Introduction to Big Data

ECAP456

Dr. Rajni Bhalla

Associate Professor

Learning Outcomes

After this lecture, you will be able to

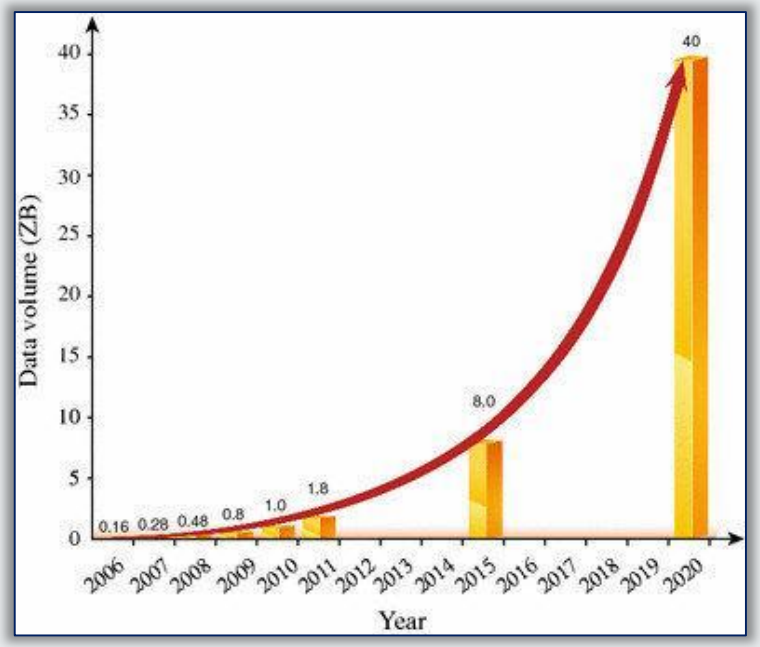
- explore concepts of distribution models
- learn types of distribution data

Distribution Models

Run databases on a large cluster

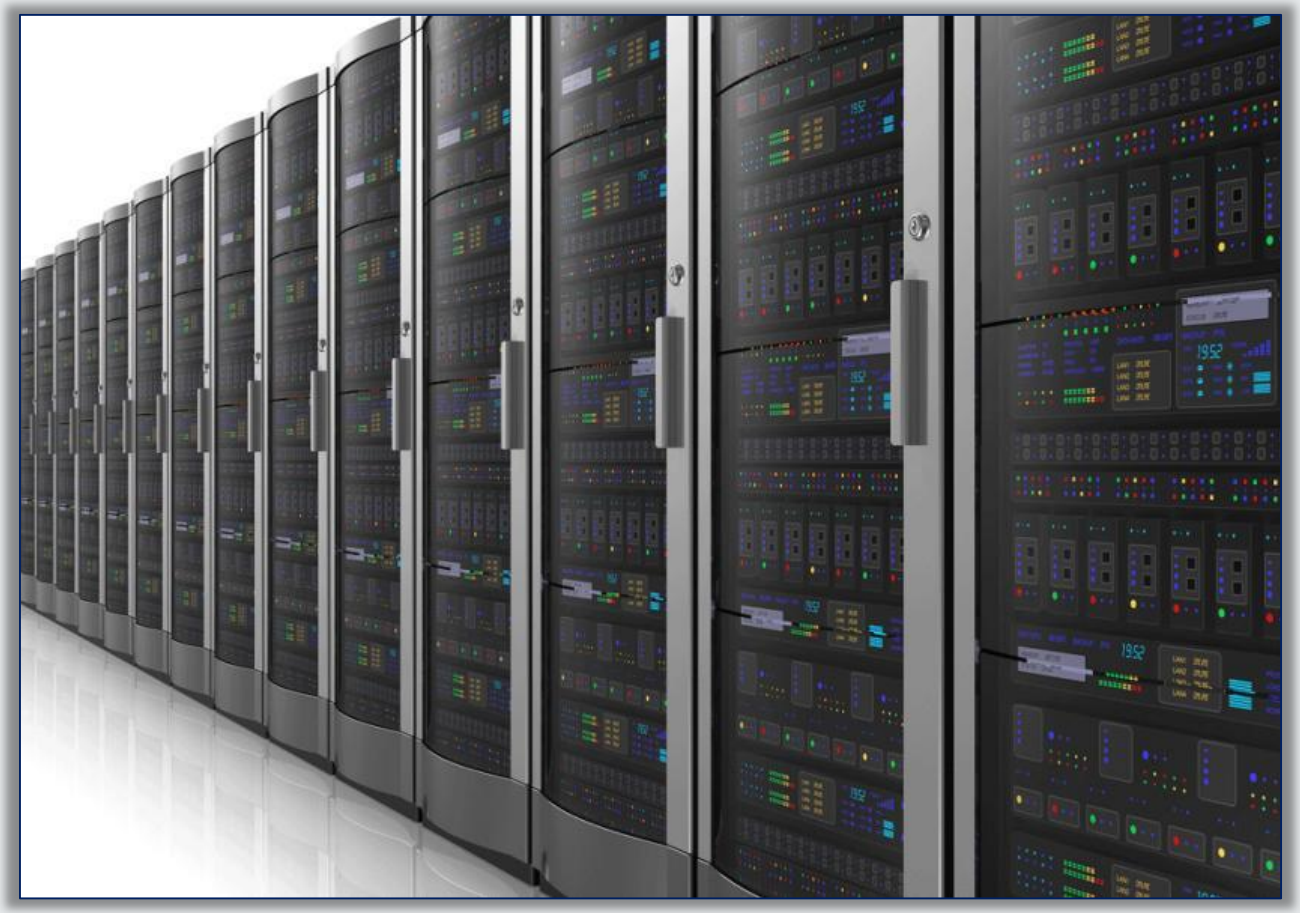
Distribution Models

Run databases on a large cluster



As data volumes increase

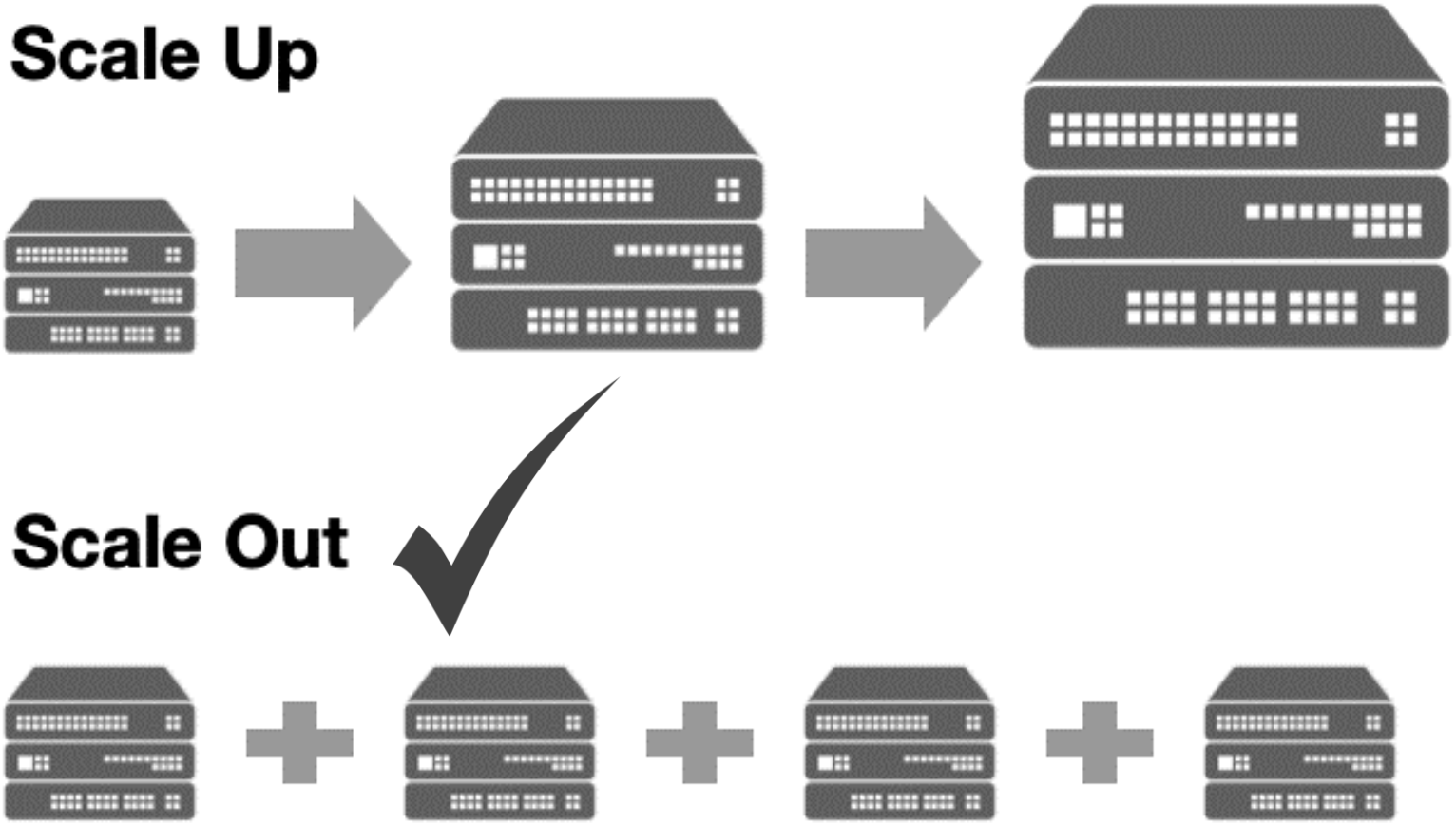
Distribution Models



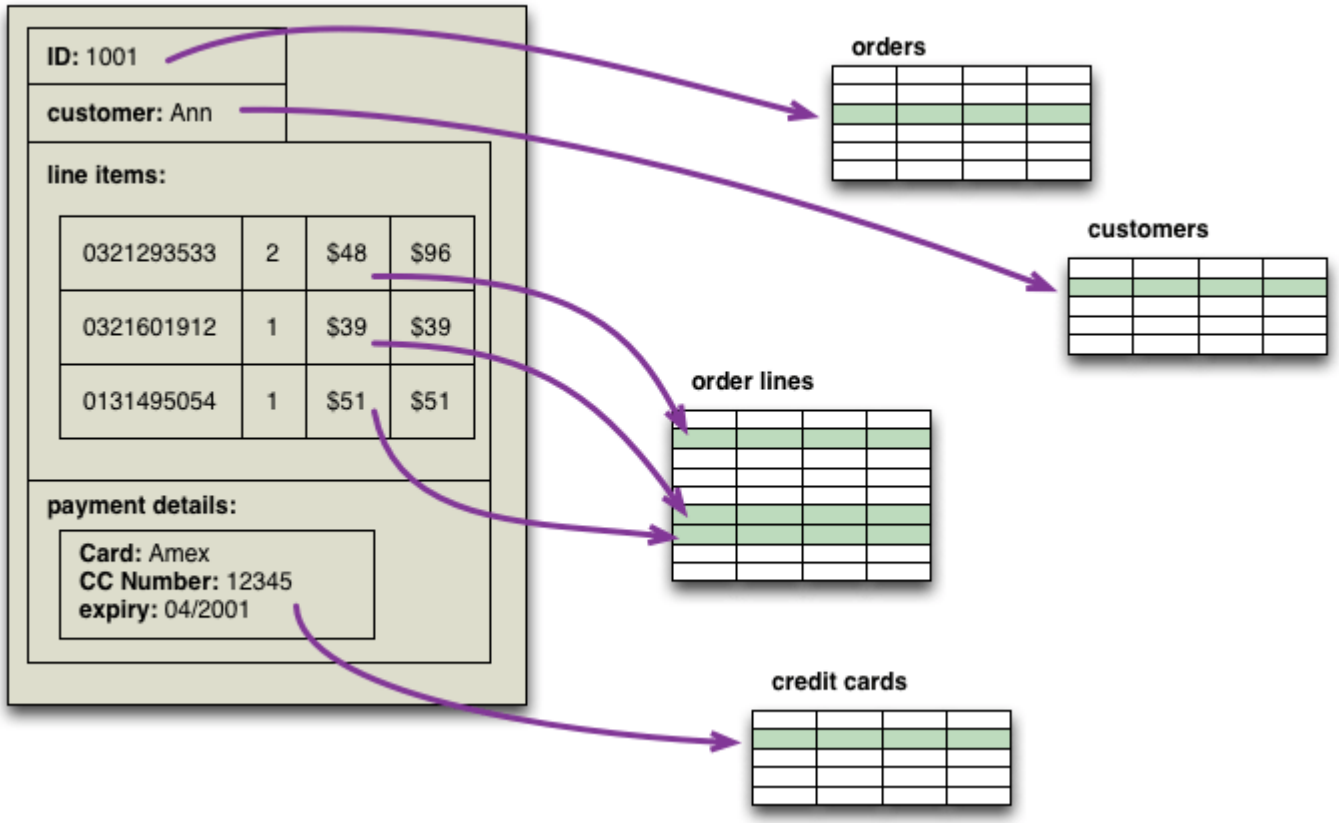
Bigger server to run the database

Distribution Models

A more appealing option is



Distribution Models



Aggregate orientation fits well with scaling out

Distribution Models

Important Benefits of Distribution Model

Handle larger quantities of data

Distribution Models

IMPORTANT BENEFITS OF DISTRIBUTION MODEL

Handle larger quantities of data

Process a greater read or write traffic

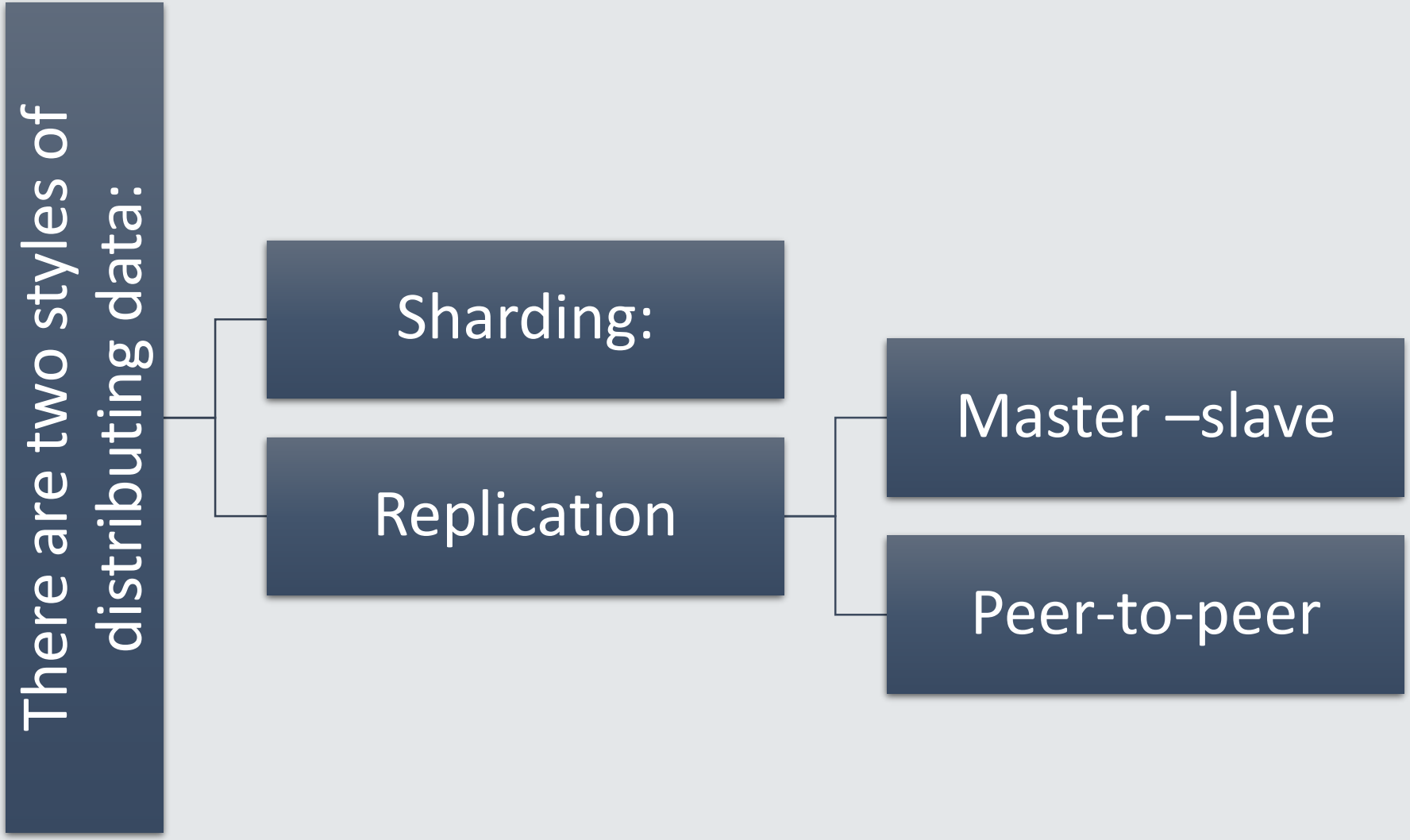
Distribution Models

Network slowdowns or breakages

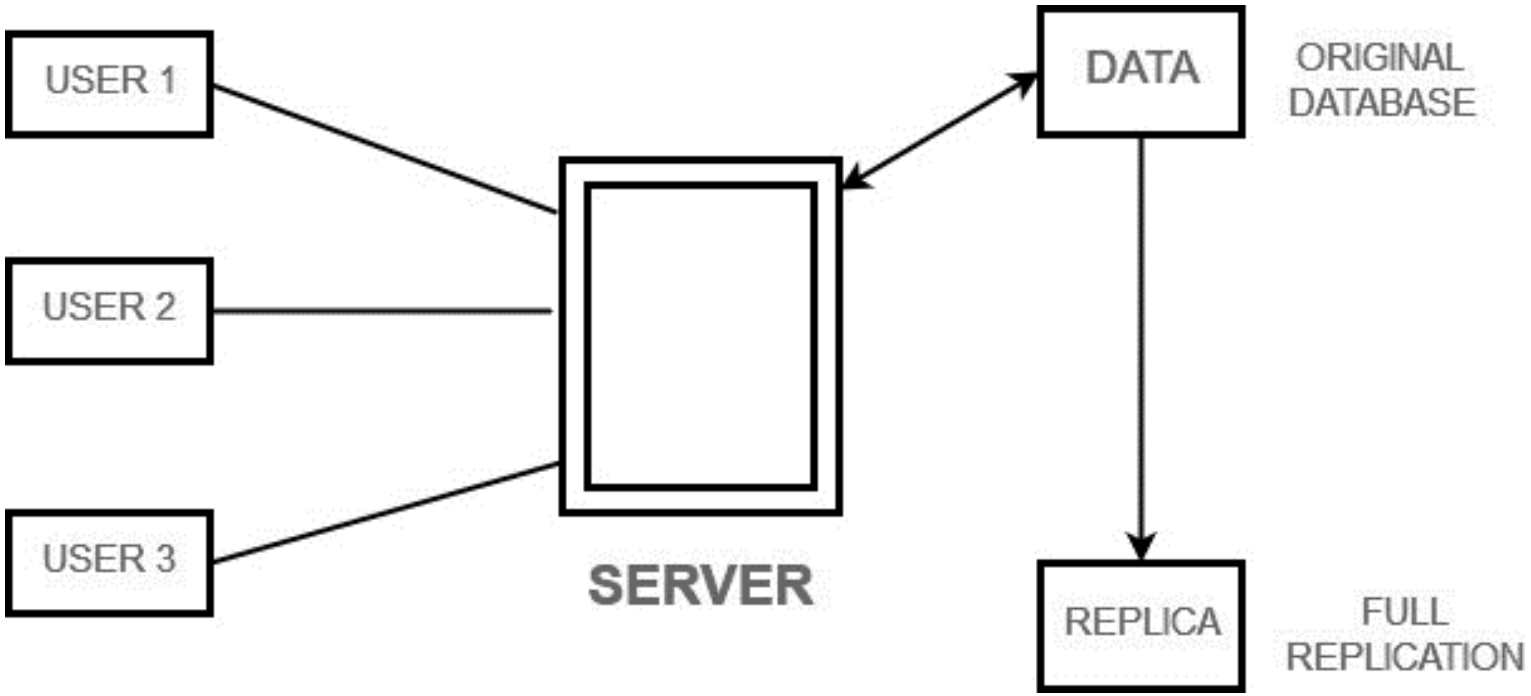
Distribution Models

These are often important benefits, but they come at a cost. Running over a cluster introduces complexity - so it's not something to do unless the benefits are convincing.

Two styles of distributing data

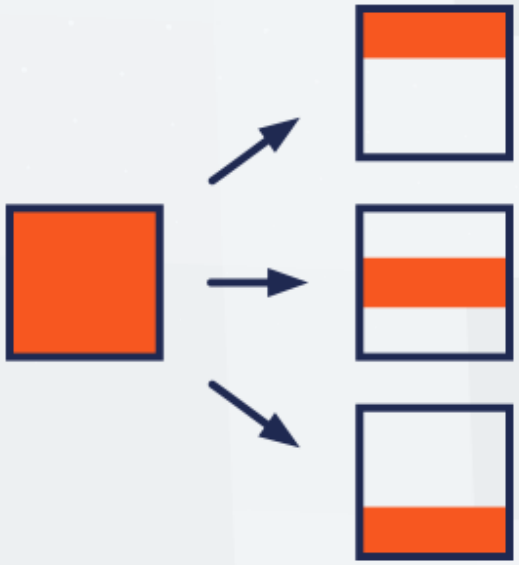


Two styles of distributing data



REPLICATION

Two styles of distributing data



Sharding

Single Server

Single Server

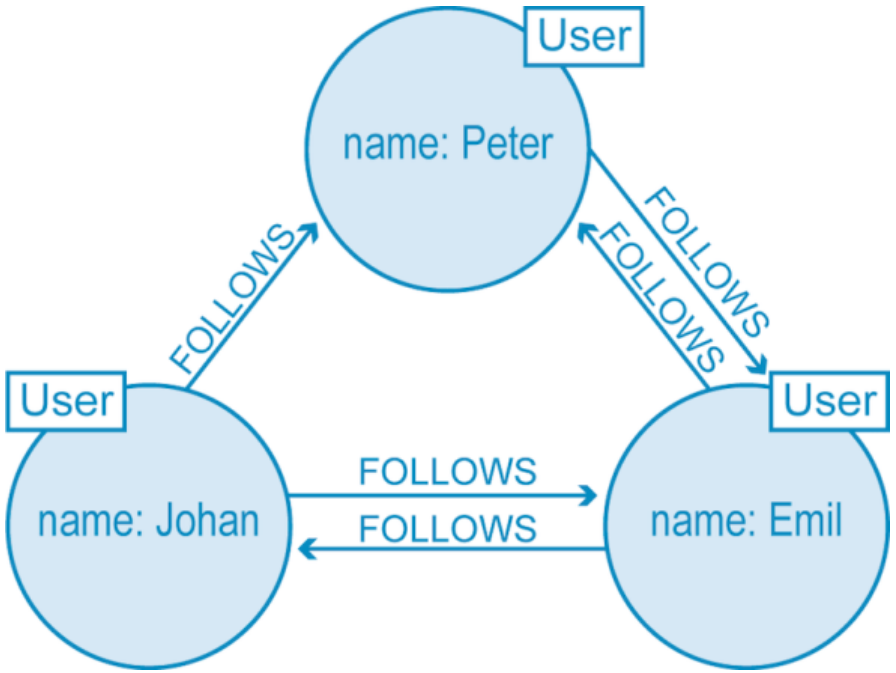
- No distribution at all.
- Run the database on a single.
- It eliminates all the complexities that the other options introduce
- Easy for operations people to manage
- Easy for application developers to reason about.

Single Server

Ques: Which database works best in single-server configuration ?

Single Server

Ques: Which database works best in single-server configuration ?

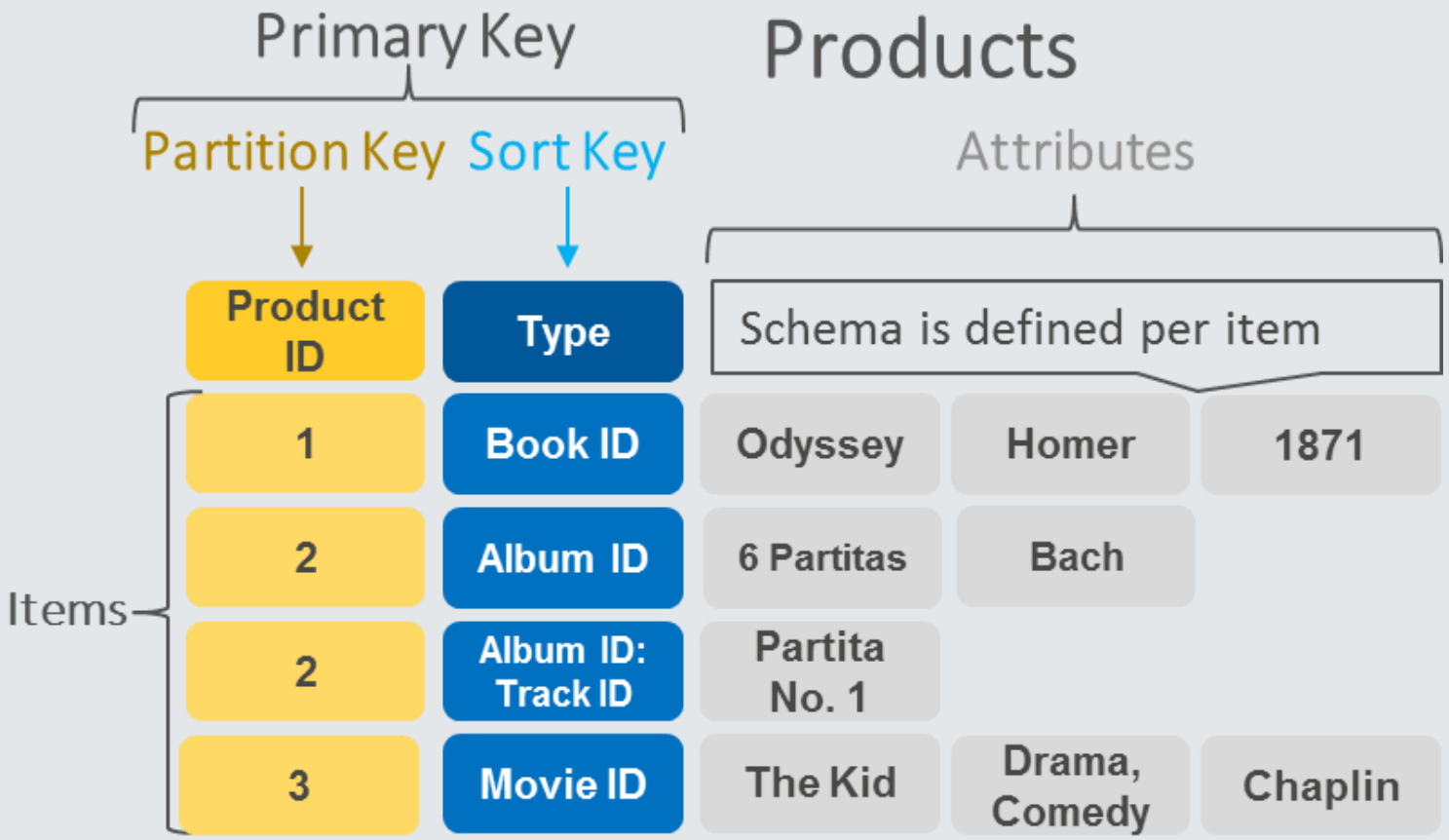


Graph Database

Single Server

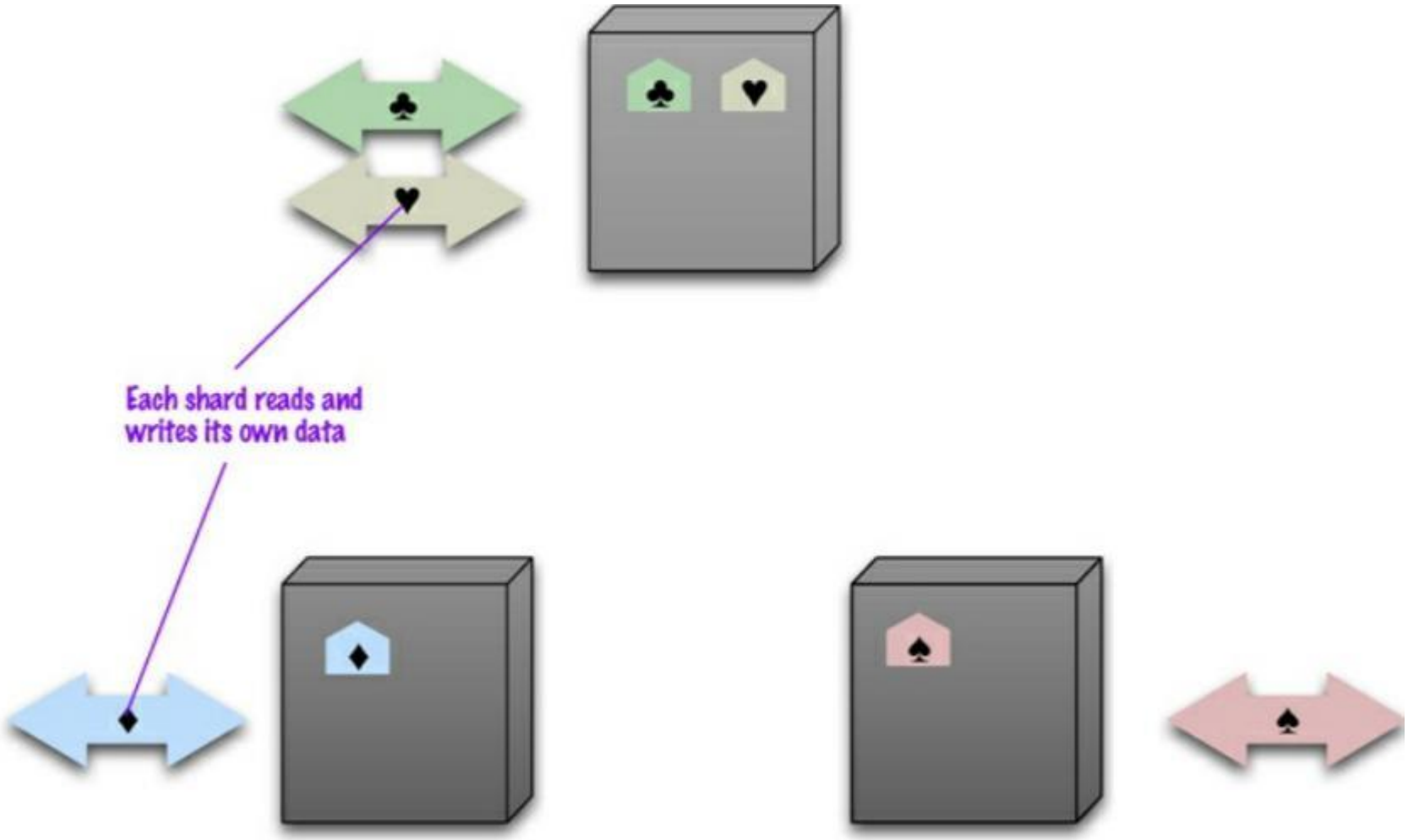
What is the most appropriate solution for processing aggregates in a single-server document?

Single Server



Key-value store

Sharding



Sharding

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical Shards

VS1

CUSTOMER ID	FIRST NAME	LAST NAME
1	Alice	Anderson
2	Bob	Best
3	Carrie	Conway
4	David	Doe

VS2

CUSTOMER ID	CITY
1	Austin
2	Boston
3	Chicago
4	Denver

Horizontal Shards

HS1

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston

HS2

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Why Is Sharding Used?

- Rapid responses from that server.
- The load is balanced out nicely between servers.
- You can store the new chunks of data, called logical shards
- Can take advantage of all the compute resources across your cluster for every query.
- Scan fewer rows when responding to a query.

Why Is Sharding Used?

- Rapid responses from that server.
- The load is balanced out nicely between servers.
- You can store the new chunks of data, called logical shards
- Can take advantage of all the compute resources across your cluster for every query.
- Scan fewer rows when responding to a query.

Why Is Sharding Used?

- Rapid responses from that server.
- The load is balanced out nicely between servers.
- **You can store the new chunks of data, called logical shards**
- Can take advantage of all the compute resources across your cluster for every query.
- Scan fewer rows when responding to a query.

Why Is Sharding Used?

- Rapid responses from that server.
- The load is balanced out nicely between servers.
- You can store the new chunks of data, called logical shards
- **Can take advantage of all the compute resources across your cluster for every query.**
- Scan fewer rows when responding to a query.

Why Is Sharding Used?

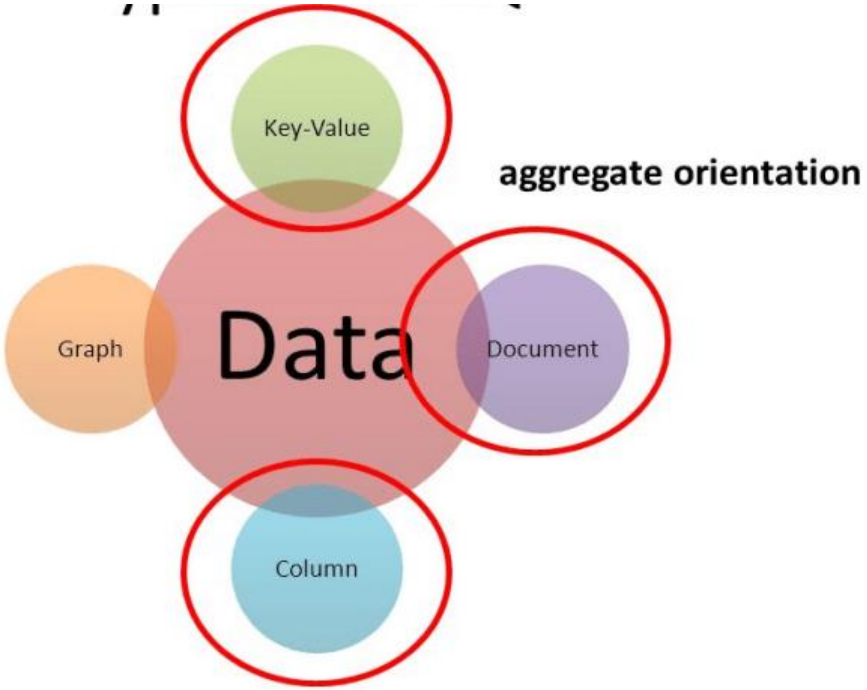
- Rapid responses from that server.
- The load is balanced out nicely between servers.
- You can store the new chunks of data, called logical shards
- Can take advantage of all the compute resources across your cluster for every query.
- Scan fewer rows when responding to a query.

Sharding

How to clump the data up so that one user mostly gets her data from a single server ?

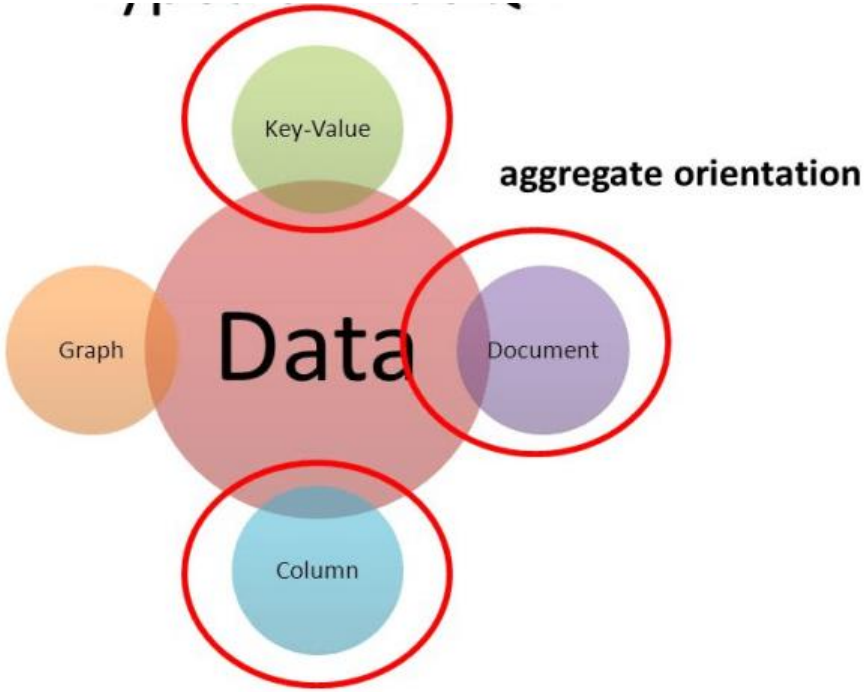
Sharding

- How to clump the data up so that one user mostly gets her data from a single server



Sharding

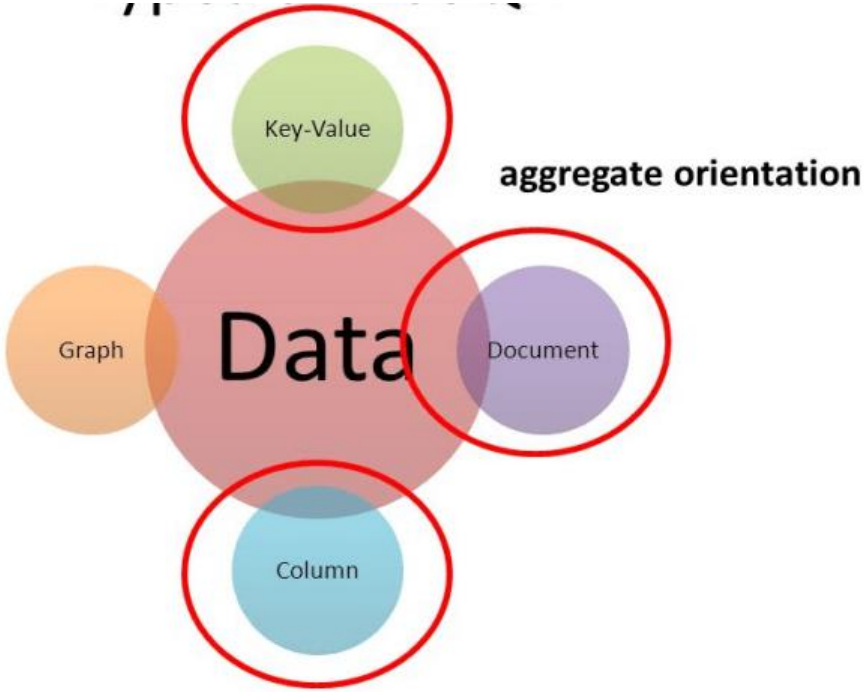
- Combine data that's commonly accessed together.
- There are several factors that can help improve performance.



Sharding

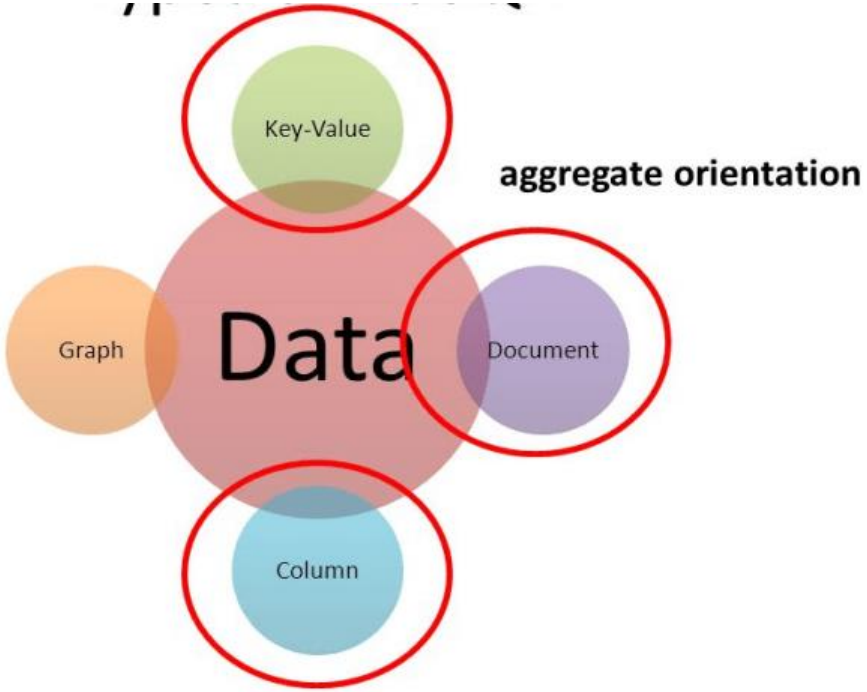
- Combine data that's commonly accessed together.

- There are several factors that can help improve performance.



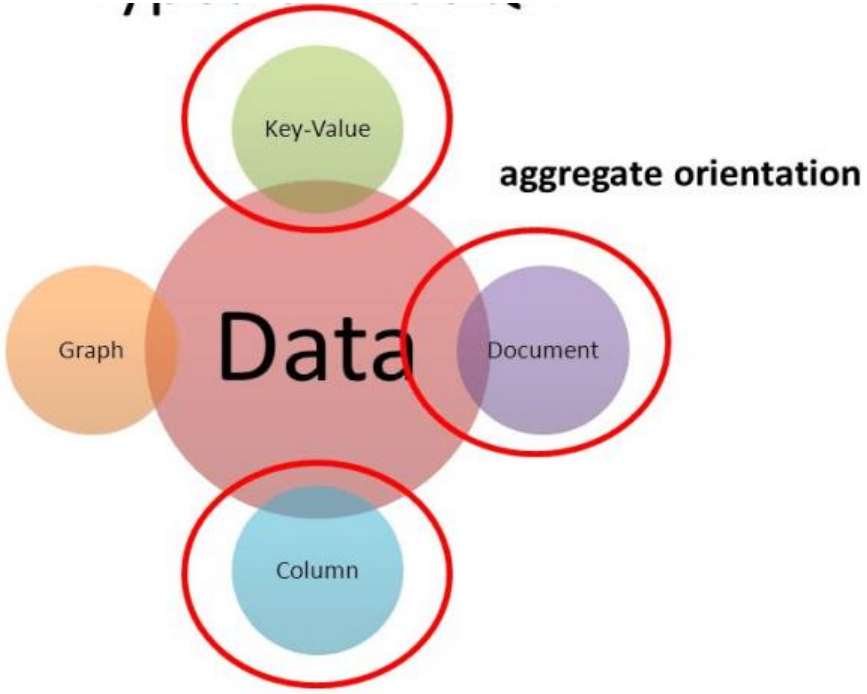
Sharding

- Place the data close to where it's being accessed.
- If you have orders for someone who lives in Boston, you can place that data in your eastern US data center



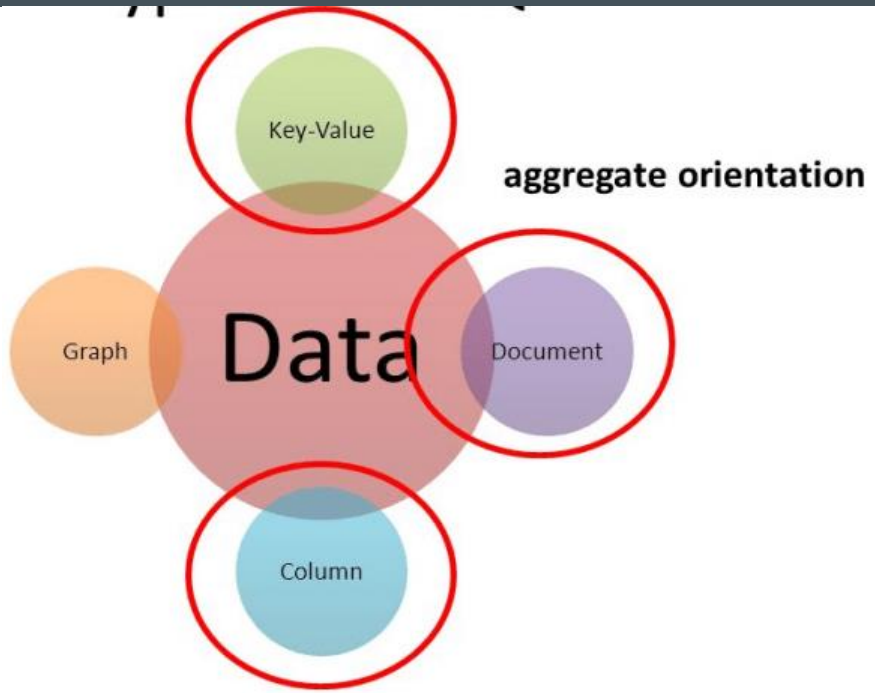
Sharding

- Place the data close to where it's being accessed.
- If you have orders for someone who lives in Boston, you can place that data in your eastern US data center



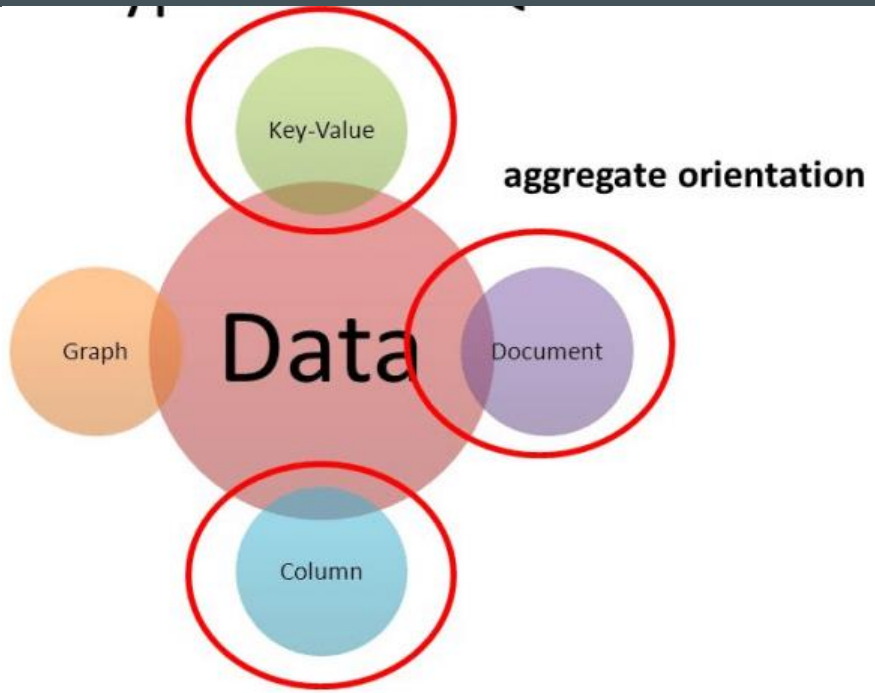
Sharding

- To keep the load even
- Puts Aggregate together
- Keeping its rows in lexicographic order.
- sorting web addresses based on reversed domain names



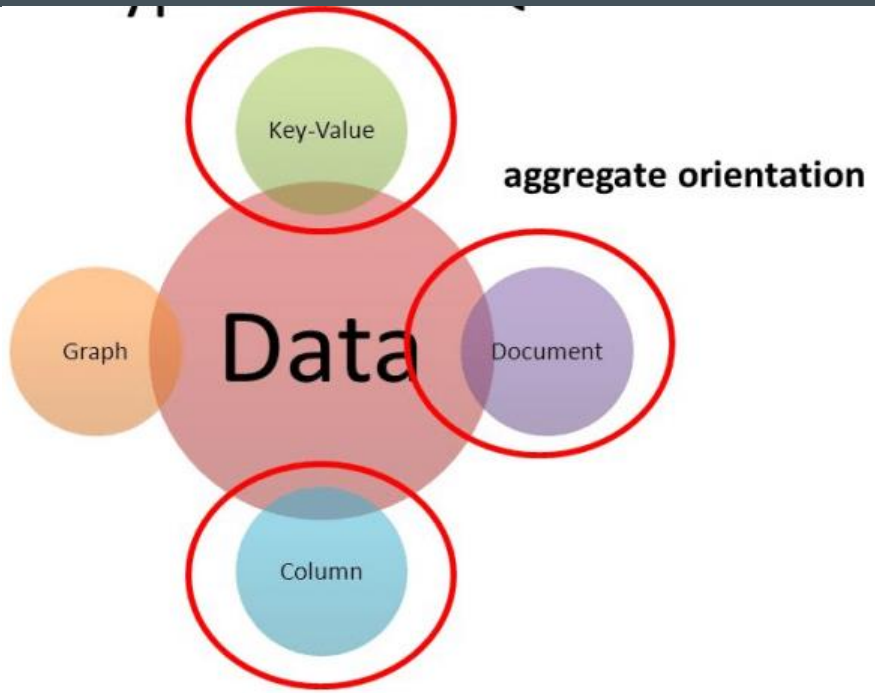
Sharding

- To keep the load even
- Puts Aggregate together
- Keeping its rows in lexicographic order.
- sorting web addresses based on reversed domain names



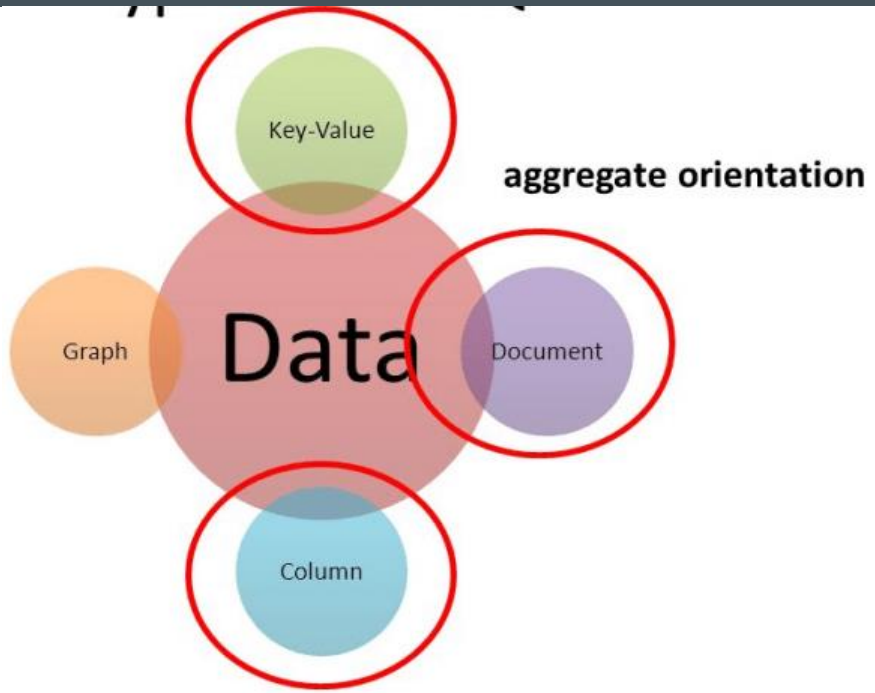
Sharding

- To keep the load even
- Puts Aggregate together
- Keeping its rows in lexicographic order.
- sorting web addresses based on reversed domain names



Sharding

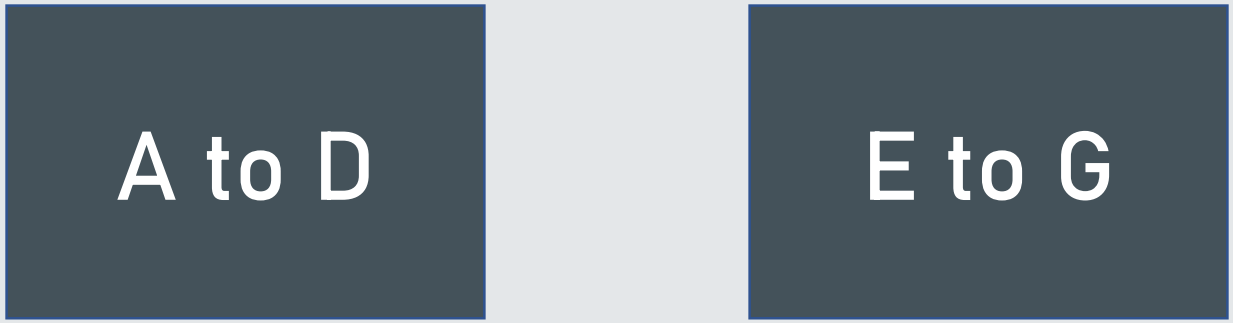
- To keep the load even
- Puts Aggregate together
- Keeping its rows in lexicographic order.



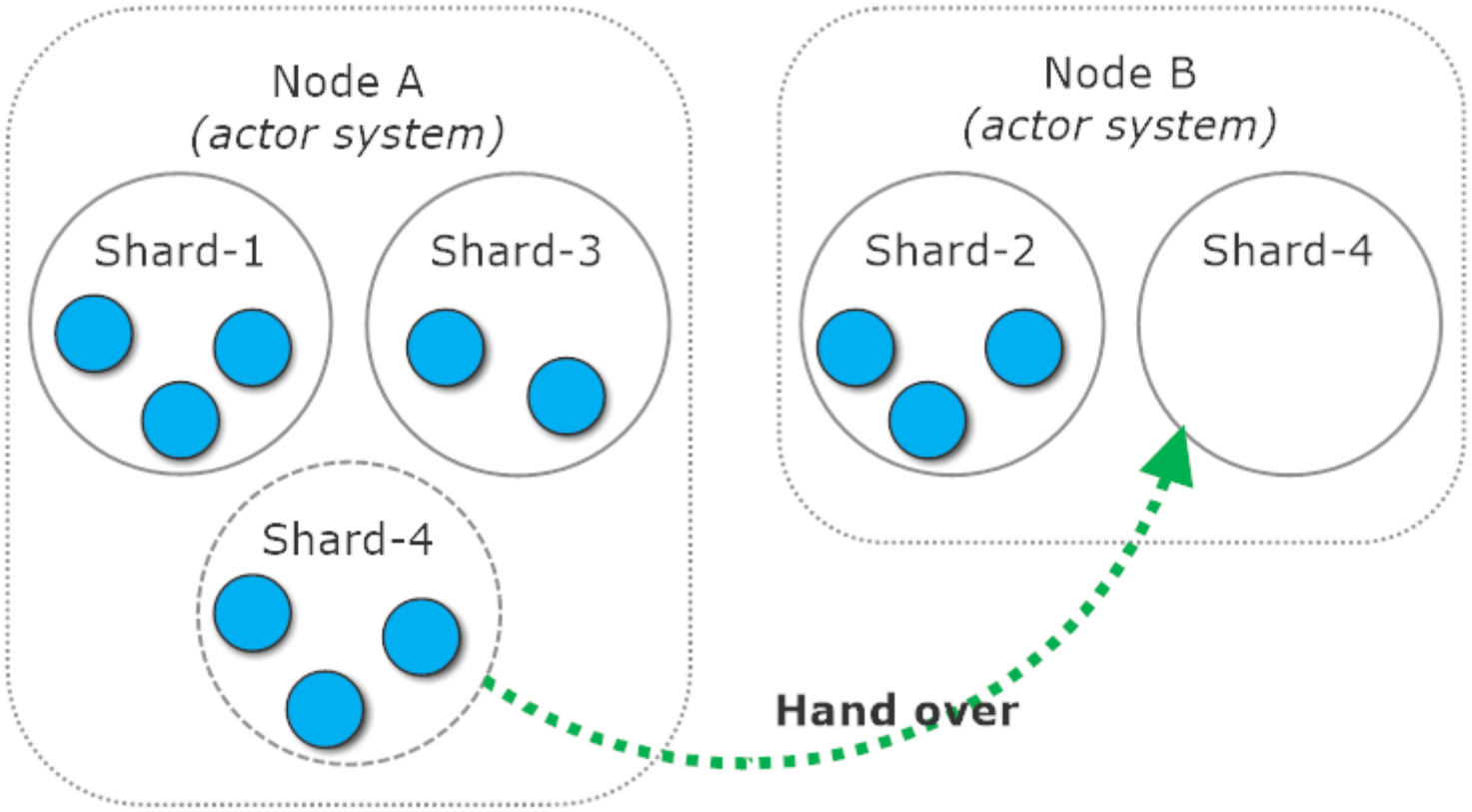
- Sorting web addresses based on reversed domain names

Sharding

- Historically most people have done sharding as part of application logic



Sharding



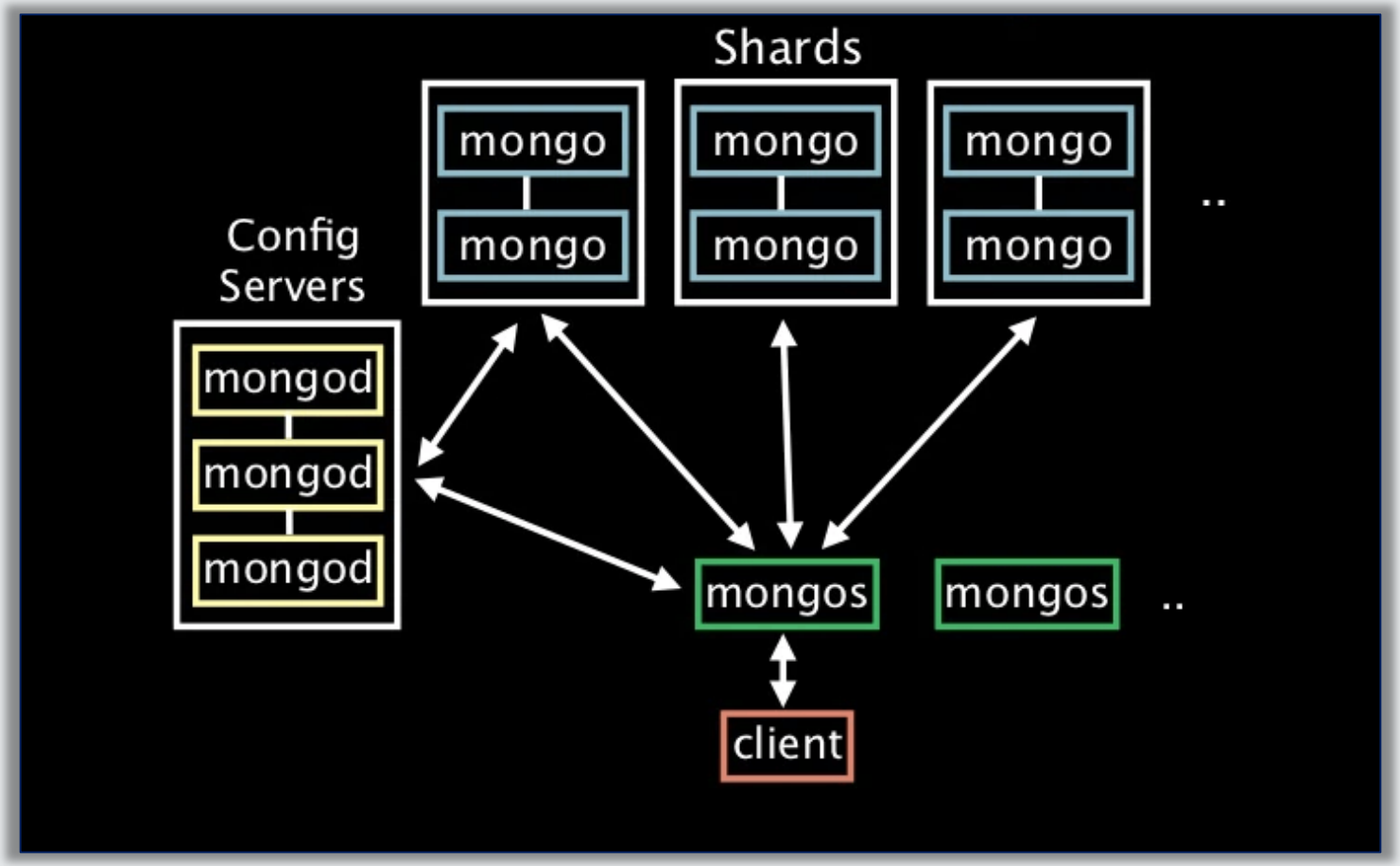
Rebalancing the Sharding

Sharding

Rebalancing the Sharding

- Changing the application code.
- Migrating the data.

Auto-Sharding



Auto-sharding Shards mongo mongo mongo

When Horizontal Sharding is Effective?

UserId	Name	Email
1	User 1	u1@nlogn.in
2	User 2	u2@nlogn.in
3	User 3	u3@nlogn.in
4	User 4	u4@nlogn.in

Original Table

UserId	Name	Email
1	User 1	u1@nlogn.in
2	User 2	u2@nlogn.in

Shard 1

UserId	Name	Email
3	User 3	u3@nlogn.in
4	User 4	u4@nlogn.in

Shard 2

Horizontal Partitioning
or
Database Sharding

When Vertical Sharding is Effective?

UserId	Name	Email
1	User 1	u1@nlogn.in
2	User 2	u2@nlogn.in
3	User 3	u3@nlogn.in
4	User 4	u4@nlogn.in

Original Table

UserId	Name
1	User 1
2	User 2
3	User 3
4	User 4

Partation 1

UserId	Email
1	u1@nlogn.in
2	u2@nlogn.in
3	u3@nlogn.in
4	u4@nlogn.in

Partation 2

Vertical Partitioning

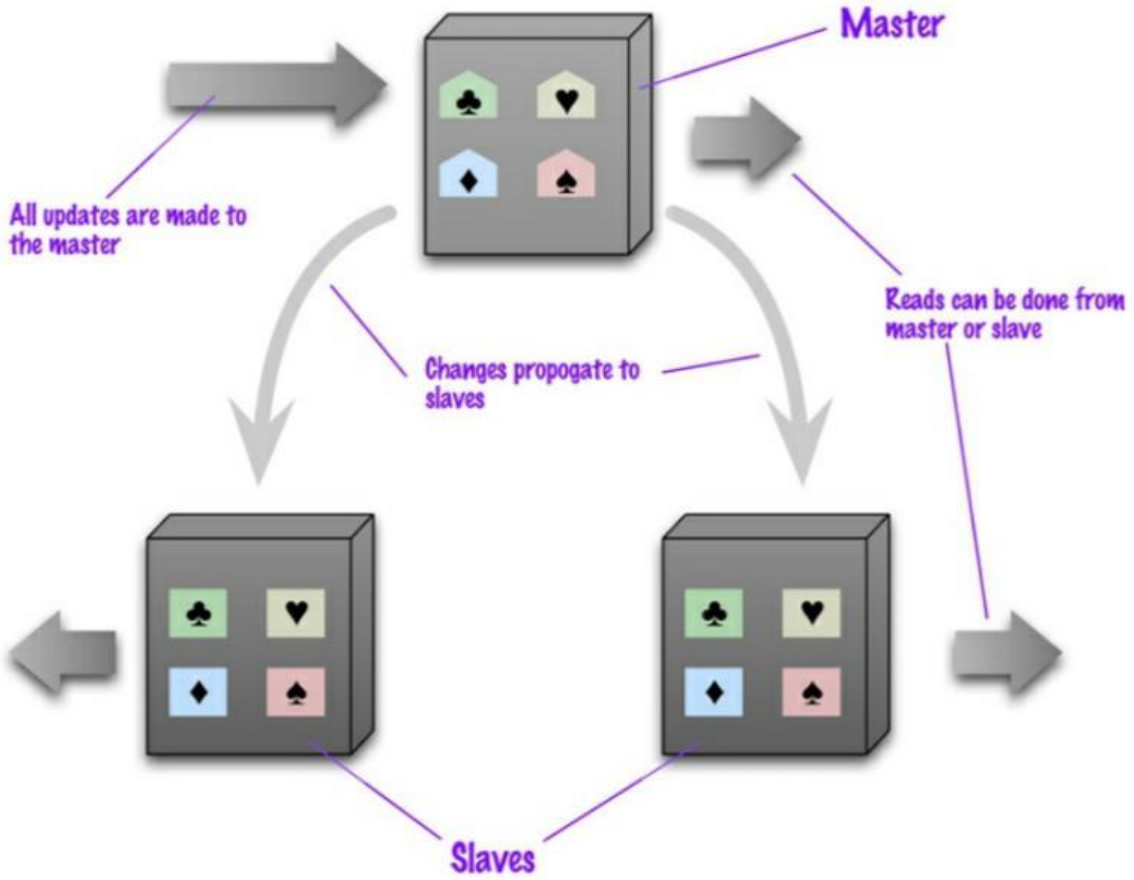
Sharding

- Sharding is particularly valuable for performance.
- Using replication, particularly with caching, can greatly improve read performance.
- Sharding provides a way to horizontally scale writes.

Sharding

- Improve resilience when used alone.
- Although the data is on different nodes, a node failure makes that shard's data unavailable just as surely as it does for a single-server solution.

Master-Slave Replication



Data is replicated from master to slaves. The master services all writes; reads may come from either master or slaves.

Master-Slave Replication

- Master-slave replication is most helpful for scaling when you have a read-intensive dataset.
- You can scale horizontally to handle more read requests by adding more slave nodes and ensuring that all read requests are routed to the slaves.

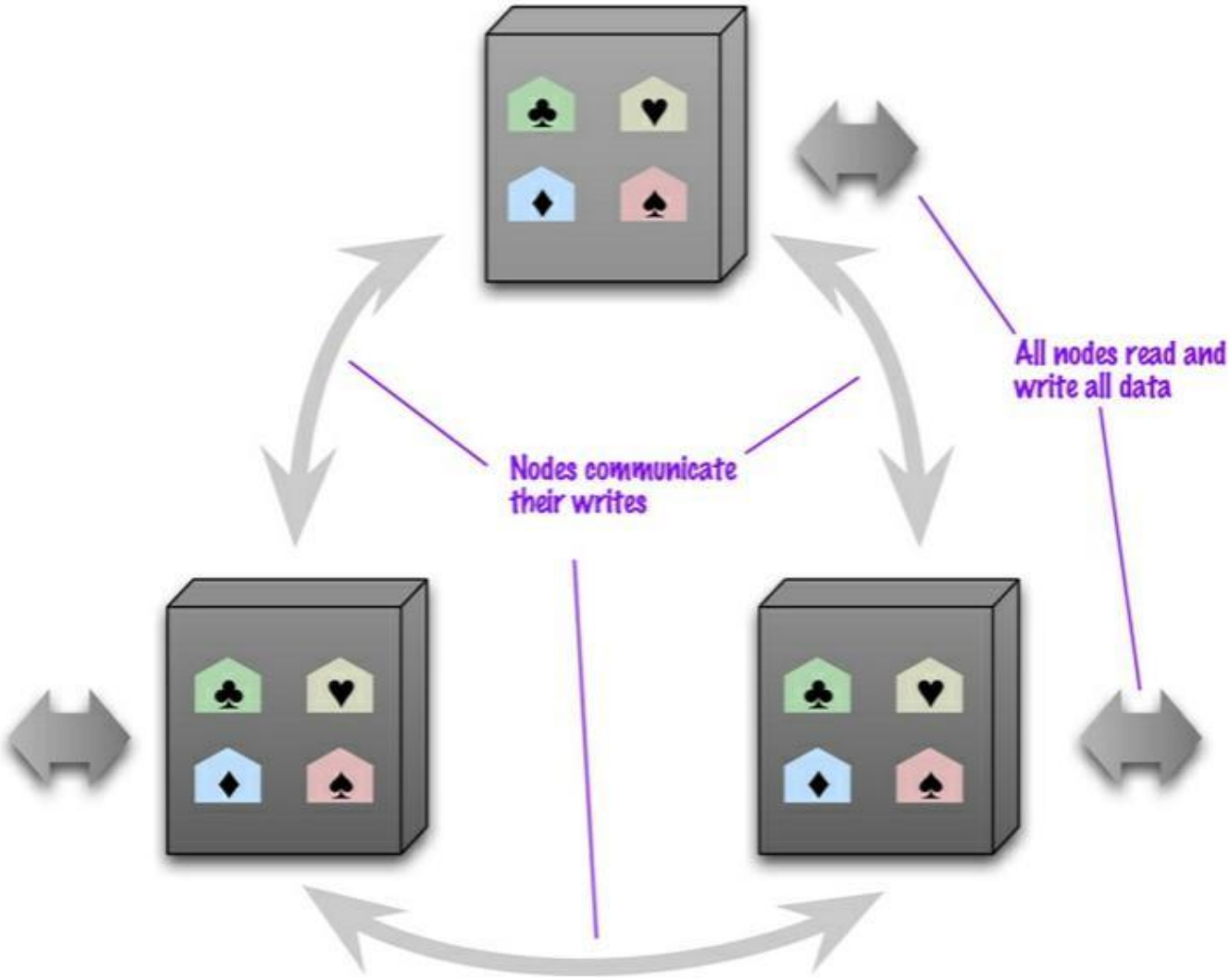
Master-Slave Replication

- You are still, however, limited by the ability of the master to process updates and its ability to pass those updates on.
- Consequently it isn't such a good scheme for datasets with heavy write traffic, although offloading the read traffic will help a bit with handling the write load.

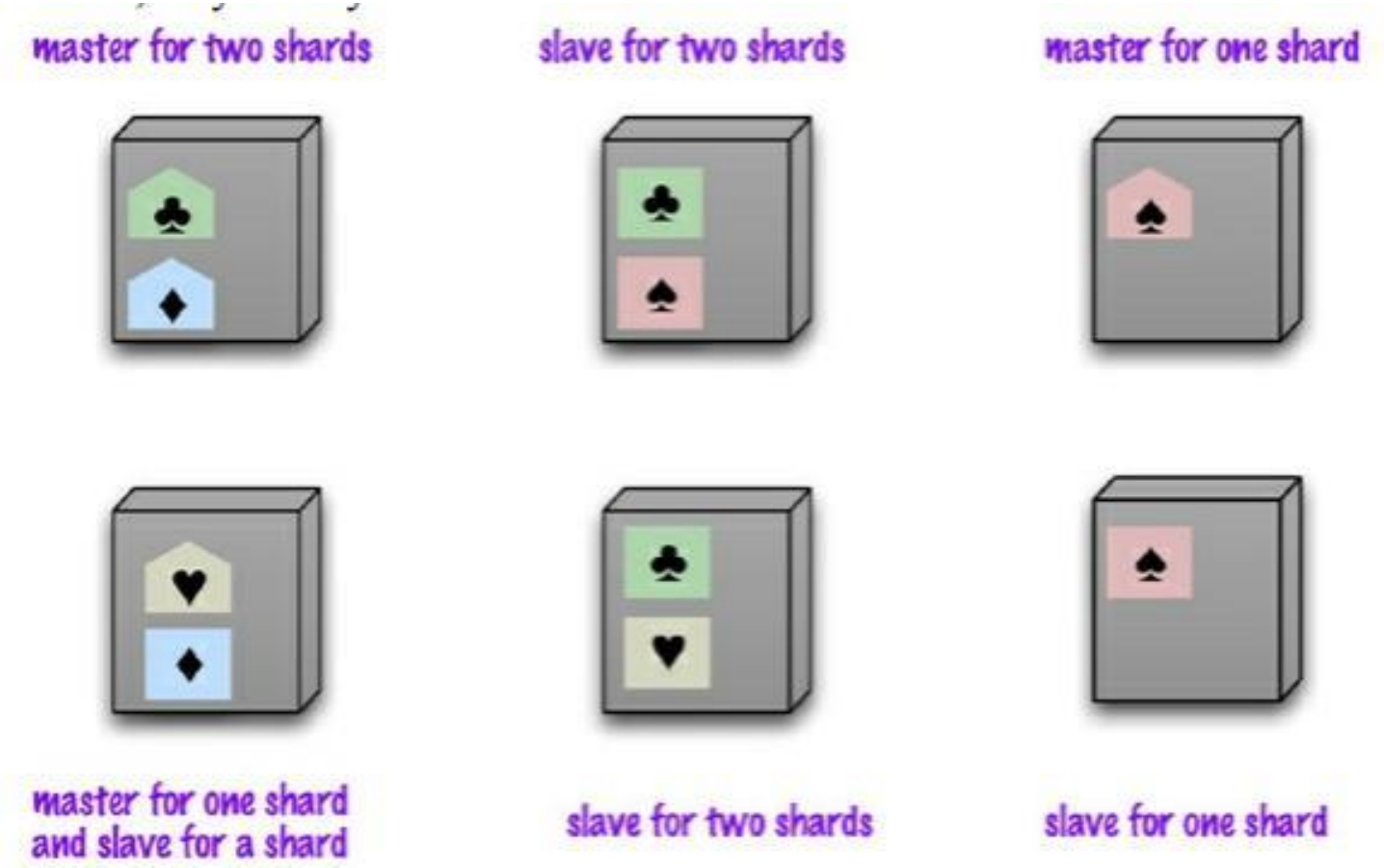
Master-Slave Replication

- A second advantage of master-slave replication is read resilience:
- The failure of the master does eliminate the ability to handle writes until either the master is restored or a new master is appointed.

Peer-to-Peer Replication

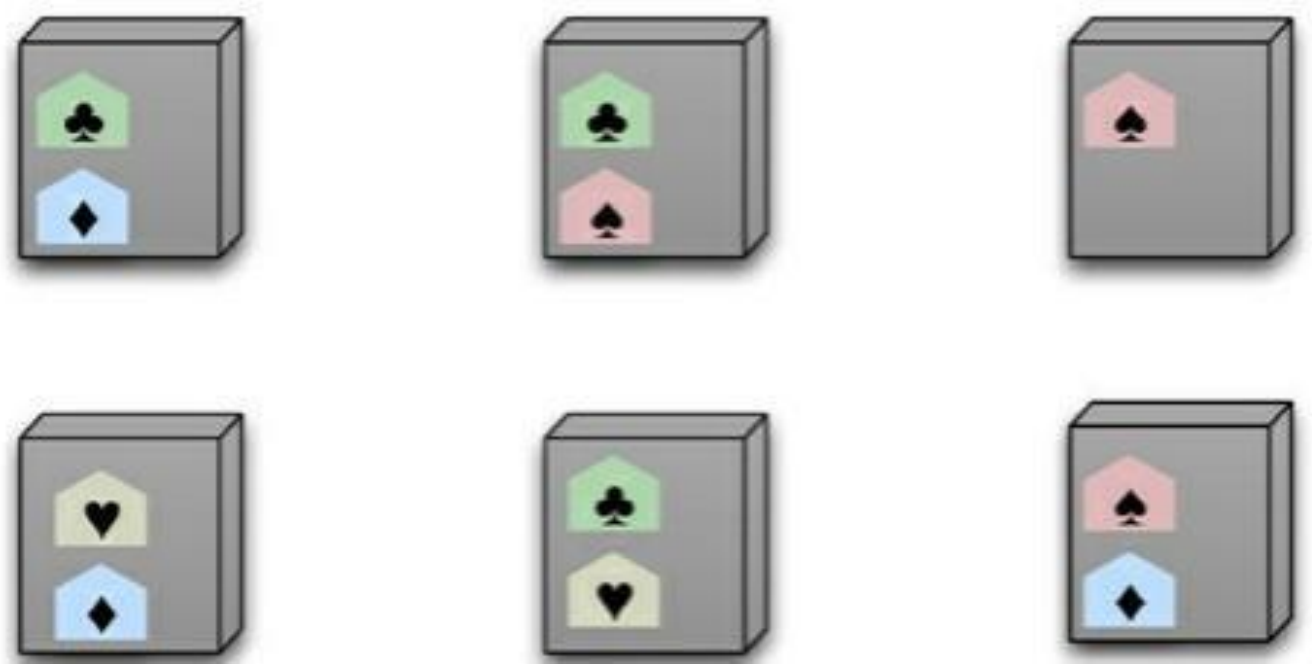


Combining Sharding and Replication



Using master-slave replication
together with sharding

Combining Sharding and Replication



Using peer-to-peer replication
together with sharding

What Is the Difference between Sharding and Partitioning?

- breaking up a large data set into smaller subsets.
- sharding implies the data is spread across multiple computers while partitioning does not.
- Partitioning is about grouping subsets of data within a single database instance.
- In many cases, the terms sharding and partitioning are even used synonymously.

What Is the Difference between Sharding and Partitioning?

- breaking up a large data set into smaller subsets.
- sharding implies the data is spread across multiple computers while partitioning does not.
- Partitioning is about grouping subsets of data within a single database instance.
- In many cases, the terms sharding and partitioning are even used synonymously.

What Is the Difference between Sharding and Partitioning?

- breaking up a large data set into smaller subsets.
- sharding implies the data is spread across multiple computers while partitioning does not.
- Partitioning is about grouping subsets of data within a single database instance.
- In many cases, the terms sharding and partitioning are even used synonymously.

What Is the Difference between Sharding and Partitioning?

- breaking up a large data set into smaller subsets.
- sharding implies the data is spread across multiple computers while partitioning does not.
- Partitioning is about grouping subsets of data within a single database instance.
- In many cases, the terms sharding and partitioning are even used synonymously.


That's all for now...