Evolution of Replication in PostgreSQL

Anuja P

Mydbops LLP

Mydbops MyWebinar 49

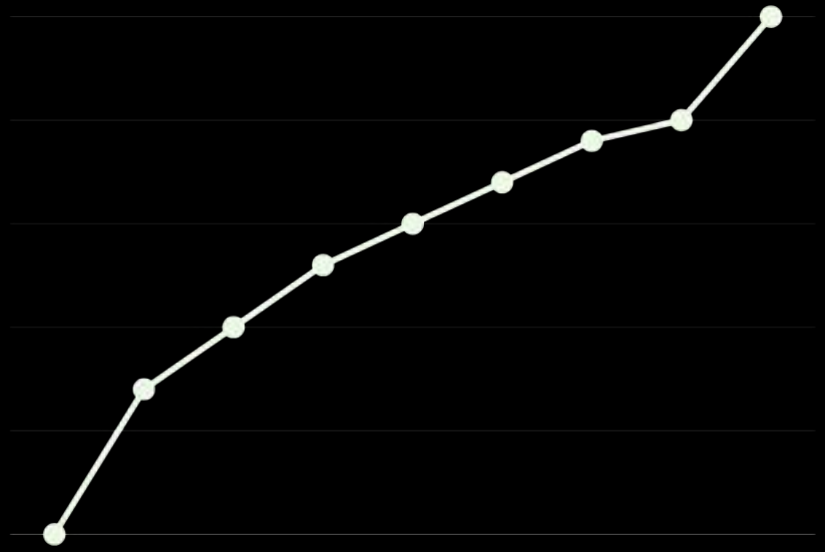
Mydbops
Scaling Databases

Mydbops

Scaling Databases

Your Trusted Database Management Partner

With 9+ Years of Expertise



Mydbops by the Numbers

9+ years
Of Expertise

800 +
Happy Clients

10 B +
DB Transactions Handled per Day

6000 +
Servers Monitored

3000 +
Tickets Handled per Day

Database Technologies

MySQL · MongoDB · PostgreSQL · MariaDB · TiDB · Cassandra

Mydbops
Scaling Databases

Agenda

- From Triggers to Core Features
- WAL
- Physical Streaming Replication
- Native Logical Replication
- Enhancements in Replication
- Q&A
- Key Takeaways

Mydbops
Scaling Databases

Evolution Timeline

2010
SEPT 20

Physical Streaming
Foundation of real-time replication

2014
DEC 18

Replication Slot & Logical Decoding
Change data capture revolution

2017
OCT 05

Publisher/Subscriber Framework
Native logical replication model

2022
OCT 13

Row and Column Filtering
Selective replication capabilities

2023
SEPT 14

Parallel Appliers
Enhanced performance scaling

2024
SEPT 26

Failover Slots
High availability improvements

Mydbops
Scaling Databases

The Early Era (Pre-9.0)

(Age of Trigger based tools)

Mydbops
Scaling Databases

The Early Era (Pre-9.0)

The Age of External "Trigger-Based" Tools

Initial Stance

Replication was outside the core scope

The Methodology

Middleware using SQL triggers to capture data changes to log tables

Key Tools

Slony-I

Gold standard for years

Complex configuration

Londiste

Part of SkyTools suite

Limitations

- High overhead (double write)
- DDL schema changes hard to manage
- Replication crashes if triggers fail

Example - Slony-I

Practical Implementation

Trigger Action

```
INSERT INTO _slony.sl_log_1 (cmd, data)
VALUES ('UPDATE', 'id=1, bal=100');
```

Doubled disk writes for every transaction

Mydbops
Scaling Databases

WAL - Write Ahead Log

Mydbops
Scaling Databases

WAL

Core Replication Component

Write-Ahead Logging (WAL) enables physical replication in PostgreSQL.

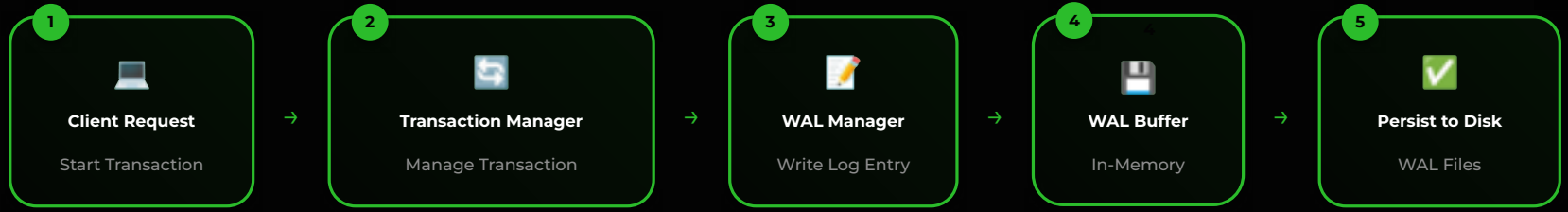
Data Changes

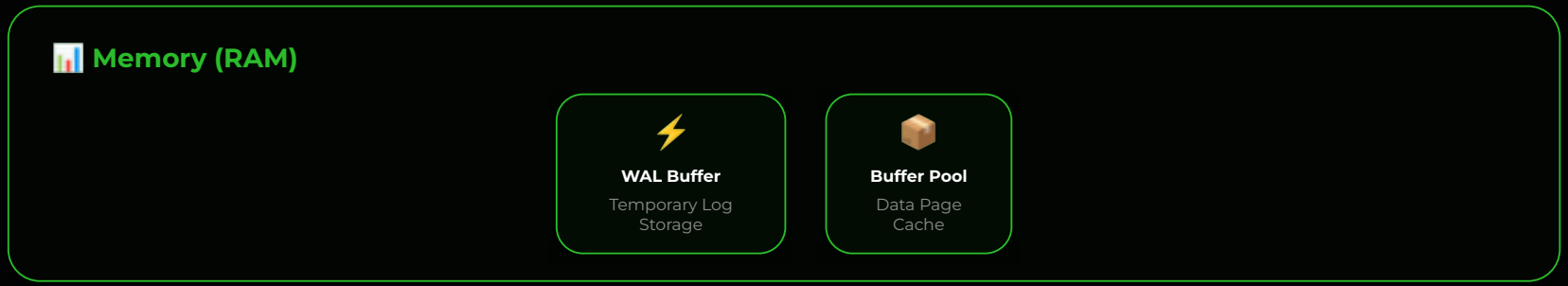
Records before applying

Data Integrity

Crash recovery

PostgreSQL WAL Architecture





WAL Sender and WAL Receiver

Core Replication Processes

WAL Sender

Primary Server

- Reads WAL
- Streams records
- Multiple connections
- Real-time sync

WAL Receiver

Standby Server

- Receives records
- Writes to WAL
- Applies changes
- Maintains sync

- Continuous streaming
- Data integrity
- Low latency

What WAL Contains?

Stored In

PostgreSQL 9.X

pg_xlog

PostgreSQL 10+

pg_wal

Physical Record Types

Data-level physical change records (non-table rows)

Page-level modifications

Transaction commit and abort records

Replay redo information

Segment Size

Wal_Segment_Size

Maximum

1024 MB

Default

16 MB

Mydbops
Scaling Databases

How WAL Logging Works: Detailed Examples

WAL Record Structure

rmgr:	Resource manager (Heap, Transaction)
len(rec/tot):	Record length / Total length in bytes
tx:	Transaction ID for tracking changes
desc:	Operation details

WAL Storage

pg_wal/ directory with 16MB segment files

Binary files with actual data and metadata

Operation Examples

INSERT tx: 738

```
rmgr:Heap len(rec/tot):65/65
desc:INSERT+INIT off 1 flags 0x00
blk ref#0: rel 1663/16388/16390 blk 0
```

UPDATE tx: 743

```
rmgr:Heap len(rec/tot):84/84
desc:UPDATE off 4 flags 0x00
blk ref#0: rel 1663/16388/16390 blk 0
```

DELETE tx: 744

```
rmgr:Heap len(rec/tot):54/54
desc:DELETE off 3 flags 0x00 KEYS_UPDATED
blk ref#0: rel 1663/16388/16390 blk 0
```

WAL: From Basic Logging to Intelligent Failover

7.1

Introduction

A write-ahead log (WAL) is now implemented.

8.0

PITR

Point-in-Time Recovery allows restore to any point.

9.0

Streaming Replication

Streaming Replication allows standby to stay more up-to-date.

9.4

Logical Decoding

Logical Decoding allows extraction of the change stream.

17

Failover / Slots

Allow logical replication slots to follow the primary after a failover.

Mydbops
Scaling Databases

Physical Replication

Version 9 Revolution

Built-in Physical Streaming Replication

Physical Replication

Database cluster byte-level replication

Relies on WAL logs

The Mechanism: WAL

Sequential record of changes

HOT STANDBY

Execute SELECT queries during recovery

Core Advantages

- Strong architecture
- Automatic DDL
- Low overhead on primary server

Mydbops
Scaling Databases

Replication Slots

Mydbops
Scaling Databases

Need of Replication Slots

THE PROBLEM

Primary

Database server

↓

WAL Files

Write-Ahead Logging

↓

Standby

Replication server

Critical Issue

Primary deleting WAL files before the standby could read them.

CONSEQUENCE

Without a mechanism to track standby progress, the primary server may **recycle or remove WAL segments** that are still needed, causing replication to **fail or require full re-sync**.

Solution Required: Replication Slots

Replication Slots

Data Protection Mechanism

WAL Retention

Data Safety

Status Tracking

Mydbops
Scaling Databases

Logical Decoding

Mydbops
Scaling Databases

Logical Turning Point

V9.4 - V10

Logical Decoding

Binary WAL → JSON/SQL

1 WAL Generated

Primary Node

WAL records created

2 Plugin Reads

Decoding Plugin

Reads WAL stream

3 Conversion

Logical Events

INSERT: row_id=1

4 Consumers

Event Stream

{"op": "i", "id": 1}

JSON/SQL Format

Human-Readable

Real-Time

What Logical Decoding Enabled

Revolutionary Capabilities

Change Data Capture

Data Integration

Kafka

Elasticsearch

Redshift

Mydbops
Scaling Databases

Change Data Capture

1 TRANSACTION HAPPENS

Binary WAL

0x4F 0x8A 0x3C

0x7B 0x2D 0xE5

0x1F 0x9C 0x4A

2 THE EVOLUTION

Decoding Plugin

wal2json

Reads binary logs & converts

3 OUTPUT

JSON Format

```
{"table": "customers",
"op": "INSERT",
"data": {
"id": 123,
"name": "John"
}}
```

4 STREAMED TO

Debezium

Kafka

Search Engine

Elastic

Data Lake

S3

Mydbops

Scaling Databases

Logical Replication

Mydbops
Scaling Databases

The Big Change - V 10

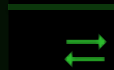
Publish/Subscribe Model

Publication

Primary Server

Select tables

Define changes



Subscription

Standby Server

Connect

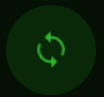
Receive changes

Initial sync

Apply updates

Selective Tables

Cross-Version



Continuous Sync

Mydbops

Scaling Databases

Example – Cross Version Migration

Publisher

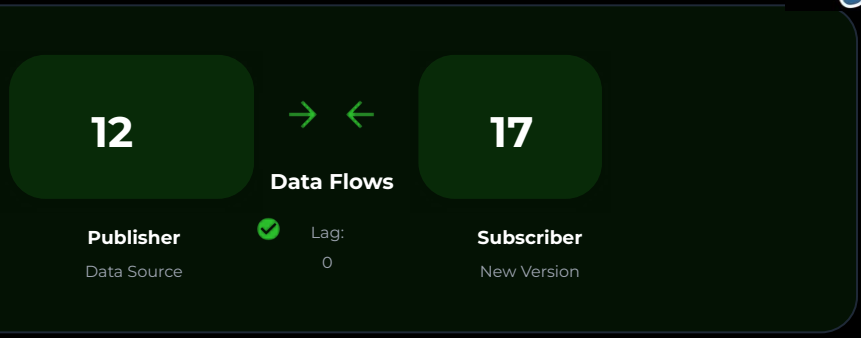
PG v12

```
CREATE PUBLICATION
upgrade_pub
FOR ALL TABLES;
```

Subscriber

PG v17

```
CREATE SUBSCRIPTION ...
PUBLICATION
upgrade_pub;
```



1. App connects to v12
   Background synchronization in progress
2. Wait for lag 0
   Verify data consistency and synchronization
3. App points to v17
   Switch application connection to new version

Upgrade Complete (Seconds of downtime)

Mydbops
Scaling Databases

Modern Enhancements (v15 - v17)

Mydbops
Scaling Databases

PG15 - Granularity & Filtering

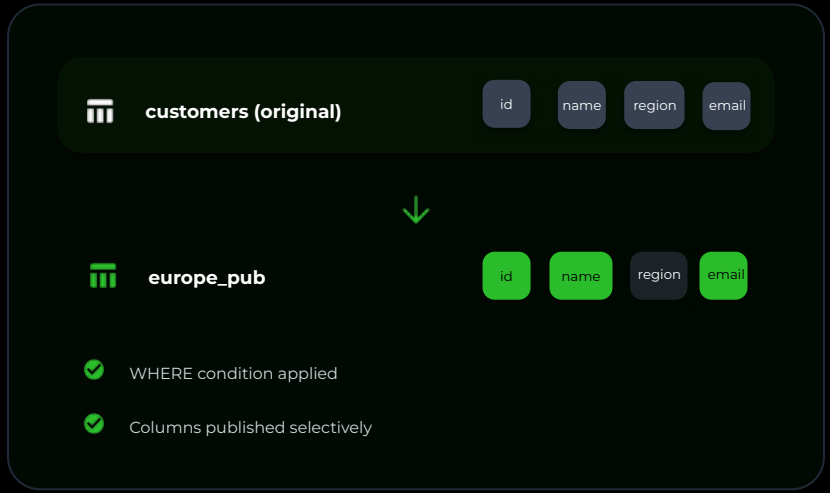
Row-Level Filtering

```
CREATE PUBLICATION europe_pub
FOR TABLE customers
WHERE (region = 'Europe');
```

Column Filtering

```
CREATE PUBLICATION europe_pub
FOR TABLE customers  (id, name,email);
```

Table Filtering Visual



Mydbops
Scaling Databases

PG16 - Parallel Apply

Parallel Streaming

```
CREATE SUBSCRIPTION fast_sub
CONNECTION 'host-pg17 dbname=prod'
PUBLICATION prod
WITH (streaming = parallel);
```

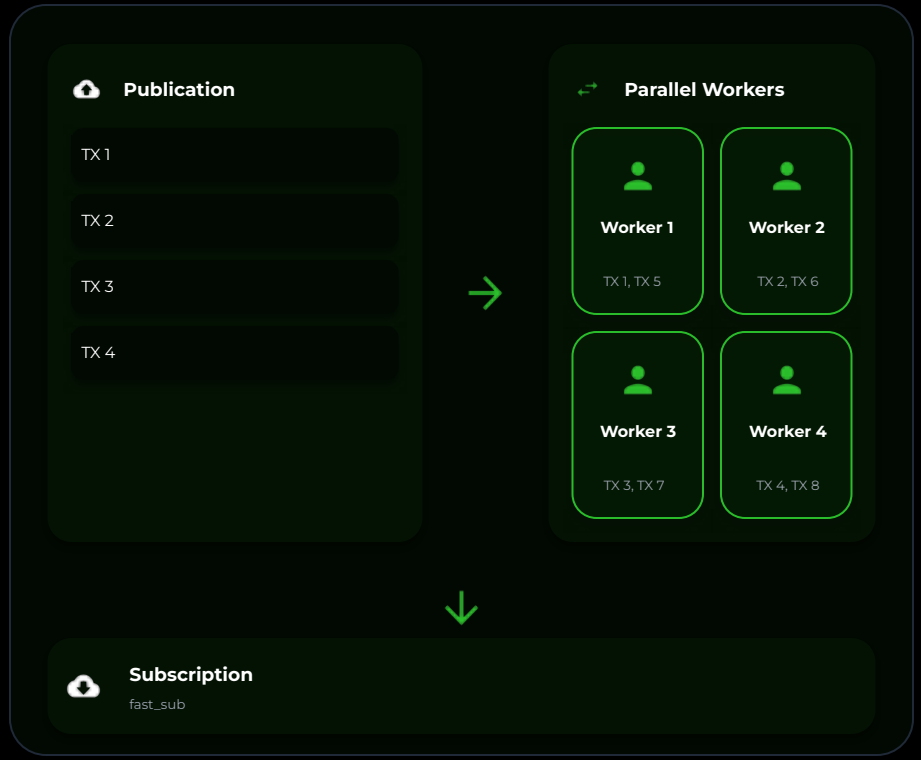
Worker Configuration

postgresql.conf

```
max_parallel_apply_workers_per_subscription = 4
```

- Increased throughput
- Faster catch-up
- Better resource utilization

Parallel Processing Architecture



PG17 - Failover Control

Slot Persistence

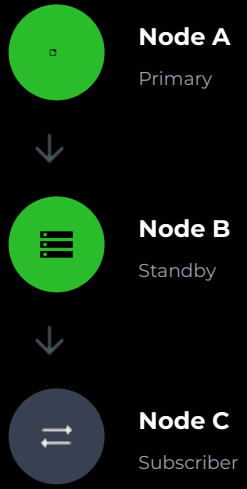
- Slots survive failover
- Automatic advancement

Hybrid HA

- Primary-Standby
- Logical + Physical

Example – PG 17 Failover Control

Hybrid HA (Physical HA protects Logical Replication)



Primary (Node A)

Define the data to be shared

```
CREATE PUBLICATION
sales_pub
FOR TABLE orders;
```

Standby (Node B)

Enable shadowing and prevent vacuum conflicts

```
sync_replication_slots = on
hot_standby_feedback = on
```

Subscriber (Node C)

Opt-in to the failover protection

```
CREATE SUBSCRIPTION
sales_sub ...
WITH (failover = true);
```

- Physical Replication
- Logical Replication

THE BREAKTHROUGH: Node B shadows Node A's logical replication slot, ensuring no data loss during failover

Mydbops
Scaling Databases

Why The Failover Parameter Matters

Active Synchronization & Zero-Touch Recovery

Node A

Primary

Real Slot

Node B

Standby

- Shadow Slot

Node C

Subscriber

Lsn: 0/4000

1 Seamless Transition

Node B maintains Shadow Slot tracking WAL stream

2 Handshake Success

Node B becomes Primary knowing Node C, no need to re-sync

3 Outcome

Zero-touch recovery, data automatically resumes from crash point

NO DATA LOSS · NO MANUAL INTERVENTION

Frequently Asked Questions

Database Replication Solutions

Mydbops
Scaling Databases

FAQ

Query

Can I use Physical and Logical replication at the same time in POSTGRESQL?

Result

Yes. Use Physical for High Availability while using Logical for granular data sharing.

Mydbops
Scaling Databases

FAQ

Query

Why does Logical Replication not replicate the Schema (DDL) by default in POSTGRESQL?

Result

It is designed for flexibility across versions and different table structures.

Mydbops
Scaling Databases

FAQ

Query

Why use Logical Replication for upgrades instead of pg_upgrade in POSTGRESQL?

Result

Zero Downtime. pg_upgrade requires a shutdown for hours; Logical runs live.

Mydbops
Scaling Databases

Conclusion

Complexity → Native Simplicity

Use Physical to protect your infrastructure; use Logical to empower your data.

Physical

Protect your Infrastructure

Logical

Empower your Data

Thank You