Database Sharding

- Sharding is **horizontal partitioning** of a database.
- Data is **split** across multiple database instances (shards).
- Each database instance holds a **subset** of the actual data.
- There are multiple ways to split data, such as:
  - **By ID,**
  - **By User,**
  - **By Region, etc.**

Key Features of Database Sharding

1) **Horizontal Scaling**
   - Instead of using a single large database,
   - Data is **distributed** across smaller, independent database instances.
2) **Partitioned Data**
   - Each database instance holds:
     - **Not the entire dataset**,
     - **but A subset of the total data.**
3) **Independent Queries**
   - Instead of running a query on the entire dataset or across all instances,
   - Queries are directed to the **appropriate database instance**.
4) **Reduced Load on a Single Database Instance**
   - Prevents overloading a single database, which improves performance.

Purposes of Database Sharding

- When the dataset is **too large** for a single database server to handle.
  - To **improve query performance**
- By reducing the amount of data processed per request.
- To **distribute load** across multiple database servers.
- For **multi-regional applications**
  - Users can retrieve data from the closest database instance.

Benefits of Database Sharding

1) **Improved Performance**
   - Smaller subsets of data reduce query execution time.
2) **Scalability**
   - New database instances can be added as data and load increase.
3) **Fault Tolerance**
   - A failure in one database instance does **not** impact the entire system.

Drawbacks of Database Sharding

1) **Increased Code Complexity**
   - Requires:
     - More **complex code**
     - Routing queries to the appropriate database instance.
2) **Rebalancing Issues**
   - Some database instances may store **significantly more data** than others.
   - The data needs to be **redistributed** when this happens.
3) **Cross-Shard Queries**
   - Joining data **across multiple shards** is difficult.

Example of Database Sharding

- A table has a **numeric ID** column.

- Let's **split** the data as follows:
- Rows with **even IDs** are stored in **Shard 1**.
- Rows with **odd IDs** are stored in **Shard 2**.

```
SELECT * FROM shard_2.user WHERE ID = 1;  -- Fetches from Shard 2
SELECT * FROM shard_1.user WHERE ID = 2;  -- Fetches from Shard 1
```