THE PULSE OF POSTGRES

EDB® WEBINAR SERIES

EDB POSTGRES

Pulse of Postgres

Installing Postgres in Containers	September 25
Installing Postgres on Windows	October 9
JSON and JSONPATH in Postgres 12	October 23
Programming the SQL Way with Common Table Expressions	November 6
EDB Postgres Kubernetes Operator	November 20
Logical Replication in PostgreSQL	December 18

Logical Replication in PostgreSQL

A step-by-step approach

THE PULSE OF POSTGRES

EDB® WEBINAR SERIES

Kuntal Ghosh, Senior Software Engineer
Marc Linster, SVP, Product Development

Agenda

- Who is EDB?
- Logical Replication in PostgreSQL
  - Basic architecture including the publisher and subscriber model
  - Configuration, administration, and monitoring
  - Limitations and future plans
- Q & A

Get your EDB Socks!

Ask a question and give us your business address in the US - and we will send you a pair of EDB Socks!

WHO IS EDB?

A global leader in open-source based Postgres software and services

- **Founded in 2004**
- **Recognized RDBMS leader by:**
  - Gartner
  - Forrester
- **Customer base > 4000**
- **300+ employees**
- **Offices worldwide**
- **Major PostgreSQL community contributor**

ONLY OPEN SOURCE BASED RDBMS IN GARTNER MQ

EDB Recognized 7 Years In A Row on Gartner's Magic Quadrant

OVER 4,000 CUSTOMERS

Customers working SMARTER, reducing RISK and being more PRODUCTIVE with EDB.

102

of the Fortune 500

337

of the Forbes Global 2000

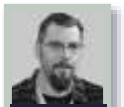
EDB POSTGRES

EDB OPEN SOURCE LEADERSHIP

NAMED EDB OPEN SOURCE COMMITTERS AND CONTRIBUTORS

✦ - *designates committers*

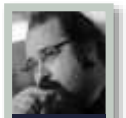

Bruce Momjian


Dave Page

CORE TEAM


Robert Haas


Amit Kapila


Devrim Gündüz

MAJOR CONTRIBUTORS


Akshay Joshi


Amul Sul


Ashesh Vashi


Ashutosh Sharma


Dilip Kumar


Jeevan Ladhe


Mithun Cy


Rushabh Lathia

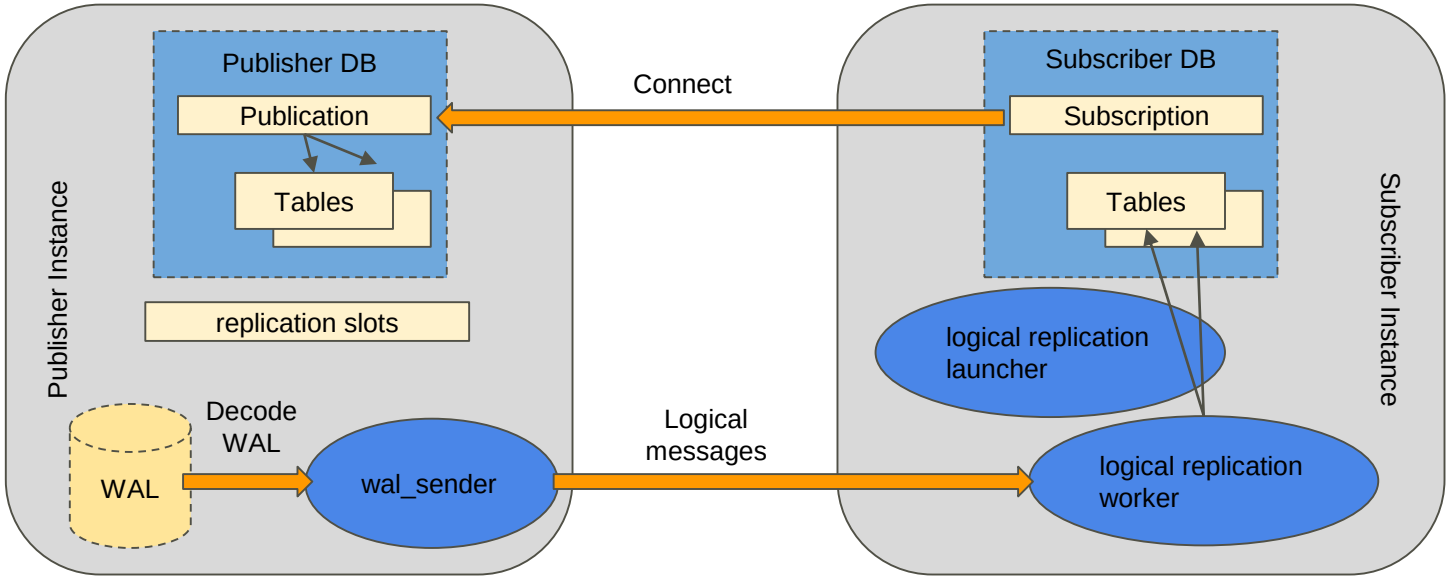
CONTRIBUTORS

Logical Replication in PostgreSQL

What is logical replication?

- Introduced in PostgreSQL 10
- Replicate part of the database
- Online upgrade across major versions of PostgreSQL
- Write access to replicated data
- Auditing
- Allowing access to different set of users to different part of the replicated data
- Cross-platform replication

The architecture



The architecture

- Create publication (pub)
- Create subscription (sub)
  - Create replication slot (internally on pub)

- Table synchronization phase
  - Spawns table synchronization workers
  - COPY table and send changes to sub
  - Logical workers applies changes to sub

- wal_sender in publisher decodes the changes from WAL and send the changes to subscriber.
- logical replication workers in subscriber apply the changes in subscribed tables.

- Wal sender process
  - One per each subscription
  - Decode WAL messages
- Logical replication launcher
  - Launches logical replication workers
- Logical replication worker
  - One per each subscription
  - Receives decoded WAL messages from Wal sender and applies it

Replication criteria

- The replicated table must exist in both publisher and subscriber
  - Use pg_dump to dump the schema from publisher and create the table in subscriber
- Same table name in publisher and subscriber
- Same column name
  - Column order doesn’t matter
  - Subscriber can have more number of columns in the replicated table
- Column type
  - Same column type
  - Different types are allowed if they can be converted implicitly

Quick setup

- Create a replication user and configure pg_hba.conf to allow replication
  - host all repuser 0.0.0.0/0 md5
- Configuration in publisher instance:
  - wal_level = logical
- Create publication in publisher instance:
  - CREATE PUBLICATION my_pub FOR TABLE t1;
- Grant SELECT privileges on published table to the replication user
- Create target table in subscriber
- Create subscription in subscriber instance:
  - CREATE SUBSCRIPTION my_sub CONNECTION 'dbname=postgres host=localhost user=repuser' PUBLICATION my_pub;

Monitoring

- pg_stat_replication
- pg_replication_slots
- pg_stat_subscription
- pg_stat_activity

Let’s try it!

Publications

- CREATE PUBLICATION my_pub1 FOR TABLE t1, t2;
  - Will create a publication for multiple tables
- CREATE PUBLICATION my_pub2 FOR ALL TABLES;
  - Will create a publication for all existing tables and tables to be created in future
  - Must be superuser
- CREATE PUBLICATION my_pub3 FOR TABLE t3 WITH (publish = 'insert,'update','delete');
  - Users can specify which actions should be logically replicated for a table

Publications

- ALTER PUBLICATION my_pub1 ADD/SET/DROP TABLE t4, t5;
  - User can either add multiple tables or modify the table list or drop multiple tables from a publication
- DROP PUBLICATION my_pub1;
  - Drop a single publication
- DROP PUBLICATION my_pub2, my_pub3;
  - Drop multiple publications

Subscriptions

- CREATE SUBSCRIPTION my_sub1 CONNECTION 'host=localhost dbname=postgres user=repuser' PUBLICATION my_pub1, my_pub2;
  - User can create a single subscription for multiple publications

Subscriptions

- CREATE SUBSCRIPTION my_sub1 CONNECTION '...' PUBLICATION '…' WITH (enabled = false, create_slot = false, slot_name = 'slot_1', copy_data = false, synchronous_commit = on);
  - (enabled = false) : Subscription will be disabled initially
  - (create_slot = false) : Replication slot will not be created for the subscription
    - Used by pg_dump while dumping subscriptions
  - slot_name : user can specify the slot name
  - (copy_data = false) : The initial table synchronization step will be skipped
  - (synchronous_commit = on) : the publisher will wait for the actual flush of the logically replicated data in subscriber

Subscriptions

- ALTER SUBSCRIPTION my_sub1 ENABLE/DISABLE;
- ALTER SUBSCRIPTION my_sub1 CONNECTION 'host=newhost ...';
- ALTER SUBSCRIPTION my_sub1 SET (slot_name = 'newslot', ...);
- ALTER SUBSCRIPTION my_sub1 SET (slot_name = NONE);

Subscriptions

- ALTER SUBSCRIPTION my_sub1 SET PUBLICATION my_pub1 REFRESH false;
  - Publication can be altered for a subscription
  - When REFRESH = false, the command will not try to refresh table information
  - To fetch table information, user has to refresh the publication explicitly
  - By default, REFRESH = true
- ALTER SUBSCRIPTION my_sub1 REFRESH PUBLICATION;

Subscriptions

- DROP SUBSCRIPTION my_sub1;
  - Drop a single subscription
- DROP SUBSCRIPTION my_sub1, my_sub2;
  - Drop multiple subscriptions

Security

- The role used for the replication connection must have the REPLICATION attribute (or be a superuser).
- To copy the initial table data, the replication role must have SELECT privilege on the published table (or be a superuser)
- To create a publication, the user must have CREATE privilege in the database.
- To add tables to a publication, the user must have ownership rights on the table.
- To publish all the tables automatically, the user must be a superuser.
- To create a subscription, the user must be a superuser.

Replica Identity

- Controls information written to WAL to identify tuple data that is being deleted or updated
- Four modes
  - DEFAULT
  - USING INDEX index
  - FULL
  - NOTHING
- Can only be modified using ALTER TABLE

Replica Identity

- DEFAULT
  - Identified with the primary key of the table
  - table public.t1: UPDATE: old-key: a[integer]:1 new-tuple: a[integer]:2 b[integer]:1
- USING INDEX index
  - Identified by the specified index entry
  - Index should be UNIQUE and must contain NOT NULL columns

Replica Identity

- FULL
  - Entire old row is written in WAL
  - Most resource consuming
  - table public.t1: UPDATE: old-key: a[integer]:1 b[integer]:1 new-tuple: a[integer]:2 b[integer]:1
- NOTHING
  - Only contains the new row
  - table public.aa: UPDATE: a[integer]:2 b[integer]:1

When replica identity is set to NOTHING or primary key is not defined with DEFAULT replica identity, UPDATE/DELETE throws following error:
ERROR: cannot delete from table ‘t1’ because it does not have a replica identity and publishes deletes
Hint: To enable deleting from the table, set REPLICA IDENTITY using ALTER TABLE.

Restrictions

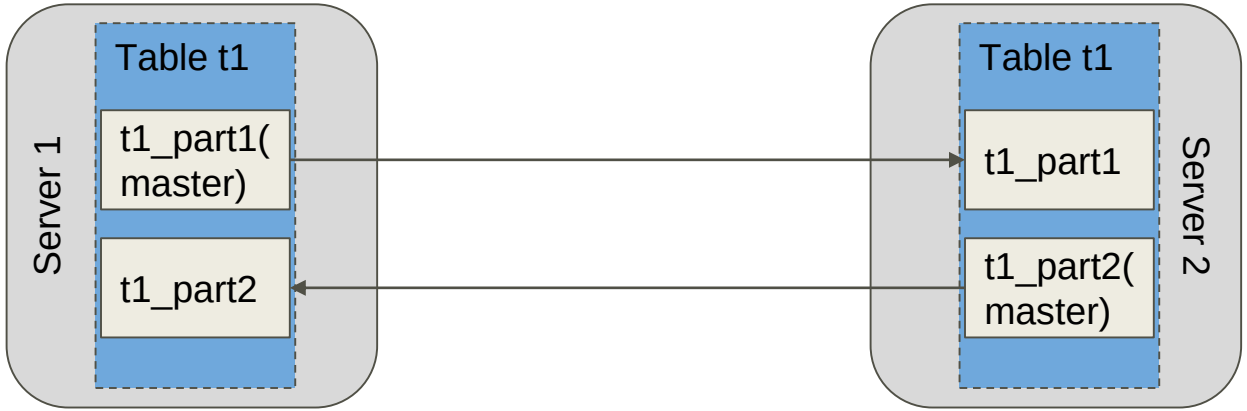
- The database schema and DDL definitions cannot be replicated
  - Initial schema can be copied by hand using pg_dump --schema-only
- The replicated table has to be a regular table
  - Not views
  - Not materialized views
  - Not foreign tables
  - Not partition root tables (patch submitted)

Restrictions

- Large objects are not replicated
- Sequence data is not replicated
  - The data in serial or identity columns backed by sequences will be replicated
  - But, the sequence itself would only reflect the start value

Restrictions

- Bi-directional replication is not (yet) supported
- Different partitions of a partitioned table can be published from different servers



Configuration parameters

- wal_level = logical (PUBLICATION)
- max_replication_slots (PUBLICATION)
  - Must be set to at least the number of subscriptions expected to connect
  - Some reserved for table synchronization
  - Increase the value in case of the following error:
    - ERROR: could not create replication slot "my_sub": ERROR: all replication slots are in use
- max_wal_senders (PUBLICATION)
  - Must be set to at least max_replication_slots plus number of physical replicas
  - Increase the values in case of the following error:
    - FATAL: number of requested standby connections exceeds max_wal_senders (currently 2)

Configuration parameters

- max_logical_replication_workers (SUBSCRIPTION)
  - at least the number of subscriptions, again plus some reserve for the table synchronization
  - Increase the values in case of the following error:
    - WARNING: out of logical replication worker slots HINT: You might need to increase max_logical_replication_workers
- max_worker_processes (SUBSCRIPTION)
  - Must be adjusted to accommodate for replication workers
  - At least set to (max_logical_replication_workers + 1)

Future scope

- Bi-directional replication/Multi-master setup
- Reduce replication lag in case of long-running transactions:
  - The changes are decoded and sent to subscriber only when the main transaction commits
  - Sudden bulk transfer in network
  - Logical decoding of in-progress transactions (patch submitted)
- Logical replication of DDL statements
- Improved security

THE PULSE
OF POSTGRES

EDB® WEBINAR SERIES

Q & A