PostgreSQL 12 and beyond

PostgreSQL Conference Japan 2019
November 15th, Tokyo
Michael Paquier

The man

- Michael Paquier.
- French, based in Tokyo.
- PostgreSQL contributor since 2009
  - Committer
  - Hacker, Blogger
- Twitter: @michaelpq
- Website: https://paquier.xyz
- Working at VMware: Packaging, Integration, Support.

Partitioning - 1

- Performance improvements with many partitions
  - COPY, switch to bulk-inserts
  - INSERT, lock partition before insertion of single row
  - SELECT, partition pruning and meta-data handling
- About performance, David Rowley, 2ndQ: https://www.2ndquadrant.com/en/blog/postgresql-12-partitioning/

Partitioning - 2

- Tablespace inheritance.
- Foreign keys to reference partitioned tables.
- Expressions for partition bounds.
- ATTACH PARTITION <=> Lock level lower.

Partitioning - 3

- Partition functions

  pg_partition_root()
  pg_partition_ancestors()
  pg_partition_tree()

```
=# SELECT * FROM pg_partition_tree('parent_tab');
    relid    | parentrelid | isleaf | level
-------------+-------------+--------+-------
 parent_tab  | null        | f      |     0
 child_0_10  | parent_tab  | t      |     1
 child_10_20 | parent_tab  | t      |     1
 child_20_30 | parent_tab  | t      |     1
 child_30_40 | parent_tab  | f      |     1
 child_30_35 | child_30_40 | t      |     2
 child_35_40 | child_30_40 | t      |     2
(7 rows)
```

Partitioning - next?

- Planner still slow with many partitions.
- More partition-wise joins?
- Logical replication and partitioned tables.
- Think carefully about partitioning strategy.
  - Redistribution.
  - OLTP and/or analytics.

Btree indexes

- Many duplicates
  - Sort in heap-storage order
  - Storage lower
  - Performance with VACUUM and INSERT
- Multi-column storage smaller.
- Pre-11 indexes compatible after pg_upgrade but require REINDEX to get benefits.

REINDEX

- REINDEX => access exclusive lock.
  - No writes and no reads.
  - Blocks production and takes time.
  - Can be monitored in v12~.
- CONCURRENTLY
  - Allows read and writes.
  - Takes longer, handles dependencies automatically.
  - pg_reorg/pg_repack.

REINDEX - next?

- Parallel job support in reindexdb --jobs (done!)
- Collation version storing
- Filtering of collation-sensitive indexes
- glibc breaking indexes randomly with its upgrades!

Generated columns

- Two kinds: stored and virtual.
- Postgres supports only stored.
- Less triggers, some restrictions on expressions.

```
=# CREATE TABLE person_data (
     person_id int,
     weight_grams numeric,
     weight_kilos numeric GENERATED ALWAYS AS
      (weight_grams / 1000) STORED);
CREATE TABLE
=# INSERT INTO person_data VALUES (1, 55000);
INSERT 0 1
=# SELECT * FROM person_data;
 person_id | weight_grams |    weight_kilos
-----------+--------------+--------------------
         1 |        55000 | 55.0000000000000000
(1 row)
```

WITH and MATERIALIZE

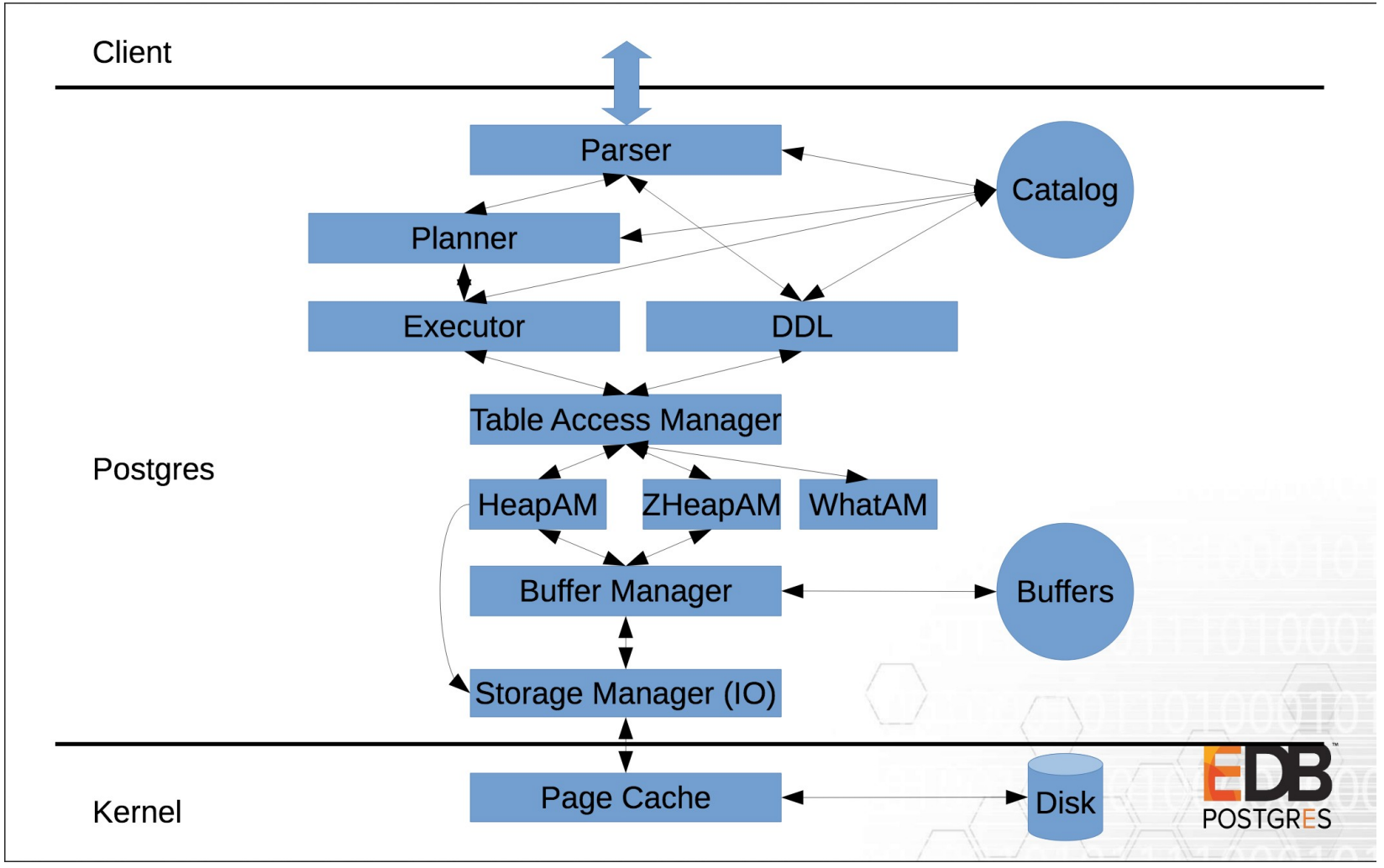
- Before 11: Always materialize (temporary copy)
  - Advantage: DML + RETURNING.
  - Not much: CTE with large scan.

```
=# EXPLAIN (COSTS OFF)
     WITH large_scan AS MATERIALIZED
     (SELECT * FROM very_large_tab)
   SELECT * FROM large_scan WHERE id = 1;
             QUERY PLAN
-----------------------------------
 CTE Scan on large_scan
   Filter: (id = 1)
   CTE large_scan
     ->  Seq Scan on very_large_tab
(4 rows)
```

```
=# EXPLAIN (COSTS OFF)
     WITH large_scan AS NOT MATERIALIZED
     (SELECT * FROM very_large_tab)
   SELECT * FROM large_scan WHERE id = 1;
                QUERY PLAN
-------------------------------------------
 Gather
   Workers Planned: 2
   ->  Parallel Seq Scan on very_large_tab
         Filter: (id = 1)
(4 rows)
```

Table Access Methods - 1

- Plugin facility to control table engine, not indexes!
- API not to be considered stable, designed to evolve and break.
- Heap is the default.
- Zedstore (columnar), zheap (UNDO-based).
- Limitation with WAL, reloptions (locks fixed in 13~).
- Two categories
  - Uses Postgres shared buffers, page format, storage..
  - The rest, can do a lot.



Andres Freund (slide 5)

- https://anarazel.de/talks/2018-10-25-pgconfeu-pluggable-storage/pluggable.pdf

Table Access Methods - 3

- Columnar, compression storage, etc.
- Example: blackhole_am
  https://github.com/michaelpq/pg_plugins/tree/master/blackhole_am

```
=# CREATE EXTENSION blackhole_am;
CREATE EXTENSION
=# CREATE TABLE blackhole_tab (a int) USING blackhole_am;
CREATE TABLE
=# INSERT INTO blackhole_tab VALUES (1);
INSERT 0 1
=# SELECT * FROM blackhole_tab;
 a
---
(0 rows)
```

Recovery

- All recovery parameters become GUCs:
  - SHOW and ALTER SYSTEM
  - trigger_file => promote_trigger_file
  - standby_mode gone
  - No multiple recovery targets
  - recovery_target_timeline ~> default to ‘latest’
- recovery.conf gone => standby.signal & recovery.signal
- pg_promote() as SQL function.

Data checksums

- pg_checksums
  - Enable and disable, no parallel support
  - Progress reporting
  - Only for offline cluster
  - Renamed from pg_verify_checksums in 12~
- Checksum failures in pg_stat_database
- Next: Online mode?

Transaction chains

- COMMIT AND CHAIN
- ROLLBACK AND CHAIN
- Error outside transaction block.

```
=# COMMIT AND CHAIN;
ERROR:  25P01: COMMIT AND CHAIN can only be used in transaction blocks
=# BEGIN;
BEGIN
=# COMMIT AND CHAIN;
COMMIT
=# BEGIN;
WARNING:  25001: there is already a transaction in progress
BEGIN
=# COMMIT;
COMMIT
```

Progress reporting

- pg_stat_progress_vacuum since v9.6.
- CREATE INDEX + REINDEX (+ CONCURRENTLY)
  pg_stat_progress_create_index
- CLUSTER and VACUUM FULL
  pg_stat_progress_cluster
- Progress phases can be confusing:
  https://www.postgresql.org/docs/current/progress-reporting.html

jsonpath

- Expressions => like XPath for XML.
- jsonpath data type, lookup at parts of JSON tree.
- Operators.
- No datetime yet (committed in v13~).
- Documentation:
- https://www.postgresql.org/docs/devel/functions-json.html

Release notes:
https://www.postgresql.org/docs/12/released

Thanks!
Questions?