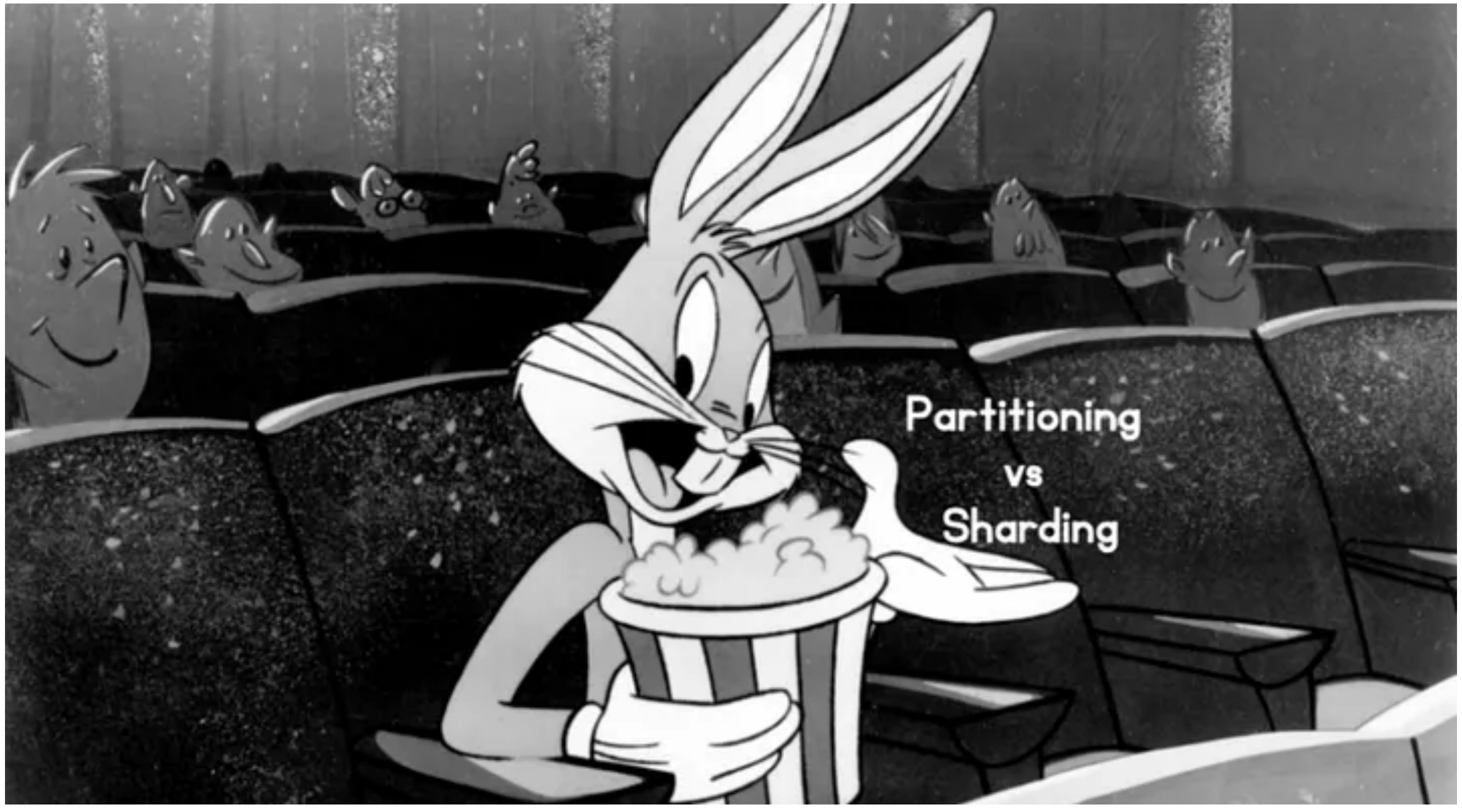
Partitioning & Sharding — choosing the right scaling method

Aman Arora Follow 5 min read · Oct 6, 2023

115 1



Partitions and shards are commonly confused and used interchangeably, yet they are distinct terms with separate meanings.

Partitions and shards are both techniques used in distributed database systems, but they serve slightly different purposes and are often implemented in distinct ways.

Database Sharding and Partitioning both offer intuitive solutions to address a common challenge — managing and querying the vast volumes of data generated by modern applications.

Let’s delve into each of them individually to help you determine which one best suits your needs with ease.

Database partitioning

Database partitioning involves subdividing our database entities (tables, indexes) into smaller entities within the same physical machine.

Doing this allows Dev and DBA to manage these entities easily and provides management and access at a finer level of granularity.

Each individual sub-entity is called a **partition**, and each partition can be accessed, managed, and configured separately.

Why Partitioning?

For Manageability
After partitioning, operations like data loads, index creation/rebuilding,

backup/recovery, etc., can be done at the partition level instead of the complete table. This also results in a reduction in time for all of these operations.

For Performance
In many cases, the result of a query can be achieved by querying a partition as a subset of partitions instead of the complete table.

Partition pruning can help provide a significant improvement in performance.

> *In* ***partition pruning****, the SQL optimiser analyses the FROM and WHERE clauses in query to eliminate unneeded partitions when building the partition access list.*

For Availability
Since data is now divided across multiple smaller entities instead of one, maintenance, recoverability times, and failure impact are lower.

Who partitions?

Partitioning can be done only when both the DBA and the developer are involved. A good understanding of the business rules and how data is loaded and queried by the application is necessary before proceeding with partitioning.

Is DB Partitioning a good fit for you?

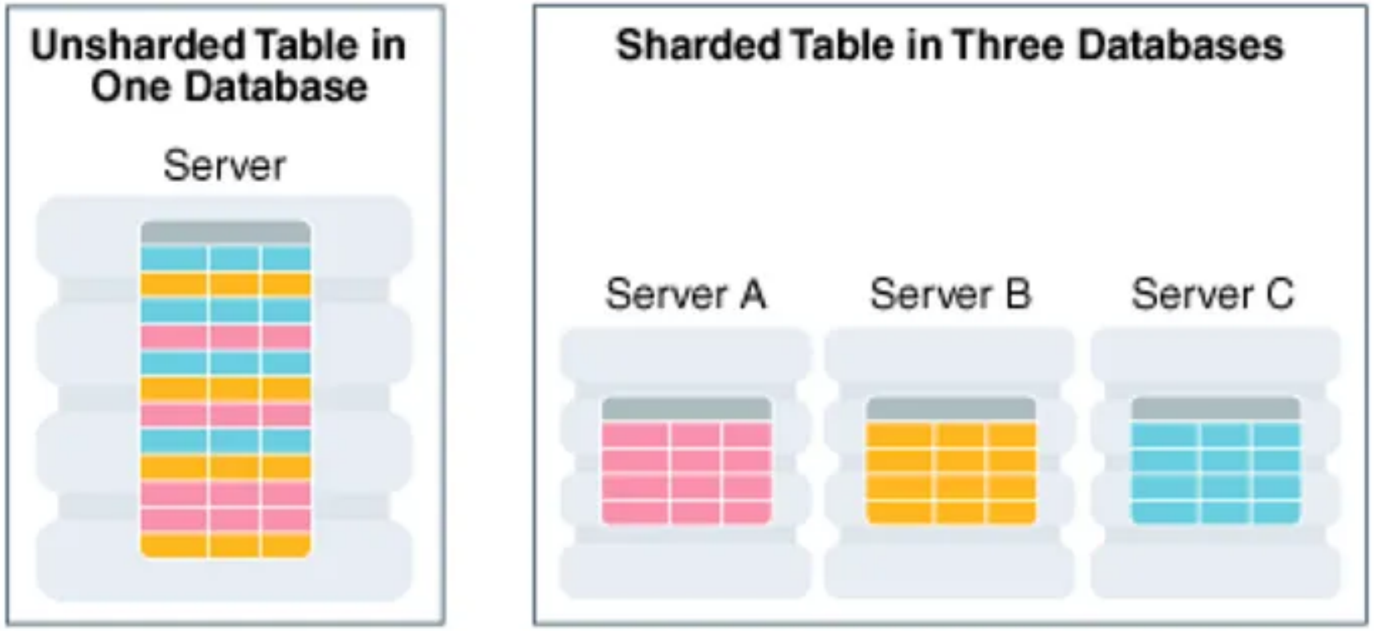
To decide whether you need database partitioning or not, consider the following -

1. **Data Volume:** If you have a large amount of data, partitioning can help you manage and organize it efficiently.
2. **Query Performance:** If you have slow-running queries, partitioning can improve performance by allowing you to target specific data subsets.
3. **Data Growth:** If your data is growing rapidly, partitioning can help you manage the growth and ensure that performance remains consistent.
4. **Data Maintenance:** If you need to regularly archive or delete old data, partitioning can make these tasks more efficient.
5. **Resource utilisation:** If you have limited hardware resources, partitioning can help you balance the load and distribute resources more effectively.

However, it is important to consider the trade-offs and potential drawbacks of partitioning, such as **the added complexity and potential difficulties** in managing and maintaining partitions.

Sharding

In sharding, tables are also divided into smaller tables, but instead of having the data on the same physical machine, the data is spread across multiple server instances.


Unsharded Table in One Database
Server
Sharded Table in Three Databases
Server A
Server B
Server C

Client Side vs Server Side Sharding

Client-side database sharding refers to a method of distributing data across

All your favorite parts of Medium are now in one sidebar for easy access.

Okay, got it

Profile

Stories

Stats

Following

Gaurav Goel

The Medium Blog

Reshma Bidikar

Find writers and publications to follow.

See suggestions

responsible for directing queries to the appropriate database.

In contrast, **server-side database sharding** is a method of distributing data across multiple databases on the server side, meaning that the server infrastructure is responsible for directing queries to the appropriate database.

Some **advantages of server-side sharding** include —

1. ease of maintenance
2. improved performance and scalability
3. simplified query logic

However, it can also lead to a more complex architecture and higher operational overhead.

On the other hand, client-side sharding can offer greater flexibility and control over the data distribution, as well as improved performance and latency in certain cases.

Sharding Sphere

GitHub — apache/shardingsphere: An ecosystem for transforming any database into a distributed database system, enhancing it with sharding and elastic scaling. Sharding Sphere can be used in two ways:

1. **As a Proxy Layer:** It sits between your application and the database, making it language-agnostic, similar to what twemproxy did for Redis.
2. **As a Java Dependency**: It can be directly integrated into Java applications, eliminating the need for separate deployment.

Vitess

Vitess is a tool that automatically optimises queries to improve database performance. It also utilizes caching mechanisms to enhance query mediation.

MySQL doesn’t natively support sharding, but as our application grows, so

> does our need for larger databases that scale. Vitess saves us from having to add sharding logic to your application.

Can I use both simultaneously ?

Yes, it is possible to use both sharding and partitioning in a distributed database system, and this approach is known as "**partitioned sharding**" or "**sharded partitioning**"

However, it's essential to understand that using both techniques simultaneously can add complexity to your database architecture and require careful planning and management.

Here's how partitioned sharding works:

1. **Sharding:** You distribute your data across multiple server instances or nodes (shards). Each shard holds a portion of the overall data.
2. **Partitioning:** Within each shard, you further subdivide the data into smaller, manageable partitions. These partitions are typically organized based on specific criteria, such as ranges of values or other logical

divisions.

The combination of sharding and partitioning allows for a fine-grained control over data distribution and query optimisation. This approach can provide benefits such as improved query performance and easier management of large datasets.

However, it’s crucial to consider the added complexity and potential challenges that come with partitioned sharding:

- **Design Complexity:** You need to carefully design your data distribution strategy, considering both sharding and partitioning criteria. This can be complex, especially for large and complex datasets.
- **Query Routing**: Query routing becomes more intricate, as you must route queries to the correct shard and partition based on the query’s requirements.
- **Maintenance and Scaling**: As your database grows, adding new shards and partitions while maintaining data integrity can become more challenging.

- **Data Migration:** Migrating data between shards and partitions can be more complex than in a system that uses only one of these techniques.
- **Resource Overhead:** Managing multiple shards and partitions can require more server resources and operational overhead.

In some cases, using either sharding or partitioning alone may suffice, while in others, combining both techniques may provide the best solution for your scaling needs.

. . .

If you enjoyed this article and found it valuable, please show your support by liking, sharing, and following me on Medium and Linkedin.

You can follow me on LinkedIn by clicking here

Your likes keep me motivated to write more while sharing helps us reach a wider audience.

Thank you for your support. Peace out✌️

Database Performance Development Scaling Programming

Written by Aman Arora

144 followers · 35 following

Follow

Responses (1)

Ishu

What are your thoughts?

Tomas Pinto
Aug 22, 2024

Thanks Aman, clear explained and well written!

53 1 reply Reply

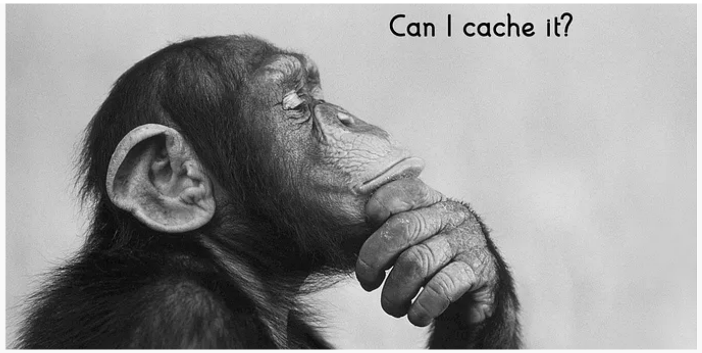
More from Aman Arora

Aman Arora

Amazon DynamoDB — Basics

This blog is a first in a multi-part series about DynamoDB. Being the first one it has been...

Jun 30, 2022 56



Aman Arora

Catching up with cache !

What is Cache?

Jun 6, 2022 90

Aman Arora

Effective Code Reviews

Aman Arora

Handling errors in backend

Code reviews are an essential part of any software development team, it helps you...

Jan 17, 2023 124 3

Errors in software applications are both unavoidable and expected. However, it is ou...

Sep 13, 2023 10

See all from Aman Arora

Recommended from Medium

Nikhil Gupta

Google Interview | Senior Software Engineer | Level 5 | Data Structur...

Implement a Birthday Reminder

Oct 19 24 1

The Latency Gambler

I Interviewed 20+ Engineers. Here's Why Most Can't Code

Over the past year as a Senior Software Engineer at a B2B SaaS company, I've...

Sep 9 2.9K 95

Subodh Shetty

Anubhav

Designing Reliable Distributed Systems: Transactional Outbox +...

If you're not a Medium member, you can still enjoy the full article for free. Click here to rea...

May 15

System Design Day 3: Mastering Application Layer Protocols—...

Welcome to Day 3 of the System Design Interview Series! In Day 2, we laid the...

May 5

In ITNEXT by Animesh Gaitonde

Solving Double Booking at Scale: System Design Patterns from Top...

Learn how Airbnb, Ticketmaster, and booking platforms handle millions of concurrent...

Oct 8 1.91K 22

NonCoderSuccess

Top 20 System Design Concepts Every Developer Should Know...

Whether you're building your first app or scaling a system to millions, understanding...

May 26 206 1