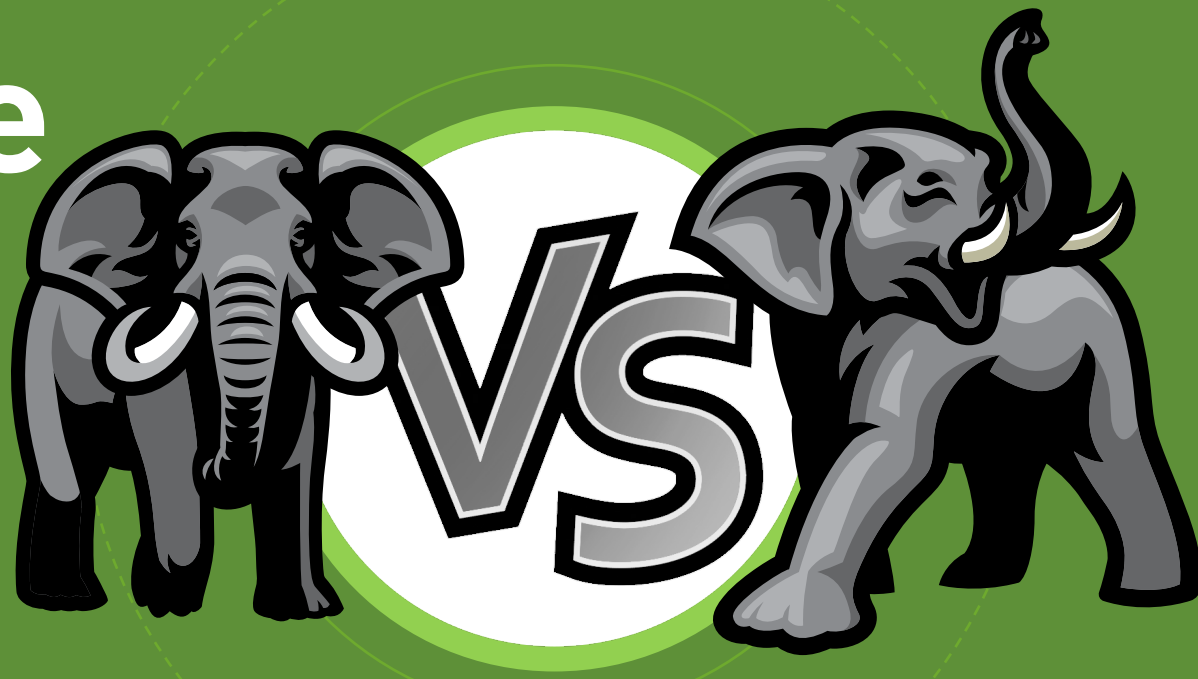
Microsoft

PGConf India

PGConf India 2025

Beginner's Guide to Partitioning vs. Sharding in Postgres



CLAIRE GIORDANO

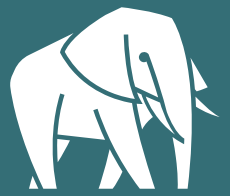
@clairegiordano.bsky.social • @PosetteConf.com

Engineer
Dev Manager
PM
Marketer
Writer
Open-source champion
Community Lead
Podcaster

@clairegiordano
@clairegiordano.bsky.social

TALKING POSTGRES
WITH CLAIRE GIORDANO

Quite a lot of Postgres work @ Microsoft



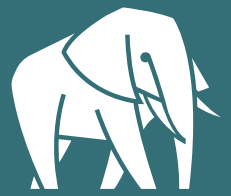
MANAGED DATABASE SERVICE

Azure Database for PostgreSQL

["Flexible Server"]

fully-managed database service for Postgres

NEW CAPABILITIES in FLEXIBLE SERVER



OPEN SOURCE WORK

PostgreSQL core

Contribute to PG open source (and review patches on many other people's work!)

Citus Open Source

Citus open-source extension to Postgres gives you Postgres at any scale. (Think: distributed Postgres.)

PG Ecosystem

Postgres extension & tooling our PG team at Microsoft maintains or contributed to in last ~8 months

PG Community

Contribute to growth & knowledge of the PostgreSQL open-source developer & user communities.

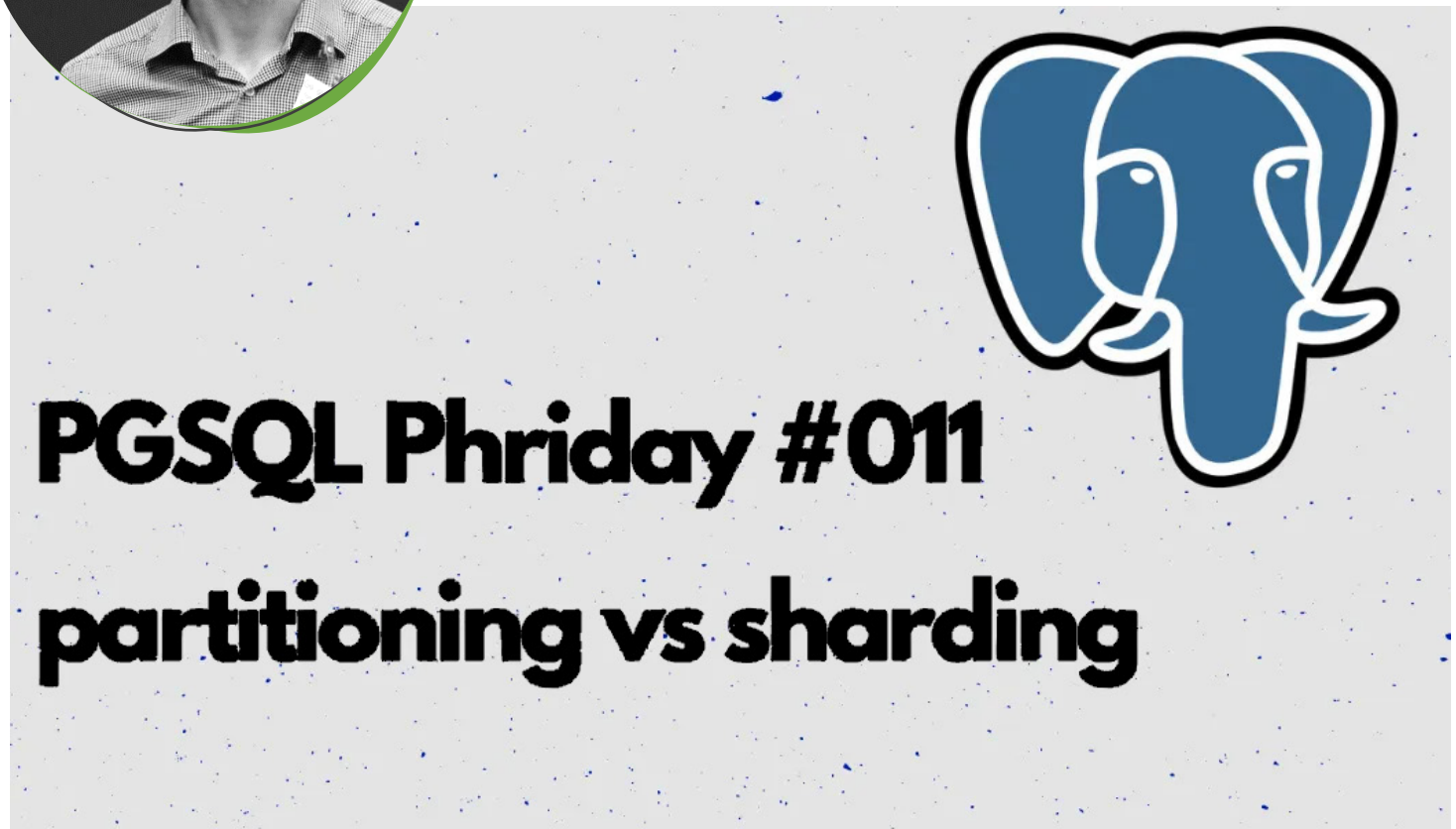
→ aka.ms/blog-pg-at-microsoft

@clairegiordano

PGSQL Phriday #011 — partitioning vs sharding in PostgreSQL

Tomasz Gintowt · Follow

2 min read · Aug 1



Inspiration for this talk

came from a #PGSQLPhriday community blog-fest in August 2023

Hosted by Tomasz Gintowt

So I wrote a thing for PGSQL Phriday

"Understanding Partitioning and Sharding in Postgres and Citus"

aka.ms/blog-parti

Microsoft

Understanding partitioning and sharding in Postgres and Citus

By Claire Giordano

Published Aug 04 2023 04:09 PM 25K Views

The topic of this month's PGSQL Phriday #011 community blogging event is partitioning vs. sharding in PostgreSQL. It seemed right to share a perspective on the question of "partitioning vs. sharding" from someone in the Citus open source team, since we eat, sleep, and breathe sharding for Postgres.

Postgres built-in "native" partitioning—and sharding via PG extensions like Citus—are both tools to grow your Postgres database, scale your application, and improve your application's performance.

What is partitioning and what is sharding? In Postgres, database partitioning and sharding are both techniques for splitting collections of data into smaller sets,

This Beginner Guide has _8_ chapters

What is Postgres partitioning?

What is sharding?

Partitioning + sharding together

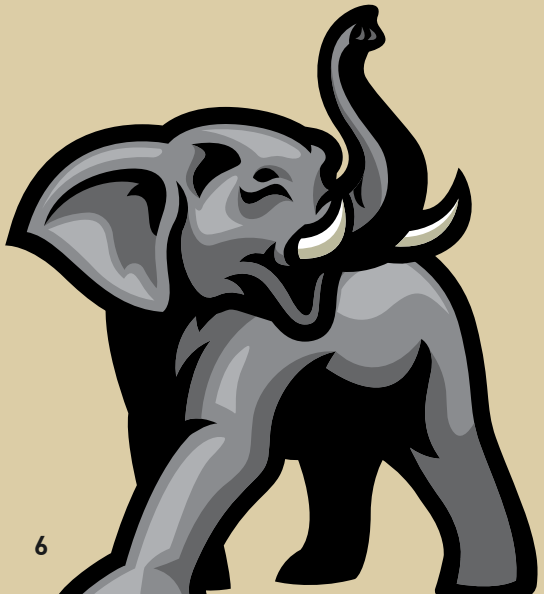
How partitioning & sharding are different

Why partitioning helps query performance

When partitioning helps query performance

Why sharding helps query performance

When sharding helps query performance



Part 1

What is Postgres partitioning?

What is sharding?

Partitioning + sharding together

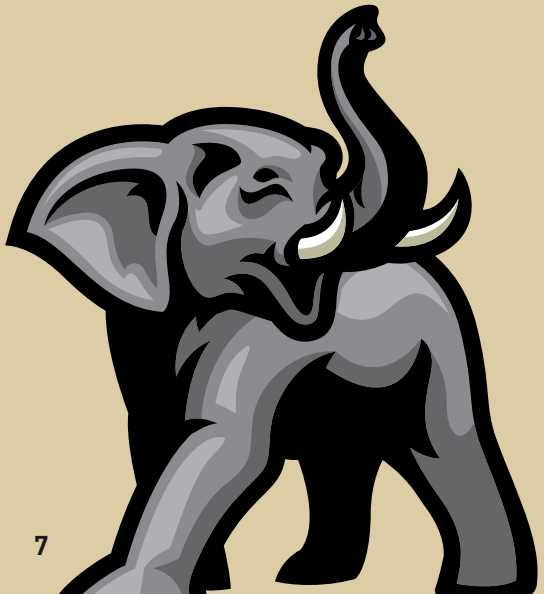
How partitioning & sharding are different

Why partitioning helps query performance

When partitioning helps query performance

Why sharding helps query performance

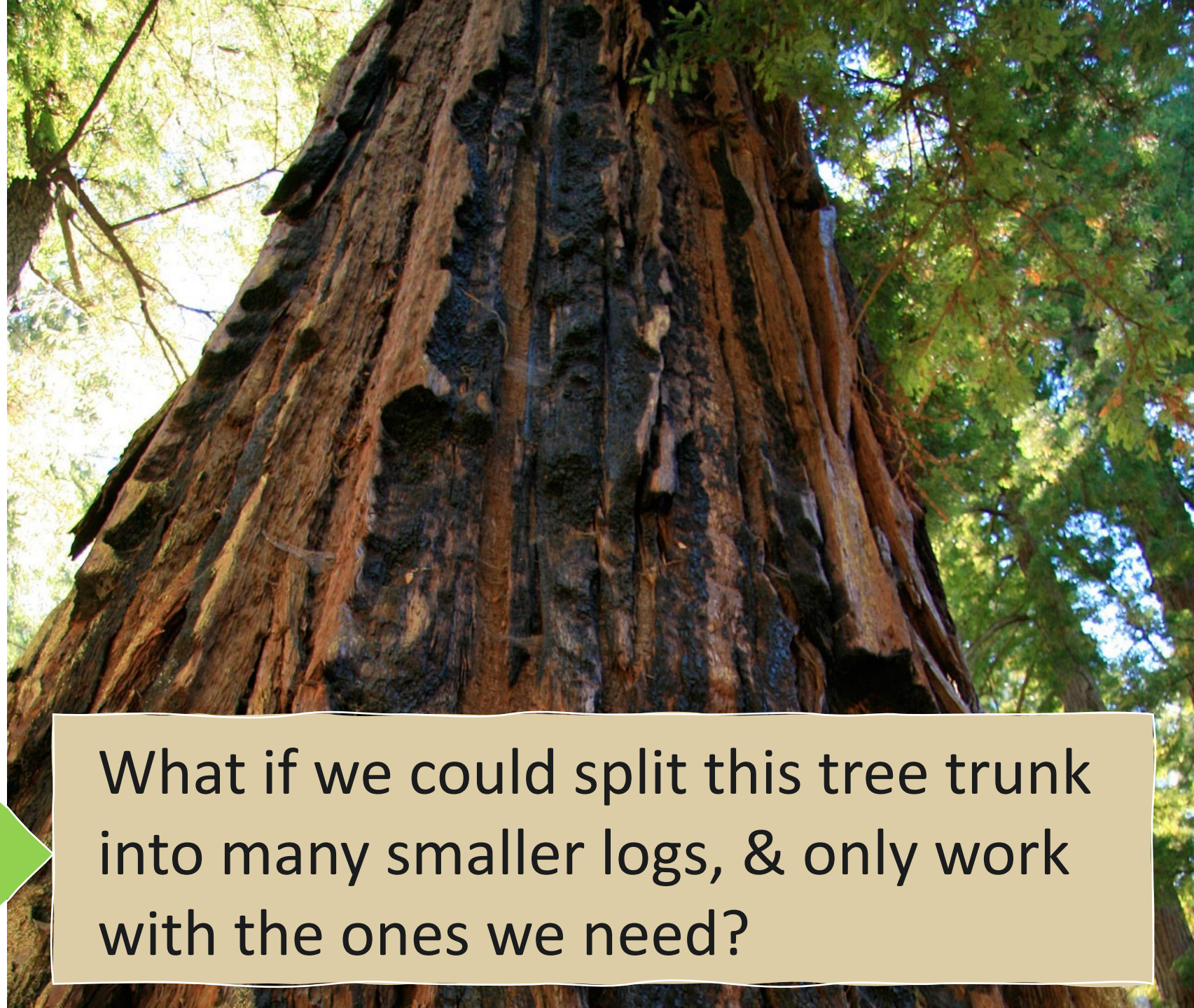
When sharding helps query performance



What is Postgres table partitioning

Let's say this huge Redwood Tree is like a big Postgres table...

What is Postgres table partitioning



What if we could split this tree trunk into many smaller logs, & only work with the ones we need?

- Splits large tables into many smaller tables (partitions)
- Built-in, native feature in Postgres
- Needs to be declared up-front when you first CREATE TABLE

So many improvements to Postgres partitioning in last 5-6 years

Shout-out to @Brandur for this blog post...

aka.ms/partitioning-2022-brandur

Partitioning in Postgres, 2022 edition

Fragment
Partitioning in Postgres, 2022 edition

Published
Oct 5, 2022

*I'm on X/Twitter at **@brandur**.*

But, a lot of work has gone into improving the operator experience since partitioning was introduced. Here's a sprinkling of new features that have come into Postgres over the last five years:

- **Postgres 10:** Brings in the original `CREATE TABLE ... PARTITION BY ...` declarative partitioning commands.
- **Postgres 11:** Support for `PRIMARY KEY`, `FOREIGN KEY`, indexes, and triggers on partitioned tables.
- **Postgres 11:** `INSERT` on the parent partitioned table routes rows to their appropriate partition.
- **Postgres 11:** `UPDATE` statements can move rows between partitions.
- **Postgres 12:** Foreign keys can reference partitioned tables
- **Postgres 12:** Improved `INSERT` performance, `ALTER TABLE ATTACH PARTITION` no longer blocks queries.
- **Postgres 13:** Support for row-level `BEFORE` triggers on partitioned tables.
- **Postgres 13:** Logical replication on partitioned tables (previously, partitions would have to be replicated individually).
- **Postgres 14:** Partitions can be detached in a non-blocking way with `ALTER TABLE ... DETACH PARTITION ... CONCURRENTLY`.

What are the benefits of Postgres Partitioning?

- Can improve query performance
- Improves performance of autovacuum /in so many cases!!
- Enables you to optimize storage costs (cheap vs. expensive)
- Improves performance of bulk deletes /if drop partitions

From Brandur.org

"In Postgres, trying to remove old rows from a large, hot table is flitting with disaster.

A long running query must iterate through and mark each one as dead, and even then nothing is reclaimed until an equally expensive vacuum runs through and frees space, and only when it's allowed to after rows are no longer visible to any other query in the system, whether they're making use of the large table or not. Each row removal land in the WAL, resulting in significant amplification.

But with partitions, deletion becomes a simple DROP TABLE. It executes instantly, and with negligible costs (partitioning has other benefits too). The trade-off is maintenance."

— Brandur

Title of post: "Partitioning in Postgres, 2022 Edition"

Declarative Postgres partitioning has 3 partitioning methods

`PARTITION BY RANGE`

Time-series & IOT

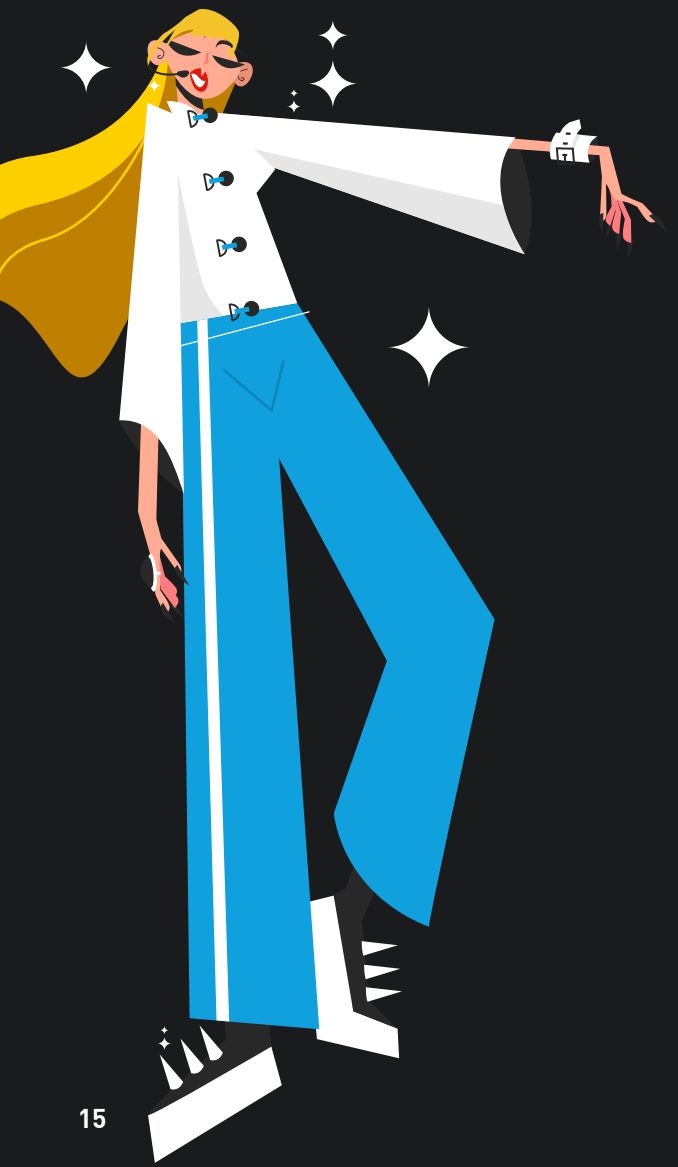
`PARTITION BY LIST`

Countries,
company divisions,...

`PARTITION BY HASH`

When partitioning by range & by list do not work... when NOT an obvious partitioning key

Tracking monthly concert revenue for Taylor Swift



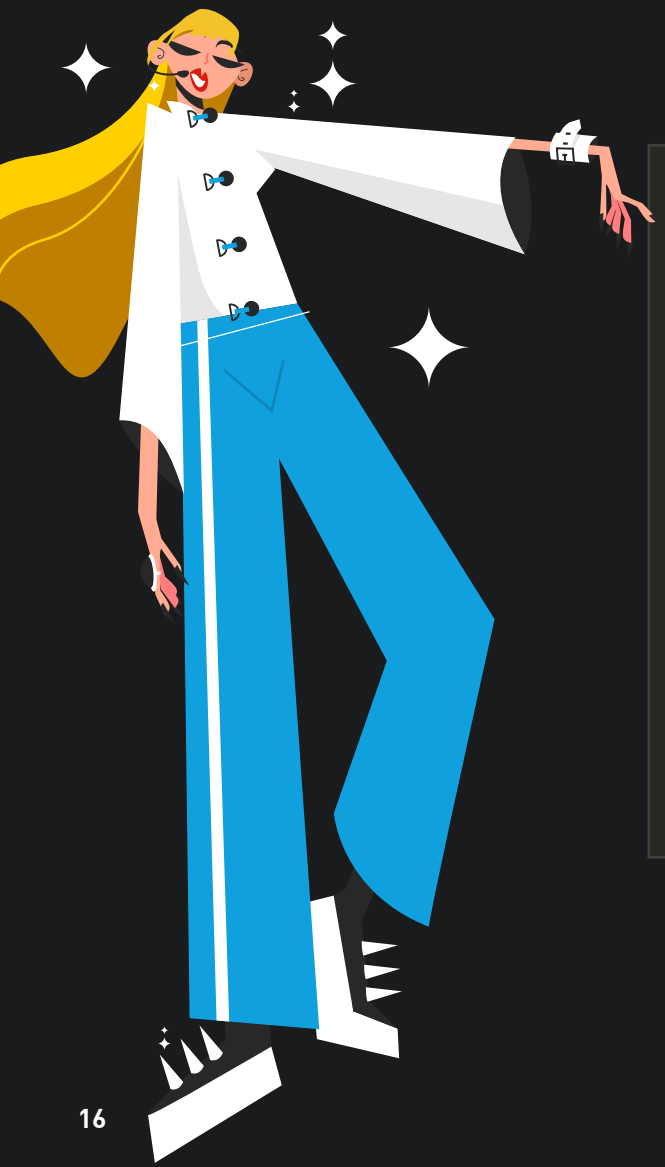
Step 1: Decide on partition method

Step 2: Decide on partition key

Step 3: Create a partitioned table

Step 4: Create partitions

Let's create a partitioned table



```sql
CREATE TABLE concert_revenue (
    city_id        int not null,
    sale_date      date not null,
    merch_sales    int,
    ticket_sales   int
) PARTITION BY RANGE (sale_date);
```

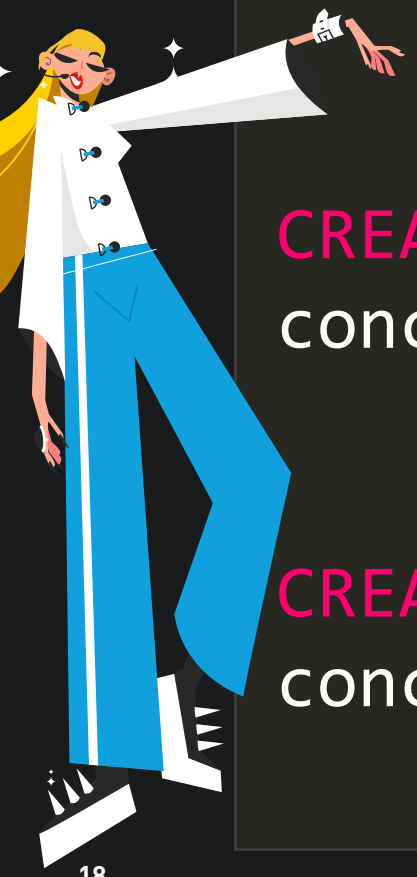
Let's create our first partition for June 2024

```
CREATE TABLE concert_revenue_cy2024m06 PARTITION OF
concert_revenue
    FOR VALUES FROM ('2024-06-01') TO ('2024-07-01');
```

TIP:
lower bound is INCLUSIVE

TIP:
upper bound (2024-07-01) is NOT inclusive

Let's create 3 partitions for Jun / Jul / Aug 2024

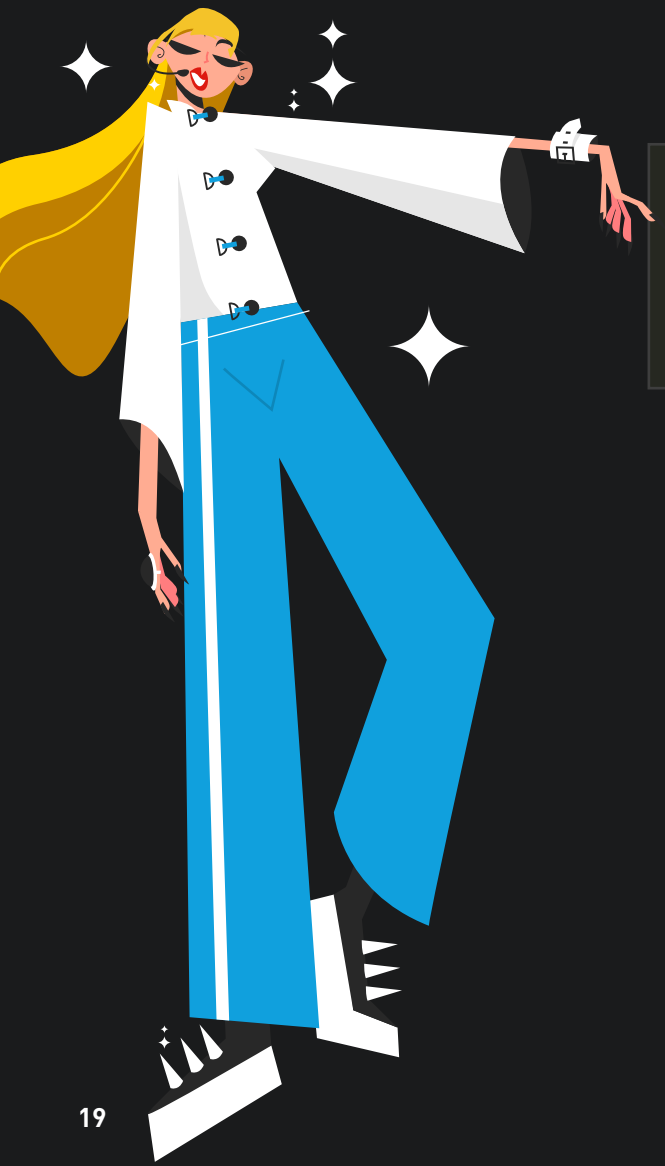


```
CREATE TABLE concert_revenue_cy2024m06 PARTITION OF
concert_revenue
    FOR VALUES FROM ('2024-06-01') TO ('2024-07-01');

CREATE TABLE concert_revenue_cy2024m07 PARTITION OF
concert_revenue
    FOR VALUES FROM ('2024-07-01') TO ('2024-08-01');

CREATE TABLE concert_revenue_cy2024m08 PARTITION OF
concert_revenue
    FOR VALUES FROM ('2024-08-01') TO ('2024-09-01');
```

Later, you can delete older data—e.g. June 2024



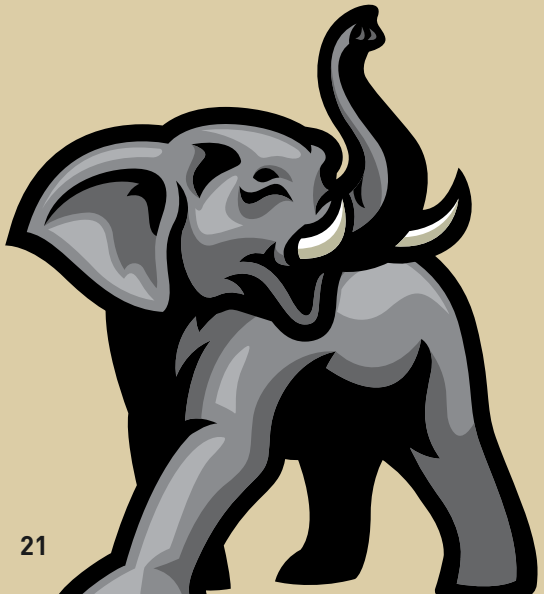
```
DROP TABLE concert_revenue_cy2024m06;
```

Partitioning is not magic! ...You need to maintain these partitions

pg_partman with pg_cron

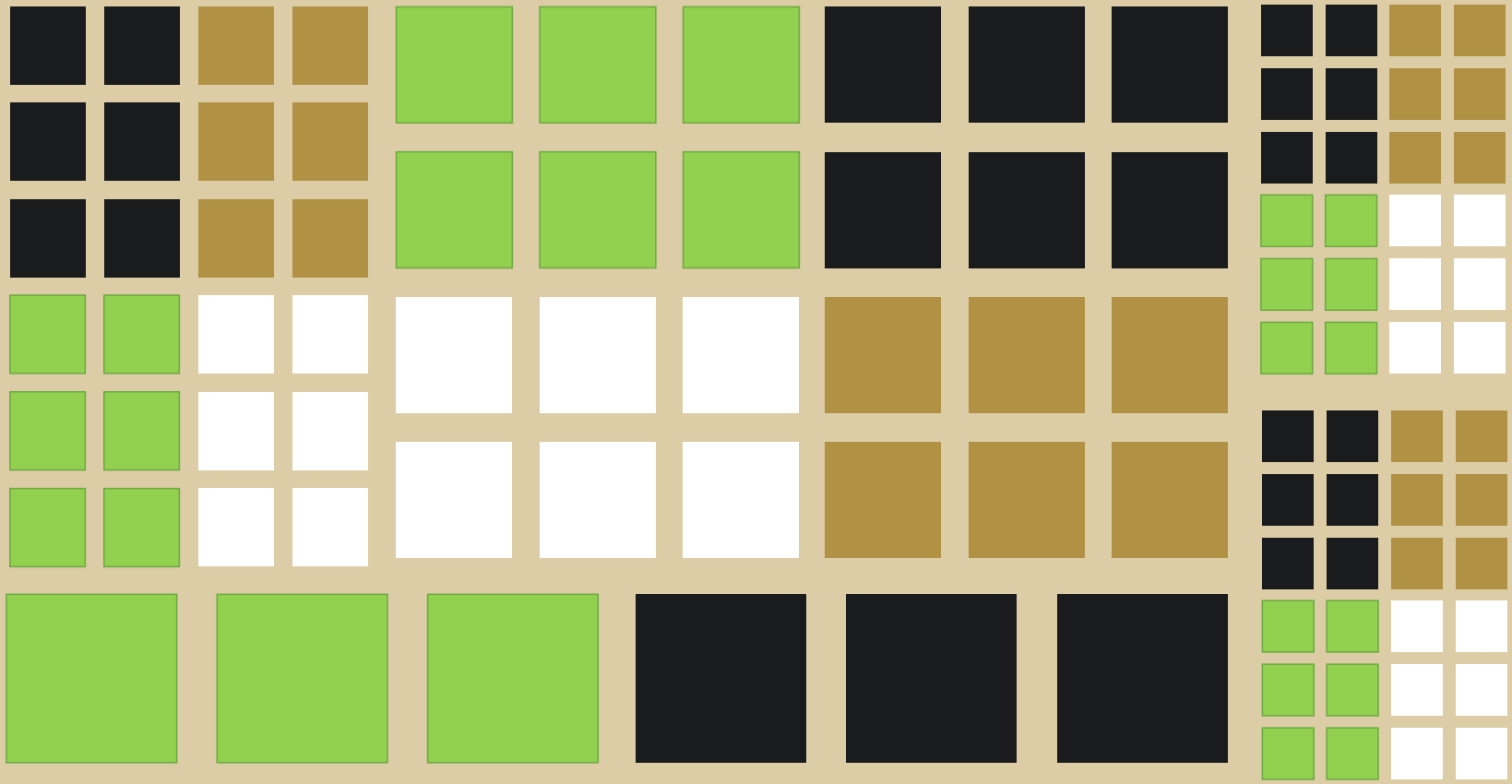
Part 2

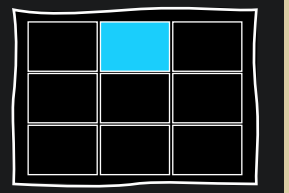
What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	



What is sharding?

→ Splitting big Postgres tables into smaller tables ("shards")





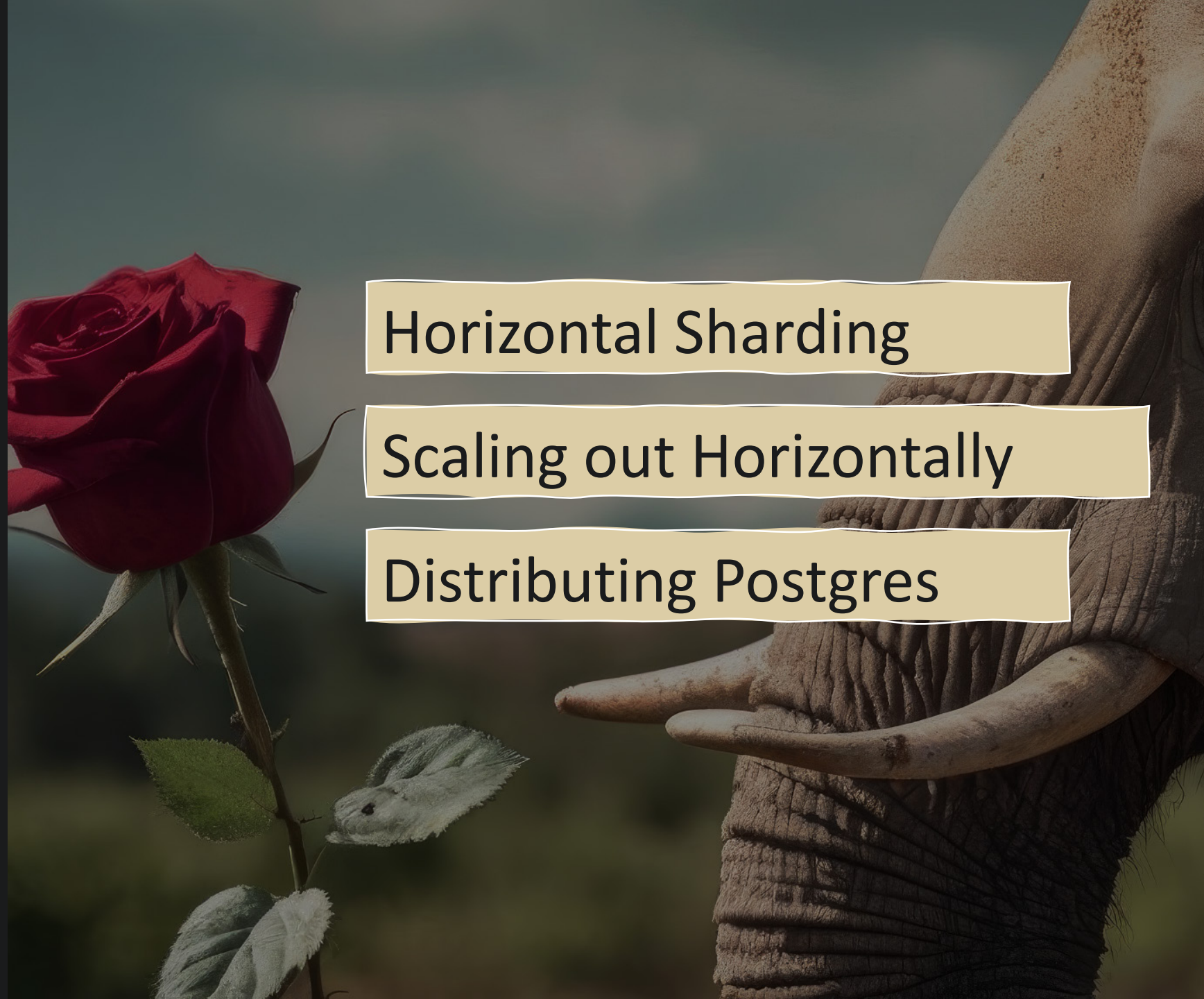
What is sharding?

→ Splitting big Postgres tables into smaller tables ("shards")

AND

→ Distributing shards across multiple nodes

Alternative / Equivalent terms for “Sharding” in Postgres



Horizontal Sharding

Scaling out Horizontally

Distributing Postgres

Back to previous metaphor

If you have one big ginormous tree trunk

& you want to split it into smaller pieces

& distribute it across multiple parts of forest

3 common ways to shard Postgres

1

Manual sharding

sometimes called “sharding at application layer”

2

Partitions + postgres_fdw

Create hash-partitioned Postgres table in which every partition is a foreign table using `postgres_fdw`

3

Citus

Extension to Postgres

Open source, & also a managed service

Bulk of sharding work done by...

1

Manual sharding

App Dev

2

Partitions + postgres_fdw

DBA

3

Citus

Citus Dev

In today's talk, we'll focus on sharding with Citus extension to Postgres

citusdata / citus

Type / to search

Code | Issues 990 | Pull requests 93 | Discussions | Actions | Wiki | Security | Insights

citus Public

Edit Pins | Watch 219 | Fork 695 | Starred 11.1k

main | Go to file t | Code

onurctirtir Avoid artifact name collisi... 30bf960 · 2 days ago 6,924 Commits

.devcontainer	Bump pg versions 14.15, 15.10, 1...	last month
.github	Avoid artifact name collision for ...	2 days ago
ci	Force LC_COLLATE=C for sort i...	last year
config	Renames configure.in to fix war...	3 years ago
images	Addd missing image to Technica...	2 years ago
src	Disable nonmaindb interface (#...	5 days ago

About

Distributed PostgreSQL as an extension

www.citusdata.com

postgres, multi-tenant, sql, database, scale, postgresql, sharding, distributed-database, citus, relational-database, database-cluster, citus-extension

Readme

AGPL-3.0 license

Code of conduct

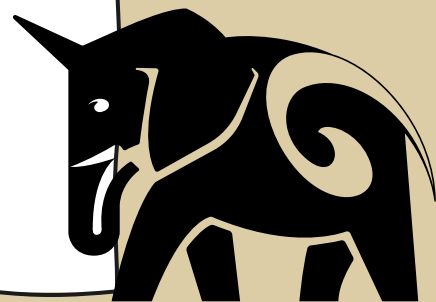
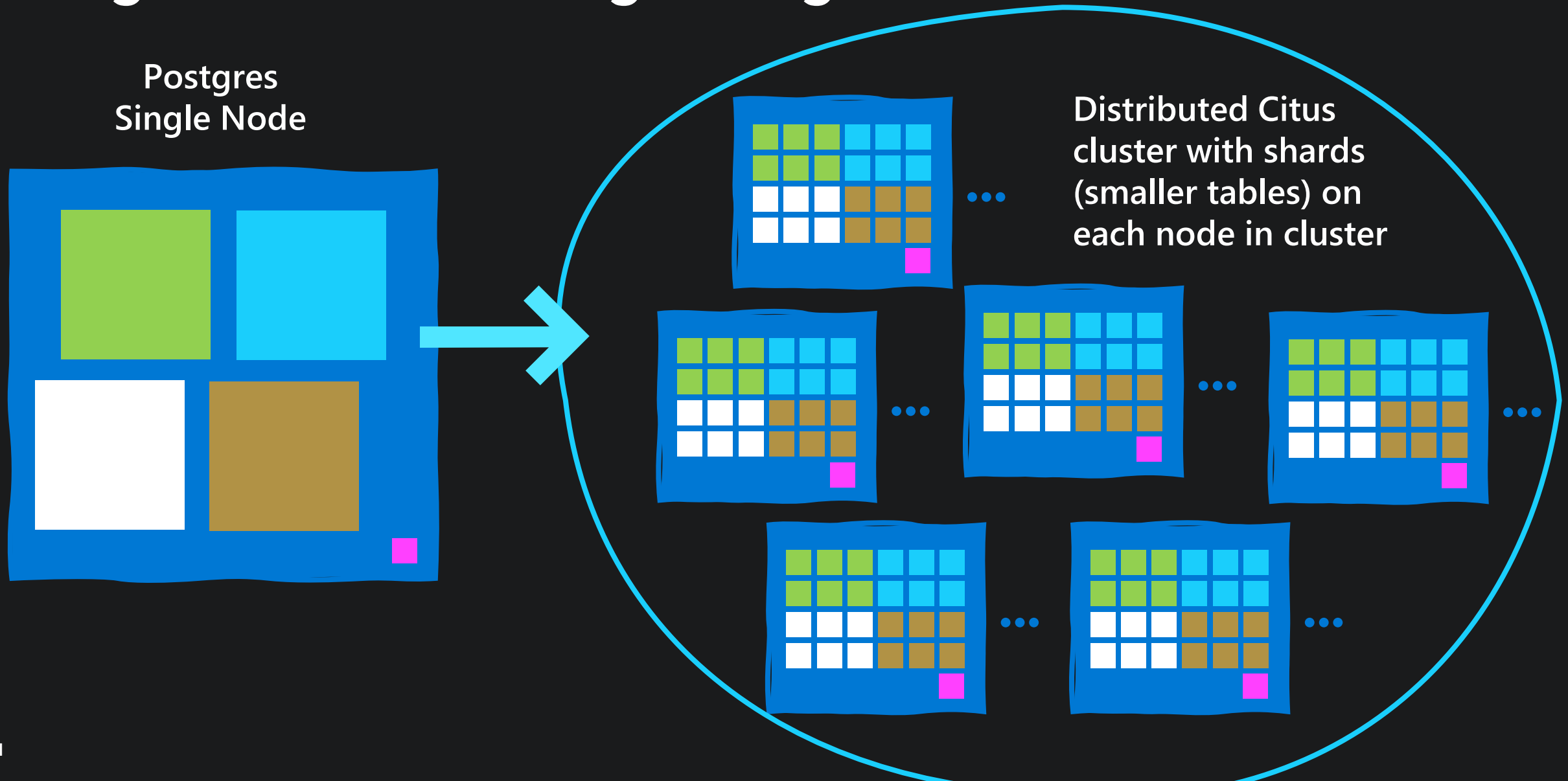


Diagram of sharding Postgres with Citus



Citus extension = LOT more than sharding

Concepts

- Citus table
- Shell table
- Metadata
- Metadata syncing
- Distributed tables
- Reference tables
- Single shard tables
- Local tables
- Shards
- Colocation group
- Shard placement
- Node
- Node group
- Coordinator
- Worker nodes
- Cluster
- Read replica cluster
- Client connections

Design Principles

- Use cases
- Schema management
- Query layer
- Transactional semantics
- Replication model

Use of PostgreSQL Hooks

- UDFs
- Planner & executor hooks
- CustomScan
- Transaction callbacks
- Utility hook
- Background workers

Citus Query Planner

- Distributed Query Planning
- Fast Path Router Planner
- Router Planner
- Query Pushdown Planning
- Recursive Planning
- Logical Planner & Optimizer
- Combine Query Planner

Citus Query Executor

- Custom scan
- Function evaluation
- Prepared statements
- Local plan caching
- Adaptive executor

DDL

Connection management

Transactions

Locking

Rebalancing

Logical Decoding / CDC

Global PID

Function call delegation

Query from any node

aka.ms/blog-citus-technical-readme

Benefit of sharding Postgres with Citus:

›››› Get more Performance & Scale than you can eek out on a single node

Can improve query performance at scale //due to additional memory, CPU, & disk—and also due to parallelism across nodes in cluster

Enables application to grow & scale // not just now, but in future

Improves performance of autovacuum //since autovacuum runs in parallel across all nodes in cluster

How to distribute tables with Citus
for row-based sharding

```
SELECT
create_distributed_table(
    'table_name',
    'sharding_key');
```

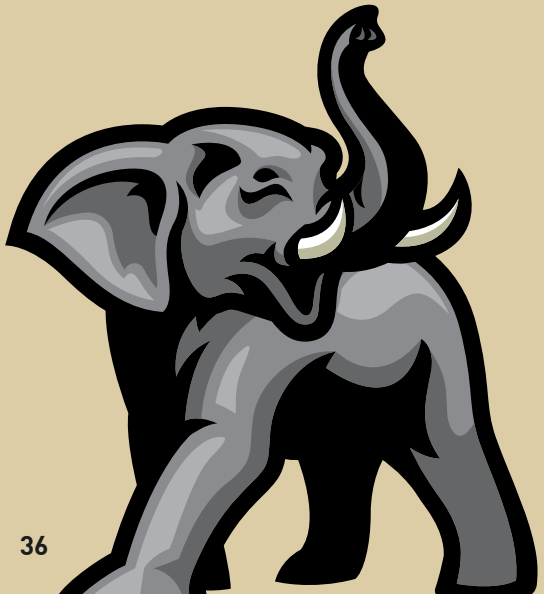
N.B. “sharding key” is often called a “distribution column”

Added in Citus 12.0, a 2nd way to shard: schema-based sharding

```
SELECT citus_schema_distribute(
    'name');
```

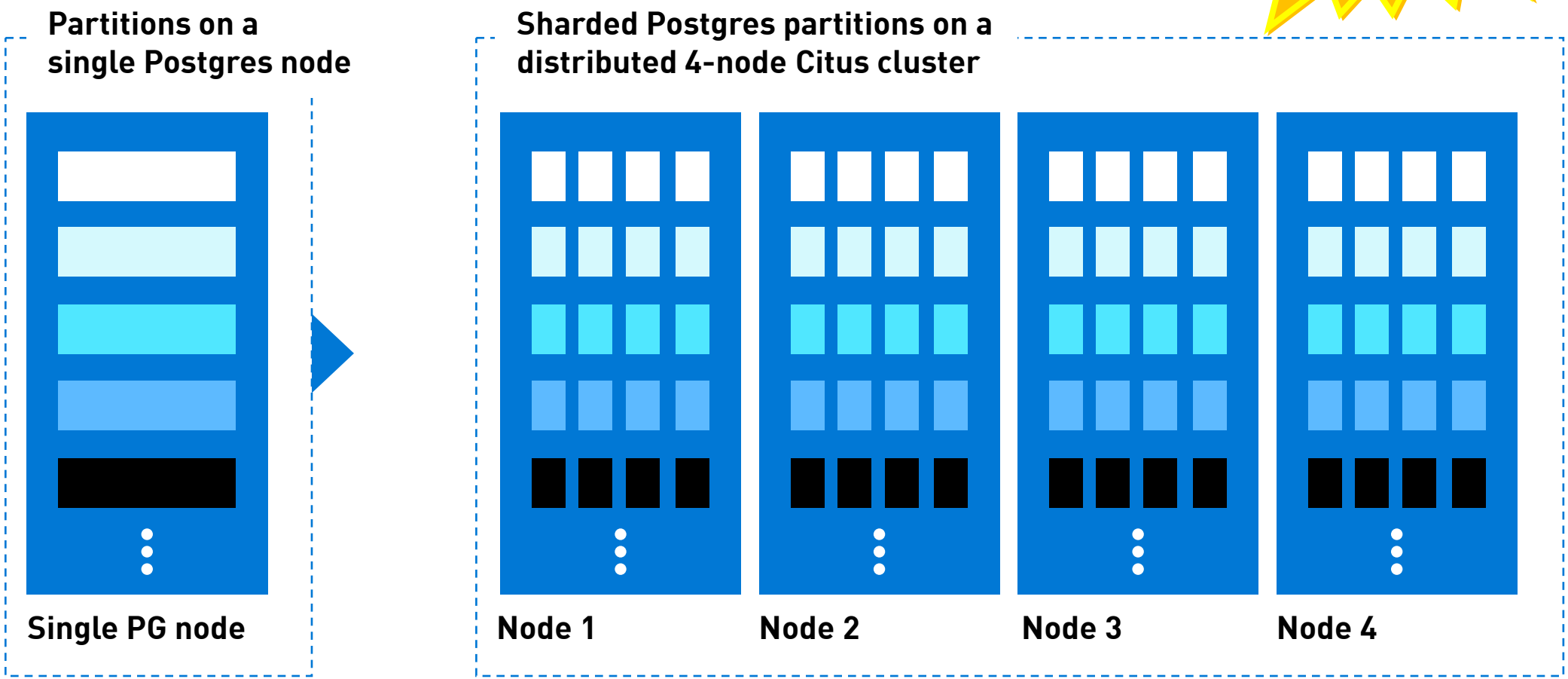
Part 3

What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	

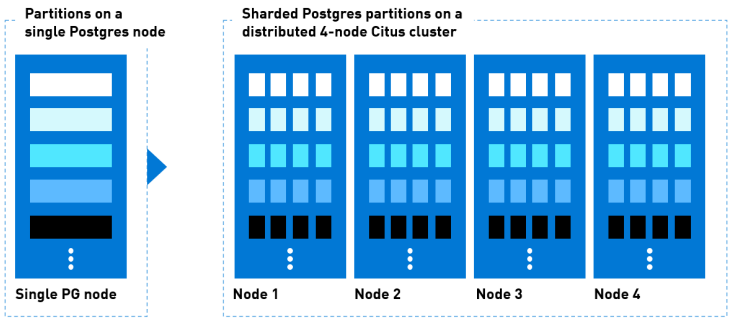


Can you Partition & Shard together?

Can you Partition & Shard together?



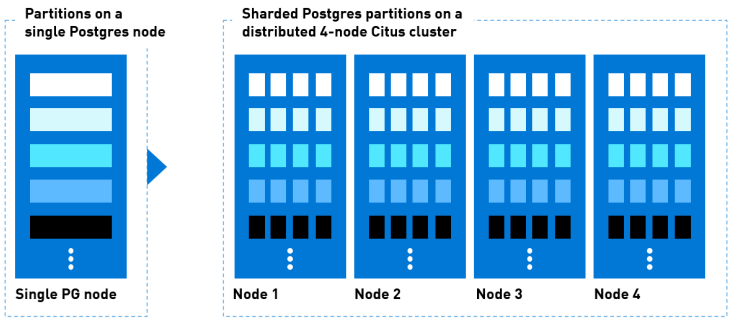
These two Citus UDFs simplify partition management



#1—Create as many partitions as necessary for given time range

```
create_time_partitions(table_name regclass,
partition_interval interval, end_at timestamp with
time zone, start_from timestamp with time zone
DEFAULT now())
```

These 2 Citus UDFs simplify partition management

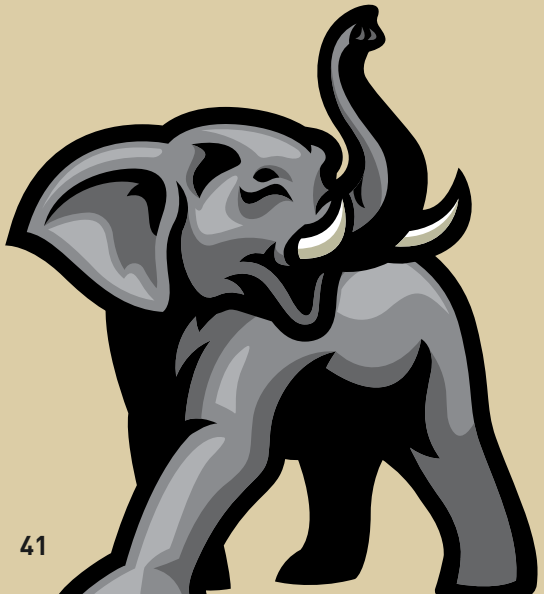


#2—Drop all partitions older than given timestamp

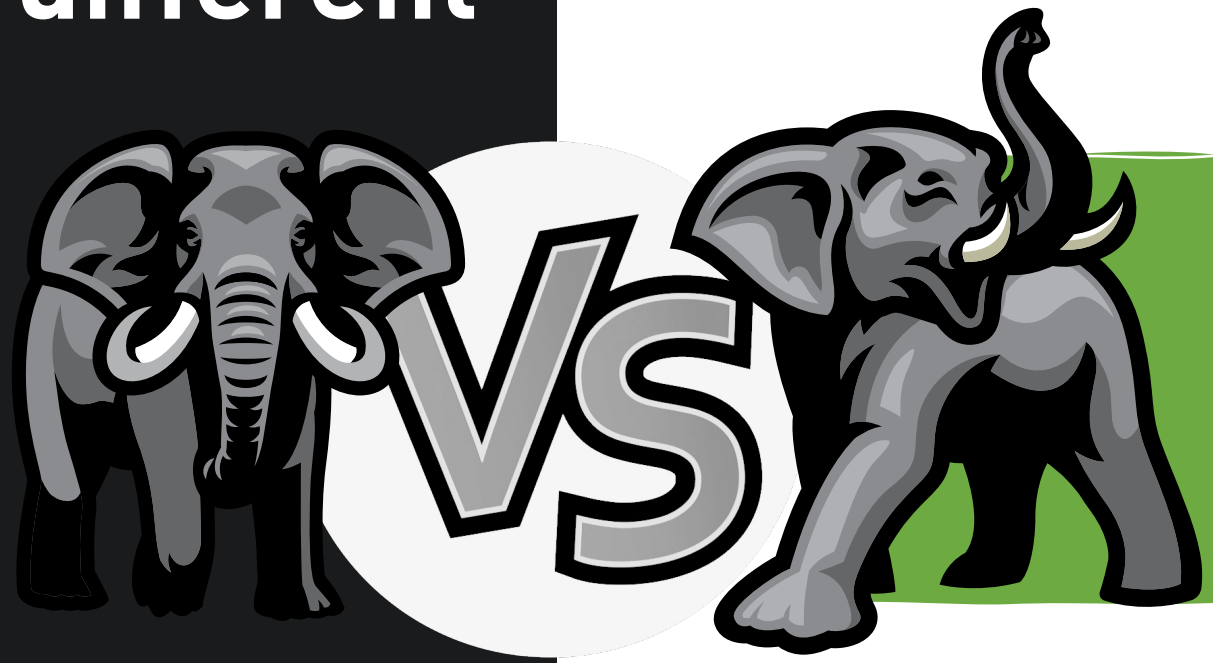
```
drop_old_time_partitions(table_name
regclass, older_than timestamp with time zone)
```

Part 4

What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	



How partitioning & sharding are different



Let’s look at a comparison table!

0

Aspects or attributes of these technologies

1

Postgres native Partitioning

2

Sharding with Citus extension

3

Partitioning + Sharding Combination

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	○		◑

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◑
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●
Better index cache hit ratios	●	●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◑
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●
Better index cache hit ratios	●	●	●
Automatic maintenance	●	●	◐

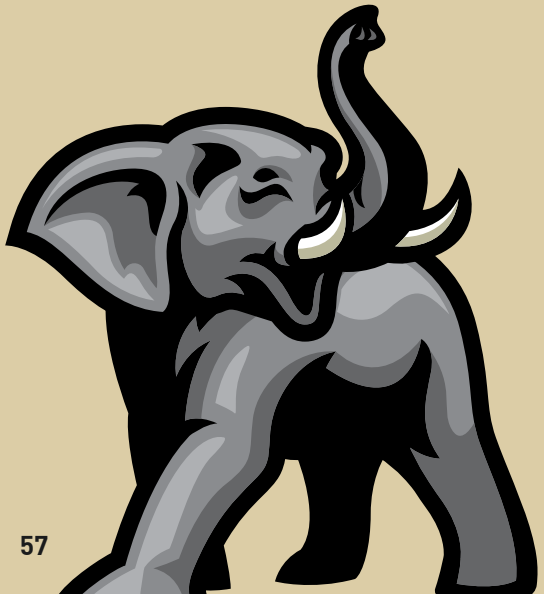
Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◑
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●
Better index cache hit ratios	●	●	●
Automatic maintenance	●	●	◐
Time series apps (e.g. IOT)	●		●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◐
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●
Better index cache hit ratios	●	●	●
Automatic maintenance	◐	●	◐
Time series apps (e.g. IOT)	●		●
Multi-tenant SaaS apps		●	●

Attribute	Partitioning	Citus Sharding	Partition + Shard Combo
Built into Postgres	●		◐
Extension to Postgres		●	◑
Single node	●		
Multi-node		●	●
Drop old data quickly	●		●
Parallel, distributed SQL/DDL/DML		●	●
Partition/Shard Pruning	●	●	●
Parallel autovacuum	●	●	●
Better index cache hit ratios	●	●	●
Automatic maintenance	◐	●	◐
Time series apps (e.g. IOT)	●		●
Multi-tenant SaaS apps		●	●

Part 5

What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	



Why partitioning can help improve query performance

Partition Pruning

Why partitioning can help improve query performance

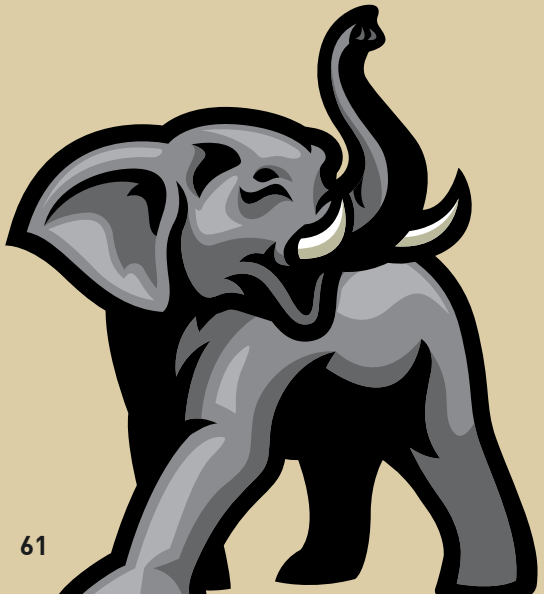
Data Locality

Why partitioning can help improve query performance

Partition Pruning & Data Locality

Part 6

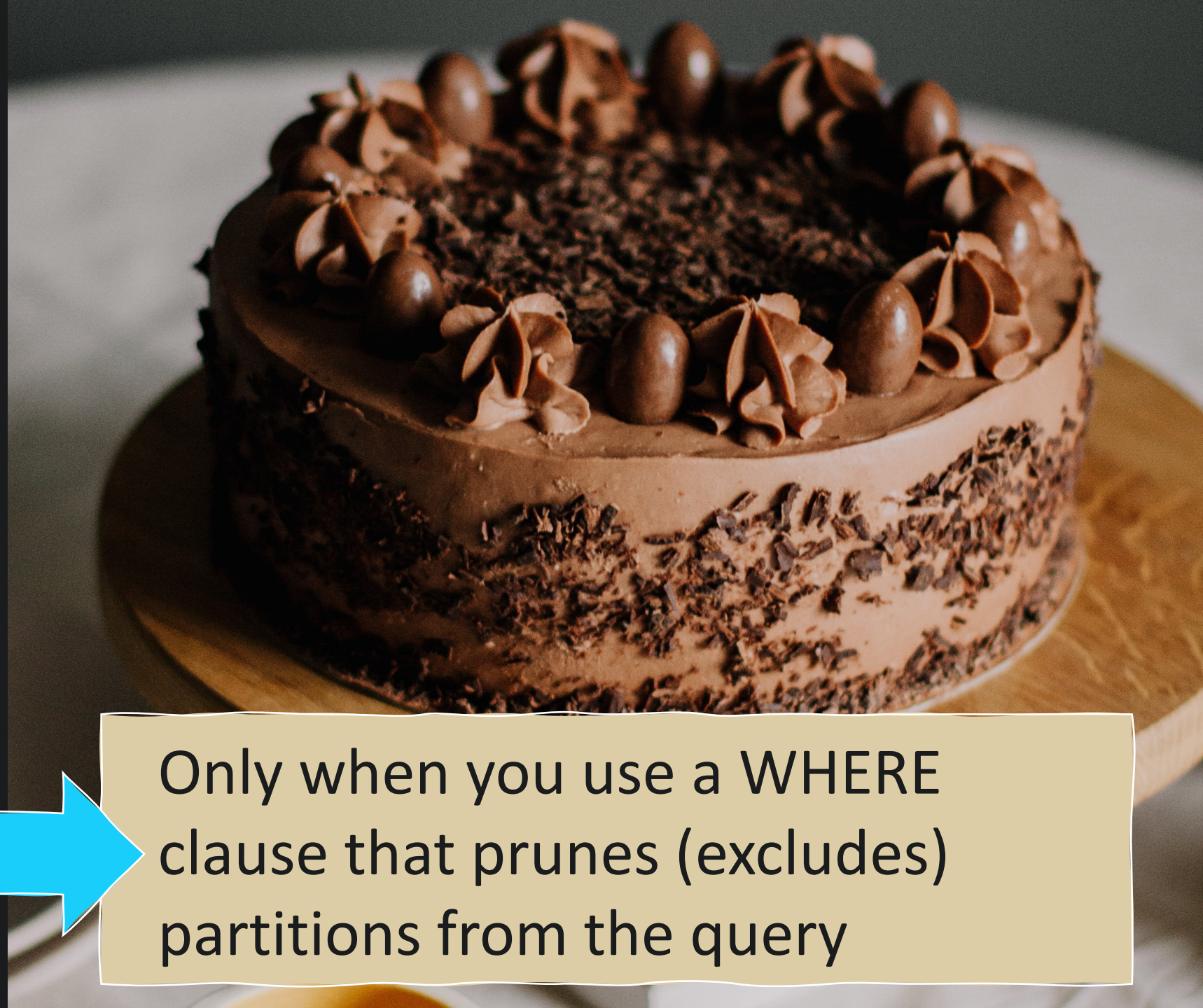
What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	



When partitioning helps improve query performance

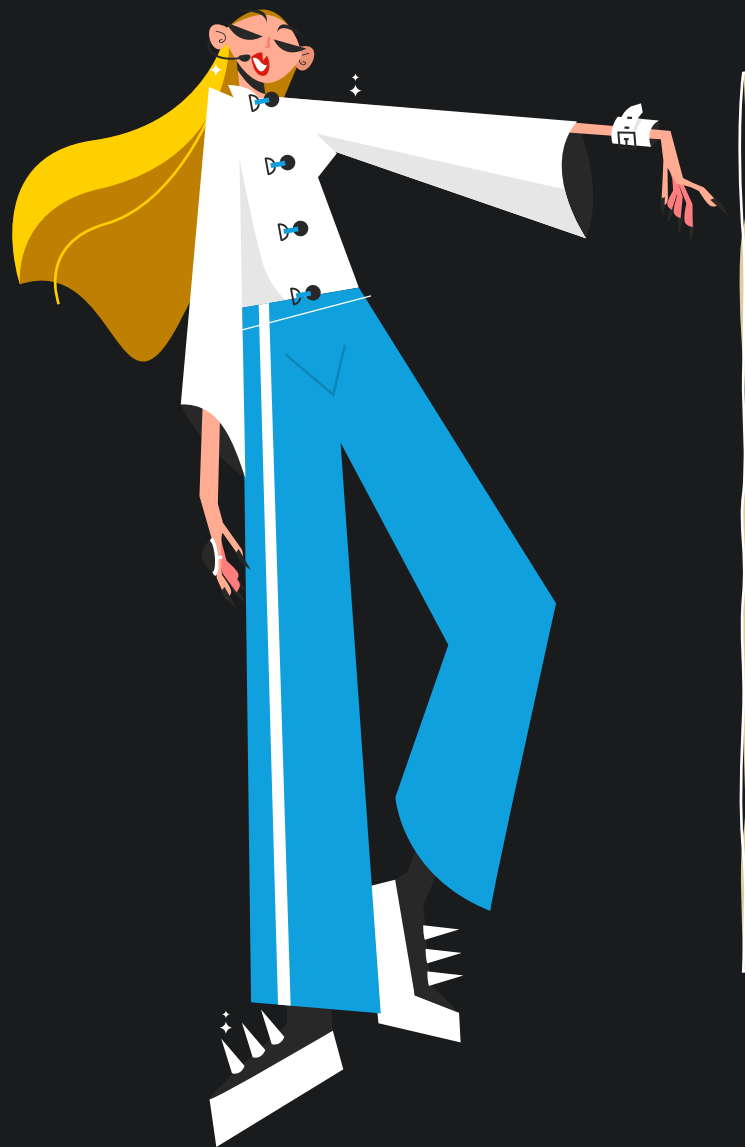
Just like when you bake a cake, need essential ingredients to get the right result

When partitioning helps improve query performance



Only when you use a WHERE clause that prunes (excludes) partitions from the query

Remember that Taylor Swift concert example...



Tracking monthly concert revenues...

What if we wanted to know what the **August revenue** was from ticket sales + merch sales?

This EXPLAIN query is missing WHERE clause

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue;
```

So it has to scan all 3 of the partitions ☹

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue;
```

```
                                QUERY PLAN
-------------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on concert_revenue_cy2024m06 concert_revenue_1
         ->  Seq Scan on concert_revenue_cy2024m07 concert_revenue_2
         ->  Seq Scan on concert_revenue_cy2024m08 concert_revenue_3
(5 rows)
```

And doesn't benefit from partition pruning ☹

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue;

                          QUERY PLAN
-----------------------------------------------------------
 Aggregate
   ->  Append
         -> Seq Scan on concert_revenue_cy2024m06 concert_revenue_1
         -> Seq Scan on concert_revenue_cy2024m07 concert_revenue_2
         -> Seq Scan on concert_revenue_cy2024m08 concert_revenue_3
(5 rows)
```

This EXPLAIN query includes a WHERE clause

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue
WHERE sale_date BETWEEN '2024-08-01' AND '2024-08-31';
```

And so it only needs to scan the Aug partition

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue
WHERE sale_date BETWEEN '2024-08-01' AND '2024-08-31';

                                    QUERY PLAN
-----------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on concert_revenue_cy2024m08 concert_revenue
       Filter: ((sale_date >= '2024-08-01'::date) AND (sale_date <=
'2024-08-31'::date))
(3 rows)
```

And so it only needs to scan the Aug partition

```
EXPLAIN (costs off) SELECT count(*) FROM concert_revenue
WHERE sale_date BETWEEN '2024-08-01' AND '2024-08-31';

                                    QUERY PLAN
-----------------------------------------------------------------------
----------------------
 Aggregate
   -> Seq Scan on concert_revenue_cy2024m08 concert_revenue
        Filter: ((sale_date >= '2024-08-01'::date) AND (sale_date <=
'2024-08-31'::date))
(3 rows)
```

Part 7

What is Postgres partitioning?

What is sharding?

Partitioning + sharding together

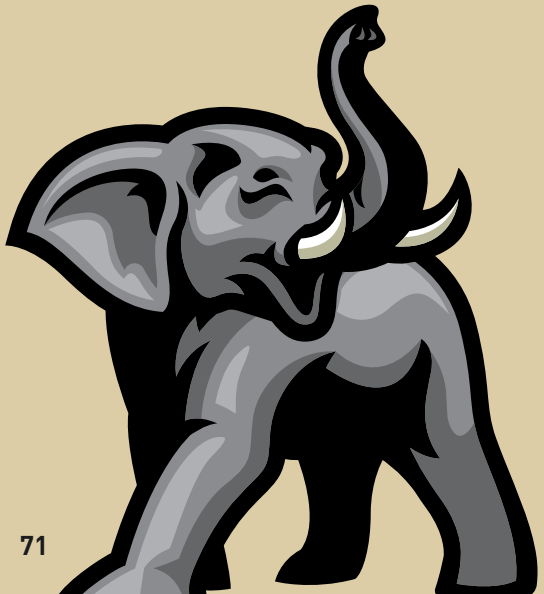
How partitioning & sharding are different

Why partitioning helps query performance

When partitioning helps query performance

Why sharding helps query performance

When sharding helps query performance



Why sharding can help query performance

More CPU, Memory, Disk

Why sharding can help query performance

More CPU, Memory, Disk

Why sharding can help query performance

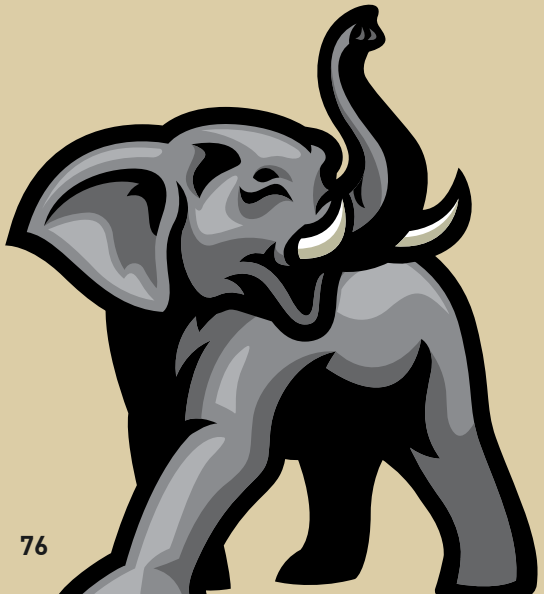
Why sharding can help query performance

More CPU, Memory, Disk

Shard pruning logic

Part 8

What is Postgres partitioning?	What is sharding?	Partitioning + sharding together
How partitioning & sharding are different	Why partitioning helps query performance	When partitioning helps query performance
Why sharding helps query performance	When sharding helps query performance	



When sharding can help improve query performance

When your application needs more cpu, memory, or disk than is possible on a single Postgres node

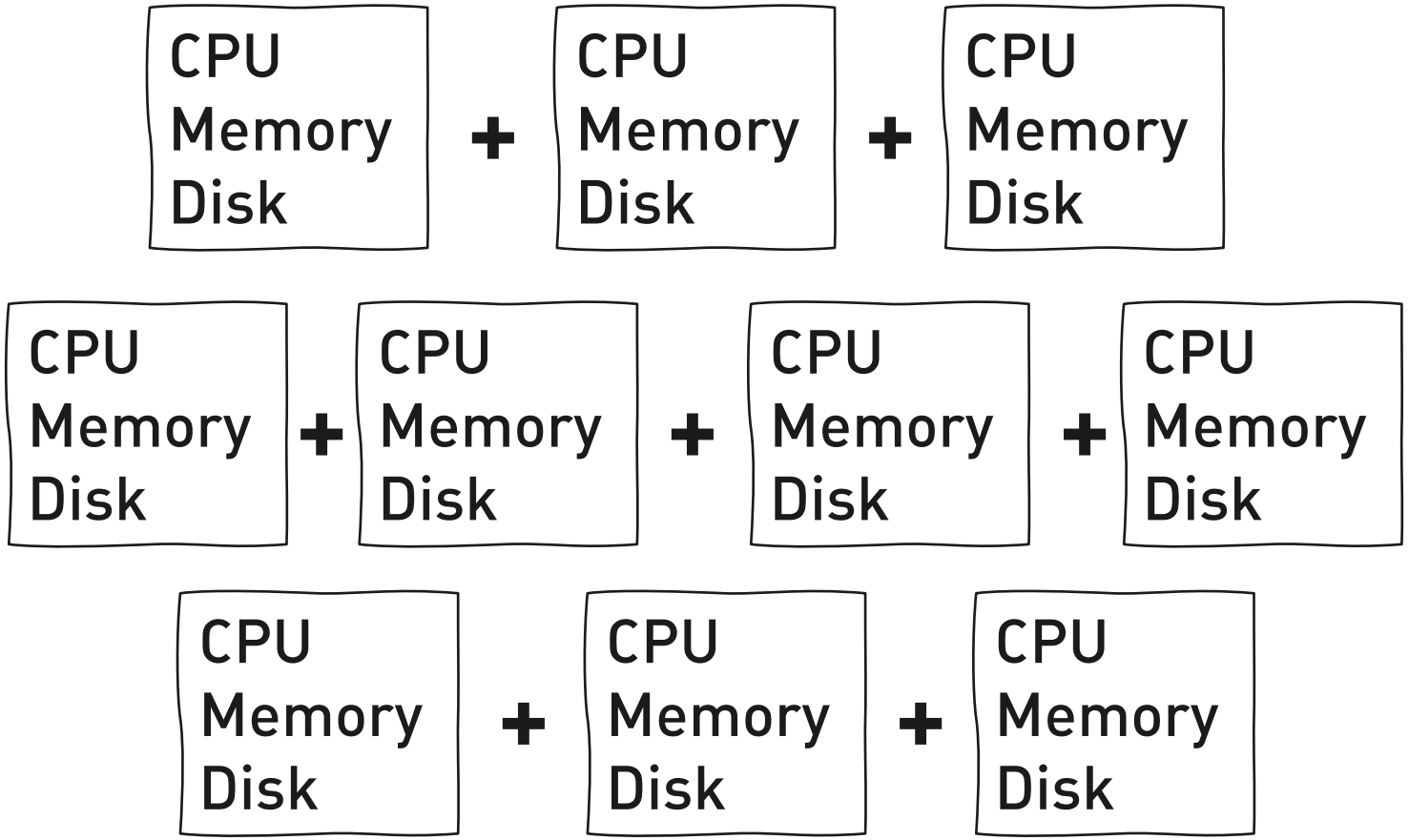
Common example:

"Cache hit ratio" has been going down

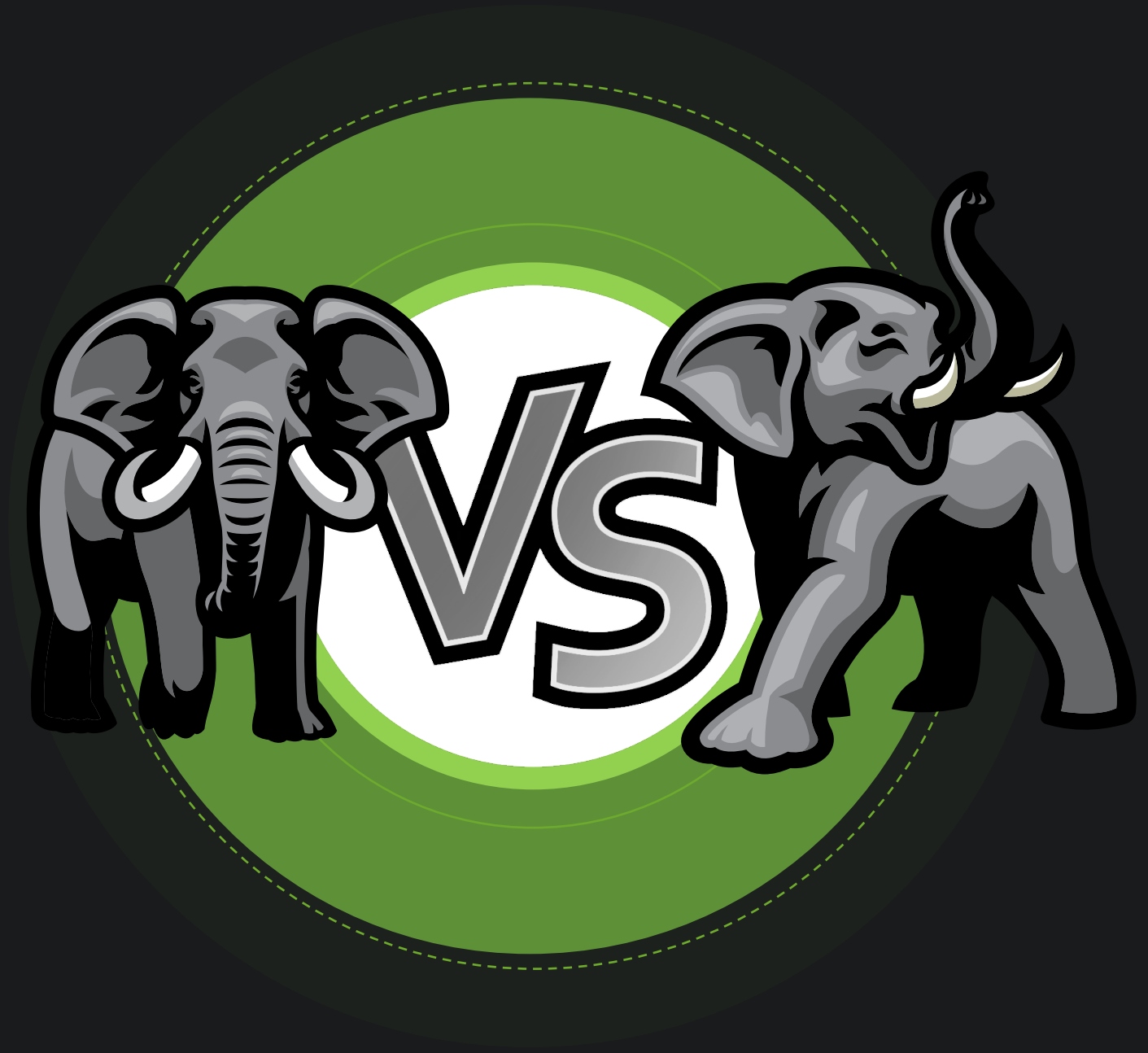
CPU
Memory
Disk

When sharding can help improve query performance

When your application benefits from more cpu, memory, or disk...



So what are the takeaways?



Takeaways about PG Partitioning & Sharding

1. Partitioning & Sharding can be "Invaluable" and "Lifesavers"
2. You don't have to pick between Partitioning & Sharding
3. Planning required
4. Keys Matter

Takeaways about PG Partitioning & Sharding

1. Partitioning & Sharding can be "Invaluable" and "Lifesavers"

2. You don't have to pick between Partitioning & Sharding

3. Planning required

4. Keys Matter

Takeaways about PG Partitioning & Sharding

1. Partitioning & Sharding can be "Invaluable" and "Lifesavers"

2. You don't have to pick between Partitioning & Sharding

3. Planning required

4. Keys Matter

Takeaways about PG Partitioning & Sharding

1. Partitioning & Sharding can be "Invaluable" and "Lifesavers"
2. You don't have to pick between Partitioning & Sharding
3. Planning required
4. Keys Matter

People to thank for inspiration &or reviews

Daniel Gustafsson

David Rowley

Isaac Alves

Adam Wølk

Jelte Fennema-Nio

Marco Slot

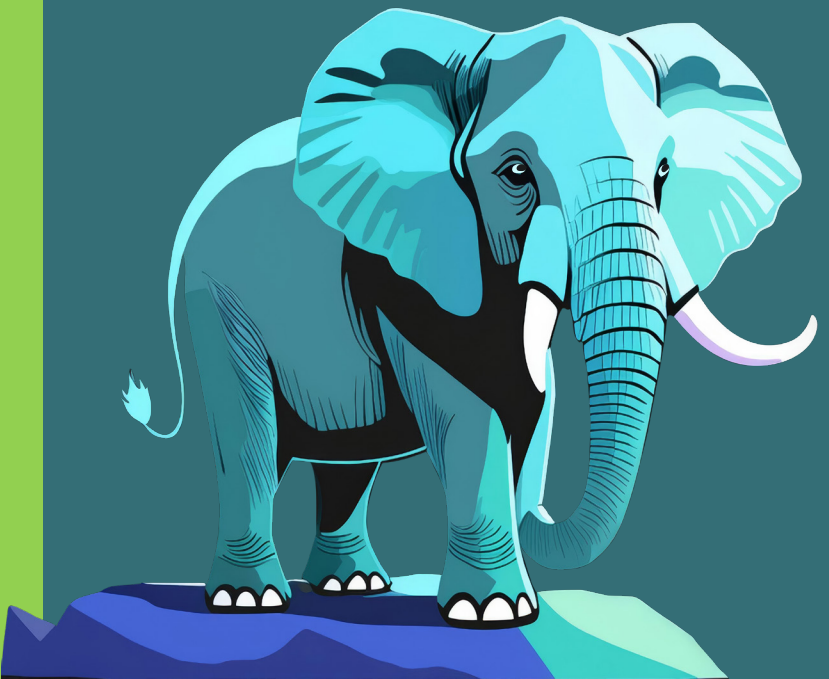
Melanie Plageman

Robert Treat

Ryan Booz

Thomas Munro

Charles Feddersen



3

Favors to ask you!

Our Talking Postgres podcast about human side of PG

Recent guests: Robert Haas, Daniel Gustafsson, Affan Dar, Andrew Atkinson, Tom Lane, & Melanie Plageman...

TALKING POSTGRES

WITH CLAIRE GIORDANO

TalkingPostgres.com

Save the date
June 10-12, 2025

POSETTE: An Event for Postgres

2025 Now in its 4th year!

A free & virtual developer event

Check out the schedule → PosetteConf.com

Microsoft

Stop by the Microsoft booth before it's all over!

Thank you

@clairegiordano@hachyderm.io

@clairegiordano.bsky.social

@clairegiordano

linkedin.com/in/claireg

TalkingPostgres.com

PosetteConf.com

PGConf India

Microsoft