Introduction to NoSQL

- Avinash Kumar

RDBMS

- RDBMS: Relational Database Management System
- Relational Databases persist data in tabular form structured to recognize relations between stored items of information.
- DB2, MySQL, PostgreSQL, SQLite etc

e_id	e_name	e_salary	e_age	e_gender	e_dept
1	Sam	95000	45	Male	Operations
2	Bob	80000	21	Male	Support
3	Anne	125000	25	Female	Analytics
4	Julia	73000	30	Female	Analytics
5	Matt	159000	33	Male	Sales
6	Jeff	112000	27	Male	Operations

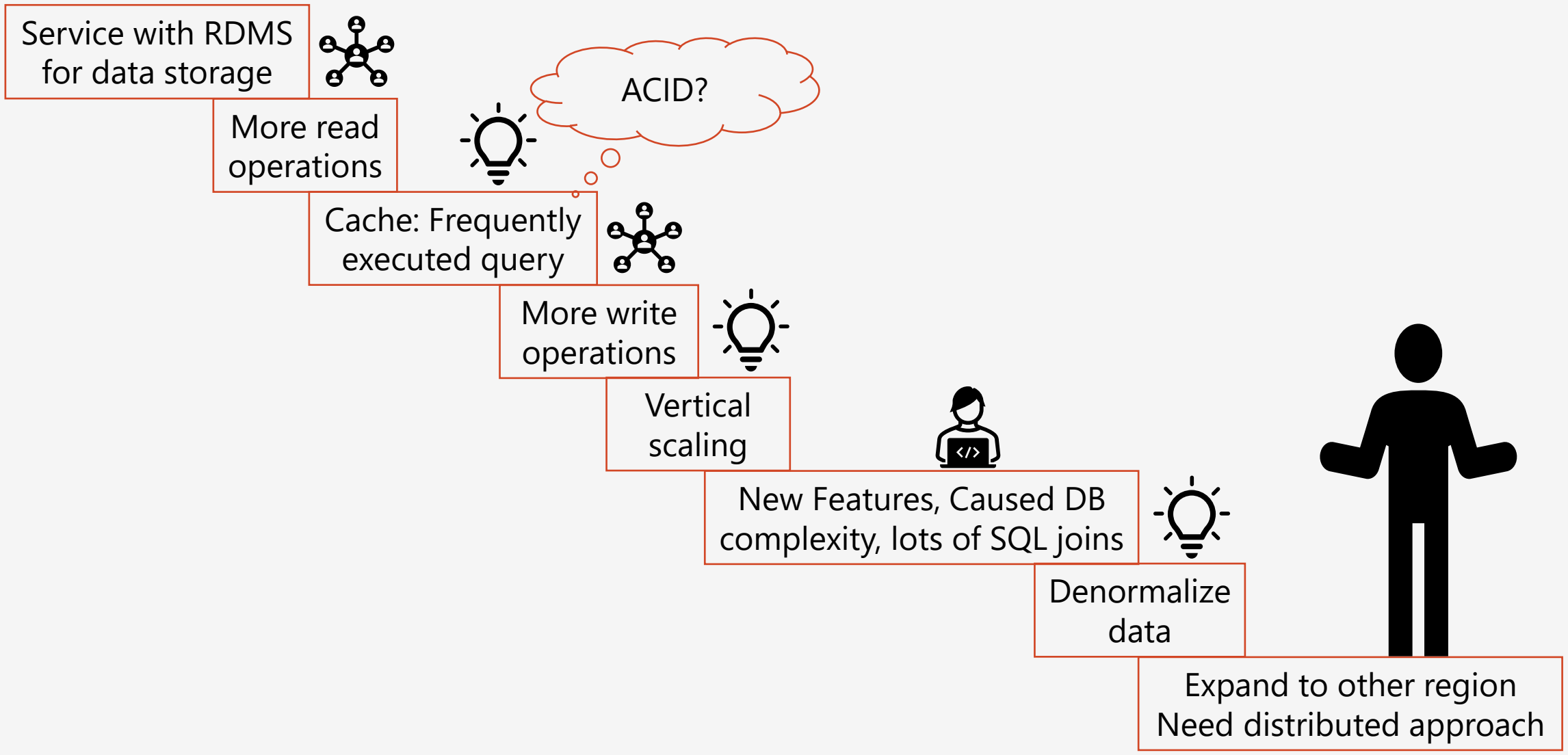
Pros:

- ACID
- SQL - Structured Query Language
- Strong consistency, concurrency and recovery.

Cons:

- Not built to be distributed.
- Joins are expensive
- Hard to scale horizontally
- Impedance mismatch

Case Analysis



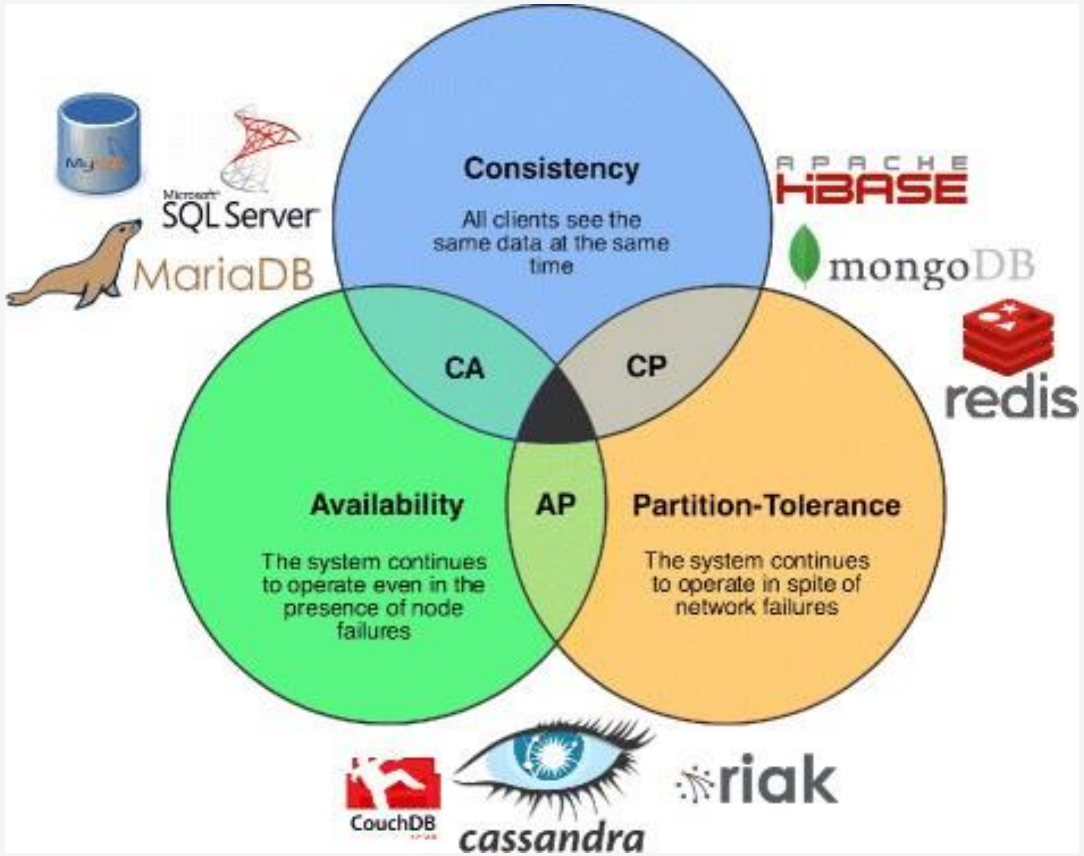
NoSQL

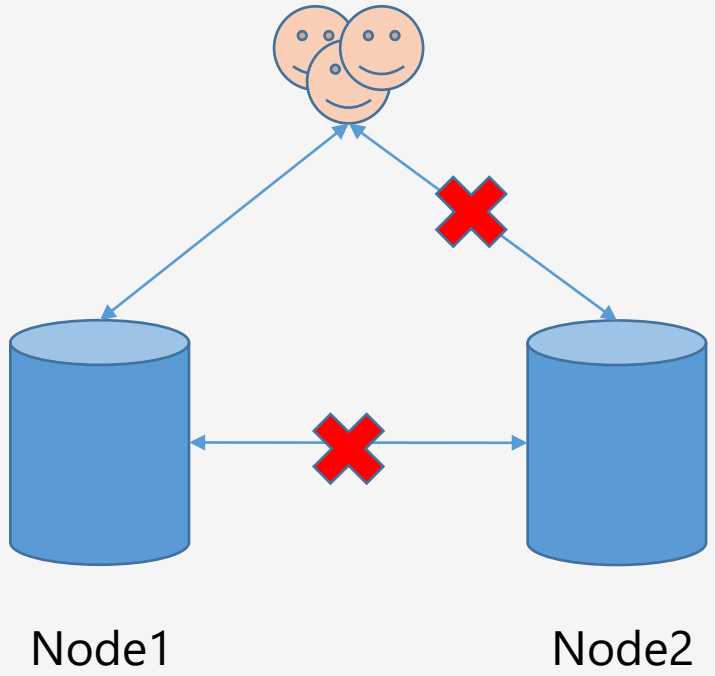
- Cluster friendly
- High availability and scalability
- Schema less, non-tabular and non-relational
- Big data capabilities
- Avoids upfront schema design
- Transactions are handled at application layer
- Provides easy and frequent changes to db
- Horizontal scaling
- Fixes impedance mismatch
- Allows faster development

Cap Theorem

According to CAP theorem a distributed system can only provide at most 2 of these three properties

- ❑ Consistency
- ❑ Availability
- ❑ Partition Tolerance





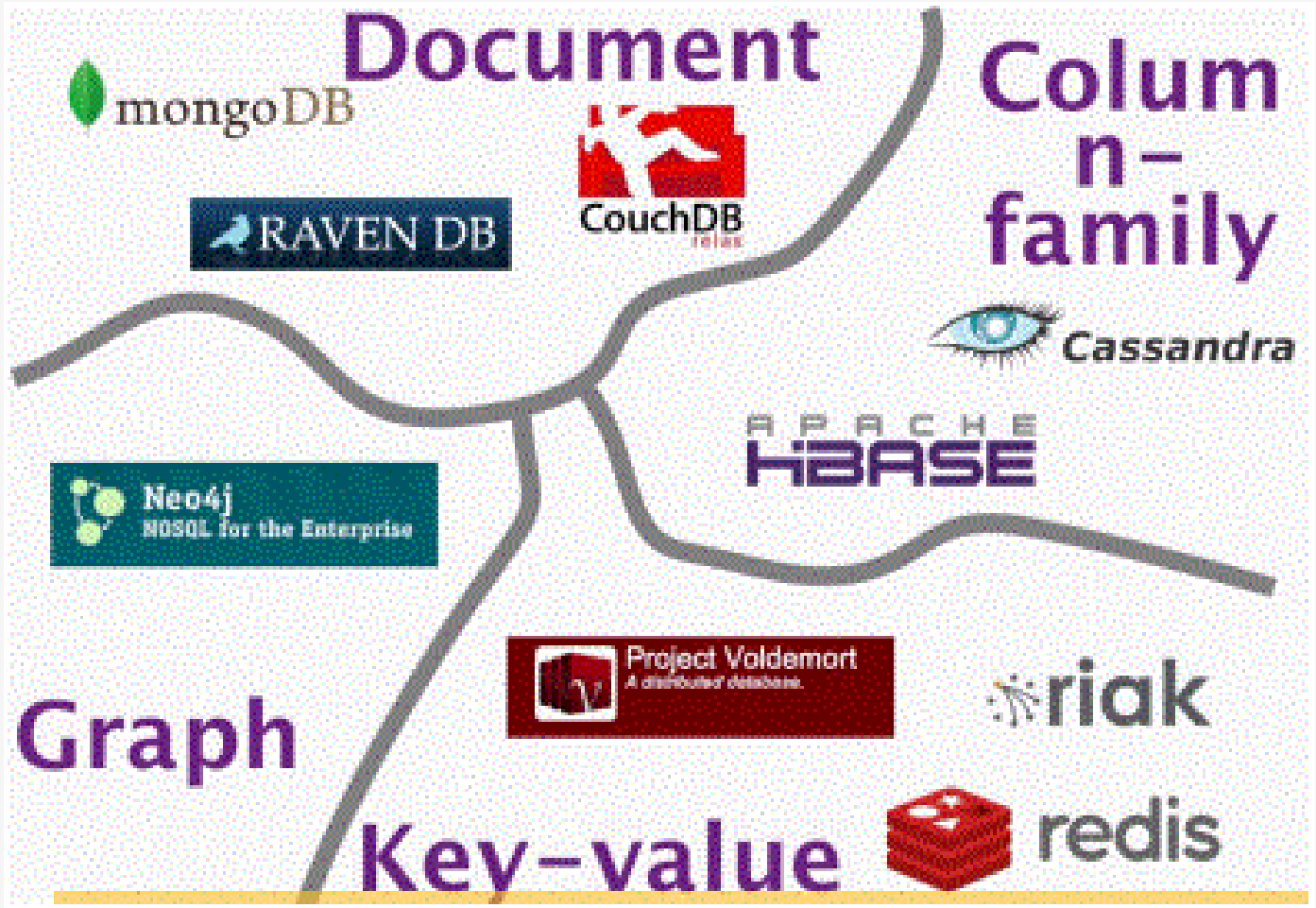
BASE

- Basically available: Nodes in a distributed environment can go down but the whole system shouldn’t be affected.
- Soft state (scalable): The state of the system and data could change over time.
- Eventually consistency: Given enough time, data will be consistent across the distributed system.

Aggregate Data Models

NoSQL databases are divided in four major data models:

- ❑ Key-Value
- ❑ Document
- ❑ Column family
- ❑ Graph



Key-Value

- Key serves as unique identifier
- Value can be anything ranging from simple objects to complex compound objects
- Redis, MemcacheDB, Riak etc.

Use cases:

- Storing user session data
- Maintaining schema-less profile
- Storing user preferences
- Storing shopping kart data

Key	Value
101	Simple string value

102	{
	“customerId”: 21
	“orderItems”: [
	{
	“productId”: 201
	“quantity”: 2
	“cost”: 420
	},
	{
	“productId”: 201
	“quantity”: 2
	“cost”: 420
	}
	]
	}

Document

- Similar to Key-value
- Difference is that, the value contains structure or semi-structured data
- Value is called document and can be in JSON, XML, BSON formats
- CouchDB, MongoDB etc.

Use cases:

- Ecommerce platforms
- Content management systems
- Analytics platform
- Blogging platform

airport_1254


```
{
  "airportname": "Calais Dunkerque",
  "city": "Calais",
  "country": "France",
  "faa": "CQF",
  "geo": {
    "alt": 12,
    "lat": 50.962097,
    "lon": 1.954764
  },
  "icao": "LFAC",
  "id": 1254,
  "type": "airport",
  "tz": "Europe/Paris"
}
```

Column family

- Data is stored in cells grouped in columns of data
- Columns are logically grouped into column families
- Column families are groups of similar data that is usually accessed together
- Cassandra, Hbase etc.

Use cases:

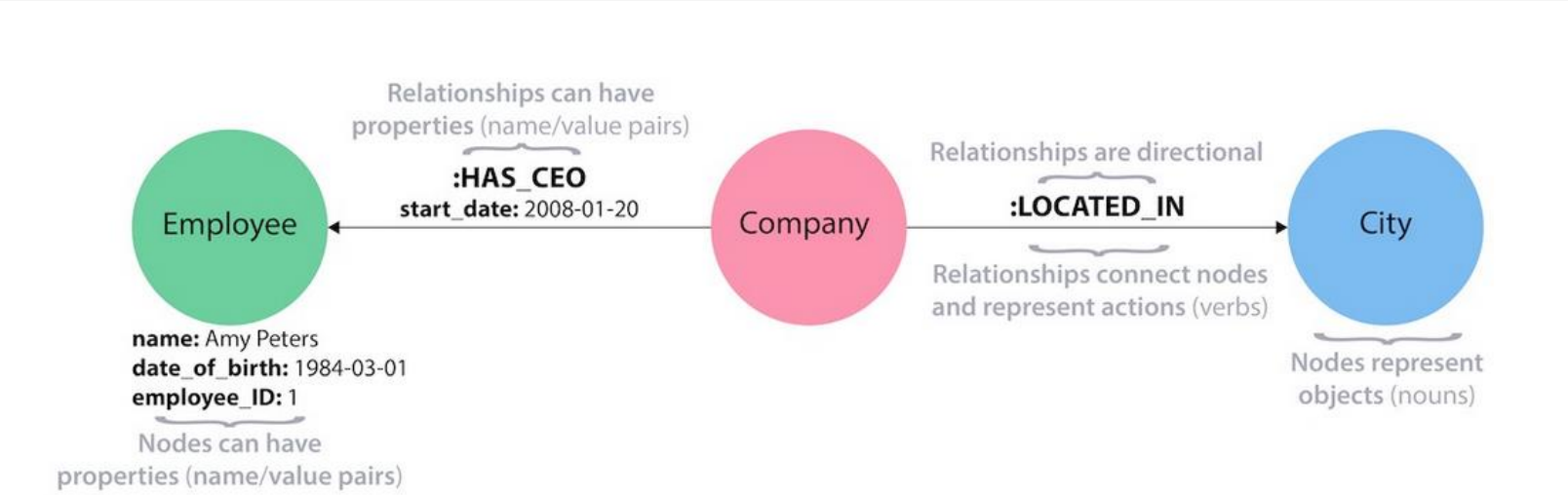
- Content management systems
- Systems that maintains counters
- Services having expiry usage
- Systems requiring heavy write requests
- Log aggregators

Graph

- Store entities and relationship as node and edge of the graph
- Entity contains the data and Relationship is how entity are interlinked
- Relationship can be defined on-the-fly
- Neo4j, OrientDB etc.

Use cases:

- Social network application
- Location based services
- Network and IT operations
- Fraud detection



Replication

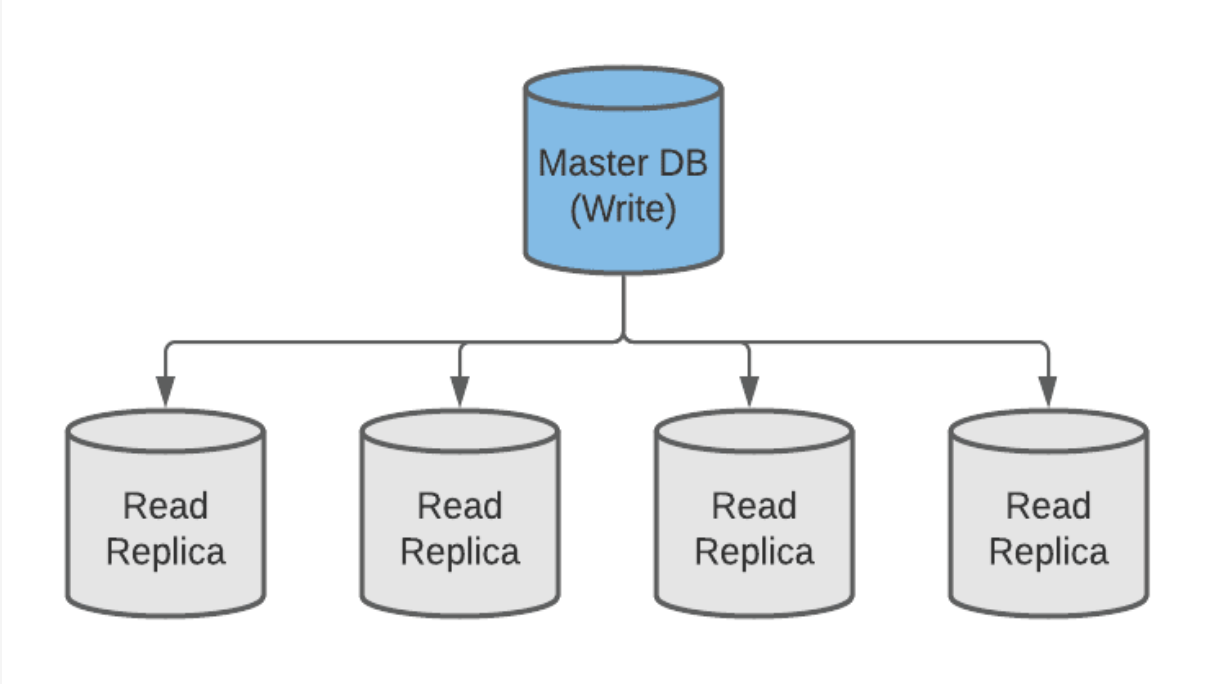
Replication copies data across multiple servers, so each bit of data can be found in multiple places.

Synchronous:

- ❑ Perform write
- ❑ Data got replicated to all nodes
- ❑ Gets back success acknowledgement

Asynchronous:

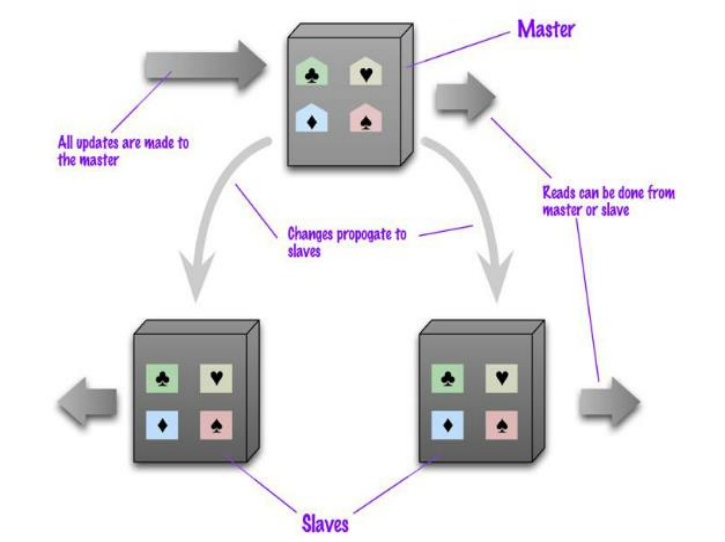
- ❑ Perform write
- ❑ Gets back success acknowledgement
- ❑ Data got replicated to all nodes in background



Type of replication

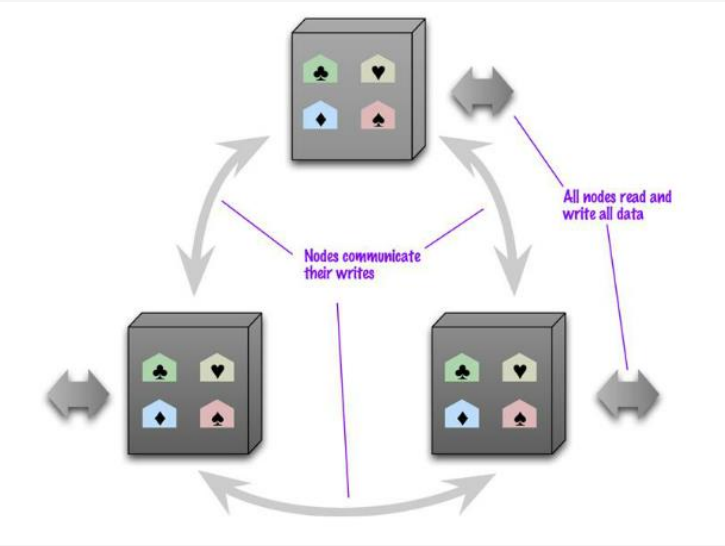
Master-Slave:

- ❑ One node is in-charge
- ❑ Write happens on master and propagates to slave nodes where read happens
- ❑ Good for read-intensive services
- ❑ Single point of failure



Peer-to-Peer:

- ❑ All nodes are equal
- ❑ Write-read can happen on any node
- ❑ Can easily withstand node failure
- ❑ Write conflicts can be the issue



Sharding

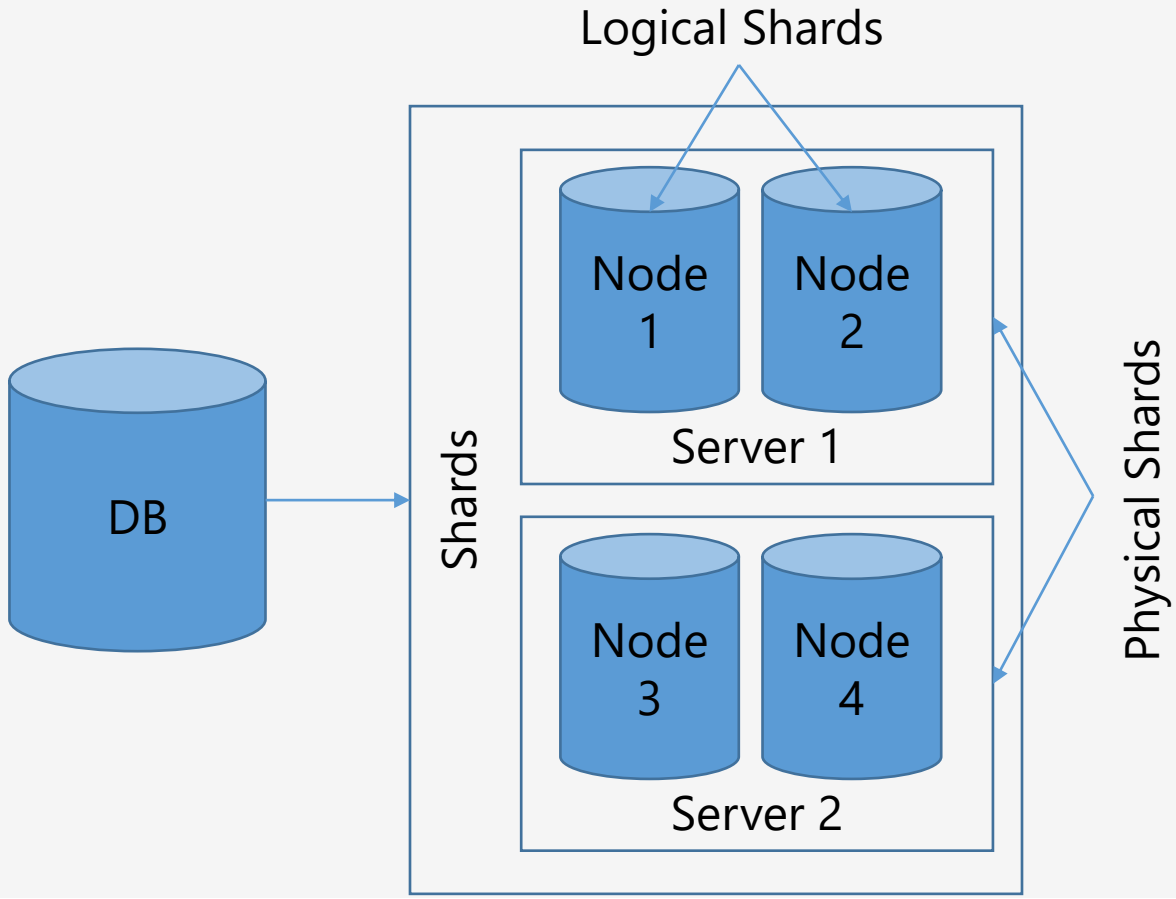
Sharding distributes different data across multiple servers, so each server acts as the single source for a subset of data.

Advantages:

- ❑ Increased Read/Write Throughput
- ❑ Increased storage capacity
- ❑ High availability

Disadvantages:

- ❑ Query overhead
- ❑ Poor shard key causes hotspots
- ❑ Increased Infrastructure Costs



Sharding Techniques

Algorithmic sharding:

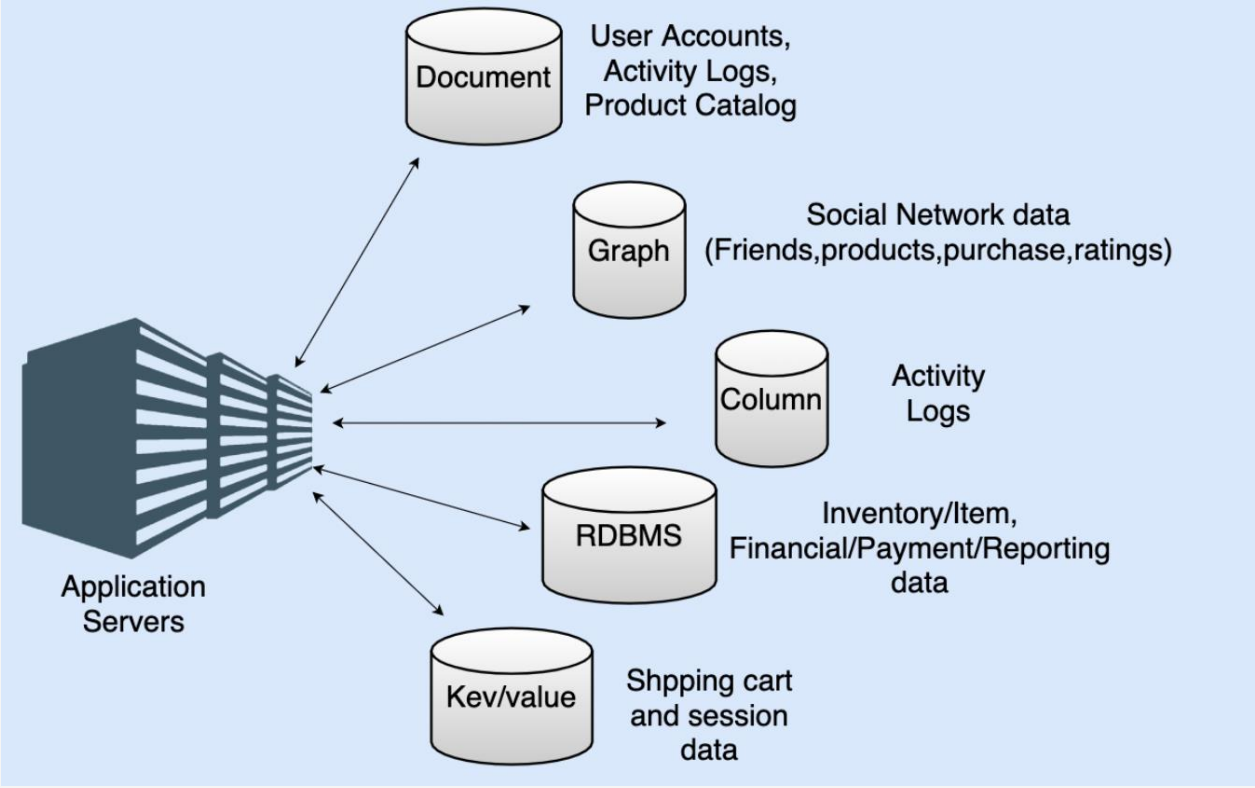
- Algorithm takes data and decides the shard it belongs
- e.g. Hash Value=ID % Number of Shards
- Increasing no of shards will require rebalance

Dynamic sharding:

- Lookup service is maintained for the data and shard mapping
- e.g. Geography based sharding
- Lookup tables should be chosen wisely so that it doesn’t grow very large
- Lookup service can be single point failure or bottleneck

Polyglot persistence

- Modern database solution for distributed systems are a combination of **SQL** and **NoSQL** solutions
- Modern applications leverage the ACID properties of RDBMS where consistency is key
- NoSQL is used for high availability data that gets the most hits
- A single application can interact with both SQL and NoSQL databases and also different kinds of NoSQL databases at a time.



Questions?

Thank you!