**Database Sharding**

Sharding means partitioning.

Suppose you have only 1 DB which stores 1million users at max.

If, your user base increases more than 1 million, suppose 4 million, in that case, we need to have more number of database and divide our data in multiple DBs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | City | Age | ID | Role |
| Nik | Delhi | 22 | 1 | SDE1 |
| Ansh | Mumbai | 12 | 2 | Intern |
| Sharad | Pune | 42 | 3 | PM |

If we divide our data (tables) by keeping **different columns** in different DBs, this is called **vertical partitioning**.

If we divide our data such that, **different number of rows** are stored on different DBs, this is called **horizontal partitioning**.

**Horizontal Partitioning = Database Sharding.**

**Shard** is a subset of data.

When we divide 4 million users into 4 shards, each of 1 million, we say, that we have created 4 logical shards, each of 1 million.

Now I can store 4 million users on two physical DBs (physical shards), where each DB will store 2 logical shards that is 2 million users each.

Advantages of sharding

1. While querying the data, searches become faster, we can query multiple instances of DB at same time.
2. We can store data at geographically different locations, so that users querying around an area will get faster result when they have a physical shard present near them.
3. If anyone physical shard goes down, we ca still serve other users till that time.

Sharding Strategies.

**Algorithmic Sharding**

When the application knows, which shard it needs to communicate and queries to that particular shard. This is called algorithmic sharding where app takes care of determining shard.

S4

S3

S2

2

S1

APP

**Dynamic Sharding**

Here, application talks to a middleware called Service Locator which looks up into all shards and determines, which shard the query needs to be sent too.

Here, app is not responsible for determining the shard. We can add and remove shards in this strategy, hence called dynamic sharding.

S2

2

S1

S3

S4

Service locator

APP

Disadvantages of Sharding

1. If we don’t choose a sharding strategy correctly, some shards may have less data, some may have a lot of data (hotspots), which will make querying slower.
2. It’s very difficult to switch back to non-sharded architecture, to combine all data. Sharding should be last way of optimization after caching and all other ways. Once data is sharded, it can’t be merged.
3. If a query requires join operation on multiple shards, then it can be quite complex and time taking process.

**Key Based Sharding**

This is algorithmic sharding strategy where application decides which shard to visit for read() or write() any operation.

In this, each row of a table is given a unique shard key.

Shard key is not primary key.

But, a primary key can qualify to become a shard key as well.

Suppose, the primary key (let’s say Roll No. of student Table) is shard key.

We currently have 3 shards.

The application will pass the roll No through **a hash function** which will direct the operation to Shard 1, 2 or 3.

Advantage:

1. We equally divide all rows in all shards.

Disadvantage:

1. Suppose number of shards increase/decrease then hash function needs to be changed. The rows which went to shard 1 previously, might go to shard 2 now. Thus, if shards are increased or decreased, the data needs to be shifted based on hash function.

**Range Based Sharding**

This is algorithmic sharding strategy where application decides which shard to visit for read() or write() any operation.

Here, data is divided in different shards based on a range.

Suppose there is a sales data of a company and shards are divided in 4 quarters of the year.

So any data generated/ or operation performed, its timestamp will be checked, if it’s of month:

Jan – Mar = 1st Shard, Apr-Jun = 2nd Shard

July-Sep = 3rd Shard, Oct-Dec = 4th Shard

Or in e-commerce website, products can be stored in different shards based on range of price.

Advantage:

1. No need to change shards of existing data when new shards are added.

Disadvantage:

1. Data will not be evenly distributed. Maybe the company has good sales in 2nd quarter but very less sales in 4th quarter, then Shard 2 might become a hotspot.

**Directory based Sharding**

This is a dynamic sharding strategy, where the data is divided based on Zones, Country, or any other parameter.  
Each shard represent a zone and information of all zones and their shards is present in a lookup table.

Look Up

Table

S4

S3

S2

2

S1

APP

Advantage:

1. Easy to add or remove shards.
2. Theoretically, there can be hotspots like in Range Based sharding, but we choose this strategy only when we are sure that our data will be evenly distributed.

Disadvantages

1. Increased latency, app needs to communicate with lookup table first and then with shard. So increased hops.
2. If Look Up table crashes, system crashes.