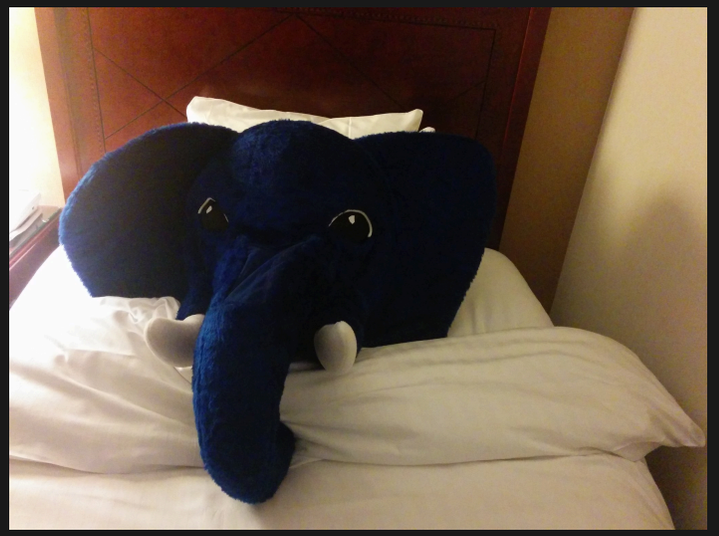
What's new in

PostgreSQL 19

PGDay Lowlands 2026
Utrecht, Netherlands



Magnus Hagander
magnus@hagander.net

Magnus Hagander

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

PostgreSQL 19

Development schedule

- June 2025 - branch 18
- July 2025 - CF1
- September 2025 - CF2
- November 2025 - CF3
- January 2026 - CF4
- March 2026 - CF5
- August 2026 - beta 3

Current status

- 3416 commits
- 4585 files changed, 541884 insertions(+), 336962 deletions(-)

New features

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

Breaking changes

Removed

- RADIUS authentication

Changed defaults

Changed defaults

- `jit=off`

Changed defaults

- `jit=off`
- `log_lock_waits=on`

Strings

- `standards_conforming_strings`
  - Unconditionally *on*

Before we begin

psql prompt

- %i for primary/standby
- %S for search_path

New features

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

Checksums

- Enable/disable online
- Slow operation
  - Rewrites whole cluster!
- Done in background

Checksums

- Enable/disable online
- Slow operation
  - Rewrites whole cluster!
- Done in background

```
SELECT pg_enable_data_checksums();
```

Checksums

- Enable/disable online
- Slow operation
  - Rewrites whole cluster!
- Done in background

```
SELECT pg_enable_data_checksums();
```

```
SELECT pg_disable_data_checksums();
```

COPY

COPY

- Support multi-line headers

COPY

- Support multi-line headers

```
COPY t FROM '/tmp/t.csv' WITH (FORMAT csv, HEADER 2)
```

COPY

- Error handling

COPY

- Error handling

```
postgres=# COPY t FROM '/tmp/t.csv' WITH (format csv, on_error ignore);
NOTICE:  2 rows were skipped due to data type incompatibility
```

COPY

- Error handling

```
postgres=# COPY t FROM '/tmp/t.csv' WITH (format csv, on_error ignore);
NOTICE:  2 rows were skipped due to data type incompatibility
```

```
postgres=# COPY t FROM '/tmp/t.csv' WITH (format csv, on_error set_null);
NOTICE:  in 2 rows, columns were set to null due to data type incompatibility
```

COPY

- JSON output
- Directly in NDJSON
  - Or wrapped in array

COPY

- JSON output
- Directly in NDJSON
  - Or wrapped in array

```
postgres=# COPY t TO stdout WITH (format json, force_array);
[
 {"a":1,"b":1}
,{"a":2,"b":2}
,{"a":null,"b":null}
]
```

COPY TO

- Now works for partitioned tables
- Don't need individual partition

Autovacuum

Autovacuum

- Parallelization
  - Now supported

Autovacuum

- Parallelization
  - Now supported
- Prioritization
  - Each table gets a "score"
  - "Distance from limit"
  - Processed in order

Monitoring and stats

Autovacuum

Autovacuum

```
postgres=# SELECT * FROM pg_stat_autovacuum_scores;
-[ RECORD 1 ]-------+--------
relid               | 16388
schemaname          | public
relname             | t
score               | 0.62
xid_score           | 6.5e-08
mxid_score          | 0
vacuum_score        | 0.28
vacuum_insert_score | 0.017
analyze_score       | 0.62
do_vacuum           | f
do_analyze          | f
for_wraparound      | f
```

Locks

Locks

```
postgres=# SELECT * FROM pg_stat_lock ORDER BY waits DESC;
     locktype      | waits | wait_time | fastpath_exceeded | stats_reset
-------------------+-------+-----------+-------------------+-------------
 relation          |     2 |  4647.493 |                 0 | ...
 transactionid     |     1 |  2884.261 |                 0 | ...
 frozenid          |     0 |         0 |                 0 | ...
 page              |     0 |         0 |                 0 | ...
 tuple             |     0 |         0 |                 0 | ...
 virtualxid        |     0 |         0 |                 0 | ...
 spectoken         |     0 |         0 |                 0 | ...
 object            |     0 |         0 |                 0 | ...
 userlock          |     0 |         0 |                 0 | ...
 advisory          |     0 |         0 |                 0 | ...
 applytransaction  |     0 |         0 |                 0 | ...
 extend            |     0 |         0 |                 0 | ...
```

pg_stat_statements

- Counters for plan types

pg_stat_statements

- Counters for plan types

```
postgres=# SELECT query, generic_plan_calls, custom_plan_calls FROM pg_stat_state

      query      | generic_plan_calls | custom_plan_calls
-----------------+--------------------+-------------------
 select * from t |                  4 |                 0
```

wal_fpi_bytes

- `pg_stat_wal`
- EXPLAIN (ANALYZE, WAL)
- VACUUM and ANALYZE logs

New wait events

- BufferCleanup, BufferExclusive, BufferShared, BufferSharedExclusive
- CopyFromRead, CopyToWrite
- CommitDelay
- (several more, totally 19 new, 2 removed)

log_min_messages

- Can be set per process type
- E.g. "backend", "checkpointer", "walreceiver"

```
SET  log_min_messages = 'backend:error,warning'
```

REPACK

REPACK

- Like `VACUUM FULL`
- But can be done `CONCURRENTLY`
  - as long as there's a primary key
  - or other identity index
- Based on logical decoding

REPACK

- Like `VACUUM FULL`
- But can be done `CONCURRENTLY`
  - as long as there's a primary key
  - or other identity index
- Based on logical decoding

```
postgres=# REPACK (CONCURRENTLY) t;
REPACK
```

Plan hinting

Plan hinting

- Two extensions
- `pg_plan_advice`
  - Give planner hints
- `pg_stash_advice`
  - Store and re-use hints

pg_plan_advice

```
ostgres=# LOAD 'pg_plan_advice';
LOAD

postgres=# EXPLAIN (PLAN_ADVICE) SELECT * FROM t WHERE a=1;
                     QUERY PLAN
-----------------------------------------------
 Seq Scan on t  (cost=0.00..1.12 rows=1 width=8)
   Filter: (a = 1)
 Generated Plan Advice:
   SEQ_SCAN(t)
   NO_GATHER(t)
(5 rows)
```

pg_plan_advice

```
postgres=# SET pg_plan_advice.advice = 'INDEX_SCAN(t public.t_pkey)';
SET

postgres=# EXPLAIN SELECT * FROM t WHERE a=1;
                           QUERY PLAN
-----------------------------------------------------------------
 Index Scan using t_pkey on t  (cost=0.14..8.15 rows=1 width=8)
   Index Cond: (a = 1)
 Supplied Plan Advice:
   INDEX_SCAN(t public.t_pkey) /* matched */
(4 rows)
```

pg_stash_advice

```
SELECT pg_create_advice_stash('mystash');
```

pg_stash_advice

```
SELECT pg_create_advice_stash('mystash');
```

```
EXPLAIN (VERBOSE) SELECT * FROM t WHERE a=1;
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on public.t  (cost=0.00..1.12 rows=1 width=8)
   Output: a, b
   Filter: (t.a = 1)
 Query Identifier: 8849960880760060411
```

pg_stash_advice

```
SELECT pg_create_advice_stash('mystash');
```

```
EXPLAIN (VERBOSE) SELECT * FROM t WHERE a=1;
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on public.t  (cost=0.00..1.12 rows=1 width=8)
   Output: a, b
   Filter: (t.a = 1)
 Query Identifier: 8849960880760060411
```

```
SELECT pg_set_stashed_advice('mystash', 8849960880760060411,
   'INDEX_SCAN(t public.t_pkey)');
```

pg_stash_advice

```
SET pg_stash_advice.stash_name = 'mystash';
```

pg_stash_advice

```
SET pg_stash_advice.stash_name = 'mystash';
```

```
EXPLAIN SELECT * FROM t WHERE a=2;
                            QUERY PLAN
-------------------------------------------------------------------
 Index Scan using t_pkey on t  (cost=0.14..8.15 rows=1 width=8)
   Index Cond: (a = 2)
 Supplied Plan Advice:
   INDEX_SCAN(t public.t_pkey) /* matched */
```

EXPLAIN (IO)

- Show I/O information in explain

```
EXPLAIN (ANALYZE, IO) SELECT * FROM t
```

EXPLAIN (IO)

- Show I/O information in explain

```
EXPLAIN (ANALYZE, IO) SELECT * FROM t
```

```
Seq Scan on t  (cost=0.00..1.10 rows=10 width=8) ...
  Prefetch: avg=1.00 max=1 capacity=94
  Buffers: shared hit=1
```

New features

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

Window functions

- IGNORE NULLs
- RESPECT NULLs (default, and always before)
- lead(), lag(), first_value(), last_value(), nth_value()

Window functions

- IGNORE NULLs
- RESPECT NULLs (default, and always before)
- lead(), lag(), first_value(), last_value(), nth_value()

```
SELECT a, last_value(b) IGNORE NULLS OVER (PARTITION BY a) FROM t
```

INSERT ON CONFLICT

- DO SELECT

INSERT ON CONFLICT

- DO SELECT

```
INSERT INTO t VALUES (20, 30)
ON CONFLICT (a) DO SELECT
RETURNING *
```

FOR PORTION OF

- Works with temporal tables
- "split" up a record

FOR PORTION OF

```
CREATE TABLE ttest (
  id int not null,
  valid daterange not null,
  name text not null,
  CONSTRAINT pk_ttest PRIMARY KEY (id, valid WITHOUT OVERLAPS)
)
```

```
 id |           valid           | name
----+---------------------------+------
  1 | [2026-01-01,2026-12-31) | Test
```

FOR PORTION OF

```
 id |           valid            | name
----+----------------------------+------
  1 | [2026-01-01,2026-12-31) | Test
```

FOR PORTION OF

```
 id |           valid           | name
----+---------------------------+------
  1 | [2026-01-01,2026-12-31) | Test
```

```
UPDATE ttest
FOR PORTION OF valid FROM '2026-05-10' TO '2026-06-15'
SET name='Updated' WHERE id=1
```

FOR PORTION OF

```
 id |           valid            | name
----+----------------------------+------
  1 | [2026-01-01,2026-12-31) | Test
```

```
UPDATE ttest
FOR PORTION OF valid FROM '2026-05-10' TO '2026-06-15'
SET name='Updated' WHERE id=1
```

```
1 | [2026-01-01,2026-05-10) | Test
1 | [2026-05-10,2026-06-15) | Updated
1 | [2026-06-15,2026-12-31) | Test
```

New features

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

wal_level

- Automatic between *replica* and *logical*
- logical enabled when slots exist
- No need to restart!

```
SHOW effective_wal_level
```

Sequences

- Logical replication can now synchronize sequences
- Not live replication, synchronization!

Sequences

- Logical replication can now synchronize sequences
- Not live replication, synchronization!

```
CREATE PUBLICATION spub FOR ALL SEQUENCES
```

Sequences

- Logical replication can now synchronize sequences
- Not live replication, synchronization!

```
CREATE PUBLICATION spub FOR ALL SEQUENCES
```

```
ALTER SUBSRIPTION ssub REFRESH SEQUENCES
```

Logical replication

- Define except rules on publications

```
CREATE PUBLICATION pub
FOR ALL TABLES
EXCEPT (TABLE public.t, TABLE public.t2);
```

WAIT FOR LSN

- Wait for LSN to reach specific state
- "Wait for rows visible on standby"

WAIT FOR LSN

- Wait for LSN to reach specific state
- "Wait for rows visible on standby"

```
SELECT pg_current_wal_insert_lsn();
 pg_current_wal_insert_lsn
---------------------------
 0/031442C0
```

WAIT FOR LSN

- Wait for LSN to reach specific state
- "Wait for rows visible on standby"

```
SELECT pg_current_wal_insert_lsn();
 pg_current_wal_insert_lsn
---------------------------
 0/031442C0
```

```
postgres=# wait for lsn '0/031442C0';
 status
---------
 success
```

New features

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

Compression defaults

- Toast compresison now *lz4*
- WAL compression now *zstd* or *lz4*
- If available

Eager aggregation

- Pushes aggregation past a join
- Aggregate first, fewer rows to join
- Faster aggregates over joins

LISTEN/NOTIFY

- Wake up only relevant backends
  - Previously all backends
- Large improvement with many individual channels
- (no change if all listen to the same)

IS DISTINCT FROM

- Optimize for non-nullable columns
- Where it's the sam eas not equals
- Indexes can be used for `IS NOT DISTINCT FROM`

I/O workers

- Pool now automatically tuned
  - (for `io_method=worker`, of course)
- Between *io_min_workers* and *io_max_workers*

SIMD

- Optimize hex_encode()/hex_decode()
- Optimize COPY FROM in text or csv

Fast-path RI

- Foreign key checking
- Non-partitioned tables
- No temporal keys
- Optimizes the check, not referential actions

Buffer locking

- Faster and more scalable!
- Separate lock managment
- freelist removed
- More atomic operations
- <add lots of complex descriptions>
- More and changed wait events!

Many different

- Lots of infrastructure

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/