# NoSQL

SCS 2209 | Database II

## Hello!

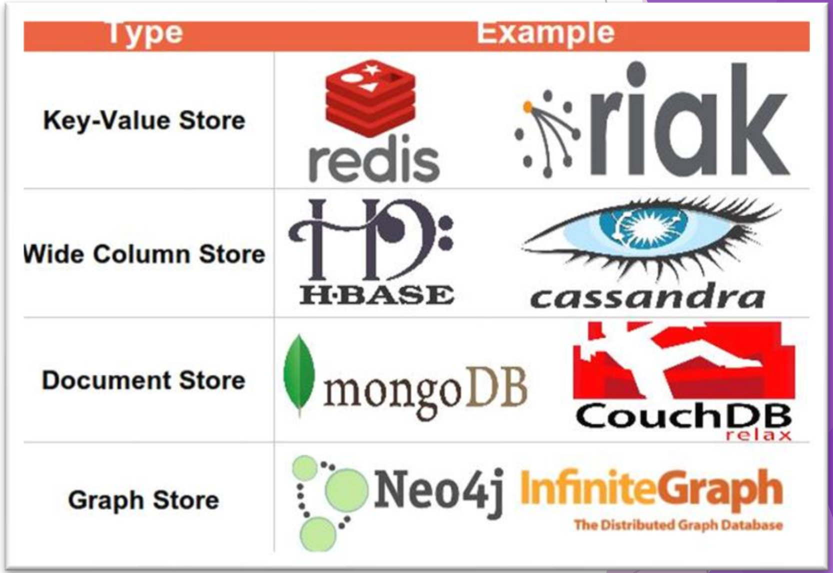
## I am Ravindu Sachintha

Senior Software Engineer,
CodeGen International

You can reach me at:
ravindusachintha53@gmail.com

## Quick Recap

- RDBMS
  - Characteristics
  - ACID properties
- NoSQL
  - Why need?
  - Big data
  - Path to today
  - Distinguishing Characteristics
  - DB Types
  - Dealing with scalability



| Type | Example | |
|---|---|---|
| Key-Value Store | redis | riak |
| Wide Column Store | HBASE | cassandra |
| Document Store | mongoDB | CouchDB relax |
| Graph Store | Neo4j | InfiniteGraph The Distributed Graph Database |

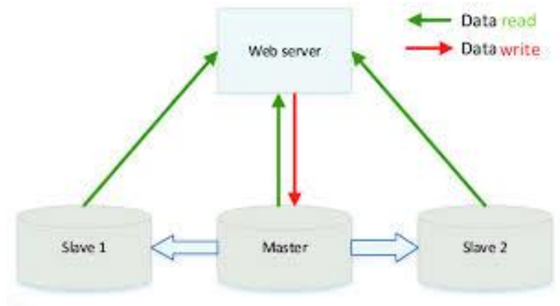
## 1.

## NOSQL in Depth

Scaling, Performance and Availability

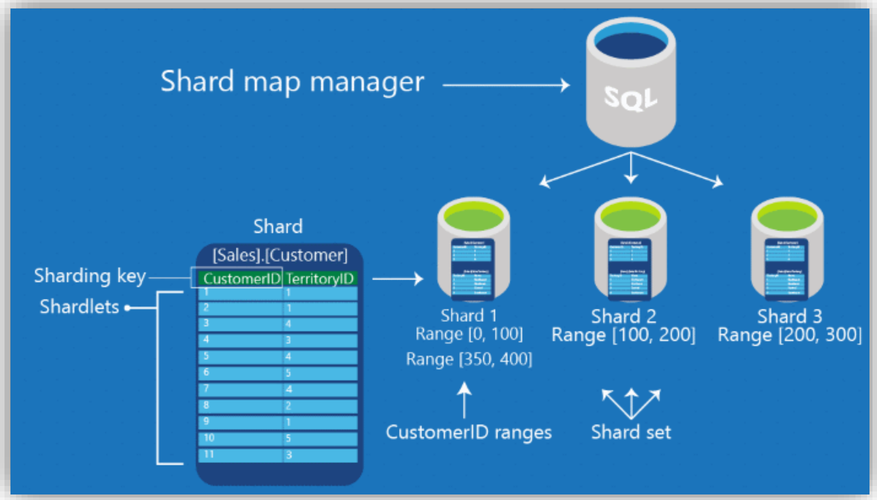
## Scaling

- Master-Slave
  - All writes are written to the master. All reads performed against the **replicated** slave databases
  - Critical reads may be incorrect as writes may not have been propagated down
  - Large data sets can pose problems as the master needs to duplicate data



## Scaling

- Sharding
  - Any DB distributed across multiple machines needs to know in what machine, a piece of data is stored or must be stored
  - A sharding system makes this decision for each row, using its key



## Sharding : Pros & Cons

- Pros
  - Increased read/write throughput
  - Increased storage capacity
  - High availability

- Cons
  - Query overhead

    Each sharded database must have a separate machine or service that understands how to route a querying operation to the appropriate shard. This introduces additional latency for every operation
  - Complexity of administration
  - Increased infrastructure costs

## NoSQL, No ACID

- RDBMSs are based on **ACID** (Atomicity, Consistency, Isolation, and Durability) properties
- NoSQL
  - Does not give importance to ACID properties
  - In some cases completely ignores them
- In distributed parallel systems it is difficult/impossible to ensure ACID properties
  - Long-running transactions don't work because keeping resources blocked for a long time is not practical

## BASE Transactions

- Acronym contrived to be the opposite of ACID
  - Basically Available,

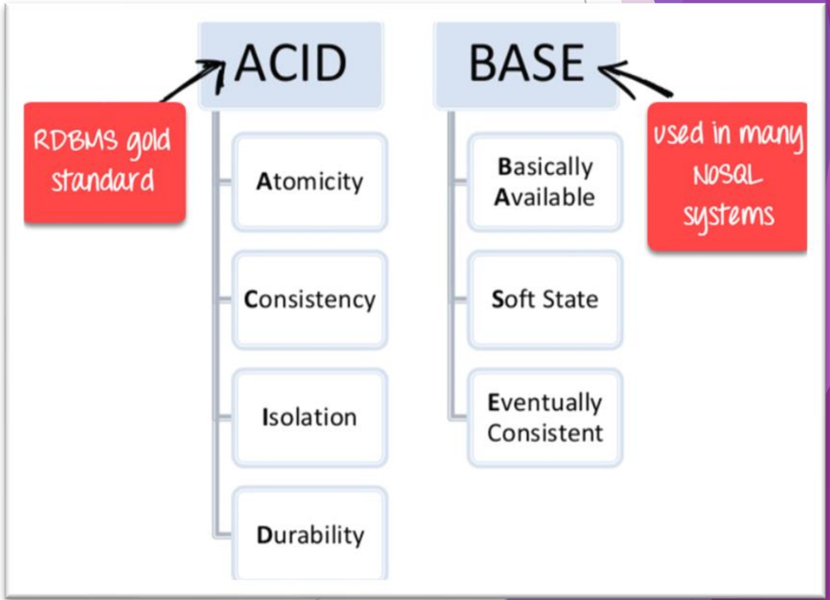
    Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure the availability of data by spreading and replicating it across the nodes of the database cluster
  - Soft state,

    Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers
  - Eventually Consistent

    The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality)

# BASE Transaction Characteristics

- Weak consistency – stale data OK
- Availability first
- Best effort
- Approximate answers OK
- Aggressive (optimistic)
- Simpler and faster



## CAP Theorem

In theoretical computer science, the CAP theorem, also named “Brewer's theorem” after computer scientist Eric Brewer, states that any distributed data store can provide only two of the following three guarantees.

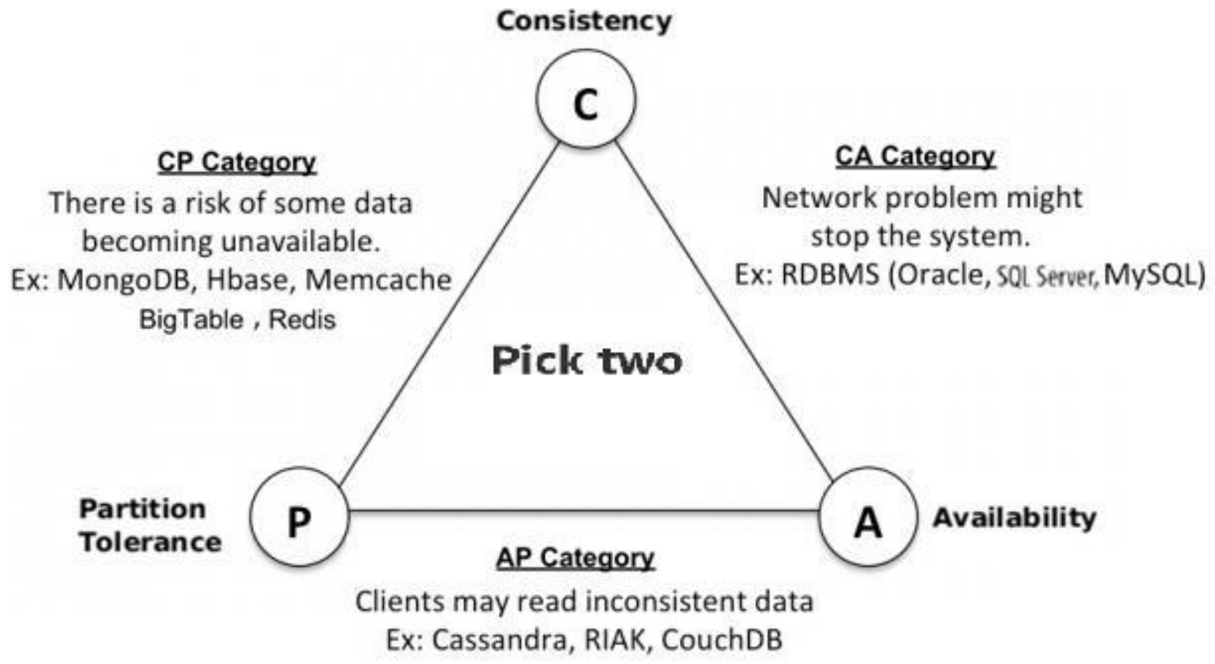
- Consistency

  Every read receives the most recent write or an error
- Availability

  Every request receives a (non-error) response, without the guarantee that it contains the most recent write.
- Partition tolerance

  The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

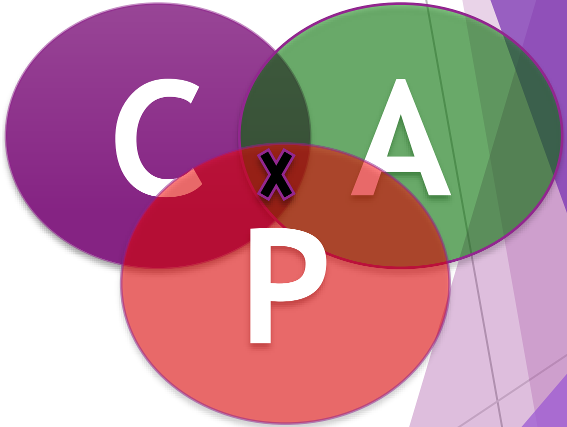
## CAP Theorem



Consistency
C
**CP Category**
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache BigTable , Redis
**CA Category**
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
**AP Category**
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

## Consistency or Availability

- Consistency and Availability is not "binary" decision
- AP systems relax consistency in favor of availability – but are not inconsistent
- CP systems sacrifice availability for consistency- but are not unavailable
- This suggests both AP and CP systems can offer a degree of consistency, and availability, as well as partition tolerance



## Performance

- There is no perfect NoSQL database
- Every database has its advantages and disadvantages
    - Depending on the type of tasks (and preferences) to accomplish
- NoSQL is a set of concepts, ideas, technologies, and software dealing with
    - Big data
    - Sparse un/semi-structured data
    - High horizontal scalability
    - Massive parallel processing
- Different applications, goals, targets, and approaches need different NoSQL solutions

## Where would we use it?

- Do you have somewhere a large set of uncontrolled, unstructured, data that you are trying to fit into a RDBMS?
  - Log Analysis
  - Social Networking Feeds (many firms hooked in through Facebook or Twitter)
  - External feeds from partners
  - Data that is not easily analyzed in a RDBMS such as time-based data
  - Large data feeds that need to be massaged before entry into an RDBMS

## Database Administration

- It does not matter if the data is deployed on a NoSQL platform instead of an RDBMS.
- Still need to address:
  - Backups & recovery
  - Capacity planning
  - Performance monitoring
  - Data integration
  - Tuning & optimization

## The Perfect Storm

- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a perfect storm
- Not a backlash/rebellion against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NoSQL offerings
  - So you have reached a point where a read-only cache and write-based RDBMS isn’t delivering the throughput necessary to support a particular application.
  - You need to examine alternatives and what alternatives are out there.
  - The NoSQL databases are a pragmatic response to the growing scale of databases and the falling prices of commodity hardware.

## Thank You