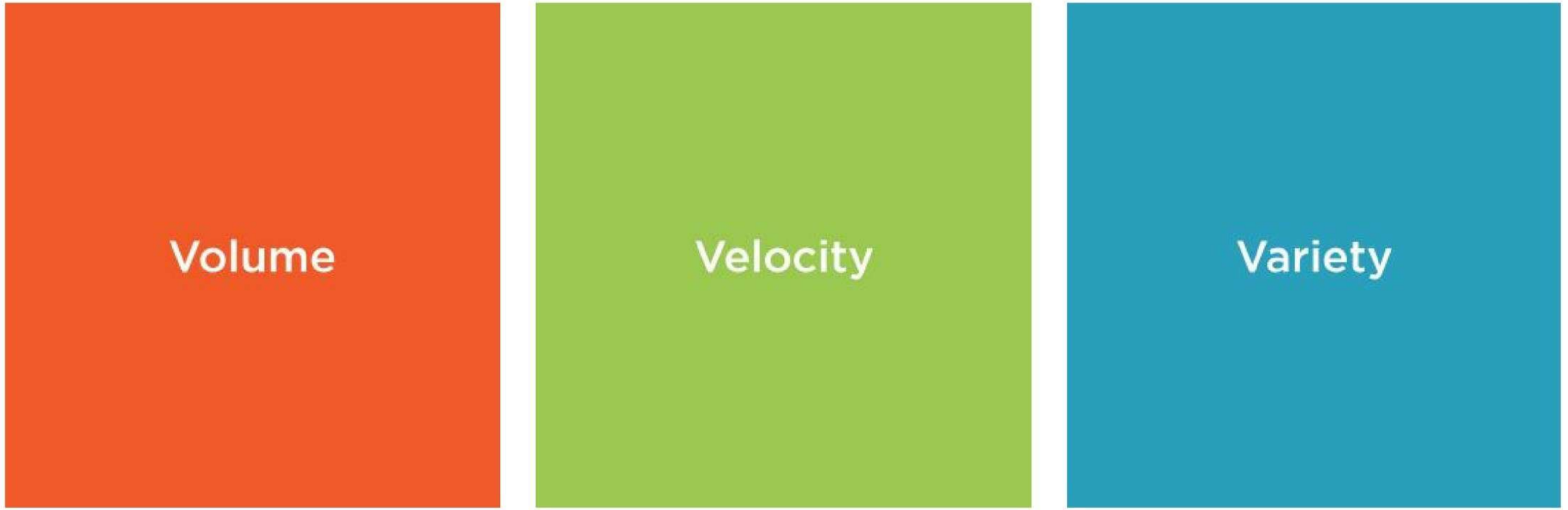
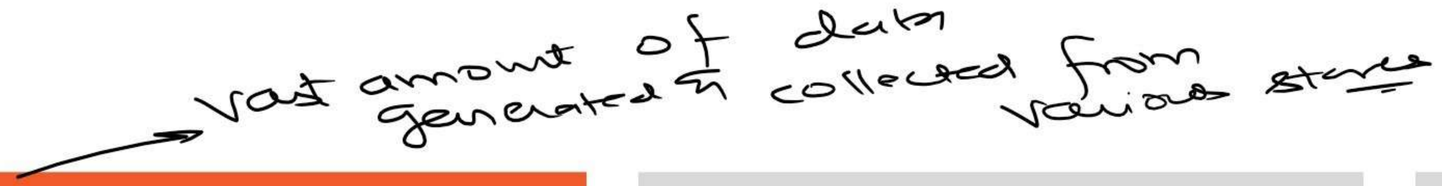
What Is Big Data?

Big Data Characteristics



Big Data Characteristics



Volume

Velocity

Variety

Big Data Characteristics

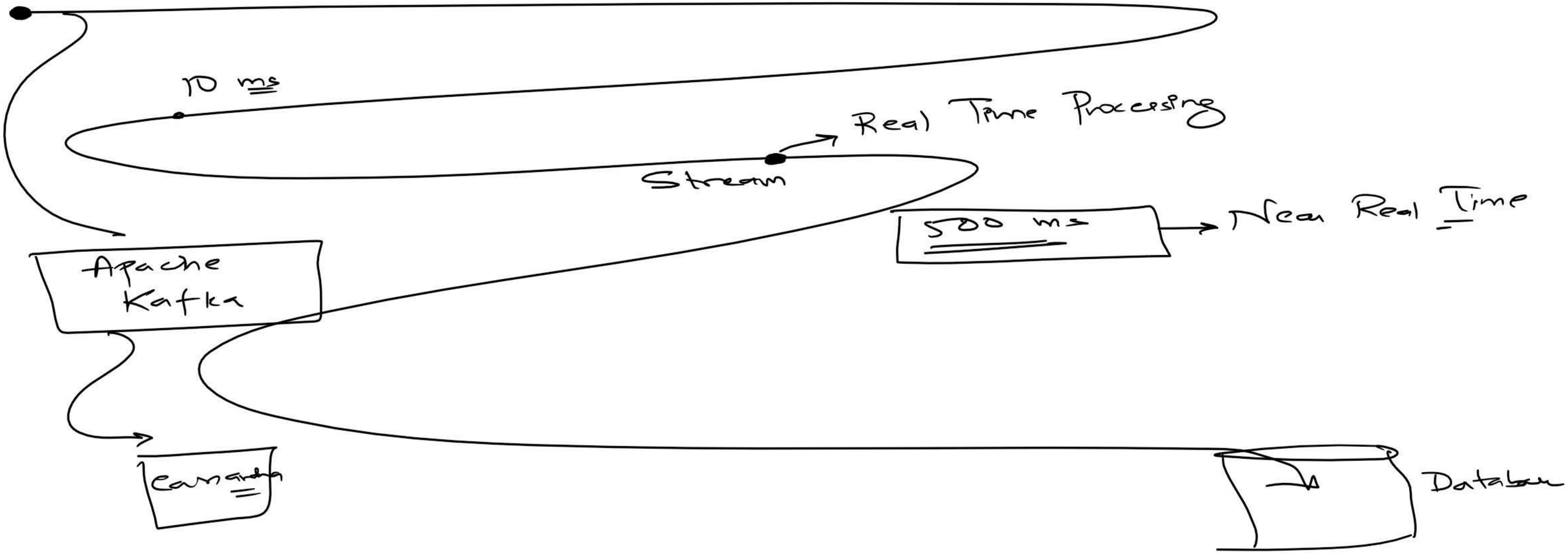
speed at which data is generated

RT / NRT

Volume

Velocity

Variety


10 ms
Real Time Processing
Stream
500 ms
Near Real Time
Apache
Kafka
Cassandra
Database

Big Data Characteristics

Volume

Velocity

Variety

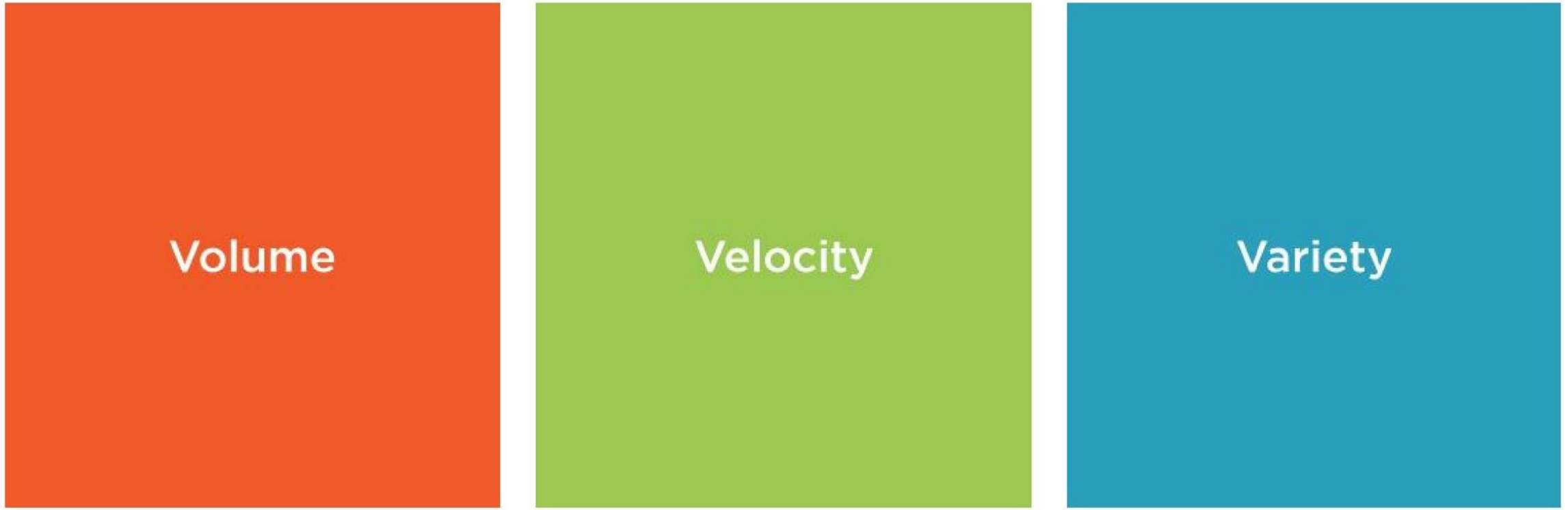
different type of data

Structured

Semi - Structured

Unstructured

Big Data Characteristics

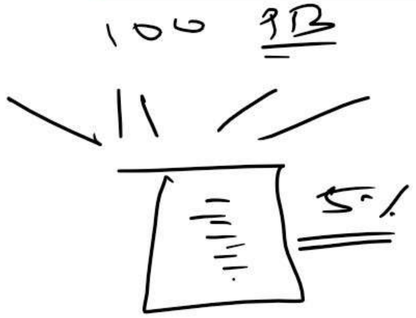


Big Data Characteristics

Volume

Velocity

Variety



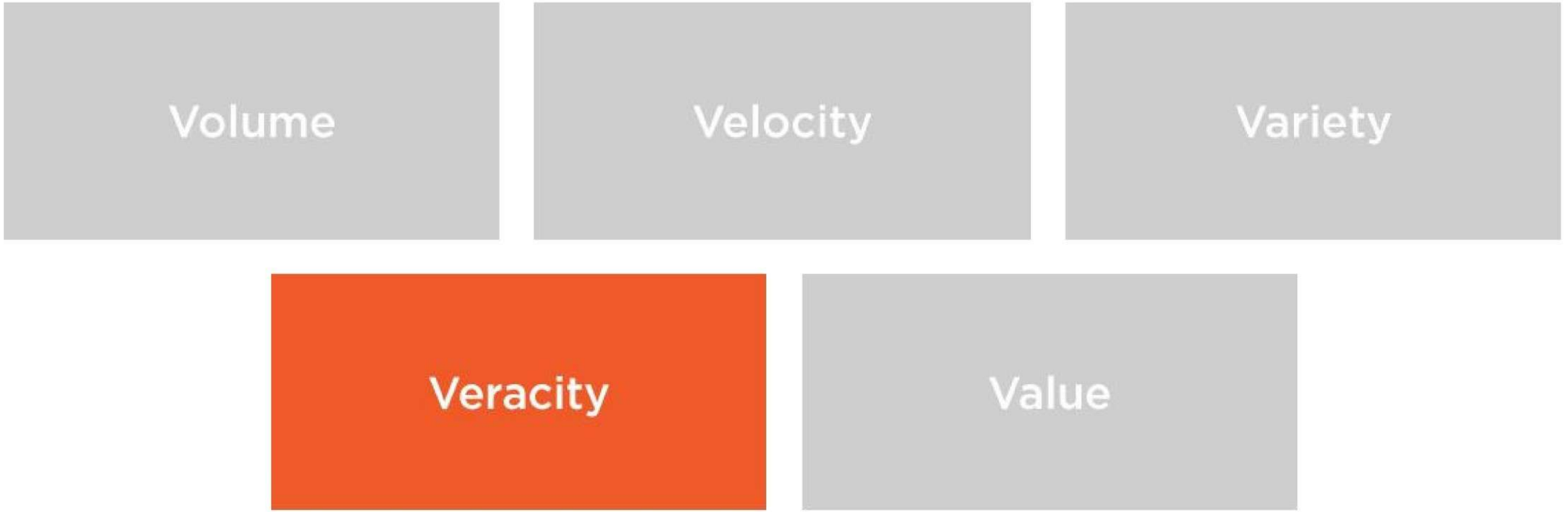
Indicates the quality & reliability of data

Veracity → IBM

Validation

Value

Big Data Characteristics



Big Data Characteristics

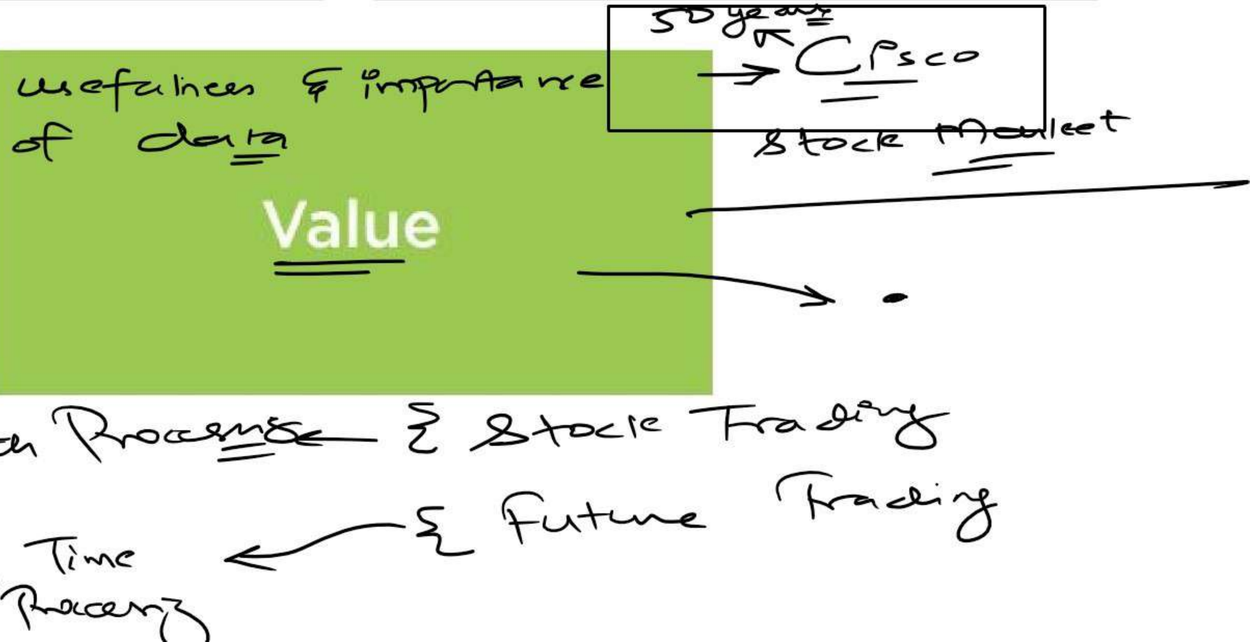
Volume

Velocity

Variety

Veracity

Value



Types of Big Data

	WHERE [Delimiters]	WHAT [What that part exactly means]
UNSTRUCTURED	✗	✗
text extract, image, videos audio, pdf		
SEMI STRUCTURED	✓	✗
csv, tsv		
STRUCTURED	✓	✓
json, db-tables		

Hello how are you

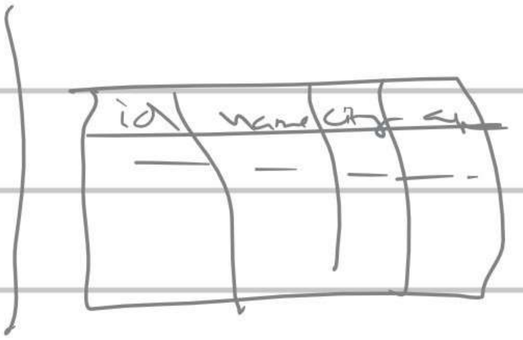
I am fine, how

had lunch?

No

1, Tom, IT, 25000, 2014-01-09

{ id: 1,
name: 'Tom',
city: "Mumbai"
}



Types of Big Data

Types of Big Data

Structured

Semi-structured

Unstructured

Types of Big Data

Structured

Semi-structured

Unstructured

(A short) History of Big Data

Big Data Projects

Flink

APACHE Spark™

APACHE hadoop

STORM

Projects Directory

Home | Committees | Projects | Releases | Statistics | Timelines | Search... | About

Project listings:

- By Name
- By Committee
- By Category
- By Programming Language
- By Number of Committers

Projects by category:

- **big-data (49):**
  - Apache Accumulo
  - Apache Airavata
  - Apache Ambari
  - Apache Apex (in the Attic)
  - Apache Avro
  - Apache Beam
  - Apache Bigtop
  - Apache BookKeeper
  - Apache Calcite
  - Apache Camel
  - Apache CarbonData
  - Apache CouchDB
  - Apache Crunch (in the Attic)
  - Apache Daffodil
  - Apache DataFu (Incubating)
  - Apache DirectMemory (in the Attic)
  - Apache Drill
  - Apache Edgent (Incubating)
  - Apache Falcon (in the Attic)
  - Apache Flink
  - Apache Flume
  - Apache Fluo
  - Apache Fluo Recipes

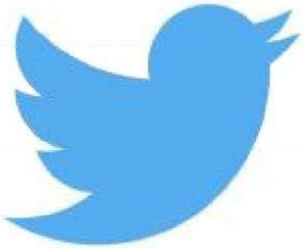
Big Data Goes Mainstream

What Do These Companies Have In Common?

CLOUDERA

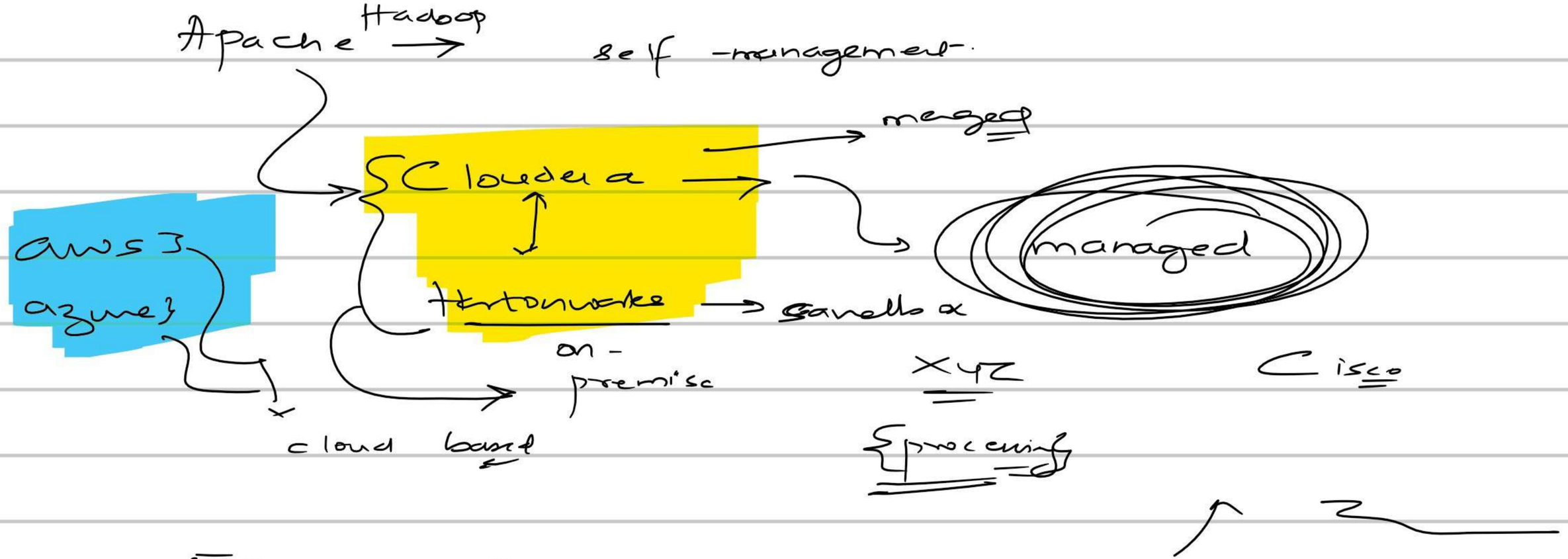
yahoo!

Google



Microsoft

aws



500 MB → Pen Drive

2 GB → HDD PD

Store — 2 GB


BUSINESS SOLUTION
GLOBAL NETWORK
MARKETING STRATEGIC


Bay
8
Bay
16
PS
6

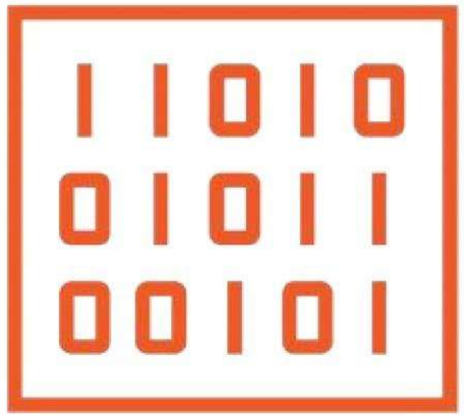

Weather Challenges
Soil Quality
Water Management

What Does Big Data Do and HOW Does It work

How Big Data Works

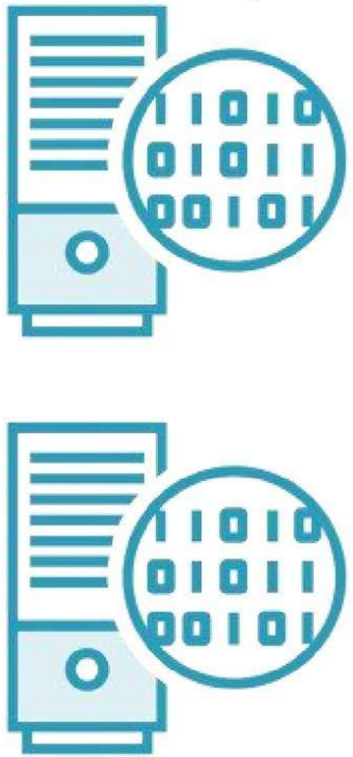
HDFS

storage

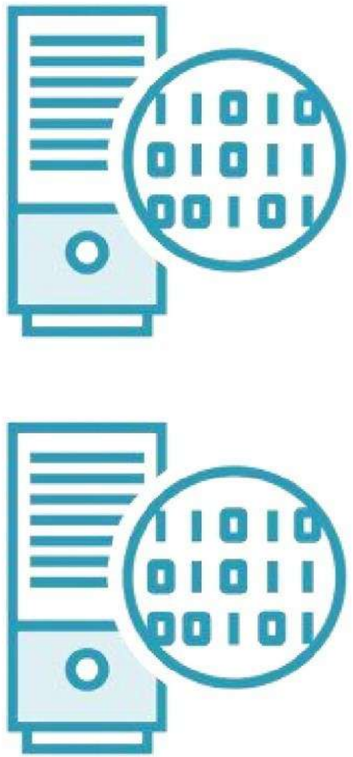


How Big Data Works

Processing [Map Reduce]

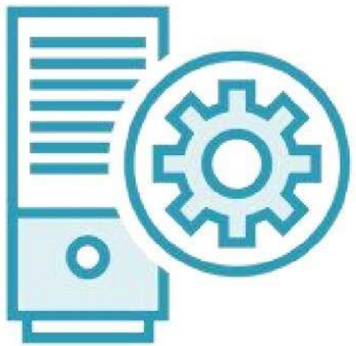


How Big Data Works

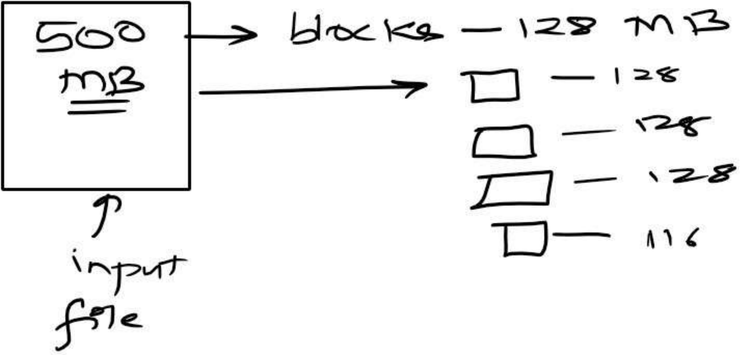


How Big Data Works

Word Count



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tincidunt rhoncus lorem, vitae lacinia dolor aliquam nec. Donec nec pretium augue. Praesent tempus risus sit amet ex molestie ullamcorper. Suspendisse potenti. Proin non libero mauris. Aenean felis leo, tincidunt sit amet felis nec, aliquam euismod ipsum. Proin porttitor efficitur purus, id congue purus pellentesque eu. Cras vitae lectus pretium, consequat metus elementum, dignissim sapien. Ut volutpat quis augue eu tristique. Sed vel purus in odio efficitur tincidunt et ac metus. Quisque eget faucibus tellus.



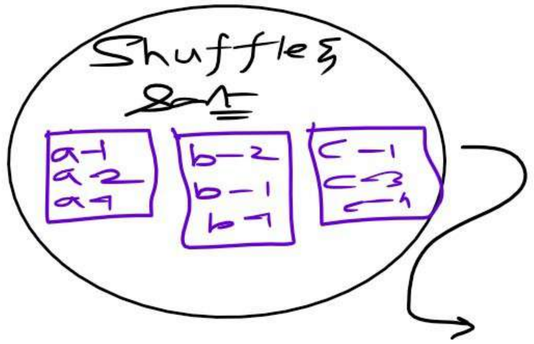
Mapper

a — 1
b — 2
c — 1

Word Count

a — 2
b — 1
c — 3

a — 4
b — 1
c — 4



Processing → Map Reduce

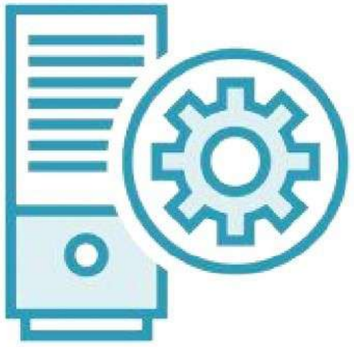
Reducer → Aggregation

a — [1, 2, 1]
b — [2, 1, 1]
c — [1, 3, 4]

a — 4
b — 4
c — 8

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tincidunt rhoncus lorem vitae lacinia dolor aliquam nec. Donec nec pretium augue. Praesent tempus risus sit amet ex molestie ullamcorper. Suspendisse potenti. Proin non libero mauris.

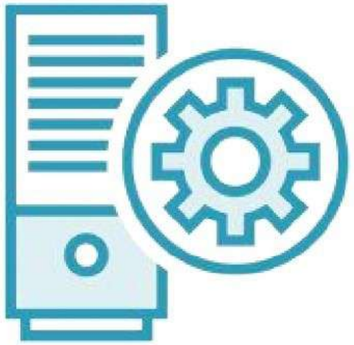
Donec — 2
lorem — 2
tempus — 1



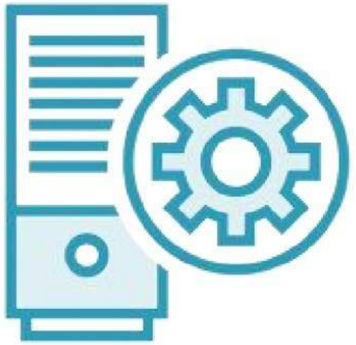
Aenean felis leo, tincidunt sit amet felis nec, aliquam euismod ipsum. Proin porttitor efficitur purus, id congue purus pellentesque eu. Cras vitae lectus pretium, consequat metus elementum, dignissim sapien. Ut volutpat quis augue eu tristique. Sed vel purus in odio efficitur tincidunt et ac metus. Quisque eget faucibus tellus.

Word Count

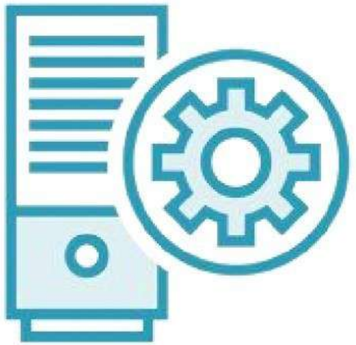
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec tincidunt rhoncus lorem, vitae lacinia dolor aliquam nec.



Aenean felis leo, tincidunt sit amet felis nec, aliquam euismod ipsum. Proin porttitor efficitur purus, id congue purus pellentesque eu.



Donec nec pretium augue. Praesent tempus risus sit amet ex molestie ullamcorper. Suspendisse potenti. Proin non libero mauris.



Cras vitae lectus pretium, consequat metus elementum, dignissim sapien. Ut volutpat quis augue eu tristique. Sed vel purus in odio efficitur tincidunt et ac metus. Quisque eget faucibus tellus.

Big Data platforms

Modern Data platformn

A future-proof architecture for Business Analytics

Your Enterprise Data Cloud


Data Hub Clusters


Data Engineering


Data Warehouse


Operational Database


Machine Learning

Control Plane

Powered by Cloudera

Feedback

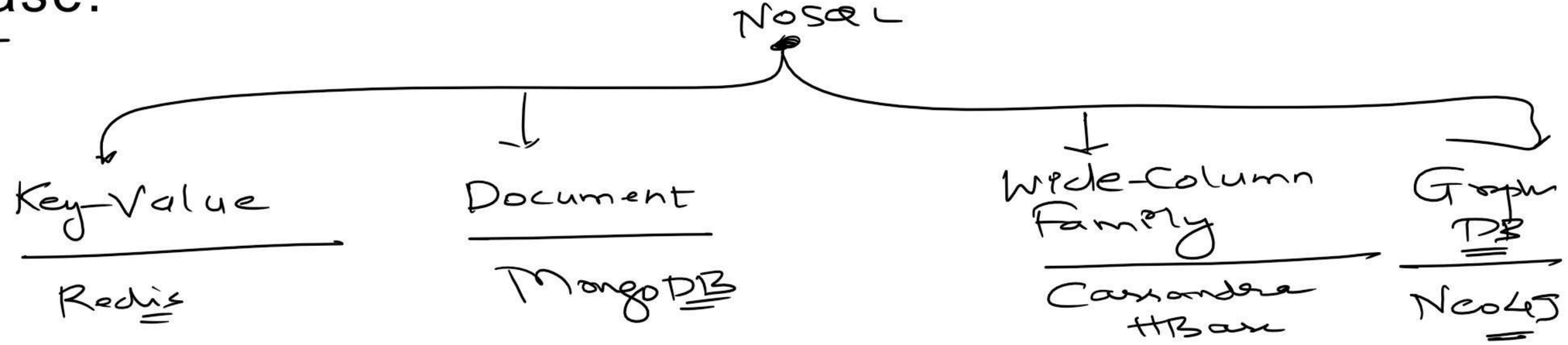
Types of Databases

No SQL

Not Only SQL

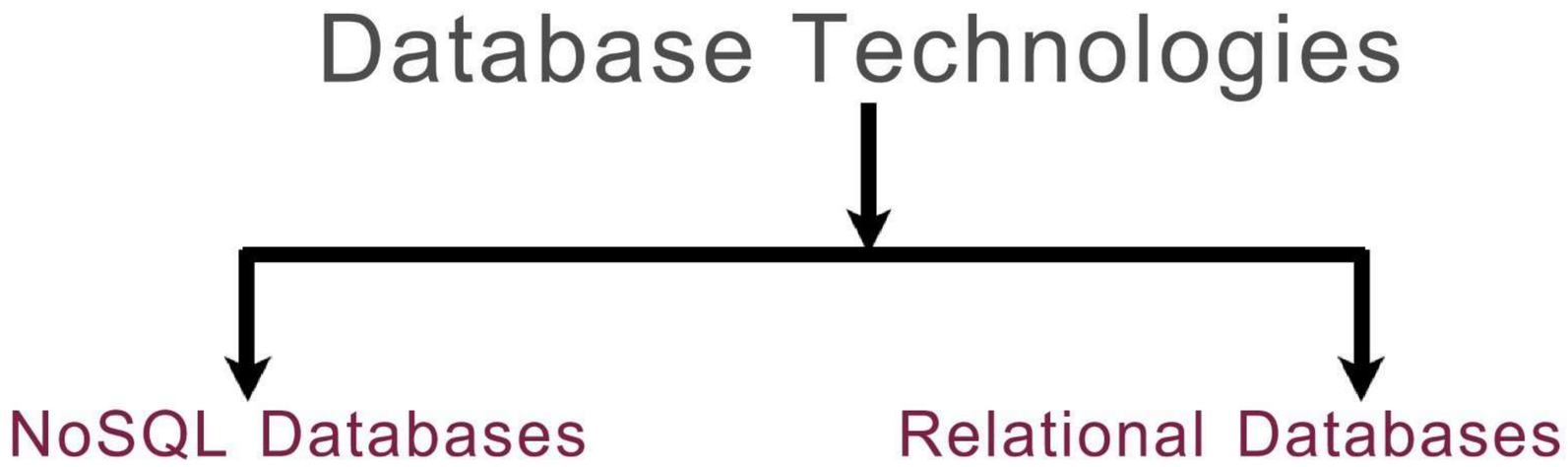
NoSQL Database

Generic term used for any non-relational database.

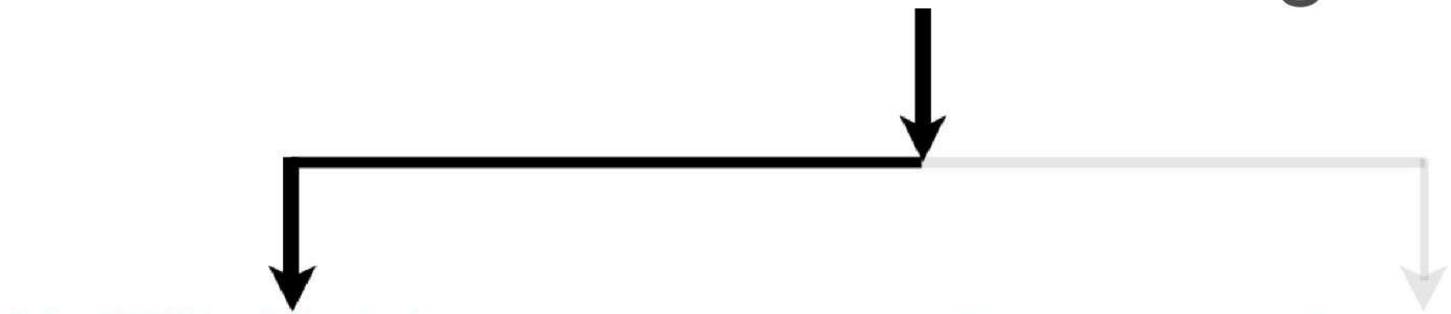


Relational Database

Generic term used for any database that stores data logically organized into relations - tables with rows and columns.


Database Technologies
NoSQL Databases
Relational Databases

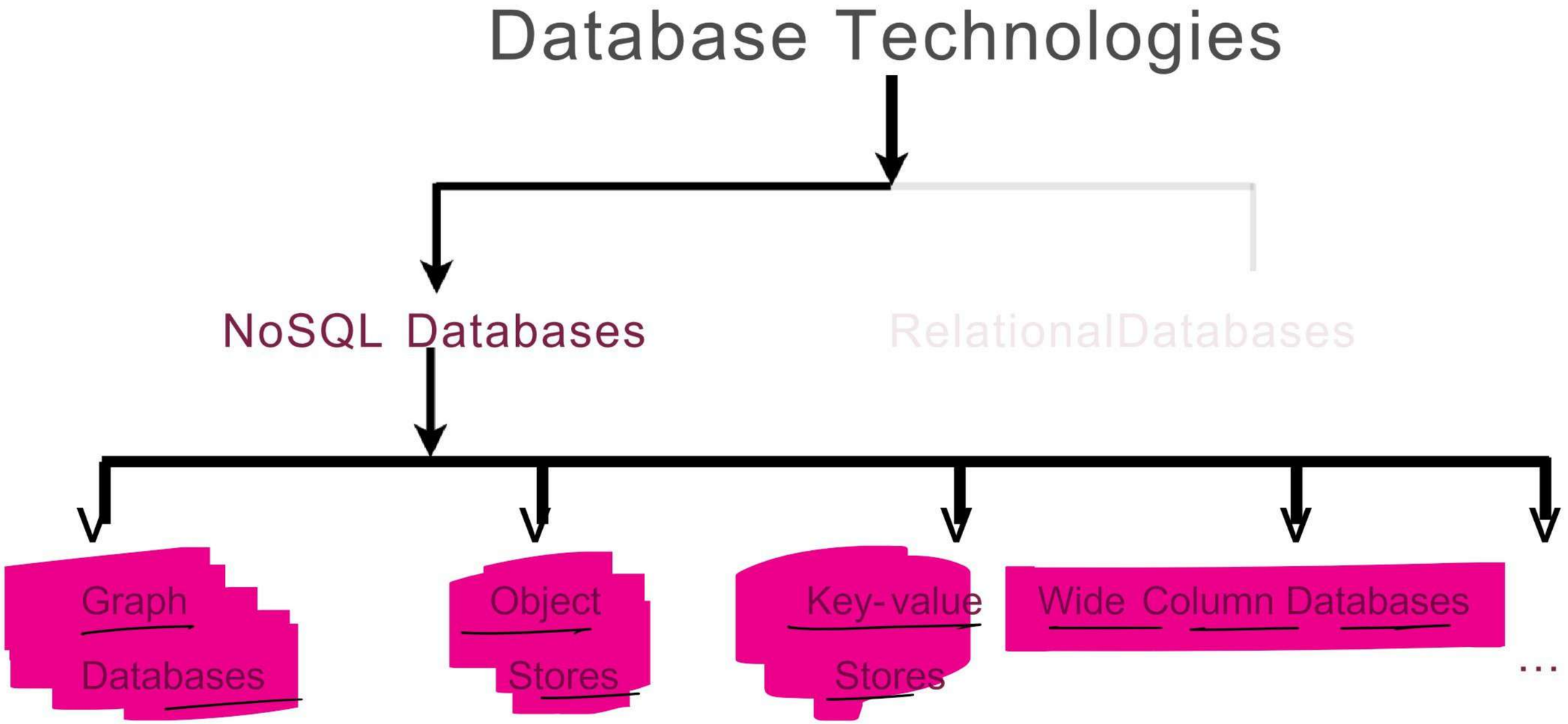
Database Technologies



NoSQL Databases

Relational Databases

Variety of data models


Database Technologies
NoSQL Databases
RelationalDatabases
Graph Databases
Object Stores
Key-value Stores
Wide Column Databases
...

Structured and Unstructured Data

Problem: Creating Customer Profiles

Imagine you work at an electronics company with different sales channels

- Online store
- Physical stores
- Third-parties

Problem: Creating Customer Profiles

Customer data from online store

- User ID
- Name
- Credit-card details
- Address
- Date of birth
- Points

Customer Profile: Online Store

User ID	Name	Credit Card	Address	DOB	Points
C1	Jill	4034	Sydney	3-Jun-1976	387
C2	John	3323	Perth	12-Feb-1987	231
C3	Mike	2300	Adelaide	30-Dec-1990	45
C4	Lilly	9831	Hobart	7-May-1981	24

A standard structure can be expected from this source

Problem: Creating Customer Profiles

Customer data from physical stores

- Credit-card details?
- Name?
- User ID?

Different details will be available for different customers

Customer Profile: Physical Stores

User ID	Name	Credit Card	Store	Points
	Brianna	1011	Hobart	345
C23	Luca		Brisbane	128
	Ayako	8346	Alice Springs	
C43			Brisbane	137

The data is unstructured

Problem: Creating Customer Profiles

Customer data from third parties

- Fields available may vary based on third party
- Format and types will be different

Amazon → Date → java.util.Date

XYZ → Date → "String"

3 different sources

Online

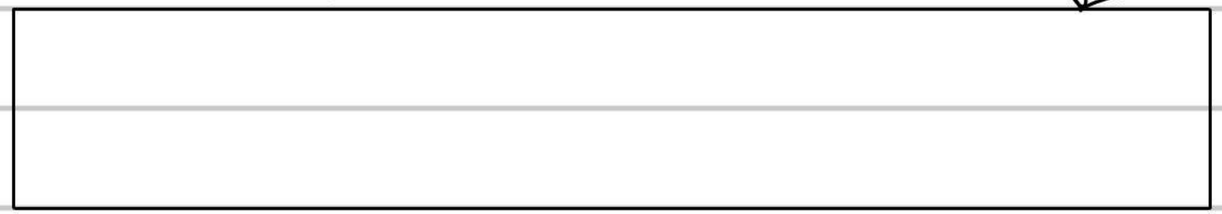
Physical

Third Party

structured

unstructured

structured but different format



RDBMS?

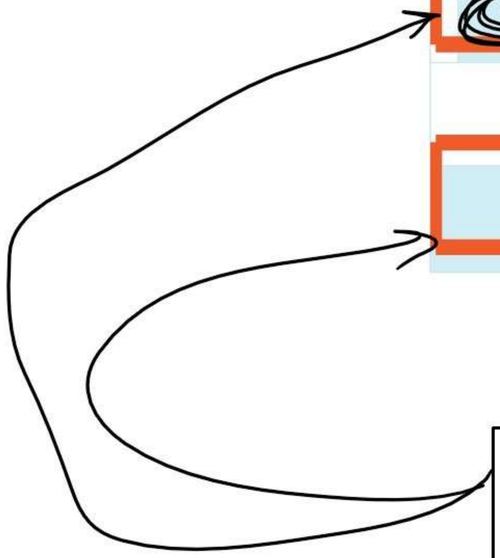
Customer Profile

	User ID	Name	Credit Card	Address	DOB	Points	Age Group
OS	C1	Jill	4034	Sydney	3-Jun-1976	387	
OS	C2	John	3323	Perth	12-Feb-1987	231	
PS		Brianna	1011	Hobart		345	
TP		John		Perth		198	31-35

A lot of fields are empty

Customer Profile

User ID	Name	Credit Card	Address	DOB	Points	Age Group
C1	Jill	4034	Sydney	3-Jun-1976	387	
C2	John	3323	Perth	12-Feb-1987	231	
	Brianna	1011	Hobart		345	
	John		Perth		198	31-35



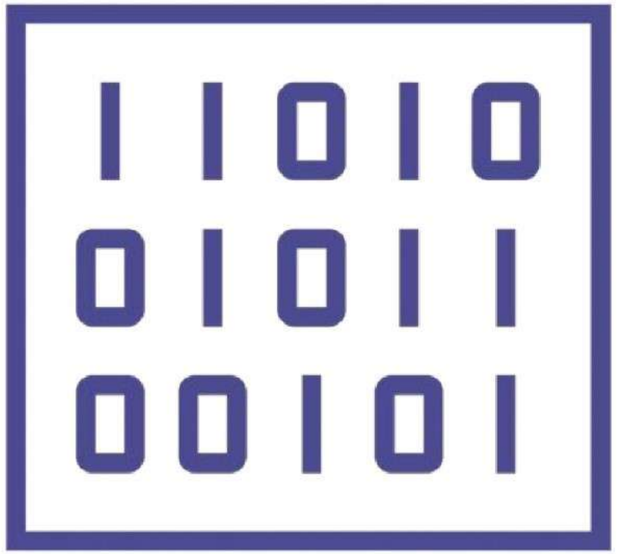
Do these refer to the same person?

Customer Profile

User ID	Name	Credit Card	Address	DOB	Points	Age Group
C1	Jill	4034	Sydney	3-Jun-1976	387	
C2	John	3323	Perth	12-Feb-1987	231	
	Brianna	1011	Hobart		345	
	John		Perth		198	31-35

This data can still be analyzed

Unstructured Data



May originate from multiple sources

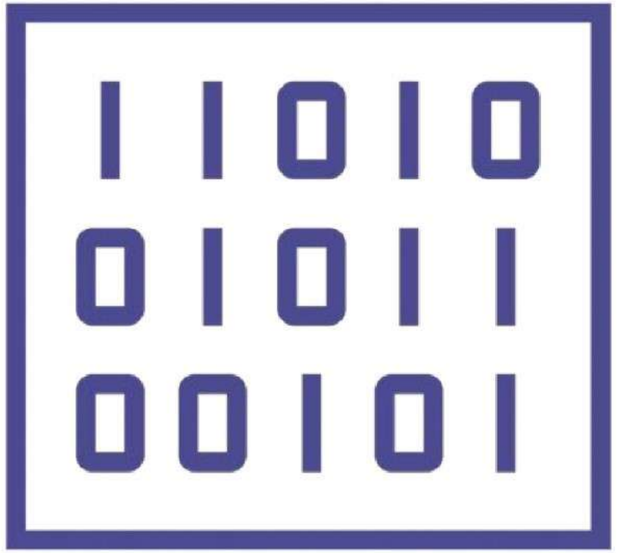
Different attributes available for same types of entities

Data types for attributes may vary

Data may be partially structured

- Some fields are always available

Unstructured Data



Relational databases are not well-suited to unstructured data

- May strictly enforce schemas

Some NoSQL databases are designed to handle this lack of structure

- E.g. document databases

An Overview of Big Data

Big Data

A field which deals with the analysis of large and complex datasets in order to extract meaningful information.

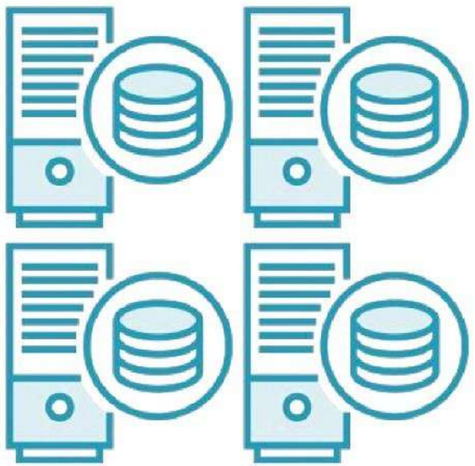
Big Data

A field which deals with the analysis of large and complex datasets in order to extract meaningful information.

Big Data

A field which deals with the analysis of large and complex datasets in order to extract meaningful information.

3 Vs of Big Data

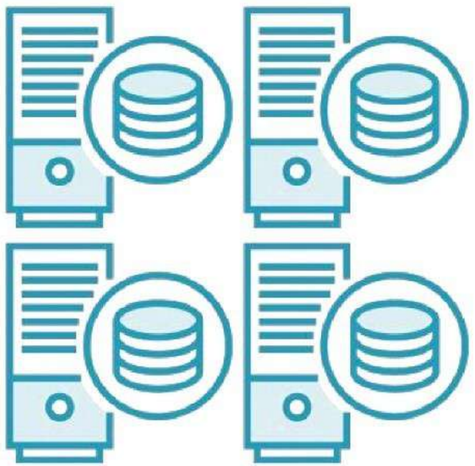


Volume: Amount of data

Variety: Number and type of sources

Velocity: Batch and streaming

Properties of Big Data Systems



Data distributed on a cluster with multiple machines

Semi-structured or unstructured data

No random access to data

Data replicated, propagation of updates take time

Different sources may have different unknown formats

Database Use Cases

Data Storage

Data Security

Transactions

Data Analysis

Transactional and Analytical Processing

Transactional Processing

Ensure correctness of individual entries

Access to recent data, from the last few hours or days

Updates data

Fast real-time access

Usually a single data source

Analytical Processing

Analyzes large batches of data

Access to older data going back months, or even years

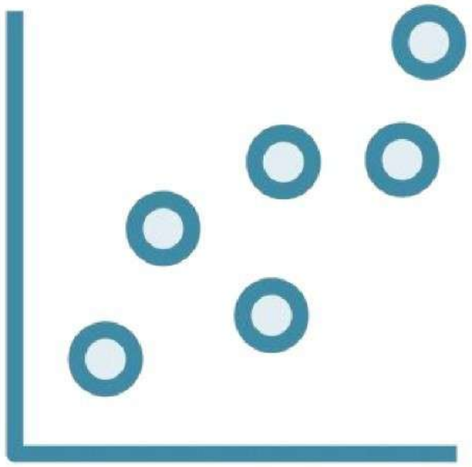
Mostly reads data

Long running jobs

Multiple data sources

Extract Transform
ETL → Load

Data Analysis and Big Data



Collect data → Extract

- In large volumes
- From multiple sources

Clean and process data } → Transform

Explore and analyze data

Extract useful information from data Support decision-making → Load

NoSQL databases are

more

suitable for Big Data

processing than RDBMS

NoSQL Databases and Big Data

5 V's → Velocity, Variety, Volume, Value, Veracity → Cassandra

Features of NoSQL Databases

Variety.
Semi-structured Data

Volume
Large Datasets

High Availability

Analytical Queries

Velocity.
Real-time and Streaming

Caching and Prototyping

Features of NoSQL Databases

Semi-structured Data	Large Datasets	High Availability
Analytical Queries	Real-time and Streaming	Caching and Prototyping

These use cases map to 3 properties of big data

Features of NoSQL Databases

Variety	Volume	High Availability
Analytical Queries	Velocity	Caching and Prototyping

The 3 Vs of big data

Features of NoSQL Databases

Variety	Volume	High Availability
Analytical Queries	Velocity	Caching and Prototyping

This can be ensured with a distributed system

Features of NoSQL Databases

Variety	Volume	High Availability
Analytical Queries	Velocity	Caching and Prototyping

Queries meant to understand data in the aggregate

Features of NoSQL Databases

Variety	Volume	High Availability
Analytical Queries	Velocity	Caching and Prototyping

Contrasts with traditional use case for RDBMS

Relational vs. NoSQL Databases

Relational Databases

Vertical scaling - often not distributed systems

Normalized data - small, inter-related tables

Optimized for efficient storage - avoid redundancy, hierarchy, nesting

ACID

Atomicity Consistency - Isolation, Durability

NoSQL Databases

Horizontal scaling - are almost always distributed systems

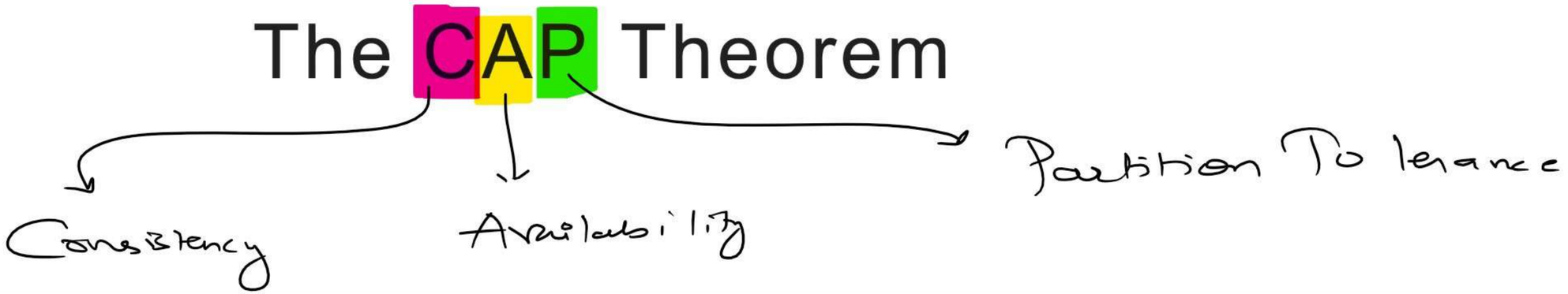
Denormalized data - large, self-contained collections

Optimized for efficient network access use hierarchy, redundancy, nesting

BASE

Basically Available Soft State Eventual Consistency.

Good, Cheap & Fast ? → Pick any 2 ?



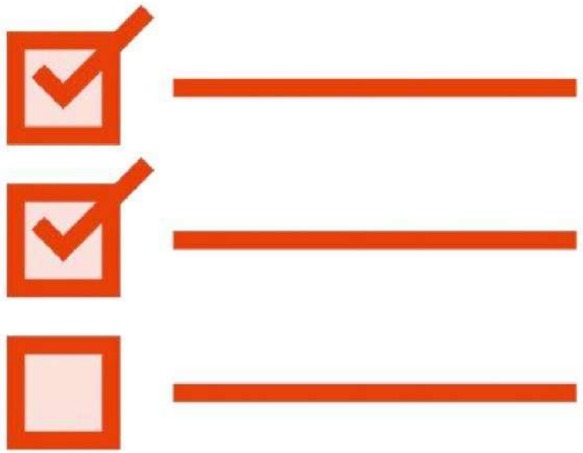
CAP Theorem

A distributed database must choose
2 out of the 3 CAP guarantees:

- Consistency
- Availability
- Partition Tolerance

The CAP Guarantees

CosmoDB → {bid allotment}



Consistency: Every read will receive most recent write, else error

Availability: Every request will receive a non-error response

Partition Tolerance: Network failures between nodes will be "managed gracefully" gracefully"

C → Hard Working

A → Loyal

P → Alive

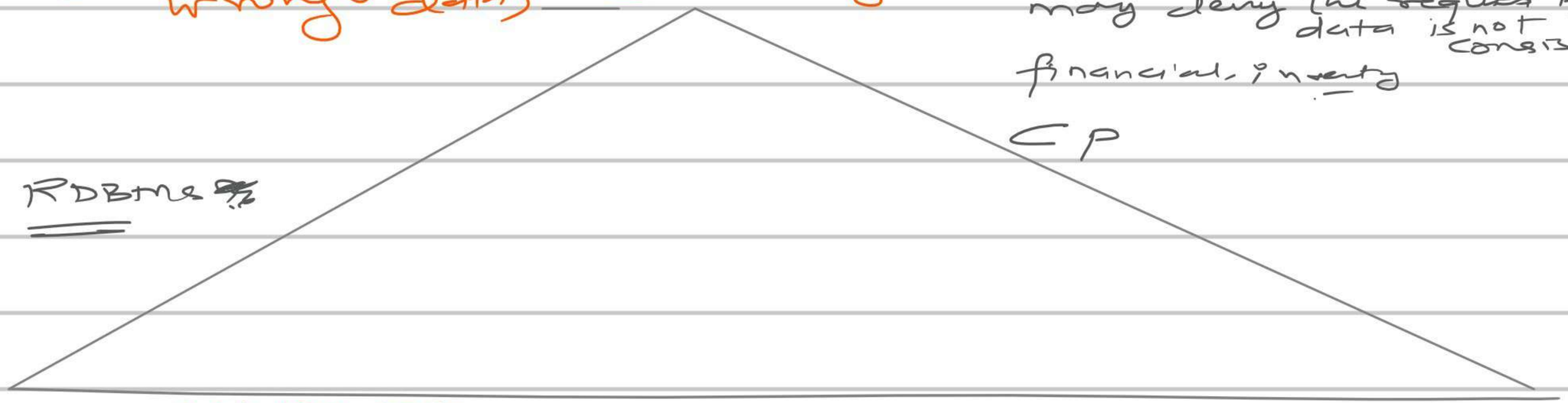
CP (Better delay Than wrong data) — Consistency → data accuracy may deny the request if data is not consistent

financial, inventory

AP

CP

CA → RDBMS



Availability (Always reply, even if confusing)

AP

Partition Tolerance.

Prioritize uptime, even if mean to show stale data

social media logs, metrics

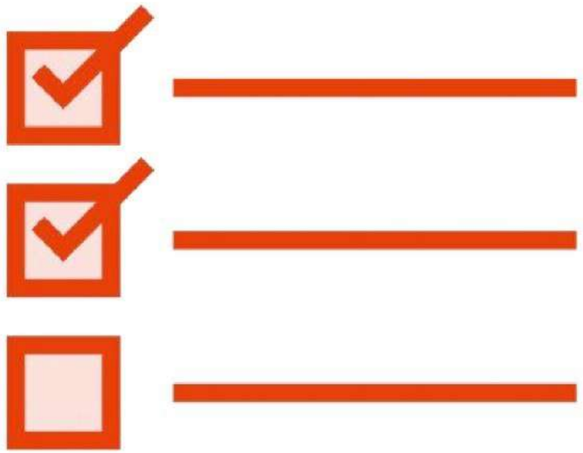
Flight Ticket —

P

C P | #Base
Mongo DB
Cassandra (strong consistency)

A P | Cassandra (default eventual consistency)
DynamoDB
CouchDB

Partition Tolerance



When network failures partition the distributed system, it must either:

- Cancel the operation
  - Compromises availability but maintains consistency
- Allow the operation
  - Compromises consistency but maintain availability

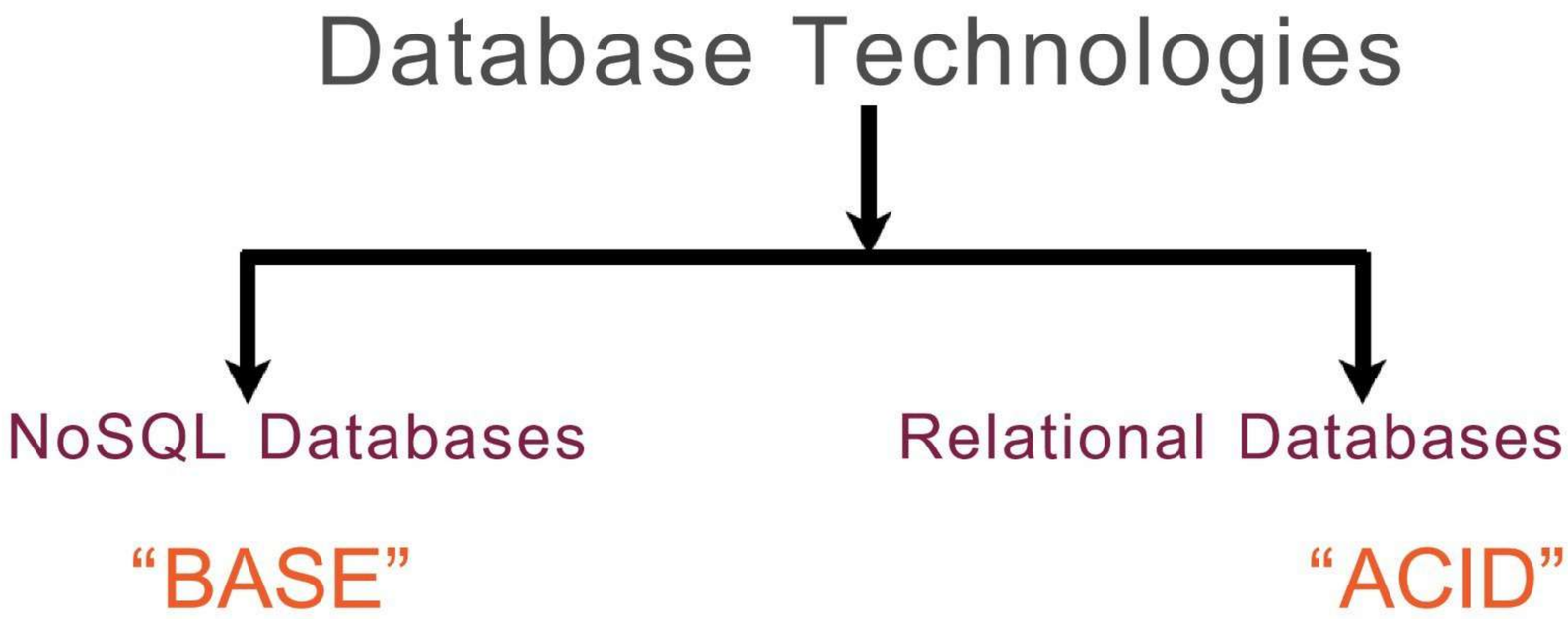
The CAP Theorem

Not possible for a distributed database to achieve all 3 CAP guarantees

Big data technologies are invariably distributed database systems

- Horizontal scaling

Hence must choose what CAP guarantees to provide


Database Technologies
NoSQL Databases
Relational Databases
“BASE”
“ACID”

BASE vs. ACID

NoSQL Databases

Choose availability over consistency

BASE: *B*asically *A*vailable, *S*oft state, *E*ventual consistency

Writes are faster - don't wait for system to become fully consistent before returning

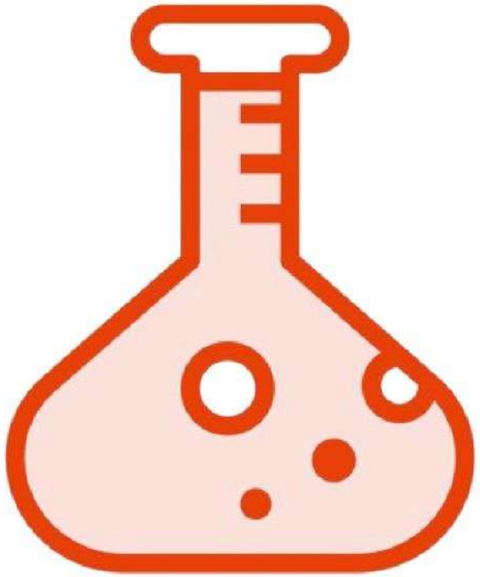
Relational Databases

Choose consistency over availability

ACID: *A*tomicity, (strong) *C*onsistency, *I*solation, *D*urability

Writes are slower - wait for full consistency of system before returning

The BASE Philosophy

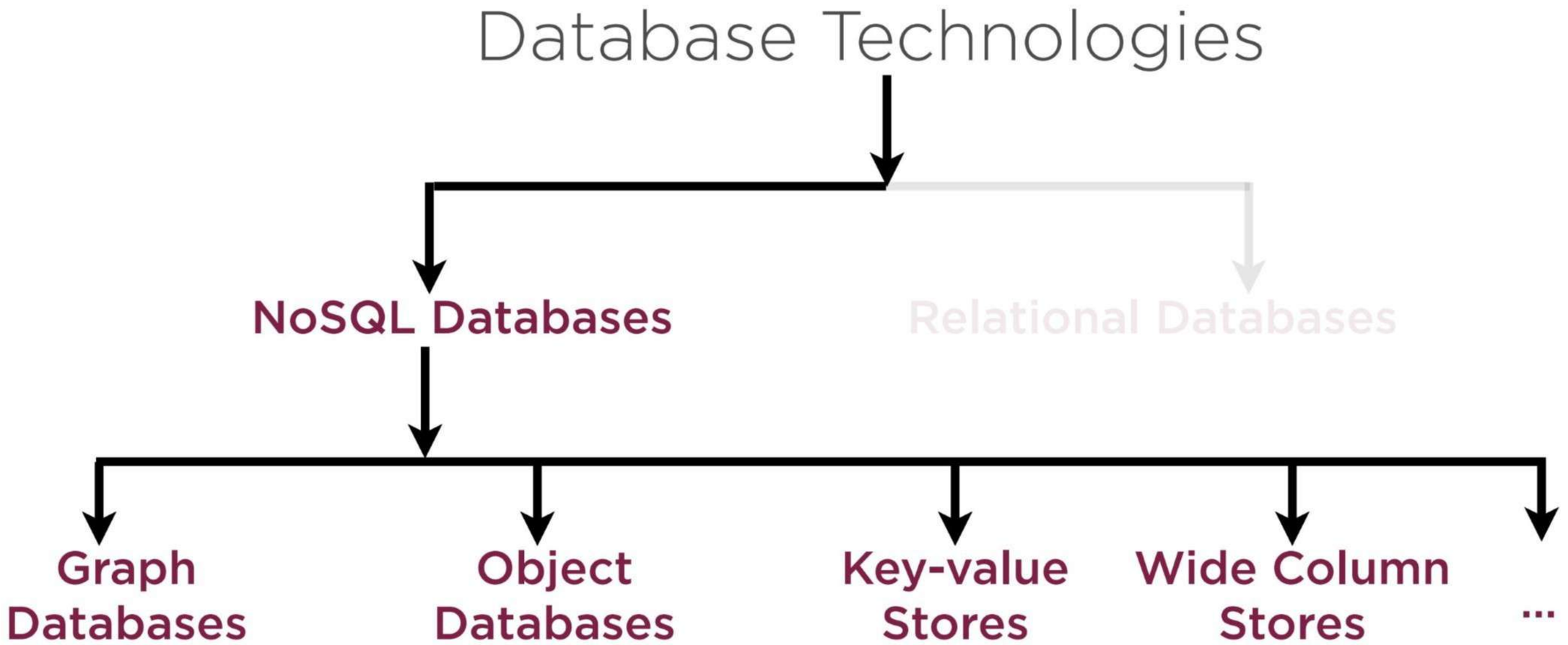


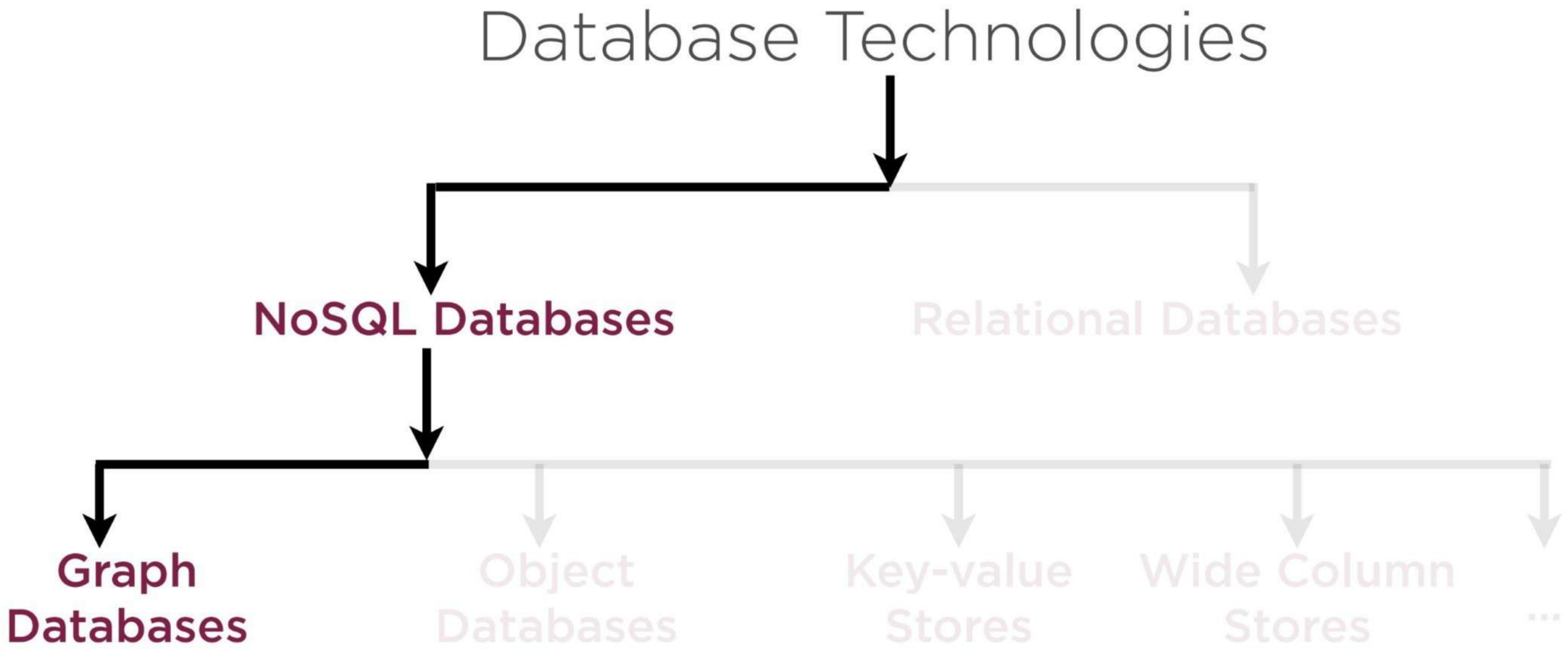
*B*asically *A*vailable: System will stay "up" using replication/sharding

*S*oft state: No guarantee that data is consistent - reads can yield stale data

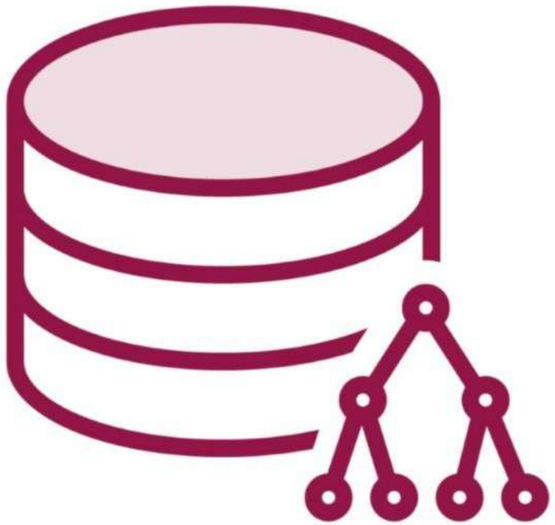
*E*ventual consistency: If we wait long enough, reads will yield consistent data

- No guarantee on how long that is


Database Technologies
NoSQL Databases
Relational Databases
Graph Databases
Object Databases
Key-value Stores
Wide Column Stores
...


Database Technologies
NoSQL Databases
Relational Databases
Graph Databases
Object Databases
Key-value Stores
Wide Column Stores
...

Graph Databases



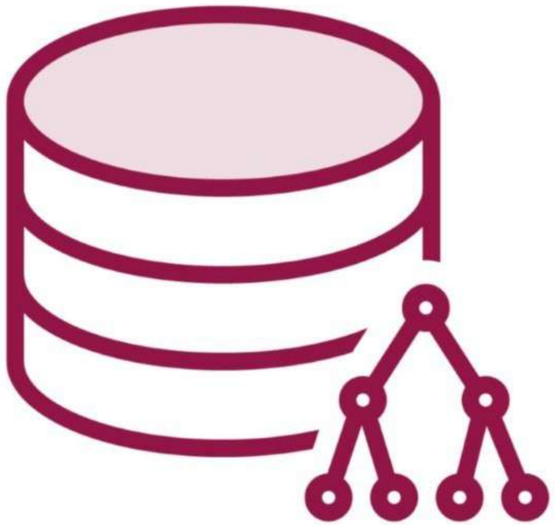
Data organized into graphs

Emphasize relationships over entities

Nodes and edges

- Nodes for entities
- Edges for relationships

Graph Databases

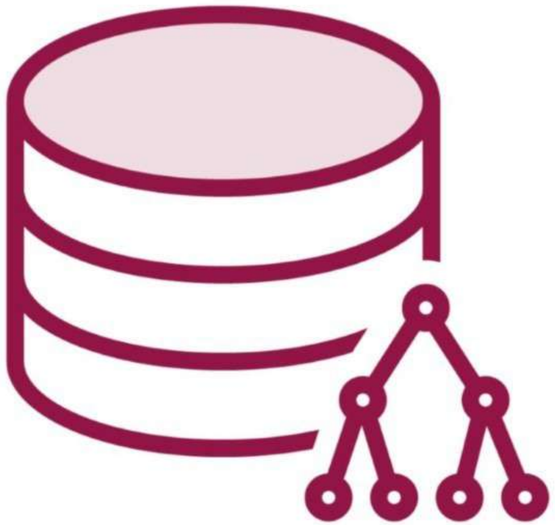


Support for semantic queries

- Query based on associations, context

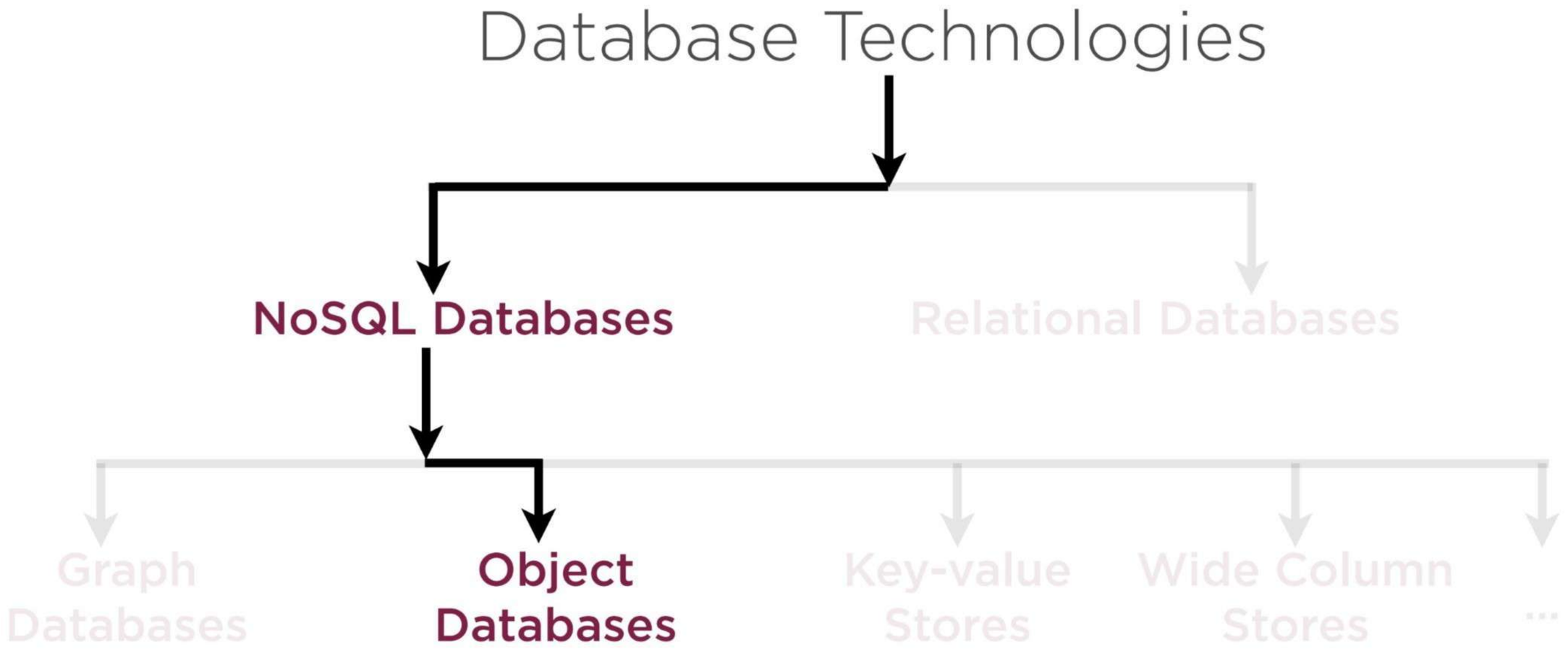
Quickly retrieve complex hierarchical structures

Graph Databases

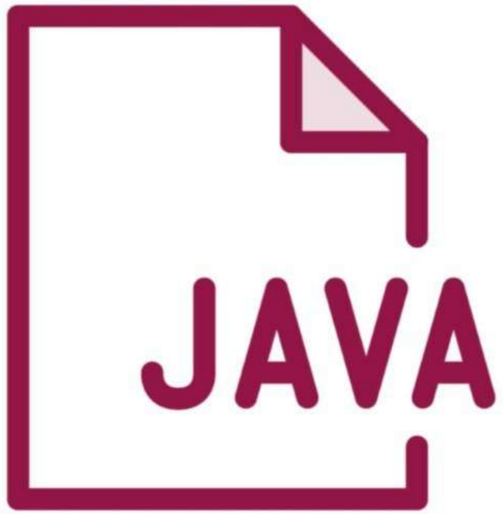


No standard query language

- SQL is ill-suited to graph databases
- Major barrier to adoption


Database Technologies
NoSQL Databases
Relational Databases
Graph Databases
Object Databases
Key-value Stores
Wide Column Stores
...

Object Databases



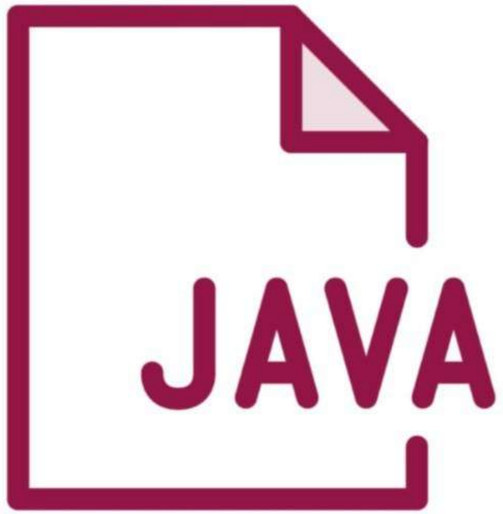
Programming languages usually model data using classes, objects

Relational databases model data using tables, rows

“Object-Relational Impedance Mismatch”

Object Databases attempt to solve this

Object Databases

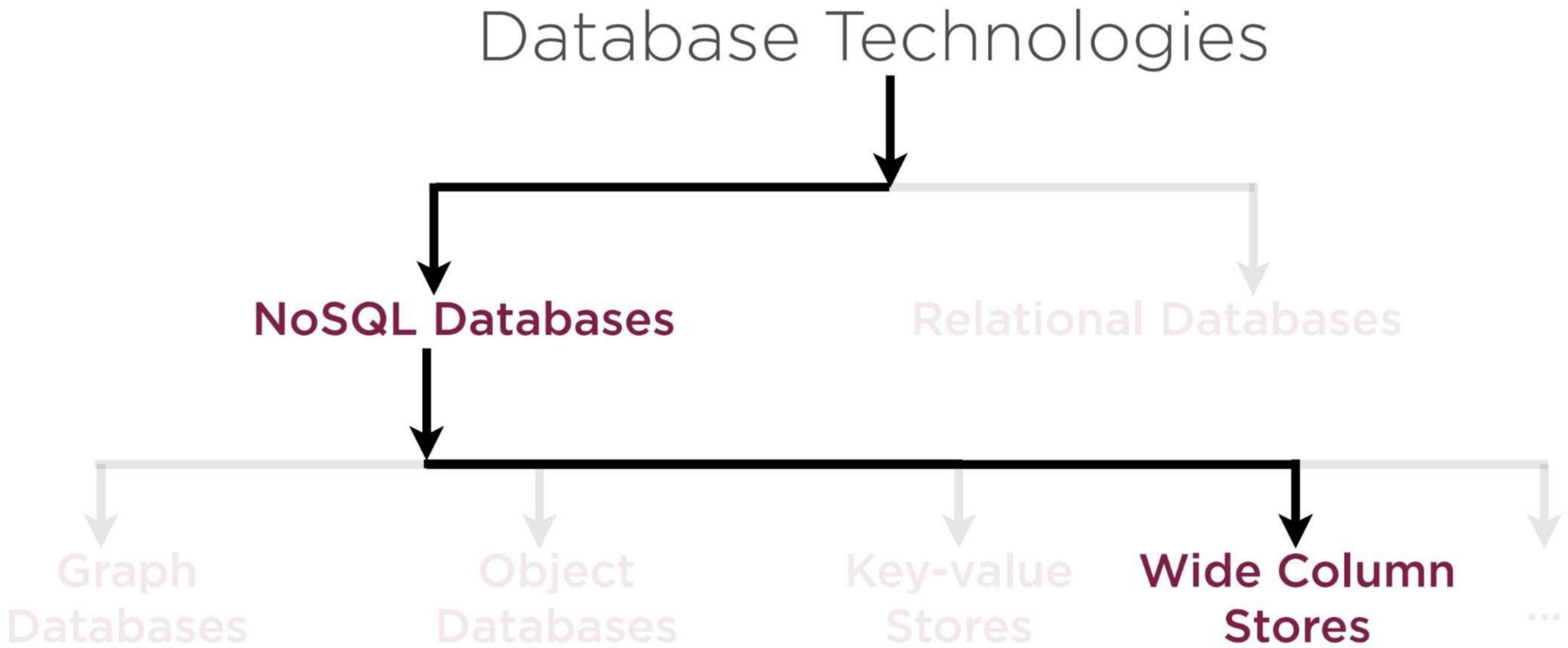


No standard language

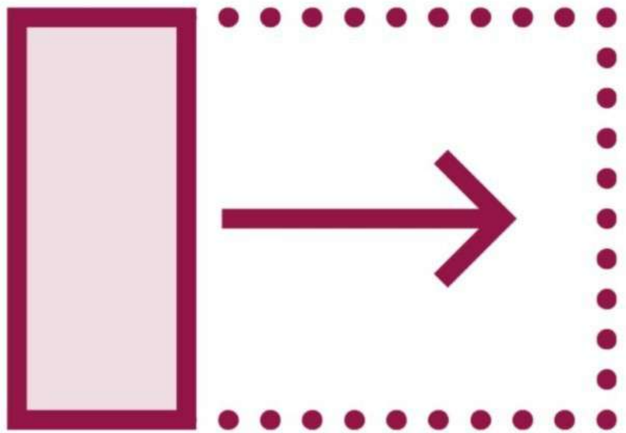
- SQL ill-suited to object databases
- Major barrier to adoption

Related to ORM frameworks

- Hibernate
- JPA


Database Technologies
NoSQL Databases
Relational Databases
Graph Databases
Object Databases
Key-value Stores
Wide Column Stores
...

Wide Column Databases



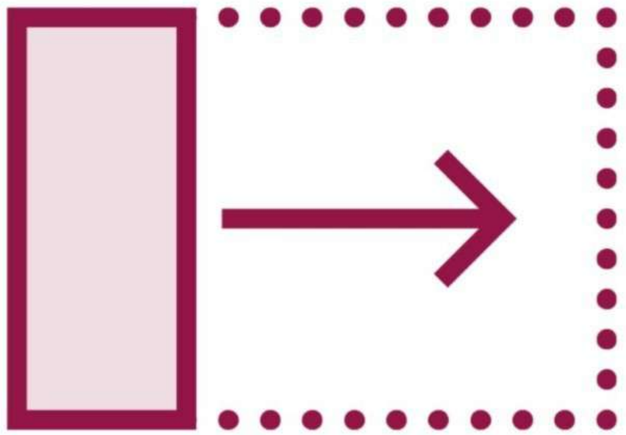
Relational databases feature fixed schemas

Altering schemas to add/remove columns is onerous

NULL values occupy significant space

Wide Column databases address these weaknesses

Wide Column Databases



Several wide column databases have achieved widespread popularity

- HBase
- Cassandra

Syntax closer to SQL

- Has helped drive adoption

Relational Data Model: Wide Tables

Id	To	Type	Content
1	mike	offer	Mobile offer
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

As columns are added, table gets wider

Relational Data Model: Wide Tables

Id	To	Type	Content	Value
1	mike	offer	Mobile offer	NULL
2	john	sale	Redmi sale	NULL
3	jill	order	Order delivered	NULL
4	megan	sale	Clothes sale	$53

Addition of columns can lead to several null values

From Wide To Long

Id	To	Type	Content	Value
1	mike	offer	Mobile offer	NULL
2	john	sale	Redmi sale	NULL
3	jill	order	Order delivered	NULL
4	megan	sale	Clothes sale	$53

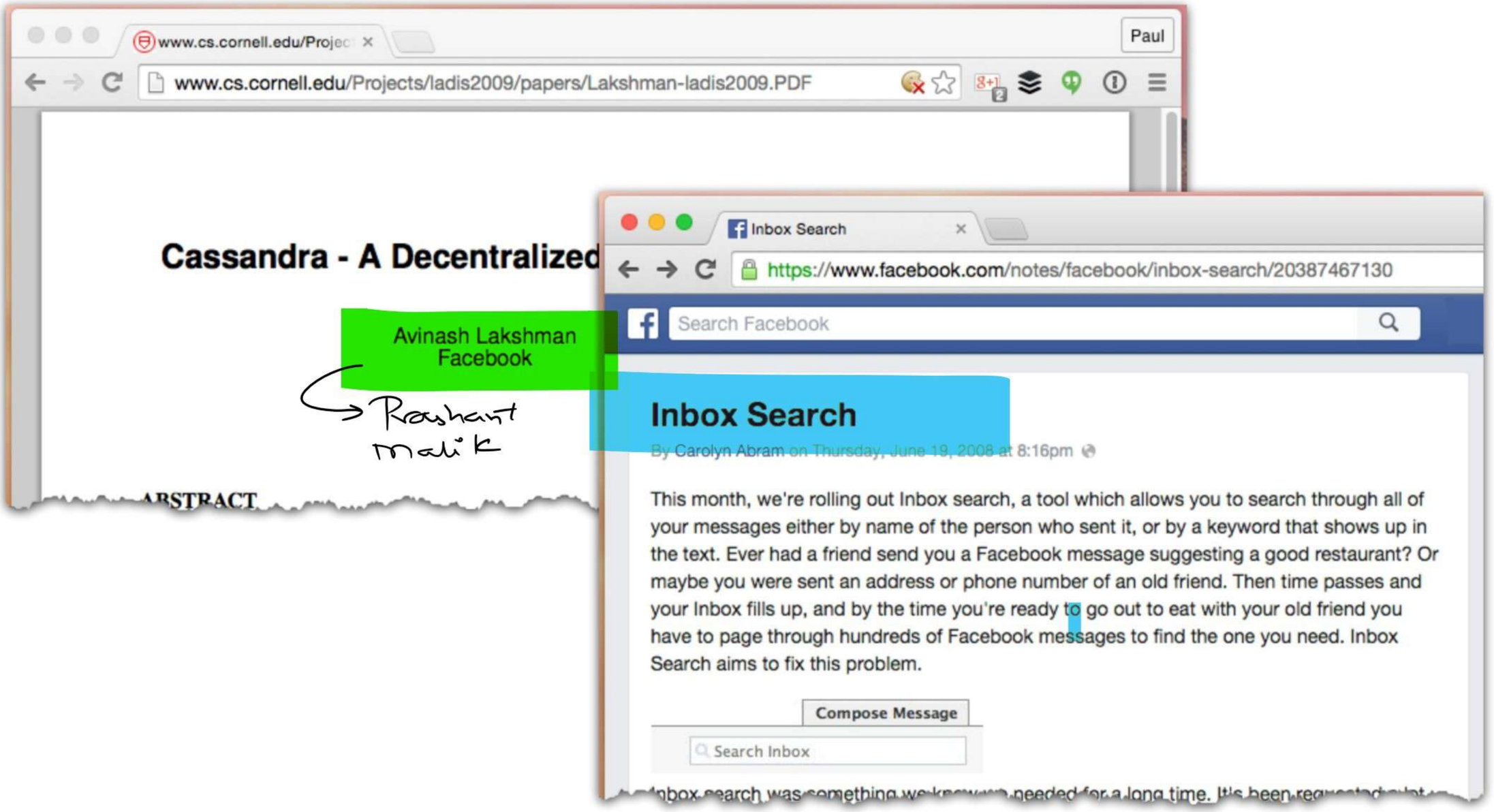
Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Mobile offer
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale
4	Value	$53

From Wide To Long

Id	To	Type	Content	Value
1	mike	offer	Mobile offer	NULL
2	john	sale	Redmi sale	NULL
3	jill	order	Order delivered	NULL
4	megan	sale	Clothes sale	$53

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Mobile offer
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale
4	Value	$53

Introduction to Cassandra

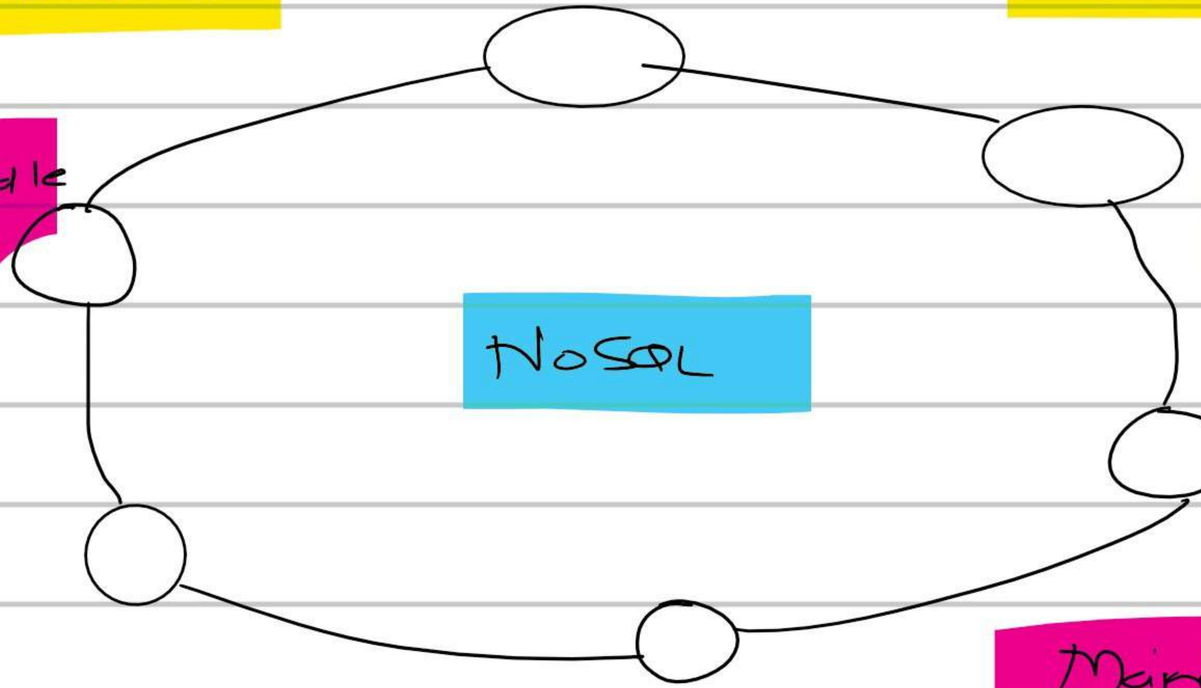

www.cs.cornell.edu/Projec
Paul
www.cs.cornell.edu/Projects/ladis2009/papers/Lakshman-ladis2009.PDF
Cassandra - A Decentralized
Avinash Lakshman
Facebook
Prashant
Malik
ABSTRACT
Inbox Search
https://www.facebook.com/notes/facebook/inbox-search/20387467130
Search Facebook
Inbox Search
By Carolyn Abram on Thursday, June 19, 2008 at 8:16pm
This month, we're rolling out Inbox search, a tool which allows you to search through all of your messages either by name of the person who sent it, or by a keyword that shows up in the text. Ever had a friend send you a Facebook message suggesting a good restaurant? Or maybe you were sent an address or phone number of an old friend. Then time passes and your Inbox fills up, and by the time you're ready to go out to eat with your old friend you have to page through hundreds of Facebook messages to find the one you need. Inbox Search aims to fix this problem.
Compose Message
Search Inbox

Apache Cassandra

open source, distributed database,

designed to handle large amount of data across commodity servers.

→ Highly Available



No SPOF
Single Point of Failure

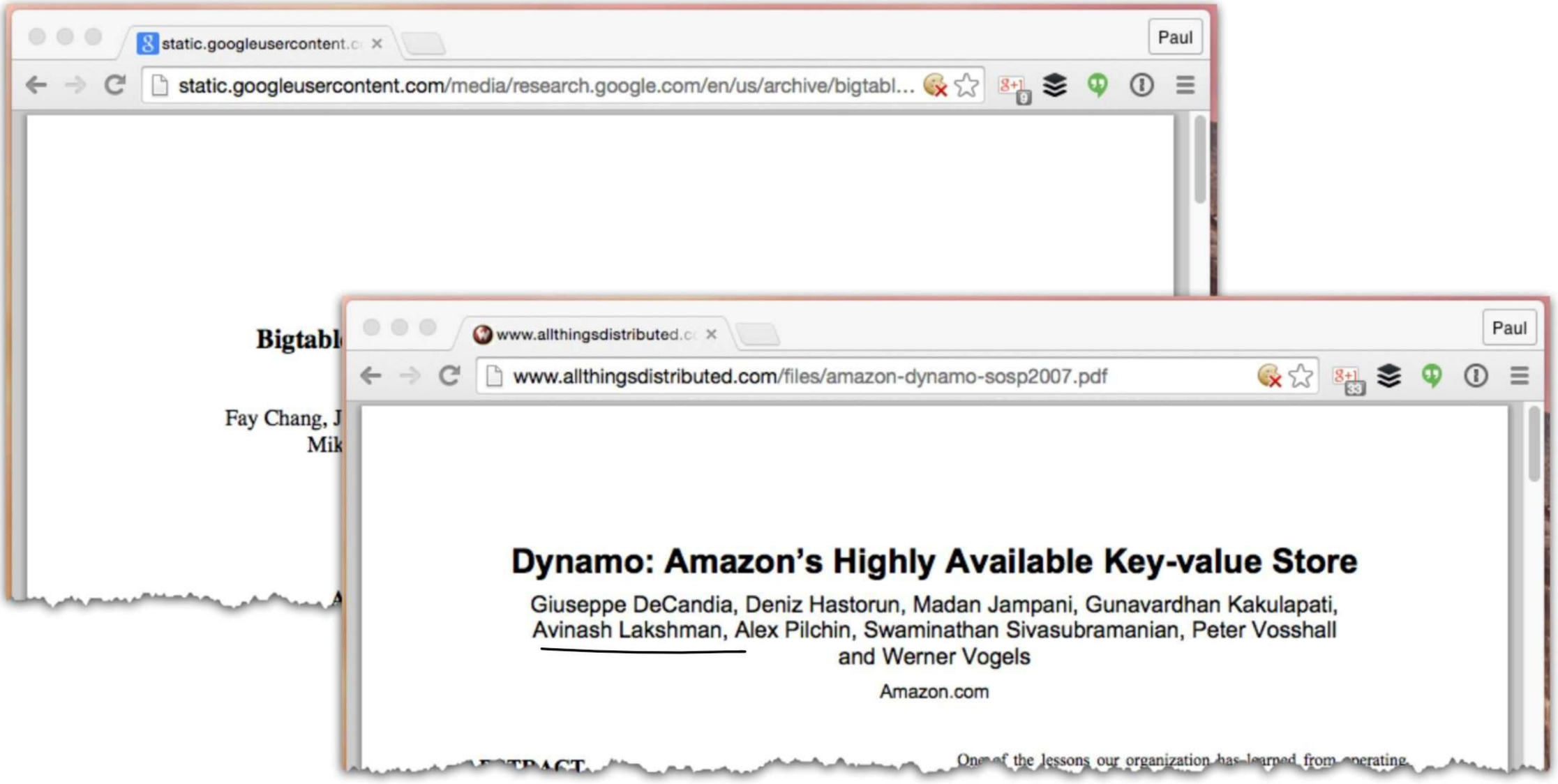
NoSQL

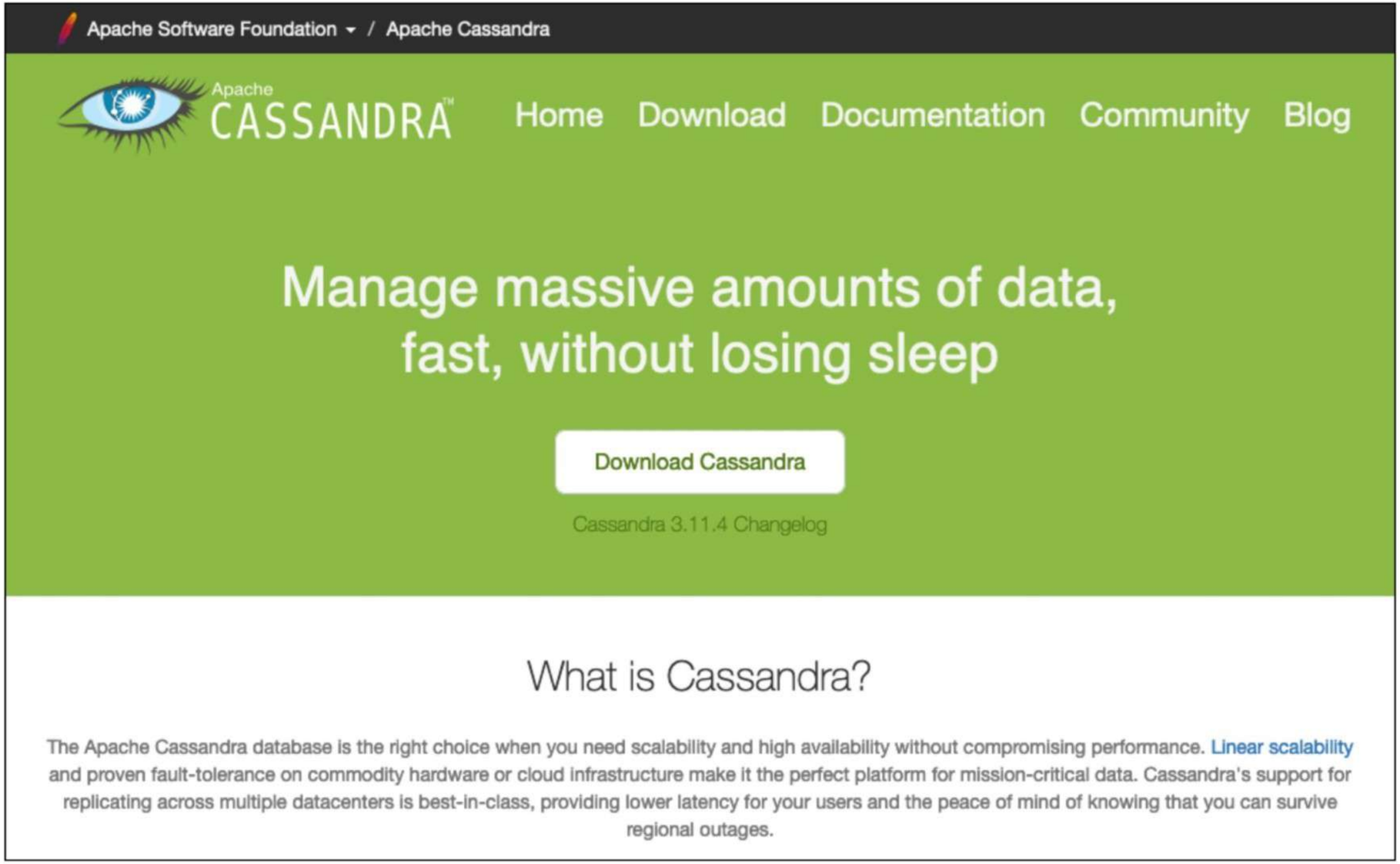
Peer to Peer Architecture

Main focus → Performance

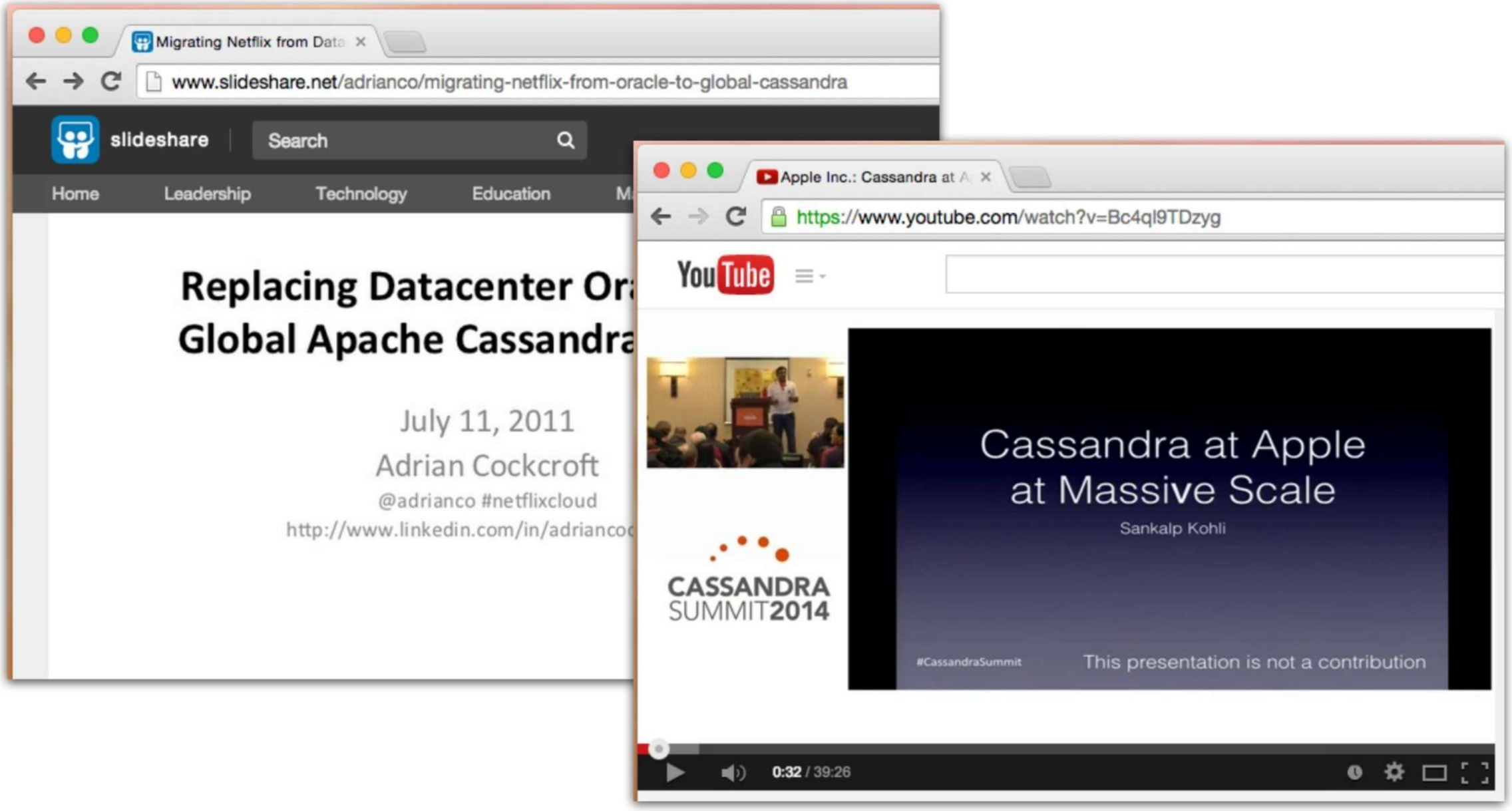
Cluster based

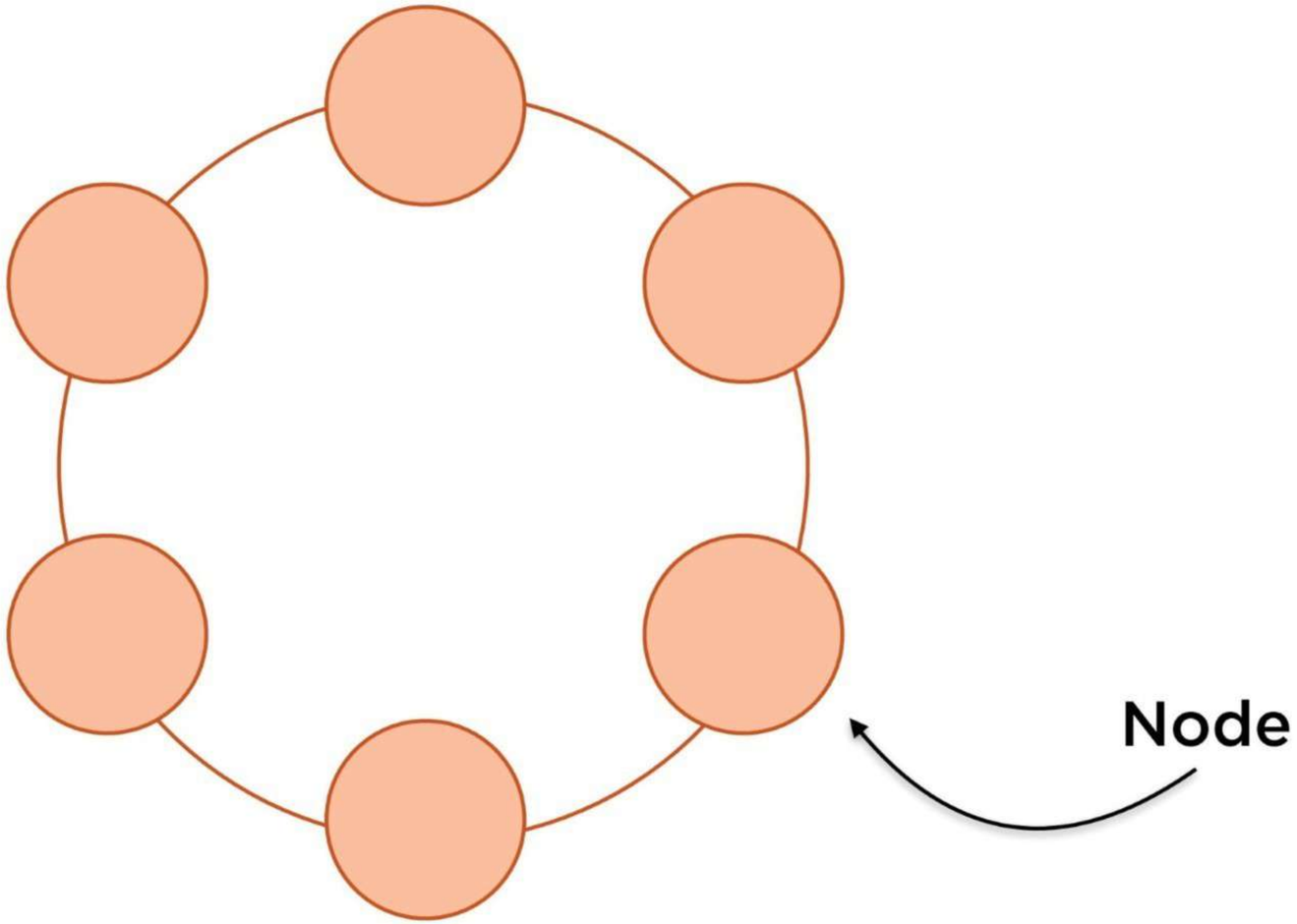
commodity reliable


static.googleusercontent.com/media/research.google.com/en/us/archive/bigtabl...
Paul
Bigtabl
Fay Chang, J
Mik
www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf
Paul
Dynamo: Amazon's Highly Available Key-value Store
Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,
Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall
and Werner Vogels
Amazon.com

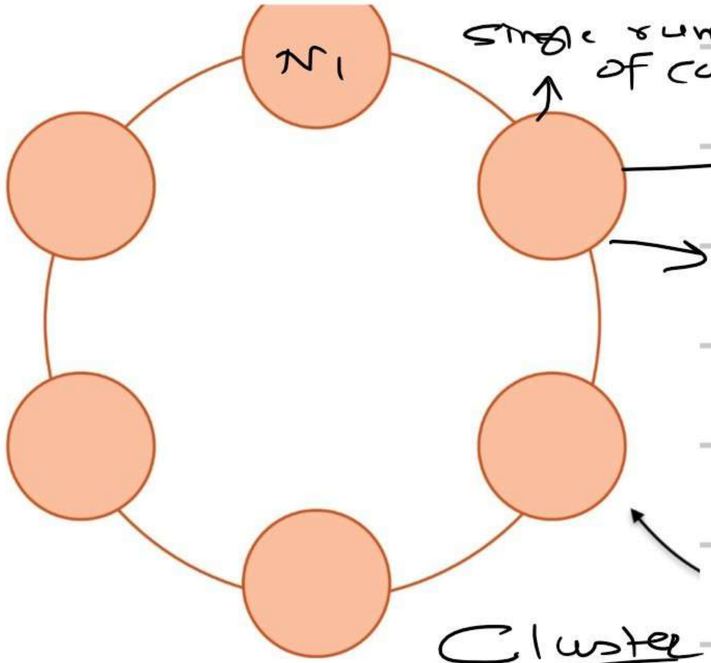


https://cassandra.apache.org/


Migrating Netflix from Data
www.slideshare.net/adrianco/migrating-netflix-from-oracle-to-global-cassandra
slideshare
Search
Home
Leadership
Technology
Education
Replacing Datacenter Ora
Global Apache Cassandra
July 11, 2011
Adrian Cockcroft
@adrianco #netflixcloud
http://www.linkedin.com/in/adriancoc
Apple Inc.: Cassandra at A
https://www.youtube.com/watch?v=Bc4ql9TDzyg
YouTube
CASSANDRA
SUMMIT2014
Cassandra at Apple
at Massive Scale
Sankalp Kohli
#CassandraSummit
This presentation is not a contribution
0:32 / 39:26


Node

Architecture of Cassandra



Single running instance of cassandra

building block of cassandra

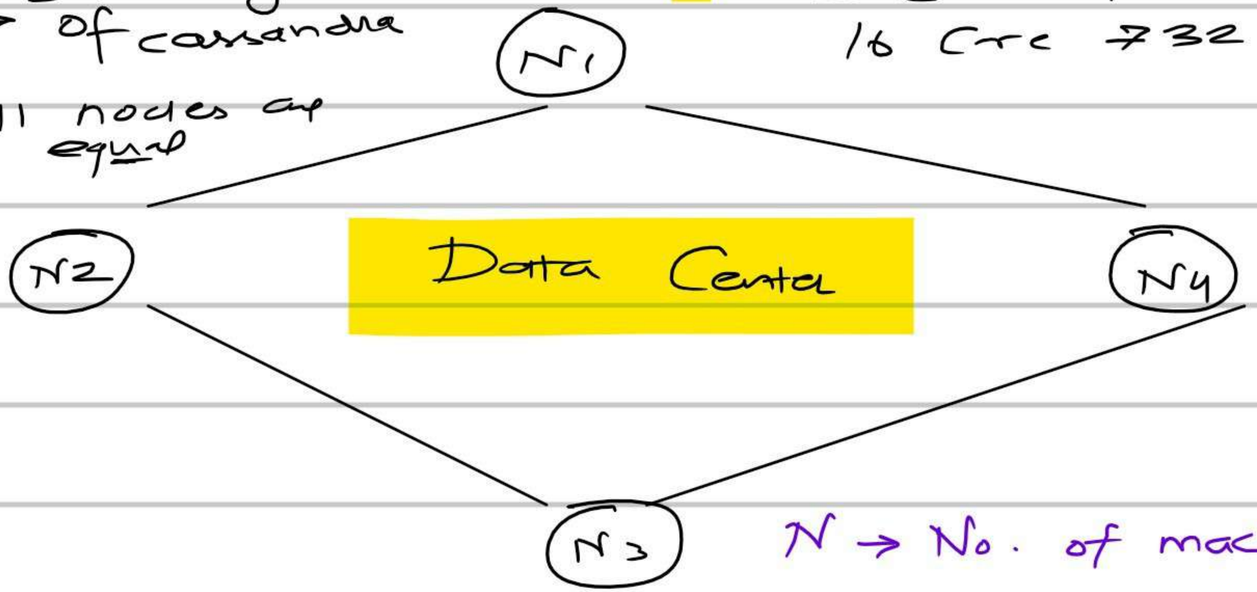
All nodes are equal

Cluster

16 Core Machine

1 Core → 2 I/O channels

16 Core → 32 I/O channels

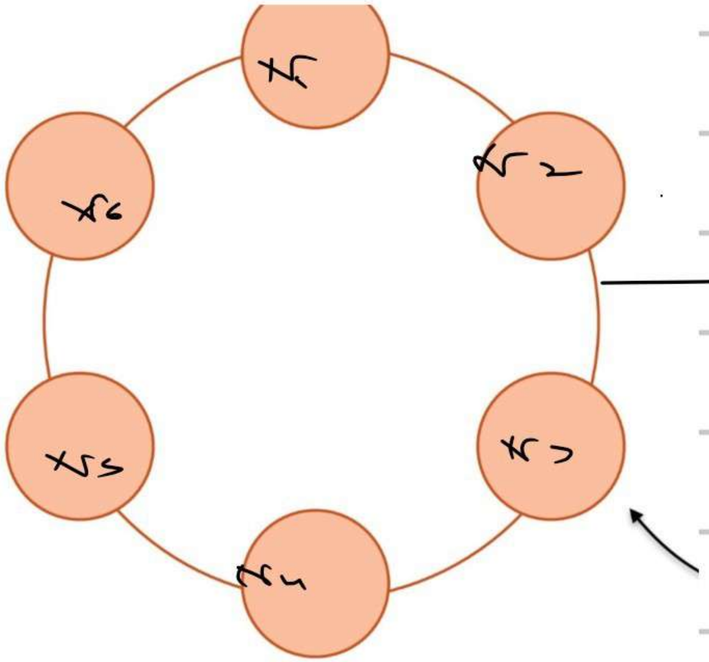


N → No. of machines

m → No. of i/o channels

4 machines x 32 I/O channel

Users = N x m

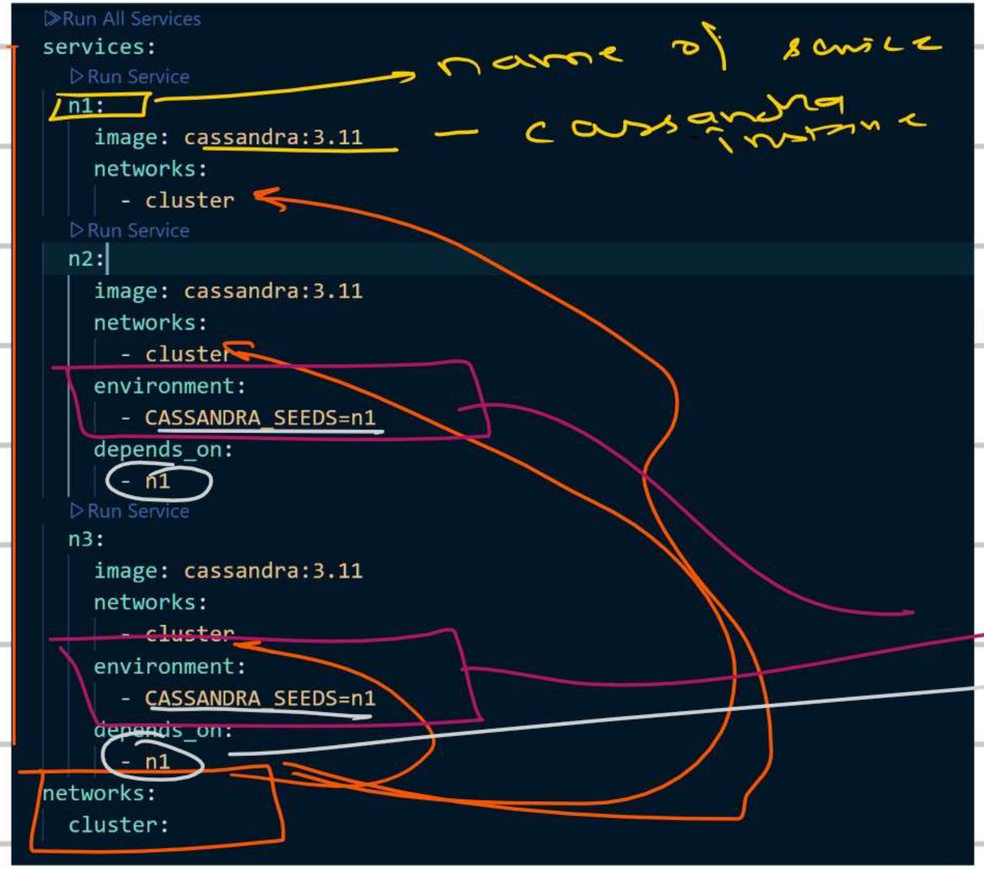


Cluster → spanner

→ collection of interconnected nodes

→ appear one single logical db to client

Cluster



name of service

— cassandra instance

n1

Cassandra — n2

n3

giving ref of n1

n2 & n3 won't start until n1 is up & running

How is data structured in Cassandra

Partition → set of rows which share the same partition key

Query First Approach

Before we design the database; we should think of queries & should design the database accordingly.

→ Data should be evenly spread across the cluster

Query → Structure

→ Minimize the partition read

No limit on storage

Accomodate the data in such a way that the query can be easily accomodated

Primary Key in Cassandra

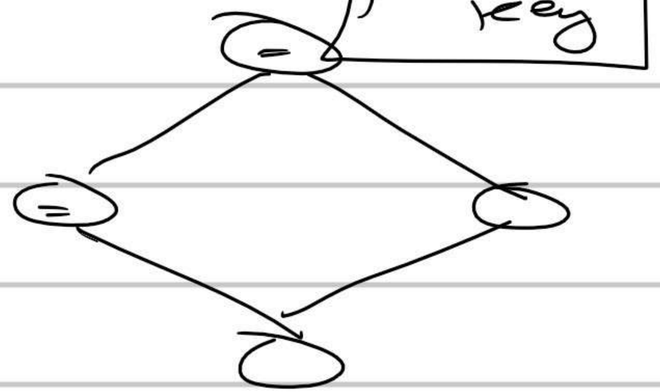
Primary Key → Partitioning Key + Clustering Key

① Single Primary key [Partition Key]

② Composite Primary Key [Partition key, Clustering key]
[Partition key, CK, CK]

③ Composite Partition Primary key [(PK, PK), CK]

While reading data is always used using partition key



Clustering Key

Partitioning key.

Employee_by_car_make

ID	Name	Age	Gender	Salary	ID	Make	Model
1	Tom	23	Male	10000	1	BMW	5 Series
2	Alex	32	Male	12000	3	Kia	Seltos
3	Mike	31	Male	11000	4	Merc	S Class
4	Sophia	30	Female	13000	2	Audi	A7
5	Penny	34	Female	12000	1	BMW	5 Series
6	Leanord	40	Male	10000	4	Merc	S Class

Each node contains data from a single partition only

Data Types Cassandra

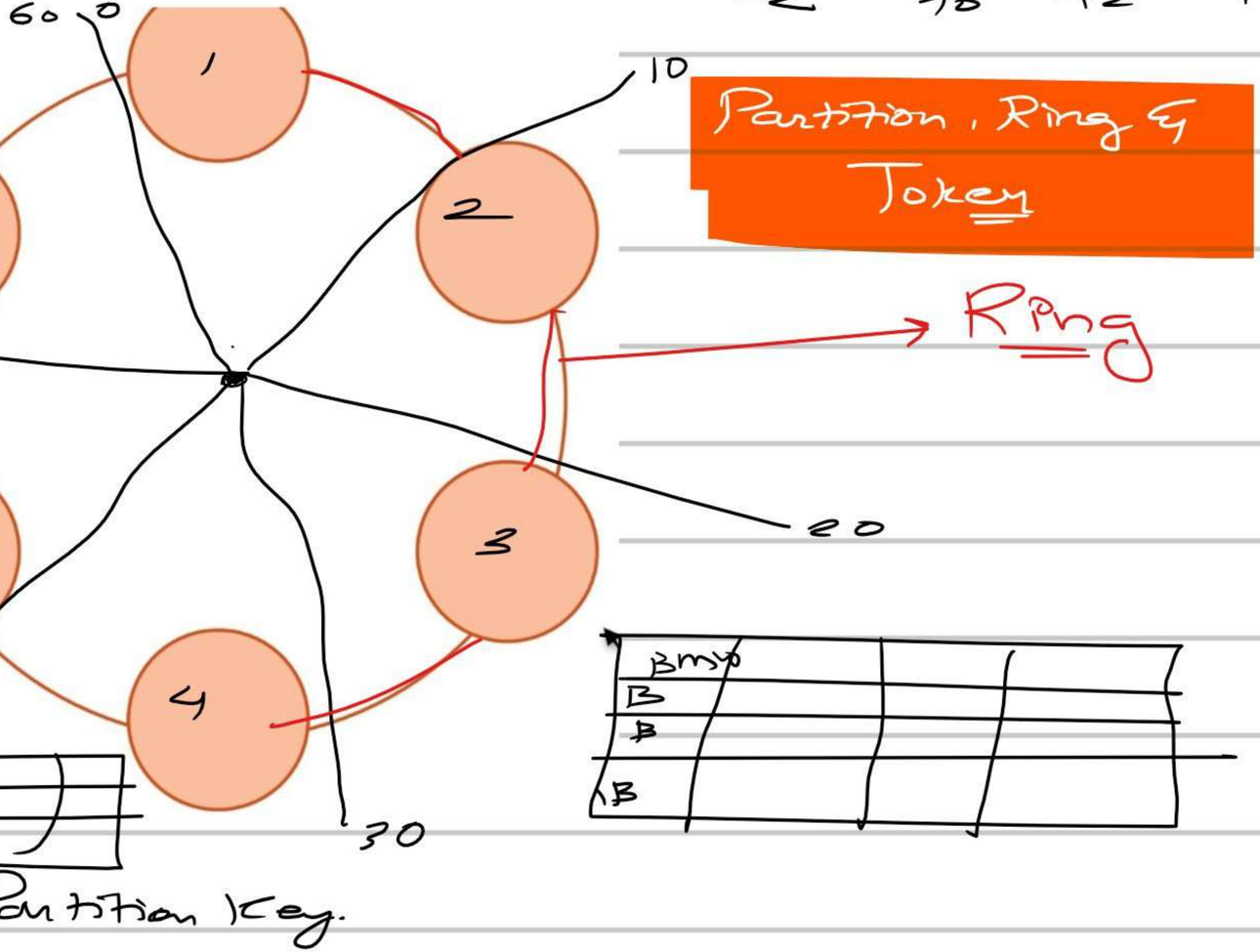
Basic → int, float, double, varchar, text, blob, bigint, boolean, decimal, timestamp

Collection → list, set & map

UD → User Defined

Nodes

↳ specify range of token



100,000 → 4

Partition key.

-2^{63} to $+2^{63}-1$

Partition, Ring & Token

Employee_by_car_make							
ID	Name	Age	Gender	Salary	ID	Make	Model
1	Tom	23	Male	10000	1	BMW	5 Series
2	Alex	32	Male	12000	3	Kia	Seltos
3	Mike	31	Male	11000	4	Merc	S Class
4	Sophia	30	Female	13000	2	Audi	A7
5	Penny	34	Female	12000	1	BMW	5 Series
6	Leanord	40	Male	10000	4	Merc	S Class

PK → Hash Function → Token value

BMW → HF → 23

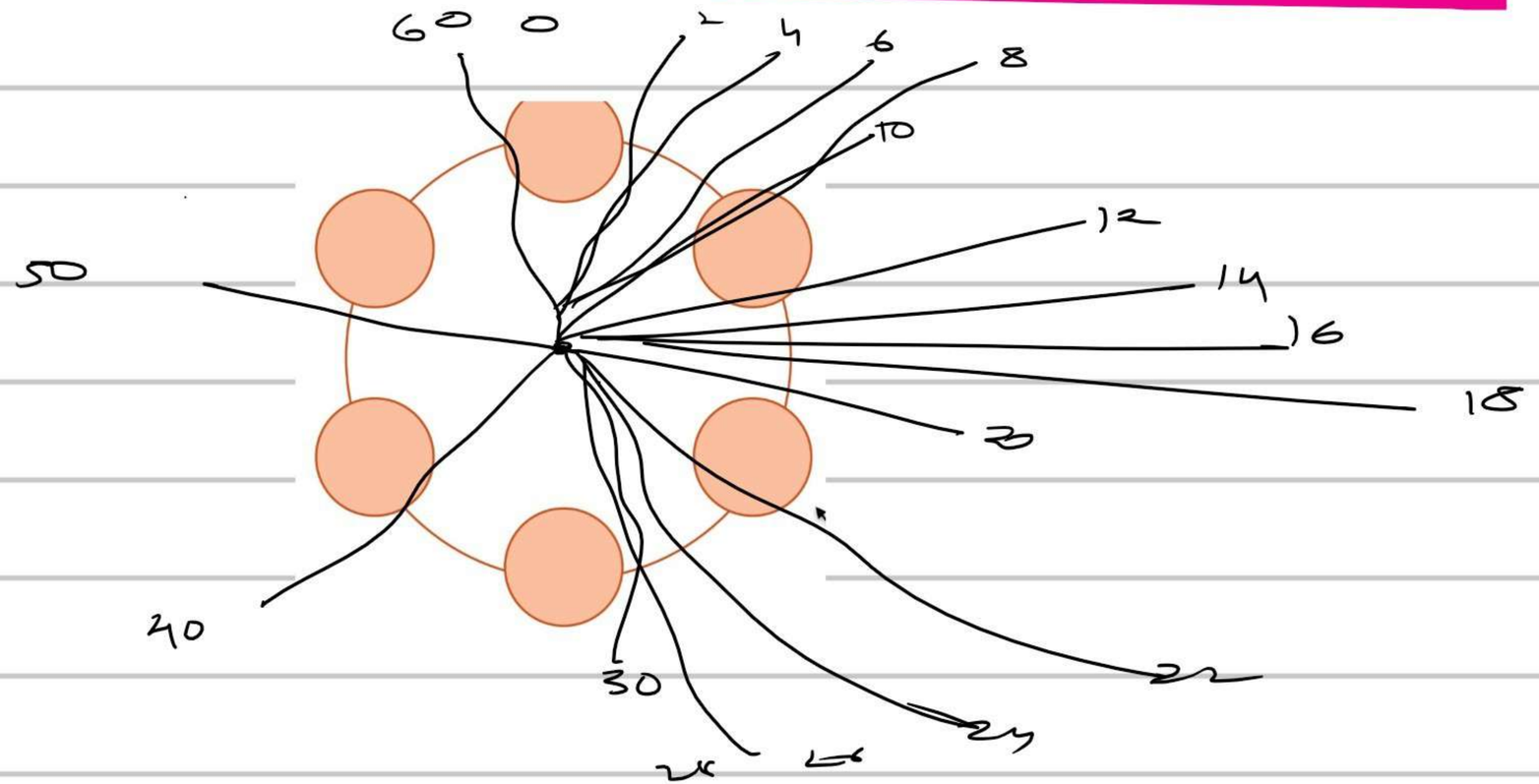
Audi → HF → 45

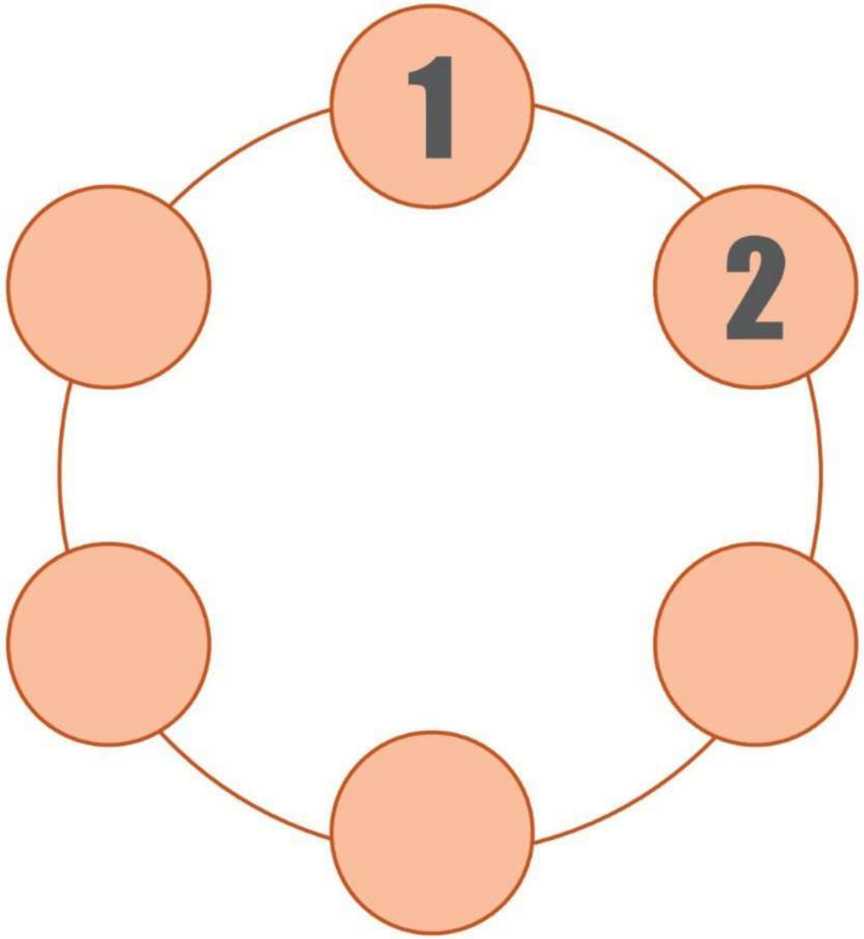
Kia → HF → 39

Merc → HF → 53

1 Node = 256 VNodes

Virtual Nodes



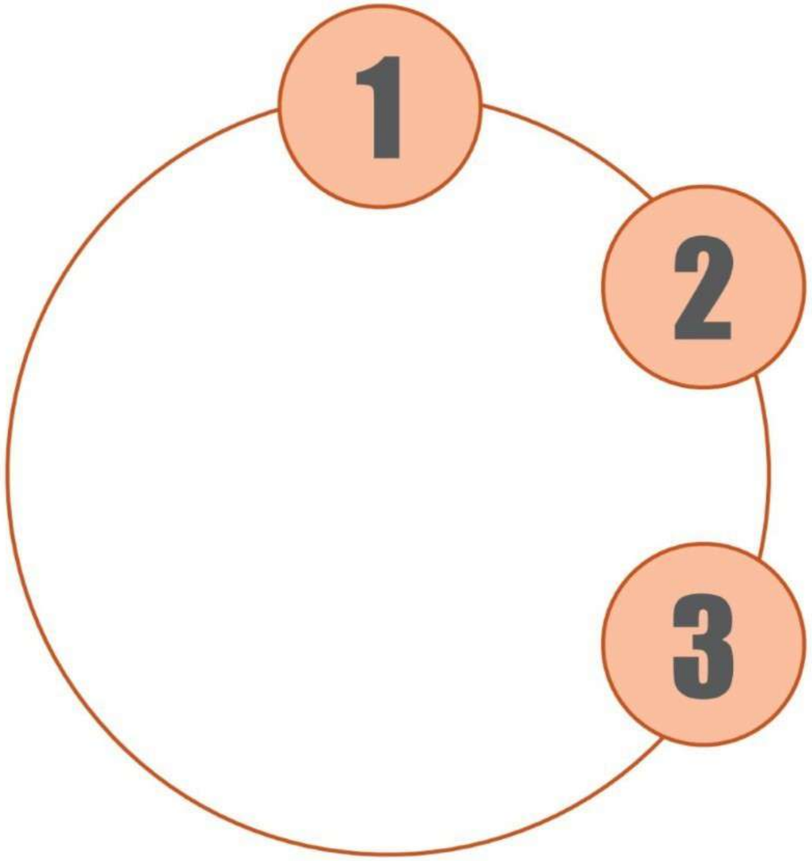


Storing Data in Cassandra

Range of all possible token values

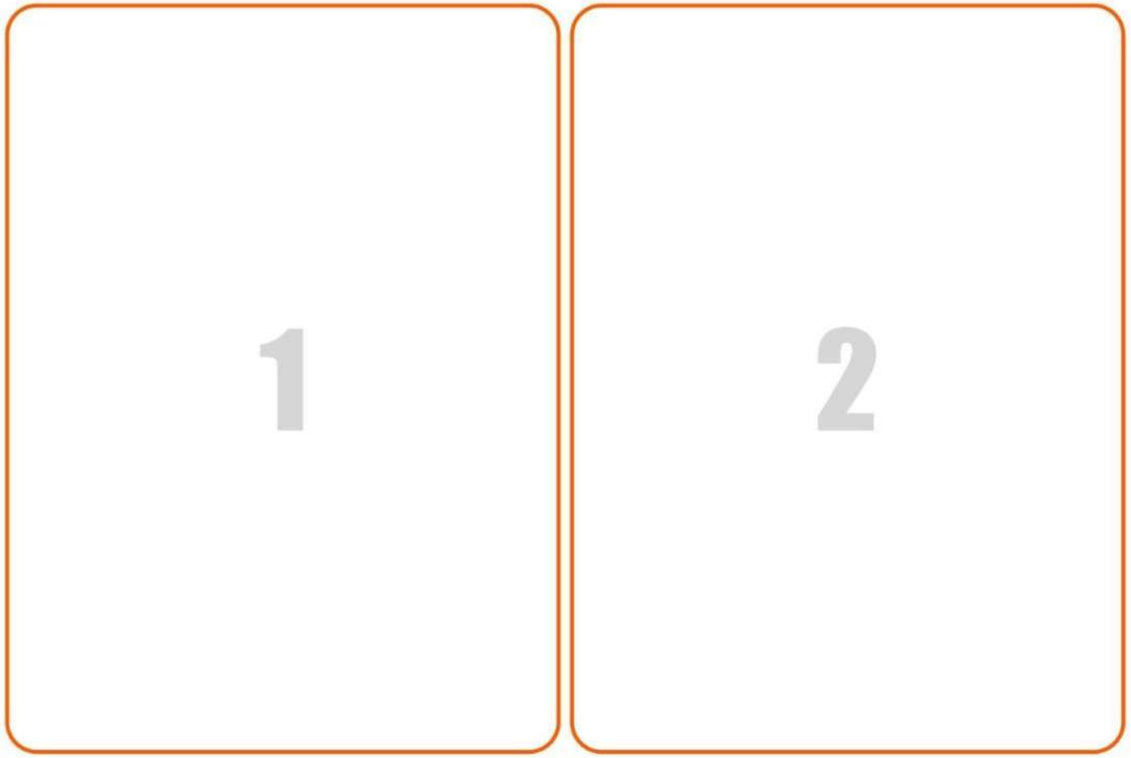
1

-2^{63} to $+2^{63}$

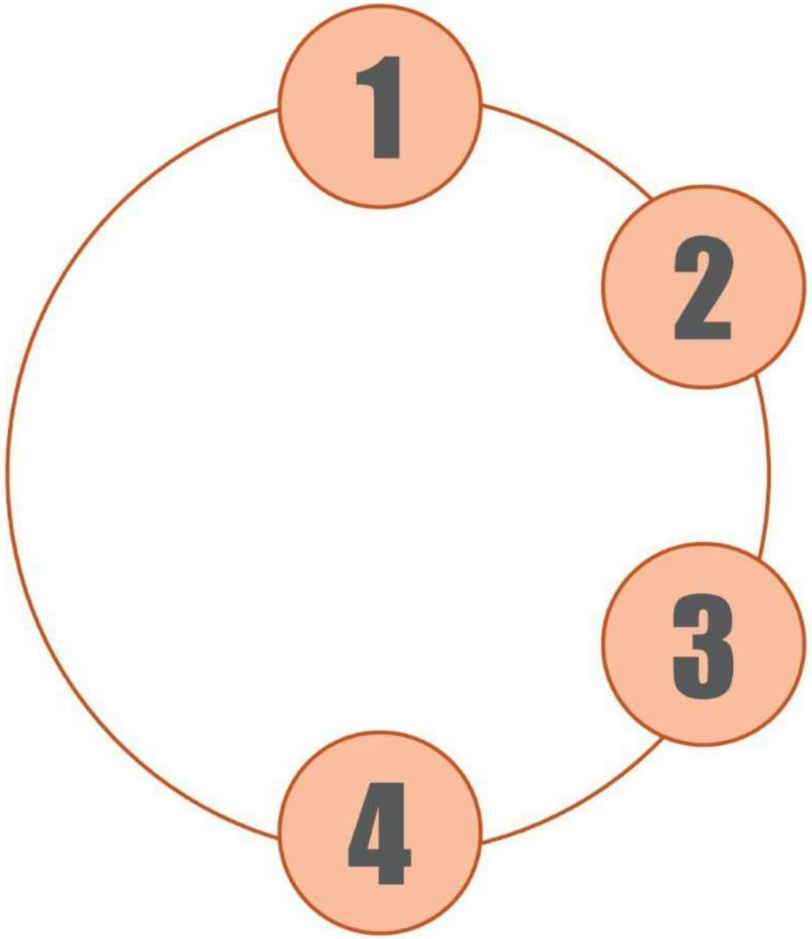


Storing Data in Cassandra

Range of all possible token values

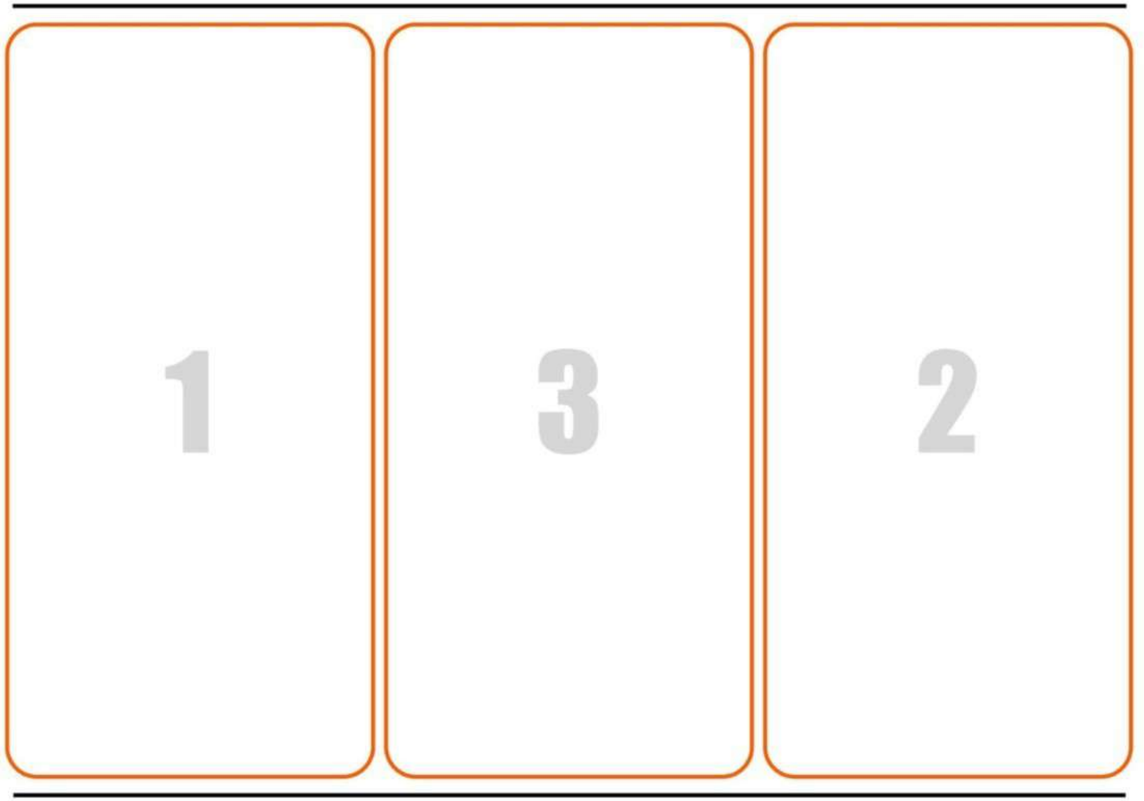


-2^{63} to $+2^{63}$

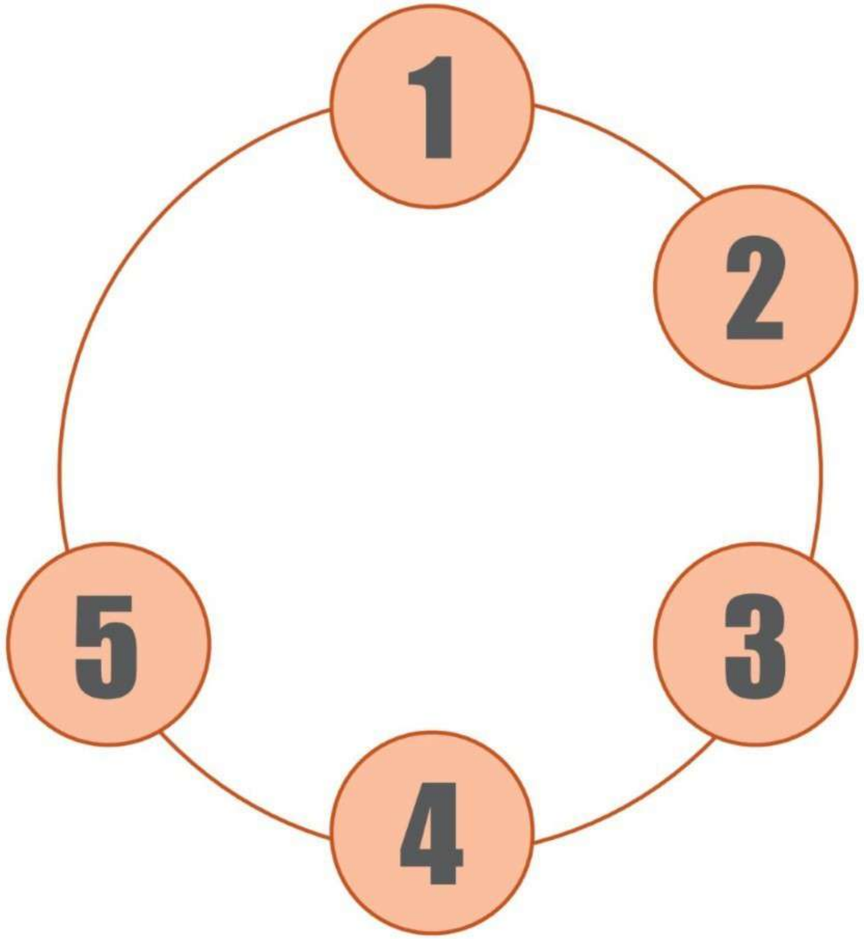


Storing Data in Cassandra

Range of all possible token values

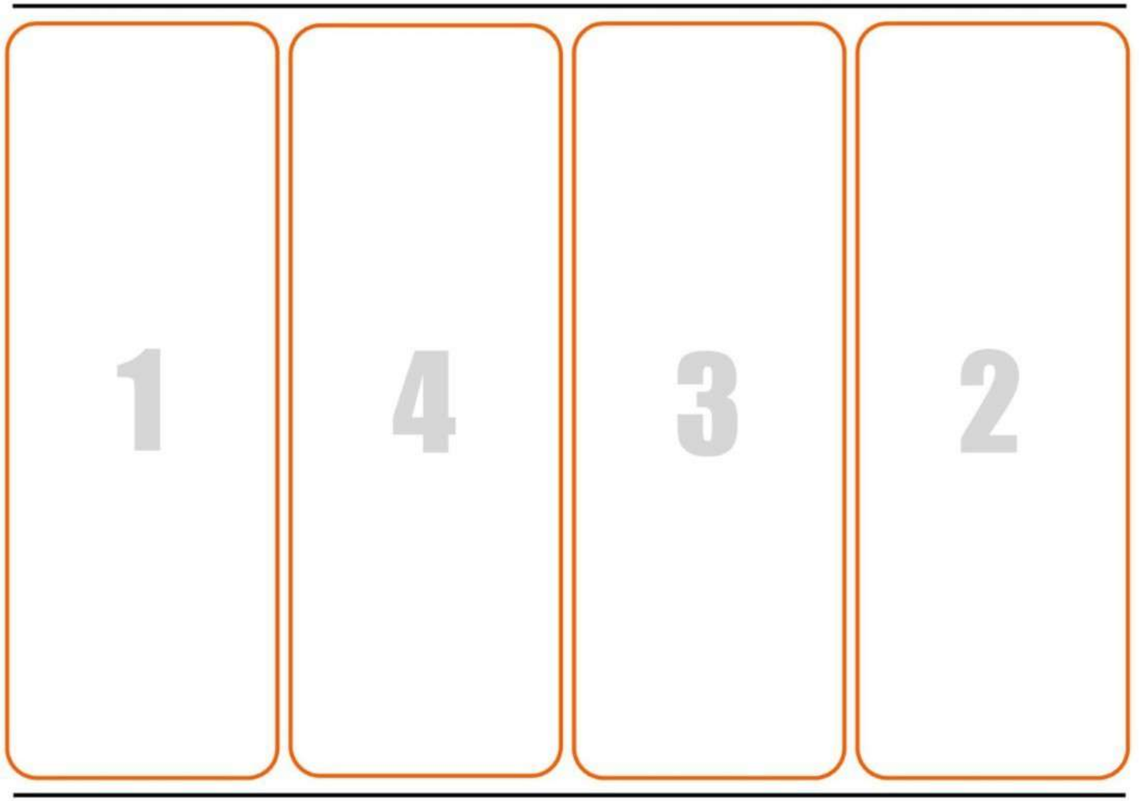


-2^{63} to $+2^{63}$

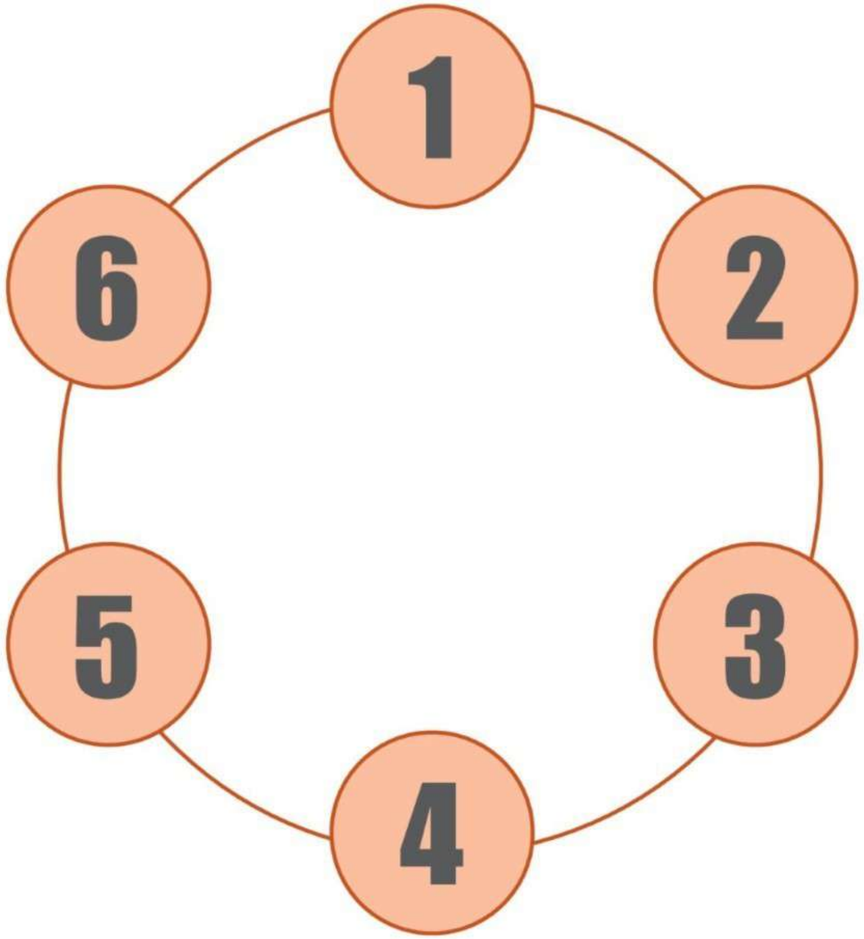


Storing Data in Cassandra

Range of all possible token values

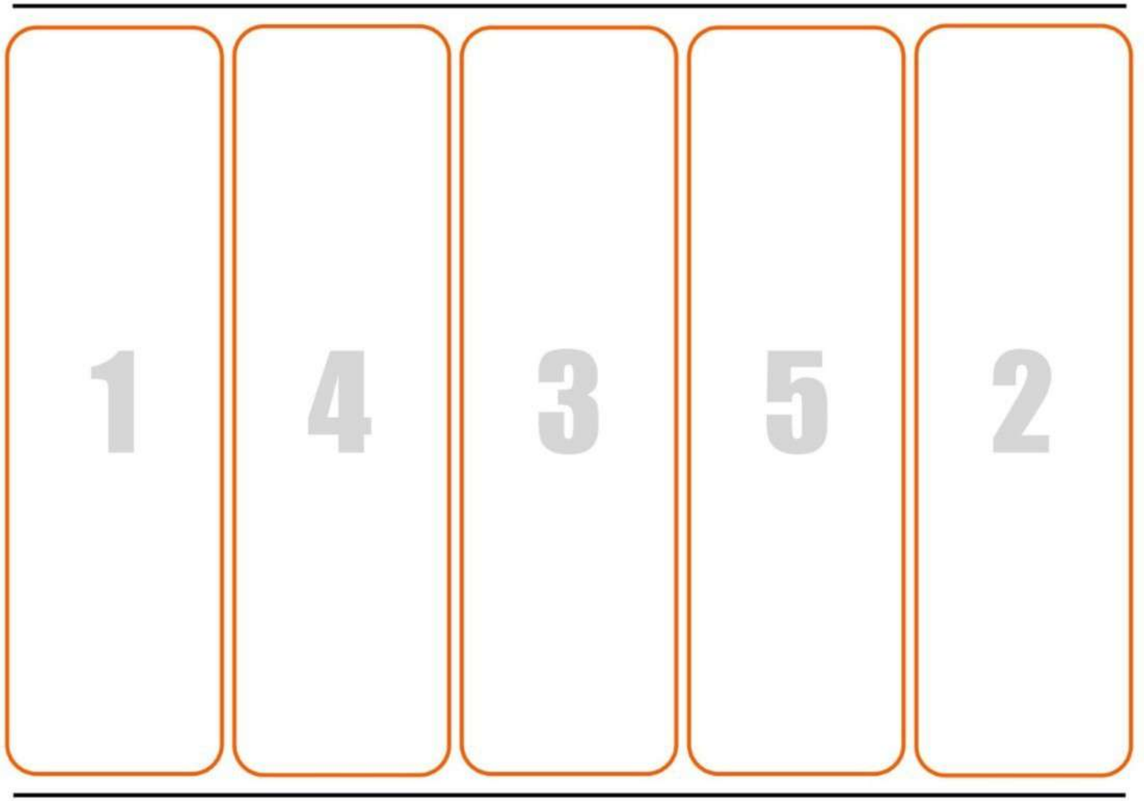


-2^{63} to $+2^{63}$

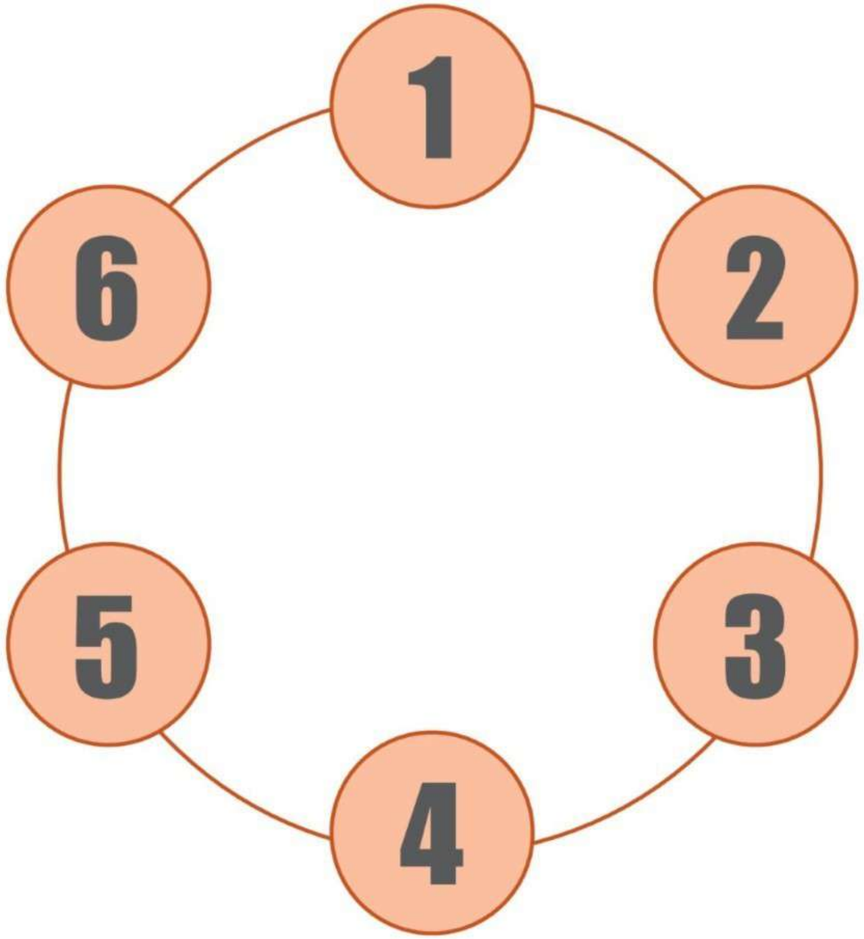


Storing Data in Cassandra

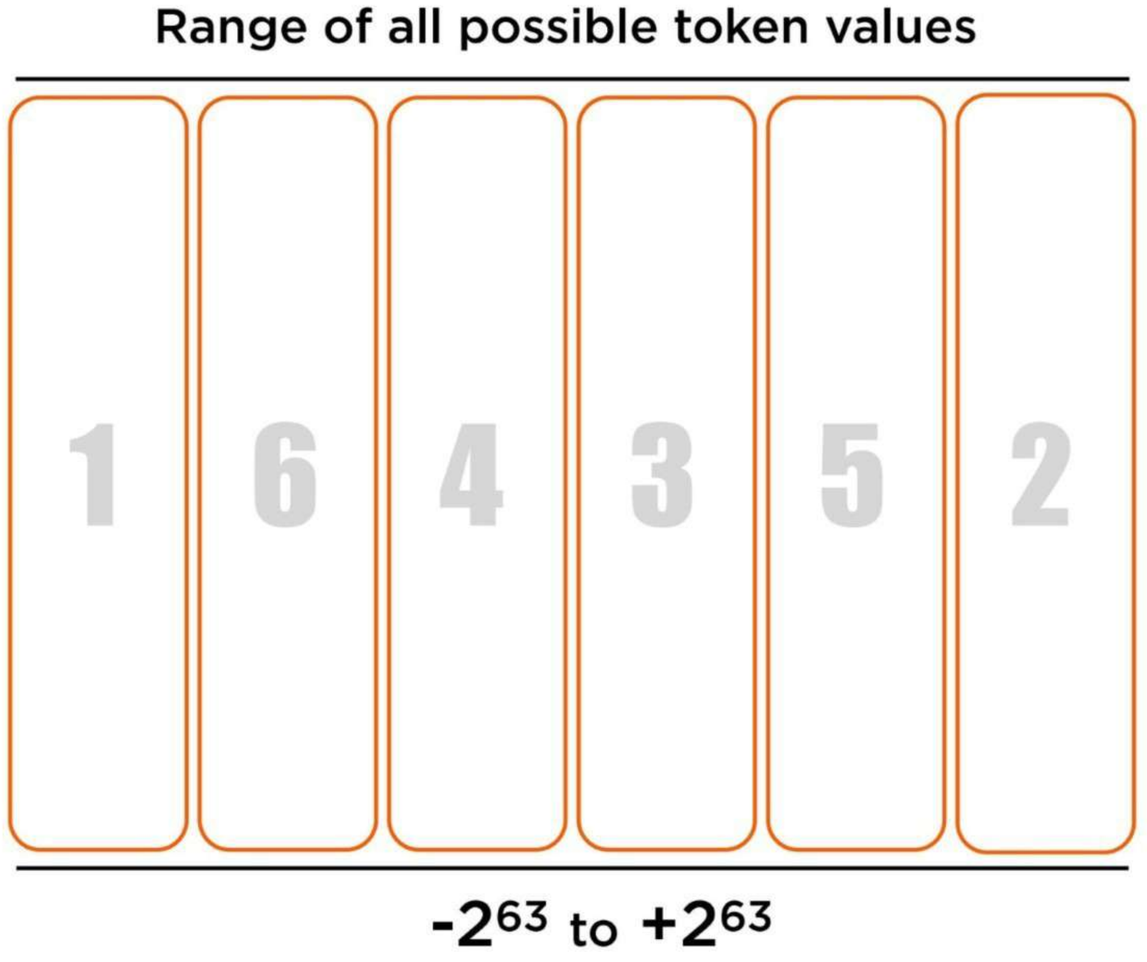
Range of all possible token values



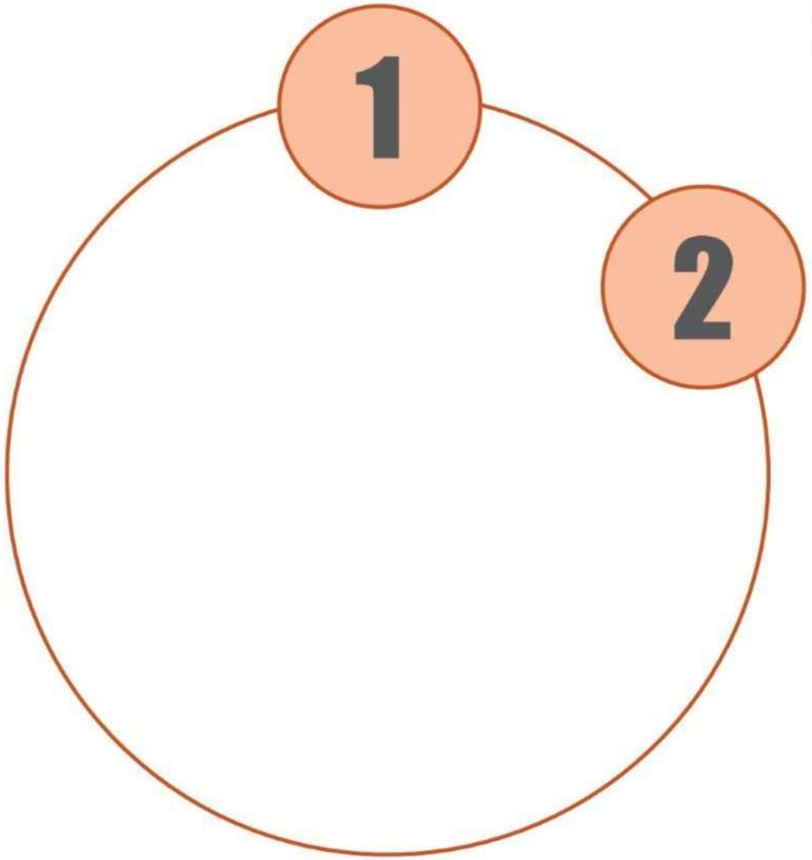
-2^{63} to $+2^{63}$



Storing Data in Cassandra



Storing Data with Virtual Nodes

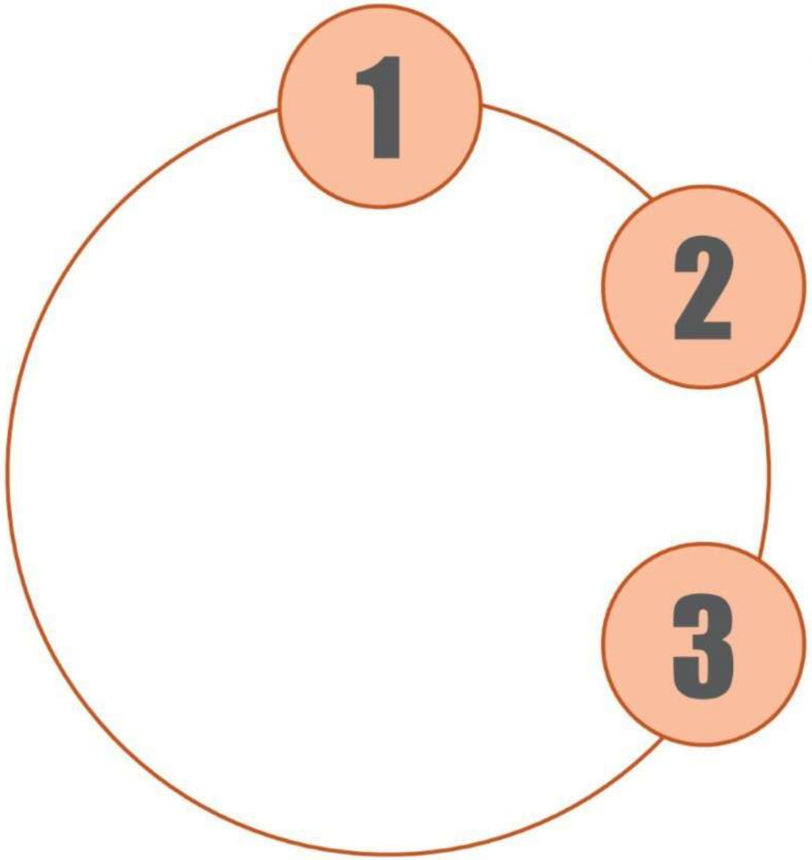


Range of all possible token values

1	1
1	1

-2^{63} to $+2^{63}$

Storing Data with Virtual Nodes

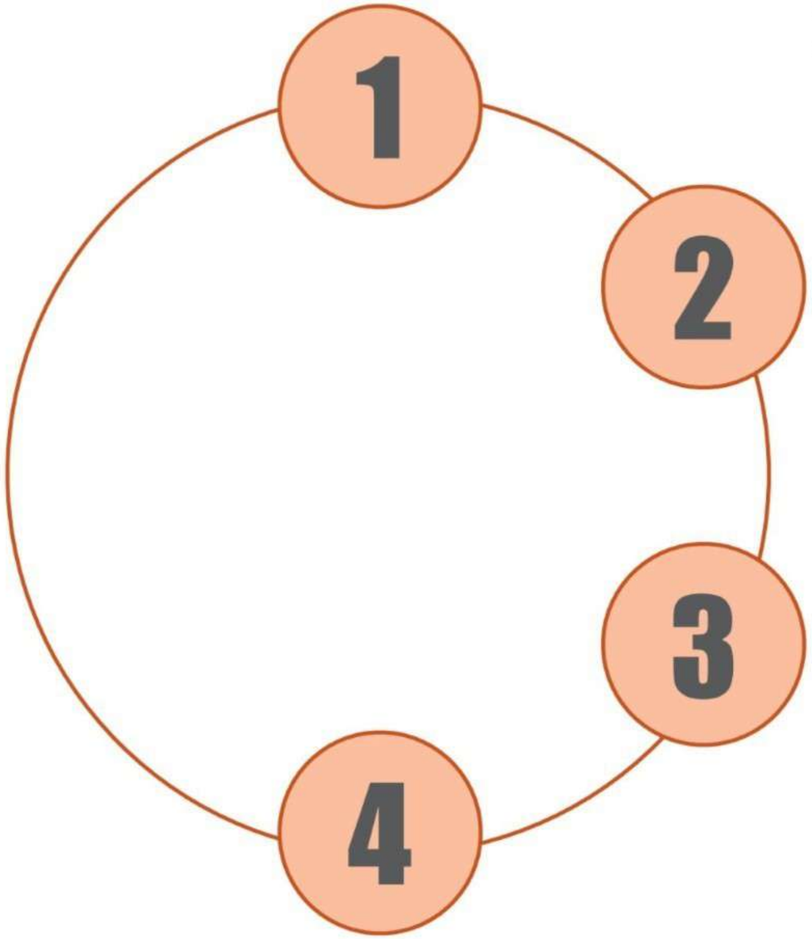


Range of all possible token values

1	2
2	1
1	2
2	1

-2^{63} to $+2^{63}$

Storing Data with Virtual Nodes

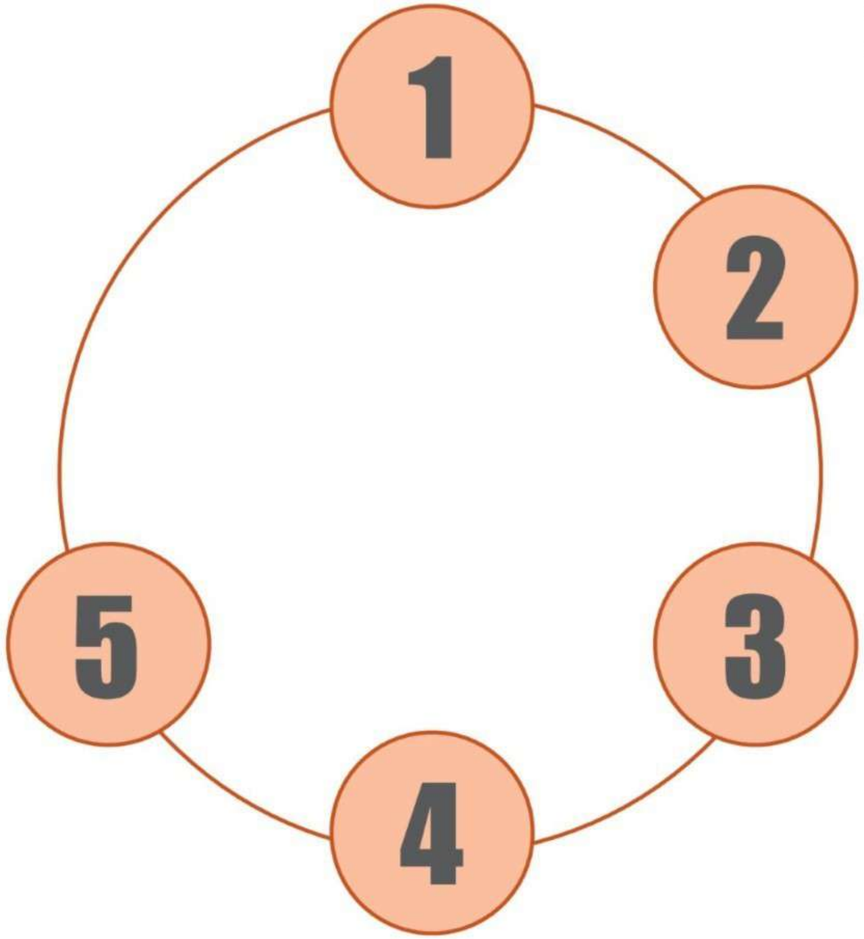


Range of all possible token values

1	2	3
2	3	1
1	2	3
3	2	1

-2^{63} to $+2^{63}$

Storing Data with Virtual Nodes

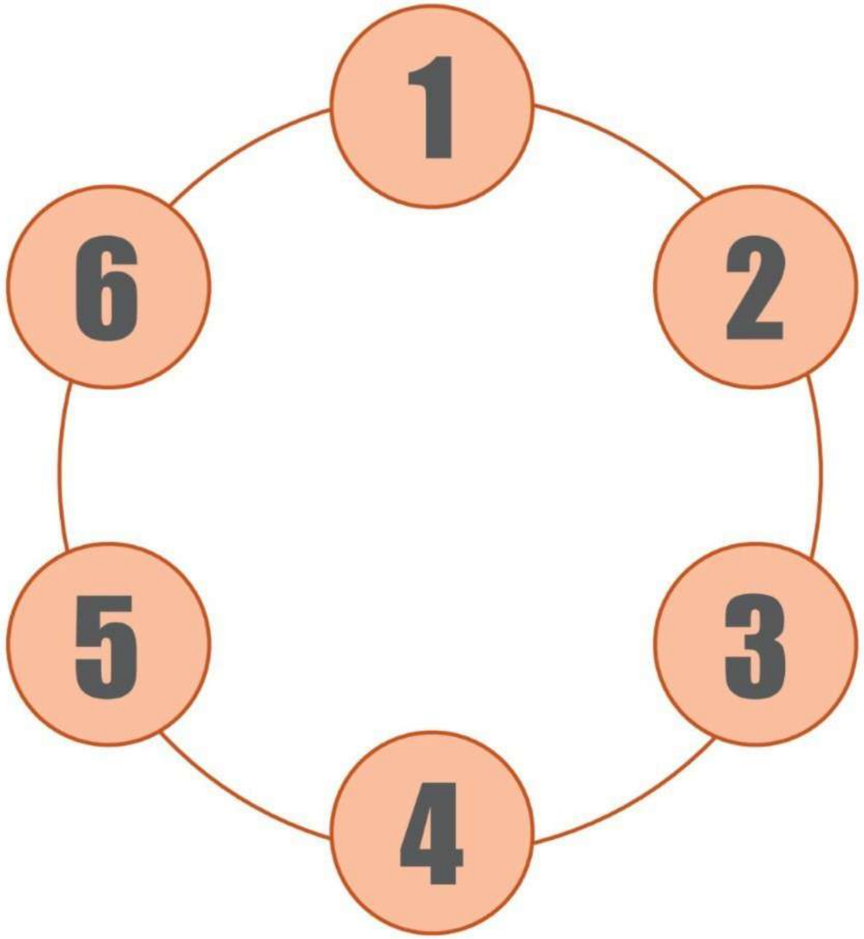


Range of all possible token values

1	2	3	4
2	4	3	1
4	1	2	3
3	2	4	1

-2^{63} to $+2^{63}$

Storing Data with Virtual Nodes

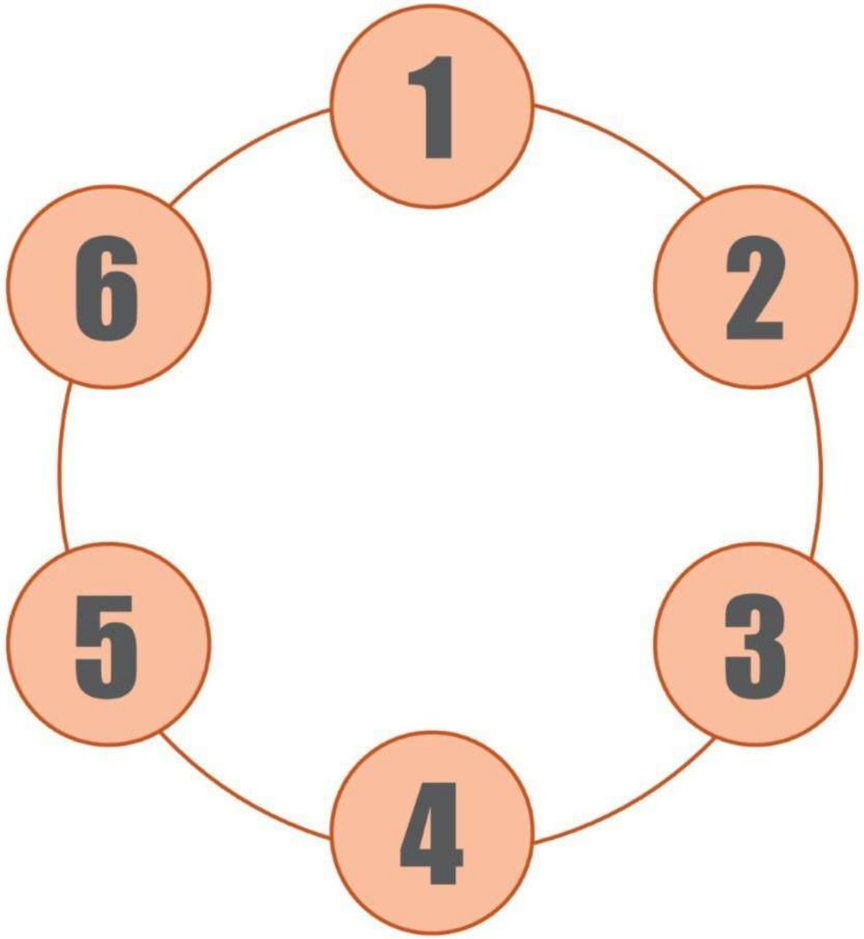


Range of all possible token values

1	2	3	5	4
5	2	4	3	1
4	1	2	3	5
3	5	2	4	1

-2^{63} to $+2^{63}$

Storing Data with Virtual Nodes

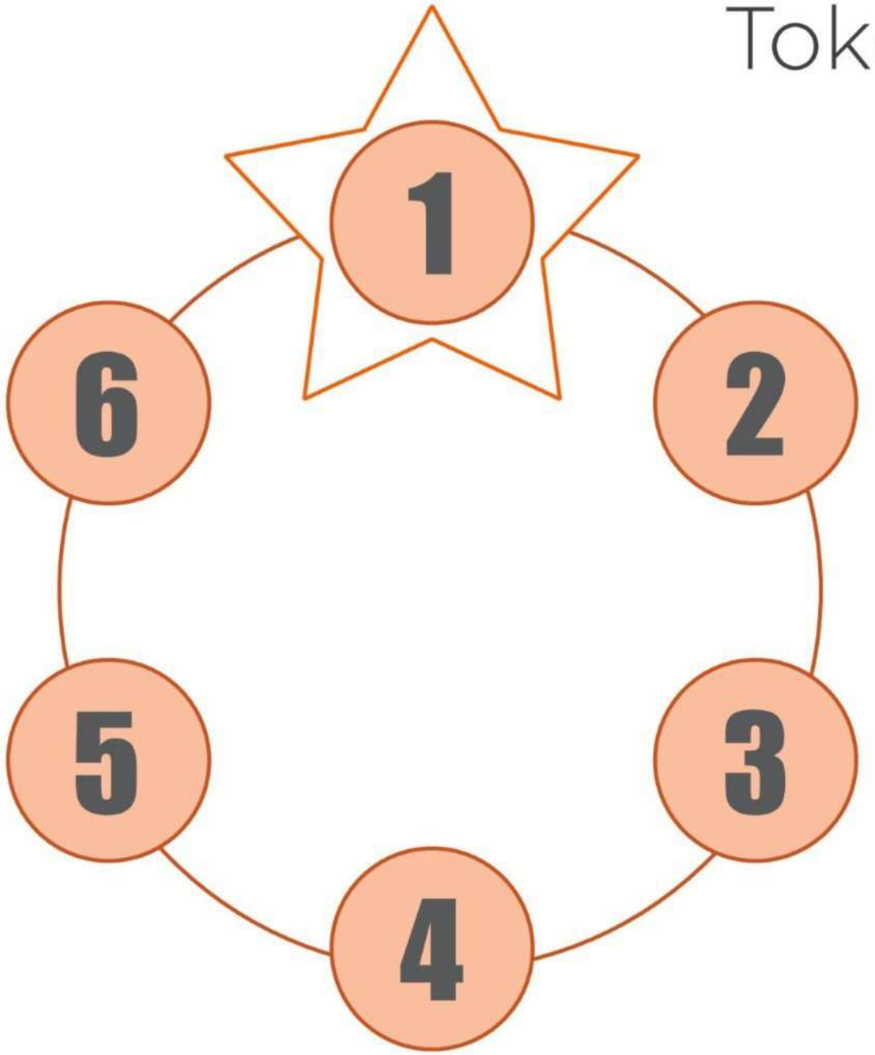


Range of all possible token values

1	2	3	5	4	6
5	2	4	6	3	1
4	6	1	2	3	5
3	5	2	4	6	1

-2^{63} to $+2^{63}$

Token Allocation via the Partitioner




```
insert into…
```

hash(partition key)

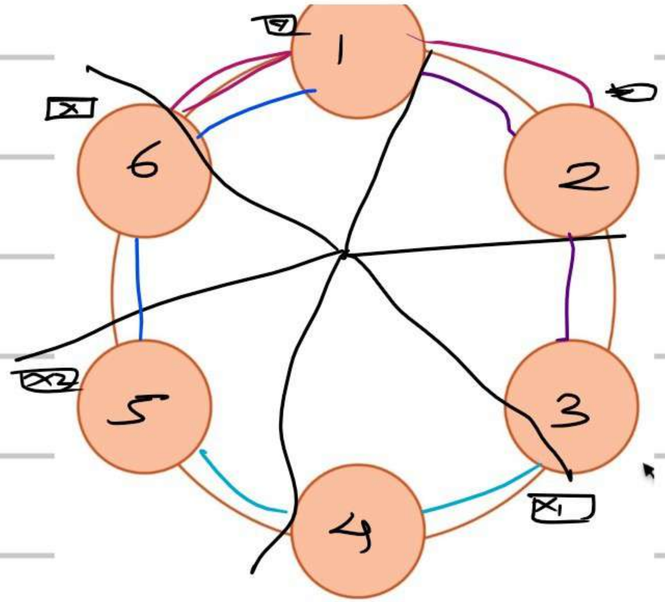
1	2	3	5	4	6
5	2	4	6	3	1
4	6	1	2	3	5
3	5	2	4	6	1

-2^{63} to $+2^{63}$

Replication Strategy → Simple

RF → 1
→ 3

→ once the token value is identified then the records are also stored in the next 2 adjacent nodes

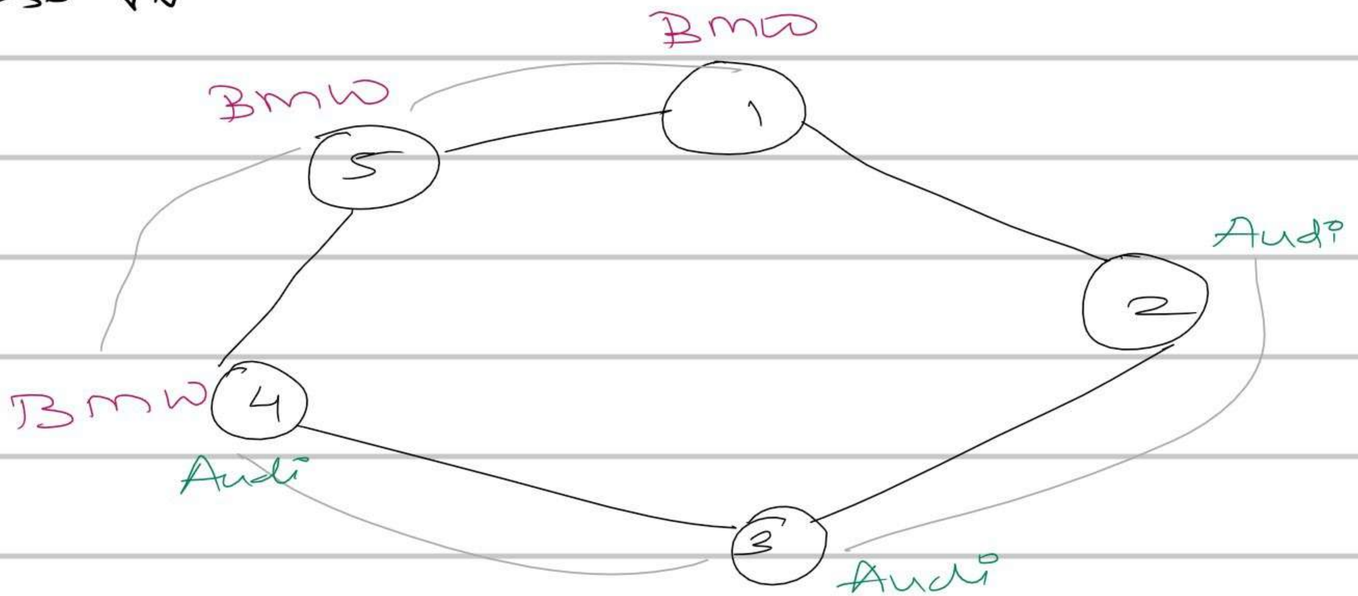


Cassandra 3 → 256
Cassandra 5 → 16

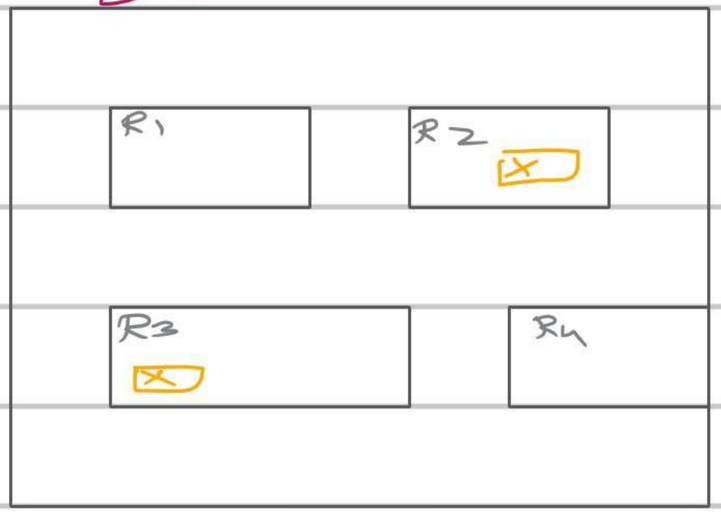
1 N → 256 GB - 8 VN
64 GB - 4 VN

1 N → 256 VN

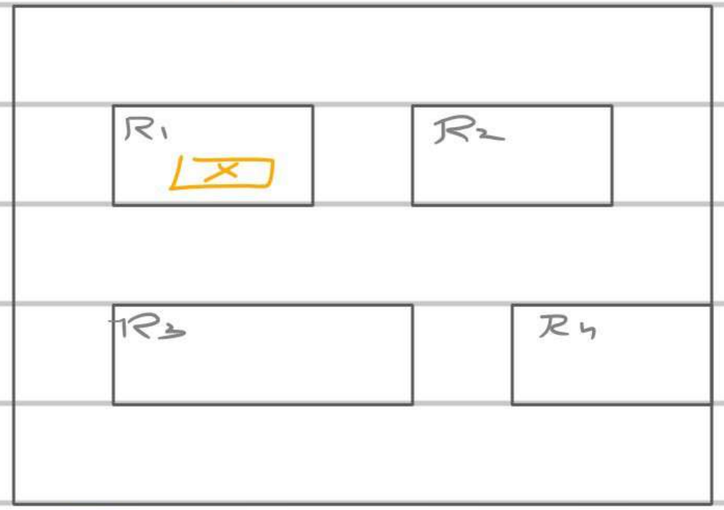
One DC &
One Rack



EU



IN



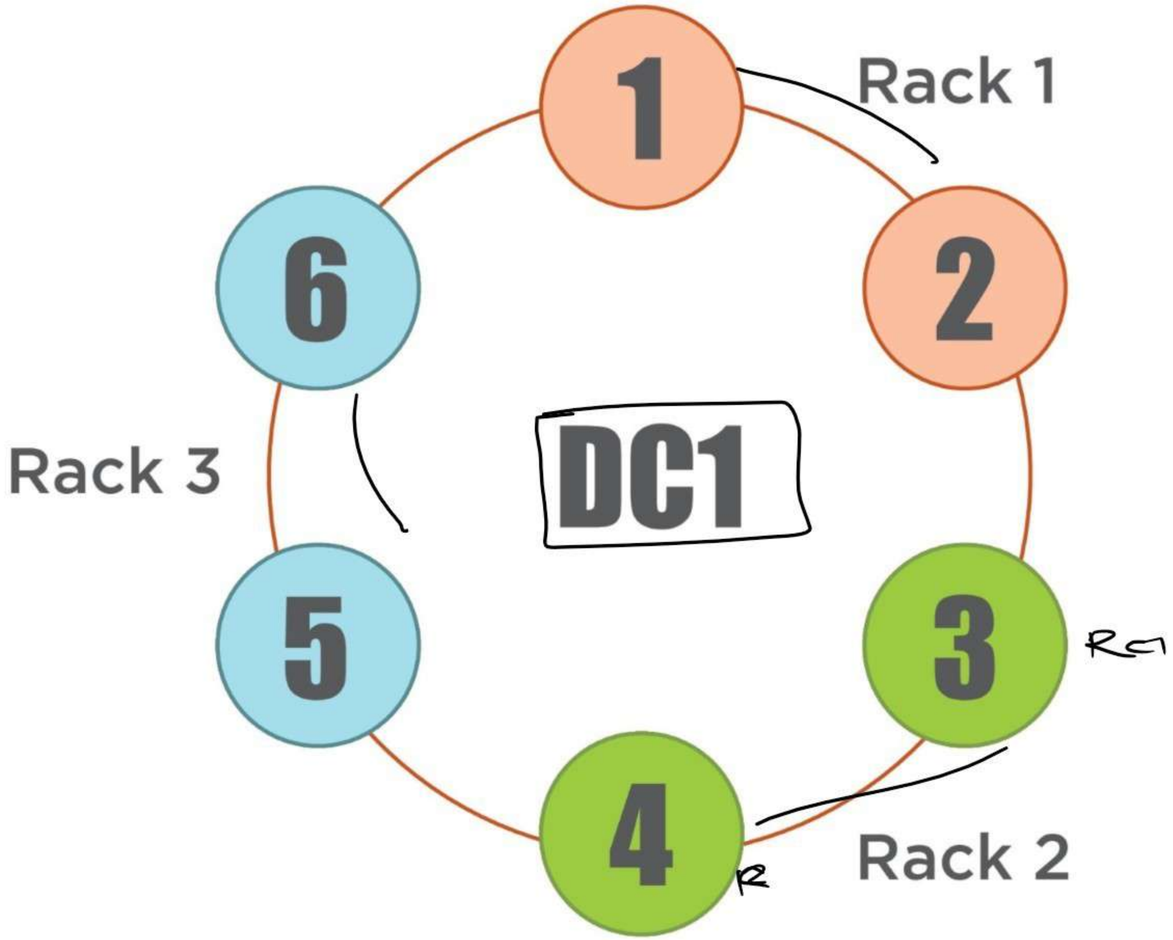
Strategy - NetworkTopology.

RF → 3

First copy ⇒ rack which is finalized

Second copy on some other data rack

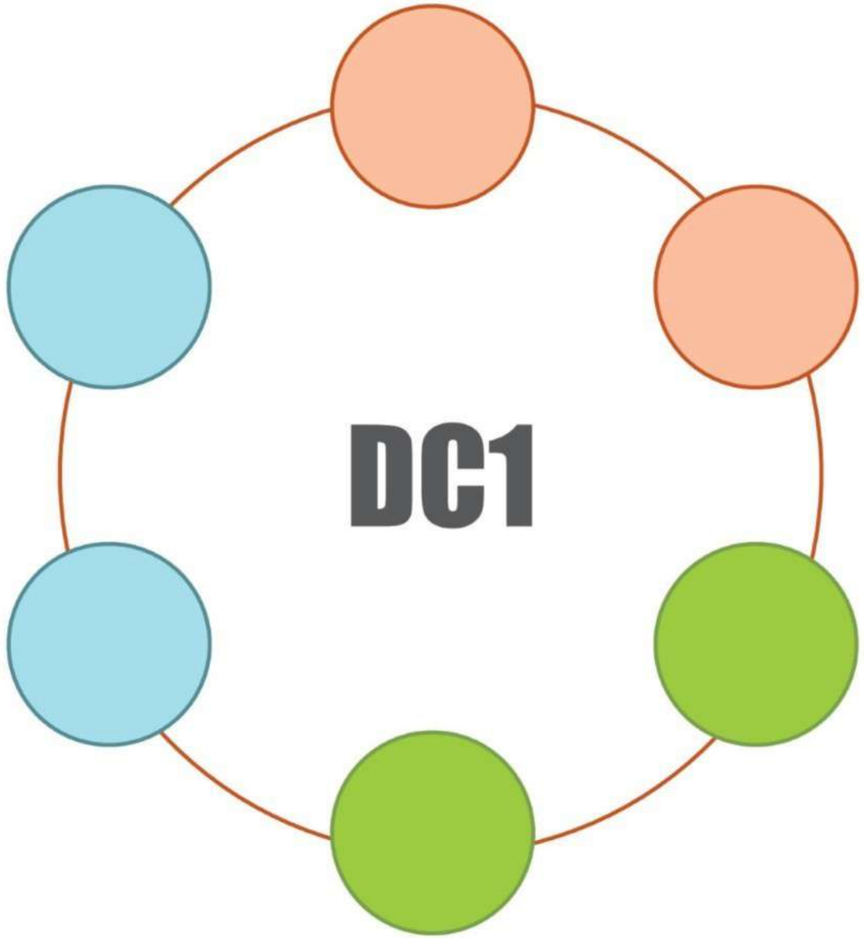
Third one on any random datacenter & rack; which doesn't have it



Snitch

SimpleSnitch

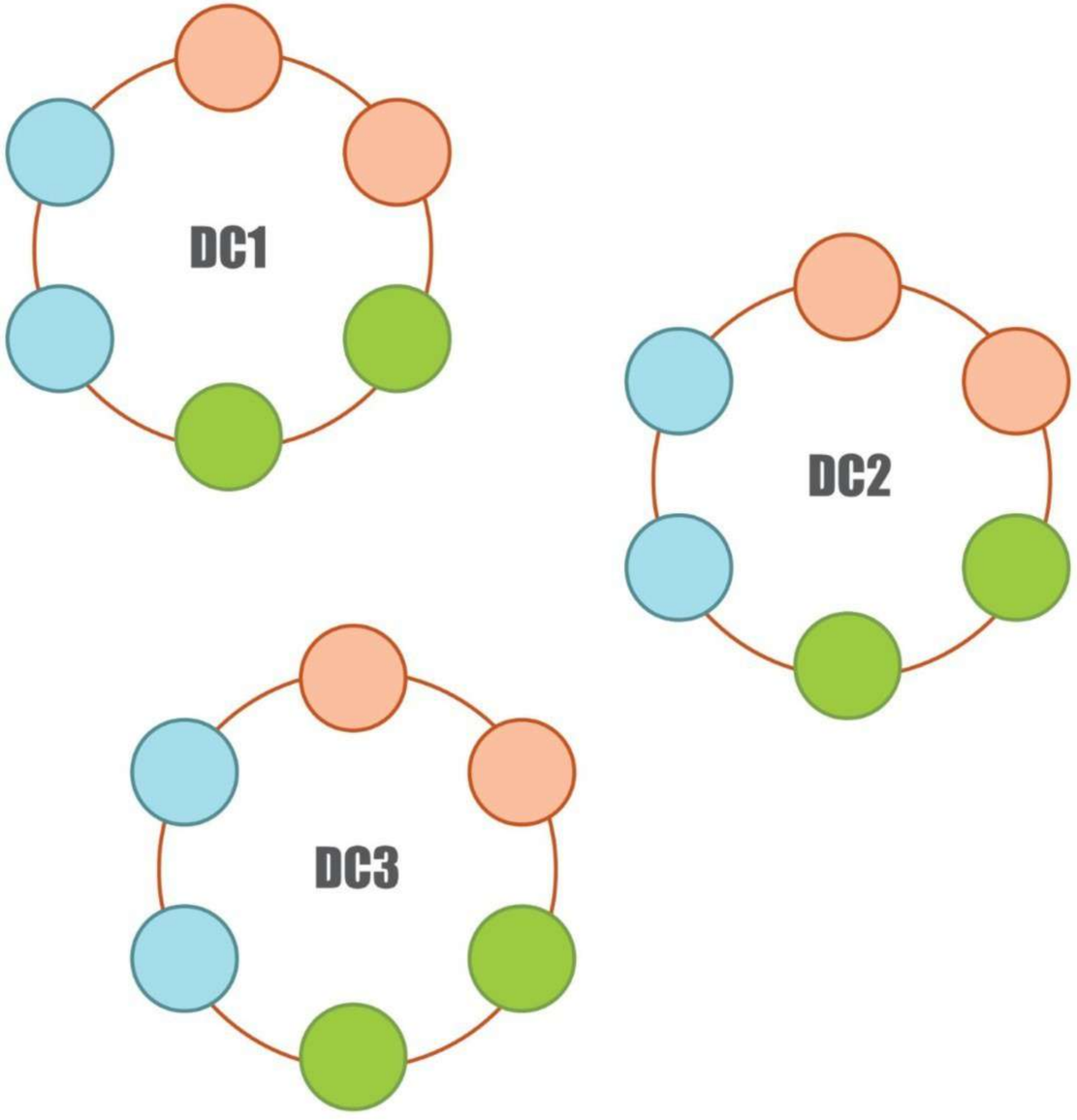
GossipingPropertyFileSnitch



Snitch

SimpleSnitch

GossipingPropertyFileSnitch



Snitch

SimpleSnitch

GossipingPropertyFileSnitch

PropertyFileSnitch

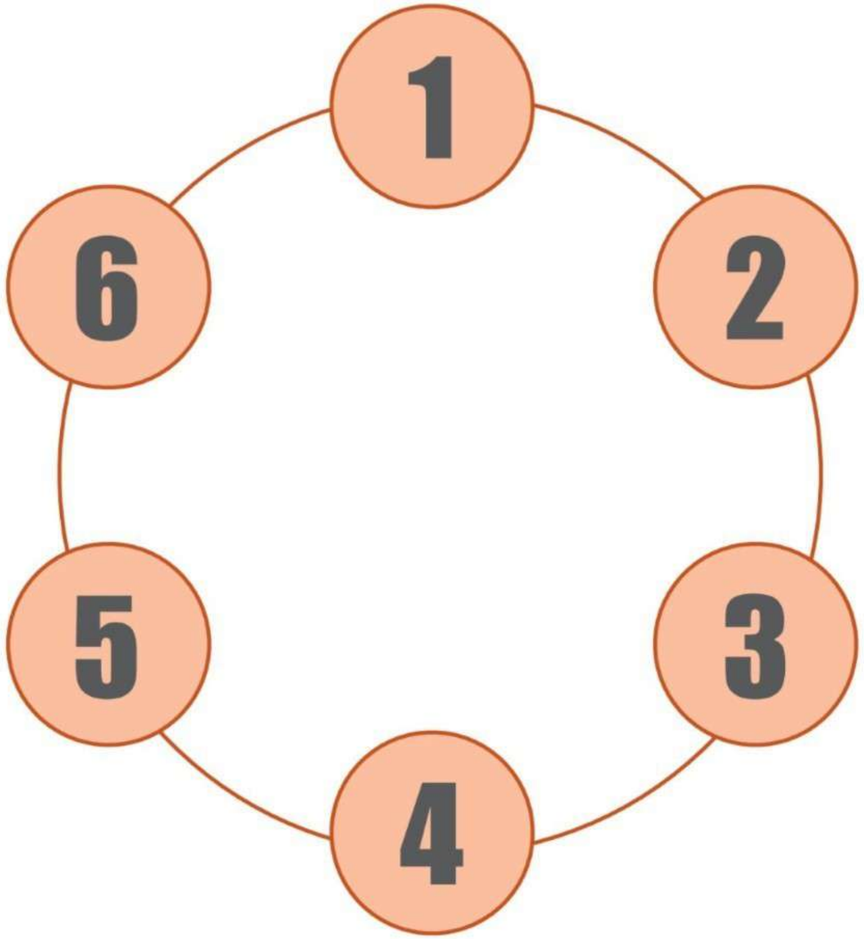
EC2Snitch

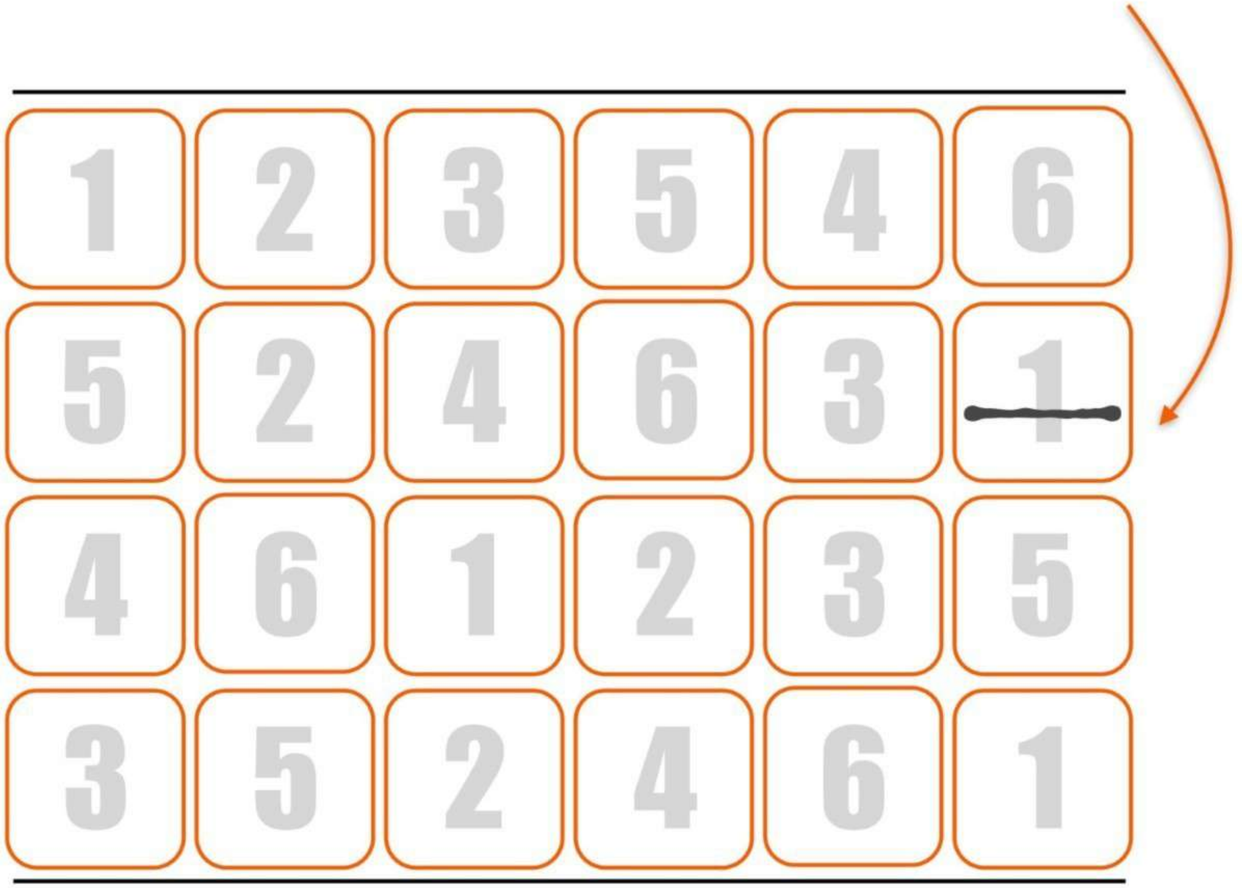
EC2MultiRegionSnitch

RackInferringSnitch

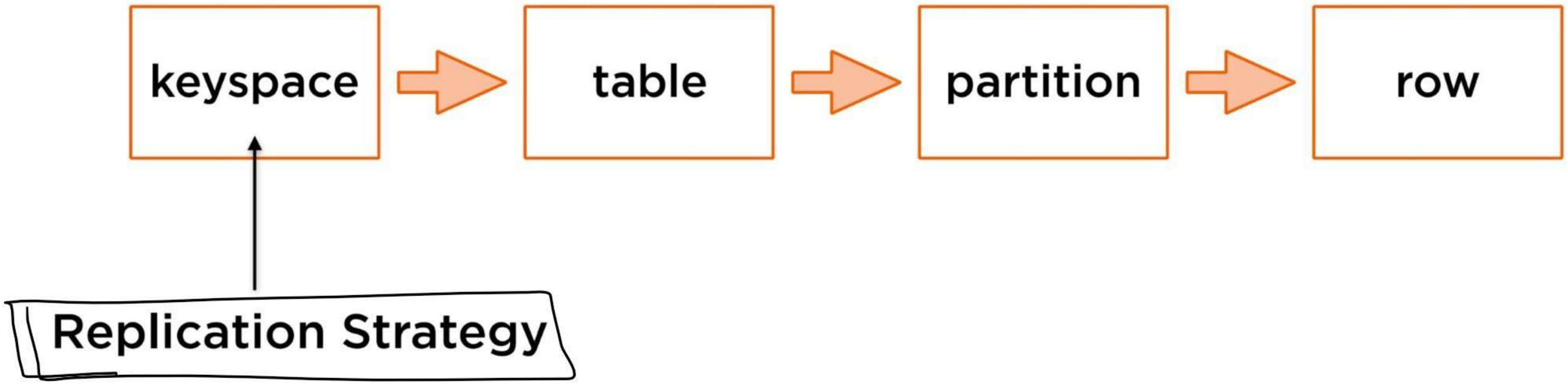
Replication Stratergy and Consistency

Replication



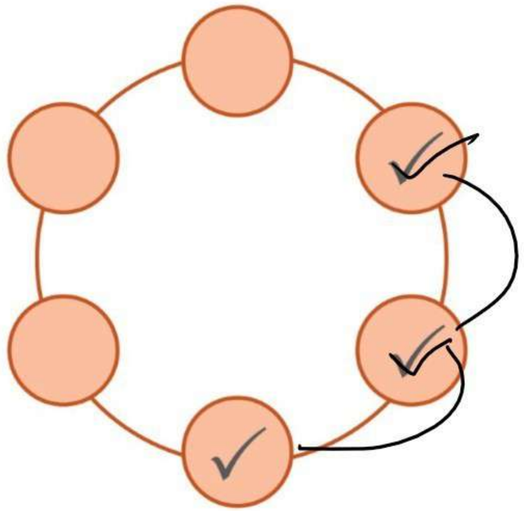


Cassandra Terminology



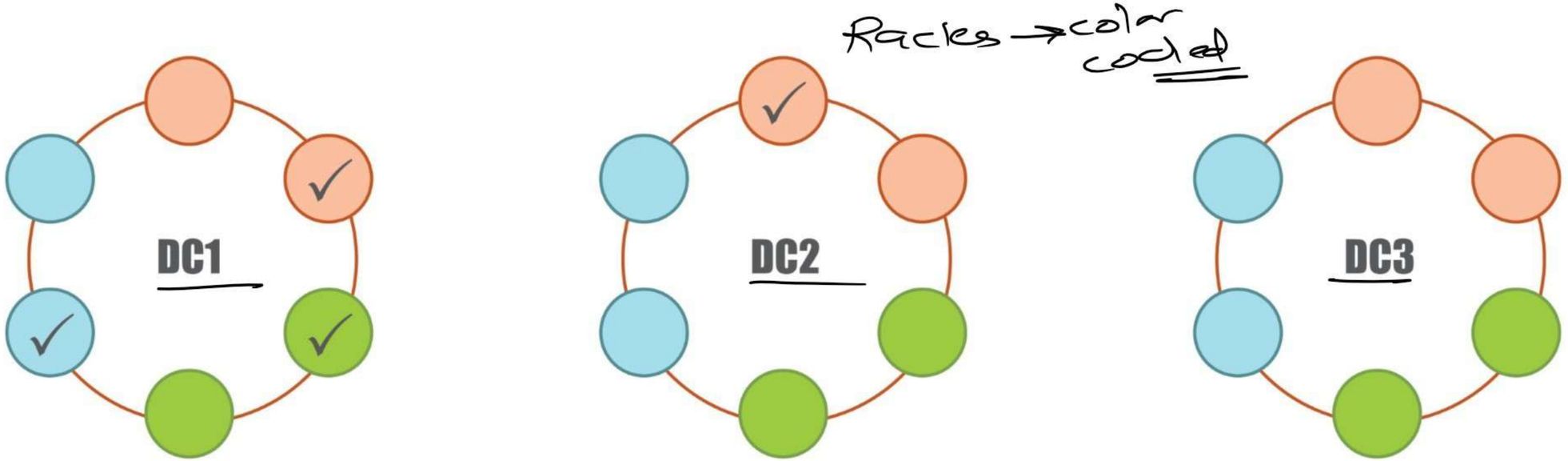
SimpleStrategy

```
create keyspace pluralsight with replication =
  {'class': 'SimpleStrategy', 'replication_factor': 3};
```



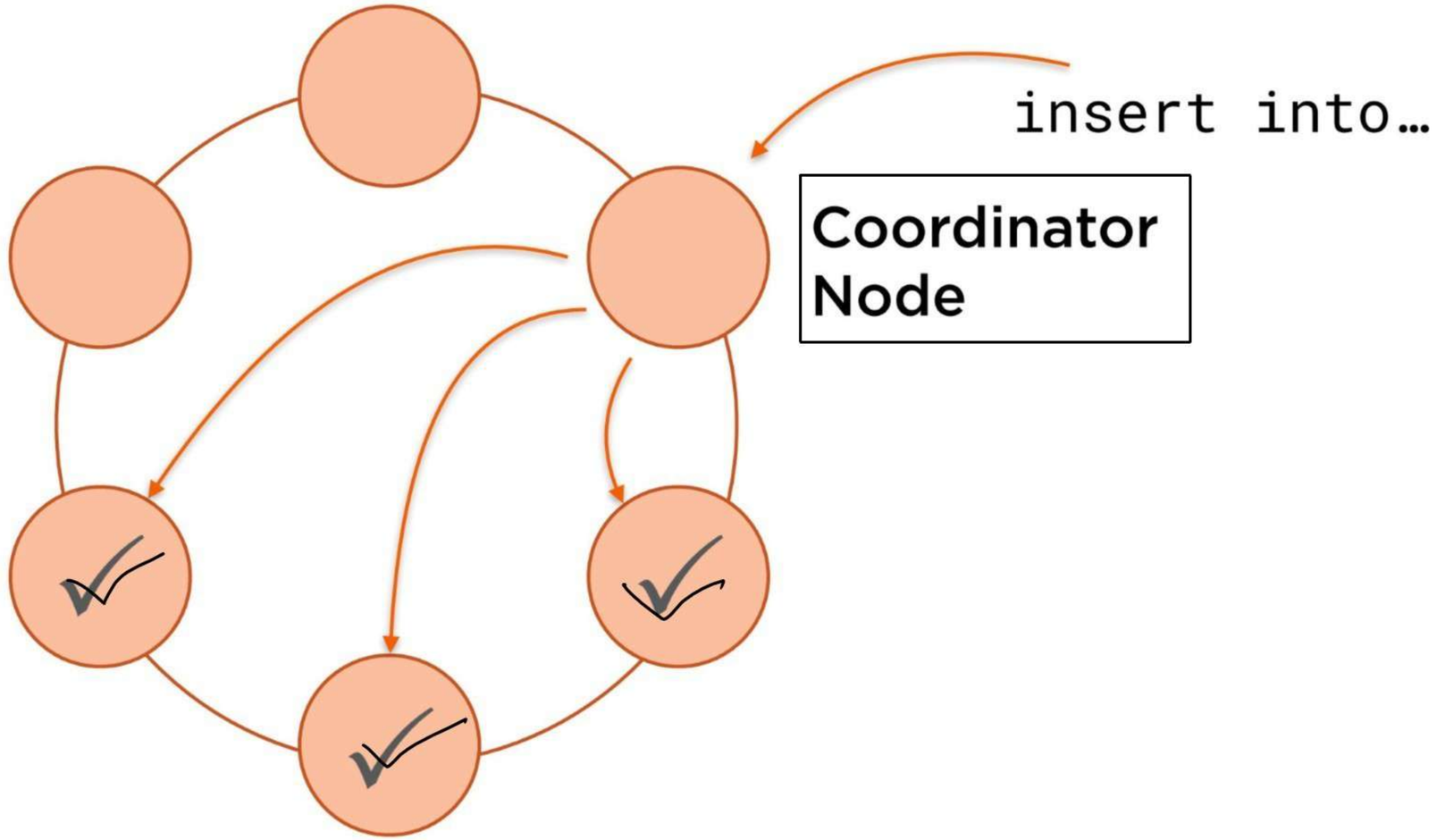
NetworkTopologyStrategy

```
create keyspace pluralsight with replication =
  {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2', 1};
```

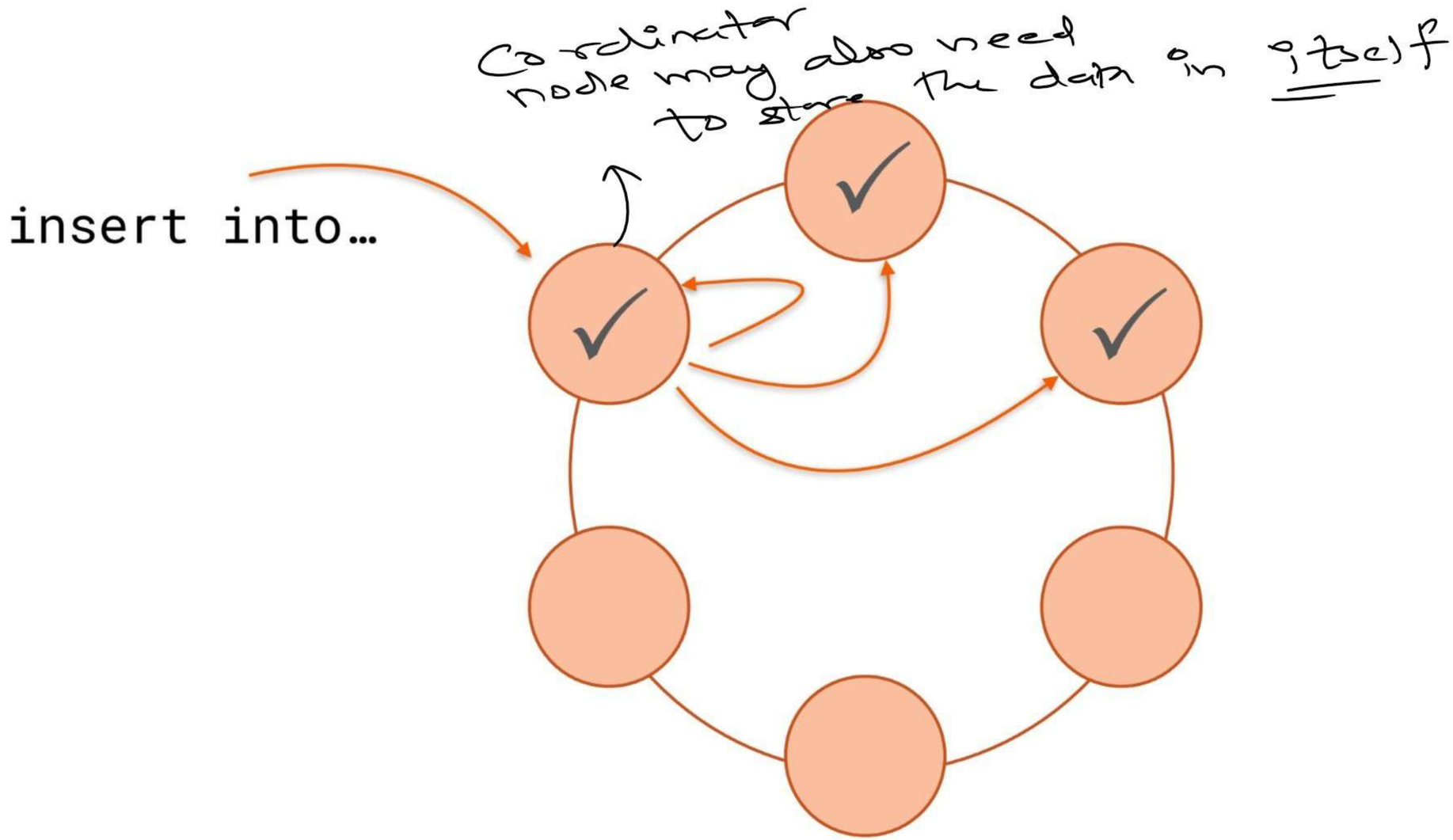


Consistency

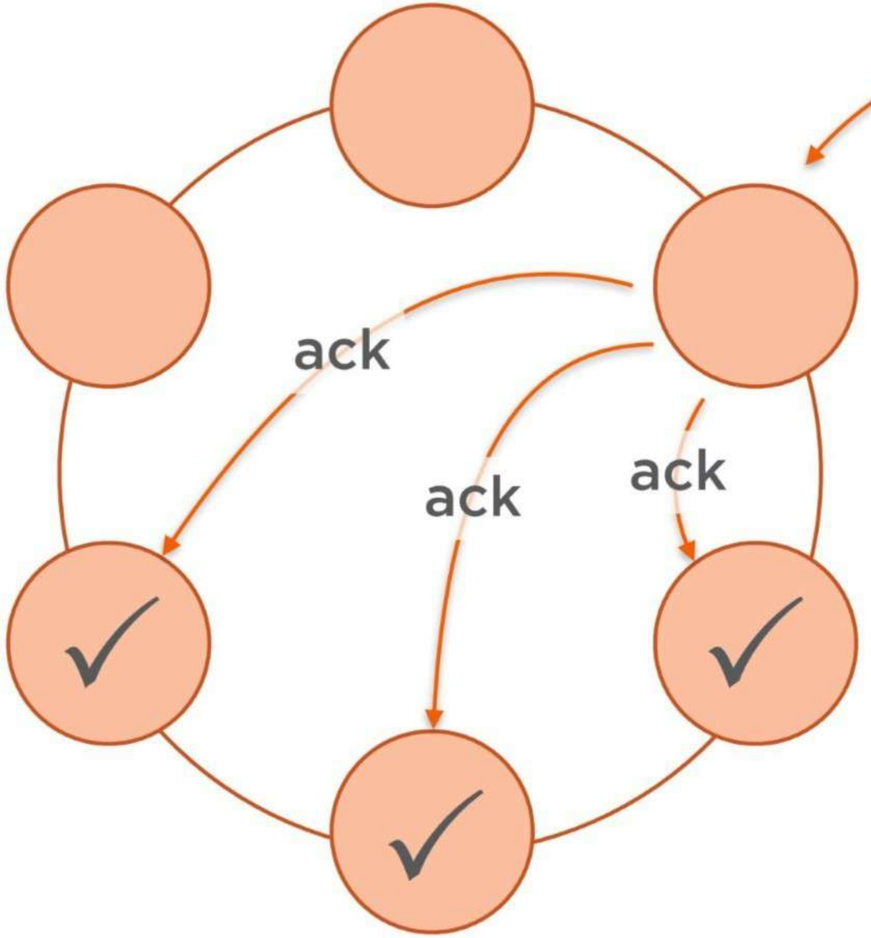
Reads and Writes in Cassandra



Reads and Writes in Cassandra



Tunable Consistency - Writes



Writing

RF = 3

$\frac{3+1}{2} = 2$

ONE → if we get ack from one of the node → write is successful

TWO

THREE

QUORUM → $\frac{\text{Replication Factor} + 1}{2}$

ALL

ANY → even if we can connect with Coordinator node

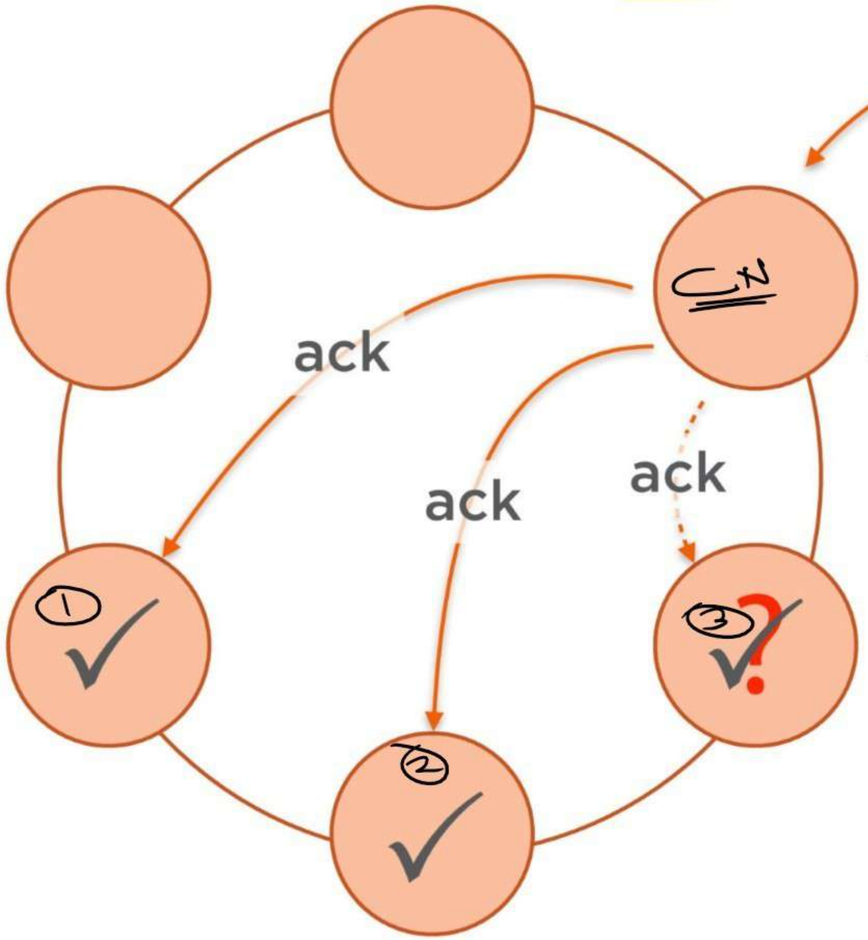
Hinted Handoff

→ important feature

RF = 3

ALL →

Successful write → 3

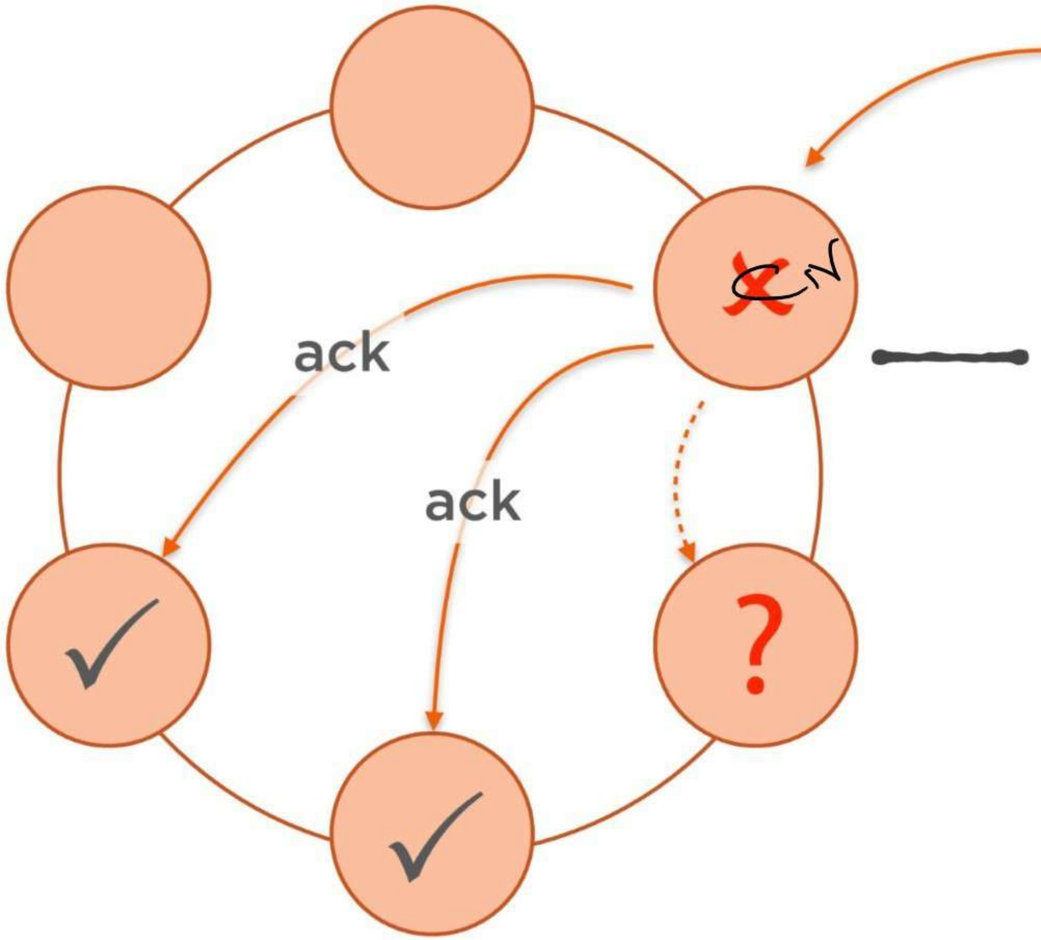


Solution

CN will store the data & keep it with itself until N3 comes back up & Then write to it.

→ Node 3 had some network issue → problem

Hinted Handoff + Read Repair



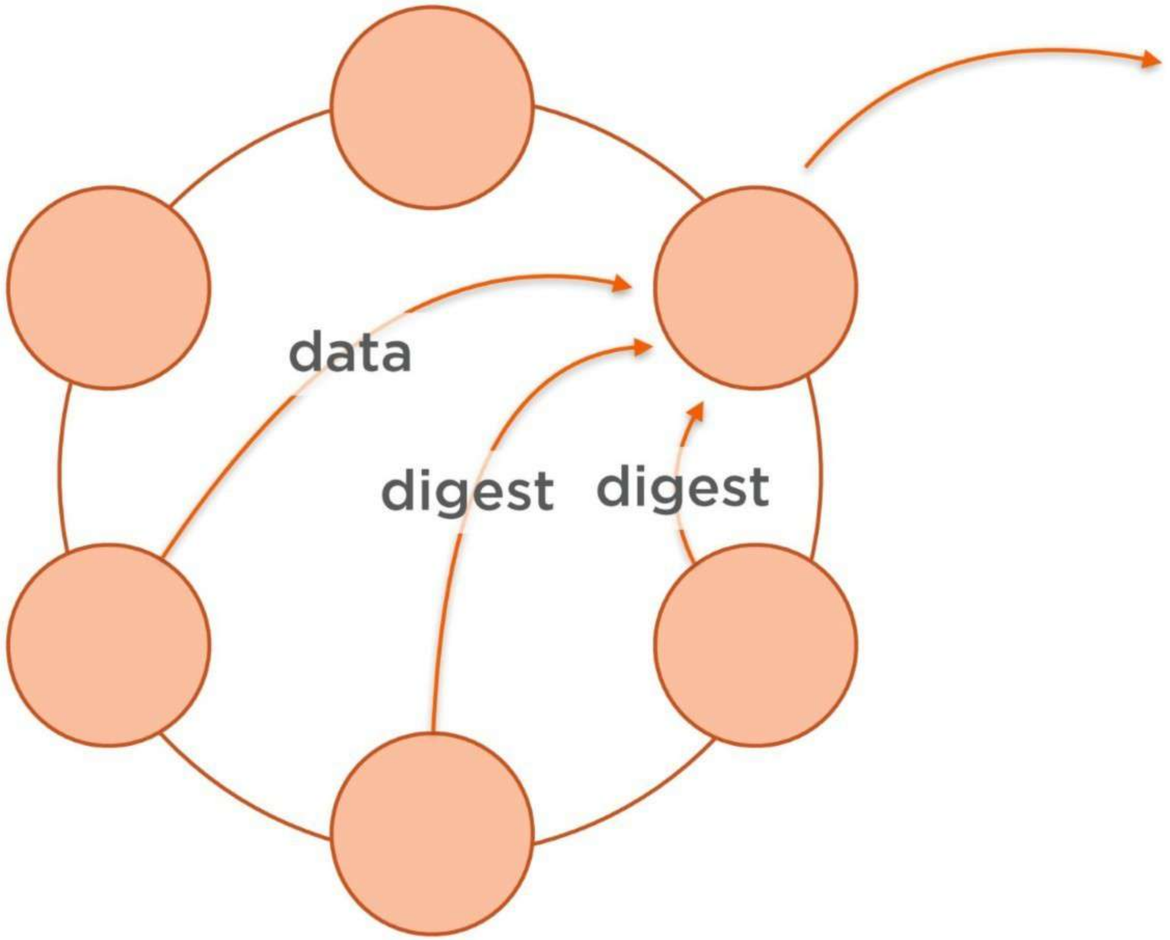
What if ?

→ N3 went of while write

→ CN was holding the data

→ while coordinator node was holding the data & before N3 came back CN died.

Tunable Consistency - Reads



ONE

TWO

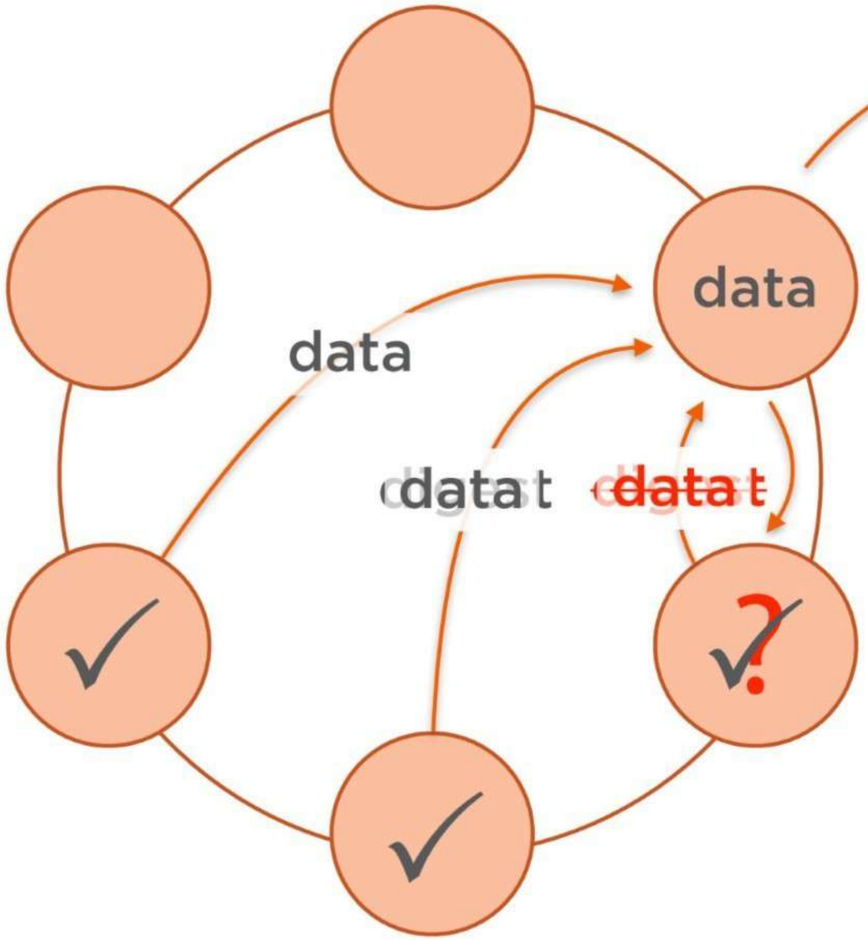
THREE

QUORUM → $\frac{RF+1}{2}$

ALL

RF = 3
CL → ALL

Read Repair

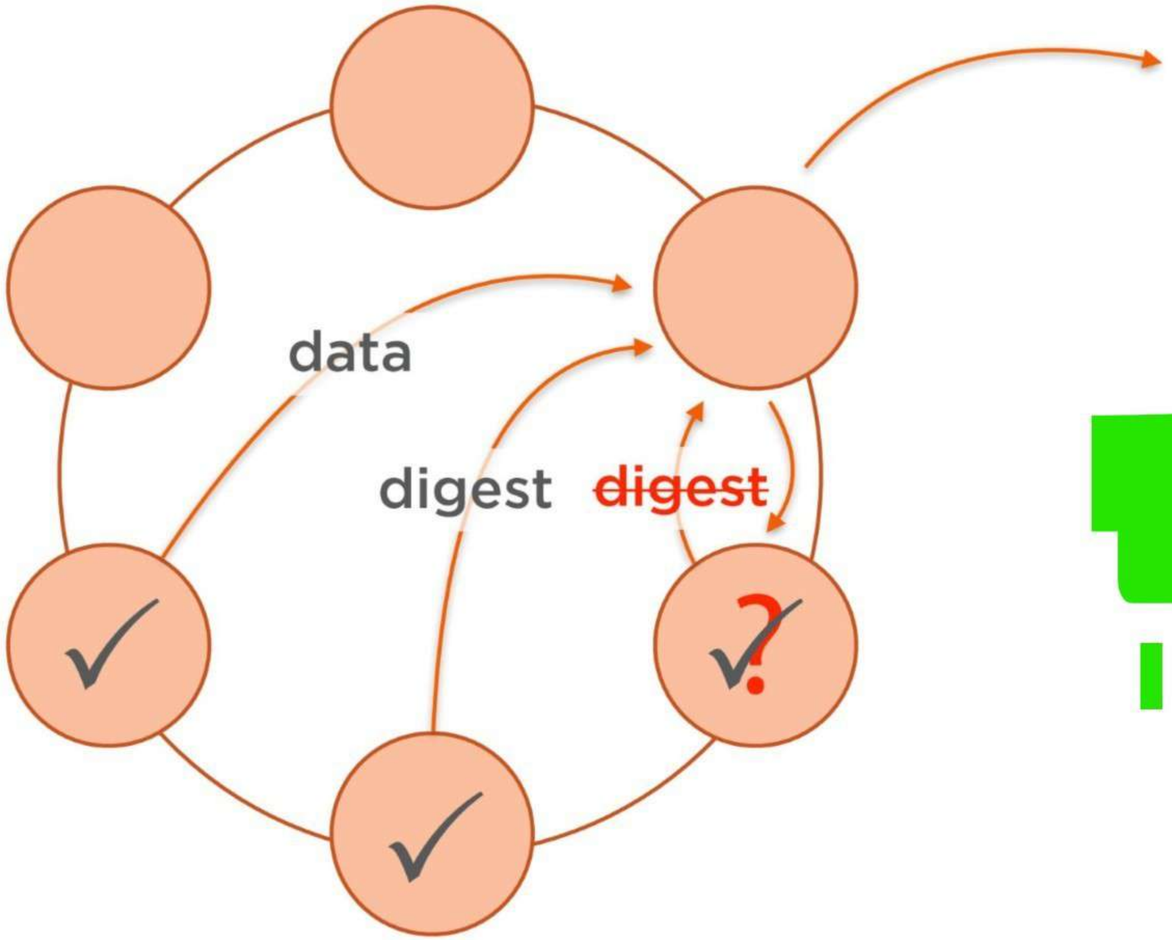


N1 → Data
N2 → digest → data? ✓
N3 → digest stale

Read Repair

→ update the data on N3 as apart of read process.

Read Repair




```
nodetool repair
```

Achieving Strong Consistency

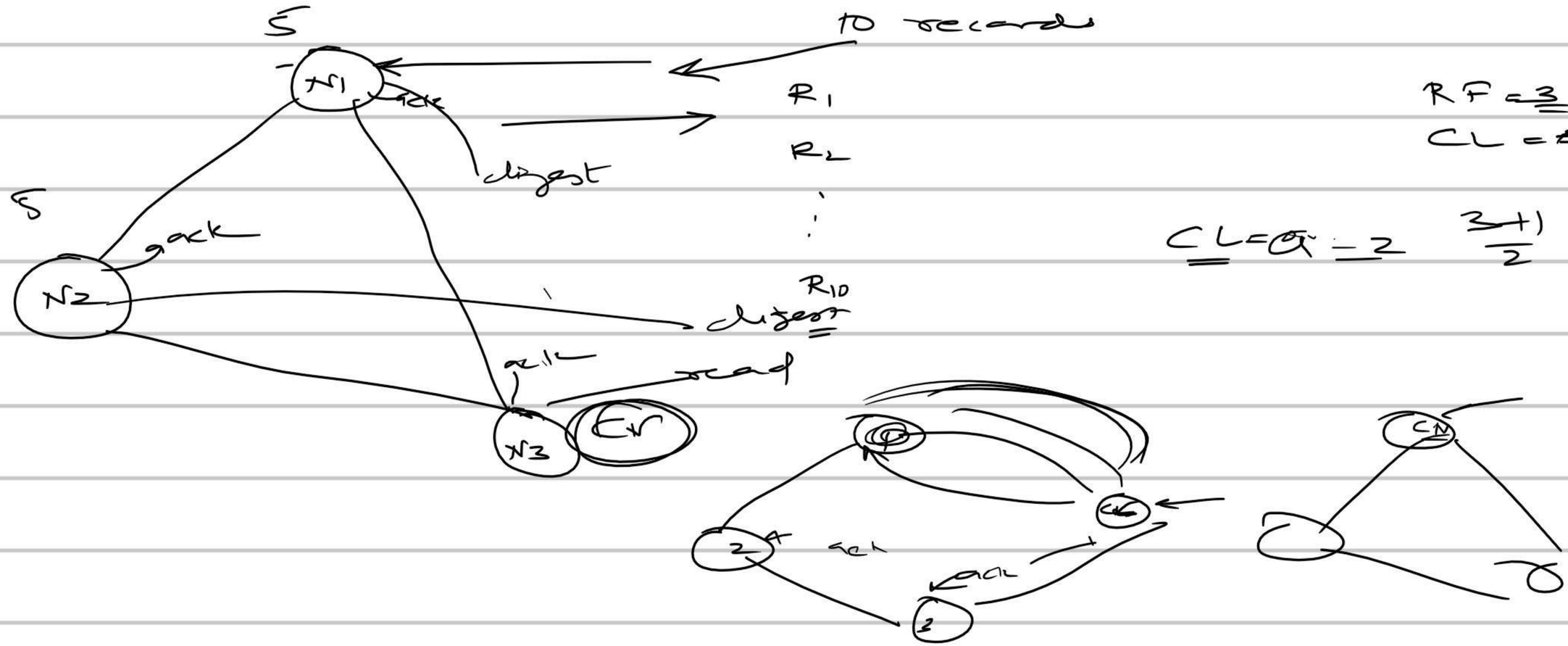
ensures that atleast one replica is common between the write & read.

(Write Consistency + Read Consistency) > Replication Factor

Quorum = $\frac{RF+1}{2} = \frac{3+1}{2} = 2$

Write Consistency		Read Consistency		Replication Factor	
1 ONE	+	2 QUORUM	3 > 3	3	✗
1 ONE	+	3 ALL	4 > 3	3	✓
1 ONE	+	1 ONE	2 > 3	3	✗
2 QUORUM	+	2 QUORUM	4 > 3	3	✓

1 (W) + 2 (R) = 3



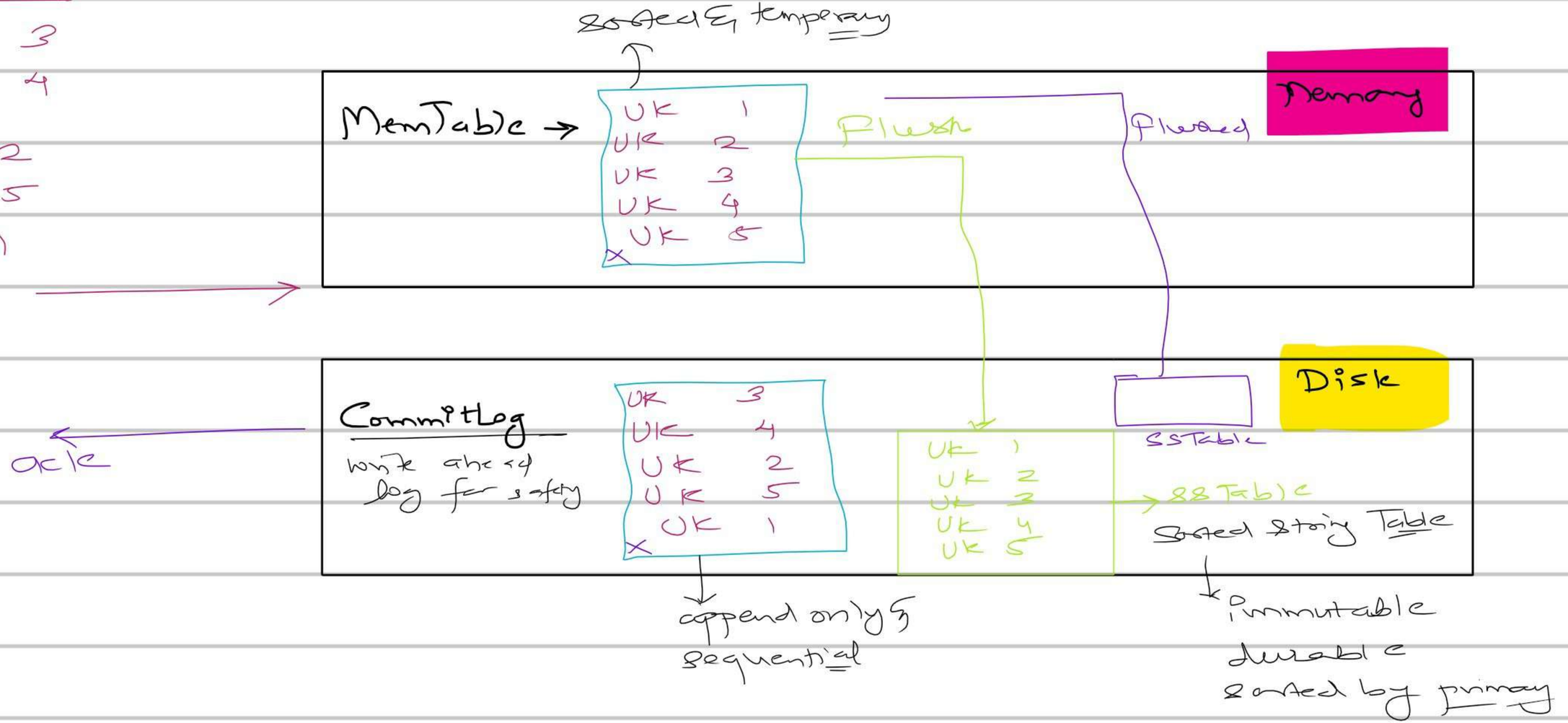
RF = 3

CL = ~~ALL~~ / Q

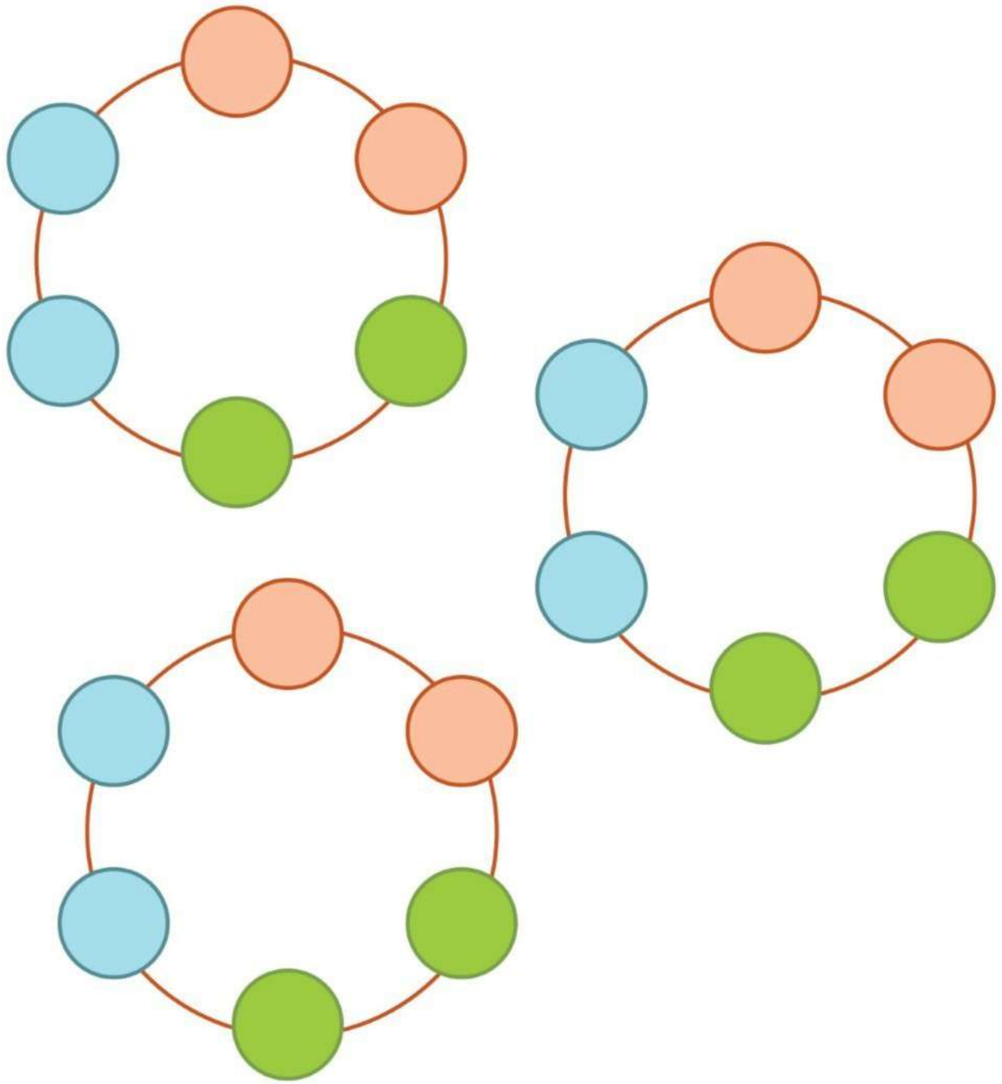
CL = Q = 2 $\frac{3+1}{2} = 2$

Cassandra Write Path

country	id
UK	3
UK	4
UK	2
UK	5
UK	1



Consistency with Multiple Data Centers



EACH_QUORUM

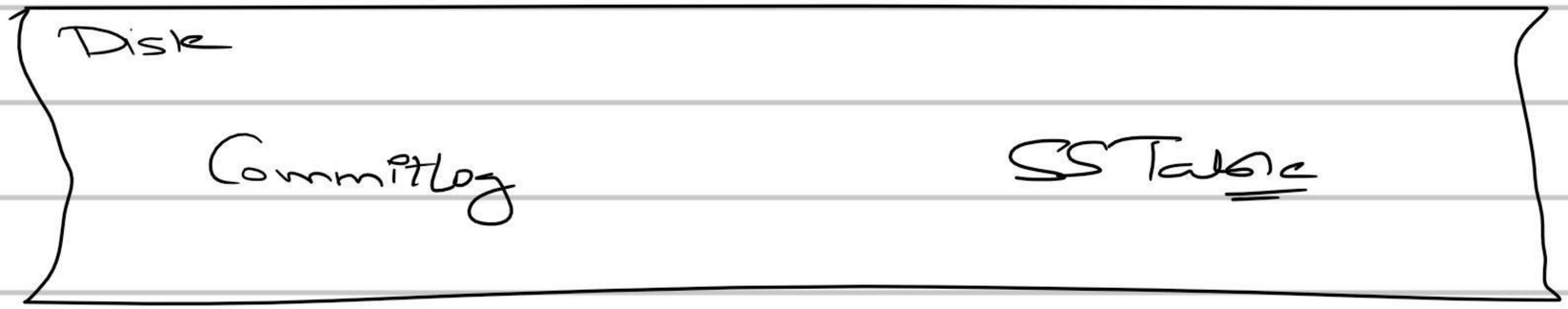
LOCAL_QUORUM

LOCAL_ONE

Cassandra Write Path

Spinning Disk

SSD



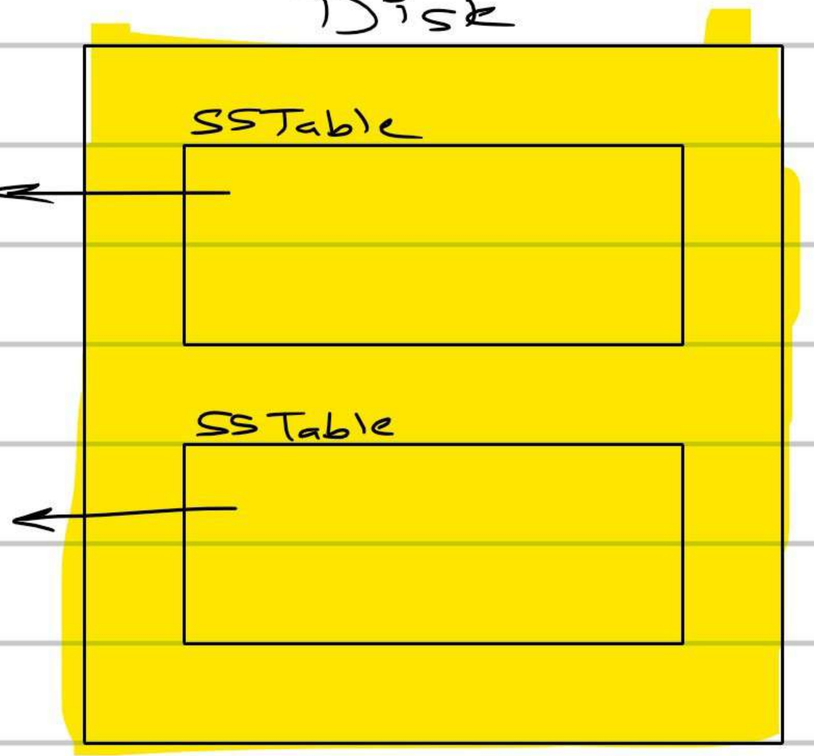
Cassandra Read Path

UK

memTable
- UK 3 (100)
- UK 10 (110)
- UK 7 (94)

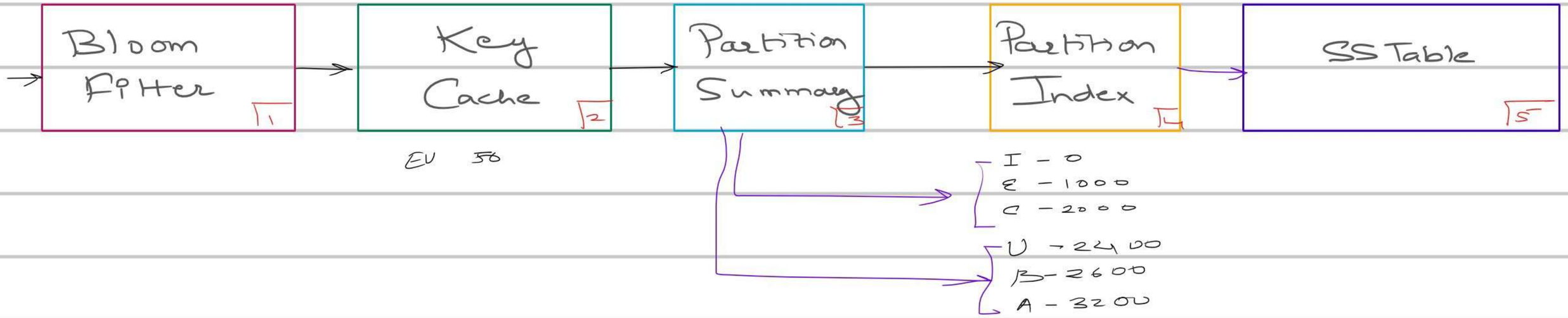
SS
- ~~UK 10 (65)~~
- UK 8 (83)

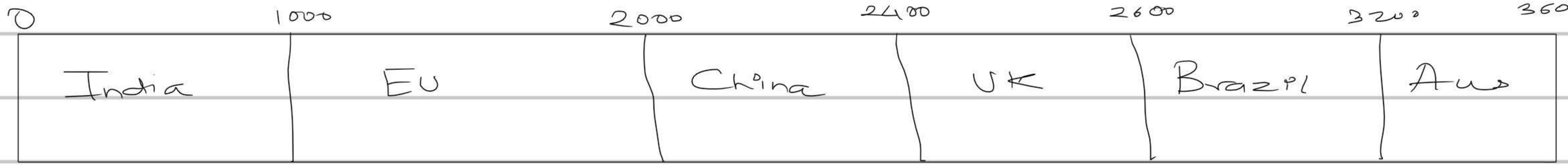
SS
- ~~UK 3 (94)~~
- UK 4 (79)



Latest Data is always in MemTable

UK 5





Bloom filter → quickly tells if data might be in SSTable or not (not 100% accurate, but fast).

Key Cache → If row key was accessed recently it will cache the exact disk location for it

Partition Index → notifies the start byte ~~of~~ location of each partition

Partition summary → is like a partition index — for partition index

Time Series Data

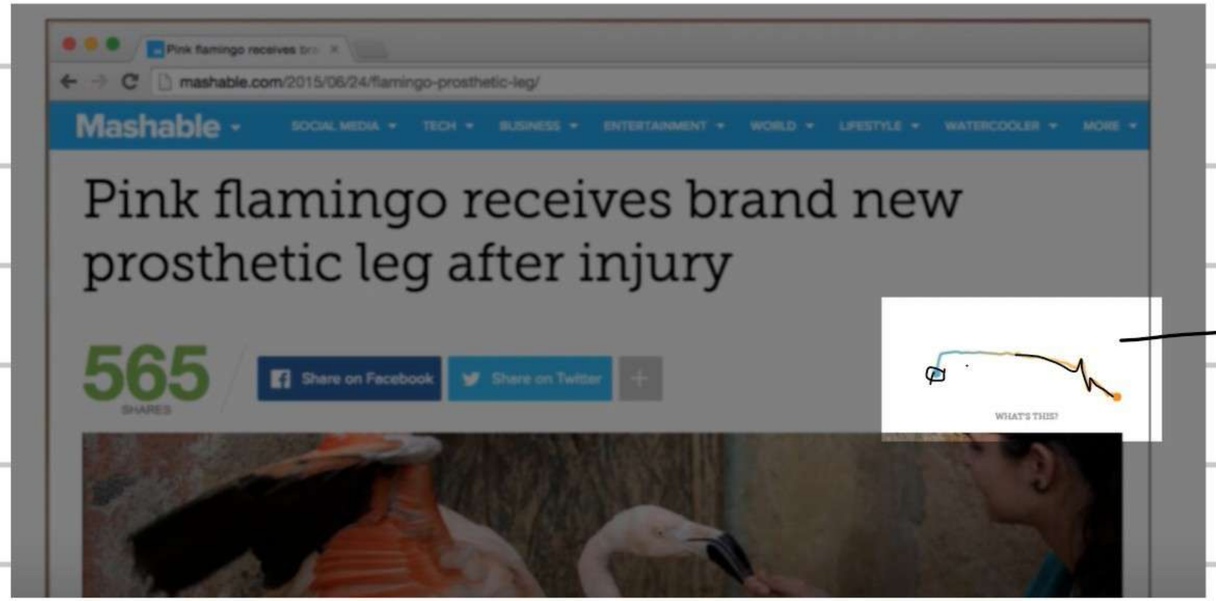
Time UUID

45b94a50-12e5-11e5-9114-091830ac5256

→ The number of 100 ns intervals since UUID epoch

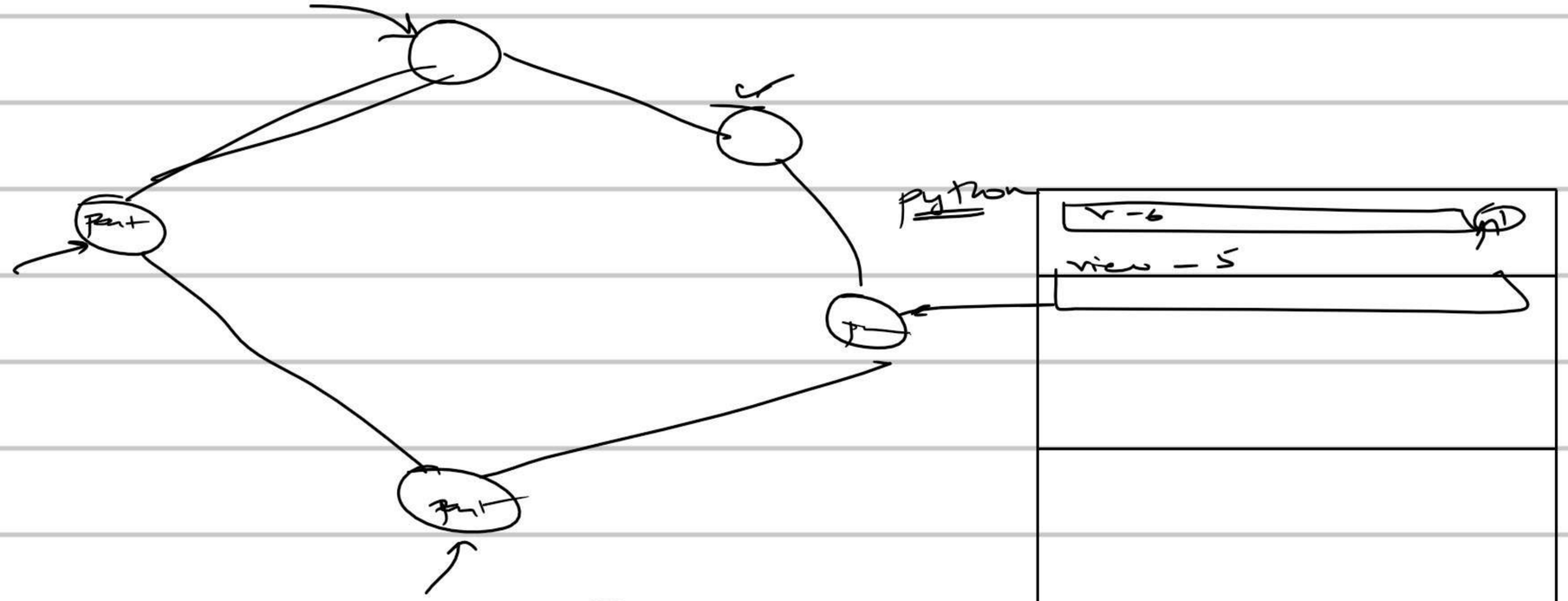
→ MAC address

→ Clock sequence number to prevent duplicates



WCL - 1

RF = 3



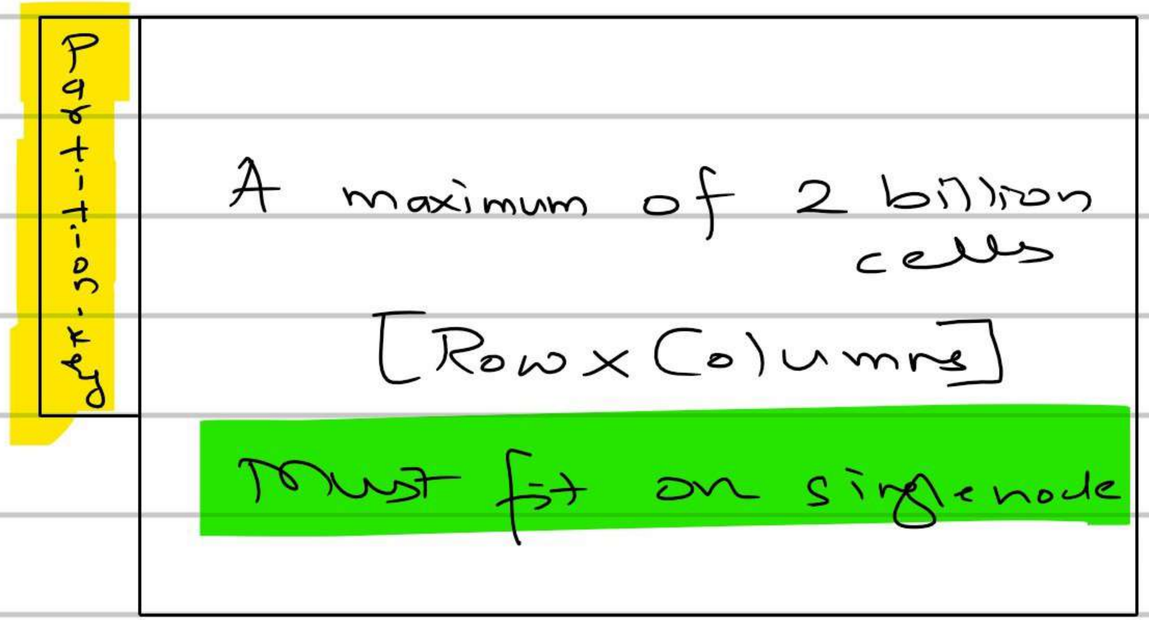
map

alter

last_login → device identification

tuple < timestamp, inet

map < text, frozen tuple < timestamp, inet >> >



sensor → you cannot use ttl
→ data is coming in every 5 seconds

Restructure course_page_views in such a way that PK won't make partition go out of memory