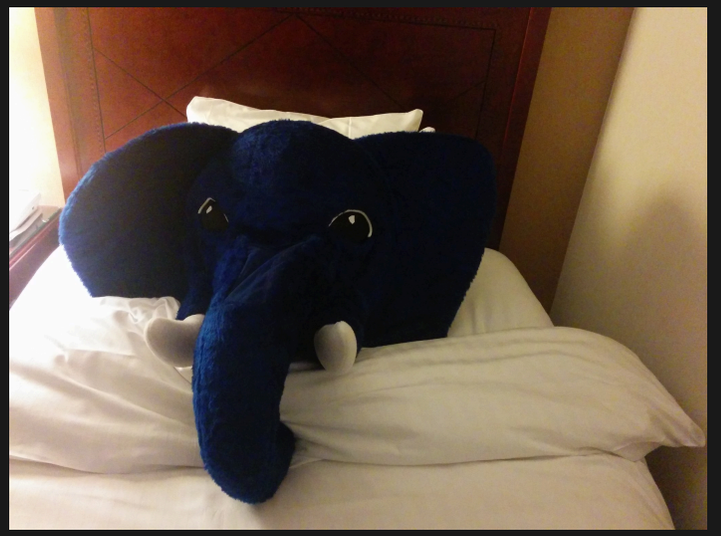
A look at the Elephants Trunk

PostgreSQL 18

PGConf.NYC 2025
New York, USA

Magnus Hagander
magnus@hagander.net



Magnus Hagander

- Redpill Linpro
  - Principal database consultant
- PostgreSQL
  - Core Team member
  - Committer
  - PostgreSQL Europe

PostgreSQL 18

Development schedule

- July 2024 - branch 17
- July 2024 - CF1
- September 2024 - CF2
- November 2024 - CF3
- January 2025 - CF4
- March 2025 - CF5
- September 2025 - Release!

Current status

- 3046 commits
- 3987 files changed, 412360 insertions(+), 211178 deletions(-)

New features

- DBA and administration
- SQL and developer
- Backup and replication
- Performance

Breaking changes

Building

Building

- Remove support for HPPA

Building

- Remove support for HPPA
- Remove support for lack of spinlocks

Building

- Remove support for HPPA
- Remove support for lack of spinlocks
- Remove support for lack of atomics

OpenSSL

- Remove support for OpenSSL older than 1.1.1

New features

- DBA and administration
- SQL and developer
- Backup and replication
- Performance

Checksums

- Enabled by default
  - Finally!
- By initdb

  `--no-data-checksums` to disable
- *NOT* on upgrades

Upgrades & stats

- Stats are transferred on *pg_upgrade*
  - `pg_stats`, not `pg_stat`
- Ready to use much faster after upgrade!
- Actually in *pg_dump*

Authentication

Authentication

- md5 deprecated
  - Can set *md5_password_warnings=off*
  - But don't!

Authentication

- OAUTHBEARER
- Log in using OAUTH bearer token
- Requires server side provider
  - Written in C
  - No default provided

Authentication

- SCRAM pass-through
- In *postgres_fdw* and *dblink*
- No need for clear-text password
- `use_scram_passthrough=true` on *SERVER*
- Must have same salt and iteration count!

TLS

- Support for TLSv1.3 cipher suites
- Support for multiple ECDH curves

Crypto

- pg_crypto can disable built-in crypto

(auto)vacuum

autovacuum_max_threshold

- Upper bound on calculated threshold
- For large tables
  - *rows * vacuum_scale_factor* too large
- Default: 100M

autovacuum_max_workers

- Change without restart
- Up to *autovacuum_worker_slots*

VACUUM [ONLY]

- For both *VACUUM* and *ANALYZE*
- Specify ONLY to not recurse into partitions
- ANALYZE particularly useful for partitioned tables

EXPLAIN ANALYZE

- *BUFFERS* enabled by default
- Show parallel bitmap scan stats
- Show memory/disk use for Materialize nodes

COPY

- log_verbosity = 'silent'

```
postgres=# COPY a FROM '/tmp/test.csv' WITH (FORMAT csv, ON_ERROR ignore);
NOTICE:  2 rows were skipped due to data type incompatibility
COPY 4
postgres=# COPY a FROM '/tmp/test.csv' WITH (FORMAT csv,
        ON_ERROR ignore, LOG_VERBOSITY silent);
COPY 4
```

Statistics

Parallel worker

- New fields
  - *parallel_workers_to_launch*
  - *parallel_workers_launched*
- Per db or statement
  - pg_stat_database
  - pg_stat_statements

VACUUM

- Per table time spent
  - total_vacuum_time
  - total_autovacuum_time
  - total_analyze_time
  - total_autoanalyze_time

VACUUM

- Time spent delaying
  - pg_stat_progress_vacuum
  - pg_stat_progress_analyze

WAL

- Now tracked in *pg_stat_io*
  - Much more granular
  - Per backend-type
- Removed from *pg_stat_wal*

WAL

- *wal_buffers_full*
  - Added to pg_stat_statements
  - In VACUUM/ANALYZE VERBOSE
  - In EXPLAIN (WAL)
- Still globally in pg_stat_wal

New features

- DBA and administration
- SQL and developer
- Backup and replication
- Performance

UUIDv7

- New generation function
- Sortable
  - Standard says milliseconds
  - PostgreSQL does 12-bit sub-millisecond
- Better for indexes

```
postgres=# select uuidv7();
 019560db-323b-7515-bb59-abd0d261440c
postgres=# select uuidv7();
 019560db-351c-7691-915f-d0460f0ddc7a
postgres=# select uuidv7();
 019560db-3d94-7b4b-9df8-1caa616fea5a
```

OLD/NEW for RETURNING

- Ability to access both old and new value
  - In UPDATE
  - And MERGE

```
postgres=# UPDATE t SET a=2 RETURNING OLD.a, NEW.a;
 1 | 2

postgres=# UPDATE t SET a=a+1 RETURNING OLD.a, NEW.a;
 2 | 3
```

OLD/NEW for RETURNING

- But also for ON CONFLICT
- Determine INSERT or UPDATE

```
postgres=# INSERT INTO t(a,b) VALUES (1,1)
   ON CONFLICT (a) DO UPDATE SET b=t.b+1
   RETURNING OLD.a, NEW.a, OLD.b, NEW.b;
   | 1 |   | 1
...
 1 | 1 | 1 | 2
...
 1 | 1 | 2 | 3
```

Virtual generated columns

- Like *STORED* virtual columns
- Except not.. stored.
- Re-calculated on each read
- Cannot be indexed
- "Partial view"

Virtual generated columns

```
CREATE TABLE test (
 a int,
 b int,
 c int GENERATED ALWAYS AS (a+b),
 d int GENERATED ALWAYS AS (a+b) STORED
)
```

Temporal keys

- PRIMARY and FOREIGN

```
CREATE TABLE temptest (
  id int,
  valid daterange,
  CONSTRAINT pk_test PRIMARY KEY (id, valid WITHOUT OVERLAPS),
  CONSTRAINT fk_test FOREIGN KEY (id, PERIOD valid)
    REFERENCES test2(id, PERIOD valid)
)
```

Temporal keys

- You probably want *btree_gist*

array_reverse()

- Yup, that simple...

New features

- DBA and administration
- SQL and developer
- Backup and replication
- Performance

pg_verifybackup

- Can now verify tar format
- (previously only plain)

Replicate generated columns

- Logical replication of generated columns
  - Only stored!

```
CREATE PUBLICATION test
FOR TABLE test
WITH (publish_generated_columns='stored')
```

```
CREATE PUBLICATION test
FOR TABLE test (a, b, d)
```

pg_stat_subscription_stats

- Collects conflict stats
- INSERT conflicts
- UPDATE conflicts
- Origin conflicts
- UPDATE missing
- DELETE missing

New features

- DBA and administration
- SQL and developer
- Backup and replication
- Performance

Many different

- Lots of infrastructure
- Often not directly exposed

VACUUM

- Use streaming I/O
- More eagerly vacuum all-visible pages
  - To make aggressive vacuum cheaper
- (... more)

Parallel CREATE INDEX

- Now also for GIN
- (in addition to btree and brin)

btree index skip-scan

- Use multi-column index for non-prefix scans
- Not as fast as dedicated index
- But fewer indexes!
- Typically with few distinct values in early columns

pg_upgrade

- Much more parallel
  - Previously just *pg_dump* and copy/link
- `--swap` mode
  - Move data directory, then overwrite catalog
  - Fast, but no rollback

General queries

- Detect redundant GROUP BY based on UNIQUE
  - Previously only PRIMARY KEY
- Proper row estimates for *generate_series*
  - Numeric and timestamp
    - Already did for integer

General queries

- Optimized tuplestore for recursive CTE
  - Much faster for some queries (25+%)
- Reduced memory usage on partitionwise join
- JSON escaping using SIMD
- Right Semi Join
- Faster numeric multiplication and division

Self Join Elimination

- Remove self-joins
  - When a table is already joined
  - And can be proven to be the same output
- Often caused by VIEWs or ORMs

```
postgres=# EXPLAIN SELECT one.a, one.b, two.a, two.b
     FROM t1 one INNER JOIN t1 two ON one.a=two.a;

                        QUERY PLAN
-----------------------------------------------------
 Seq Scan on t1 two  (cost=0.00..32.60 rows=2260 width=16)
```

Asynchronous I/O

- Worker or io_uring
  - Default: worker
- Faster prefetching
- Foundation for direct-io
  - But not there yet
- Only read (for now)

Many infrastructure

- No direct visibility
- Just runs faster
- (almost every version)

There's always more

There's always more

- Lots of smaller fixes
- Performance improvements
- etc, etc
- Can't mention them all!

Please help!

- Download and test!
  - apt packages available
  - rpm/yum packages available

Thank you!

Magnus Hagander
magnus@hagander.net
bsky: @magnus.hagander.net
https://www.hagander.net/talks/