# Dynamo:

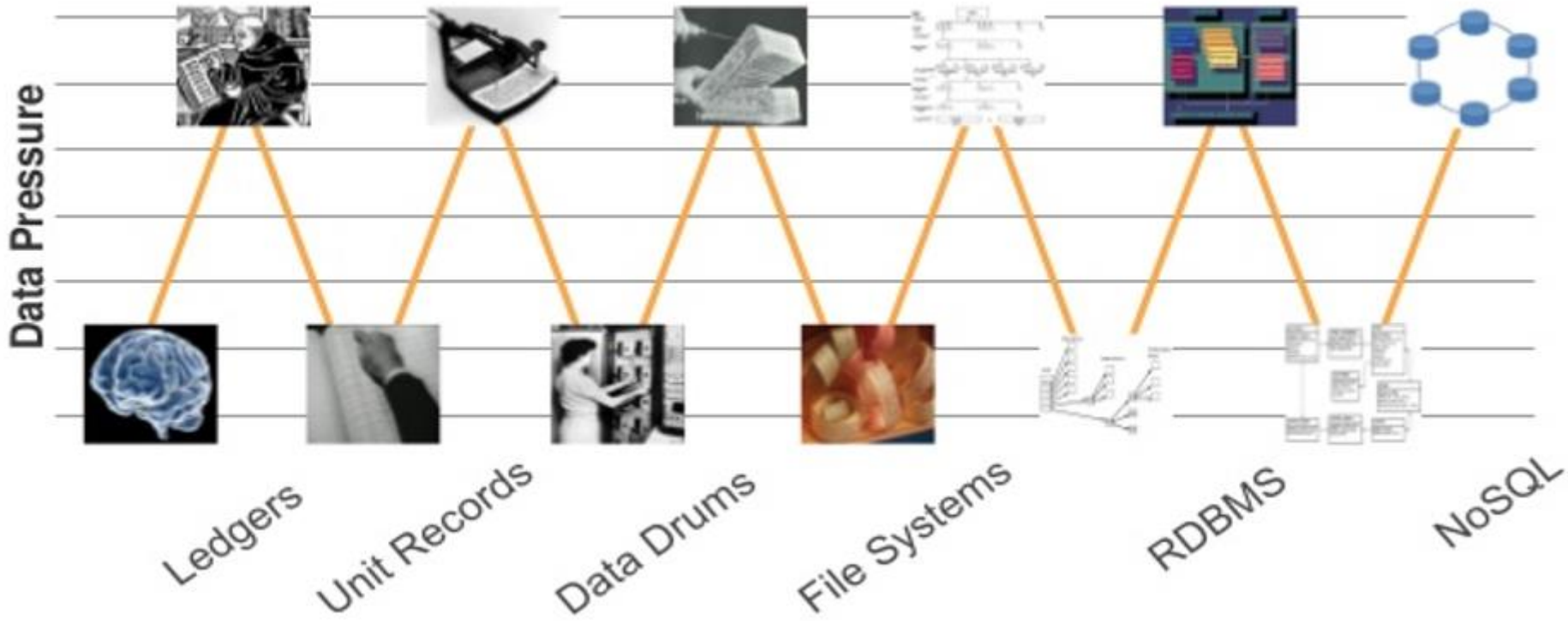
## Amazon's Highly Available Key-value Store

JITESH SONI

CMPT 843, SPRING – 2017, SFU

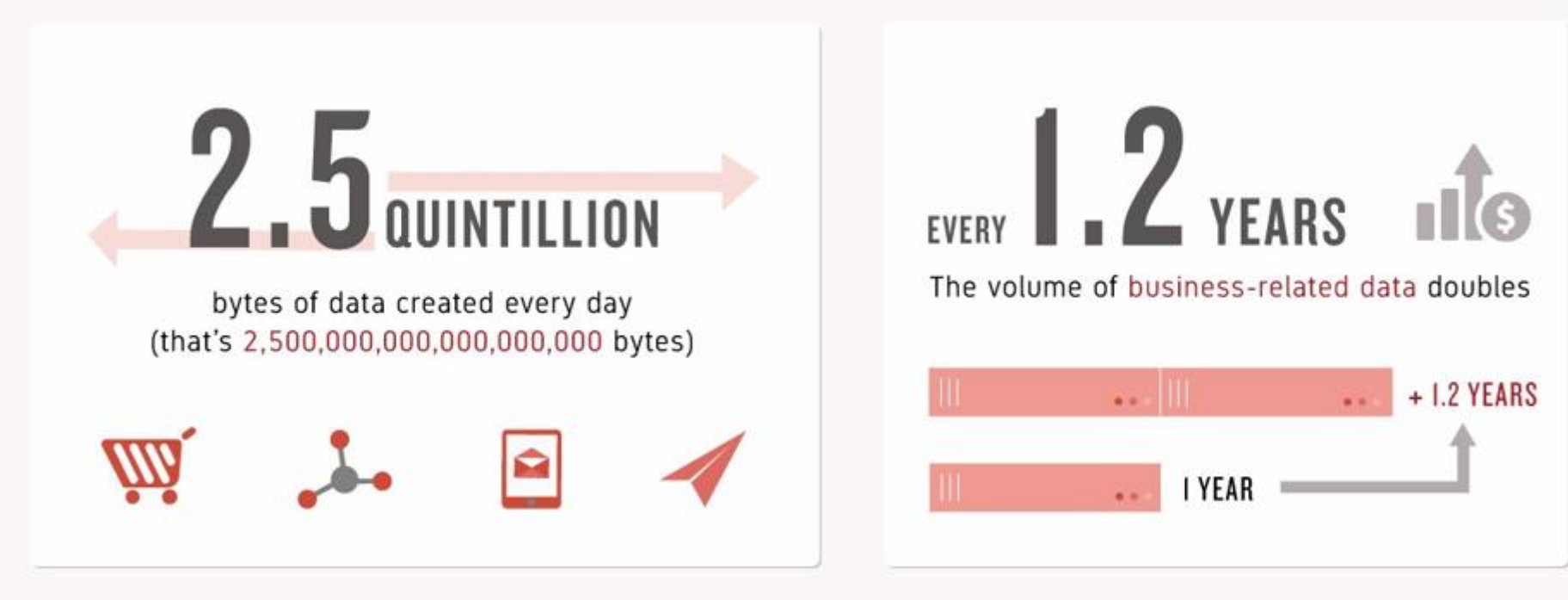
https://sfu-db.github.io/dbsystems/

# Timeline of Database Technology




From: https://image.slidesharecdn.com/gettingstartedwithamazondynamodb-160714160609/95/getting-started-with-amazon-dynamodb-4-638.jpg?cb=1468512393

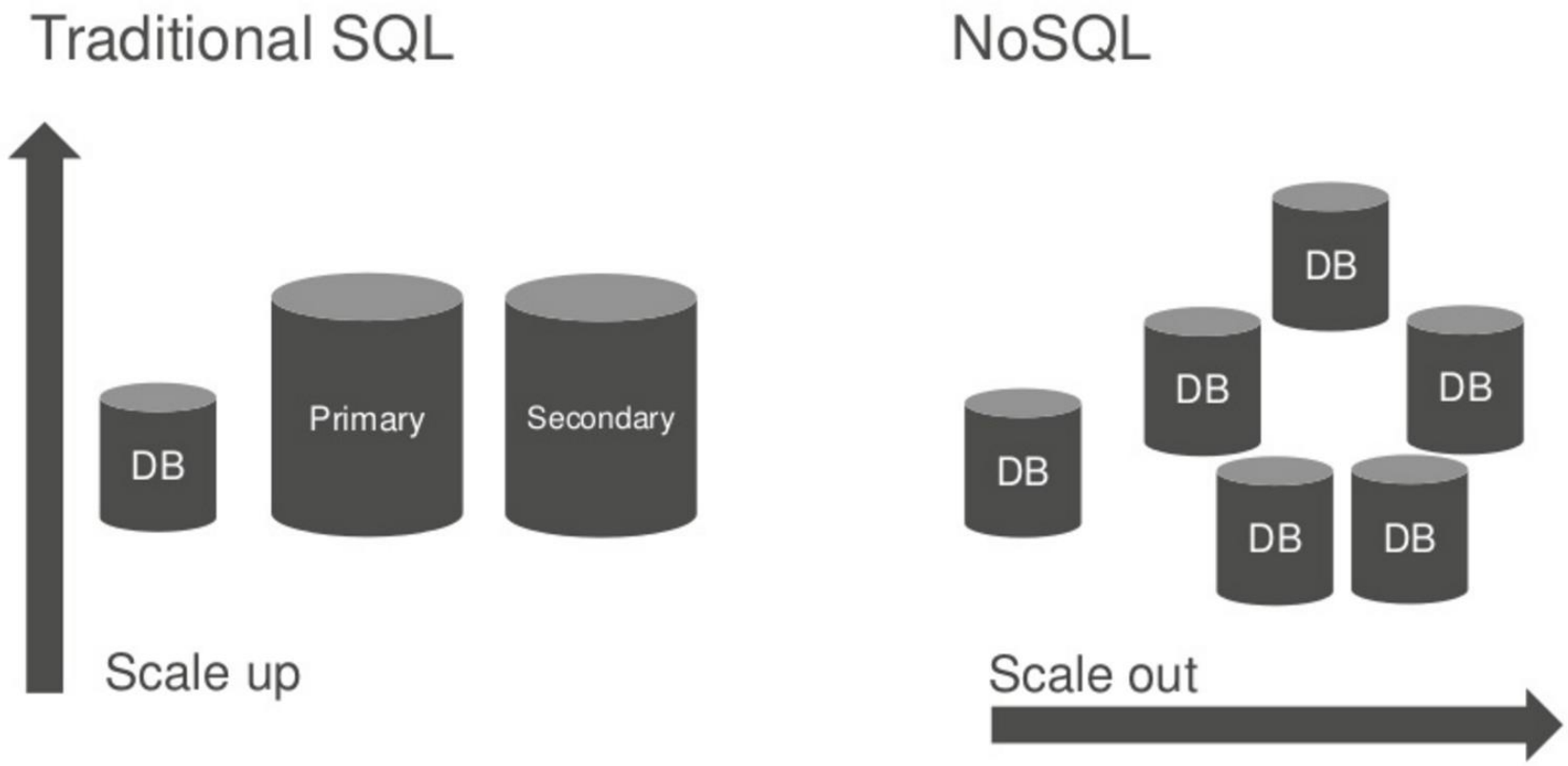
# Volume?




90% of the data in the world today has been created in the last two years alone

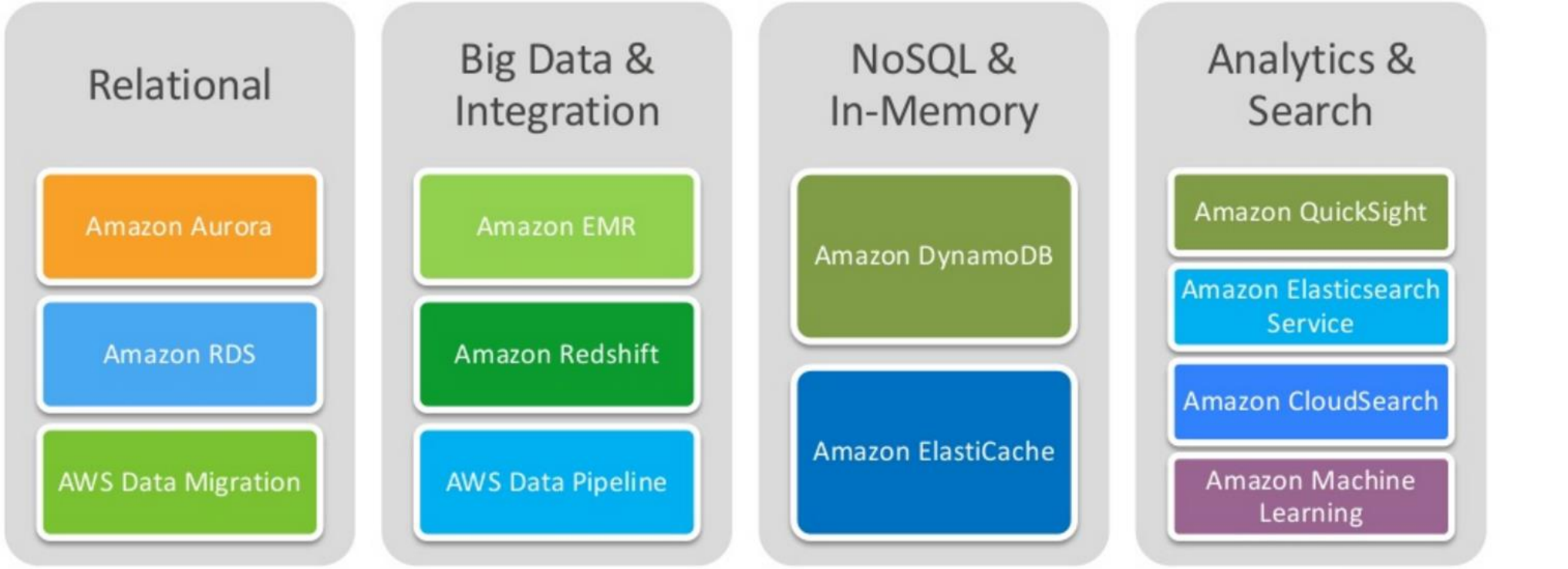
From: https://www.sfu.ca/computing/current-students/graduate-students/academic-programs/bigdata.html

# Motivation? Cost & Scale




From: https://www.slideshare.net/AmazonWebServices/getting-started-with-amazon-dynamodb-64031545

# AWS Database Landscape




From: https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307

# Hosting Your Database

- App optimization
- Scaling
- High availability
- Database backups
- DB s/w patches
- DB s/w installs
- OS patches
- OS installation
- Server maintenance
- Rack & stack
- Power, HVAC, net


From: https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307

# Hosting Your Database in Amazon EC2

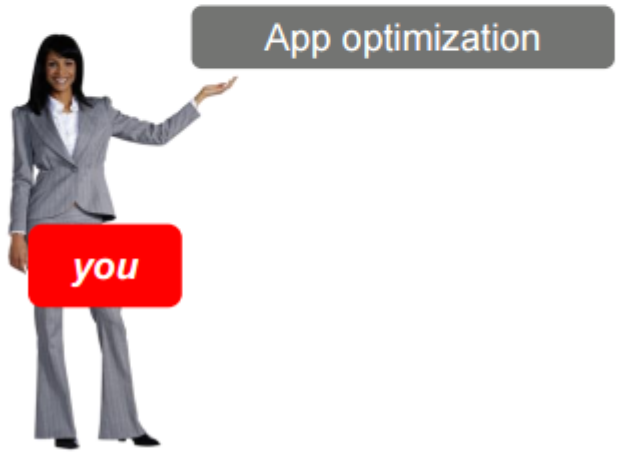

- App optimization
- Scaling
- High availability
- Database backups
- DB s/w patches
- DB s/w installs
- OS patches

- OS installation
- Server maintenance
- Rack & stack
- Power, HVAC, net

amazon web services

From: https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307

# Using an AWS Managed Database Service




- Scaling
- High availability
- Database backups
- DB s/w patches
- DB s/w installs
- OS patches
- OS installation
- Server maintenance
- Rack & stack
- Power, HVAC, net

amazon web services

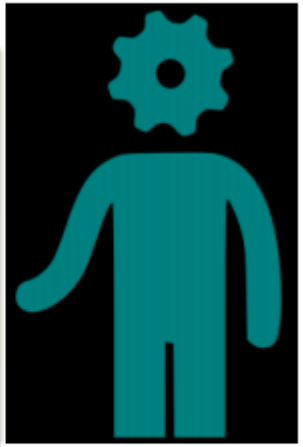
From: https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307

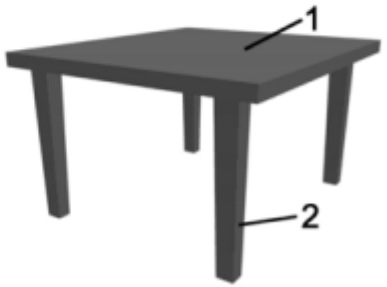
# Why Amazon DynamoDB?

## Relational

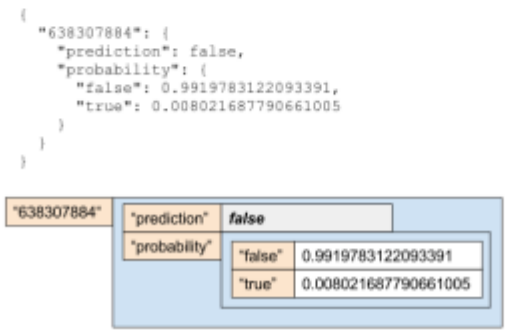
| Optimized for storage | Optimized for compute |
|---|---|
| Normalized/relational | Denormalized/hierarchical |
| Ad hoc queries | Instantiated views |
| Scale vertically | Scale horizontally |
| Mature solution | Emerging technology |

## NoSQL






1 = Row
2 = Column




From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307/11?src=clipshare

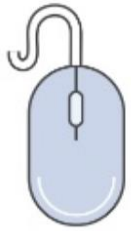
# Problem

- Design a always available database which functions under strict operational requirements and should be highly scalable
- Should be able to use resources efficiently
- Should scale out horizontally
- Should not have strict hardware requirements

# System Assumptions & Requirements

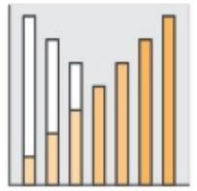
- Query Model: Simple read and write operations
- ACID (Atomicity, Consistency, Isolation, Durability): Choose eventual consistency for higher availability
- Dynamo does not provide any isolation guarantees and permits only single key updates
- Meet stringent latency requirements measured at $99.9^{th}$ percentile of the distribution; This was a new industry standard.

# Amazon DynamoDB

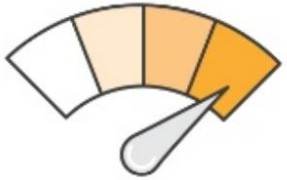


Fully managed NoSQL

Document or key-value



Scales to any workload



Fast and consistent

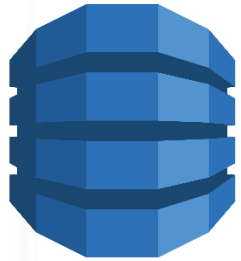
Access control

Event driven programming

From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-deep-dive-on-amazon-dynamodb-dat304/6?src=clipshare

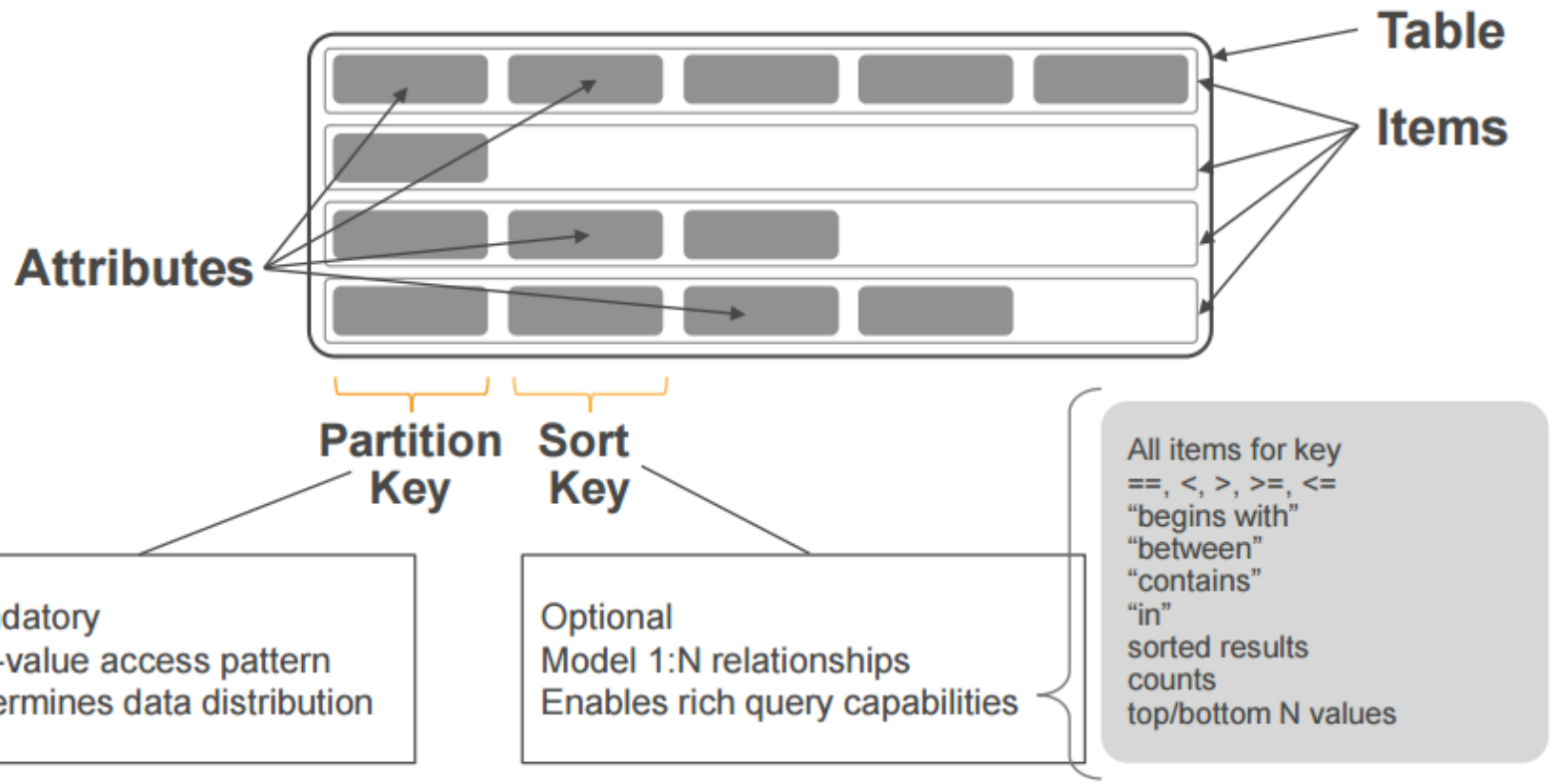
# Non-Relational Managed NoSQL Database Service

- Schema-less data model
- Consistent low latency performance (single digit ms)
- Predictable provisioned throughput
- Seamless Scalability
- No storage limits
- High durability and availability replication between 3 facilities
- Easy Administration – AWS scales for you
- Cost modelling on throughput and size



From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307/11?src=clipshare

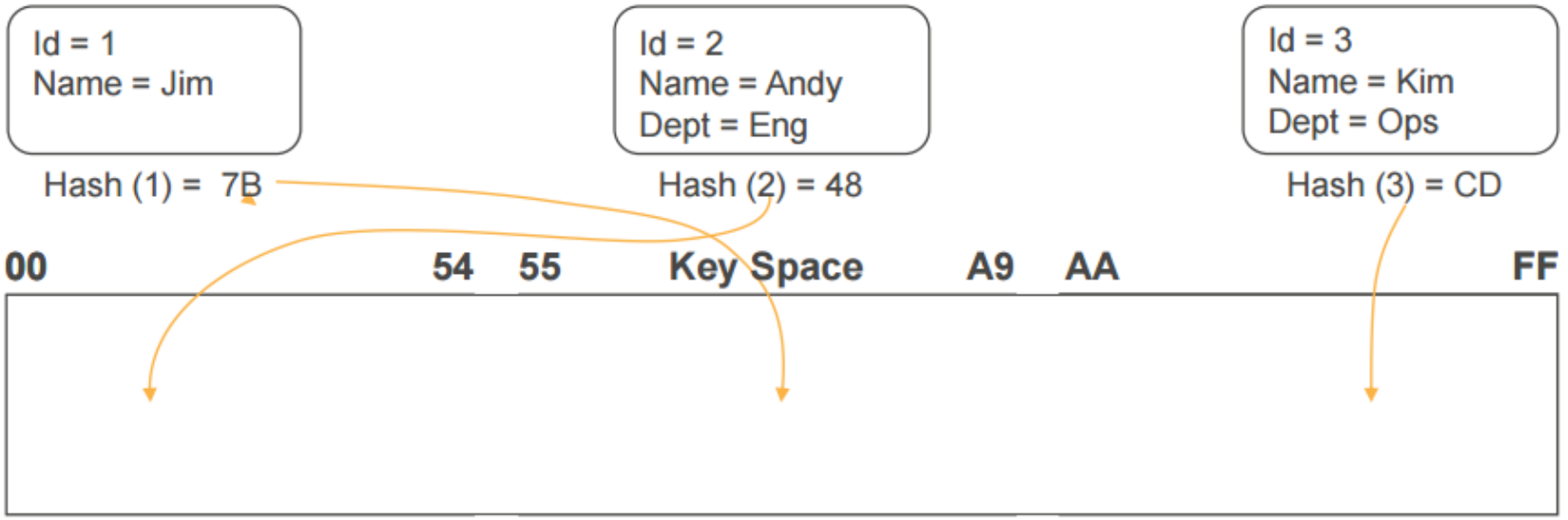
# Table




From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-deep-dive-on-amazon-dynamodb-dat304/6?src=clipshare

# Partition Keys

Partition key uniquely identifies an item
Partition key is used for building an unordered hash index
Allows table to be partitioned for scale




From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-deep-dive-on-amazon-dynamodb-dat304/6?src=clipshare

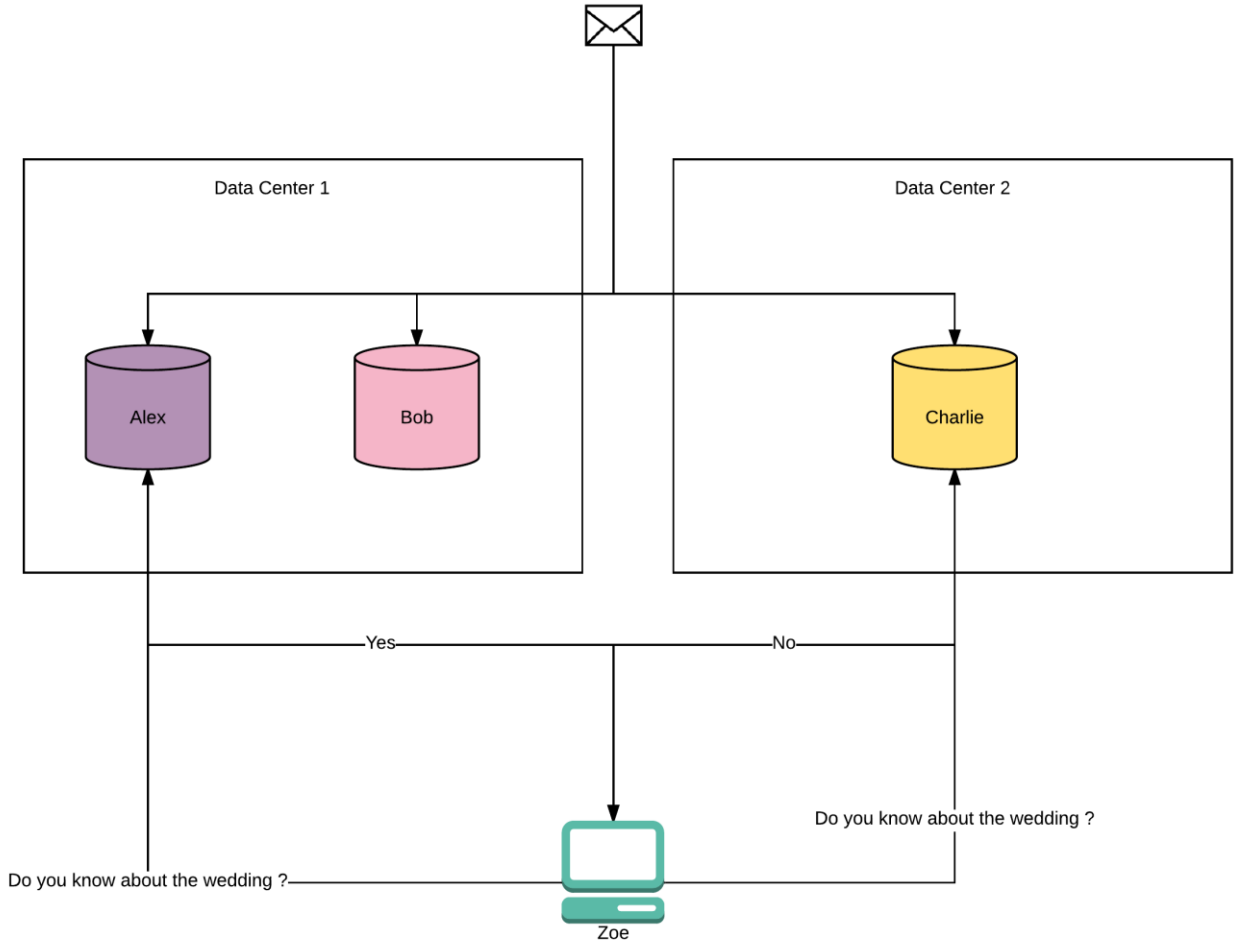
# Design Concepts:1

- Eventually consistent means for certain periods of time, more than 1 version of the same data exists in the system
- When do we resolve these conflicts ? Read or Write ?
- We cannot reject customer writes
- Last write wins

# Eventual Consistency (Wedding Invitation)

Suppose you mail out your wedding invitation to your Friends Alex, Bob and Charlie.

1. Alex and Bob get the wedding invitation on the same day.
2. Alex tells Charlie about the wedding invitation. In the meanwhile, Charlie might feel betrayed as he has not been invited.
3. Charlie receives the invitation a few days later and now is really excited about the wedding.

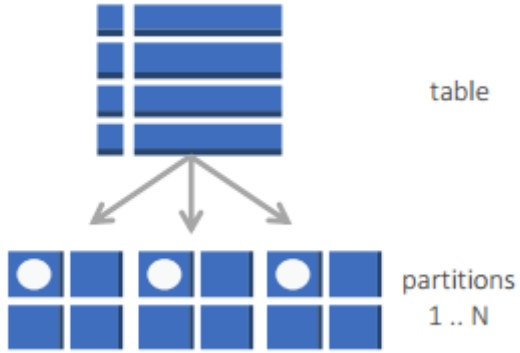


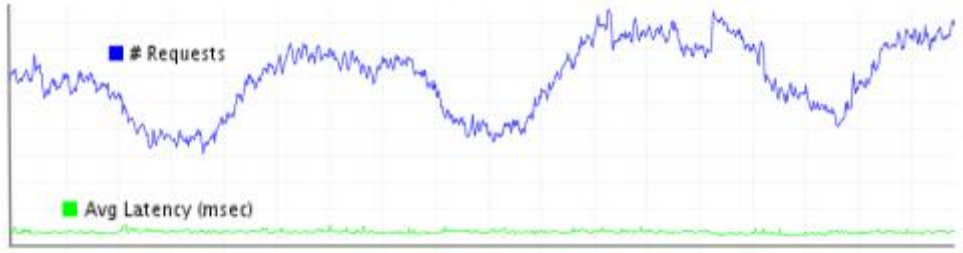
# Design Concepts: 2

- Incremental scalability: Add nodes with minimal overhead
- Symmetry: Every node has the same set of responsibility
- Decentralization: Peer to peer communication over central node design

# Scalability

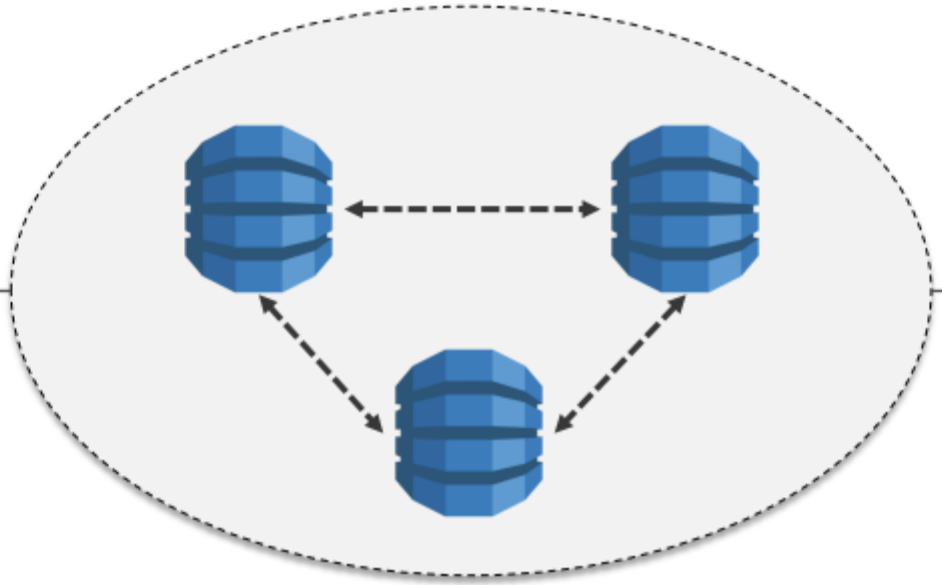
- No throughput limit
- No storage limit
- DynamoDB automatically partitions data when:
  - Data set grows
  - Provisioned capacity increases




From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307/11?src=clipshare

# Durability



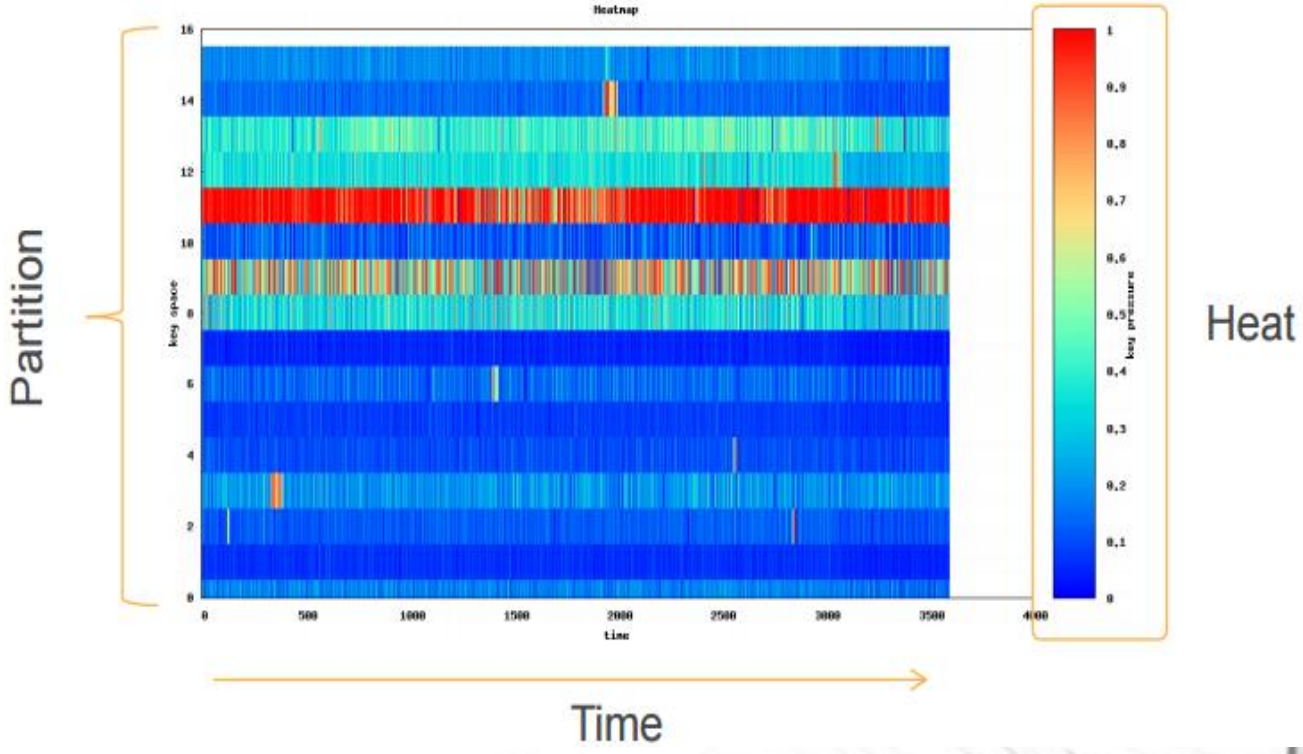
**WRITES**
3-way replication
Quorum acknowledgment
Persisted to disk (custom SSD)

**READS**
Strongly or eventually consistent
No trade-off in latency

From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-introduction-to-managed-database-services-on-aws-dat307/11?src=clipshare

# Throttling (Hot Keys!)

- Non-uniform workloads
  - Hot keys/hot partitions
  - Very large items
- Mixing hot data with cold data
  - Use a table per time period




From: http://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-deep-dive-on-amazon-dynamodb-dat304/6?src=clipshare

# How to avoid Throttling?

- " To get the most out of DynamoDB throughput, create tables where the partition key has a large number of distinct values, and values are requested fairly uniformly, as randomly as possible." - Amazon DynamoDB, Developer Guide

# System Interface

- Get
- Put
- Scan*
- Query*

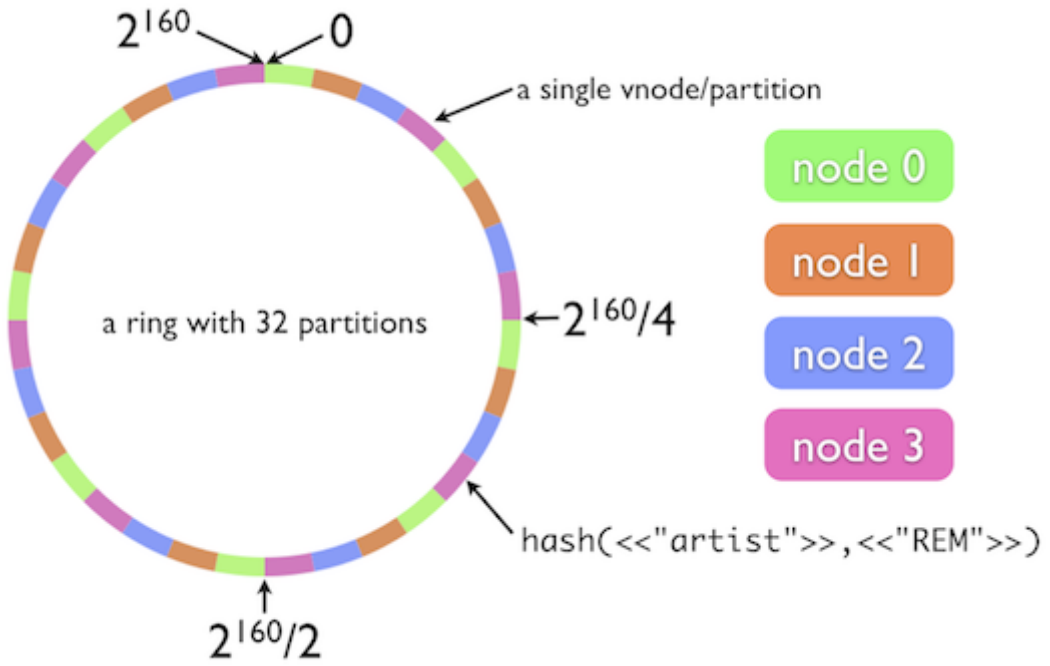
MD5(Key) -> 128 bit identifier, which is used to determine the storage node where the data is saved

* Not covered in the paper

# Architecture- Partitioning

- Consistent hashing is a special kind of hashing such that when a hash table is resized and consistent hashing is used, only K/n keys need to be remapped on average, where

- K is the number of keys
- n is the number of slots




From: http://cloudacademy.com/blog/dynamodb-replication-and-partitioning-part-4/

# Architecture- Partitioning (Challenges)

- Random position assignment of each node leads to non-uniform data and load distribution.
- DynamoDB uses the concept of **virtual nodes**. A virtual node looks like a single data storage node but each node is responsible for more than one virtual node.
- The number of virtual nodes that a physical node is responsible can be decided on it's capacity accounting for heterogeneity

# Architecture: Replication

- Every data item is replicated at N hosts. Each key is assigned to a coordinator node, which is responsible for READ and WRITE operation of that particular key.
- The job of this coordinator node is to ensure that every data item that falls in its range is stored locally and replicated to N - 1 clockwise successor nodes.

# Architecture: Versioning

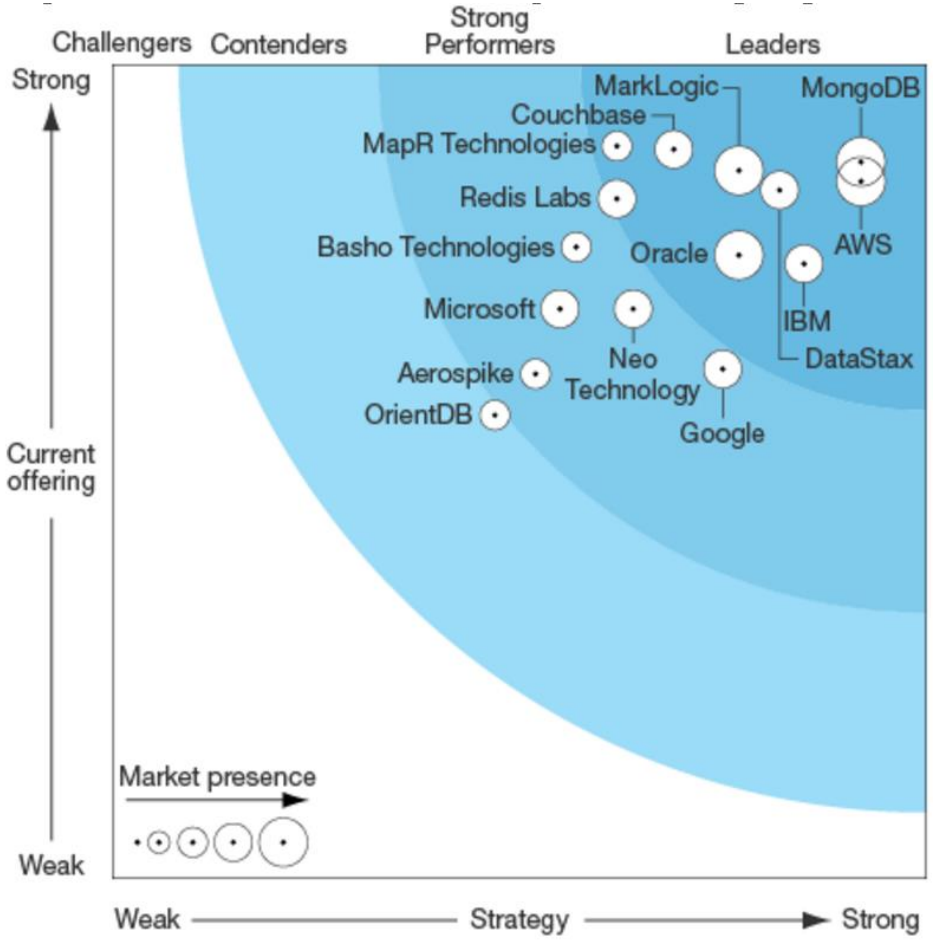
- Put operations returns before updating all records
- Dynamo use vector clocks protocol for versioning the data. It also stores timestamp to help resolve conflicts.
- *Optimistic locking* is a strategy to ensure that the client-side item that you are updating (or deleting) is the same as the item in DynamoDB. If you use this strategy, then your database writes are protected from being overwritten by the writes of others — and vice-versa.

From: http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBMapper.OptimisticLocking.html

# Reconciliation

- Business logic specific reconciliation: Clients are responsible to implement this logic when multiple conflicting versions of an object exists
- Timestamp based reconciliation: Last write wins

# NoSQL Offerings




From: https://webassets.mongodb.com/_com_assets/cms/Forrester-Wave-Big-Data-NoSQL-Q3-2016-m7xwqfac23.gif

# Contribution

- A new system with tunable parameters of N, R & W
- 99.9% availability was introduced as new industry standard to measure performance
- Eventually consistent model
- Providing an interface for client applications to reconcile conflict versions

# Conclusion

- A new horizontally scalable distributed key-value store complying with stringent performance requirements was developed
- Dynamo was more transparent in the way system worked, rather than being a black box in comparison to relational database.
- Application developers had more flexibility and control over the system; to tune parameters to best suite needs of their application
- Emphasis on the increasing importance of availability, performance over consistency

# Thank You !