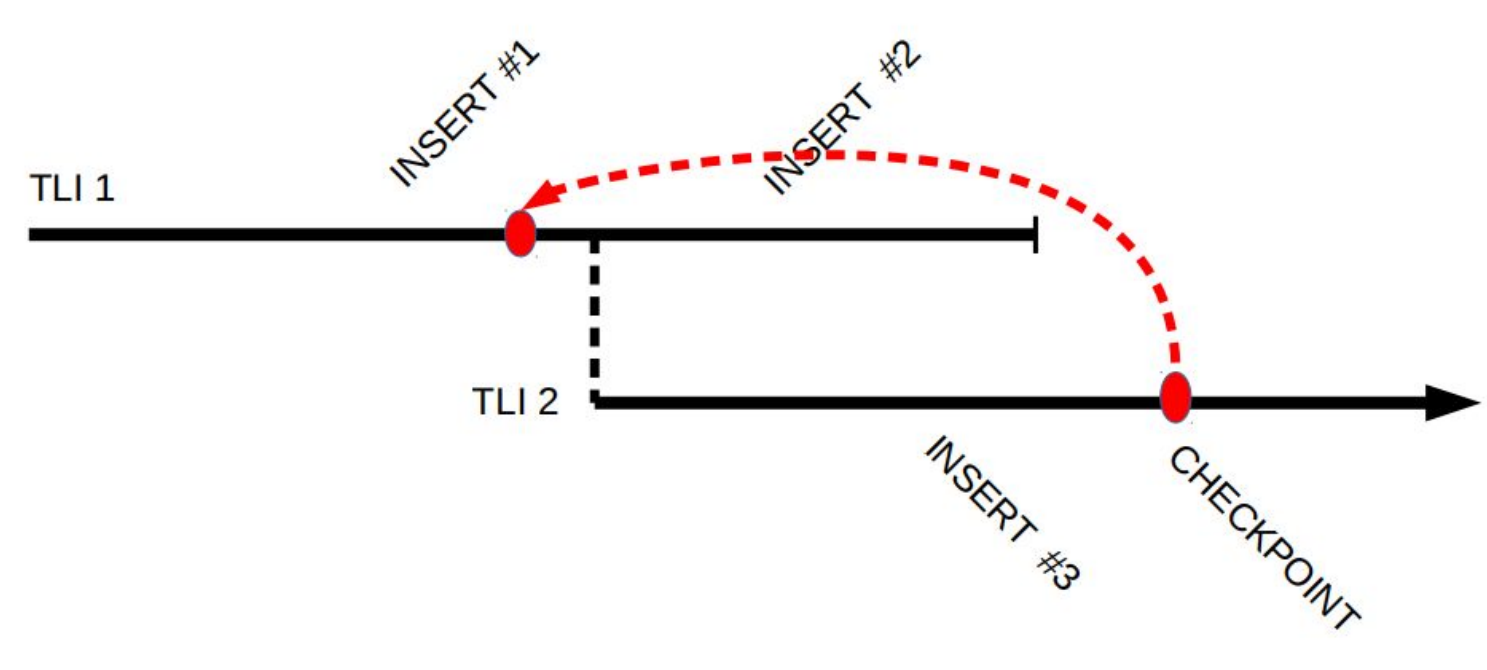
Réplication / Hot Standby

pg_rewind = possibilité de FAIL BACK après un FAIL OVER

Avant la version 9.5, lorsque l'on basculait d'un noeud A vers un noeud B

Il fallait systématique reconstruire le noeud A

Réplication / Hot Standby



Skipped Unlocked

```
# SELECT * FROM a FOR UPDATE NOWAIT;
ERROR: could not obtain lock on row in relation "a"


# SELECT * FROM a FOR UPDATE SKIP LOCKED;
a   | b   | c
----+----+----
2   |  2  | 2
3   |  3  | 3
```

Gestion de journaux (WAL / XLOG)

Paramètre checkpoint_segments supprimé !

Remplacé par des limites sur le dossier pg_xlog :

- min_wal_size (default 80MB)
- max_wal_size (default 1GB)

L'agencement des checkpoints est géré automatiquement à partir ces 2 valeurs...

Mais aussi….

ALTER TABLE ... SET LOGGED / UNLOGGED

recovery_target_action = promote / pause / shutdown

ALTER SYSTEM RESET

cluster_name

pg_stat_ssl

pg_stat_statements améliorée : min_time, max_time, mean_time, stddev_time

En avant vers la version 9.6

Parallélisme !

Actuellement : 1 requête = 1 coeur

Avec 9.6 : “Dispatcher” un scan sequentiel sur plusieurs coeurs

En avant vers la version 9.6

Des avancées vers la réplication logique

Syntaxe de Partitionnement

Sharding (FDW join pushdown , FDW DML pushdown)

....

Première beta en juin ?

Sources

https://wiki.postgresql.org/wiki/What%27s_new_in_PostgreSQL_9.5

https://www.youtube.com/watch?v=qluVWI1UKiM

https://momjian.us/main/writings/pgsql/features.pdf

Suivre l’actualité PostgreSQL

www.postgresql.fr

planet.postgresql.org

Prochain RDV à Toulouse ?

http://www.meetup.com/fr-FR/PostgreSQL-User-Group-Toulouse/