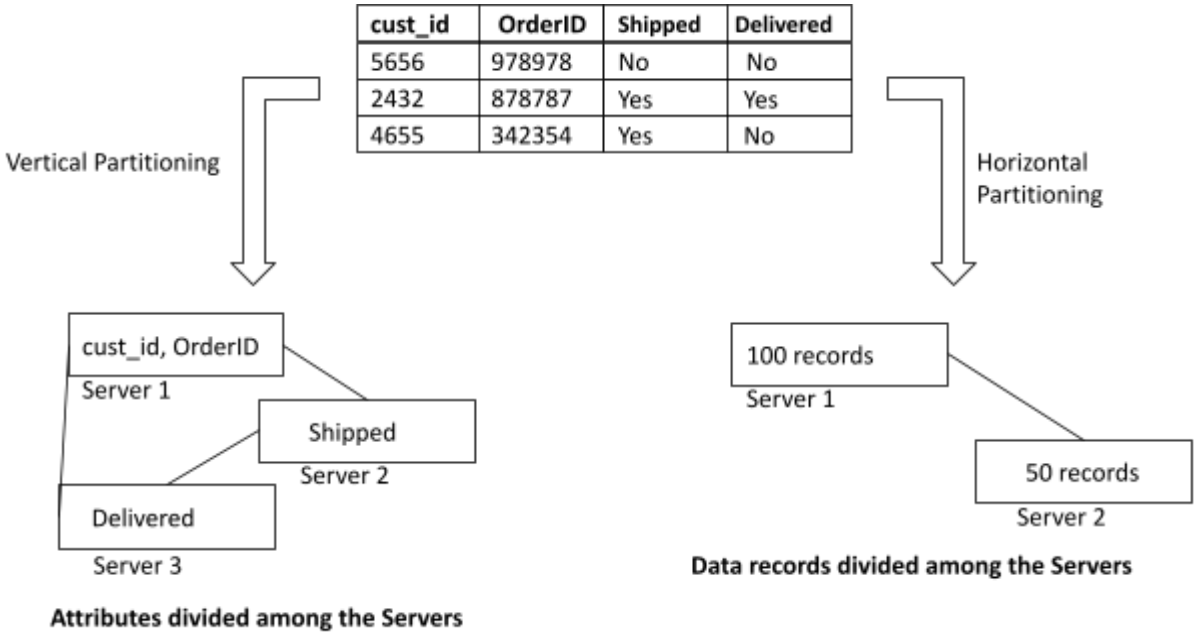
Partitioning:
It is a technique used to divide stored database objects into separate servers. Due to this, there is an increase in performance, controllability of the data. We can manage huge chunks of data optimally.

When we horizontally scale our machines/servers, we know that it gives us a challenging time dealing with relational databases as it's quite tough to maintain the relations.
But if we apply partitioning to the database that is already scaled out i.e. equipped with multiple servers, we can partition our database among those servers and handle the big data easily.

Now to partition the data they are two techniques:

- **Vertical Partitioning**: In this, we partition the given data vertically i.e column wise. So, if we are provided with table students with attributes student id, name, courseid, address, we can store this data by distributing it among servers where we store studentid, name is one server, courseid in another and address in the third one.
- **Horizontal Partitioning**: In this, we partition the given data horizontally i.e. row wise. So, have chunks of a certain size of data stored at different servers.
  Like, to refer to the same example as given in the above point. Now we will be storing different numbers of entries or rows in different servers as we store the first 100 entries in server 1 and other 100 entries (i.e. from 101 to 200) in another server (attributes remain the same.).

Let’s visualise it with an example:



Above is an image representing how data partitions happen vertically and horizontally.

Some Advantage of Partitioning:

- it plays a key role in building systems with extremely high availability requirements.
- Improves performance and manageability.
- It helps in reducing the cost of storing a large amount of data.
- We can upload rarely used data into cheap data storage devices.

Sharding:

It is a method of partitioning and storing a single logical set of data in multiple databases which is stored on multiple machines/servers.
It is an extension to Horizontal Partitioning.
The smaller chunks of the data that are created after Shrading are called Shards.

cust_id	OrderID	Shipped	Delivered
5656	978978	No	No
2432	878787	Yes	Yes
4655	342354	Yes	No
1443	656677	Yes	No

Shrad 1

cust_id	OrderID	Shipped	Delivered
5656	978978	No	No
2432	878787	Yes	Yes

Shrad 2

cust_id	OrderID	Shipped	Delivered
4655	342354	Yes	No
1443	656677	Yes	No

On Shrading, the table is divided into “Shards”, each new table has the same schema but unique distinct tuples.

Why Shrading ?

Let’s take a very basic example: Consider a giant database of college/university. In this database, all the details of students, being a present student or an alumni, exists.

So, it might contain a huge number of entries, let's say if a college has been there for 100 years and has had 1000 thousands in each batch. The total number of students will be 1,00,000.

Now when we need to fetch some details of a student from this Database, each time around 1 lakh transactions have to be done to find that student, which seems awful.

Now, if we divide the data into ‘Shrads” batch wise, where each shrad contains nearly 1000 entries.

Now the transaction cost is reduced by the factor of 100. As now we have to fetch the information from only 1000 entries and that’s possible because of Shrading.

Advantages:

- High availability of the data.
- Increase in Storage capacity
- Read/Write operations speed increased.

Disadvantages:

- The cost of infrastructure required increases
- Complexity for administration to maintain the database increase

Hence, Sharding can be a good solution if one needs to scale their database horizontally. But with few benefits it does bring complexity in application. It might be necessary at some places but the money and time to maintain such a model could be overwhelming as compared to benefits at certain situations.