A look at the Elephants Trunk
PostgreSQL 17

SCaLE 2024
Pasadena, CA, USA

Magnus Hagander
magnus@hagander.net

Magnus Hagander

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

PostgreSQL 17

Development schedule

- June 2023 - branch 16
- July 2023 - CF1
- September 2023 - CF2
- November 2023 - CF3
- January 2024 - CF4
- March 2025 - **CF5**

Current status

- 1609 commits
- 3472 files changed, 153396 insertions(+), 78308 deletions(-)

New features

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

Breaking changes

Building

- Windows MSVC builds
- AIX support
- --disable-thread-safety

Removed features

- adminpack
- db_user_namespace
- snapshot too old

pg_stat_bgwriter

- Removed checkpoints_timed & req
- Removed write_time & sync_time
- Removed buffers_checkpoint, backend & fsync

Breaking change

- search_path during maintenance ops
  - Secured by default!
  - Must be explicit!

New features

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

psql

- \watch termination
- When rows drop below count

psql

- \watch termination
- When rows drop below count

```
postgres=# <query>
postgres-# \watch m=2
```

timeout

- *transaction_timeout*

Event triggers

Event triggers

- REINDEX event triggers

Event triggers

- REINDEX event triggers
- Login event triggers

Event triggers

- REINDEX event triggers
- Login event triggers
    - Footgun extraordinaire!

Event triggers

- REINDEX event triggers
- Login event triggers
    - Footgun extraordinaire!
- *event_triggers=false*

Wait events

- *pg_wait_events*

```
postgres=# SELECT * FROM pg_wait_events  WHERE name='PgSleep';
  type    |   name   |                    description
----------+----------+-------------------------------------------------------
 Timeout | PgSleep | Waiting due to a call to pg_sleep or a sibling function
(1 row)
```

Wait events

- Custom wait events for extensions

Statistics!

pg_stat_bgwriter

- Removed checkpoints_timed & req
- Removed write_time & sync_time
- Removed buffers_checkpoint, backend & fsync

pg_stat_checkpointer

```
postgres=# select * from pg_stat_checkpointer ;
-[ RECORD 1 ]-------+-------------------------------
num_timed           | 3
num_requested       | 0
restartpoints_timed | 0
restartpoints_req   | 0
restartpoints_done  | 0
write_time          | 4314
sync_time           | 7
buffers_written     | 43
```

pg_stat_statements

- Local block I/O
- Entry time

```
...
local_blk_read_time    | 0
local_blk_write_time   | 0
...
stats_since            | 2024-03-09 15:39:11.483719+01
minmax_stats_since     | 2024-03-09 15:39:11.483719+01
```

pg_stat_statements

- Normalize parameters in CALL

```
...
queryid              | 1774110370767368945
query                | call dummyproc($1,$2)
...
```

pg_stat_vacuum_progress

- Shows index progress

```
...
phase              | vacuuming indexes
...
indexes_total      | 5
indexes_processed  | 3
```

COPY

```
postgres=# COPY dummy FROM '/tmp/test.csv' WITH (FORMAT csv);
2024-03-09 15:53:00.105 CET [3613894] ERROR:  invalid input syntax for type integer
2024-03-09 15:53:00.105 CET [3613894] CONTEXT:  COPY dummy, line 2, column b: "fo
2024-03-09 15:53:00.105 CET [3613894] STATEMENT:  COPY dummy FROM '/tmp/test.csv
ERROR:  invalid input syntax for type integer: "foo"
CONTEXT:  COPY dummy, line 2, column b: "foo"
```

COPY

- Error handling!

```
postgres=# COPY dummy FROM '/tmp/test.csv' WITH (FORMAT csv);
2024-03-09 15:53:00.105 CET [3613894] ERROR:  invalid input syntax for type integ
2024-03-09 15:53:00.105 CET [3613894] CONTEXT:  COPY dummy, line 2, column b: "fo
2024-03-09 15:53:00.105 CET [3613894] STATEMENT:  COPY dummy FROM '/tmp/test.csv
ERROR:  invalid input syntax for type integer: "foo"
CONTEXT:  COPY dummy, line 2, column b: "foo"
```

```
postgres=# COPY dummy FROM '/tmp/test.csv' WITH (FORMAT csv, ON_ERROR 'ignore');
NOTICE:  1 row was skipped due to data type incompatibility
COPY 2
```

Maintenance permissions

- Grant maintenance tasks to non-table-owners
    - VACUUM, ANALYZE
    - CLUSTER
    - REINDEX
    - REFRESH MATERIALIZED VIEW
    - LOCK TABLE

Maintenance permissions

```
postgres=# GRANT MAINTAIN ON mytable TO testuser;
GRANT


postgres=# GRANT pg_maintain TO testuser;
GRANT
```

builtin locale provider

- Only for "C"
- Faster!
- **Stable!**

New features

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

Binary and octal

```
postgres=# SELECT to_bin(123), to_oct(123);
 to_bin   | to_oct
----------+---------
 1111011  | 173
(1 row)
```

Infinite intervals

```
postgres=# SELECT now() + 'infinity';
 ?column?
----------
 infinity

postgres=# SELECT 'infinity'::timestamptz - now();
 ?column?
----------
 infinity
(1 row)
```

Partitioned tables

Identity columns

- Works properly

Temporal PRIMARY KEY/UNIQUE

```sql
CREATE TABLE test (
  id int,
  valid tstzrange,
  CONSTRAINT pk_test PRIMARY KEY (id, valid WITHOUT OVERLAPS)
)
```

PQchangePassword

- New libpq function
- Use to…. Change passwords!
- Used to be psql-only

JSONPATH

- Many new operators
- Convert between "data types"
- E.g. *.string()* and *.boolean()*

New features

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

pg_dump

- Get list of include/exclude from file

```
$ cat /tmp/t.list
include table foo
include table bar
include table something.*
exclude table_data something.foobar
$ pg_dump -Fc -d postgres --filter /tmp/t.list -f ...
```

Incremental pg_basebackup

- Back up only changed pages/blocks
- Uses *wal summarizer*

```
summarize_wal = on
#wal_summary_keep_time = '10d'
```

Incremental pg_basebackup

- Backup references manifest from full backup

```
$ pg_basebackup -Fp -D /backup/full
...
...
$ pg_basebackup -Fp --incremental=/backup/full/backup_manifest -D /backup/incr
```

Incremental pg_basebackup

- To restore, use *pg_combinebackup*

```
$ pg_combinebackup -o /backup/combined /backup/full /backup/incr
```

Incremental pg_basebackup

- To restore, use *pg_combinebackup*

```
$ pg_combinebackup -o /backup/combined /backup/full /backup/incr
```

- Or combine a long chain if needed

```
$ pg_combinebackup -o /backup/combined /backup/full /backup/incr /backup/incr2
```

Preserve subscriptions across upgrades

- Preserves *full* subscription state
- pg_upgrade
- Upgrade without rebuilding subscribers

Slot synchronization

- Sync logical replication slots
    - Between phsyical replicas
- *failover* enabled on each slot
    - pg_create_logical_replication_slot()
    - CREATE SUBSCRIPTION
- Enable *sync_replication_slots* on standby
- Configure *standby_slot_names*

New features

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

Many infrastructure

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

COPY performance

- *uuid_out*
- COPY TO when encoding matches

Self-join removal

- Table joined to itself
- In cases where it's not necessary
- Not-uncommon ORM style query
- Replaced with just one scan

Redundant NOT NULL

```
postgres=# CREATE TABLE foo (a int NOT NULL);
CREATE TABLE

postgres=# INSERT INTO foo SELECT * FROM generate_series(1,1000);
INSERT 0 1000
```

Redundant NOT NULL

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NOT NULL;
                           QUERY PLAN
-------------------------------------------------------------
 Seq Scan on foo  (cost=0.00..159.75 rows=11418 width=4)
   Filter: (a IS NOT NULL)
(2 rows)
```

Redundant NOT NULL

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NOT NULL;
                         QUERY PLAN
-----------------------------------------------------------
 Seq Scan on foo  (cost=0.00..159.75 rows=11418 width=4)
   Filter: (a IS NOT NULL)
(2 rows)
```

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NOT NULL;
                         QUERY PLAN
-----------------------------------------------------------
 Seq Scan on foo  (cost=0.00..159.75 rows=11475 width=4)
(1 row)
```

Redundant NOT NULL

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NULL;
                           QUERY PLAN
---------------------------------------------------------------
 Seq Scan on foo  (cost=0.00..159.75 rows=57 width=4)
    Filter: (a IS NULL)
(2 rows)
```

Redundant NOT NULL

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NULL;
                         QUERY PLAN
-----------------------------------------------------------
 Seq Scan on foo  (cost=0.00..159.75 rows=57 width=4)
    Filter: (a IS NULL)
(2 rows)
```

```
postgres=# EXPLAIN SELECT * FROM foo WHERE a IS NULL;
                   QUERY PLAN
------------------------------------------------------
 Result  (cost=0.00..0.00 rows=0 width=0)
    One-Time Filter: false
(2 rows)
```

Parallelism

- CREATE INDEX for BRIN

SLRU caches

- Divide cache i banks
- Separate locking
- Configure each size independently
  - *xxxx_buffers*
- pg_stat_slru

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

@magnushagander

https://www.hagander.net/talks/