Database Sharding

Database Management System
Sir Sajjad Nawaz

BSIT-M2 (2020-24)
Muhammad Usman Razwan 48
Haiqa Mushtaq 13
Nabrass Gull 08
Anas Inam 47

November 1, 2022

Abstract

The following paper describes some huge problems with the scaling of databases and how those issues can be solved or avoided by a technique in which we break the data into small partitions or shards and distribute those shards across our network of servers. This not only helps us to combat the issues with scaling of the database but also provides better latency and a secure database.

Database Scaling

INTRODUCTION

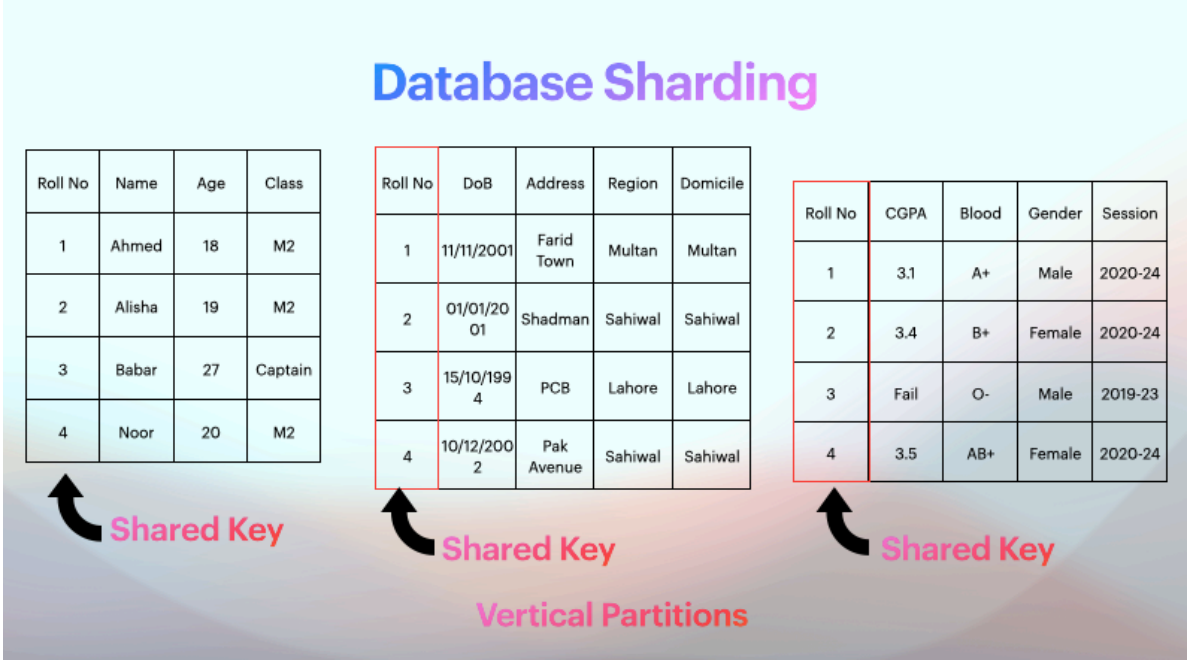
Scaling of the DBMS is defined as "**the ability to expand the capacity of a database system in order to support larger amounts or requests and/or store more data without sacrificing performance**" [1]. With a growing user base, the load on our database will most likely grow as our application will save a larger amount of data. Whether it's the number of connections needed, the amount of data stored, or the increased processing power, any database will eventually hit a limit of what it can and can't handle. Increased usage of our application brings three main common problems to our database server:

1. The CPU or memory becomes overloaded, and the DB server cannot respond to all requests, or at least not in a reasonable time.
2. Database server runs out of storage and cannot store all data.
3. Network gets overloaded, and latency is increased.

We can address these issues by an increased capacity server or by optimizing our application code, caching, and indexing query patterns. These optimizations will increase the efficiency of our application and will bring us some relief, but a time will come when system recourse limits will be reached. At this point, we will have to consider scaling our database horizontally, vertically, or both.

DATABASE SHARDING

Database Sharding can be defined as "Database sharding is the process of storing a large database across multiple machines. A single machine, or database server, can store and process only a limited amount of data. Database sharding overcomes this limitation by splitting data into smaller chunks, called shards, and storing them across several database servers. All database servers usually have the same underlying technologies, and they work together to store and process large volumes of data." [2]. In simple words, we can say that We partition or divide the database into small parts, chunks, Shards, etc. Then these shards are distributed in multiple servers, and the index of shards is stored on a single computer. The request is first sent to the main computer where the index of shards is stored, and then that computer routes the request to the respective machine where that respective data is being held. Data can be shared vertically, horizontally, etc.



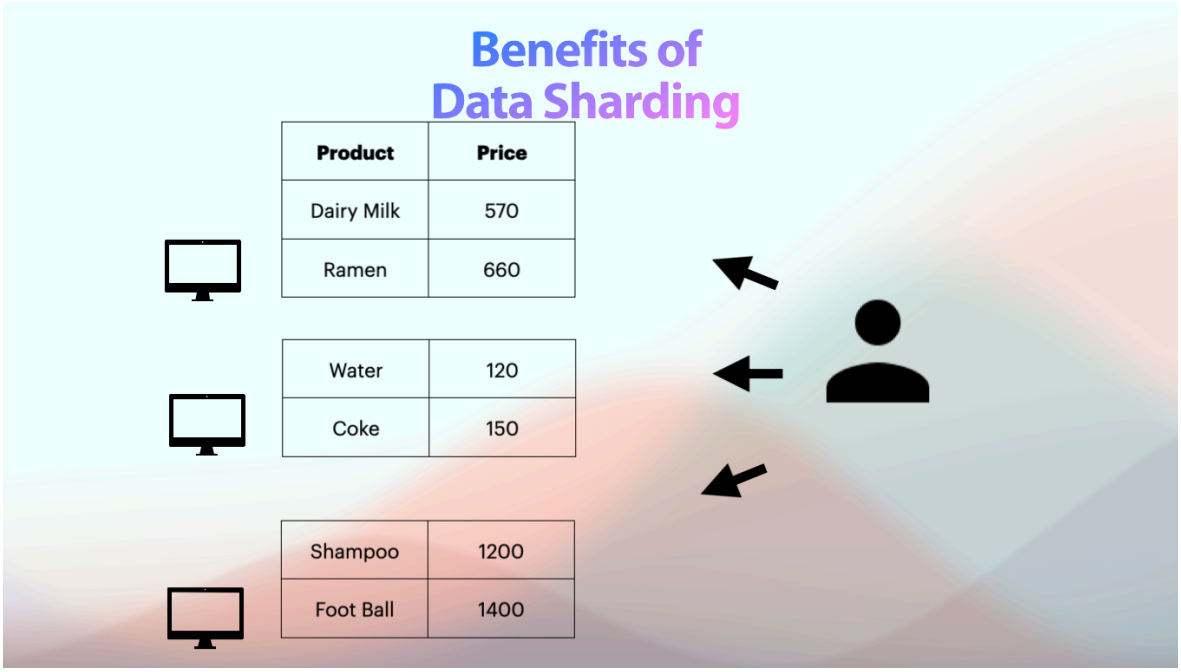
BENEFITS OF DATABASE SHARDING

Sharding allows us to scale our database to handle the increased load to a nearly unlimited degree providing Increased Read/Write throughput, Storage capacity, and high availability.

- **Increased read/write throughput** — By distributing the dataset across multiple shards, read and write operation capacity is increased as long as read and write operations are confined to a single shard.

- **Increased storage capacity** — Similarly, by increasing the number of shards, you can also increase overall total storage capacity, allowing near-infinite scalability.

- **High availability** — Finally, shards provide high availability in two ways. First, since each shard is a replica set, every piece of data is replicated. Second, even if an entire shard becomes unavailable since the data is distributed, the database remains partially functional, with part of the schema on different shards.



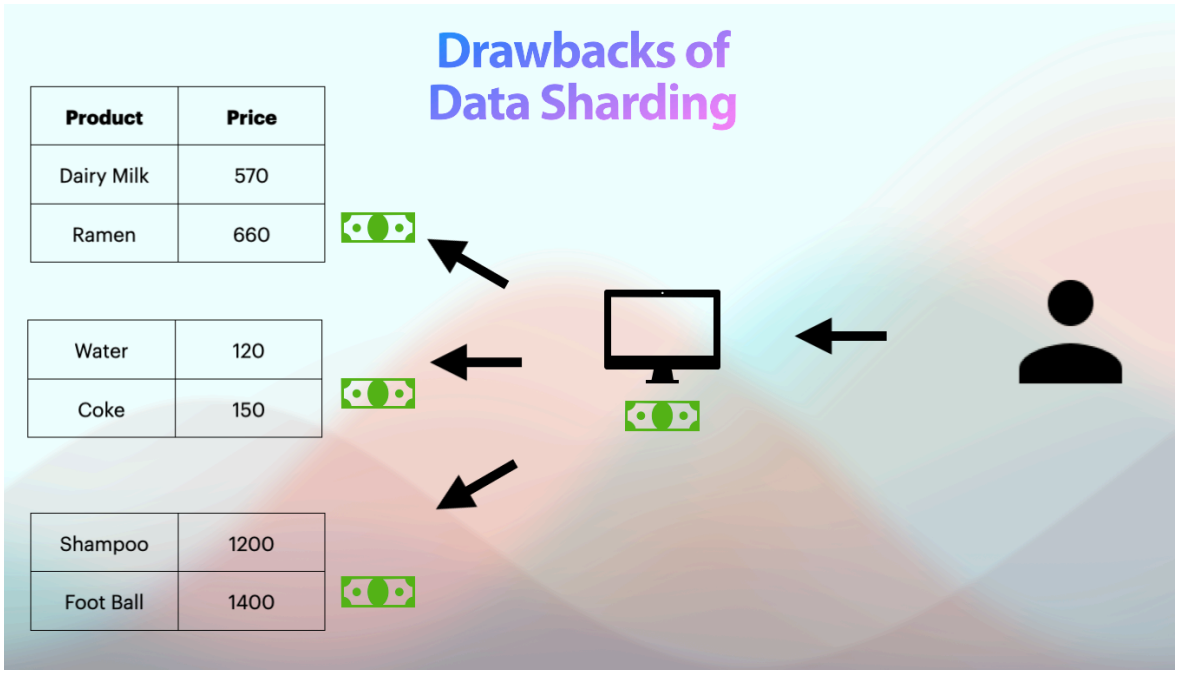
DRAWBACKS OF DATABASE SHARDING

There're always drawbacks to something in this world, Same way Database Sharding also has some drawbacks.

- **Query overhead** — Each sharded database must have a separate machine or service which understands how to route a querying operation to the appropriate shard. This introduces additional latency on every operation. Furthermore, if the data required for the query is horizontally partitioned across multiple shards, the router must then query each shard and merge the result. This can make an otherwise simple operation expensive and slow response times.

- **The complexity of administration** — With a single unsharded database, only the database server requires upkeep and maintenance. With every sharded database, on top of managing the shards themselves, additional service nodes must be maintained. Plus, any data updates must be mirrored across each replicated node in cases where replication is being used. Overall, a sharded database is a more complex system that requires more administration.

- **Increased infrastructure costs** — Sharding, by its nature, requires additional machines and computing power over a single database server. While this allows your database to grow beyond the limits of a single machine, each additional shard comes with higher costs. The cost of a distributed database system, especially if it is missing the proper optimization, can be high.



Adoption

If we see closely, many big enterprise-level organizations are adopting this technique not only because of the advantages but also because they do not have to hire a new team of developers because many huge Database providers have started to roll out this functionality with just a click of a button which makes it very easy for the big organizations to adopt this technology. Today we will talk about two giant corporations, Facebook and Google. When Google Started using database sharding, it sharded each of its service's data, such as Gmail, in its shard and stored google meets data in its own shared which not only helped them to store more data but also process and perform more operations on their data. Now, let us talk about Facebook. They shared data region-wise. For example, data from Punjab is to be stored in one shard, and data from Sindh is stored in another datacentre in another shard of data. This helped Facebook gain processing capabilities, leading them to perform more analysis and collect more data, thus, more revenue for them. So it is a win-win situation.

CONCLUSION

Database Sharding can effectively combat the issues we face with scaling databases and might be an ideal way to scale the database to an unlimited degree without worrying about hardware recourses limits. The drawbacks of Database Sharding are not much of an issue for enterprise-level databases or clients.

BIBLIOGRAPHY

1. MongoDB.
2. Amazon Web Services (AWS).