NoSql

NoSql

- Flexible schema
  - Does not use a structured query language.
    - In RDBMs normalized models.
  - Easy to migrate.
  - Suitable for semi-structured, complex, nested data.
- Typically do not support transactions.
  - Relax some ACID properties to ensure scalability.
- High performance.
- Open Source/specific API.

NoSql Key-value databases

- Key-value databases
  - Store/update/retrieve record with an associate key.

    →Put(key, value)
    →Get(Key)

- Examples
  - Bigtable, Apache HBase, Dynamo, Cassandra, MongoDB, Azure etc.

- Document stores (MongoDB)
  - data follow a specific data representation, example JSON format.
  - Execute simple queries based on stored values.

Partitioning/sharding

- Key-value databases
  - Records are partitioned among a cluster,
    each nodes performs lookups and updates on a subset of records.

- Challenges: manage request that must access data from multiple shards
  → replicas in order to ensure availability in case of failure,
  → keep replicas consistent,
  → expensive joins if tables are stores on different nodes,
  depends on the speed of the communication network.

Sharding

- Types of partitioning: horizontal partitioning (example sharding), vertical partitioning

- Partition is done on attributes refereed as

  partitioning key or shard keys

  → range partitioning divide data into ranges based on the key value

  → hash partitioning even data distribution but range-queries target more shards

Sharding in MongoDb

- Chunk: lower and upper range based on the shared key.

- Architecture:
  - Mongos: query routers
  - Config Servers
  - Shards (replicas)

- If queries do not include the shard, mongos performs a broadcast operation.

CAP theorem

- No distributed database can guarantee more than two of the following:

- Consistency: read the most recent write or an error,
  (linearizable consistency) once an operation is complete, it is visible to all nodes.
  eventual consistency
- Availability: every request receives a non-error response
- Partition tolerance: system operates despite arbitrary number of messages being lost

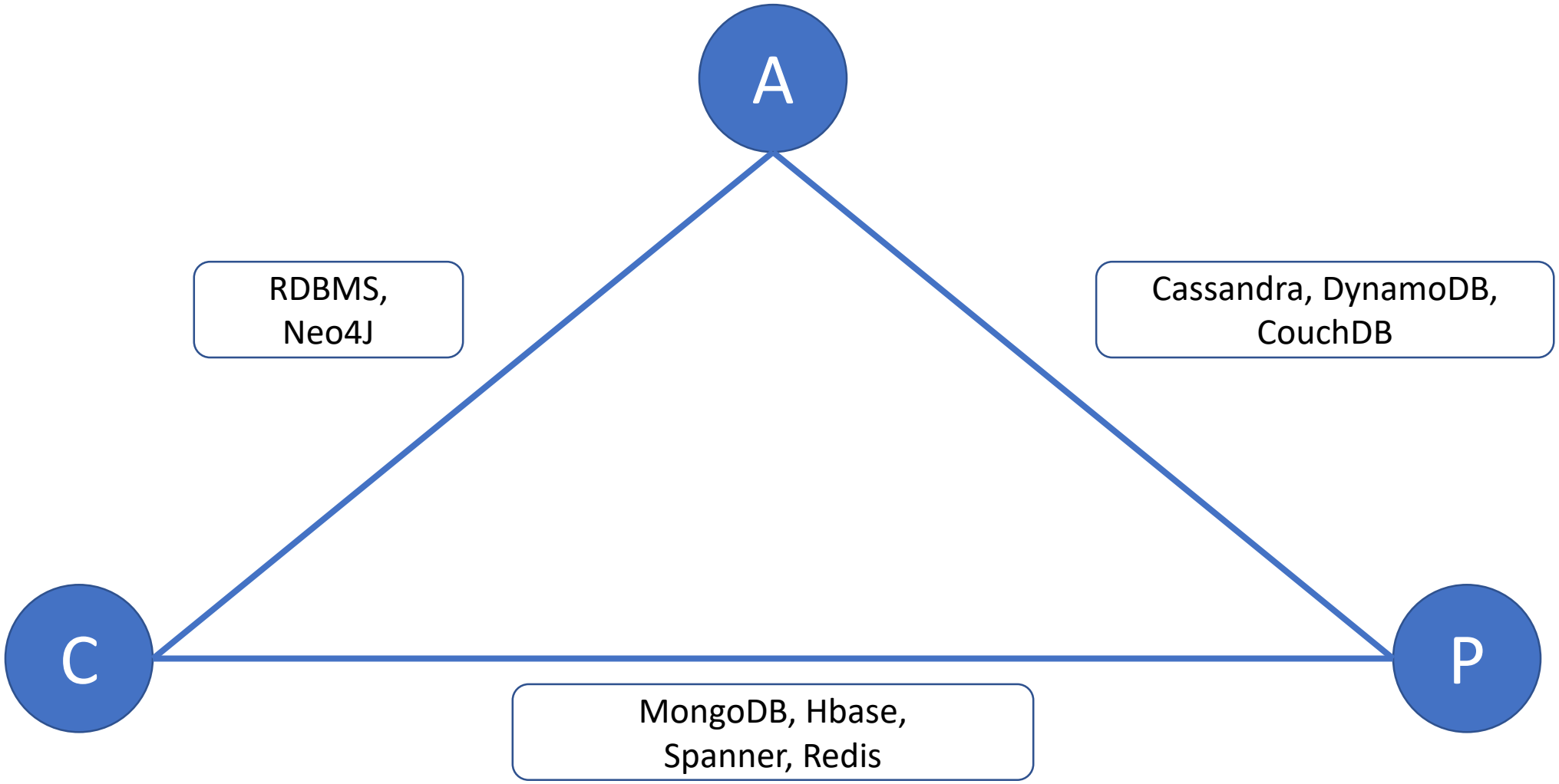
- No distributed database can guarantee more than two of the following:

- Consistency: read the most recent write or an error,
- Availability: every request receives a non-error response
  non-failing nodes receiving requests returns a response
  high availability
- Partition tolerance: system operates despite arbitrary number of messages being lost

- No distributed database can guarantee more than two of the following:

- CP: sacrifice availability, consistency and partition tolerance
- AP: sacrifice consistency, availability and partition tolerance
- CA: sacrifice partition tolerance, consistency and availability

- Alternative: PACELC


A
RDBMS,
Neo4J
Cassandra, DynamoDB,
CouchDB
C
P
MongoDB, Hbase,
Spanner, Redis

CAP Theorem

- MongoDB **CP** datastore.

- Each replica set one primary nodes receives write operations.
- Secondary nodes replicate primary node’s operations.

- If case of failure of the primary node, a secondary node replace it (node with the most recent log).
- The cluster becomes available only when all the secondary nodes replicate the primary node.

CAP Theorem

- Cassandra **CP** datastore.

- **Eventually consistent**: it's not guarantee that all replicas have the same data.
- Consistency level: number of replicas that needs to respond to a read/write operation.
  - ONE: closest replica
  - QUORUM: synchronize → majority,

Consistency levels

- **Strict consistency:** global clock, all reads seen instantaneously by all processors.
- **Sequential consistency:** global order on write operations.
- **Atomic consistency or linearizability:** global order on operations that do not overlap in time.
- **Casual consistency:** global order on related write operations.
- **Eventually consistent**: if there are no writes for a period of time that is system dependent, every node will “see” the value of the last write.

BASE

BASE

- Basically Available: low latency, high availability
- Soft state: nodes are updated without any input.
- Eventually consistent

Mongo DB

Mongo DB and SQL

Mongo	RDBMS
Document: set of key-value pairs, similar to JSON objects	row in a table
Collection: set of documents, documents in a collection may have different sets of fields	table
Field in JSON document	column
$lookup and embedded documents	joins
…	
https://docs.mongodb.com/manual/reference/sql-comparison/	

Mongo API

Use/create/delete database	
show dbs	show available databases
use database_identifier	create database/switch to database
db.dropDatabase()	drop selected database
Use/create/delete collection	
db.createCollection(id_collection)	
show collections	
db.createCollection("cappedCollection", {capped:true, size: 10000, max:3})	fixed size collection, replace oldest record
db.cappedCollection.drop()	drop collection
https://docs.mongodb.com/manual/core/databases-and-collections/	

Mongo Keys and indexes

Mongo keys and indexes

- Mongo automatically creates a key for the inserted objects.
  - _id attribute
  - Index on _id is created by default, structure:
    - a 4-byte *timestamp value*
    - a 5-byte *random value*
    - a 3-byte *incrementing counter*, initialized to a random value
- Single field index
- Compound index
- Multi key index
- Geospatial index
- Text index
- Hashed index