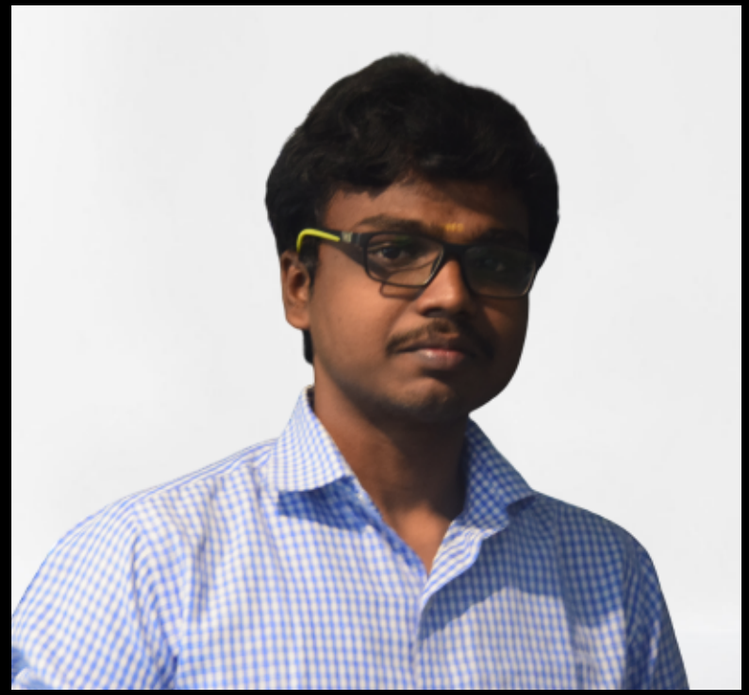
What is new in PostgreSQL 14

Aakash M,
Database Reliability Engineer,
Mydbops

October 23, 2021
Mydbops 10th MyWebinar

About Me

- Database Reliability Engineer
- Focusing on MySQL and PostgreSQL
- Expertise on performance tuning and troubleshooting
- Active blogger



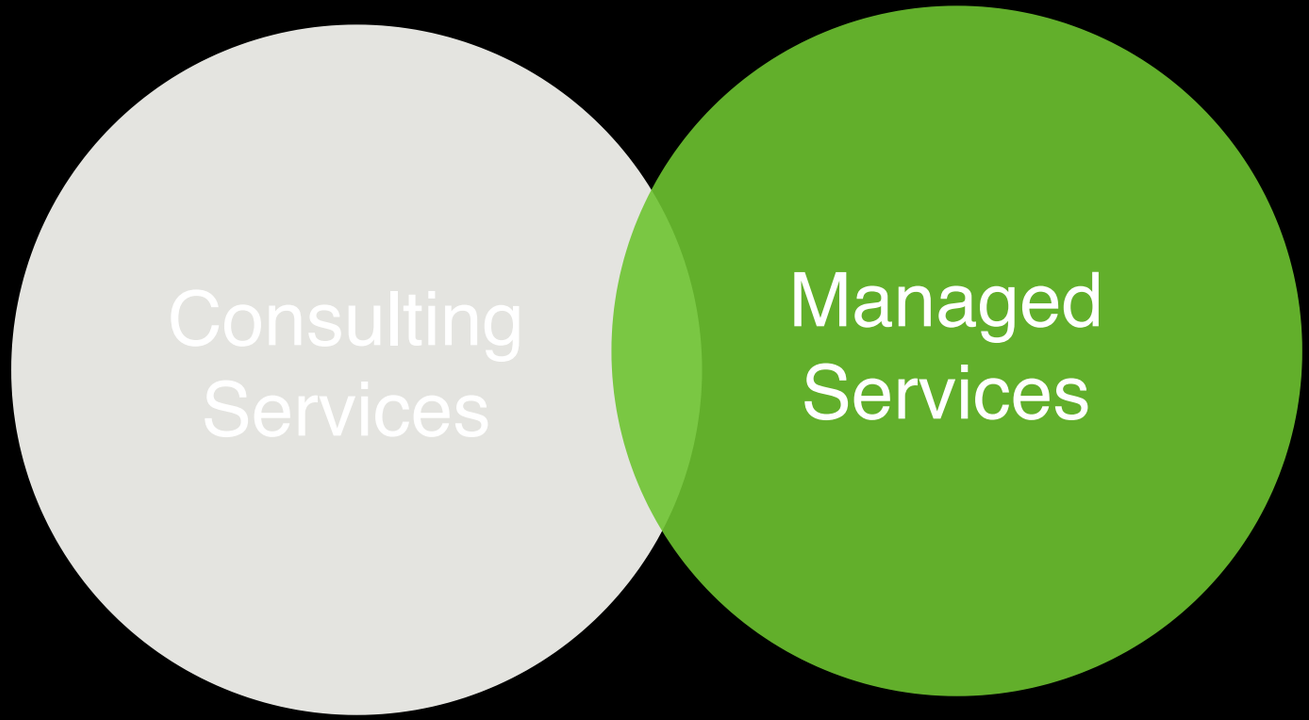
Mydbops

About Mydbops

- Services on top open source databases
- Founded in 2016
- 70 Members team
- Assisted over 500+ Customers
- AWS Partner and a PCI Certified Organisation

Mydbops

Mydbops Services



Focuses on Top Opensource database MySQL, MongoDB and PostgreSQL

Mydbops

Our Clients

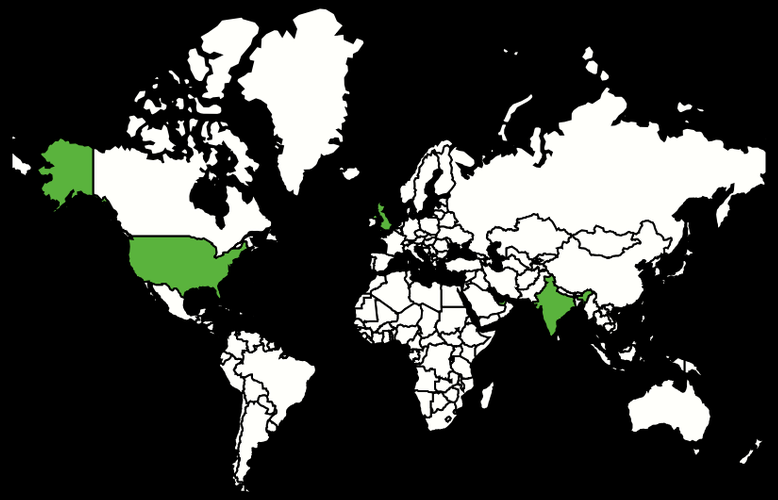
500 + Clients In 5 Yrs. of Operations

Flipkart

SWIGGY

yulu

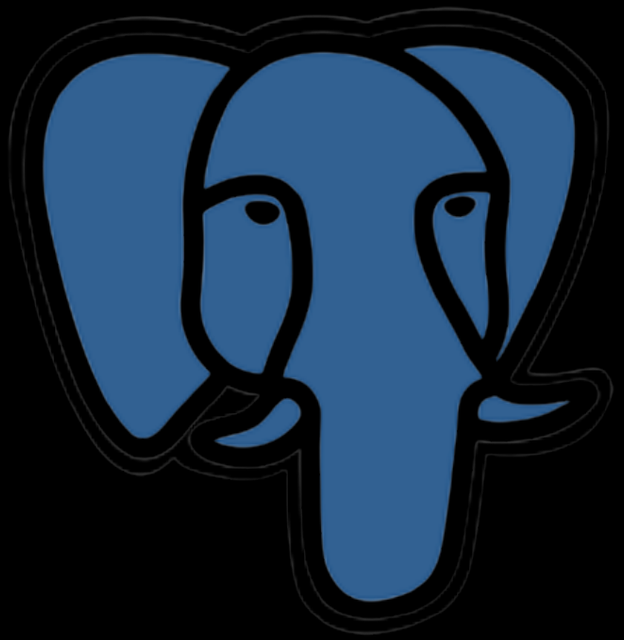
NYKAA



DREAM11

Mydbops

PostgreSQL 14



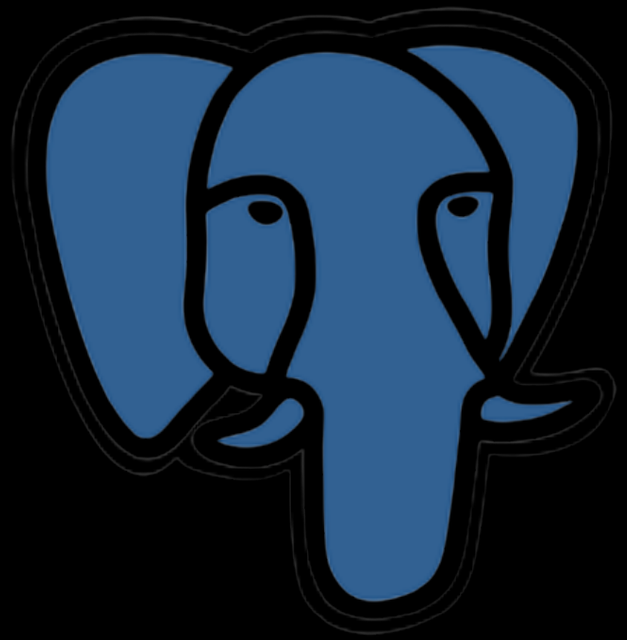
Mydbops

Overview

- Released on September 30, 2021
- With a lot of new features and bug fixed
- Every Quarter - Minor Releases
- Every Third Quarter - Major Release

Mydbops

New Features



Mydbops

PostgreSQL 14 Features

01 Server Side Enhancements

02 Improvements to Query Parallelism

03 Security Enhancements

04 Features for Application Developers

Mydbops

PostgreSQL 14 Features

05 Enhancements in Monitoring

06 Enhancements in pg_stat_statements

Mydbops

Server Side Enhancements

Mydbops

Reducing B Tree Index Bloat

BLOAT

- Dead tuples - Bloat
- Vacuum - Mark it as Reusable
- Index page scan becomes huge
- Unnecessary pages in RAM

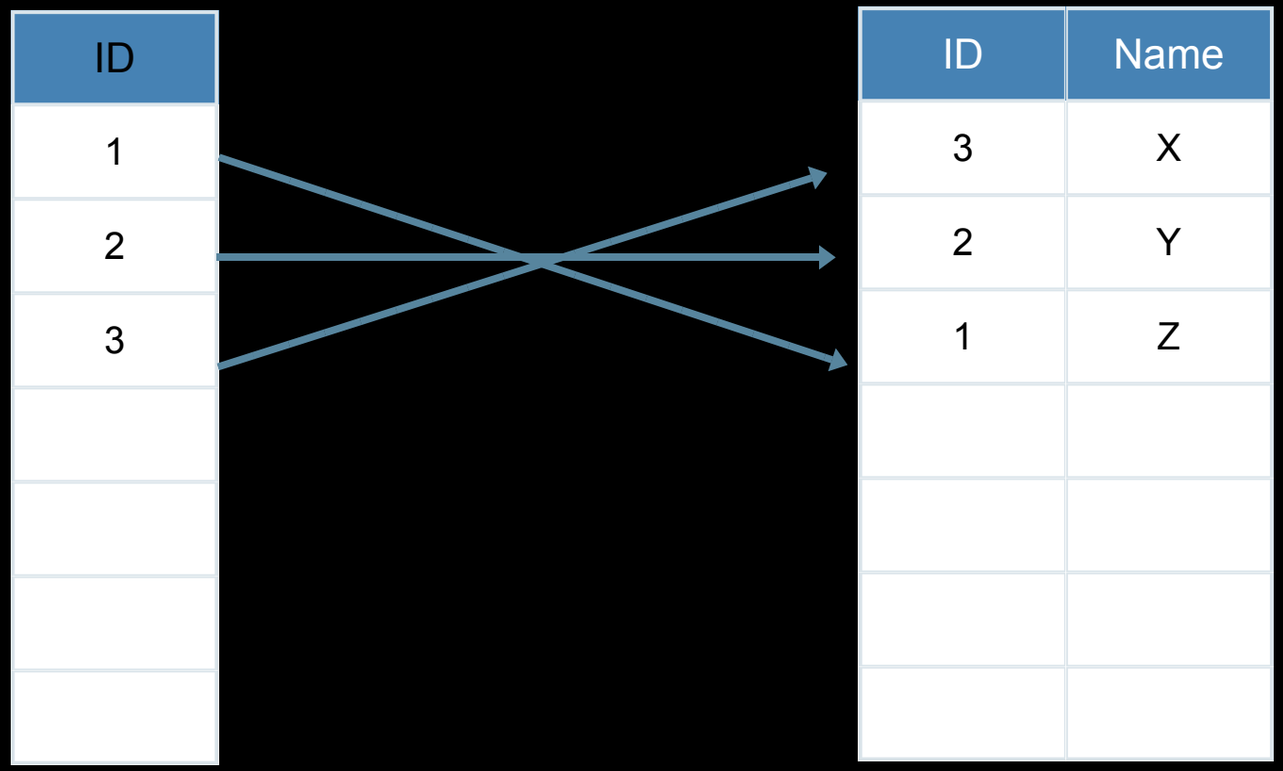
Mydbops

Heap Only Tuple

- Version 8.3
- No need to update index

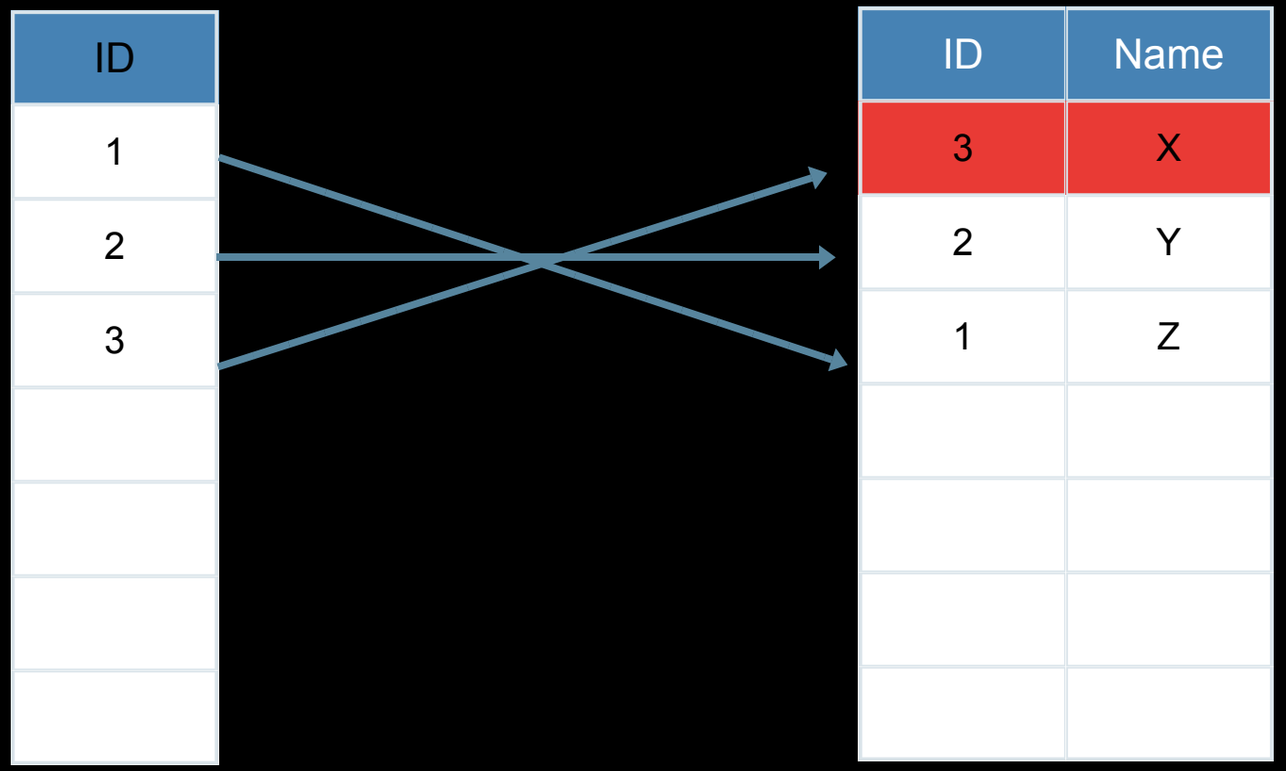
Mydbops

Heap Only Tuple



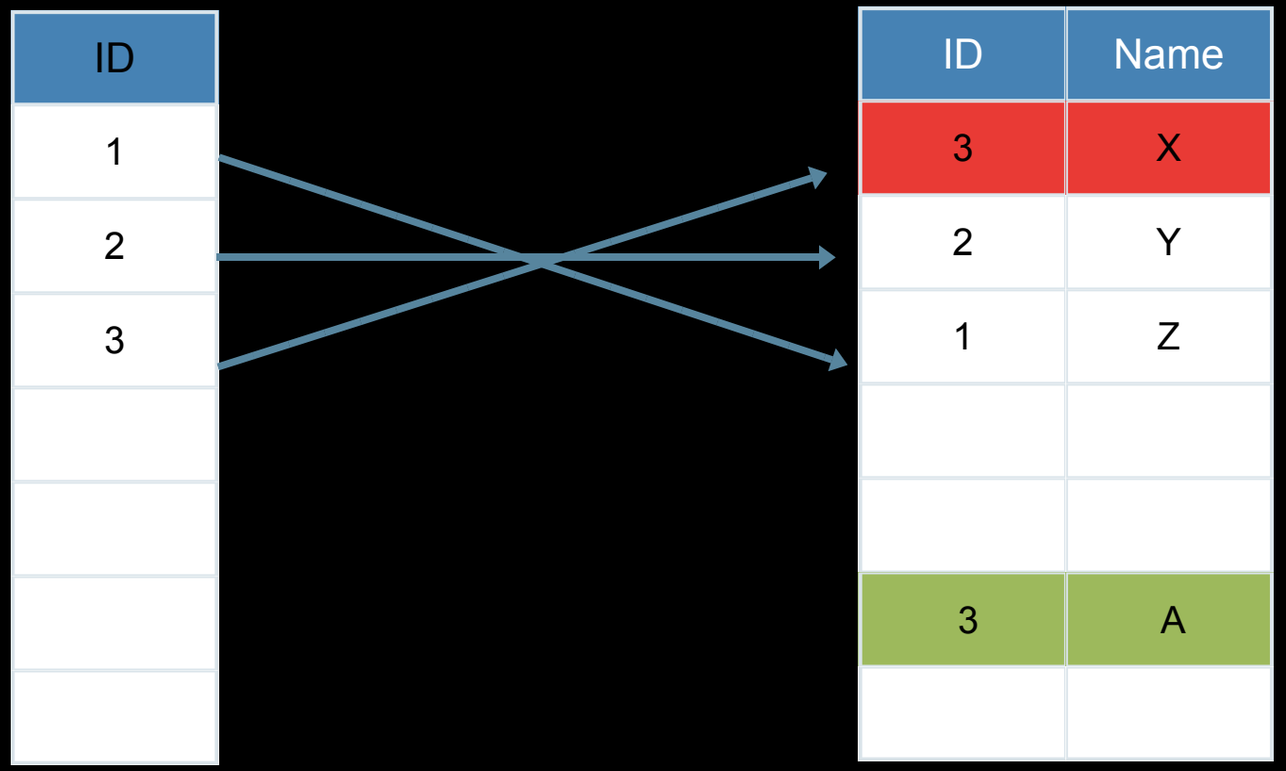
Mydbops

Heap Only Tuple



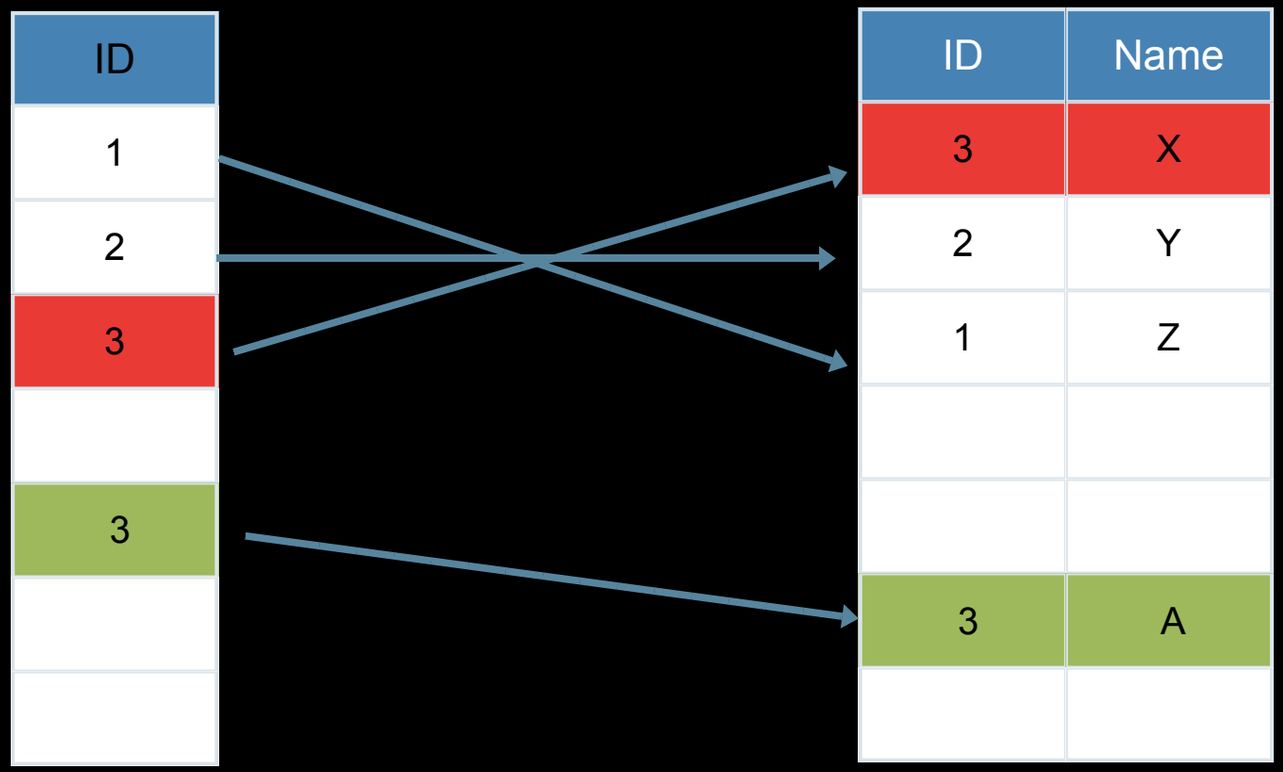
Mydbops

Heap Only Tuple



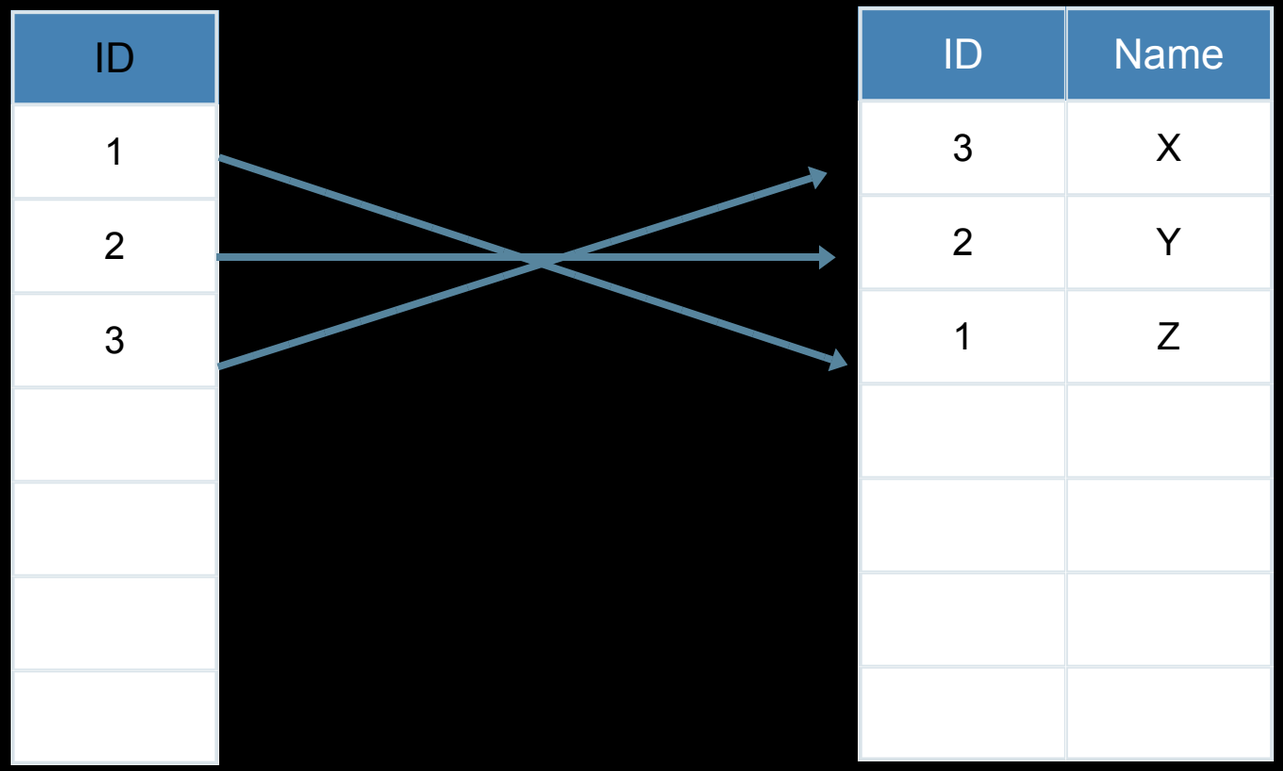
Mydbops

Heap Only Tuple



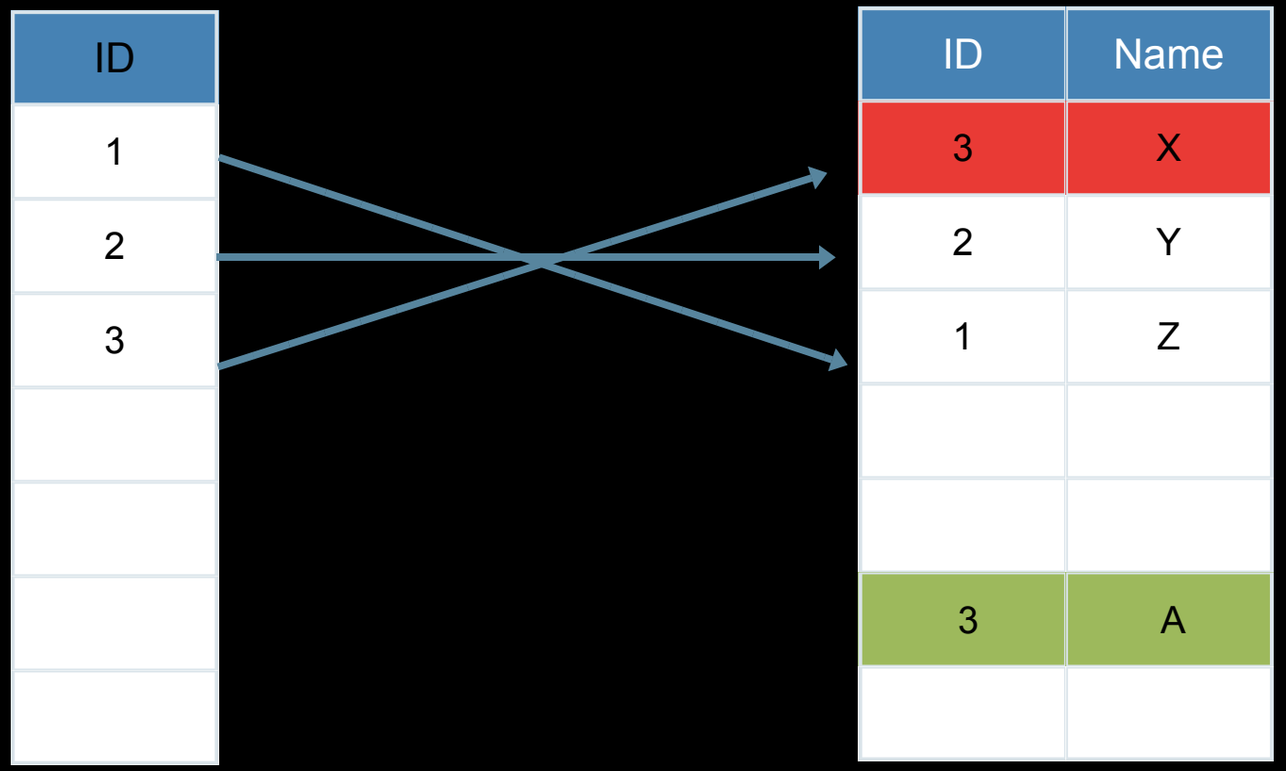
Mydbops

Heap Only Tuple



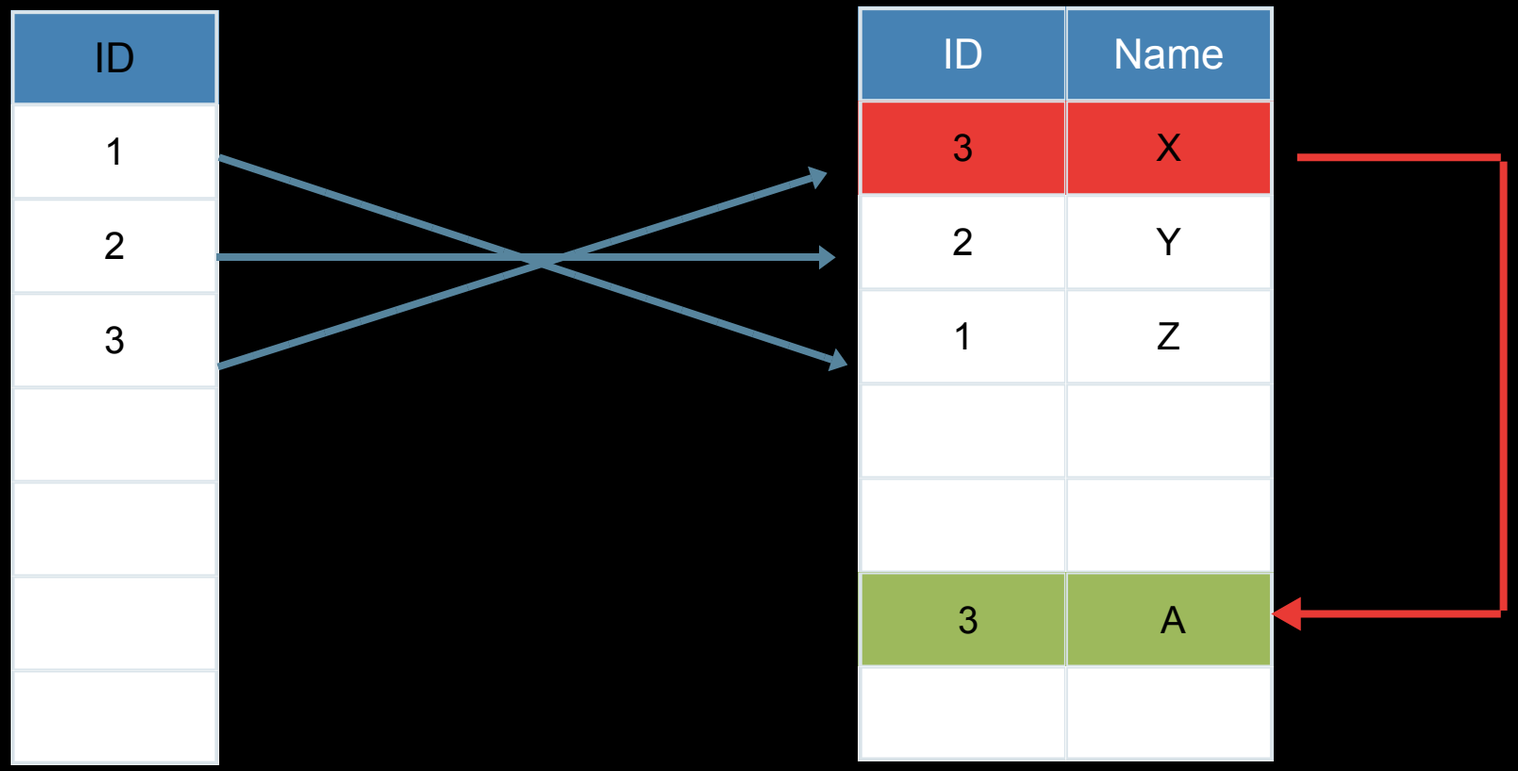
Mydbops

Heap Only Tuple



Mydbops

Heap Only Tuple



Mydbops

Killed Index Tuples

```
CREATE UNLOGGED TABLE test (
    id integer PRIMARY KEY,
    val text NOT NULL
) WITH (autovacuum_enabled = off);

INSERT INTO test SELECT i, 'text number ' || i FROM
generate_series(1, 1000000) AS i;


DELETE FROM test WHERE id BETWEEN 501 AND 799500;

Analyze test;
```

Mydbops

Killed Index Tuples

```
test=# EXPLAIN (ANALYZE, BUFFERS, COSTS OFF, TIMING OFF)
SELECT * FROM test WHERE id BETWEEN 1 AND 800000;
                          QUERY PLAN
-----------------------------------------------------------
-----
 Index Scan using test_pkey on test (actual rows=1000
loops=1)
   Index Cond: ((id >= 1) AND (id <= 800000))
   Buffers: shared hit=7284
 Planning:
   Buffers: shared hit=25
 Planning Time: 0.204 ms
 Execution Time: 95.223 ms
(7 rows)
```

Killed Index Tuples

```
test=# EXPLAIN (ANALYZE, BUFFERS, COSTS OFF, TIMING OFF)
SELECT * FROM test WHERE id BETWEEN 1 AND 800000;
                          QUERY PLAN
-----------------------------------------------------------
-----
 Index Scan using test_pkey on test (actual rows=1000
loops=1)
   Index Cond: ((id >= 1) AND (id <= 800000))
   Buffers: shared hit=2196
 Planning:
   Buffers: shared hit=8
 Planning Time: 0.196 ms
 Execution Time: 3.815 ms
(7 rows)
```

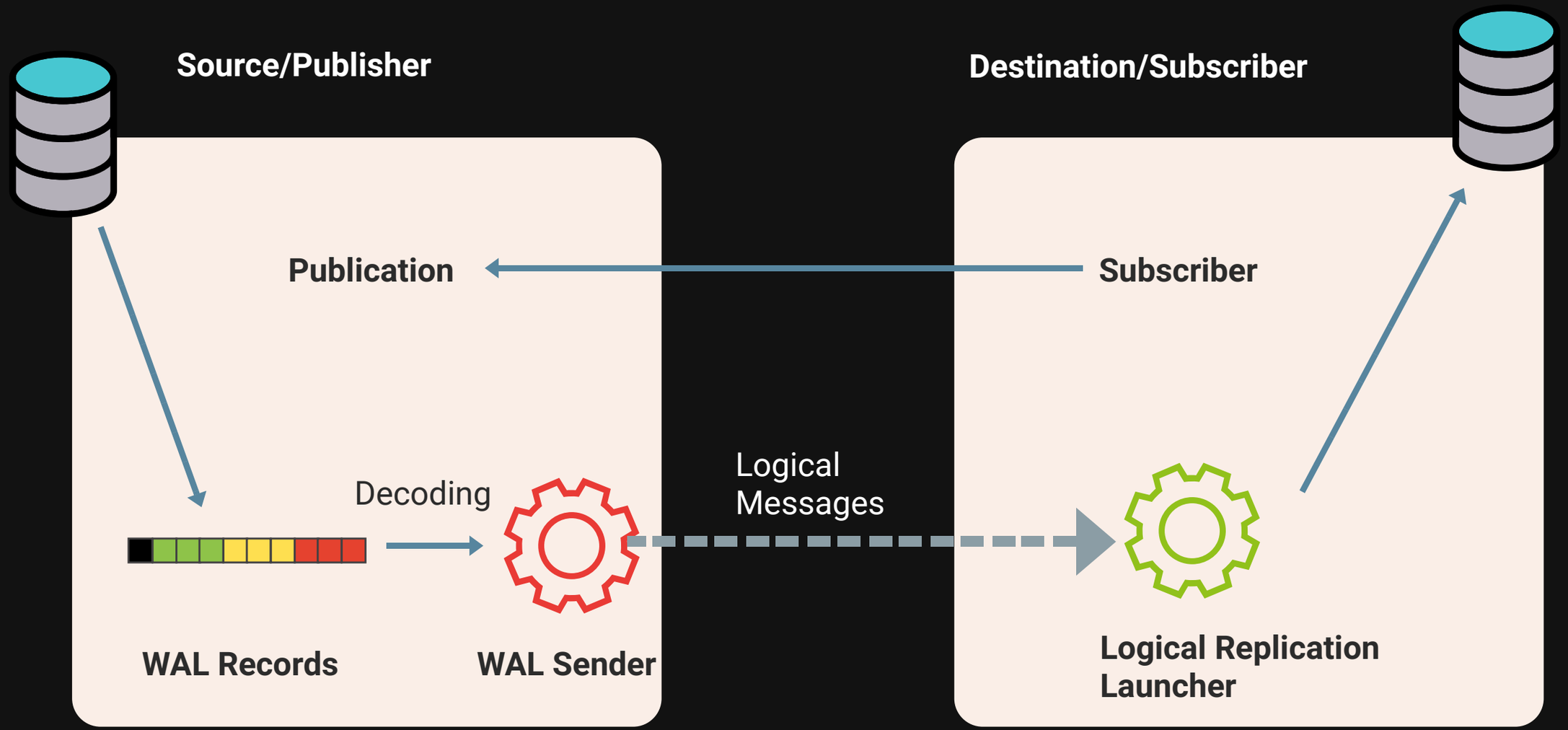
Bottom-Up Index Tuple Deletion

- Introduced in Version 14
- Delete Index Entries pointed to dead tuples
- Performing the operation of the VACUUM

Mydbops

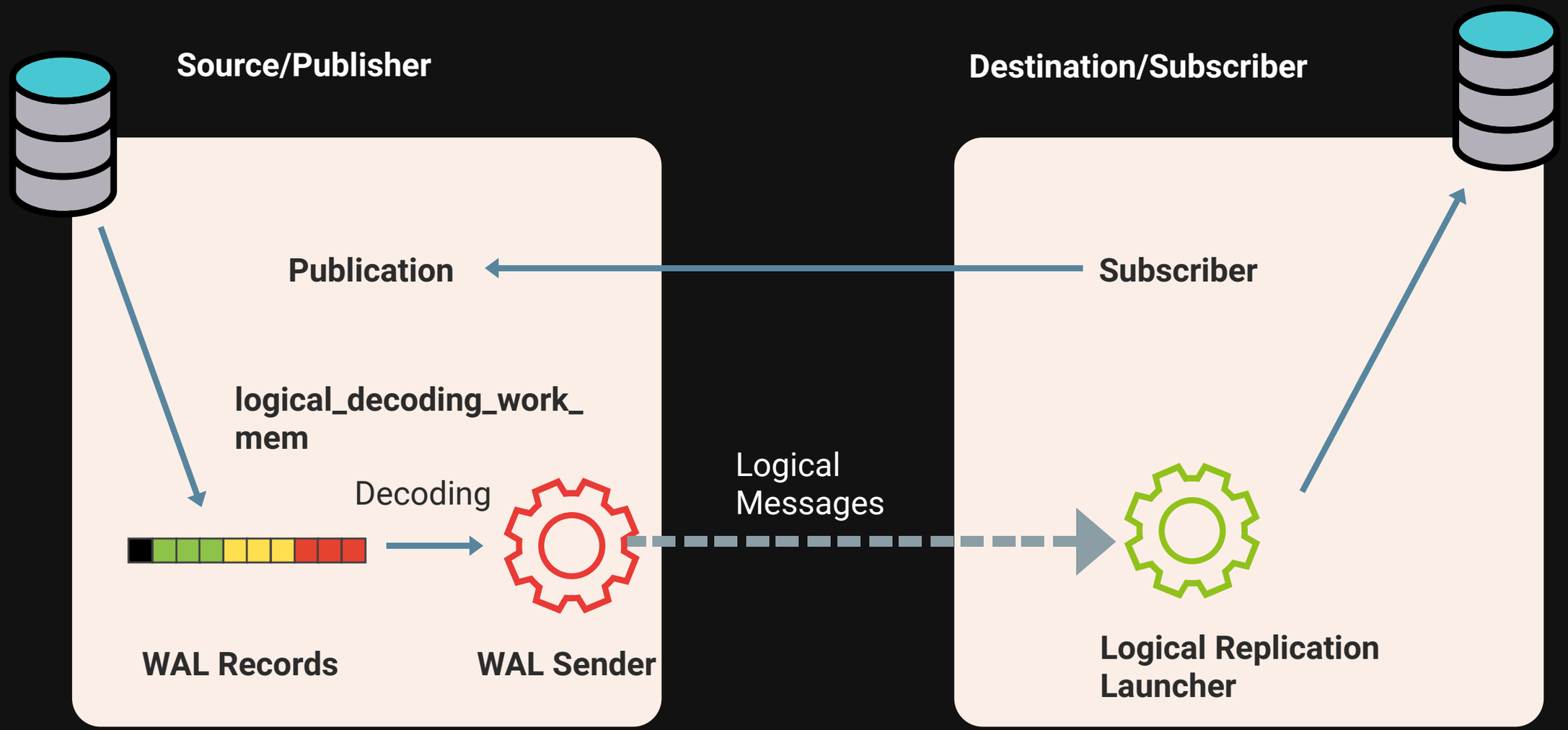
Replicating in-progress Transactions

Logical Replication - Publisher/Subscriber Model



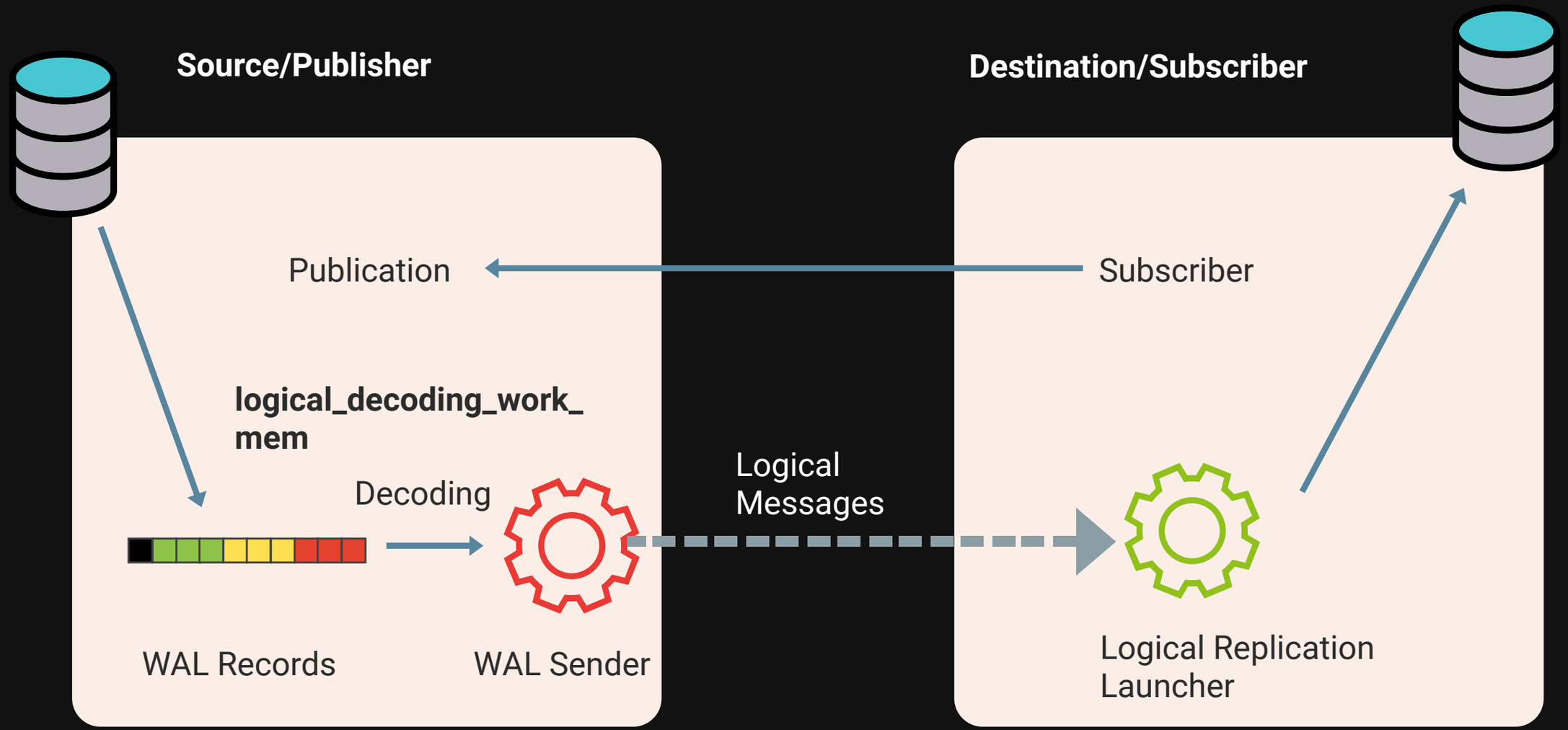
Mydbops

logical_decoding_work_mem - PostgreSQL 13



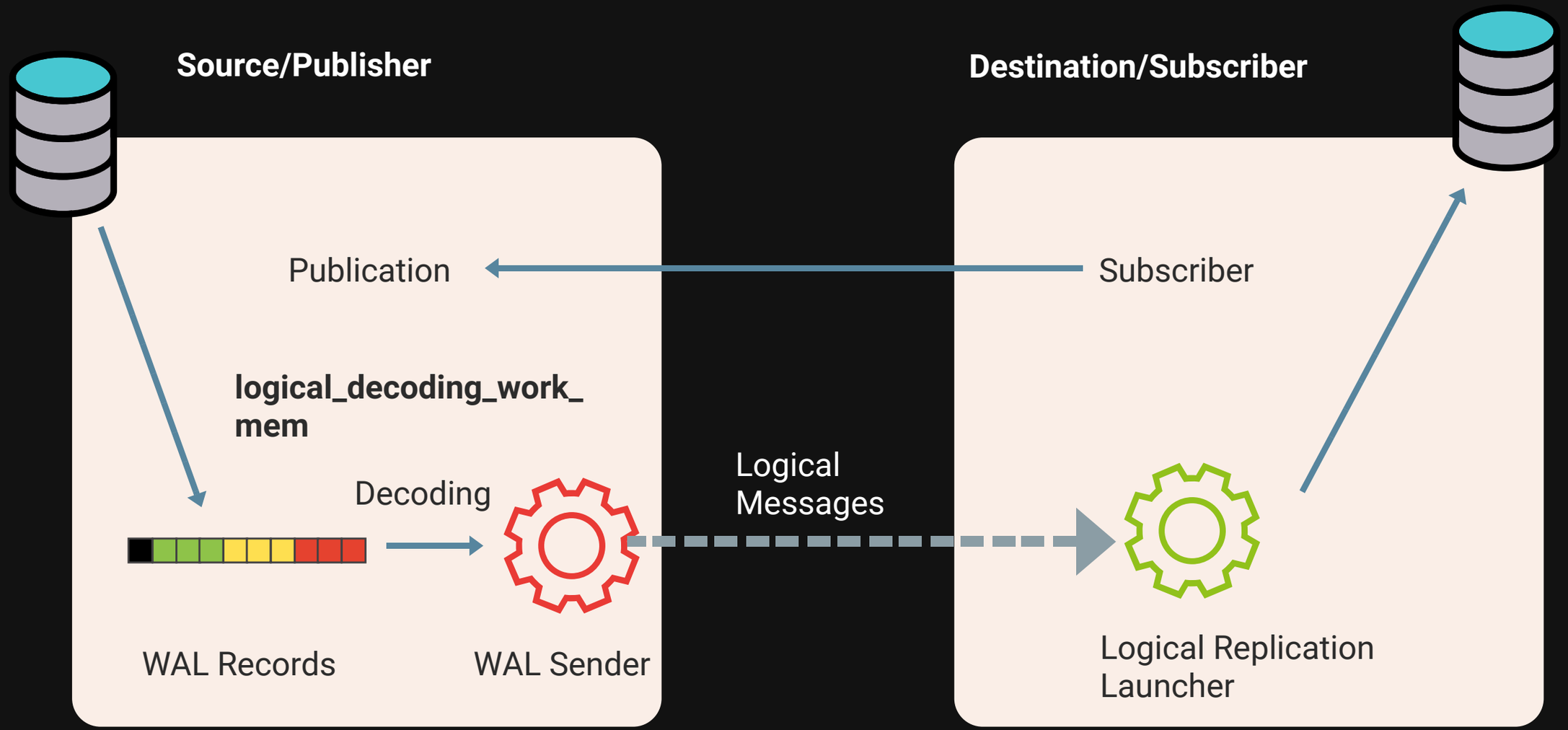
Mydbops

Logical Decode Work Mem - PostgreSQL 13



Mydbops

Streaming in-progress Transactions - PostgreSQL 14



Mydbops

Streaming in-progress Transactions - PostgreSQL 14

```
[publisher configurations]:

wal_level = logical
synchronous_standby_names = '*'
logical_decoding_work_mem=1MB
```

Mydbops

Streaming in-progress Transactions - PostgreSQL 14

```
[publisher setup]


CREATE TABLE test(id int ,name text);
CREATE PUBLICATION test_pub FOR TABLE test;

[subscription setup]


CREATE TABLE test(id int ,name text);
CREATE SUBSCRIPTION test_sub CONNECTION 'host=127.0.0.1
port=5432 dbname=postgres' PUBLICATION test_pub;
```

Mydbops

Streaming in-progress Transactions - PostgreSQL 14

```
[publisher setup]


CREATE TABLE test(id int ,name text);
CREATE PUBLICATION test_pub FOR TABLE test;

[subscription setup]


CREATE TABLE test(id int ,name text);
CREATE SUBSCRIPTION test_sub CONNECTION 'host=127.0.0.1
port=5432 dbname=postgres' PUBLICATION test_pub;
```

Mydbops

Streaming in-progress Transactions - PostgreSQL 14

```
INSERT INTO test SELECT i, REPEAT('x', 10) FROM
generate_series(1,5000000) AS i;

Time Taken for Commit : 6.7 Secs
```

Streaming in-progress Transactions - PostgreSQL 14

```
ALTER SUBSCRIPTION test_sub SET(STREAMING = ON)

INSERT INTO test SELECT i, REPEAT('x', 10) FROM
generate_series(1,5000000) AS i;

Time taken for COMMIT: 3.5 secs
```

LZ4 Toast Compression

TOAST

- Mechanism used to avoid exceeding the size of data block
- Try Compress
- Wider field values into a smaller pieces
- Default, this is 2KB
- Every table will have its own toast table

Mydbops

New Compression Configuration

- pglz - in built compression
- lz4 - New compression method added
- New Configuration Variable - default_toast_compression
- --with-lz4

Mydbops

Idle Session Timeout

Mydbops

Idle_session_timeout

- New variable
- Almost the same purpose of idle_in_transaction_session_timeout

```
postgres=#  select name,setting,unit,user,pending_restart from
pg_settings where name = 'idle_session_timeout' \gx
-[ RECORD 1 ]---+---------------------
name            | idle_session_timeout
setting         | 0
unit            | ms
context         | user
pending_restart | f
```

Mydbops

idle_in_transaction_session_timeout

- Available since PostgreSQL 9.6
- Maximum allowed idle time between queries, when **in transaction**

```
postgres=#  select name,setting,unit,context,pending_restart from
pg_settings where name = 'idle_in_transaction_session_timeout' \gx
-[ RECORD 1 ]---+-----------------------------------
name            | idle_in_transaction_session_timeout
setting         | 0
unit            | ms
context         | user
pending_restart | f
```

Mydbops

idle_in_transaction_session_timeout

- Opened a new session

```
postgres=# set idle_in_transaction_session_timeout=2000;
SET
postgres=#
```

- Started a new transaction

```
postgres=# start transaction;
START TRANSACTION
postgres=*#  select name,setting from pg_settings where name =
'idle_in_transaction_session_timeout' \gx
-[ RECORD 1 ]-----------------------------
name    | idle_in_transaction_session_timeout
setting | 2000
```

Mydbops

idle_in_transaction_session_timeout

```
postgres=# start transaction;
START TRANSACTION
postgres=*#  select name,setting from pg_settings where name =
'idle_in_transaction_session_timeout' \gx
-[ RECORD 1 ]-------------------------------
name    | idle_in_transaction_session_timeout
setting | 2000

postgres=*#  select name,setting from pg_settings where name =
'idle_in_transaction_session_timeout' \gx
FATAL:  terminating connection due to idle-in-transaction timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset:
Succeeded.
postgres=#
```

Idle_session_timeout

- Maximum idle time between the queries, when not in a transaction

```
postgres=# set Idle_session_timeout = 5000;
SET
postgres=#  select name,setting from pg_settings where name =
'idle_in_transaction_session_timeout' \gx
-[ RECORD 1 ]-----------------------------
name    | idle_in_transaction_session_timeout
setting | 0

postgres=#  select name,setting from pg_settings where name =
'idle_in_transaction_session_timeout' \gx
FATAL:  terminating connection due to idle-session timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset:
Succeeded.
```

Mydbops

Idle_session_timeout

- Helps to handle idle connections
- Apply only for the upcoming connections
- Should have higher value than application timeout

Mydbops

Improvement to Query Parallelism

Refresh Materialized View

Mydbops

Materialized View

- Database Object
- Stores only the result of the query
- Refresh needs to be done

Mydbops

Materialized View

- Loaded data in PostgreSQL 13

```
(postgres13)=> CREATE TABLE table_test (grp int, data numeric);
CREATE TABLE
Time: 3.282 ms
(postgres13)=>
(postgres13)=> INSERT INTO table_test SELECT 100, random() FROM
generate_series(1, 5000000);
INSERT 0 5000000
Time: 8064.350 ms (00:08.064)
(postgres13)=>
(postgres13)=> INSERT INTO table_test SELECT 200, random() FROM
generate_series(1, 5000000);
INSERT 0 5000000
Time: 9250.536 ms (00:09.251)
(postgres13)=>
```

Materialized View

- Loaded data in PostgreSQL 14

```
(postgres14)=> CREATE TABLE table_test (grp int, data numeric);
CREATE TABLE
Time: 13.577 ms
(postgres14)=>
(postgres14)=> INSERT INTO table_test SELECT 100, random() FROM
generate_series(1, 5000000);
INSERT 0 5000000
Time: 7204.919 ms (00:07.205)
(postgres14)=>
(postgres14)=> INSERT INTO table_test SELECT 200, random() FROM
generate_series(1, 5000000);
INSERT 0 5000000
Time: 8618.068 ms (00:08.618)
```

Mydbops

Materialized View

- Executing a query on both version

```
(postgres13)=> SELECT grp, avg(data), count(*) FROM table_test
GROUP BY 1;
 grp |            avg             |  count
-----+----------------------------+---------
 100 | 0.49997968954960787679 3 | 5000000
 200 | 0.50012354616793038739 3 | 5000000
(2 rows)
Time: 960.802 ms

(postgres14)=> SELECT grp, avg(data), count(*) FROM table_test
GROUP BY 1;
 grp |            avg             |  count
-----+----------------------------+---------
 100 | 0.50017301919687580407 2 | 5000000
 200 | 0.50006568233528992873 9 | 5000000
(2 rows)
Time: 965.867 ms
```

Mydbops

Materialized View

- Creating materialized view for the static table

```
(postgres13)=> CREATE MATERIALIZED VIEW test_view AS SELECT grp,
avg(data), count(*) FROM table_test GROUP BY 1;
SELECT 2
Time: 983.138 ms

(postgres14)=> CREATE MATERIALIZED VIEW test_view AS SELECT grp,
avg(data), count(*) FROM table_test GROUP BY 1;
SELECT 2
Time: 978.225 ms
(postgres14)=>
```

Mydbops

Materialized View

- Querying the materialized view instead of table

```
(postgres13)=> select * from test_view;
 grp |          avg           |  count
-----+------------------------+---------
 100 | 0.49997968954960787679 | 5000000
 200 | 0.50012354616793038739 | 5000000
(2 rows)
Time: 0.561 ms

(postgres14)=> select * from test_view;
 grp |          avg           |  count
-----+------------------------+---------
 100 | 0.50017301919687580407 | 5000000
 200 | 0.50006568233528992873 | 5000000
(2 rows)
Time: 0.540 ms
```

Mydbops

Refresh Materialized View

- Refreshing the materialized view to get the updated result

```
(postgres13)=> refresh materialized view test_view;
REFRESH MATERIALIZED VIEW
Time: 2027.911 ms (00:02.028)
(postgres13)=>

(postgres14)=> refresh materialized view test_view;
REFRESH MATERIALIZED VIEW
Time: 961.503 ms
```

Mydbops

Refresh Materialized View

- Explain plan with parallel workers

```
(postgres14)=> explain  SELECT grp, avg(data), count(*) FROM
table_test GROUP BY 1;
                                                  QUERY PLAN
-------------------------------------------------------------------------------
-----------------------------------
 Finalize GroupAggregate  (cost=127997.34..127997.87 rows=2
width=44)
   Group Key: grp
   ->  Gather Merge  (cost=127997.34..127997.81 rows=4 width=44)
         Workers Planned: 2
         ->  Sort  (cost=126997.32..126997.32 rows=2 width=44)
               Sort Key: grp
               ...... on table_test  (cost=0.00..95747.02
rows=4166702 width=15)
(9 rows)

Time: 0.438 ms
```

Mydbops

Refresh Materialized View

- Other than refresh, remaining use parallelism
- For version < 14, it is better to recreate it than refresh
- During refresh, it will lock the content
- Concurrent option, but PK is mandatory

Mydbops

Return Query

Mydbops

Security Improvements

Mydbops

Default Authentication Method

Mydbops

Default Authentication Method - scram-sha-256

- Previously MD5 hashing method
- Easy to reconstruct
- Expensive Hash Function
- Introduced in V10, now made it as default
- Client needs to support it

```
authentication method 10 not supported
```

Mydbops

Predefined Roles

Mydbops

Predefined Roles - pg_read_all_data or pg_write_all_data

- Be a superuser
- Provide access to the complete database

Mydbops

Predefined Roles - pg_read_all_data

- Read all data(Tables, views and sequences)
- Select access on objects and usage access on schemas

Mydbops

Predefined Roles - pg_write_all_data

- Write all data(Tables, views and sequences)
- Insert, Update, Delete access on objects and usage access on schemas

Mydbops

Predefined Roles - pg_read_all_data or pg_write_all_data

```
(postgres14)=> create user testing;
CREATE ROLE
Time: 5.468 ms
(postgres14)=> set role to 'testing';
SET
Time: 0.392 ms
(postgres14)=> SELECT grp, avg(data), count(*) FROM table_test
GROUP BY 1;
ERROR:  permission denied for table table_test
Time: 2.010 ms
(postgres14)=>
```

Predefined Roles - pg_read_all_data or pg_write_all_data

```
(postgres14)=> reset role;
RESET
Time: 0.189 ms

(postgres14)=> grant pg_read_all_data to testing;
GRANT ROLE
Time: 4.202 ms
(postgres14)=>
(postgres14)=> set role to 'testing';
SET
Time: 0.148 ms
(postgres14)=> SELECT grp, avg(data), count(*) FROM table_test
GROUP BY 1;
 grp |          avg             |  count
-----+--------------------------+---------
 100 | 0.50017301919687580 4072 | 5000000
 200 | 0.50006568233528992 8739 | 5000000
(2 rows)
```

Mydbops

Features for Application Developers

New JSON Syntax

New JSON Syntax

- Since PostgreSQL 9.2
- Had unique syntax

Old Syntax:

```
where (jsonb_column->>'name')::varchar = 'Aakash:
DBRE(2021)';
```

New Syntax:

```
where jsonb_column['name'] = '"Aakash: DBRE(2021)"';
```

Mydbops

Multirange Datatypes

Mydbops

Range

- Beginning and End value
- Datetime or Integer
- This train is running between X and Y

Range - Classic SQL

```
create table track_activity (
id serial primary key,
appname varchar(100),
start_ts timestamp,
end_ts timestamp
);

insert into track_activity (appname,start_ts,end_ts) values
('Whatapp', '2021-10-01 4:00', '2021-10-01 7:00');
insert into track_activity (appname,start_ts,end_ts) values
('Facebook', '2021-10-01 7:00', '2021-10-01 11:00');
insert into track_activity (appname,start_ts,end_ts) values
('Whatsapp', '2021-10-01 11:00', '2021-10-01 21:00');
insert into track_activity (appname,start_ts,end_ts) values
('Instagram', '2021-10-01 22:00', '2021-10-01 23:00');
```

Mydbops

Range - Classic SQL

```
(postgres14)=> select * from track_activity;
 id |  appname  |       start_ts      |        end_ts
----+-----------+---------------------+---------------------
  1 | Whatapp   | 2021-10-20 04:00:00 | 2021-10-20 07:00:00
  2 | Facebook  | 2021-10-20 07:00:00 | 2021-10-20 11:00:00
  3 | Whatsapp  | 2021-10-20 11:00:00 | 2021-10-20 21:00:00
  4 | Instagram | 2021-10-20 22:00:00 | 2021-10-20 23:00:00
(4 rows)
```

Mydbops

Range - Classic SQL

```
(postgres14)=> select id, appname from track_activity where
start_ts <= '2021-10-20 11:00'::timestamp and end_ts >= '2021-10-
20 12:00'::timestamp;
 id | appname
----+-----------
  3 | Whatsapp
(1 row)
```

Mydbops

Range - Classic SQL

```
(postgres14)=> select id, appname from track_activity where
(start_ts <= '2021-10-20 11:00'::timestamp and end_ts >= '2021-10-
20 12:00'::timestamp) or (start_ts <= '2021-10-20
17:00'::timestamp and end_ts >= '2021-10-20 18:00'::timestamp);
 id | appname
----+-----------
  3 | Whatsapp
(1 row)
```

Mydbops

Range - Using Range Data Type

```
create table track_activity_range (
id serial primary key,
appname varchar(100),
time_ts tsrange
);

insert into track_activity_range (appname,time_ts) values
('Whatapp', '[2021-10-20 4:00, 2021-10-20 7:00]');
insert into track_activity_range (appname,time_ts) values
('Facebook', '[2021-10-20 7:00, 2021-10-20 11:00]');
insert into track_activity_range (appname,time_ts) values
('Whatsapp', '[2021-10-20 11:00, 2021-10-20 21:00]');
insert into track_activity_range (appname,time_ts) values
('Instagram', '[2021-10-20 22:00, 2021-10-20 23:00]');
```

Mydbops

Range - Using Range Data Type

```
(postgres14)=> select id, appname from track_activity_range where
time_ts @> '2021-10-20 12:00'::timestamp;
 id ┆ appname
----+----------
  3 | Whatsapp
(1 row)
```

Mydbops

Range - Using Range Data Type

```
(postgres14)=> select * from track_activity_range where id=3;
 id | appname  |                    time_ts
----+----------+---------------------------------------------
  3 | Whatsapp | ["2021-10-20 11:00:00","2021-10-20 21:00:00"]
(1 row)


(postgres14)=> update track_activity_range set time_ts = time_ts -
'[2021-10-20 11:00,2021-10-20 11:30]'::tsrange where id = 3;
UPDATE 1


(postgres14)=> select * from track_activity_range where id=3;
 id | appname  |                    time_ts
----+----------+---------------------------------------------
  3 | Whatsapp | ("2021-10-20 11:30:00","2021-10-20 21:00:00"]
(1 row)
```

Range - Using Range Data Type

```
(postgres14)=> select * from track_activity_range where id=3;
 id | appname  |                      time_ts
----+----------+-------------------------------------------------
  3 | Whatsapp | ("2021-10-20 11:30:00","2021-10-20 21:00:00"]
(1 row)
```

```
(postgres14)=> update track_activity_range set time_ts = time_ts -
'[2021-10-20 12:00,2021-10-20 12:30]' where id = 3;
ERROR:  result of range difference would not be contiguous
Time: 0.594 ms
(postgres14)=>
```

Mydbops

Range - Using Multi Range Data Type

```
create table track_activity_multirange (
id serial primary key,
appname varchar(100),
time_ts tsmultirange
);

insert into track_activity_multirange (appname,time_ts) values
('Whatapp', '{[2021-10-20 4:00, 2021-10-20 7:00]}');
insert into track_activity_multirange (appname,time_ts) values
('Facebook', '{[2021-10-20 7:00, 2021-10-20 11:00]}');
insert into track_activity_multirange (appname,time_ts) values
('Whatsapp', '{[2021-10-20 11:00, 2021-10-20 21:00]}');
insert into track_activity_multirange (appname,time_ts) values
('Instagram', '{[2021-10-20 22:00, 2021-10-20 23:00]}');
```

Mydbops

Range - Using Multi Range Data Type

```
(postgres14)=> select * from track_activity_multirange;
 id |   appname   |                          time_ts
----+-------------+-----------------------------------------------------
  1 | Whatapp     | {["2021-10-20 04:00:00","2021-10-20 07:00:00"]}
  2 | Facebook    | {["2021-10-20 07:00:00","2021-10-20 11:00:00"]}
  3 | Whatsapp    | {["2021-10-20 11:00:00","2021-10-20 21:00:00"]}
  4 | Instagram | {["2021-10-20 22:00:00","2021-10-20 23:00:00"]}
(4 rows)
```

Mydbops

Range - Using Multi Range Data Type

```
(postgres14)=> update track_activity_multirange set time_ts =
time_ts - '{[2021-10-20 12:00,2021-10-20 12:30]}' where id = 3;
UPDATE 1


(postgres14)=> select * from track_activity_multirange where id=3;
 id | appname  |
time_ts
----+----------+-----------------------------------------------
-----------------------------------------
  3 | Whatsapp | {["2021-10-20 11:00:00","2021-10-20 12:00:00"),
("2021-10-20 12:30:00","2021-10-20 21:00:00"]}
(1 row)
```

Mydbops

OUT Parameter for Stored Procedure

Mydbops

Out Parameter for Stored Procedure

- Only function has been supported
- Stored Procedure since V11
- Only supports IN , INOUT parameters
- More compatible with Oracle's Implementation of procedures

Mydbops

Enhancements in Monitoring

pg_stat_progress_copy

Mydbops

pg_stat_progress_copy

- Reporting progress of COPY commands

```
ppostgres=# select * from pg_stat_progress_copy \watch 1
                                                Tue Oct 19 13:25:45
2021 (every 1s)

 pid | datid | datname | relid | command | type |
bytes_processed | bytes_total | tuples_processed |
tuples_excluded
-----+-------+---------+-------+---------+------+-----------------
----+-------------+-------------------+-----------------
(0 rows)
```

Mydbops

pg_stat_progress_copy

- Session 1 - Load CSV with 10M records

```
postgres=# \copy (    select i, now() - random() * '2
years'::interval, random()
    from generate_series(1,10000000) i) to /dev/null;
```

Mydbops

pg_stat_progress_copy

- Session 2 - Monitoring pg_stat_progress_copy table continuously

```
                                                Tue Oct 19 13:31:31
2021 (every 1s)

  pid  | datid | datname  | relid | command | type |
bytes_processed | bytes_total | tuples_processed |
tuples_excluded
-------+-------+----------+-------+---------+------+----------
-------+-------------+--------------------+-----------------
 28382 | 14486 | postgres |     0 | COPY TO | PIPE |
2267809 |           0 |              41232 |               0
(1 row)
```

Mydbops

pg_stat_progress_copy

```
datname  | command | tuples_processed
----------+---------+------------------
 postgres | COPY TO |                0
(1 row)

  Tue Oct 19 13:37:24 2021 (every 1s)

 datname  | command | tuples_processed
----------+---------+------------------
 postgres | COPY TO |           704370
(1 row)

  Tue Oct 19 13:37:25 2021 (every 1s)

 datname  | command | tuples_processed
----------+---------+------------------
 postgres | COPY TO |          1612781
(1 row)
```

pg_stat_progress_copy

```
postgres=# \copy (    select i, now() - random() * '2
years'::interval, random()
    from generate_series(1,10000000) i) to /dev/null;
COPY 10000000
postgres=#
```

```
 datname | command | tuples_processed
---------+---------+------------------
(0 rows)

 Tue Oct 19 13:40:26 2021 (every 1s)

 datname | command | tuples_processed
---------+---------+------------------
(0 rows)
```

Mydbops

Enhancements in pg_stat_statements

Mydbops

pg_stat_statements

- Very useful extension
- Tracking execution statistics of all SQL statements
- Inexpensive
- Supports in all infra
- Load this data in grafana
- Stores the hash value

Mydbops

pg_stat_statements

```
postgres=# select query,calls from pg_stat_statements where userid=10;
                            query                                | calls
-----------------------------------------------------------------+-------
 select * from pg_stat_statements_info                           |     1
 select query,calls from pg_stat_statements where userid=$1      |     1
 select * from pg_stat_statements where userid=$1                |     3
 select version()                                                |     1
 select * from pg_stat_statements                                |     2
 create extension pg_stat_statements                             |     1
 show shared_preload_libraries                                   |     1
(7 rows)
```

Mydbops

pg_stat_statements

- When the given queries happened
- Queries with exact values

Mydbops

Enhancements in pg_stat_statements

- System View - pg_stat_statements_info
- Shows since the query stats are available

```
postgres=# select * from pg_stat_statements_info;
 dealloc |          stats_reset
---------+-------------------------------
       0 | 2021-10-18 14:28:12.693117+00
(1 row)
```

Mydbops

Enhancements in pg_stat_statements

- New log prefix added - %Q
- Also store the query id in the log file as well

```
postgres=# select queryid,query from pg_stat_statements limit 1;
       queryid       |                    query
---------------------+-----------------------------------------
 2403671352246453279 | select queryid,query from pg_stat_statements
(1 row)
```

In Error log:

```
2021-10-18 14:55:15 UTC [10840]: [2-1]
user=postgres,db=postgres,app=psql,client=
[local],query_id=2403671352246453279 LOG:  duration: 3.580 ms
```

Mydbops

Enhancements in pg_stat_statements

- New column added - rows
- No of rows processed

```
postgres=# select query,rows from pg_stat_statements limit 1;
                  query                       | rows
----------------------------------------------+------
 select queryid,query from pg_stat_statements |    7
(1 row)
```

Mydbops

Planning for Upgrade

- Logical / Physical Upgrade
- Perform dry-run if possible
- Version < 12 requires reindexing.
- Analyze needs to be performed
- Perform functional testing of application

Mydbops

References

- https://sql-info.de/postgresql/postgresql-14/articles-about-new-features-in-postgresql-14.html
- https://www.cybertec-postgresql.com/en/index-bloat-reduced-in-postgresql-v14/
- https://www.postgresql.org/about/news/postgresql-14-released-2318/
- https://www.enterprisedb.com/blog/logical-decoding-large-progress-transactions-postgresql

Mydbops

Thank You

Reach Us : Info@mydbops.com