

CERN PGDay 2026

Friday, 6 February 2026 - Friday, 6 February 2026

CERN

Book of Abstracts

Contents

Closing	1
Opening	1
Operational hazards of managing PostgreSQL DBs over 100TB	1
When Kafka Met Elephant: A Love Story about Fast Ingestion	1
Vacuuming Large Tables: How Recent Postgres Changes Further Enable Mission Critical Workloads	2
A new PostgreSQL backend for CERN Tape Archive scheduling for LHC Run 4	2
The Alchemy of Shared Buffers: Balancing Concurrency and Performance	3
Incremental Backup with PostgreSQL17	3
The (very practical) Postgres Sharding Landscape	4
DCS Data Tools - PostgreSQL/TimescaleDB Implementation for ATLAS DCS Time-Series Data	4

29

Closing

Author: Markus Wanner^{None}

Co-authors: Maurizio De Giorgi¹; Tobias Bussmann²

¹ *CERN*

² *Swiss PostgreSQL Users Group*

Closing notes with sponsors review and special thanks

59

Opening

Author: Markus Wanner^{None}

Co-authors: Maurizio De Giorgi¹; Tobias Bussmann²

¹ *CERN*

² *Swiss PostgreSQL Users Group*

Welcome from the organizing team, logistics info and notices, sponsors presentation and special thanks

63

Operational hazards of managing PostgreSQL DBs over 100TB

Author: Teresa Lopes¹

¹ *Adyen*

Picture this: you start a new role, eager to learn and contribute with your ideas! Your next task is to get familiar with the database setup, and then you start encountering these massive PostgreSQL databases —100TB, 200TB, 300TB...

And you start questioning yourself: how do you backup (and restore) a +100TB database? And how about HA? Performance? Vacuum?

It should work the same way as for a 100GB database, right? Well, maybe not exactly.

Blog posts and best practice guides make PostgreSQL seem straightforward—until you push it to its limits. At extreme scale, you will find yourself questioning the most fundamental assumptions about how PostgreSQL works.

Over the last years, my team at Adyen has been exploring the boundaries of what PostgreSQL can do, and today I will share our findings with you (at least the ones I can!).

69

When Kafka Met Elephant: A Love Story about Fast Ingestion

Authors: Barbora Linhartova^{None}, Jan Suchanek^{None}

What happens when a stream-obsessed writer meets a data-loving elephant? You get a fast, open, and surprisingly elegant story about modern data ingestion.

This talk shows how PostgreSQL 18 can act as the central hub of an open Data Mesh, powered by Kafka and JDBC Sink connectors. We'll explore how new features like Asynchronous I/O, JSON_TABLE, and parallel COPY dramatically boost ingest speed —and how pg_duckdb brings lightning-fast analytics directly inside PostgreSQL.

Expect practical architecture examples, performance results, and a clear takeaway: when Kafka met the Elephant, streaming finally became simple —and fast.

75

Vacuuming Large Tables: How Recent Postgres Changes Further Enable Mission Critical Workloads

Author: Robert Treat^{None}

We have all heard about Postgres vacuum horror stories and tales of transaction wraparound disasters. Even if you've never been through one yourself, you may be concerned that you might someday experience it, or maybe you even know people who have avoided Postgres altogether due to fear of it happening to them. But it doesn't have to be this way.

In this talk, we will explore this topic through a review of a real-world wraparound incident, walking through the challenges and hope offered by changes in the last several Postgres releases. We'll explore the inner workings of the Postgres vacuum process, its dual purpose of managing both disk bloat and transaction IDs (XIDs), and how recent innovations have improved this process, particularly for large tables with heavy transaction loads. We'll dive deep into key features like: - How index de-duplication minimizes vacuum workloads and improves efficiency. - How on/off index vacuuming gives you better control during maintenance windows. - Why one key autovacuum enhancement may be the key for eliminating XID wraparounds.

Beyond theory, we'll revisit the real-world incident and see how each new Postgres feature could have helped mitigate the impact. While no single change is a silver bullet, the combined effect is significant. We'll also explore how the radix tree implementation in Postgres 17 might be the possible final piece of the puzzle for providing a manageable solution to XID wraparound risks.

This talk equips DBAs with a deeper understanding of recent Postgres advancements and their practical application in managing high-transactional and/or mission-critical workloads. You'll leave with actionable insights to optimize your Postgres environment and concrete arguments to convince management that it is worth the upgrade for the potential to minimize the risk of XID wraparound incidents.

84

A new PostgreSQL backend for CERN Tape Archive scheduling for LHC Run 4

Author: Jaroslav Guenther¹

Co-author: Konstantina Skovola

¹ CERN

The CERN Tape Archive (CTA) stores over one exabyte of scientific data. To orchestrate storage operations (archival) and access operations (retrieval), the CTA Scheduler coordinates concurrent data movements across hundreds of tape servers, relying on a Scheduler Database (Scheduler DB) to manage the metadata of the in-flight requests. The existing objectstore-based design of the CTA Scheduler DB is a complex transactional management system. This talk presents the development of a new PostgreSQL-based backend for the CTA Scheduler as an off-the-shelf solution which simplifies implementation and is expected to significantly reduce future development and operational costs. We describe the implementation of all main CTA workflows and explain how PostgreSQL addresses the limitations of the objectstore-based system, providing the foundation for the tenfold increase in data throughput expected during LHC Run 4.

95

The Alchemy of Shared Buffers: Balancing Concurrency and Performance

Author: Josef Machytka^{None}

Talk dissects PostgreSQL's shared buffers: the three-layer design (buffer pool → BufferDesc headers → buf_table hash), the hit/miss/I-O lifecycle, and how pins, per-page LWLocks, and atomic BM_* flags coordinate page-granular concurrency. We will trace clock-sweep and buffer rings that prevent big scans and VACUUM from polluting the cache, then follow WAL-before-data through bgwriter and the checkpointer to show how writes are made smoother. We will also discuss tuning and what's new in PG 17 (vectored reads), and PG 18 (async reads) and what can come next - we will dive into new design proposals discussed in 2025 - to allow dynamic resize of shared_buffers via segmented shared memory.

Key takeaways

- Architecture & locks: three layers; pins + per-page LWLocks + partitioned mapping yield high concurrency
- Replacement & pollution control: clock-sweep keeps hot pages; buffer rings fence off bulk scans and VACUUM
- Durability & flushing: WAL-before-data with bgwriter/checkpointer turns random writes into managed I/O
- Tuning: balance double buffering, size shared_buffers sensibly, smooth checkpoints, and verify with EXPLAIN (BUFFERS) and pg_stat_bgwriter
- Roadmap: PG18's async reads cut scan latency; dynamic shared_buffers resizing is being prototyped

97

Incremental Backup with PostgreSQL17

Author: Marion Baumgartner^{None}

In this talk, we will explore the powerful incremental backup feature introduced in PostgreSQL 17. Rather than backing up the entire database, incremental backups focus on the changes made since the last backup. This approach significantly reduces time and storage requirements compared to full backups, making incremental backups an efficient, time-saving solution for data backup processes.

The talk will begin by explaining the fundamental concepts of incremental backups. We will then analyse how to handle retention over multiple backup increments. Finally, we will examine the key benefits of incremental backups, such as reduced backup times, lower storage costs and minimized impact on the database performance.

To provide a clear and practical understanding, I will present a series of examples and tests that we have conducted. These will illustrate how to implement and utilize incremental backups, making

the concept easier to grasp.

98

The (very practical) Postgres Sharding Landscape

Author: Álvaro Hernández^{None}

Sharding enables horizontal scaling of Postgres, by logically splitting your database into several subsets (“shards”). Each shard will be a primary or, for high availability purposes, a primary-replica cluster. This allows you to scale writes horizontally, allowing Postgres to reach database sizes of multiple terabytes or even petabytes.

Sharding solutions for Postgres exist in multiple forms, such as embedded into Postgres as an extension, using native Postgres techniques and as external components, including Citus, Apache ShardingSphere, PgDog, native partitions with FDW, and others. This talk will:

- Analyze the evolving landscape of sharding solutions for Postgres.
- Offer advice and recommendations on which one to select.
- Provide open source code for simple demos so you can quickly evaluate them in your own environment.
- Present a real-world case study: planning the migration of a 1PB+ workload to sharded Postgres at a major US telco.

If demo Gods will allow, demos of sharding Postgres will be performed live.

113

DCS Data Tools - PostgreSQL/TimescaleDB Implementation for ATLAS DCS Time-Series Data

Author: Dimitrios Matakias¹

Co-author: Paris Moschovakos

¹ CERN

The ATLAS Detector Control System manages over 68 TB of time-series data accumulated since 2007. This presentation describes the practical implementation and operational deployment of TimescaleDB—a PostgreSQL extension—to modernize DCS data access for the ATLAS experiment. We share our experience as PostgreSQL users and administrators implementing a production time-series database solution in a large-scale scientific environment. The implementation delivers significant value through improved user experience: scientists and detector experts now access 18 years of historical data through simple Python APIs, Grafana dashboards, and C++ interfaces, eliminating the need for complex SQL knowledge. TimescaleDB’s native compression achieves 11× storage reduction (68 TB to 6.6 TB), while query performance improvements enable real-time correlation studies across multiple detector systems. The PostgreSQL ecosystem enabled rapid development—Grafana integration, LDAP authentication, and CERN’s Database-on-Demand (DBOD) infrastructure worked seamlessly, accelerating deployment. This PostgreSQL-based approach has unlocked new operational capabilities: detector experts built custom monitoring applications, predictive maintenance workflows identify hardware failures before they impact data collection, and commissioning teams correlate real-time measurements across detector subsystems. The system now serves production users with continuous availability and sub-minute data latency. We discuss practical deployment challenges, infrastructure

lessons learned while working with CERN DBOD team, performance tuning strategies (chunk sizing, WAL management, JIT optimization), and the organizational impact of choosing PostgreSQL's open-source ecosystem for long-term scientific data management.