# PostgreSQL JSONB Operator Classes of GIN Indexes and Their Usage

Josef Machytka
3 min read · 1 day ago



Throughout 2024, I worked on an internal project exploring the use of JSONB data in PostgreSQL and its various indexing options. My goal was to gather enough insights to provide practical recommendations to clients, as most online articles about JSONB data and GIN indexes are introductory, often repeating documentation with rather trivial examples.

The findings from this project were compelling enough to present at two conferences in June — P2D2 2024 and Swiss PG Day 2024 — as well as at the Berliner PostgreSQL Meetup in October 2024. I focused especially on GIN indexes, as they optimize some important use cases effectively. Therefore I want to give you series of articles which will summarize my findings.

## Understanding GIN Indexes

Before I dive into details, it is important to note that GIN indexes are designed for equality searches. They function like the index at the end of a book, where you look for an exact word and find the pages where it appears. Even a trigram index for `LIKE` searches uses equality, just equality of trigrams.

PostgreSQL provides two operator classes for JSONB:

1. Default `jsonb_ops`
2. Additional `jsonb_path_ops`

Each operator class supports specific use cases, so understanding their behavior is key. I will compare them in this article side by side. When creating a GIN index, we must specify the operator class as follows:

```
CREATE INDEX indexname ON tablename USING GIN (jsonb_data jsonb_ops);

CREATE INDEX indexname ON tablename USING GIN (jsonb_data jsonb_path_ops);
```

If no operator class is specified, PostgreSQL defaults to `jsonb_ops`. While versatile, `jsonb_ops` always leads to larger indexes.

## Search for Values Using `@?` and `@@` Operators

Both `@?` and `@@` operators perform similar functions but differ in syntax. For simpler cases, `@@` may be more straightforward. Here's an example. Please note that SQL/JSON conditions in PostgreSQL queries use double quotes for marking strings.

```
WHERE jsonb_data @? '$.description ? (@ == "Senator")'

WHERE jsonb_data @@ '$.description == "Senator"'
```

### `jsonb_ops`: Searching Across Unknown JSON Paths

The `jsonb_ops` operator class excels when searching for values across unknown JSON paths. Using wildcards in `jsonpath`, we can search anywhere in a JSONB document. This is particularly useful when dealing with data from multiple sources or when schemas can change unexpectedly.

Example query:

```
WHERE jsonb_data @@ '$.** == "doc_type_1"'
```

This query finds all records with the value `doc_type_1` anywhere in the JSONB document. However, this flexibility comes at a cost: GIN indexes with `jsonb_ops` can reach 60–80% of the table size.

### `jsonb_path_ops` : Searching Known JSON Paths

The `jsonb_path_ops` operator class is limited to fully known `jsonpath` queries. Wildcard can be use only for search inside arrays. This operator class is ideal for stable schemas with well-defined JSON paths. It creates smaller indexes (20–30% of the table size) and offers better performance for its specific use case.

Example query:

```
WHERE jsonb_data @@ '$.metadata.header.topics[*] == "python3"'
```

### Checking Containment with `@>` Operator

The `@>` operator checks whether one JSONB object contains another. Both operator classes support this operator, but the `jsonpath` must be fully known. If containment checks are our only use case, `jsonb_path_ops` is a better choice due to its smaller index size and faster performance.

Example query:

```
WHERE jsonb_data @> '{"payload":{"commits":[{"author":{"name": "Jane Joy"}}]}}'
```

The structure of keys and values must match exactly to find a match.

### Checking for Key Existence on the Top Level

The `jsonb_ops` operator class provides three additional operators for checking top-level key or array element existence: `?`, `?|`, and `?&`. However, PostgreSQL is able to gather some basic statistics for top level elements in JSONB object. Therefore query planner may ignore indexes if the schema is stable and top-level keys have very high frequencies.

For volatile schemas or data from many diverse sources with different schemas, these operators and corresponding GIN index can be very useful. Otherwise, such searches will result in sequential scans simply because of too high occurrences of keys in data.

## Summary

When using GIN indexes with JSONB data, understanding your use cases is crucial. While it might be tempting to create a huge “catch-all” GIN index with `jsonb_ops` operator class, such an approach can backfire due to index size and mismatched query patterns. A thoughtful analysis of data and query requirements is essential to optimize performance. In upcoming articles, I will delve deeper into these topics and provide additional insights.

Image created by the author using DeepDreamGenerator

Postgresql | Jsonb | Gin Index | Query Optimization | Json

Edit profile

## Written by Josef Machytka

53 Followers · 10 Following

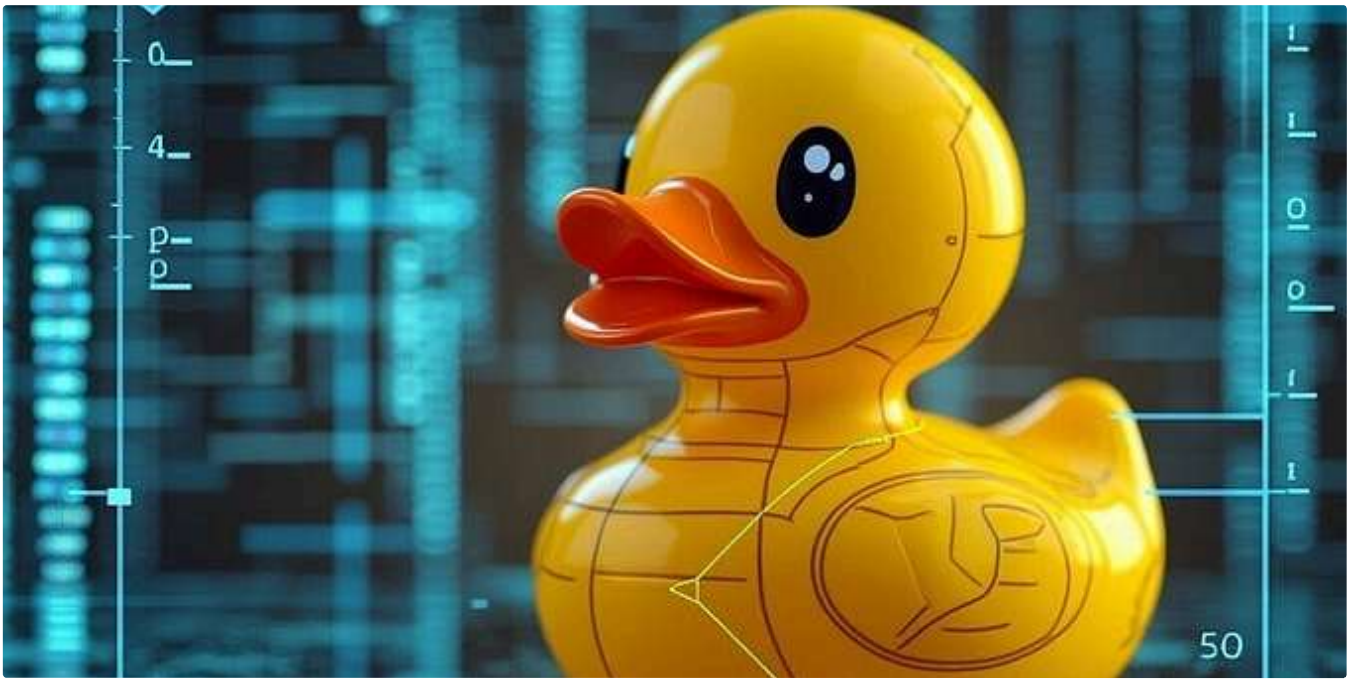
I work as PostgreSQL specialist & database reliability engineer at NetApp Deutschland, Open Source Services division.

## 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

```
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
```

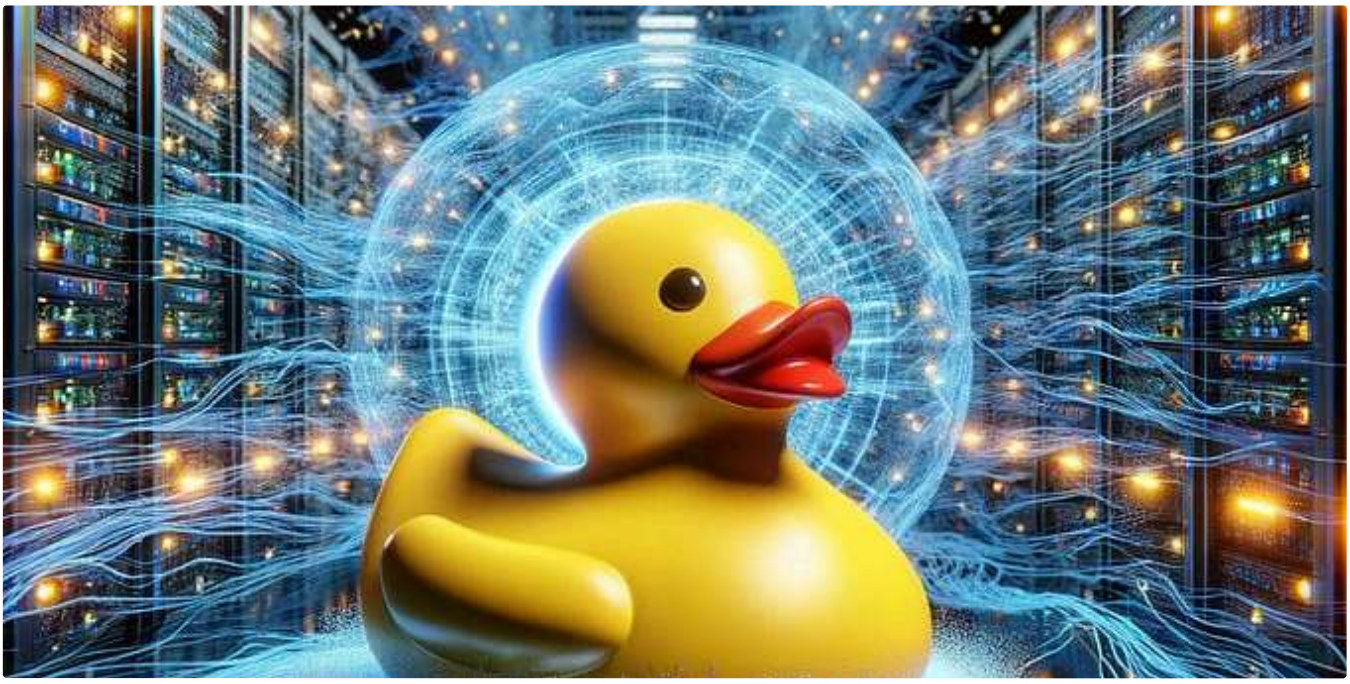
```
ert 10 rows of data
 INTO special_data_types (name, status, permissions, small_number, medium_number, description, data, created_at)
e', 'active', 'read,write', 5, 1000, 'Alice description', 'Alice data', '2023-01-01'),
, 'inactive', 'read', 10, 2000, 'Bob description', 'Bob data', '2023-02-01'),
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

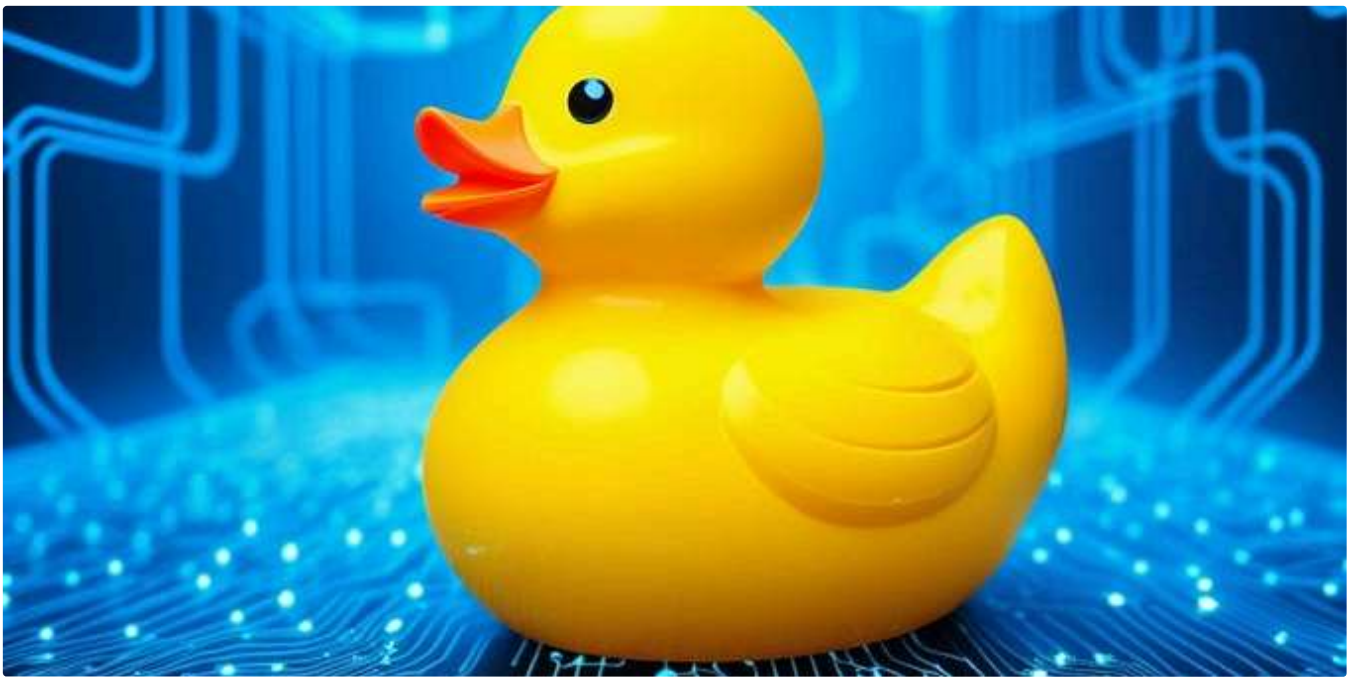


Josef Machytka

## 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...

Dec 5, 2024 11



Josef Machytka

## Extending DuckDB ETL Capabilities with Python

DuckDB has recently become my go-to solution for small ETL tasks. It is an exceptional database created by people who quite obviously...

Nov 24, 2024 26

See all from Josef Machytka

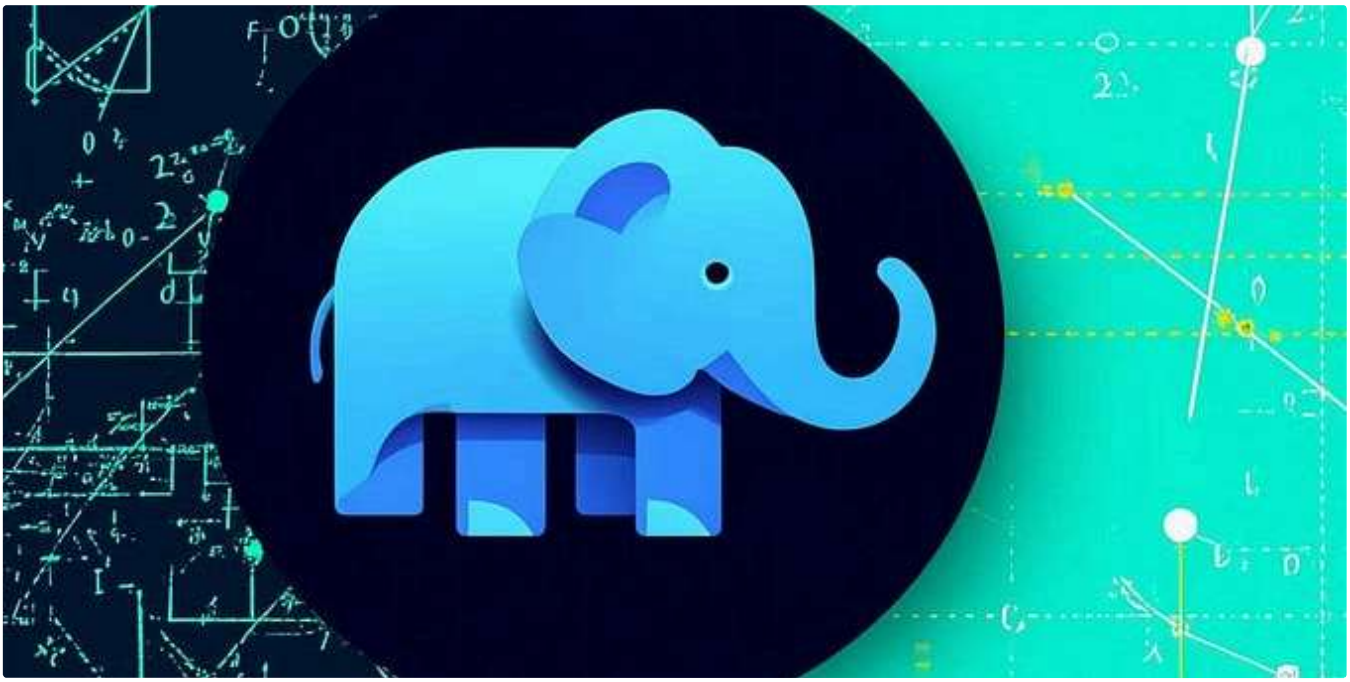
## Recommended from Medium


Miftahul Huda

## Introduction to PostgreSQL Schemas: A Practical Guide

Build scalable, secure, and maintainable databases with practical schema patterns and Go implementation

Dec 29, 2024 7



Phantompete

## Vector Search with pgvector and OCI Database with PostgreSQL—Part 1

In this post we will go through the basics of leveraging pgvector combined with OCI Database with PostgreSQL, vector creation and prepare...

Dec 5, 2024 4

### Lists

**Staff picks**

796 stories · 1558 saves

**Stories to Help You Level-Up at Work**

19 stories · 912 saves

**Self-Improvement 101**

20 stories · 3195 saves

**Productivity 101**

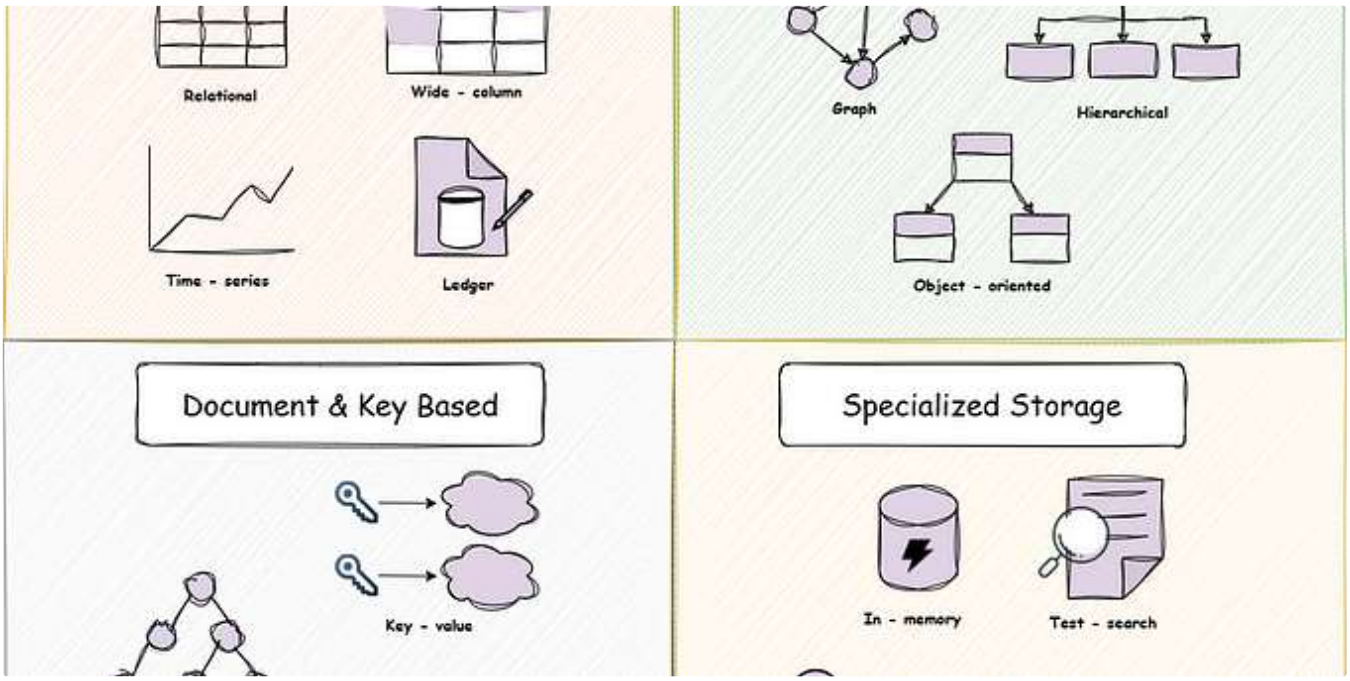
20 stories · 2707 saves

Sruthi Ganesh

## Comparing Query Performance in PostgreSQL: JSONB vs Join Queries

When working with PostgreSQL, it's common to encounter scenarios where you need to decide between different query patterns to achieve...

Jan 2 27




In Level Up Coding by Rishabh Agarwal

## 15 Databases, 15 Use Cases—The Ultimate Guide That No One Asked For (But Everyone Needs)

Stop Using the Wrong Database for the Right Problem!

Dec 29, 2024 645 9

In Dev Genius by Aleksei Aleinikov

## 10 Ways to Work with Large Files in Python: Effortlessly Handle Gigabytes of Data!

Handling large text files in Python can feel overwhelming. When files grow into gigabytes, attempting to load them into memory all at once...

Dec 1, 2024 304 3