PostgreSQL : Replication, WAL, WAL Decoding, and the Journey Toward Zero-ETL

- PostgreSQL is known for its robustness and extensibility, and at the heart of its durability and replication mechanisms lies the Write-Ahead Log (WAL). Whether you're ensuring high availability with cluster replication or building a modern Change Data Capture (CDC) pipeline, understanding how PostgreSQL uses WAL — and how it compares with tools like AWS DMS and Zero-ETL — is critical.

What is WAL?

- The Write-Ahead Log (WAL) is PostgreSQL's foundational mechanism for ensuring durability and crash recovery. Every time data is modified, PostgreSQL first writes a record of the change to the WAL — before applying it to the actual data files.

Note : if you are familiar with Oracle, this is related to Oracle REDMO management.

This guarantees:

- ⍰Durability (the "D" in ACID) — no committed transaction is lost.
- ⍰Crash Recovery — WAL is replayed to recover state after a crash.
- ⍰Foundation for Replication — WAL powers both physical and logical replication.

WAL Decoding: Making Change Data Streamable

- While WAL ensures resilience, it isn't readable by humans or directly usable for downstream systems.

- WAL Decoding is the process of converting WAL's binary entries into logical, row-level change events like:

```
{"action": "INSERT", "table": "orders", "columns": {"id": 1, "status": "shipped"}}
```

This decoding enables:

- Logical Replication
- Change Data Capture (CDC)
- Integration with Kafka, Redshift, or S3

Tools for WAL Decoding:

- **pgoutput:** Built-in logical decoding plugin used by native logical replication.
- **test_decoding:** Simple text-based plugin, great for learning and debugging.
- Third-party tools like Debezium for PostgreSQL, which integrates with Kafka Connect to build real-time data pipelines.

Physical vs Logical Replication in PostgreSQL

PostgreSQL supports two core types of replication: Physical (Streaming) and Logical.

1. Physical (Streaming) Replication

- Streams raw WAL segments to a replica server.
- Maintains a byte-for-byte copy of the primary.
- Used for:
- High availability
- Read scaling (hot standby)
- Requires same PostgreSQL version and similar configurations.
- Reference: PostgreSQL Physical Replication Guide

2. Logical Replication (Publication/Subscription)

- Introduced in PostgreSQL 10.
- Uses WAL decoding to send row-level changes.
- You define a:
- Publication on the source
- Subscription on the target

Benefits:

- Table-level granularity
- Cross-version replication
- Heterogeneous targets

Limitation:

- Only supports DML (INSERT, UPDATE, DELETE) — not DDL (schema changes).

Example :

Scenario: While setting up provivisoned Postgresql cluster , we wanted to copy from Serveleess based Postgresql cluster, we wanted to sync the data so used the following replication

```
on Source DB:
CREATE PUBLICATION replication_publication FOR TABLE
   <schema1>.<table_name>,
 <schema2>.<table_name2>

on Target DB:
CREATE SUBSCRIPTION replication_subscription
CONNECTION 'host={sourceEndpoint} port={port} dbname={dbname} user={db_user}
password={password}'
PUBLICATION replication_publication
WITH (create_slot = true, enabled = false, copy_data = true);
ALTER SUBSCRIPTION replication_subscription ENABLE;

Cancelling the publication
Target db :
ALTER SUBSCRIPTION replication_subscription DISABLE;
drop SUBSCRIPTION replication_subscription

on Source DB:
SELECT pg_drop_replication_slot('replication_subscription');
OR
drop PUBLICATION replication_publication
```

WAL-Based Replication vs AWS DMS

AWS Database Migration Service (DMS) is a cloud-native service for migrating and replicating data across heterogeneous systems (e.g., PostgreSQL → Redshift, MySQL → Kafka).

How DMS Uses WAL:

- DMS can be configured with a logical replication slot.

- It uses plugins like pgoutput or test_decoding to read WAL and emit structured changes.
- Supports full-load + incremental CDC.

Further reading:

- DMS PostgreSQL Source Docs
- Logical Decoding with DMS

The Rise of Zero-ETL and Its Relationship

Zero-ETL represents a cloud-native evolution of CDC, offering seamless, near real-time integration between OLTP and analytical systems — without the manual effort of building pipelines.

Example: Amazon Aurora Zero-ETL integration with Amazon Redshift

These systems:

- Internally use WAL or binlog-based decoding under the hood.
- Push data directly to analytical stores like Redshift in near real time.
- Offer out-of-the-box reliability, scaling, and schema evolution handling.
- Check this for More details on how Aurora Zero-ETL works

Traditional Logical Replication (WAL-based)

- WAL decoding runs inside the DB engine, adding load.
- Shared WAL storage causes I/O contention.
- Single-threaded, limited parallelism.
- Filtering happens after decoding, increasing latency.

Aurora Zero-ETL Improvements

- Dual WAL streams:
- One for regular PostgreSQL/Aurora functions (recovery, HA, replicas).
- One dedicated for Zero-ETL, stored in a custom, optimized layer.
- WAL decoding offloaded from the DB engine → no CPU/memory contention.
- Pushdown filtering/sorting at the storage layer.
- Parallel, scalable CDC processing optimized for Redshift.