A look at the Elephants Trunk

PostgreSQL 17

PGDay Chicago 2024
Chicago, IL, 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 2024 - CF5
Current status

- 2061 commits
- 3623 files changed, 210344 insertions(+), 89974 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
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
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`

```sql
code
postgres=# SELECT * FROM pg_wait_events WHERE name='PgSleep';

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>PgSleep</td>
<td>Waiting due to a call to <code>pg_sleep</code> or a sibling function</td>
</tr>
</tbody>
</table>

(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
```
postgres=# select * from pg_stat_checkpointer ;
-
<table>
<thead>
<tr>
<th>RECORD 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_timed</td>
</tr>
<tr>
<td>numRequested</td>
</tr>
<tr>
<td>restartpoints_timed</td>
</tr>
<tr>
<td>restartpoints_req</td>
</tr>
<tr>
<td>restartpoints_done</td>
</tr>
<tr>
<td>write_time</td>
</tr>
<tr>
<td>sync_time</td>
</tr>
<tr>
<td>buffers_written</td>
</tr>
</tbody>
</table>
```
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

<table>
<thead>
<tr>
<th>queryid</th>
<th>1774110370767368945</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>call dummyproc($1,$2)</td>
</tr>
</tbody>
</table>
pg_stat_vacuum_progress

- Shows index progress

<table>
<thead>
<tr>
<th>phase</th>
<th>vacuuming indexes</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexes_total</td>
<td>5</td>
</tr>
<tr>
<td>indexes_processed</td>
<td>3</td>
</tr>
</tbody>
</table>
COPY
dummy FROM '/tmp/test.csv' WITH (FORMAT csv);

ERROR: invalid input syntax for type integer: "foo"
CONTEXT: COPY dummy, line 2, column b: "foo"
- 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 integer:
"foo"
CONTEXT:  COPY dummy, line 2, column b: "foo"

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" and "C.UTF-8"
- Faster!
- Stable!
Direct TLS handshake

- Without negotiation
  - Saves a roundtrip
  - Friendlier to proxies
- Always with ALPN
- sslnegotiation=direct
Split and merge partitions

- Change partition layout
- Fairly naive implementation
  - (for now)

```sql
ALTER TABLE t
SPLIT PARTITION t_y2020 INTO (
  PARTITION t_y2020_h1 FOR VALUES FROM ('2020-01-01') TO ('2020-07-01'),
  PARTITION t_y2020_h2 FOR VALUES FROM ('2020-07-01') TO ('2021-01-01')
)
```
Split and merge partitions

- Change partition layout
- Fairly naive implementation
  - (for now)

```
ALTER TABLE t
MERGE PARTITIONS (
  t_y2020_h1, t_y2020_h2
)
INTO t_y2020
```
New features

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

- New libpq function
- Use to.... Change passwords!
- Used to be psql-only
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)
CREATE TABLE test (
    id int,
    valid tstzrange,
    CONSTRAINT pk_test PRIMARY KEY (id, valid WITHOUT OVERLAPS),
    CONSTRAINT fk_test FOREIGN KEY (id, PERIOD valid)
        REFERENCES test2(id, PERIOD valid)
)
MERGE

• WHEN NOT MATCHED BY SOURCE
MERGE

- WHEN NOT MATCHED BY SOURCE

MERGE INTO t1
USING t2 ON t1.id=t2.id
WHEN MATCHED THEN
  UPDATE SET something=true
WHEN NOT MATCHED THEN
  INSERT (id, something) VALUES (t2.id, true)
WHEN NOT MATCHED BY SOURCE THEN
  DELETE
MERGE INTO t1
  USING t2 ON t1.id=t2.id
  WHEN MATCHED THEN
    UPDATE SET something=true
  WHEN NOT MATCHED THEN
    INSERT (id, something) VALUES (t2.id, true)
RETURNING merge_action(), t1.*
Many new operators
Convert between "data types"
E.g. `.string()` and `.boolean()`
SQL/JSON functions

- New functions from the standard
- JSON_EXISTS()
- JSON_QUERY()
- JSON_VALUE()
JSON_TABLE

- Convert JSON to relational
- Like XMLTABLE
- Single value to multiple columns
- In one pass
```sql
SELECT jt.* FROM my_films,
JSON_TABLE (js, '$.favorites[*]' COLUMNS (
  id FOR ORDINALITY,
  kind text PATH '$.kind',
  title text PATH '$.films[*].title' WITH WRAPPER,
  director text PATH '$.films[*].director' WITH WRAPPER)) AS jt;
```

<table>
<thead>
<tr>
<th>id</th>
<th>kind</th>
<th>title</th>
<th>director</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>comedy</td>
<td>[&quot;Bananas&quot;, &quot;The Dinner Game&quot;]</td>
<td>[&quot;Woody Allen&quot;, &quot;Francis Veber&quot;]</td>
</tr>
<tr>
<td>2</td>
<td>horror</td>
<td>[&quot;Psycho&quot;]</td>
<td>[&quot;Alfred Hitchcock&quot;]</td>
</tr>
<tr>
<td>3</td>
<td>thriller</td>
<td>[&quot;Vertigo&quot;]</td>
<td>[&quot;Alfred Hitchcock&quot;]</td>
</tr>
<tr>
<td>4</td>
<td>drama</td>
<td>[&quot;Yojimbo&quot;]</td>
<td>[&quot;Akira Kurosawa&quot;]</td>
</tr>
</tbody>
</table>
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`

```bash
$ pg_combinebackup -o /backup/combined /backup/full /backup/incremental
```
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 physical replicas
- `failover` enabled on each slot
  - `pg_create_logical_replication_slot()`
  - CREATE SUBSCRIPTION
- Enable `sync_replication_slots` on standby
- Configure `standby_slot_names`
pg_createsubscriber

- New commandline tool
- Convert physical to logical
- Much faster initial build!
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 into 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/