Logical Decoding

Presented to:

Dave Cramer
October 2018

Crunchy Data

Leading provider of trusted open source PostgreSQL technology, support and training.

Powering Innovation With The World's Most Advanced Open Source Database

CRUNCHY Enterprise PostgreSQL

TL;DR Logical Decoding

- Change data capture built in
- Insert, update, delete captured
- Changes are streamed *after* the transaction commits
- Third party change data capture (debezium)

Overview

- Uses Write Ahead Log mechanism to capture the changes
- Each row that is changed is sent to the output plugin
- Unlogged and temp tables do not go through WAL so not decoded

Overview

- DDL is not decoded
- Must have a user with replication permissions
- Each replication stream corresponds to a replication slot. Slots provide fine grain control over each replication stream.
- Potential security issues. Replication user sees all changes

Plugins

- The API provides the data in a binary format
- The output plugin decodes the binary into a format usable by external services
- Output plugins are written in C as shared libraries

Plugins

- Default plugin is test_decoding
- JSON decoder Wal2json https://github.com/eulerto/wal2json
- Protobuf decoder https://github.com/xstevens/decoderbufs

Change Data Capture

- Typically done by some combination of triggers, log tables and polling
- Triggers on tables insert changes into a “log table”
- Log table is polled for new data
- Causes bloat in the database, lots of inserts and deletes

Change Data Capture

- Changes are pushed by the server.
- Server does not delete the change until it is confirmed
- Can do logical replication of a single database, and or table.

Server requirements

- Configuration
  - wal_level = logical (at least logical)
  - max_replication_slots = some number greater than 0
  - synchronous_commit = on (if you want synchronous replication)
- have to have a replication user, and configure pg_hba.conf

SQL functions

- pg_create_logical_replication_slot(‘slot_name’, ‘decoder_name’, [temporary boolean])
  - Creates a slot and associates the slot with the decoder
  - As of pg10 can be temporary
- pg_logical_slot_get_changes(‘slot_name’, null, null, null)
  - Get any changes from the last request for changes

SQL functions

- There are get and peek functions
- Also ways to control the output format
  - Include-timestamp
  - Include-xid
- Control the number of changes, with upto_lsn, and n_entries

Care and feeding of slots

- It’s important to drop the slot when finished with it
  - pg_drop_replication_slot(‘slot_name’)
- The server will keep all the WAL associated with that slot unless the WAL are consumed and confirmed
- Can fill up your server and cause postgresql to stop.

Monitoring

- pg_replication_slots
  - Slot name, plugin, type, database
  - Active, active pid
  - Xmin oldest tx the db needs to retain (can’t be vacuumed)
  - Restart and confirmed LSN(Log Sequence Number) or position in the WAL.

Monitoring

- pg_stat_replication
  https://www.postgresql.org/docs/10/static/monitoring-stats.html#PG-STAT-REPLICATION-VIEW
  - State of replication
  - Lag write, flush, replay
  - Location various LSN's sent, write, flush, replay
  - Sync state async, sync, quorum
  - Write, flush and replay
    - Written but not flushed to disk or applied
    - Written and flushed but not applied

Replica Identity

- Determines how much data is written to the WAL on UPDATE or DELETE
  - DEFAULT: old data are only written to the WAL when the PK is changed
  - FULL: old data are always written
  - NOTHING: old data is never written
  - USING INDEX: old data is written if the column in the index is updated
    - Index needs to be unique, and cannot contain null values, or expressions

JDBC support

- Logical replication is built in to the JDBC driver
- Same requirements. Replication user, wal_level, etc

Steps to setup

- Create a slot, this just uses the **pg_create_logical_replication_slot function**
- Open a replication connection.
  - Must be 9.4
  - Simple query mode
  - Replication mode is ‘database’ (this means logical replication)
- Get a replication stream
- Read the changes
- Demo

THANK YOU!

Dave Cramer
dave.cramer@crunchydata.ca