Special NoSQL Benefits of MongoDB in this Example

Flexibility

- We can store different amounts of data for each user.
- **Alice** has multiple posts, while **Bob** only has one.
- You can easily add new fields (e.g., "**location**" or "**profile picture**") later without disrupting the structure.

Nested Data

- **Relationships** are modeled using arrays (e.g., **followers**, **following**, **posts**) directly embedded within the user document, making it simpler to retrieve all of a user's data in one query.
- In traditional relational databases, representing the **many-to-many relationship** between users (**followers** and **following**) would require multiple tables (e.g., users, follows, posts). Here, we can use **embedded arrays**, which simplifies querying and **eliminates the need for joins.**

Simplified Queries

- You can still query for relationships (**followers/following**) or **posts** efficiently.
- The documents are more lightweight (than SQL), making them faster to process and query.

No Joins

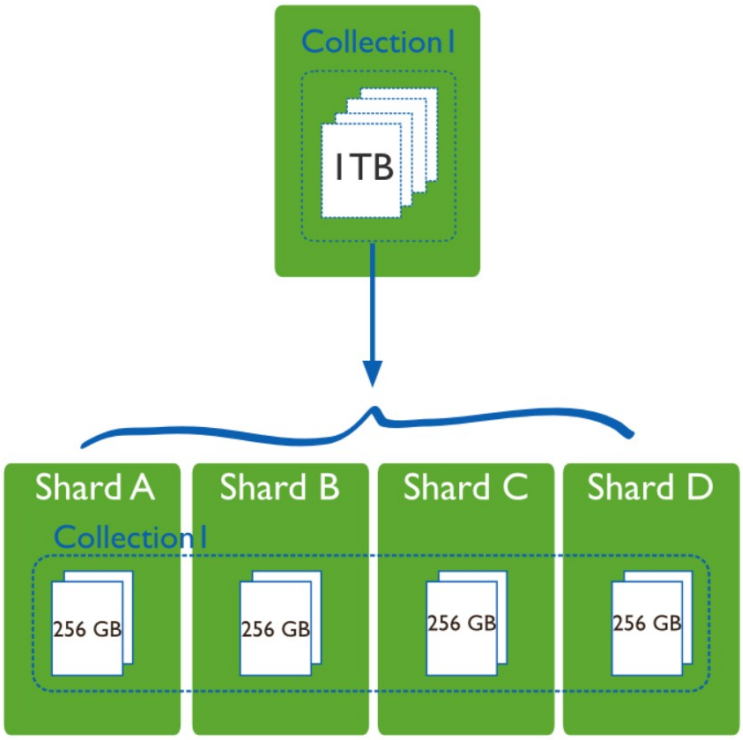
- This structure allows us to access all necessary information in a single document, avoiding the need for joins or complex SQL queries, which is one of the major benefits of NoSQL databases like MongoDB.

Scalability

- MongoDB is designed to **scale horizontally**.
- If the number of users grows significantly, we can easily distribute the data across multiple servers (**sharding**), and MongoDB will handle distributing user data (including **posts**, **followers**, and **following** relationships) automatically across those servers.

Sharding

- Sharding is a method for distributing a single dataset across multiple databases, which can then be stored on multiple machines.
- This allows for larger datasets to be split into smaller chunks and stored in multiple data nodes, increasing the total storage capacity of the system.



- Ranged or Dynamic Sharding, takes a field (_id) on the record as an input and, based on a predefined range, allocates that record to the appropriate shard.