EDB™

EDB Postgres Distributed
The Next Generation of Postgres High Availability

AUTHORED BY:

Marc Linster
VP Solutions Architecture

Kelly Poole
VP, Product Management

Petr Jelinek
VP, Chief Architect

POWER TO POSTGRES

Contents

Introduction

Over the last five years, the definition of High Availability (HA) has changed. HA used to refer to technology protecting users from hardware failures, network glitches, and software faults. Today, HA technology makes sure that software services are always on—365 days a year, 24 hours a day. HA products still protect users from failures, but as hardware, networks, power supplies, and storage devices have become much more reliable, near-zero downtime maintenance and management have moved to the forefront of the HA debate. Near-zero downtime, or "Always On," has become a must-have for successful digital transformation in a global economy.

EDB Postgres Distributed (PGD), previously known as Postgres-BDR or BDR, has a wide field of possible applications, such as master data management, sharding, and data distribution. In this whitepaper, we only focus on high availability, where PGD provides unparalleled protection from infrastructure issues, combined with near zero-downtime maintenance and management capabilities. The PGD Always On architecture for Postgres has become the industry-leading solution for highly available Postgres.

This white paper introduces the PGD technology and the EDB Postgres Distributed Always On architecture.

1. Overview of High Availability in Postgres

Overview of High Availability in Postgres

There are two fundamental approaches to high availability (HA) in ACID-compliant database systems: (1) shared storage with interconnected memory and (2) database replication. Oracle Real Application Cluster (RAC) is an example of the shared storage approach, which often requires expensive hardware to provide good performance. Postgres uses a replication method, which allows Postgres HA solutions to run on commodity hardware and in every cloud.

Traditional Postgres HA architectures leverage physical streaming replication, a binary mechanism in which a primary server continuously sends database transaction logs, or write ahead logs (WAL), to a standby server. This ensures that the standby has a copy of the transactions the primary has executed. In case of a failure on the primary server, the standby could easily be promoted to become an active primary and resume the service. **Failover Manager from EDB** and **repmgr** are industry-leading technologies supporting up to 99.99% availability for Postgres.

HA with streaming replication is limited in two key ways: (1) in case of failure, it takes a minimum of 20-30 seconds to identify and verify the failure, fence the old primary, and promote the standby to become the new primary because physical replication only supports a single primary node at a time; (2) the primary server and the replica need to be binary compatible, which means they must be on the same major Postgres version, which in turn requires that both the primary and standby be taken offline together during a major version upgrade. Streaming replication also does not help execute other maintenance operations, such as reindexing, without downtime. These limitations constrain streaming replication to applications that require 99.99% of availability and can have planned downtimes for major upgrades.

HA based on logical multi-master replication changes the paradigm. In this paradigm : (1) each member of an HA cluster can accept transactions at any time, so that in the case of a failure of the current primary, the application can immediately start transacting on another server without first waiting for the cluster infrastructure to ascertain the failure and promote a replica; (2) logical replication supports Postgres servers of different major versions, which means rolling upgrades are possible without ever shutting down the service. While logical replication has other advantages, these two capabilities allow EDB Postgres Distributed to use logical replication to create true Always On architectures that can achieve up to 99.999% availability.

2. Replication in Postgres

Replication in Postgres

2.1 History of replication in Postgres

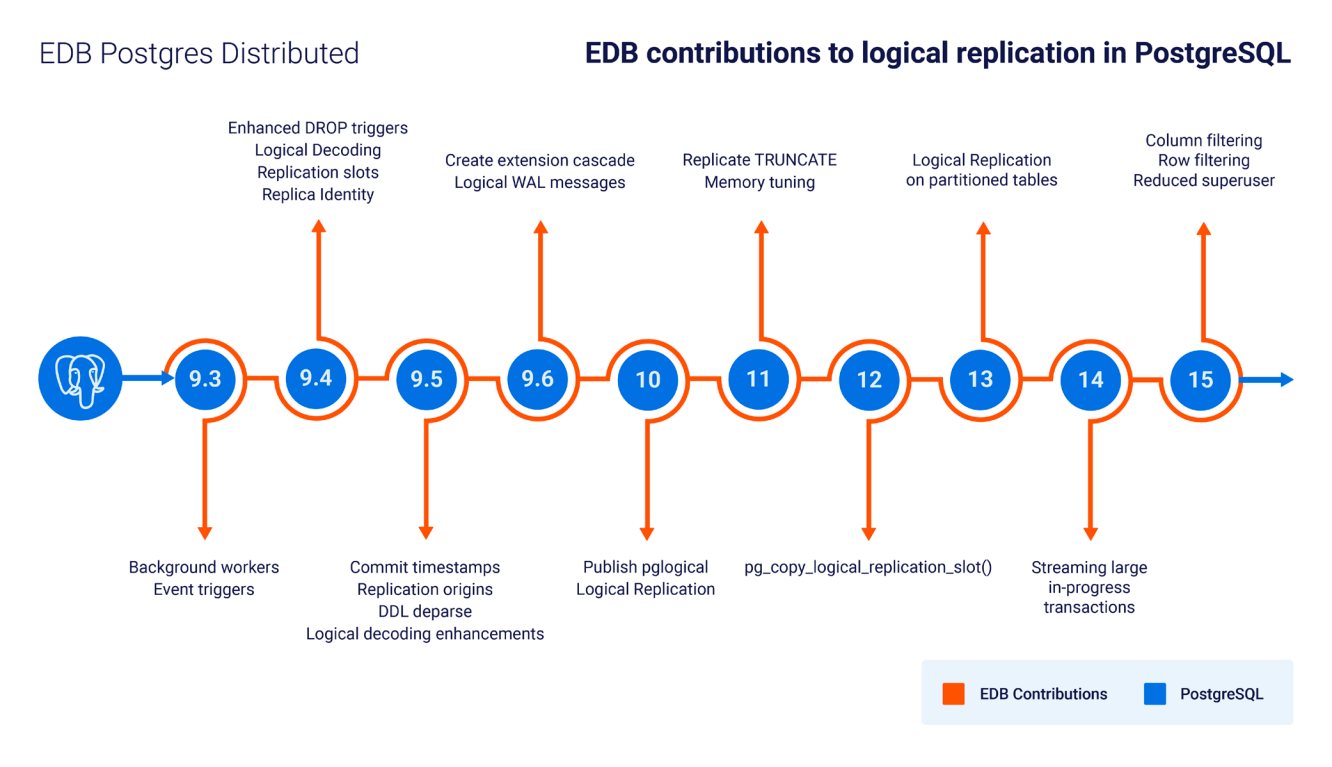

Figure 1: EDB Contributions to logical replication in PostgreSQL

The very first replication and high availability solutions for Postgres, Londiste and Slony, were external projects that used trigger-based logical replication.

Postgres streaming replication was introduced in Postgres 9.0 to improve the performance and reduce the latency of transaction log-shipping, which had been introduced in 8.2. Streaming replication also reduced the extent of data loss in case of a failure of the primary database server. Postgres 9.1 introduced hot standby feedback and synchronous replication.

Subsequently, this infrastructure was built out to add the foundational capabilities for robust logical replication. For example: Event Triggers that fired on DDL statements, replication slots to coordinate activity on streaming standbys, replication of truncate statements, support of Logical Replication on Partitioned Tables, and so on. The EDB team contributed and/or committed most of these features from the PGD project for the PostgreSQL community.

2.2 The Limitations of PostgreSQL Logical Replication for HA

PostgreSQL Native Logical Replication (PNLR) was introduced in Postgres 10 in 2017. PNLR made major version upgrades possible without trigger-based technologies such as Slony or Londiste, which was a significant improvement over existing replication technology.

However, even in 2023 , there are still fundamental limitations with PNLR when being considered as part of highly available solutions. Examples of such limitations include but are not limited to:

- **DDL operations are not replicated. Operations such as creating and dropping tables need to be performed independently on each node.** Since there is no way to ensure transaction consistency of performing DDL operations, additional maintenance windows, where the database is receiving no database traffic, are required. This detracts from the system's ability to be highly available.

- **In PNLR there is no ability to fail over. If either the source or the target node goes down, during a heavy period of replication traffic, a significant risk exists that the entire replication process for a given set of tables will need to be restarted, and previously replicated data will need to be resent.** Therefore, Postgres servers using PNLR are at greater risk than Postgres physical replicas for having significant data loss when taking over as the primary data source for an application as a result of the current primary data source failing.

- **To properly identify updated and deleted rows during replication, logical replication systems require that each row in a replicated table have a primary key.** The primary key needs to be unique across all nodes in the cluster. PNLR provides no built-in mechanism to ensure that newly created and unique records on different nodes don't have the same primary key. The risk is that completely different records in a cluster are represented as the same record. Creating the assurance of unique primary keys is entirely the responsibility of the application developer. Therefore, for advanced cases such as High Availability, PNLR is error prone, which can result in data divergences and inconsistencies across the cluster.

- **PNLR is not integrated with backup and recovery solutions.**

- **While PNLR is a nice feature, it does not come with best practices and proven architectures for achieving common tasks, such as procedures to use PLNR for upgrades and maintenance operations.** Such operations need to be built and extensively tested for each deployment.

- **PNLR only replicates in one direction.** Therefore, when used in an upgrade scenario, if things go wrong after the upgrade and you encounter an issue, going back to a previous version of the database software is very complex and potentially impossible if not properly planned for in advance.

3. Overcoming PostgreSQL HA limitations with EDB Postgres Distributed

Overcoming PostgreSQL HA limitations with EDB Postgres Distributed

3.1 What is EDB Postgres Distributed?

EDB Postgres Distributed (PGD) is a Postgres database solution that enables Extreme High Availability (EHA)for Postgres clusters with up to 99.999% availability.. It does this using a standard Postgres extension through logical replication of data and schema in a mesh-based multi-master architecture, plus a proxy for directing connections globally and locally, and a robust set of features and tooling to manage conflicts and monitor performance. This means applications with the most stringent demands can be run with confidence on Postgres.

PGD is asynchronous by default, uses a mesh topology, and creates active-active architectures based on Postgres logical replication.

PGD was built by the team that contributed many of the foundational replication capabilities in Postgres. It was designed in conjunction with customers to allow them to replace complex, costly legacy solutions with modern, highly available databases capable of rolling upgrades.

3.2 What is different about EDB Postgres Distributed?

EDB Postgres Distributed takes Postgres replication to the next level. It builds on logical replication

- Automatic DDL and DML replication in an active-active mesh network
- Failover and switchover infrastructure to re-route traffic in case of failures or during maintenance operation
- Advanced conflict detection and conflict management
- Management of distributed sequences
- Differentiated replication sets to control which data gets replicated and to which downstream databases

- Cluster expansion/consolidation
- Rolling database software upgrades
- Rolling schema change/migration using cross-schema replication
- Recovery from user error through solid integration with backup and recovery tools. Improved security model making sure that all changes get replayed with minimal security privileges

EDB Postgres Distributed has a wide field of possible applications, such as master data management, sharding, and data distribution. In this whitepaper, we only focus on EDB Postgres Distributed for high availability, where it delivers several additional features that allow administrators to control the latency/consistency trade-off:

Column-Level Conflict Resolution. Ability to have per column level granularity for conflict resolution so that UPDATEs on different fields can be 'merged' without losing either change.

Conflict-free Replicated Data Types (CRDT). Mathematically proven data types with demonstrated consistency in asynchronous multi-master update scenarios, for when conflicts are expected.

Transform Triggers. Filters, modifies or transforms incoming data with triggers activated on data before they are applied.

Conflict Triggers. Custom conflict resolution techniques that can be implemented with triggers that are activated when a conflict is detected.

Eager Replication. Provides conflict free replication by ensuring that all nodes can commit before allowing the local node to commit.

Commit At Most Once (CAMO). A consistency feature that helps an application understand whether a commit was received or the transaction aborted, even across single node failure. That way an application can be written to be idempotent and need never miss a transaction or commit it more than once, while remaining available most of the time.

Group Commit. Synchronous replication, which requires a user defined quorum within a replication group before committing a transaction.

Rapid node join (bdr_init_physical). Utility to rapidly join nodes into the cluster using physical replication; important for large Postgres databases.

LiveCompare. High speed verification utility to confirm that replication is exactly accurate across nodes, working with multiple Postgres databases, across Postgres cluster nodes and also against Oracle. Designed and integrated with EDB Postgres Distributed for continuous live usage.

4. Always On Architectures

Always On Architectures

Always On architectures for EDB Postgres Distributed are highly standardized, proven, and reliable Postgres architectures for achieving up to 99.999% database availability on premises or in the public cloud. Always On is available in several variations. In this paper we focus on the variation that is designed to provide local HA in two redundant data centers—an active data center and a passive data center—both configured identically. Other variations of the Always On architecture include, but are not limited to:

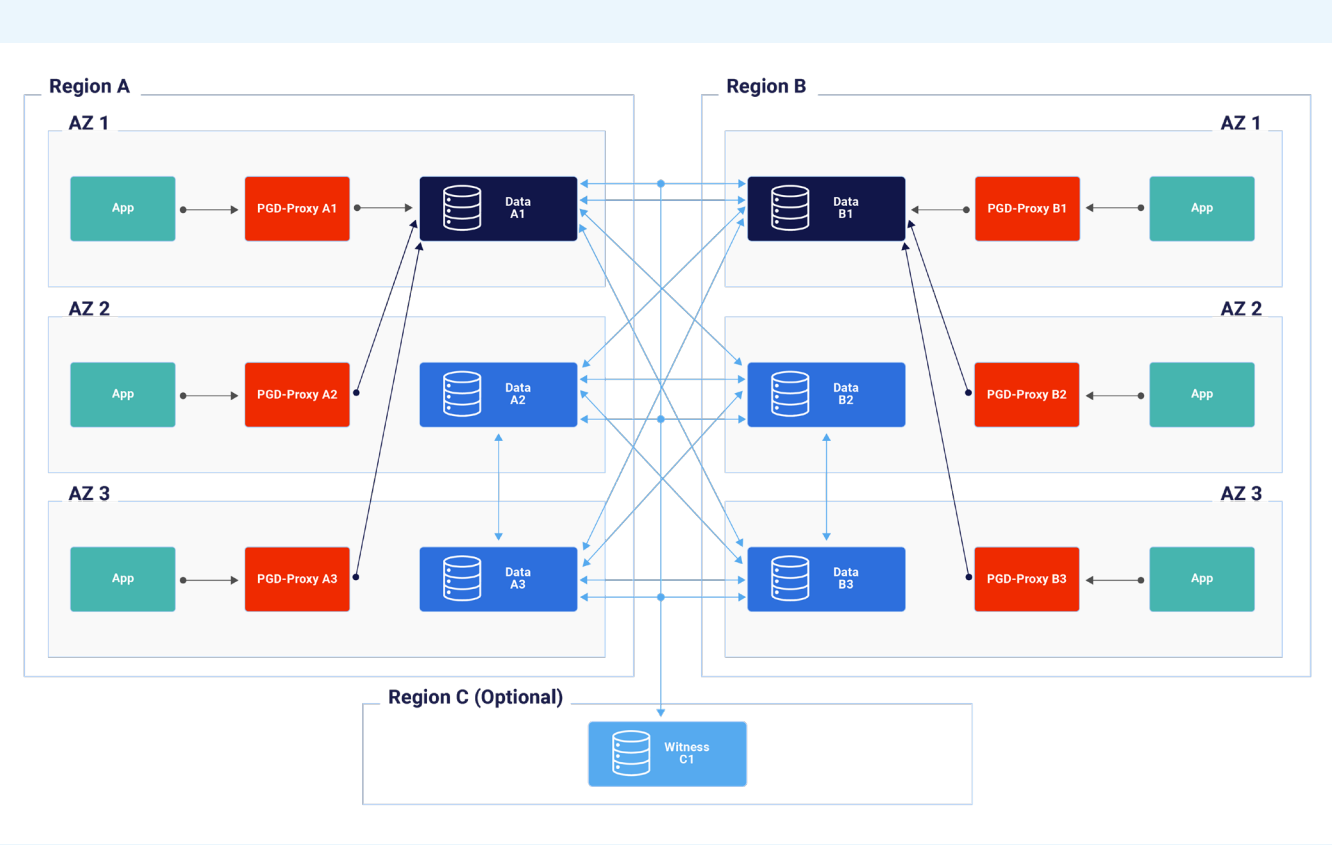

Figure 2: EDB Postgres Distributed EDB Postgres Distributed Always On architecture for local high availability in two data centers locations

4.1 Key components of the EDB Postgres Distributed Always On architecture

Lead Master. A Postgres server in read/write mode with the EDB Postgres Distributed extension installed. The Lead Master receives the write transactions and the read requests from the application. It is in a logical replication relationship with all other PGD nodes.

Shadow Master. The Shadow Master is a read/write enabled Postgres server that does not currently receive traffic from the application, but it continuously replicates from the Lead Master and all other PGD nodes. In case of failure or switchover, the Shadow Master can take over almost immediately for the Lead Master.

Logical Standby. The Logical Standby is a read-only Postgres node that replicates from the Shadow Master. It can be used for offloading of read transactions. Its primary role is to replace one of the masters in case of hardware failure, so that local HA can be re-established quickly.

PGD-Proxy. The PGD-Proxy provides connection routing and makes sure that in case of a failure of the Lead Master, all subsequent database connections are successfully and reliably redirected to the Shadow Master.

pgBouncer. pgBouncer provides connection pooling to allow multiple application instances to connect to the PGD databases. Depending on the usage pattern, it can be configured in session mode or in transaction mode.

Multi-host connection string. Applications use a multi-host connection string to allow for rapid failover to a second PGD-proxy pair in case of a hardware failure.

Barman. The Barman server provides backup and recovery, especially for PITR use cases to protect against operator error or data corruption.

PGD Witness Node. A witness node is positioned in a third location to provide a quorum in case of a data center failure.

Trusted Postgres Architect (TPA). EDB's tool to deploy and operationally manage trusted Postgres architectures, such as Always On.

EDB Postgres Enterprise Manager® can monitor PGD and its components.

4.2 How it works and how it solves problems

EDB Postgres Distributed Always On handles failures, switchovers, and other maintenance operations within the 99.999% EHA boundaries. Here we illustrate a few of the most common scenarios:.

4.2.1 What happens when the Lead Master fails?

PGD consensus detects the failure of the node and switches all traffic at the PGD Proxy level to one of the Shadow Masters, which now becomes the new Lead Master. the Shadow Master. Failure detection and switching of the traffic happens within a few seconds. As the Shadow Master is able to receive update transactions at any time, service resumes almost immediately.

4.2.2 How do I switch the traffic from Lead Master to Shadow Master to execute maintenance operations?

Using Trusted Postgres Architect, the administrator drains all connections on pgBouncer, and then reconnects pgBouncer to the Shadow Master. The use of Trusted Postgres Architect helps to ensure that all EDB Postgres Distributed Always On components are aligned during this operation.

4.2.3 What happens when one of my PGD Proxy nodes fails?

The applications use a multi-host connection string. When the server at the primary address in the connection string fails, then the application can immediately connect to the second PGD Proxy server. PGD makes sure that both PGD Proxy pairs point to the same PGD master and traffic immediately resumes.

4.2.4 What happens when a data center fails?

The management plane of the application will either redirect the application connections to the secondary data center, or—what is generally considered best practice—all inbound traffic is redirected to application servers in the second data center.

4.3 Using EDB Postgres Distributed with EDB's Postgres server distributions

PGD is implemented as a Postgres extension and works with several distributions of PostgreSQL. PGD 3.7.9

and later supports EDB Postgres Advanced Server in Postgres or Oracle compatibility mode, EDB Postgres Extended Server (formerly known as 2ndQuadrant Postgres), and community PostgreSQL.

EDB Postgres Distributed's essential features are available no matter which version of Postgres you use. However, some of the advanced features such as group commit, eager consistency and CAMO (commit at most once) are only available using supported versions of EDB Postgres Extended Server (11-15) and will be available with EDB Postgres Advanced Server (14-15).

The following tables provide further details on both essential and advanced features of PGD available based on the Postgres version used.

Feature	PostgreSQL	Postgres Extended	Postgres Advanced
Supported Version	11-15	11-15	11-15
Rolling application and database upgrades to address the largest source of downtime	✓	✓	✓
Asynchronous replication with origin based row-level last-update wins eventual consistency	✓	✓	✓
Synchronous replication (PostgreSQL core is not conflict-free)	✓	✓	✓
DDL replication with granular locking supports changes to application schema, ideal for use in continuous release environments	✓	✓	✓
Sub-groups with subscribe-only nodes enable data distribution use cases for applications with very high read scaling requirements	✓	✓	✓
Sequence handling provides applications different options for generating unique surrogate ids that are multi-node aware	✓	✓	✓
Tools to monitor operation and verify data consistency	✓	✓	✓
Parallel apply allows multiple writer processes to apply transactions on the downstream node improving throughput up to 5X	✓	✓	✓
Conflict-free replicated data types (CRDTs) provide mathematically proven consistency in asynchronous multi-master update scenarios	✓	✓	✓
Column-level conflict resolution enables per column last-update wins resolution to merge updates	✓	✓	✓
Transform triggers execute on incoming data for modifying or advanced programmatic filtering	✓	✓	✓
Conflict triggers provide custom resolution techniques when a conflict is detected	✓	✓	✓
Eager replication provides conflict free replication by synchronizing across cluster nodes before committing a transaction	✓	✓	✓
Commit at most once (CAMO) consistency guards application transactions even in the presence of node failures	✓	✓	(v14+)
Single decoding worker improves performance on upstream node by doing logical decoding of WAL once instead of for each downstream node	✓	v13+	(v14+)
Tooling to assess applications for distributed database suitability	✓	✓	(v14+)

5. Conclusion

Conclusion

EDB Postgres Distributed, with the Always On architecture, is the industry leading solution for Postgres High Availability. EDB Postgres Distributed's Always On architectures enables customers for the first time to use Postgres for 99.999% EHA availability solutions—a domain that was traditionally reserved for a few select commercial database products.

Learn more about EDB Postgres Distributed

About EDB

EDB provides enterprise-class software and services that enable businesses and governments to harness the full power of Postgres, the world's leading open source database. With offices worldwide, EDB serves more than 1,500 customers, including leading financial services, government, media and communications and information technology organizations. As one of the leading contributors to the vibrant and fast-growing Postgres community, EDB is committed to driving technology innovation. With deep database expertise, EDB ensures extreme high availability, reliability, security, 24x7 global support and advanced professional services, both on premises and in the cloud. This empowers enterprises to control risk, manage costs and scale efficiently. For more information, visit **www.enterprisedb.com**.

EDB Postgres Distributed
The Next Generation of Postgres High Availability

EnterpriseDB Corporation
34 Crosby Drive
Suite 201
Bedford, MA 01730

POWER TO POSTGRES