What's New for Developer's in PostgreSQL 17

Deepak Mahto
DataCloudGaze Consulting

About Me

I am Deepak Mahto, and I like to call myself your Database Guy.

- Founder at DataCloudGaze Consulting.
- Conduct training on PostgreSQL and Migration at DatabaseGyaan.
- Loves to blog on Databases Learnings.
- I live in Mumbai and have a 3-year-old child.
- Loves to explore street food.

Agenda

- Key upcoming features in PostgreSQL 17 for Developers.
    - Optimizer
    - Utility Command
    - Partitions
    - Merge
    - PL/pgSQL
    - Partitions
    - JSON
    - Trigger
    - Privilege
    - And more..

PostgreSQL 17 - Optimizer Sections

Optimizer - Null Constraints and Performance.

Improve optimization of IS NOT NULL and IS NULL query restrictions.

```
pg17=# alter table t1 alter column col1 set not null;
ALTER TABLE

pg17=# \d t1
                    Table "public.t1"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 col1   | integer |           | not null |
 col2   | integer |           |          |
 col3   | numeric |           |          |
 col4   | text    |           |          |
```

Optimizer - Null Constraints and Performance.

Improve optimization of IS NOT NULL and IS NULL query restrictions.

PG 17 – IS NULL Filter

```
pg17=# explain (analyze, buffers,
costs off) select count(1) from t1
where col1 is null;

Aggregate (actual
time=0.003..0.004 rows=1 loops=1)
-> Result (actual
time=0.001..0.001 rows=0 loops=1)
One-Time Filter: false
Planning Time: 0.040 ms
Execution Time: 0.021 ms
(5 rows)
```

PG 17 – IS Not NULL Filter

```
pg17=# explain (analyze, buffers,
costs off) select count(1) from t1
where col1 is not null;

Aggregate (actual
time=0.032..0.033 rows=1 loops=1)
Buffers: shared hit=1
-> Seq Scan on t1 (actual
time=0.011..0.020 rows=100
loops=1)
Buffers: shared hit=1
Planning Time: 0.046 ms
Execution Time: 0.051 ms
(6 rows)
```

no Filter Clause is specified

Optimizer - enable_group_by_reordering

Allow GROUP BY columns to be internally ordered for better plan.

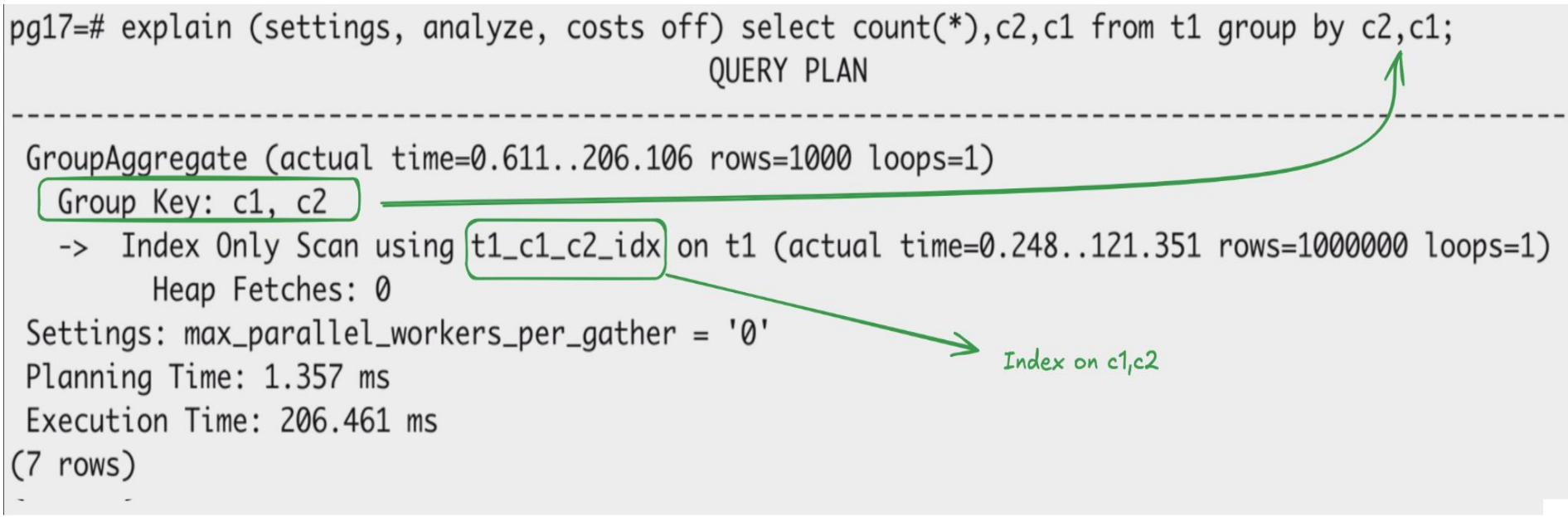
```
pg17=# set enable_group_by_reordering to off;
SET
pg17=# explain (settings, analyze, costs off) select count(*),c2,c1 from t1 group by c2,c1;
                                    QUERY PLAN
-------------------------------------------------------------------------------------
 HashAggregate (actual time=325.221..325.876 rows=1000 loops=1)
   Group Key: c2, c1
   Batches: 1  Memory Usage: 3153kB
   ->  Seq Scan on t1 (actual time=0.196..167.200 rows=1000000 loops=1)
 Settings: enable_group_by_reordering = 'off', max_parallel_workers_per_gather = '0'
 Planning Time: 1.499 ms
 Execution Time: 328.934 ms
(7 rows)
```

New Flag in PostgreSQL 17, by default it is enable.

Optimizer - enable_group_by_reordering

Allow GROUP BY columns to be internally ordered for better plan.

```
pg17=# explain (settings, analyze, costs off) select count(*),c2,c1 from t1 group by c2,c1;
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------------
 GroupAggregate (actual time=0.611..206.106 rows=1000 loops=1)
   Group Key: c1, c2
   ->  Index Only Scan using t1_c1_c2_idx on t1 (actual time=0.248..121.351 rows=1000000 loops=1)
         Heap Fetches: 0
 Settings: max_parallel_workers_per_gather = '0'
 Planning Time: 1.357 ms
 Execution Time: 206.461 ms
(7 rows)
```



Optimizer - Correlated IN Clause Transformation.

Allow correlated IN subqueries to be transformed into optimal joins

```
explain (analyze, costs off)
select * from test_in_correlated1 t1 where t1.col1 in (select t2.col1*99999
from test_in_correlated2 t2 where t1.col2= t2.col2);
```

PG16

```
                                   QUERY PLAN
-------------------------------------------------------------------------------------
 Seq Scan on test_in_correlated1 t1 (actual time=4546.820..4546.829 rows=0 loops=1)
   Filter: (SubPlan 1)
   Rows Removed by Filter: 10000
   SubPlan 1
     ->  Seq Scan on test_in_correlated2 t2 (actual time=0.023..0.451 rows=10 loops=10000)
           Filter: (t1.col2 = col2)
           Rows Removed by Filter: 9990
 Planning Time: 0.216 ms
 Execution Time: 4546.909 ms
(9 rows)
```

SubPlan is looped 10k times to resolve correlated filter

Optimizer - Correlated IN Clause Transformation.

Allow correlated IN subqueries to be transformed into optimal joins

```
explain (analyze, costs off)
select * from test_in_correlated1 t1 where t1.col1 in (select t2.col1*99999
from test_in_correlated2 t2 where t1.col2= t2.col2);

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
Hash Join (actual time=31.352..31.355 rows=0 loops=1)
  Hash Cond: ((t2.col2 = t1.col2) AND ((t2.col1 * 99999) = t1.col1))
  ->  HashAggregate (actual time=12.418..13.889 rows=10000 loops=1)
        Group Key: t2.col2, (t2.col1 * 99999)
        Batches: 1  Memory Usage: 1169kB
        ->  Seq Scan on test_in_correlated2 t2 (actual time=0.712..8.530 rows=10000 loops=1)
  ->  Hash (actual time=15.098..15.098 rows=10000 loops=1)
        Buckets: 16384  Batches: 1  Memory Usage: 645kB
        ->  Seq Scan on test_in_correlated1 t1 (actual time=1.619..13.035 rows=10000 loops=1)
Planning Time: 7.713 ms
Execution Time: 32.389 ms
(11 rows)
```

Correlated IN Clause is Transform to Hash Join

Optimizer - Improve CTE Plans

Use CTE columns statistics and sort order for better plan and estimates.

PG16

```
pg16=# explain with x as materialized (select c3 from t1 b)
pg16-# select count(1) from t1 a where c3 in (select * from x);
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=62754.00..62754.01 rows=1 width=8)
   CTE x
     ->  Seq Scan on t1 b  (cost=0.00..15406.00 rows=1000000 width=4)
   ->  Hash Join  (cost=22504.50..46098.00 rows=500000 width=0)
         Hash Cond: (a.c3 = x.c3)
         ->  Seq Scan on t1 a  (cost=0.00..15406.00 rows=1000000 width=4)
         ->  Hash  (cost=22502.00..22502.00 rows=200 width=4)
               ->  HashAggregate  (cost=22500.00..22502.00 rows=200 width=4)
                     Group Key: x.c3
                     ->  CTE Scan on x  (cost=0.00..20000.00 rows=1000000 width=4)
(10 rows)
```

table t1 and same column c3 is use within CTE and outside

estimated 1000k rows

500k is estimated.

c3 column stats is not preserve outside of CTE.

Optimizer - Improve CTE Plans

Use CTE columns statistics and sort order for better plan and estimates.

```
pg17=# explain with x as materialized (select c3 from t1 b)
select count(1) from t1 a where c3 in (select * from x);
                                  QUERY PLAN
------------------------------------------------------------------------------
 Aggregate  (cost=95190.00..95190.01 rows=1 width=8)
   CTE x
     ->  Seq Scan on t1 b  (cost=0.00..15406.00 rows=1000000 width=4)
   ->  Hash Semi Join  (cost=36407.00..77284.00 rows=1000000 width=0)
         Hash Cond: (a.c3 = x.c3)
         ->  Seq Scan on t1 a  (cost=0.00..15406.00 rows=1000000 width=4)
         ->  Hash  (cost=20000.00..20000.00 rows=1000000 width=4)
               ->  CTE Scan on x  (cost=0.00..20000.00 rows=1000000 width=4)
(8 rows)
```

c3 column stats is used outside CTE

c3 stats outside CTE help's to choose better plan

PostgreSQL 17 - PL/pgSQL

DATACLOUDGAZE

PL/pgSQL - New Array Declaration

Allow plpgsql %TYPE and %ROWTYPE specifications to represent arrays of non array type.

```
pg17=# do language plpgsql
$$
<<block1>>
declare
var1 testplpgsql17.col1%type[];
var2 testplpgsql17%rowtype[];

begin
SELECT array_agg(col1),array_agg(tbl.*)
into var1, var2
FROM (select * from testplpgsql17 limit 2) as tbl ;

raise notice 'PL\pgSQL PG 17 - type[] - %',var1;
raise notice 'PL\pgSQL PG 17 - rowtype[] - %',var2;

end block1;
$$;
NOTICE:  PL\pgSQL PG 17 - type[] - {1,2}
NOTICE:  PL\pgSQL PG 17 - rowtype[] - {"(1,1,9.9)","(2,2,19.8)"}
DO
```

Declare Array using %type[] and %rowtype[] for non array type.

PostgreSQL 17 - Utility Command

DATACLOUDGAZE

Utility Command - Explain (Memory)

Allow EXPLAIN to report optimizer memory usage

```
   ->  Seq Scan on pgbench_accounts_3499 (actual time=0.010..0.010 rows=0 loops=1)
         Filter: (bid = 1)
   ->  Seq Scan on pgbench_accounts_3500 (actual time=0.004..0.004 rows=0 loops=1)
         Filter: (bid = 1)
 Planning:
   Memory: used=34007kB  allocated=41819kB
 Planning Time: 623.421 ms
 Execution Time: 313.460 ms
(8730 rows)
```

3.5k partition scan

With large set of partitions to scan planning operation memory need is increases.

```
explain (analyze, costs off, memory) select * from pgbench_accounts where bid =1;

Completing running Execution plan for many partitions scans , check Planning Memory usage.


explain (analyze, costs off, memory) select * from pgbench_accounts where aid =1;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Seq Scan on pgbench_accounts_1 pgbench_accounts (actual time=0.029..0.040 rows=1 loops=1)
   Filter: (aid = 1)
   Rows Removed by Filter: 57
 Planning:
   Memory: used=56kB  allocated=68kB
 Planning Time: 1.081 ms
 Execution Time: 0.075 ms
(7 rows)

Completing running Execution plan for few partitions scans, check Planning Memory usage.
```

Planning memory reduces with lower partitions on scans.

Utility Command - Explain (serialize)

Allow Explain command to show overhead of convert and de-toasting and sending it over the wire.

```
                                Table "public.testtoast"
 Column |  Type   | Collation | Nullable |                  Default
--------+---------+-----------+----------+-------------------------------------------
 col1   | bigint |           | not null | nextval('testtoast_col1_seq'::regclass)
 col2   | text   |           |          |

 table_schema | table_name | table_toast | table |  index   | total
--------------+------------+-------------+-------+----------+-------
 public       | testtoast  | 26 MB       | 96 kB | 0 bytes | 26 MB
```

```
Running - explain (analyze, buffers) select * from testtoast
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Seq Scan on testtoast  (cost=0.00..14.00 rows=600 width=72) (actual time=1.297..2.838 rows=600 loops=1)
   Buffers: shared hit=8
 Planning:
   Buffers: shared hit=8
 Planning Time: 0.229 ms
 Execution Time: 2.928 ms
(6 rows)
```

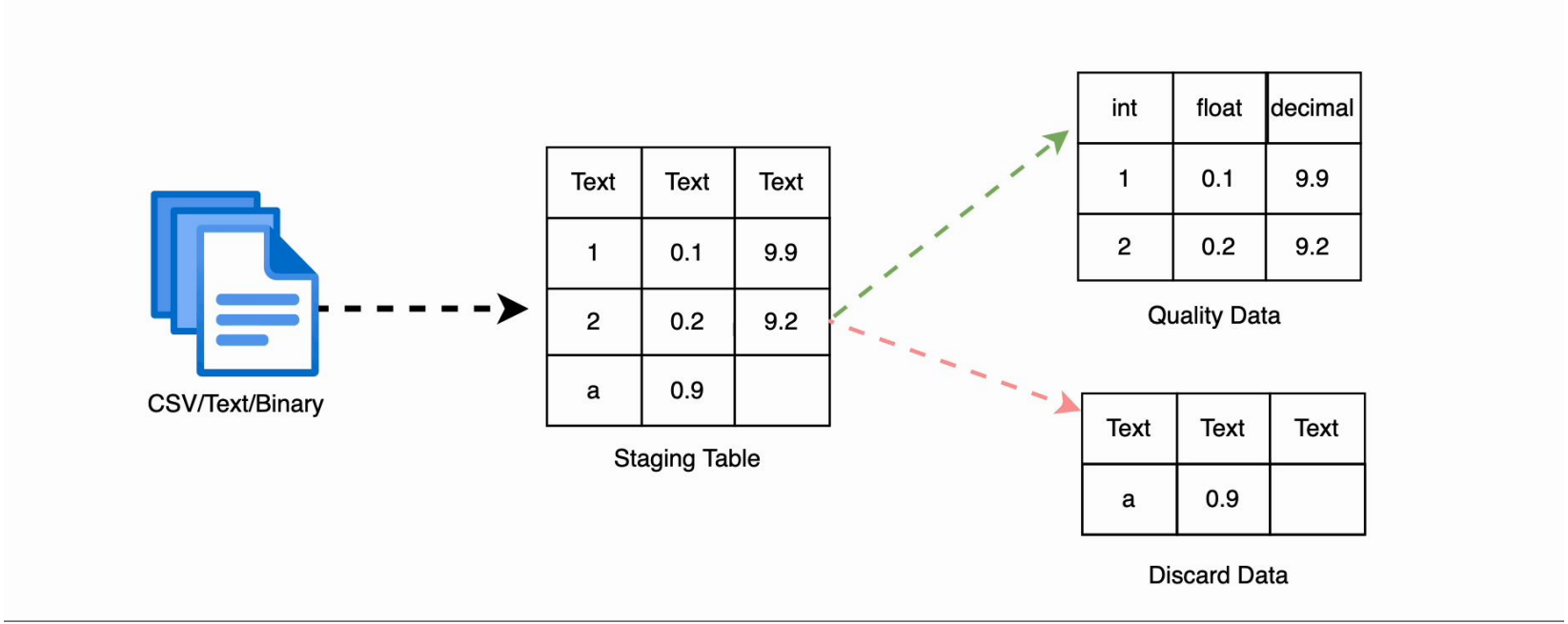
With our serialize, we can't measure overhead of serializing toasted columns over the wire.

```
Running - explain (analyze, buffers,serialize) select * from testtoast
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Seq Scan on testtoast  (cost=0.00..14.00 rows=600 width=72) (actual time=0.025..0.436 rows=600 loops=1)
   Buffers: shared hit=8
 Planning Time: 0.101 ms
 Serialization: time=1475.352 ms  output=2225399kB  format=text
   Buffers: shared hit=1829 read=3084
 Execution Time: 1476.359 ms
(6 rows)
```

DATACLOUDGAZE

Utility Command - Copy

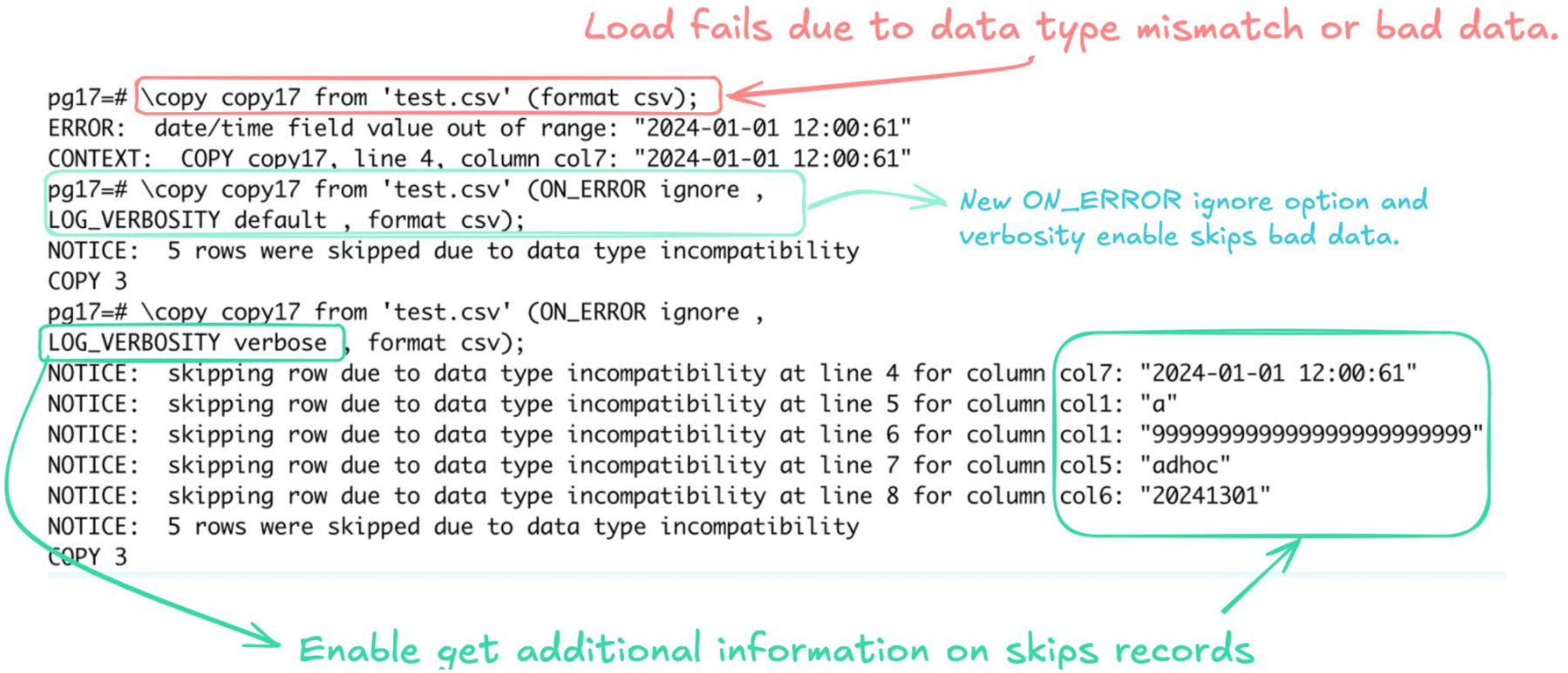
New option to ignore Bad record and Track it with Verbosity option.



Utility Command - Copy

- Handling data type incompatibilities with ON_ERROR options
- Improved logging with LOG_VERBOSITY
- Observability via pg_stat_progress_copy with tuples_skipped

Utility Command - Copy command is more user friendly



Utility Command - tuples_skipped

```
1 | select command, tuples_processed , tuples_excluded ,
2 | tuples_skipped from pg_stat_progress_copy;
```

```
-[ RECORD 1 ]----+----------
command          | COPY FROM
tuples_processed | 3
tuples_excluded  | 0
tuples_skipped   | 5
```

PostgreSQL 17 - New JSON Functions.

DATACLOUDGAZE

SQL/JSON Function - JSON_TABLE

JSON_TABLE() to convert JSON data to a table representation

```
pg17=# SELECT jt.* FROM
 my_films,
 JSON_TABLE ( js, '$.favorites[*]'
   COLUMNS (
    id FOR ORDINALITY,
    kind text PATH '$.kind',
   NESTED PATH '$.films[*]' COLUMNS (
      title text FORMAT JSON PATH '$.title' OMIT QUOTES,
      director text PATH '$.director' KEEP QUOTES))) AS jt;
 id |   kind   |       title        |      director
----+----------+--------------------+--------------------
  1 | comedy   | Nuvvu Naaku Nachav | "K. Vijaya Bhaskar"
  1 | comedy   | Bangalore Days     | "Anjali Menon"
  2 | horror   | Pizza              | "Karthik Subbaraj"
  3 | thriller | Kalki              | "Prasanth Varma"
  4 | drama    | Bahubali           | "S. S. Rajamouli"
  4 | drama    | Salaar             | "Prashanth Neel"
(6 rows)
```

Allow to Query Json as Table in PostgreSQL

SQL/JSON Constructor Functions

```
pg17=# select  json_scalar(random(1,10)), json_scalar(now()::timestamp without time zone);
 json_scalar |          json_scalar
-------------+-------------------------------
 2           | "2024-08-30T00:27:22.058111"
```

```
pg17=# select JSON_SERIALIZE('{"kind": "drama","films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}' RETURNING text);
                                json_serialize
-------------------------------------------------------------------------------
 {"kind": "drama","films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}
(1 row)

pg17=# select JSON_SERIALIZE('{"kind": "drama","films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}' RETURNING bytea);
                                                                json_serialize
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------
 \x7b226b696e64223a20226472616d61222c2266696c6d73223a205b7b227469746c65223a2022426168756269222c226469726563746f72223a2022532e2
0532e2052616a616d6f756c69227d5d7d
```

```
pg17=# select JSON('{"kind": ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}');
ERROR:  invalid input syntax for type json
DETAIL:  Expected JSON value, but found ",".
CONTEXT:  JSON data, line 1: {"kind": ,...
pg17=# select JSON('{"kind": "drama","films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}');
                                  json
-------------------------------------------------------------------------------
 {"kind": "drama","films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}
```

SQL/JSON query functions

```
pg17=# SELECT JSON_EXISTS('{"kind":"drama" ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}', '$.films[0]');
 json_exists
-------------
 t
(1 row)

pg17=# SELECT JSON_EXISTS('{"kind":"drama" ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}', '$.films[1]');
 json_exists
-------------
 f
(1 row)

pg17=# SELECT JSON_QUERY('{"kind":"drama" ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}', '$.films[*].title');
 json_query
------------
 "Bahubali"
(1 row)

pg17=# SELECT JSON_VALUE( {"kind":"drama" ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}', '$.kind');
 json_value
------------
 drama
(1 row)

pg17=# SELECT JSON_VALUE('{"kind":"drama" ,"films": [{"title": "Bahubali","director": "S. S. Rajamouli"}]}', '$.films[0].title');
 json_value
------------
 Bahubali
(1 row)
```

PostgreSQL 17 - Partitions.

DATACLOUDGAZE

Partition - Identity Column support

```
CREATE TABLE pg17_partition (
  id          int8 generated always AS IDENTITY ,
  created_at     timestamp without time zone NOT NULL,
  addinfo       TEXT
) PARTITION BY RANGE ( created_at ) ;
```

Idenity space is shared across Partitions.

```
CREATE TABLE pg17_partition_old PARTITION OF pg17_partition FOR VALUES FROM (MINVALUE) TO ('2022-01-01');
CREATE TABLE pg17_partition_2022 PARTITION OF pg17_partition FOR VALUES FROM ('2022-01-01') TO ('2023-01-01');
CREATE TABLE pg17_partition_2023 PARTITION OF pg17_partition FOR VALUES FROM ('2023-01-01') TO ('2024-01-01');
CREATE TABLE pg17_partition_2024 PARTITION OF pg17_partition FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');
CREATE TABLE pg17_partition_2025 PARTITION OF pg17_partition FOR VALUES FROM ('2025-01-01') TO ('2026-01-01');

INSERT INTO pg17_partition (created_at, addinfo)
SELECT (now() - '5 years'::INTERVAL * random())::timestamp without time zone, 'Add info - ' || i FROM generate_series(1,10) i;
```

Based on Insertion order Identity Space is Shared Across Partitions.

```
pg17=# select * from pg17_partition;
 id |         created_at         |    addinfo
----+----------------------------+---------------
  3 | 2021-07-08 20:22:36.767674 | Add info - 3
  9 | 2021-08-25 10:33:17.049274 | Add info - 9
 10 | 2020-07-11 21:02:40.761274 | Add info - 10
  1 | 2022-09-29 19:10:33.225274 | Add info - 1
  7 | 2022-01-16 08:46:41.116474 | Add info - 7
  8 | 2022-04-13 02:33:59.327674 | Add info - 8
  2 | 2023-08-11 17:00:42.278074 | Add info - 2
  5 | 2023-10-14 23:00:36.748474 | Add info - 5
  4 | 2024-05-14 20:11:12.911674 | Add info - 4
  6 | 2024-04-14 01:56:31.545274 | Add info - 6
(10 rows)
```

DATACLOUDGAZE

Partition - Split existing Partition Support

Reverted in 17 Beta 3

```
pg17=# ALTER TABLE pg17_partition
       SPLIT PARTITION pg17_partition
       (PARTITION pg17_partition_2024_01 FOR VALUES FROM ('2024-01-01') TO ('2024-04-01'),
       PARTITION pg17_partition_2024_02 FOR VALUES FROM ('2024-04-01') TO ('2024-07-01'),
    PARTITION pg17_partition_2024_03 FOR VALUES FROM ('2024-07-01') TO ('2024-10-01'),
    PARTITION pg17_partition_2024_04 FOR VALUES FROM ('2024-10-01') TO ('2025-01-01'));
ALTER TABLE
pg17=# \dt pg17_partition_2024*
                List of relations
 Schema |          Name          | Type  |  Owner
--------+------------------------+-------+----------
 public | pg17_partition_2024_01 | table | postgres
 public | pg17_partition_2024_02 | table | postgres
 public | pg17_partition_2024_03 | table | postgres
 public | pg17_partition_2024_04 | table | postgres
(4 rows)
```

Split existing partition to further small subsets.

Acquire access exclusive locks

Hash partition is not supported.

Partition - Merge existing Partition Support

Reverted in 17 Beta 3

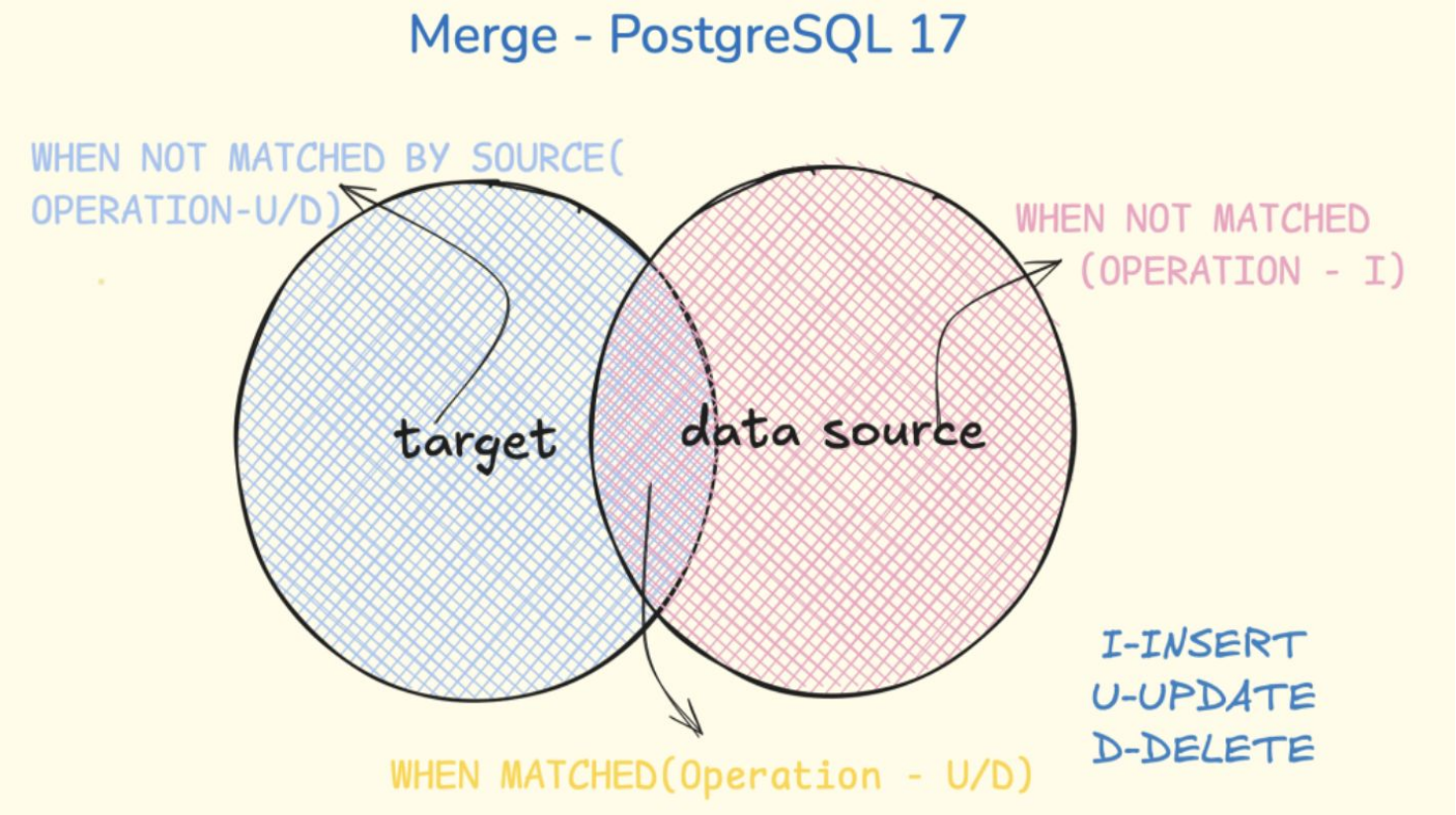
```
pg17=# ALTER TABLE pg17
       MERGE PARTITION
                 pg17_p
                 pg17_partition_2022 )
         INTO pg17_partition_old;
ALTER TABLE

pg17=# \dt pg17_partition*
                      List of relations
 Schema |          Name           |       Type        |  Owner
--------+-------------------------+-------------------+----------
 public | pg17_partition          | partitioned table | postgres
 public | pg17_partition_2023     | table             | postgres
 public | pg17_partition_2024_01  | table             | postgres
 public | pg17_partition_2024_02  | table             | postgres
 public | pg17_partition_2024_03  | table             | postgres
 public | pg17_partition_2024_04  | table             | postgres
 public | pg17_partition_2025     | table             | postgres
 public | pg17_partition_old      | table             | postgres
(8 rows)
```

Merge existing Partition

PostgreSQL 17 - Merge

DATACLOUDGAZE

Merge Command in PostgreSQL 17



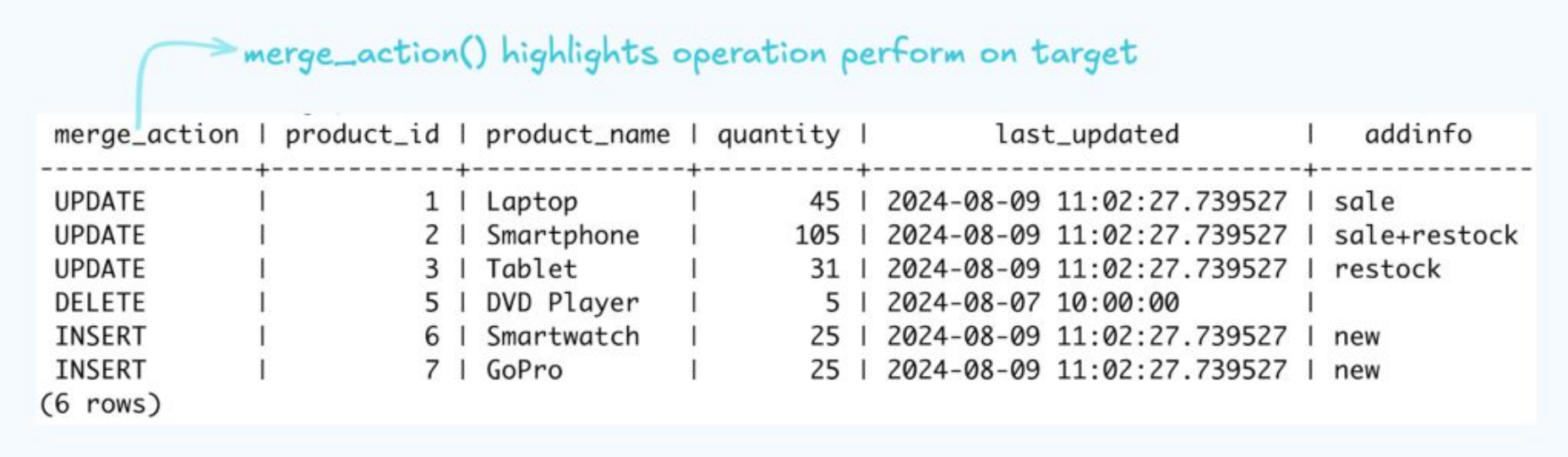
Merge Command in PostgreSQL 17

Fun Fact: Internal Join Strategies in PostgreSQL 17 MERGE

when_clause	join_method
all (matched, not_matched, not_mached_by_source)	*full join*
only matched	*inner join*
only not matched	*right join*
only not_mached_by_source	*left join*

Merge Command in PostgreSQL 17

New merge_action() Function: Highlights operations performed on the target table, now with added support for the RETURNING clause.




```
 merge_action | product_id | product_name | quantity |        last_updated        |   addinfo
--------------+------------+--------------+----------+----------------------------+-------------
 UPDATE       |          1 | Laptop       |       45 | 2024-08-09 11:02:27.739527 | sale
 UPDATE       |          2 | Smartphone   |      105 | 2024-08-09 11:02:27.739527 | sale+restock
 UPDATE       |          3 | Tablet       |       31 | 2024-08-09 11:02:27.739527 | restock
 DELETE       |          5 | DVD Player   |        5 | 2024-08-07 10:00:00        |
 INSERT       |          6 | Smartwatch   |       25 | 2024-08-09 11:02:27.739527 | new
 INSERT       |          7 | GoPro        |       25 | 2024-08-09 11:02:27.739527 | new
(6 rows)
```

PostgreSQL 17 - psql command line

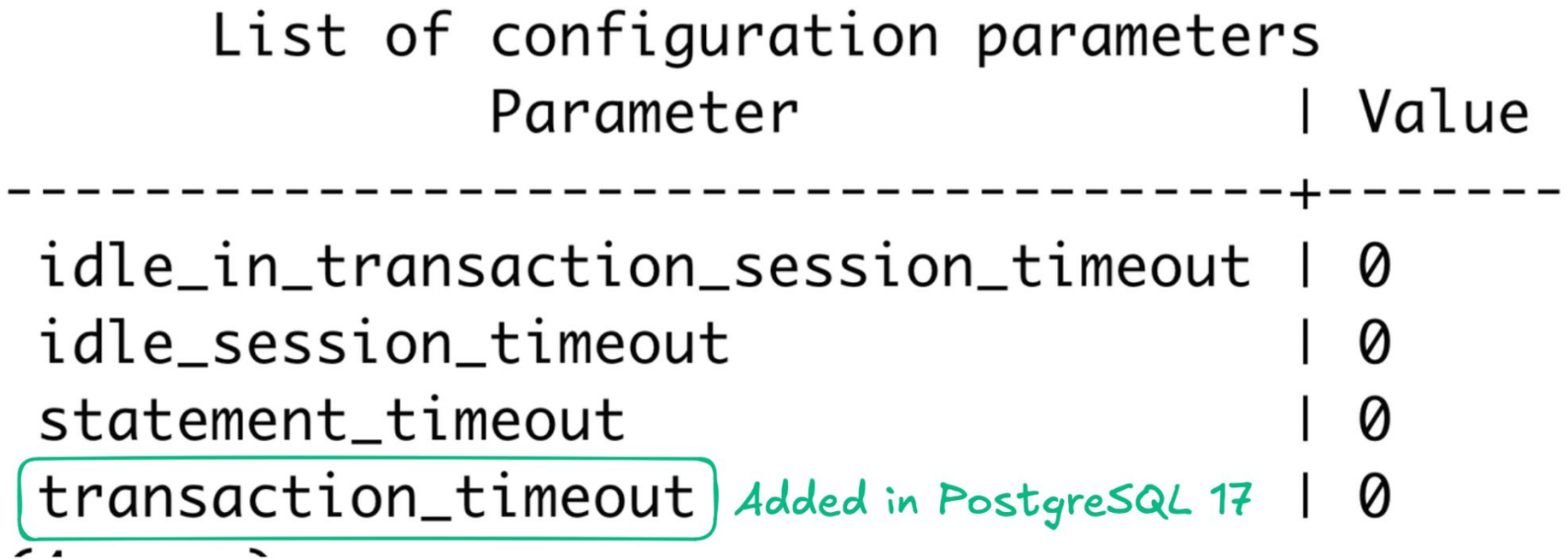
Allow psql's \watch to stop after a min number of rows returned

```
pg17=# select generate_series(1,random(1,10));
 generate_series
-----------------
               1
               2
               3
               4
               5
               6
               7
               8
               9
              10
(10 rows)

pg17=# \watch i=1 m=3
```

PostgreSQL 17 - Server Configuration

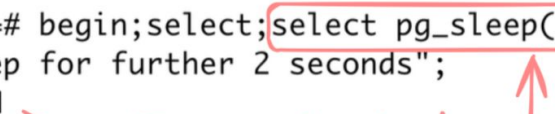
Terminating connection due to transaction_timeout

Set timeout for Transactions that can help to overcome vacuum and table bloat challenges.



Terminating connection due to transaction_timeout

```
pg17=# set transaction_timeout=3000;
SET
pg17=# begin;select;select pg_sleep(2) as "sleep for 2 sec";select;select pg_sleep(2) as
"sleep for further 2 seconds";
BEGIN
--
(1 row)

 sleep for 2 sec
-----------------

(1 row)

--
(1 row)

FATAL:  terminating connection due to transaction timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
```



Configurable SLRU cache

- Postgres 17 adds configurable SLRU cache sizes to boost performance in large workloads.
- Splitting SLRU caches into multiple banks with locks enhances scalability.
- Configuration changes require a restart, key for workloads facing contention

```
 List of configuration parameters
         Parameter          | Value
----------------------------+-------
 commit_timestamp_buffers   | 256kB
 multixact_member_buffers   | 256kB
 multixact_offset_buffers   | 128kB
 notify_buffers             | 128kB
 serializable_buffers       | 256kB
 subtransaction_buffers     | 256kB
 transaction_buffers        | 256kB
(7 rows)
```

PostgreSQL 17 - Privileges

DATACLOUDGAZE

Privileges - maintain / pg_maintain at user level.

Allow granting the right to perform maintenance operations

```
pg17=> set role myuser1;
SET
pg17=> select count(1) from t1;
  count
---------
 1000000
(1 row)

pg17=> vacuum t1;
WARNING:  permission denied to vacuum "t1", skipping it
VACUUM
pg17=> analyze t1;
WARNING:  permission denied to analyze "t1", skipping it
ANALYZE
pg17=> set role postgres;
SET
pg17=# grant maintain on t1 to myuser1;
GRANT
pg17=# set role myuser1;
SET
pg17=> vacuum analyze t1;
VACUUM
```

myuser1 can select on t1.

myuser1 cannot analyze or vacuum.

new "maintain" privilege
grant myuser1 to perform maintainence
related activity.

PostgreSQL 17 - Builtin Collation

DATACLOUDGAZE

New Builtin Collation

- Built-in collation provider with UTF-8 encoding
- Provides sorting semantics similar to 'C' collation
- Guaranteed immutable for consistent sort results across systems

```
pg17=# \dO pg_c_utf8|ucs_basic
                                                    |
   Schema   |   Name    | Provider | C
------------+-----------+----------+-
 pg_catalog | pg_c_utf8 | builtin  |
 pg_catalog | ucs_basic | builtin  |
(2 rows)
```

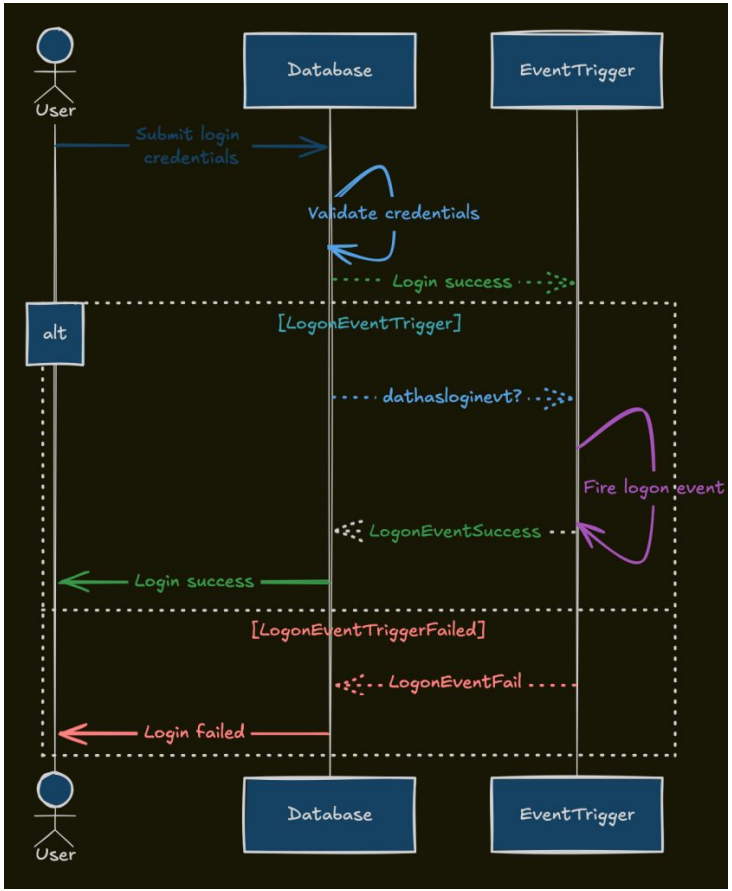
PostgreSQL 17 - Event Trigger on Logon

DATACLOUDGAZE

Event Triggers - Login

Implement login related usecases using login Event Trigger

```
CREATE OR REPLACE FUNCTION mandate_app_info()
RETURNS event_trigger AS $$
BEGIN
  IF nullif(current_setting('application_name'),'') is null then
    RAISE EXCEPTION 'Login not allowed for missing application_name'
USING HINT = 'Please set application_name context';
  END IF;
END;
$$ LANGUAGE plpgsql;

CREATE EVENT TRIGGER on_login_trigger ON login
EXECUTE PROCEDURE mandate_app_info();
ALTER EVENT TRIGGER on_login_trigger ENABLE ALWAYS;
```



Event Triggers - Login in action

```
asmitamahto@dcg pghyd2024 % docker exec -it pg17b3 psql -h localhost -U postgres -d 'dbname=pg17 application_name='
psql: error: connection to server at "localhost" (::1), port 5432 failed: FATAL:  Login not allowed for missing applica
tion_name
HINT:  Please set application_name context
CONTEXT:  PL/pgSQL function mandate_app_info() line 4 at RAISE
asmitamahto@dcg pghyd2024 % docker exec -it pg17b3 psql -h localhost -U postgres -d 'dbname=pg17 application_name=psql'

psql (17beta3 (Debian 17~beta3-1.pgdg120+1))
Type "help" for help.

pg17=# \dy
                          List of event triggers
       Name       | Event |  Owner   | Enabled |     Function     | Tags
------------------+-------+----------+---------+------------------+------
 on_login_trigger | login | postgres | always  | mandate_app_info |
(1 row)

pg17=# \dconfig event_triggers
List of configuration parameters
   Parameter    | Value
----------------+-------
 event_triggers | on
(1 row)
```

Connection allowed

configuration to control logon trigger

Getting Started and Test your self.

Running PostgreSQL 17 Beta within Docker.

docker run -d --name=pg17b3 -e POSTGRES_PASSWORD=********** postgres:17beta3 -p 5432:5435

```
CONTAINER ID   IMAGE              COMMAND                  CREATED      STATUS         PORTS      NAMES
f660743d3f2a   postgres:17beta3   "docker-entrypoint.s…"   6 days ago   Up 46 hours    5432/tcp   pg17b3
```

docker exec -it pg17b3 psql -h localhost -U postgres -d postgres

Thank you!

https://www.linkedin.com/in/mahtodeepak/

https://x.com/mahtodeepak05

https://databaserookies.wordpress.com/

https://www.datacloudgaze.com/

https://www.databasegyaan.com

Code : PGHYD30 , 30% off