Testing Multiple Use Cases and Indexing Options for JSONB Data in PostgreSQL

Josef Machytka

3 min read · 5 days ago

This article serves as the introductory text and table of contents for series of my articles publishing details from my project which explored the usage of JSONB data in PostgreSQL. Below, you will find a list of the already published texts (listed from newest to oldest):

- PostgreSQL JSONB Operator Classes of GIN Indexes and Their Usage
- Rebuilding a Hash Index Can Take Ages in PostgreSQL
- PostgreSQL Cost Calculations Seem to Depend on Some Arbitrary Magical Numbers
- TOASTed JSONB data in PostgreSQL: performance tests of different compression algorithms (on NetApp-credativ blog)

Introduction to the Project

JSON documents have become a “must-have” for relational databases. Developers, both frontend and backend, often favor this format for its flexibility, which reduces the need for frequent schema changes and simplifies application updates. In last years JSON has gained immense popularity, even becoming a standardized format for sending data from Web of Things (WoT) devices.

To meet these demands, modern databases must efficiently store, process, and query JSON data. PostgreSQL excels in this area, thanks to its advanced support for JSON, particularly the JSONB data type, and its extensive set of operators and functions. Notably, PostgreSQL 17 introduces the new `JSON_TABLE` function, enabling us to query JSON data and present it in a relational view with relative ease.

In November 2023, I started a long term internal project to continuously investigate JSONB use cases and indexing options of JSONB data in PostgreSQL. This idea was born out of practical challenges faced by our clients, who sought detailed and actionable insights into optimizing JSONB data usage, particularly in the context of indexing and performance.

Why JSONB Challenges Persist

While existing online resources on JSONB and indexes are rather abundant, they often provide only surface-level guidance, frequently only echoing PostgreSQL documentation with simple, very trivial examples. And they often lead to mistaken conclusions about how it all works.

Very typical problem of developers is their insistence on forcing PostgreSQL to use indexes under all circumstances. Many of them attempt to create absolutely "future-proof" solutions without fully understanding the real-world shape of their data, often relying on small, artificially generated and very evenly distributed datasets for testing.

Another widespread issue stems from inadequately configured PostgreSQL development instances, which can drastically influence the behavior of the query planner during testing. Design patterns present another layer of confusion. Developers frequently struggle to decide when to use a single large table versus when partitioning might be a far better performing choice. Therefore my project aimed to bridge these gaps by providing deeper, more practical insights and actionable solutions.

Sharing the Results

The findings from this project were compelling enough to present at two PostgreSQL conferences — P2D2 2024 and Swiss PG Day 2024 — as well as at the Berliner PostgreSQL Meetup in October 2024. These presentations focused on the practical application of indexes and optimization techniques, providing a high-level summary of the project's most significant outcomes.

However, a conference talks offer limited time to dive into the details. To complement these presentations, this series of articles will delve deeper into the results, sharing practical lessons, most interesting insights, and comprehensive analyses. I will also cover new functionality introduced in PostgreSQL 17 and its implications for JSONB data handling.

Looking Ahead

The series will continue to expand as the project evolves, incorporating the latest advancements and findings. Whether you're a developer, database administrator, or simply curious about JSONB in PostgreSQL, I hope these articles provide you with valuable knowledge and actionable strategies.

Image created by the author using DeepDreamGenerator

Postgresql Json Jsonb Gin Index

Edit profile

Written by Josef Machytka

58 Followers · 15 Following

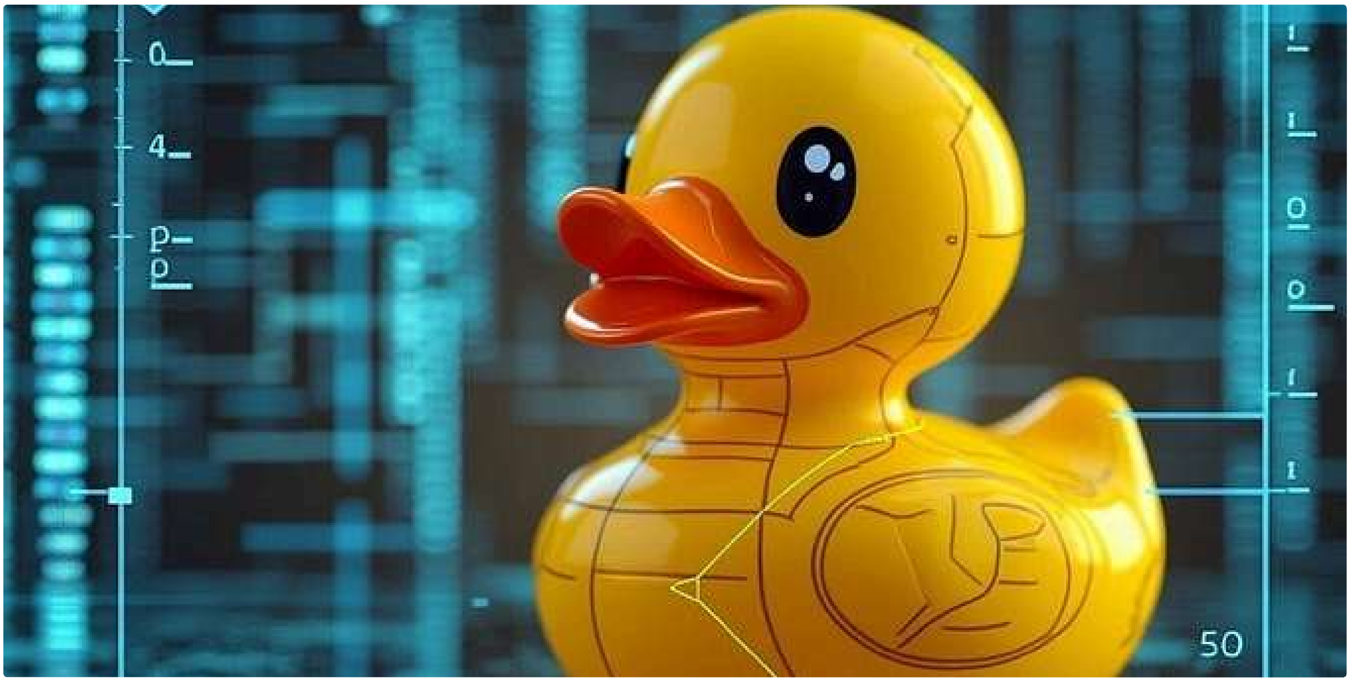
I work as PostgreSQL specialist & database reliability engineer at NetApp Deutschland, Open Source Services division.

No responses yet

What are your thoughts?

Respond

More from Josef Machytka

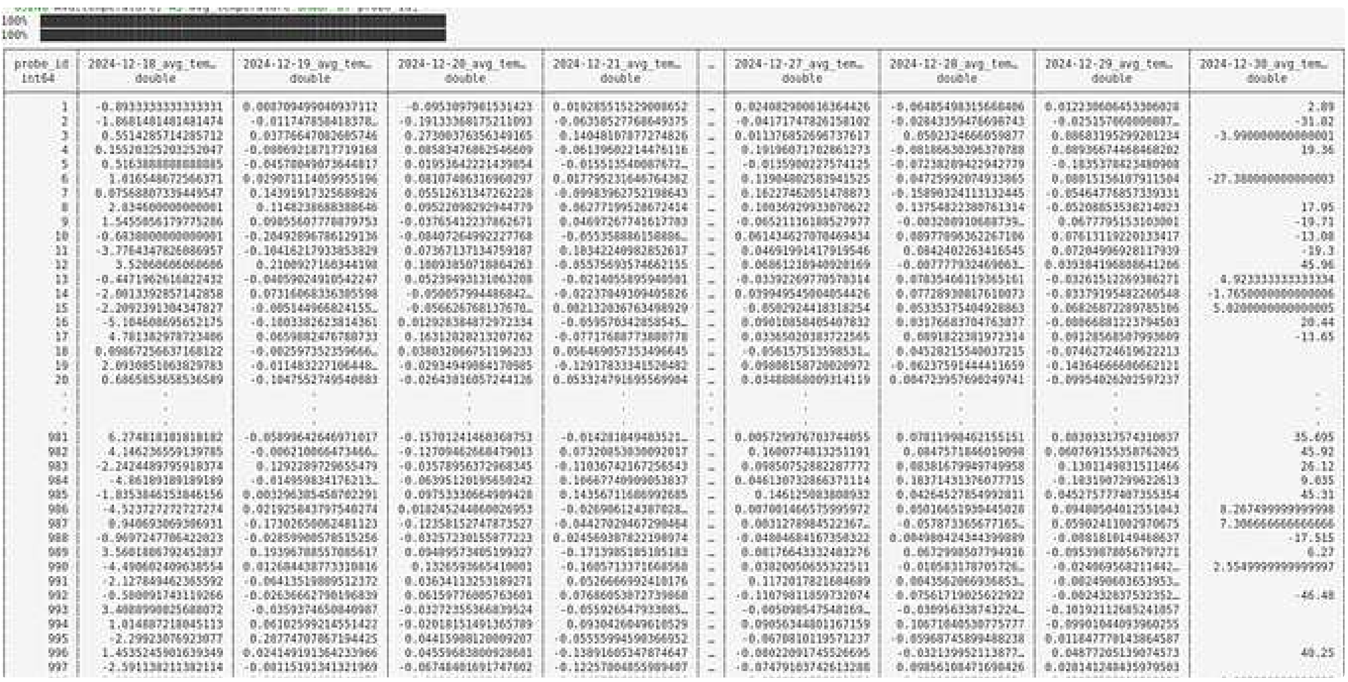


Josef Machytka

DuckDB Database File as a New Standard for Sharing Data?

This is not my original idea; I came across it in an excellent article titled “DuckDB Beyond the Hype” by Alireza Sadeghi. However, it…

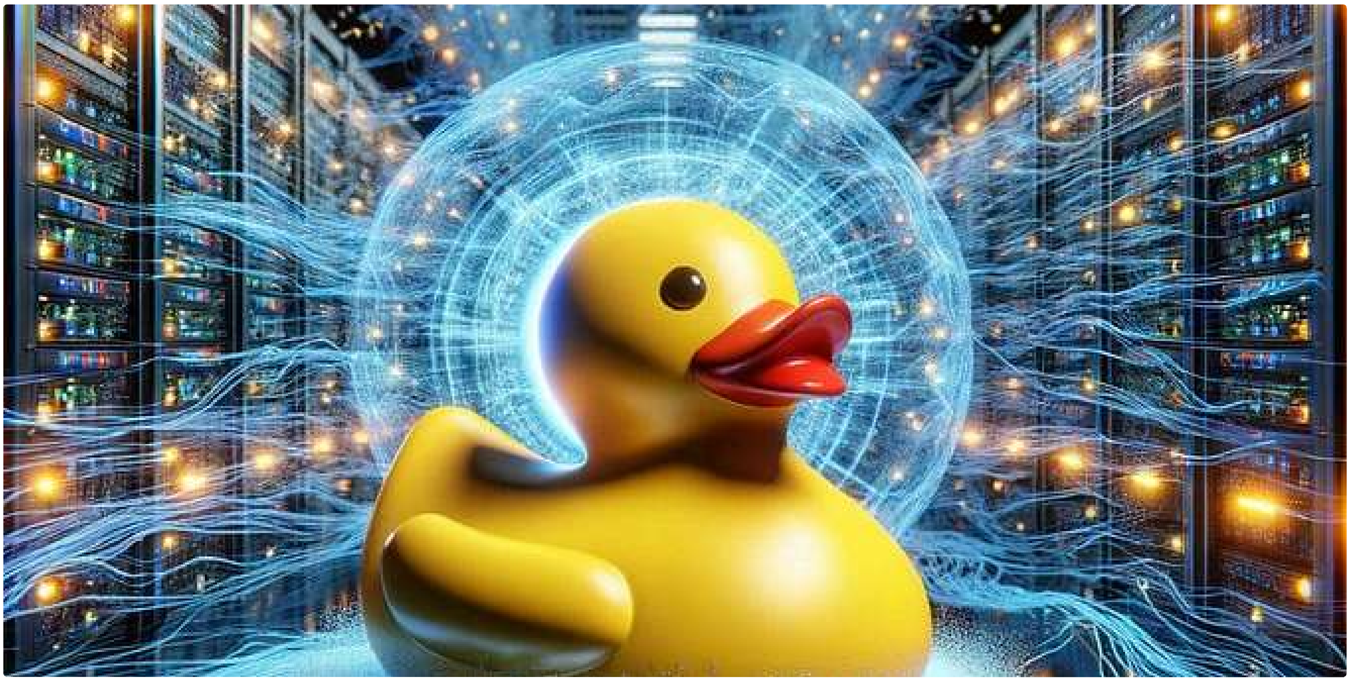
Dec 30, 2024 17 2



Josef Machytka

DuckDB Performance Problems with Inappropriate Pivoting Queries on Very Large Datasets

The DuckDB documentation clearly states that this tool is designed for handling datasets fitting into memory. I fully understand that I'm...



Josef Machytka

Quick and Easy Data Exports to Parquet Format Using DuckDB

The Parquet format has become almost an industry standard for Data Lakes and Data Lakehouses, thanks to its efficiency and compact storage...


```
ate the table
 TABLE special_data_types (
 INT AUTO_INCREMENT PRIMARY KEY,
me VARCHAR(50) NOT NULL,
atus ENUM('active', 'inactive', 'pending') NOT NULL,
rmissions SET('read', 'write', 'execute') NOT NULL,
all_number TINYINT NOT NULL,
dium_number MEDIUMINT NOT NULL,
scription TEXT,
ta BLOB,
eated_at DATE NOT NULL


ert 10 rows of data
 INTO special_data_types (name, status, permissions, small_number, medium_number, description, data, created_at)
e', 'active', 'read,write', 5, 1000, 'Alice description', 'Alice data', '2023-01-01'),
, 'inactive', 'read', 10, 2000, 'Bob description', 'Bob data', '2023-02-01'),
lie', 'pending', 'write,execute', 15, 3000, 'Charlie description', 'Charlie data', '2023-03-01'),
d', 'active', 'read,write,execute', 20, 4000, 'David description', 'David data', '2023-04-01'),
, 'inactive', 'execute', 25, 5000, 'Eve description', 'Eve data', '2023-05-01'),
k', 'pending', 'read,write', 30, 6000, 'Frank description', 'Frank data', '2023-06-01'),
e', 'active', 'read', 35, 7000, 'Grace description', 'Grace data', '2023-07-01'),
', 'inactive', 'write,execute', 40, 8000, 'Hank description', 'Hank data', '2023-08-01'),
, 'pending', 'read,write,execute', 45, 9000, 'Ivy description', 'Ivy data', '2023-09-01'),
', 'active', 'execute', 50, 10000, 'Jack description', 'Jack data', '2023-10-01');
```

Josef Machytka

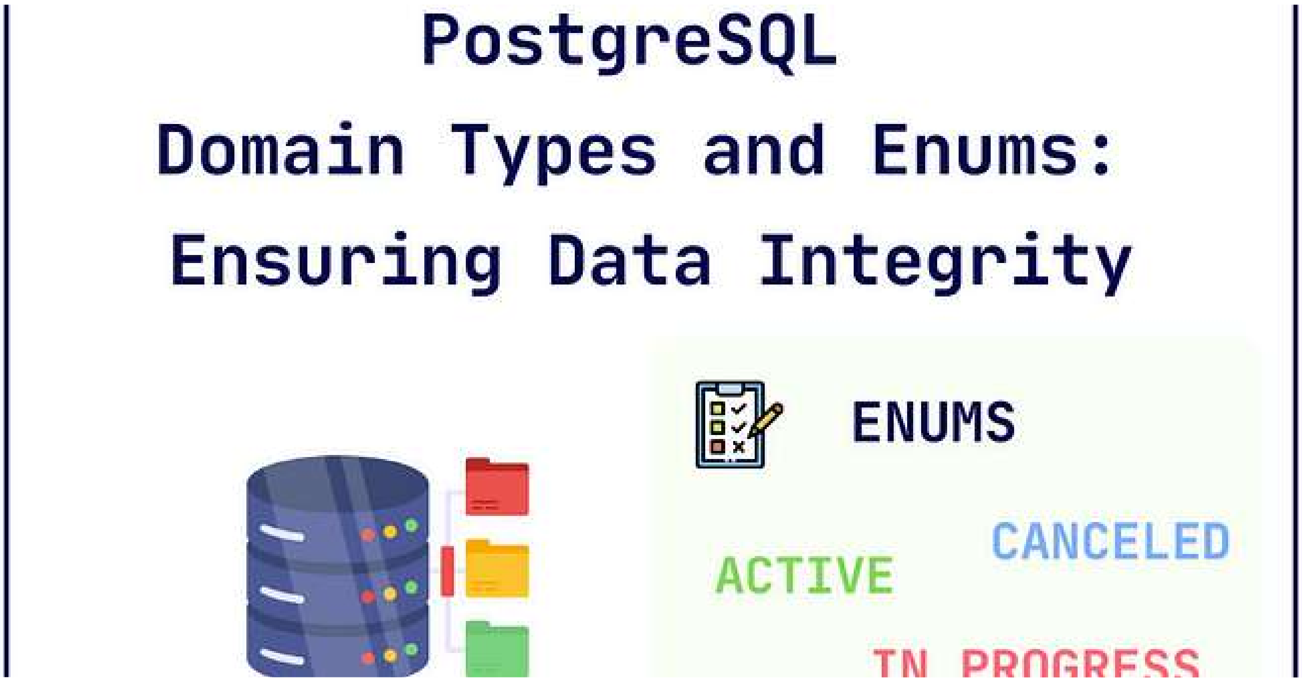
DuckDB as a Rudimentary Data Migration Tool

After exploring how to use DuckDB as an intelligent ETL tool for PostgreSQL, and how to extend its ETL capabilities with simple Python…

Nov 30, 2024 26 2

See all from Josef Machytka

Recommended from Medium



Java Jedi

PostgreSQL Domain Types and Enums: Ensuring Data Integrity

Welcome back to the third post in our series on advanced PostgreSQL features. In the previous posts, we explored how to design a database…

Jan 6 14

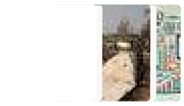
In Stackademic by Mayur Koshti

Native Support for Advanced Data Types in PostgreSQL

Practical Examples of Using PostgreSQL's Advanced Data Types

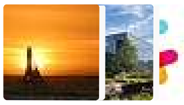
4d ago 73 1

Lists



Staff picks

800 stories · 1569 saves



Stories to Help You Level-Up at Work

19 stories · 920 saves

Self-Improvement 101

20 stories · 3223 saves

Productivity 101

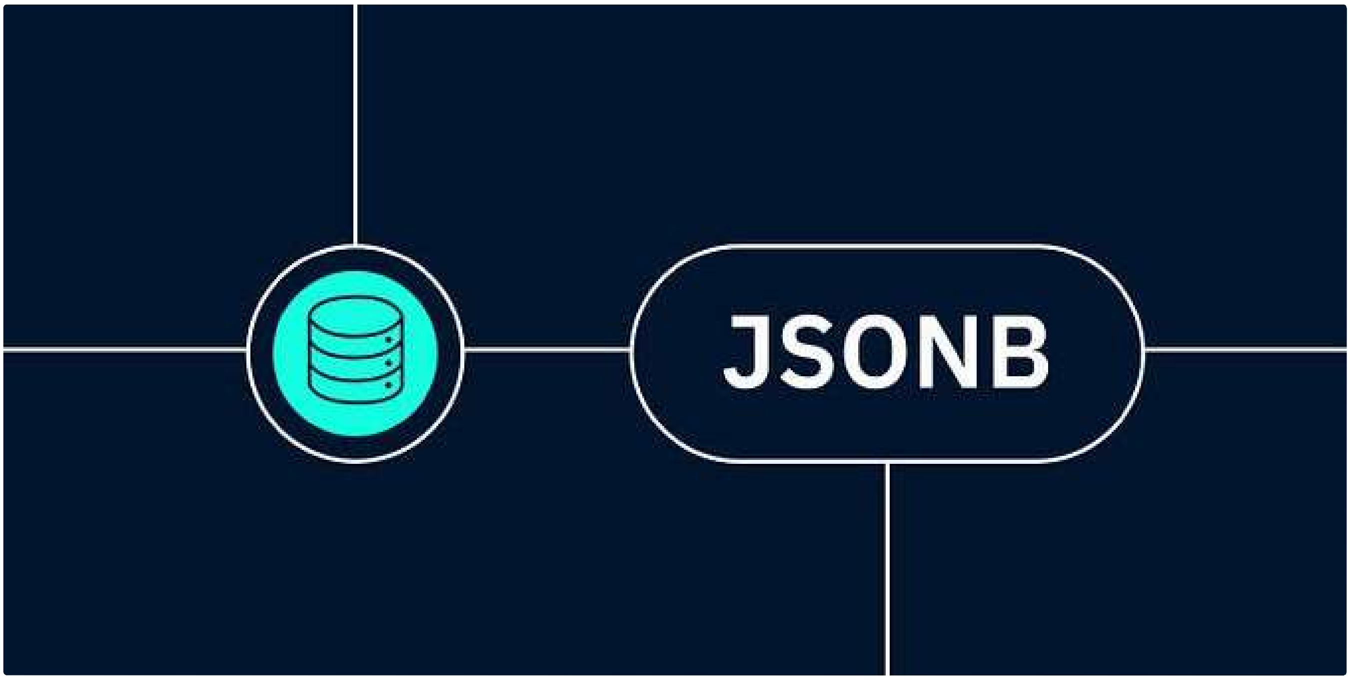
20 stories · 2723 saves

In Towards Data Science by Jiayan Yin

Advanced SQL Techniques for Unstructured Data Handling

Everything you need to know to get started with text mining

Jan 8 126 2



In Mirakl Tech Blog by Mirakl Labs

How JSONB can improve your PostgreSQL queries

By Thomas Fouan Software Engineer at Mirakl

Dec 3, 2024 30

Kemal Öz

Postgres Security 101: User Access and Authorization (4/8)

Managing user access and authorization is one of the foundational elements of database security in PostgreSQL. A well-defined user...

Oct 4, 2024 93

Skilled Coder

Unique and Lesser Known SQL Tricks

Uncommon but extremely useful tricks

Oct 19, 2024 156 5

See more recommendations