AT82.02

DATA MODELING AND MANAGEMENT

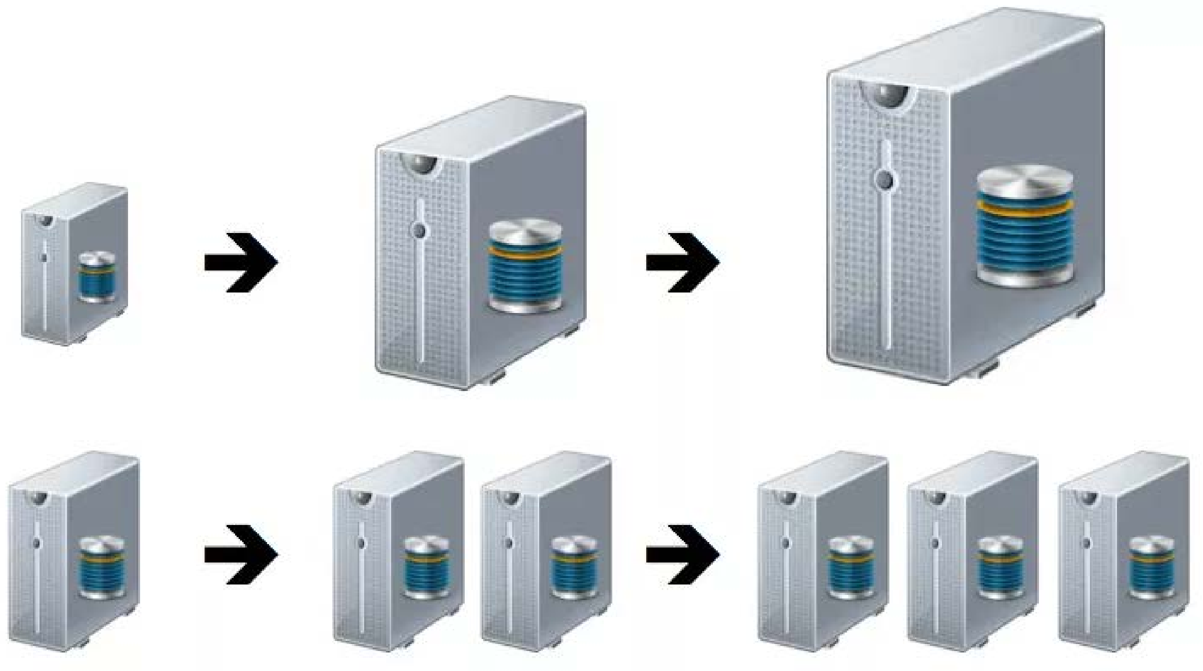
UNIT 3-1: DATA SHARDING & REPLICATION MODELS

CHUTIPORN ANUTARIYA (CHUTI AT AIT DOT AC DOT TH)

Co-funded by the Erasmus+ Programme of the European Union

RECAP

Scale UP vs. Scale OUT



Distribution Model

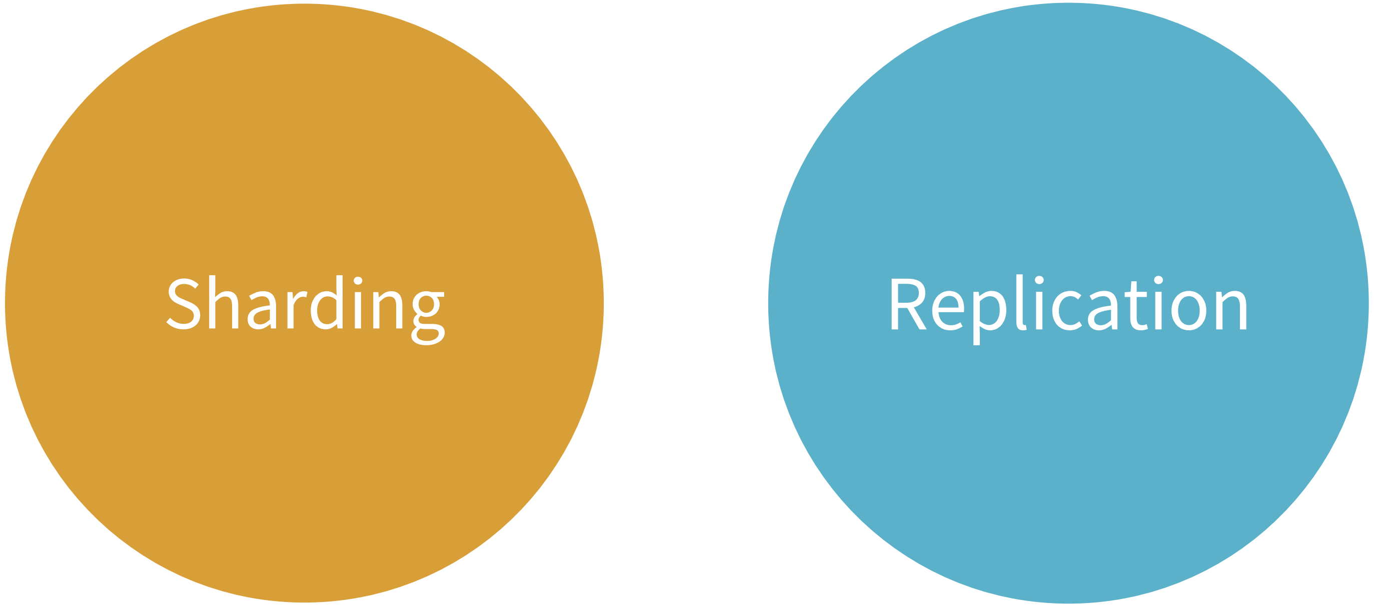
Scale out is more appealing since we can run databases on a cluster of servers.

Depending on the distribution model, we can obtain a data store that can give us the ability:

- To handle large quantity of data,
- To process a greater read or write traffic, or
- To have more availability in the case of network slowdowns of breakages

However, running over a cluster introduces complexity.

Distribution Model



Sharding

Replication

Sharding vs. Replication

Sharding distributes data across multiple nodes. So each node acts as a single source for a subset of data.

Replication copies data across multiple nodes. So each bit of data can be found in multiple places.

Distribution Models

- Single Server
- Sharding
- Master-Slave Replication
- Peer-to-Peer Replication
- Combining Sharding and Replication

Single Server

First and simplest distribution option

- no distribution at all

Run the database on a single machine that handles all the reads and writes.

This option eliminates all the complexity that other options introduce.

- Easy for operational people to manage.
- Easy for application developer to understand.

Distribution Models

Single Server

Sharding

Master-Slave Replication

Peer-to-Peer Replication

Combining Sharding and Replication

DATA SHARDING: WHAT IS IT?

Database partitioning is the process of making data splits into a database in order to distribute large amounts of data into smaller segments and can be distributed among two or more unique servers.

Each fragment can be a table, or a different physical database maintained on a separate instance of the database server.

TYPES OF SHARDING

Vertical sharding

Vertical partitioning is a method of storing data from different columns into separate fragments.

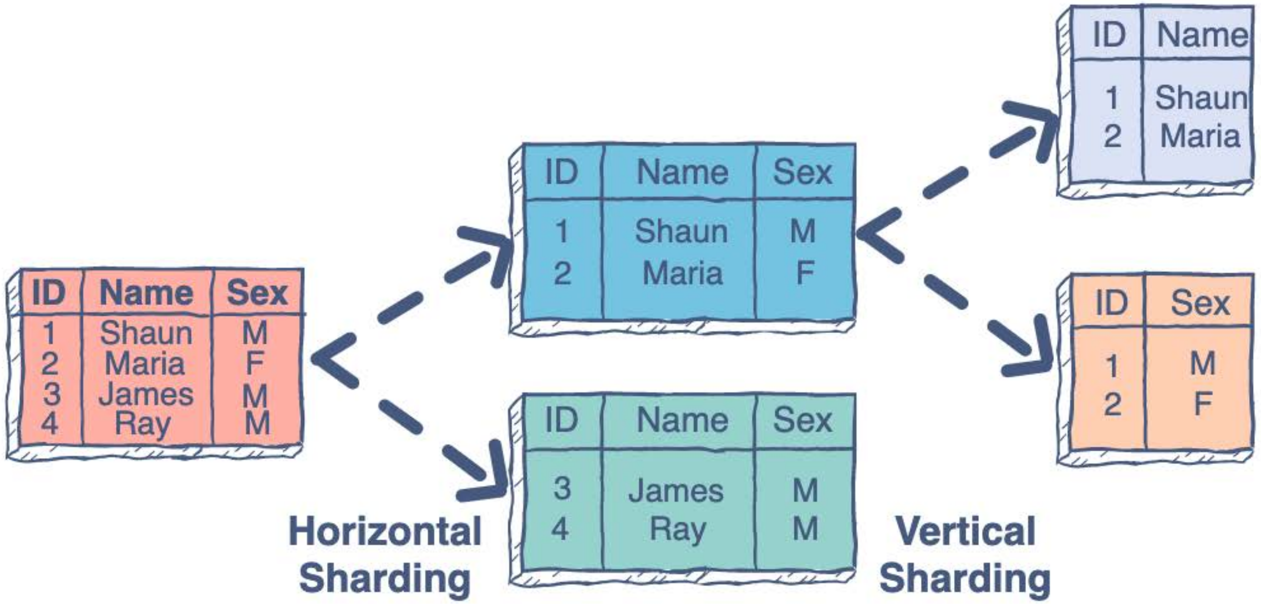
Horizontal sharding

Horizontal sharding is an arrangement that is carried out on condition that the rows of a database table are linked in a distinct way.

Domain specific sharding

The term Domain specific sharding is used when a logical division is drawn within the application data, storing it in different. Generally, this type of division is implemented at application level.

DATABASE SHARDING EXAMPLE



ID	Name	Sex
1	Shaun	M
2	Maria	F
3	James	M
4	Ray	M

ID	Name	Sex
1	Shaun	M
2	Maria	F

ID	Name	Sex
3	James	M
4	Ray	M

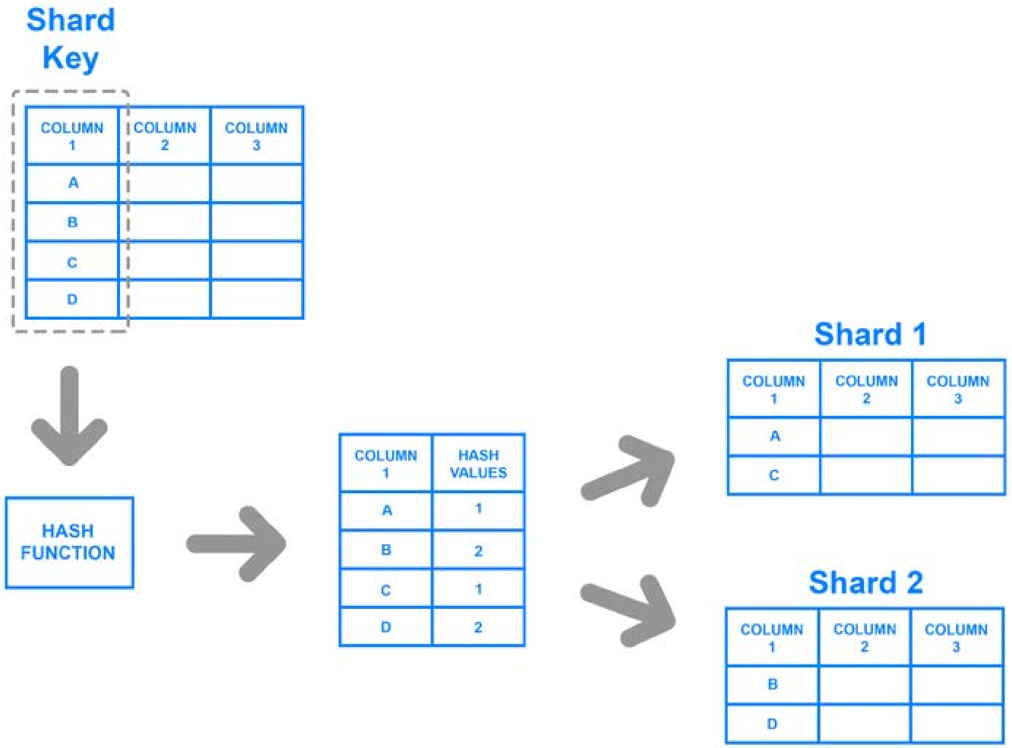
ID	Name
1	Shaun
2	Maria

ID	Sex
1	M
2	F

Some data within the database remains present in all shards (vertical sharding), but some appear only in single shards (horizontal sharding). The following figure illustrates vertical sharding and horizontal sharding.

SHARDING ARCHITECTURES

Key Based Sharding



COLUMN 1	COLUMN 2	COLUMN 3
A		
B		
C		
D		

COLUMN 1	HASH VALUES
A	1
B	2
C	1
D	2

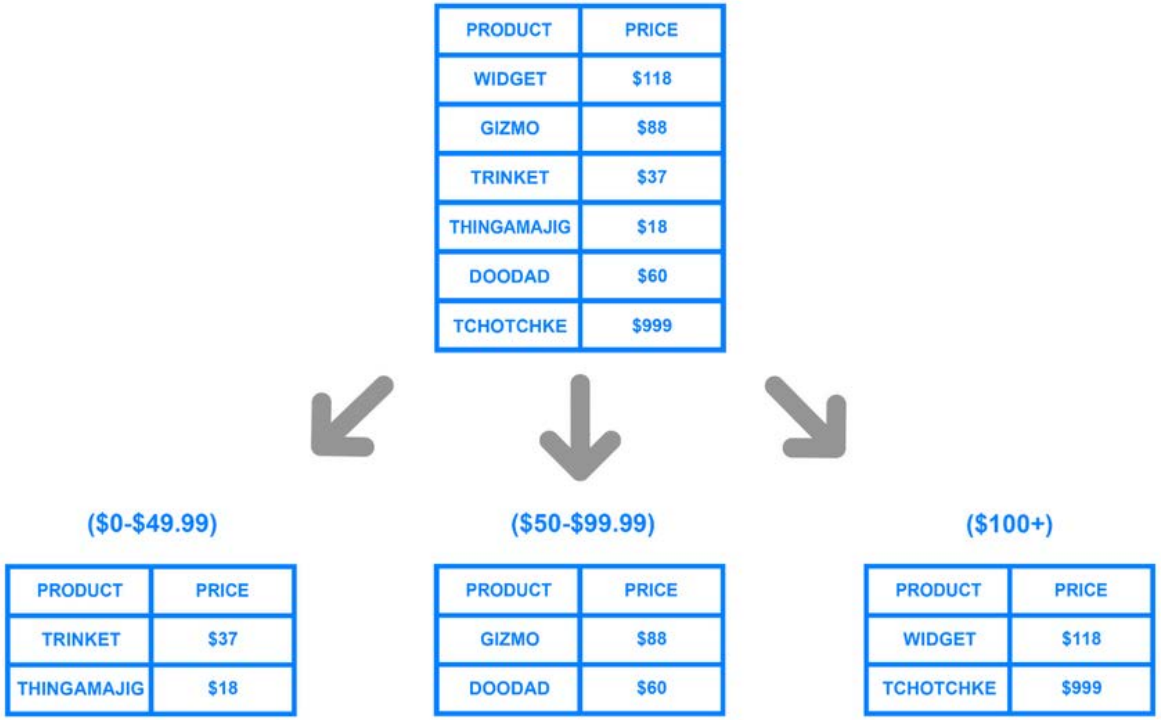
Shard 1

COLUMN 1	COLUMN 2	COLUMN 3
A		
C		

Shard 2

COLUMN 1	COLUMN 2	COLUMN 3
B		
D		

Range Based Sharding



PRODUCT	PRICE
WIDGET	$118
GIZMO	$88
TRINKET	$37
THINGAMAJIG	$18
DOODAD	$60
TCHOTCHKE	$999

($0-$49.99)

PRODUCT	PRICE
TRINKET	$37
THINGAMAJIG	$18

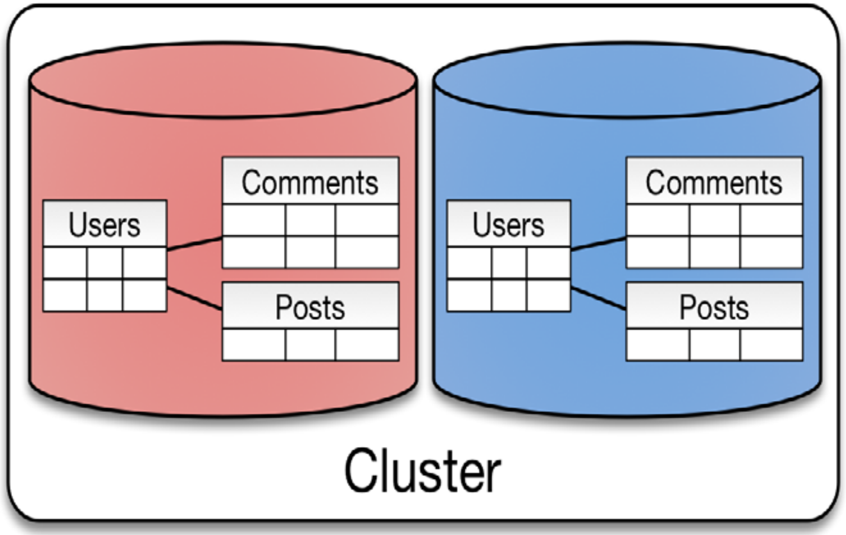
($50-$99.99)

PRODUCT	PRICE
GIZMO	$88
DOODAD	$60

($100+)

PRODUCT	PRICE
WIDGET	$118
TCHOTCHKE	$999

Sharding: Approaches



Data that is accessed together should be stored in the same node.

Queries within a single physical shard are efficient.

Stronger consistency semantics can be achieved within a shard.

If access is based on physical location, we can place data close to where it is accessed.

Another factor is trying to keep data balanced: We should arrange aggregates so they are evenly distributed in order that each node receive the same amount of the load.

Sharding: Approaches

In general, many NoSQL databases offers **auto- sharding.**

This can make much easier to use sharding in an application.

Sharding is especially valuable for performance because it **improves read and write** performances.

It scales read and writes on the different nodes of the same cluster.

Sharding: Right Time

Some databases are intended to be sharded at the beginning of development and certainly in production.

Some start with a single node, and then distribute and shard.

However, sharding very late may create trouble

- especially if done in production, where the database became unavailable during the moving of the data to the new shards.

Distribution Models

Single Server

Sharding

Master-Slave Replication

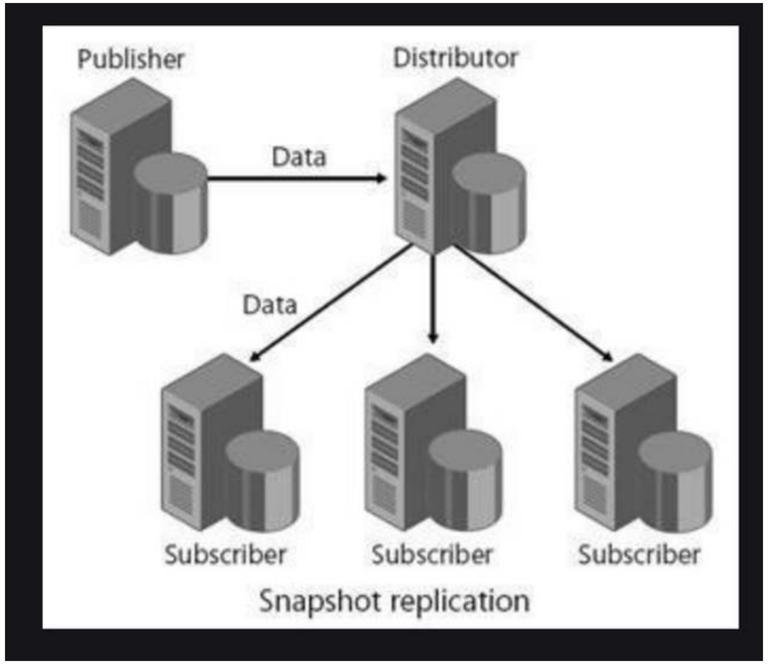
Peer-to-Peer Replication

Combining Sharding and Replication

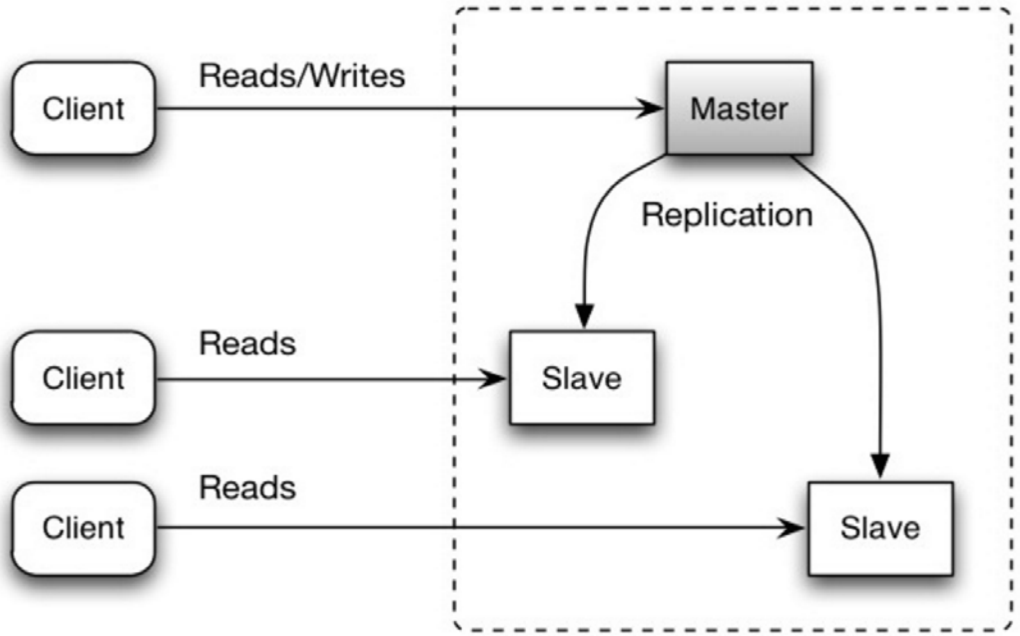
REPLICATION: WHAT IS IT?

Replication refers to a database configuration in which several copies of the same data set are housed on separate machines. The main reason for replication is redundancy.

Many of the advantages of using replication include fast recovery in the event of failure of one of the machines hosting the databases. A rapid fail-over to a secondary machine minimizes break time and keeping an active copy of the database acts as a backup to minimize data loss.



Master-Slave Replication



With Master-Slave Replication, we replicate data across multiple nodes.

Data is replicated from master to slaves.

The master service all write.

Reads may come from either master or slaves.

Analyze BENEFITS / LIMITATIONS of Master-Slave Replication

Is Master-Slave Replication more appropriate for read-intensive or write-intensive datasets?

To scale horizontally and to handle more reads, what can we do?

What-If the master fails:

- how to handle read requests?
- how about writes?

Master-Slave Replication is
read resilience.

- Slaves can handle read requests.
- Writes are not allowed until the master restored
- Recovery after a failure of a master is speeded up: a slave can be appointed as master, reducing downtime.

How about

consistency or inconsistency

issue?

Problem of inconsistency

- Clients reading different slaves may see different values, because the changes haven’t all propagated to the slaves.
- In the worst case, a client cannot read a write it just made.

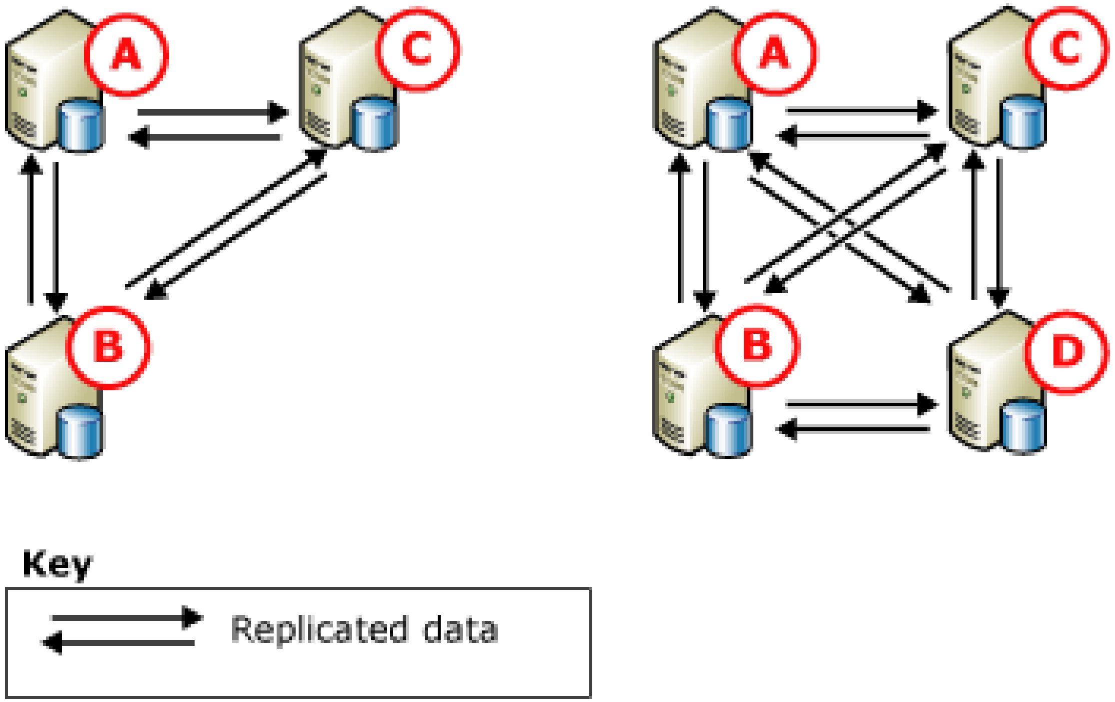
Master-Slave Replication

- Read scalability, but problem on scalability of writes.
- Resilience against failure of a slave, but not of a master.
- Master is still a single point of failure.

Distribution Models

- Single Server
- Sharding
- Master-Slave Replication
- Peer-to-Peer Replication
- Combining Sharding and Replication

Peer-to-Peer Replication



Peer-to-Peer Replication overcomes to problems of Master-Slave Replication by NOT having a master.

Analyze BENEFITS / LIMITATIONS of Peer-to-Peer Replication

Peer-to-Peer Replication

- It provides a scale-out and high-availability solution by maintaining copies of data across multiple nodes.
- We can easily add nodes for performances.

Peer-to-Peer Replication

- The biggest complication is **consistency!!**
- When we can write on different nodes, we increase the probability to have inconsistency on writes.
- Example: two clients attempt to write/update the same data at the same time: write-write conflict.

Distribution Models

Single Server

Sharding

Master-Slave Replication

Peer-to-Peer Replication

Combining Sharding and Replication

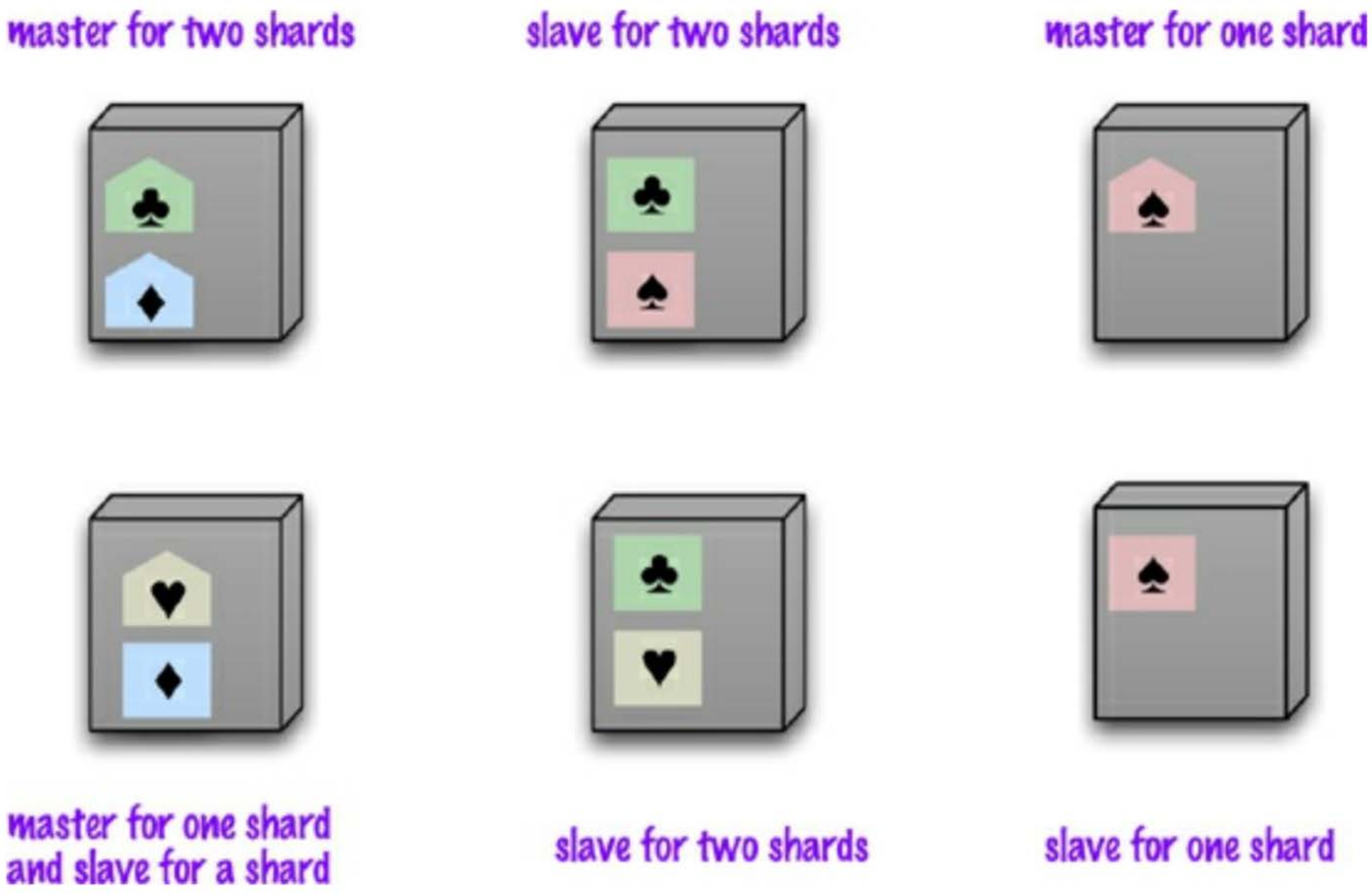
Combining Sharding and Replication

Sharding distributes data across multiple nodes. So each node acts as a single source for a subset of data.

Replication copies data across multiple nodes. So each bit of data can be found in multiple places.

We can have multiple masters, but each data item only has a single master.

Combining Master-Slave Replication and Sharding

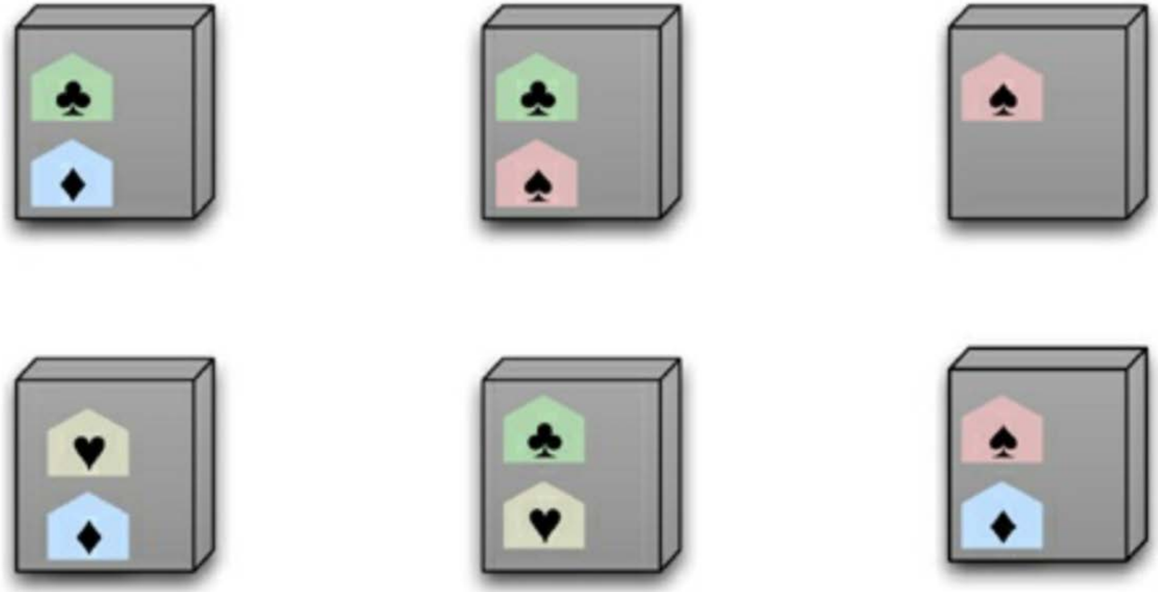


Combining Peer-to-Peer Replication and Sharding

A good starting point is to have a replication factor of 3, so each shard is present on 3 nodes.

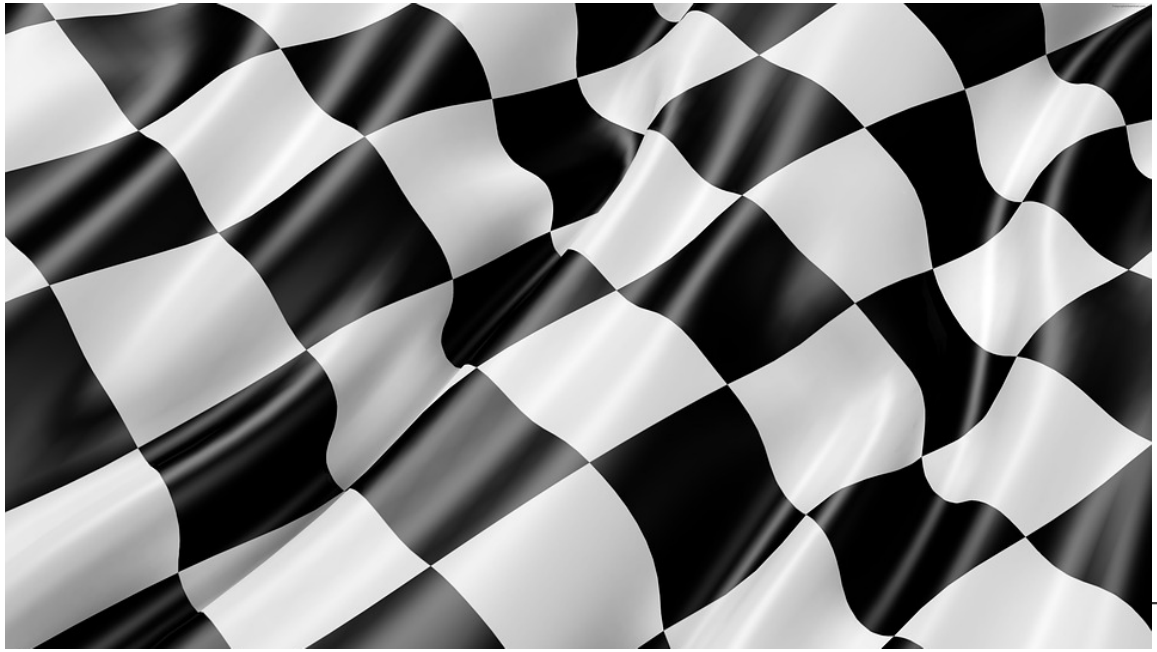
If a node fails, a shard of that node will be built on other nodes.

Combining Peer-to-Peer Replication and Sharding



the REFERENCE

- P. Sadalage and M. Fowler: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Addison-Wesley Professional, 2013



Thank you.

Let’s Summarize!