**Sharding vs. partitioning: What’s the difference?**

Sharding and partitioning are techniques to divide and scale large databases.

Sharding distributes data across multiple servers, while partitioning spits tables within one server.

Sharding and partitioning are common ways to improve performance, manageability, and availability of larger databases.

[**What is sharding?**](https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference#what-is-sharding-)

Sharding, also known as horizontal partitioning, is a database partition approach that divides the database schema and distributes them across multiple instances or servers into smaller parts that are faster and easier to manage.

When database is sharded, a replica of the schema is created. This is then used to divide data to be stored in a shard based on a shard key.

To make this possible, a special logic or identifier called a “Shard key” is used to determine which specific instance or server holds the data to query.

How does sharding work?

Consider a scenario of a social media platform with millions of users worldwide. In this case, we can implement sharding based on geographical regions. E.g. instance1 will have data of users in North America, while instance2 will have data of users in Europe, and so forth.

A basic implementation of database sharding based on geographical regions in MySQL. The users table holds user data, while the user\_regions table maps each region to a specific database instance. This allows for distributing user data across multiple database instances based on their respective regions.

A screenshot of a computer program

Description automatically generated

Query to retrieve the user record for the username “johndoe” from database instance responsible for storing users in the North America region.

A computer screen shot of a code

Description automatically generated

By geographically dividing the data, sharding allows for localized access and efficient management of user information. This approach proves particularly beneficial when it comes to optimizing performance for user interactions. Imagine a user in Europe trying to retrieve their profile information. Instead of traversing through the entire database, the system can use a shard key to quickly pinpoint the specific shard (instance) where the data is located, leading to faster response times and a better user experience.

However, it's important to consider certain factors to ensure fair distribution of data across instances. The varying user populations across different regions should be taken into account. For instance, North America may have a significantly larger user base compared to other regions. To address this, a more intelligent sharding strategy can be used, such as a combination of geographical and demographic factors. This way, the distribution of data can be more balanced and reflective of the user distribution across the regions.

**What are the advantages of using sharding?**

* Improved response time.
* Increased read/write throughput.
* Increased storage capacity.
* Availability.
* Reduced outage time.

**What are the disadvantages of using sharding?**

* Sharding brings the complexity of managing tables that are distributed across multiple servers.
* It can be difficult to manage database queries.
* As data grows merging shards can become more complicated to handle.
* Using wrong sharding architecture can lower the performance.
* Solution : Using **Vitess**, this technology helped scale YouTube, Etsy, GIthub.

**What is partitioning?**

It is nothing but spitting up the database, it is commonly used to mean “vertical partitioning”. i.e. diving tables in a database into smaller sub-smaller tables or partitions.

These partitions can be accessed or managed separately to enhance performance, maintainability, and availability of the databse.

A screen shot of a computer program

Description automatically generated

The provided code shows how to partition a database table based on the id column. The PARTITION BY RANGE clause is used to divide a table into multiple partitions. In this case, there are three partitions defined. There is partition p\_0, p\_1, and p\_2.

A black background with white text

Description automatically generated

Insert user data into the users table:

A screen shot of a computer screen

Description automatically generated

Now that the partitions are created, you can retrieve user data from the appropriate partition based on the ID range.

A screenshot of a computer program

Description automatically generated

[**What are the advantages of database partitioning?**](https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference#what-are-the-advantages-of-database-partitioning-)

Vertical partitioning a database allows you to distribute data across multiple physical or logical storage units called partitions. By dividing the data, you can improve query performance by reducing the amount of data that needs to be scanned or accessed. Database partitioning adds efficiency to querying the database. It also makes maintenance operations easier. When a database is partitioned, you can target a specific partition to query a record, rather than traversing through the entire dataset. For example, if you partition a database by date, queries that only need to access data from the last month can be executed much faster than if they had to access all the data in the database.

Accessing a database in parts can give security control. If you are storing some confidential information in the database, you can allow a certain group of users to access only partitions that don't have confidential information.

[**What are the disadvantages of database partitioning?**](https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference#what-are-the-disadvantages-of-database-partitioning-)

One of the disadvantages of database partitioning is the complexity it brings. Partitioning can streamline certain maintenance tasks, but it can also complicate other aspects. Complexity can increase the chances for errors. For instance, the management of backups and recovery procedures can become more difficult to handle when you have multiple partitions. It can also lead to a false sense of security: If you're not careful, having multiple partitions could lead to a data loss disaster. Juggling partitions can lead to wasted space, and it may be unnecessary for the average user.

[**What are the differences between sharding and partitioning?**](https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference#what-are-the-differences-between-sharding-and-partitioning-)

While sharding and partitioning share the common goal of dividing a large database into smaller ones, they have different approaches to achieve this. When sharding a database, the data is distributed across multiple servers, resulting in new tables spread across these servers. On the other hand, partitioning involves splitting tables within the same database instance. Sharding is referred to as horizontal scaling, and it makes it easier to scale as you can increase the number of machines to handle user traffic as it increases. Partitioning splits based on the column value(s). All columns should be retained when partitioned – just different rows will be in different tables. It is also easier to manage data with partitioning, as all partitions are in one database instance.

[**Conclusion**](https://planetscale.com/learn/articles/sharding-vs-partitioning-whats-the-difference#conclusion)

In conclusion, both sharding and partitioning are powerful techniques that enable scaling and efficient data management in large databases. By understanding their differences and considering factors such as data distribution, performance optimization, and manageability, you can choose the most appropriate approach for your specific database architecture. Implementing sharding or partitioning can significantly enhance the performance and scalability of your database, allowing it to handle increasing user traffic and serve millions of requests effectively.

# How is sharding different from partitioning?

You're hosting a massive dinner party, but instead of one long table, you decide to go all out and have multiple smaller tables scattered around your house. Now, sharding and partitioning are like your strategies for seating your guests.

Partitioning is like dividing your guests into groups based on some criteria, let's say age or dietary preferences. So, you might have a table for the young ones who want chicken nuggets, another for the sophisticated adults who want fine wine and cheese, and maybe a table for the vegans who refuse to touch anything that had a face. Each group gets its own space, but they're still in the same house.

Now, sharding takes it up a notch. Imagine if your dinner party got so huge that you had to rent out multiple houses just to fit everyone. Sharding is like that. You split your guests across different locations entirely. So, instead of cramming everyone into your cozy abode, you spread them out across various venues. That way, no single house has to bear the burden of accommodating the entire horde of hungry revelers.

So, in short, partitioning is like organizing your guests into different sections of the same house, while sharding is like outsourcing your dinner party to multiple venues to avoid overcrowding and chaos. Whether you're talking about databases or dinner parties, the goal is the same: to manage the crowd without losing your sanity!