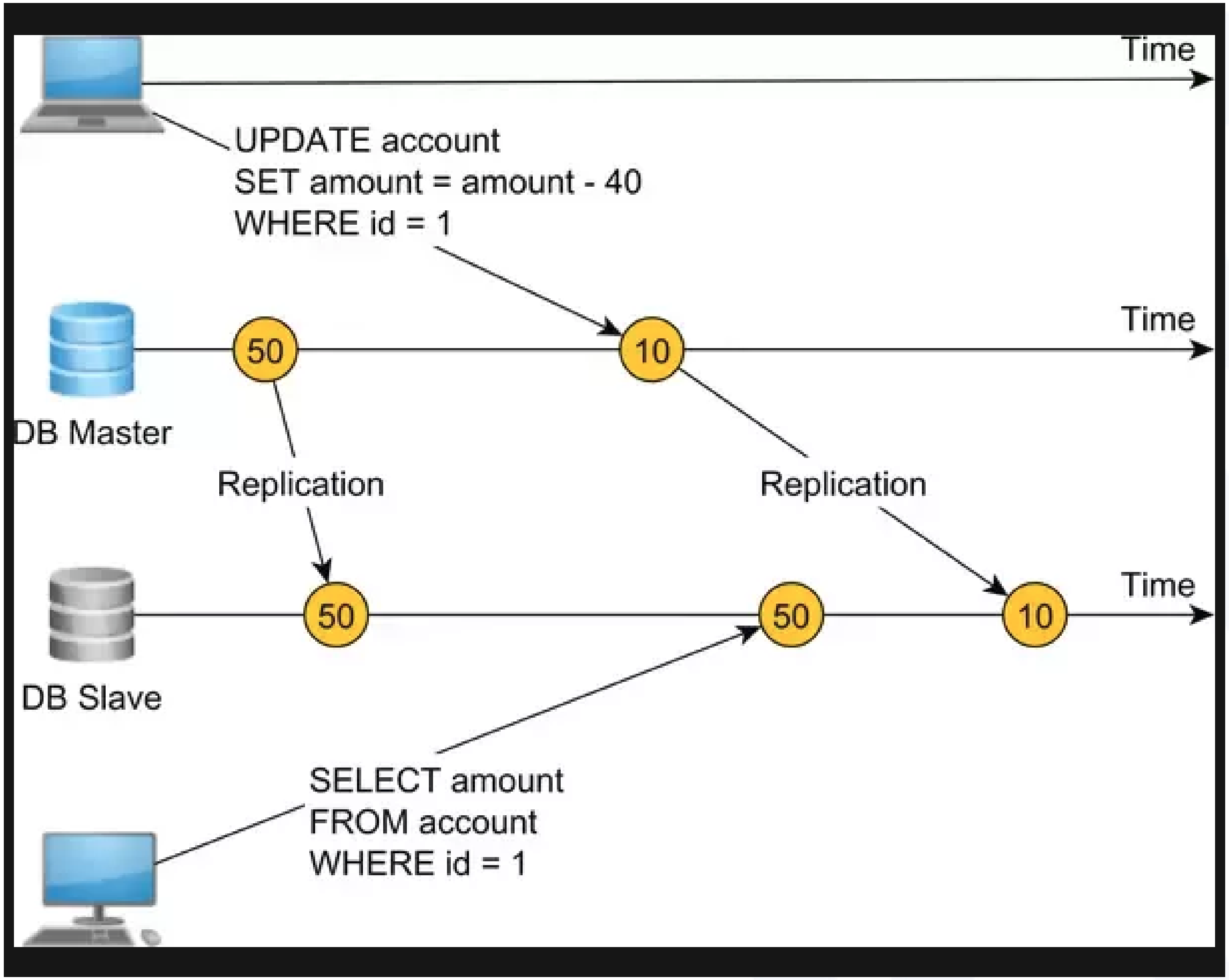
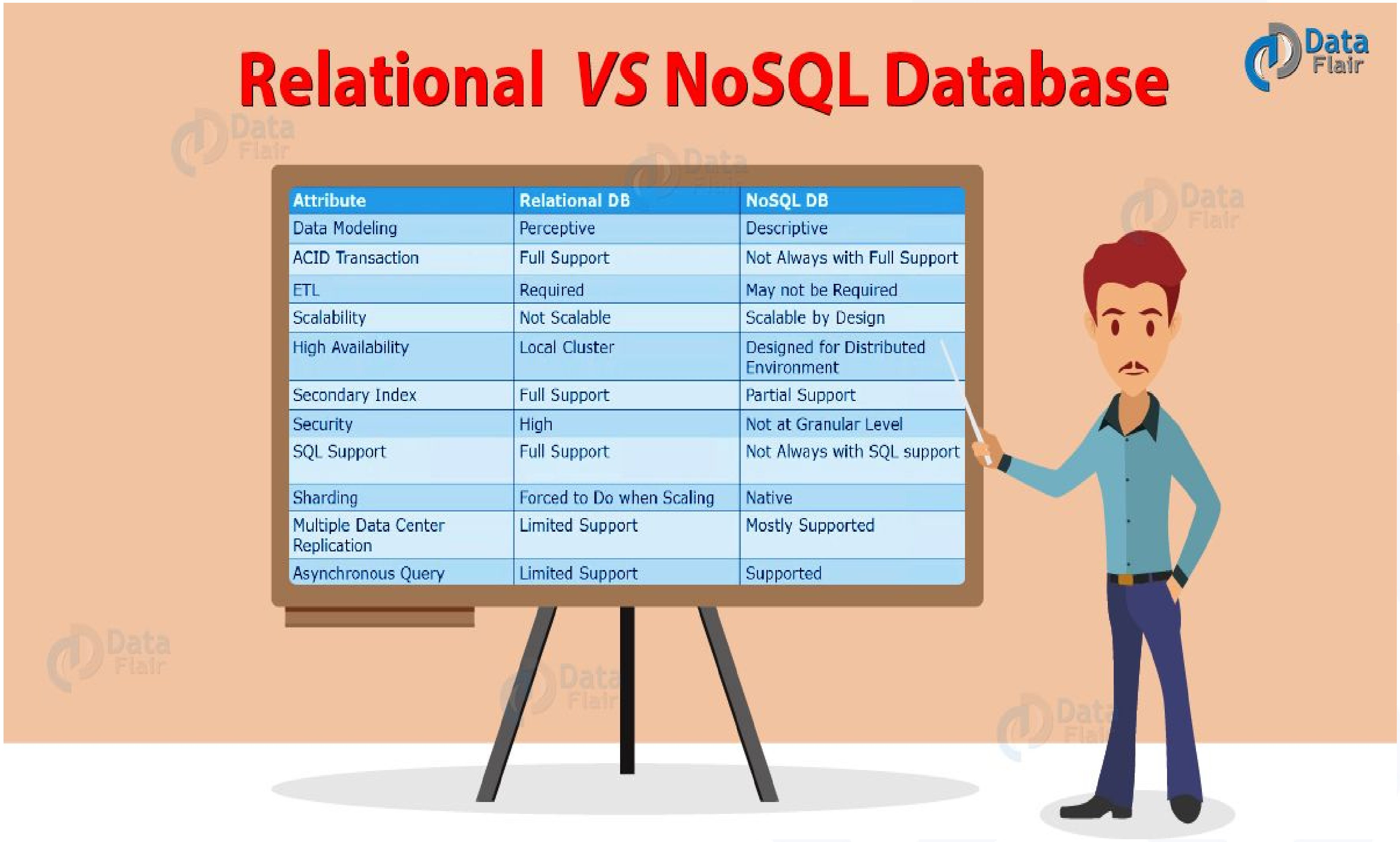
CONSISTÊNCIA e REPLICAÇÃO

santiago.robles@unimetrocamp.edu.br

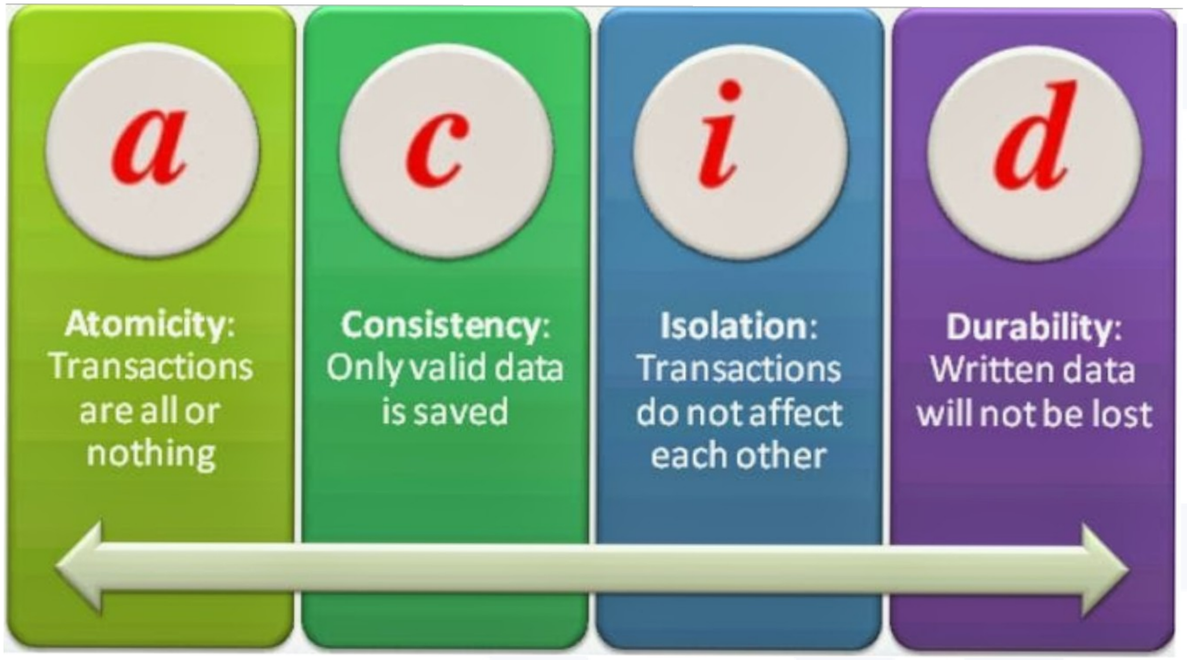
outubro 2019


Time
UPDATE account
SET amount = amount - 40
WHERE id = 1
Time
50
10
DB Master
Replication
Replication
Time
50
50
10
DB Slave
SELECT amount
FROM account
WHERE id = 1



Relational *VS* NoSQL Database

Attribute	Relational DB	NoSQL DB
Data Modeling	Perceptive	Descriptive
ACID Transaction	Full Support	Not Always with Full Support
ETL	Required	May not be Required
Scalability	Not Scalable	Scalable by Design
High Availability	Local Cluster	Designed for Distributed Environment
Secondary Index	Full Support	Partial Support
Security	High	Not at Granular Level
SQL Support	Full Support	Not Always with SQL support
Sharding	Forced to Do when Scaling	Native
Multiple Data Center Replication	Limited Support	Mostly Supported
Asynchronous Query	Limited Support	Supported


a
c
i
d
Atomicity: Transactions are all or nothing
Consistency: Only valid data is saved
Isolation: Transactions do not affect each other
Durability: Written data will not be lost

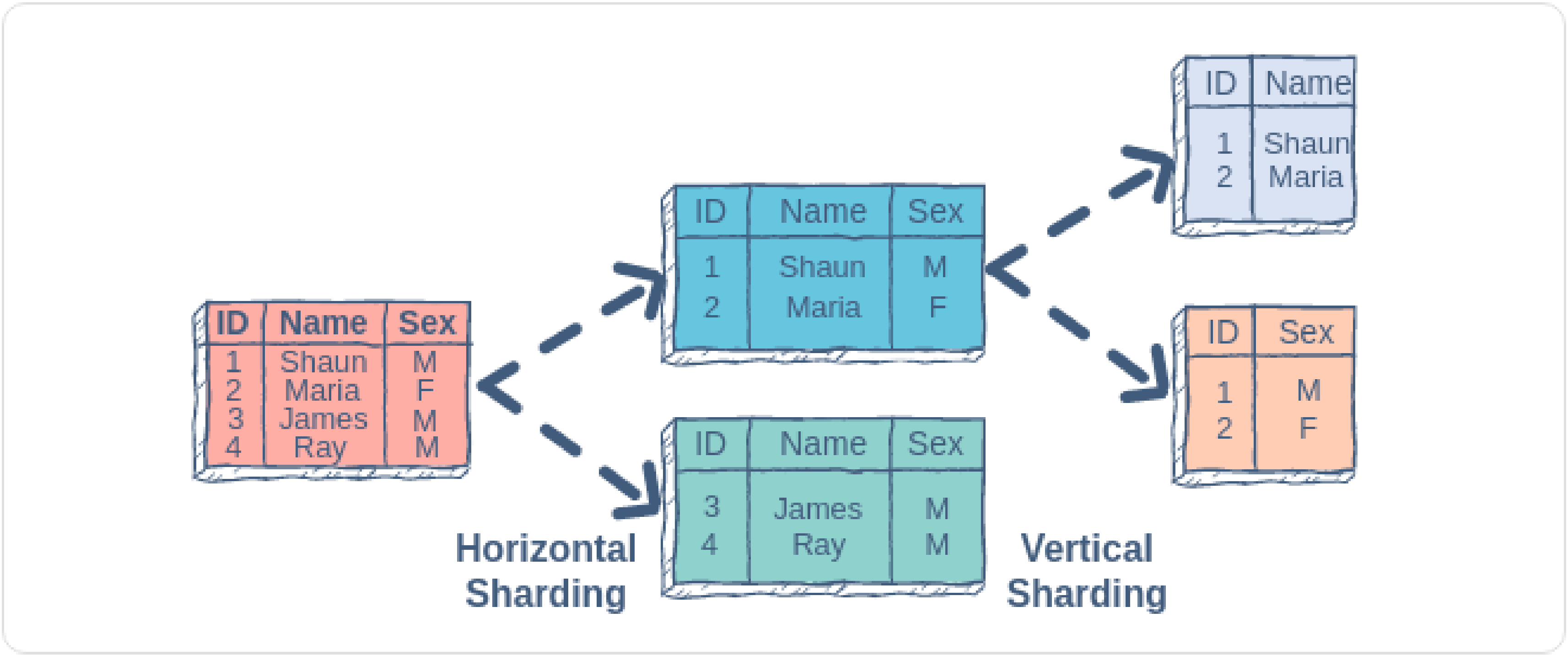
Atomicidade: Em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômicas.

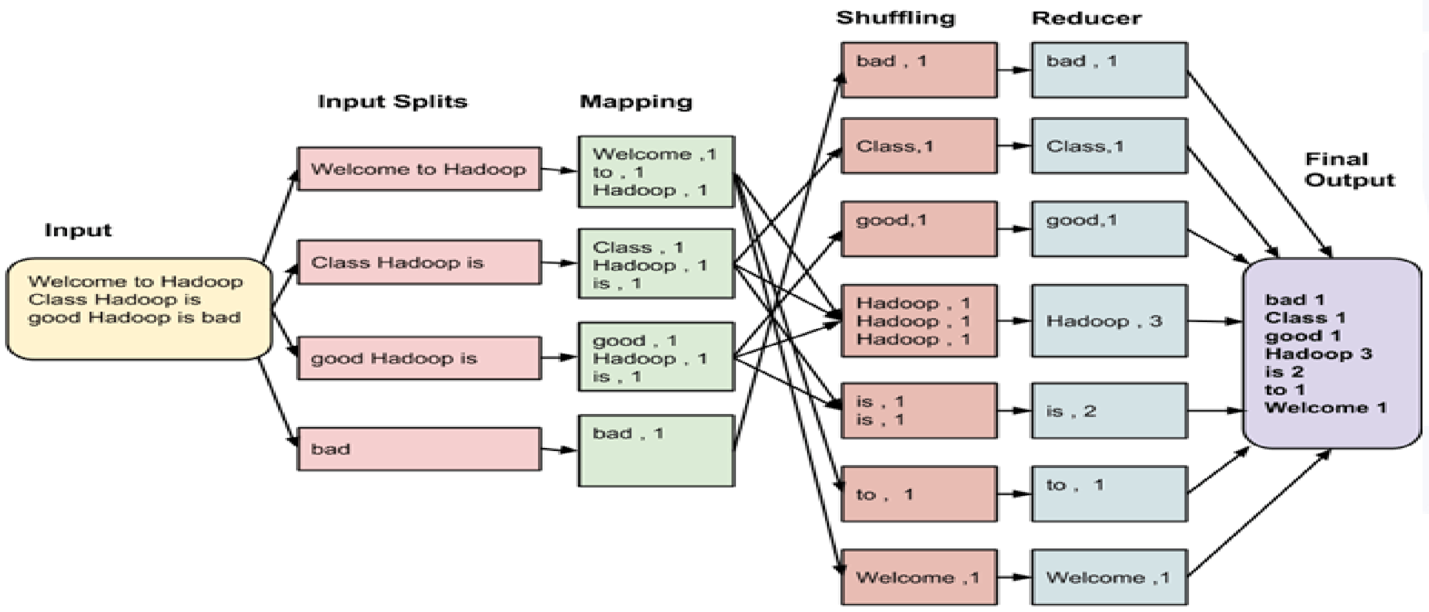
Consistência: A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.

Isolamento: Uma transação em andamento mas ainda não validada deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.

Durabilidade: Dados validados são registados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto.

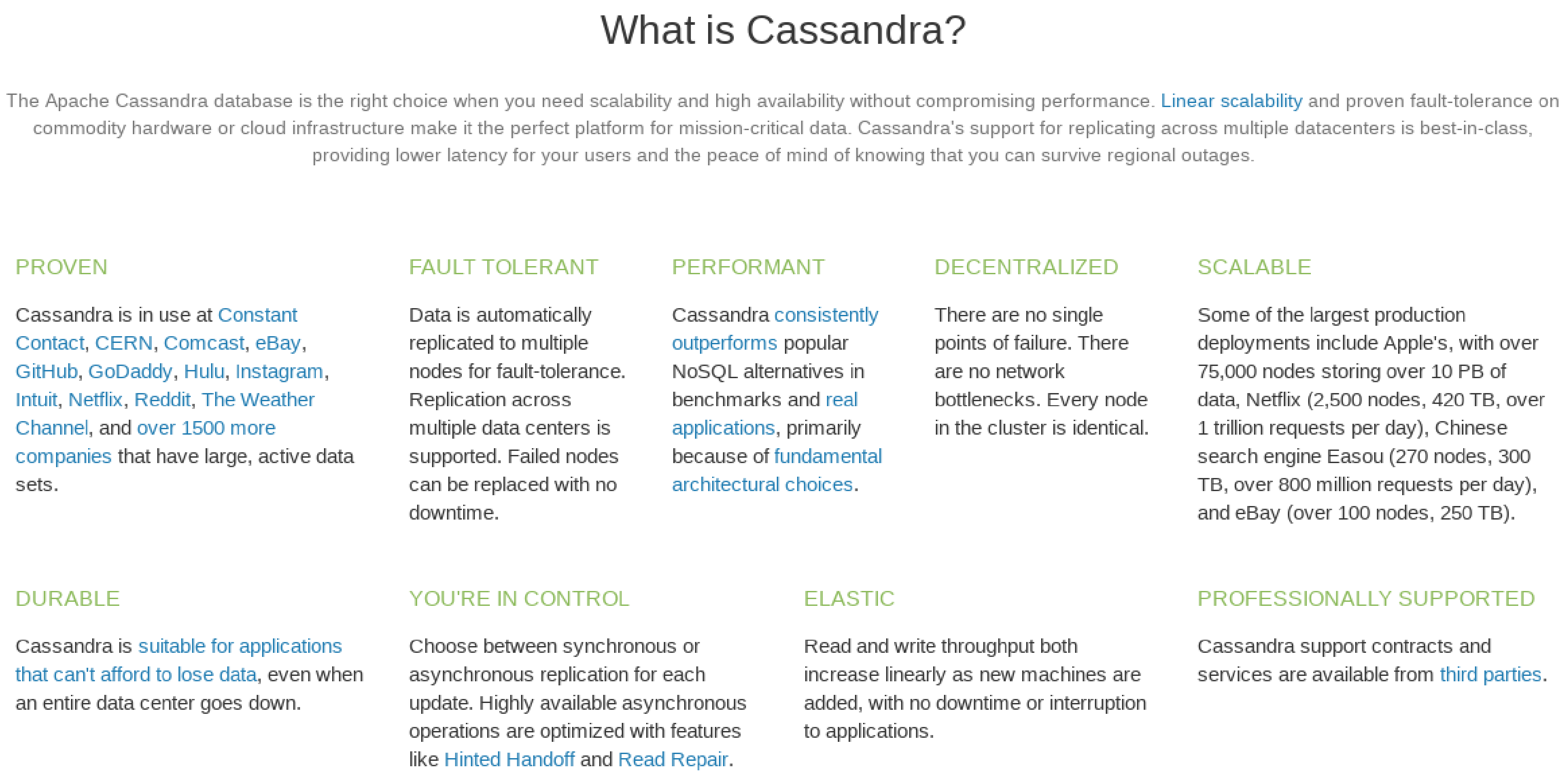
SHARDING


ID | Name | Sex
1 | Shaun | M
2 | Maria | F
3 | James | M
4 | Ray | M
Horizontal Sharding
ID | Name | Sex
1 | Shaun | M
2 | Maria | F
ID | Name | Sex
3 | James | M
4 | Ray | M
Vertical Sharding
ID | Name
1 | Shaun
2 | Maria
ID | Sex
1 | M
2 | F


Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Input Splits
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Mapping
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
Shuffling
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
Reducer
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
Final Output
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

- MapReduce é uma técnica de tratamento e um modelo de programa de computação distribuída baseada em java.
- O algoritmo contém duas funções importantes: Mapa e reduzir.
- Mapa leva um conjunto de dados e converte-a em outro conjunto de dados, onde cada um dos elementos são decompostos em Tuplas (pares chave/valor).
- Reduzir, a partir de um mapa como uma entrada e combina esses dados Tuplas em um conjunto menor de Tuplas

http://cassandra.apache.org/



What is Cassandra?

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

PROVEN

Cassandra is in use at Constant Contact, CERN, Comcast, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Reddit, The Weather Channel, and over 1500 more companies that have large, active data sets.

FAULT TOLERANT

Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

PERFORMANT

Cassandra consistently outperforms popular NoSQL alternatives in benchmarks and real applications, primarily because of fundamental architectural choices.

DECENTRALIZED

There are no single points of failure. There are no network bottlenecks. Every node in the cluster is identical.

SCALABLE

Some of the largest production deployments include Apple's, with over 75,000 nodes storing over 10 PB of data, Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day), Chinese search engine Easou (270 nodes, 300 TB, over 800 million requests per day), and eBay (over 100 nodes, 250 TB).

DURABLE

Cassandra is suitable for applications that can't afford to lose data, even when an entire data center goes down.

YOU'RE IN CONTROL

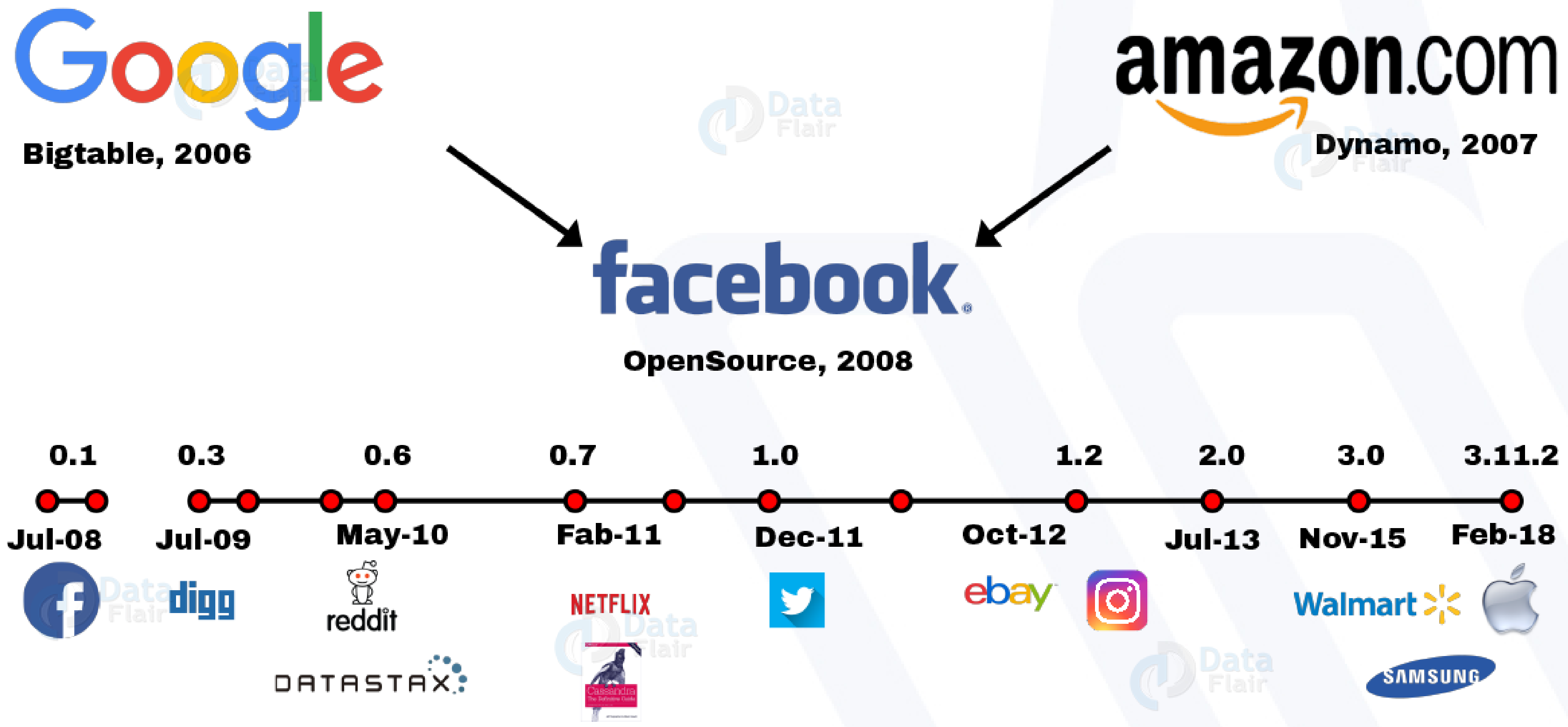
Choose between synchronous or asynchronous replication for each update. Highly available asynchronous operations are optimized with features like Hinted Handoff and Read Repair.

ELASTIC

Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.

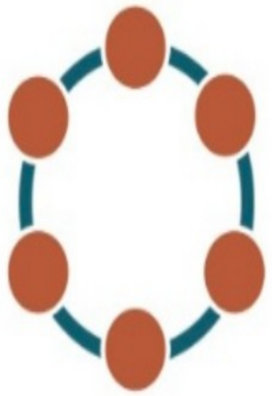
PROFESSIONALLY SUPPORTED

Cassandra support contracts and services are available from third parties.


Bigtable, 2006
Dynamo, 2007
OpenSource, 2008
0.1
0.3
0.6
0.7
1.0
1.2
2.0
3.0
3.11.2
Jul-08
Jul-09
May-10
Fab-11
Dec-11
Oct-12
Jul-13
Nov-15
Feb-18

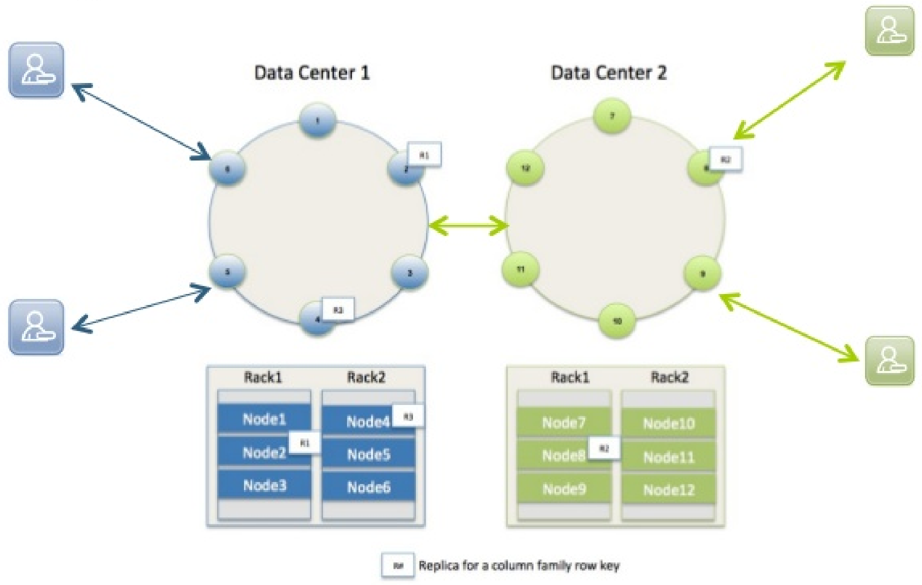
- Peer to peer communication protocol.
- Nodes are arranged in ring format.
- Data is replicated to multiple nodes.
- Nodes periodically exchange info. they have.
- Nodes also exchange their own info.
- Each message has its associated version.
- No master-slave concept, and hence no single point of failure.

- A ring has several nodes.
- Each node is assigned a Partition value.
- Data processing is based on the Partition Key.
- When a client makes a request to a node, it becomes the coordinator for that request.
- The coordinator determines which node in the ring should process upon that request.



Reading and Writing in Cassandra

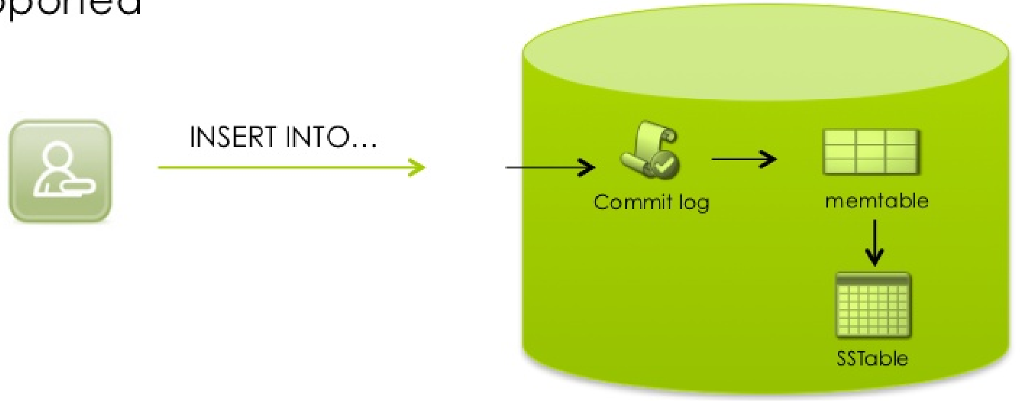
Cassandra is a peer-to-peer, read/write anywhere architecture, so any user can connect to any node in any data center and read/write the data they need, with all writes being partitioned and replicated for them automatically throughout the cluster.



www.datastax.com

Writes in Cassandra

- Data is first written to a commit log for durability
- Then written to a memtable in memory
- Once the memtable becomes full, it is flushed to an SSTable (sorted strings table)
- Writes are atomic at the row level; all columns are written or updated, or none are. RDBMS-styled transactions are not supported

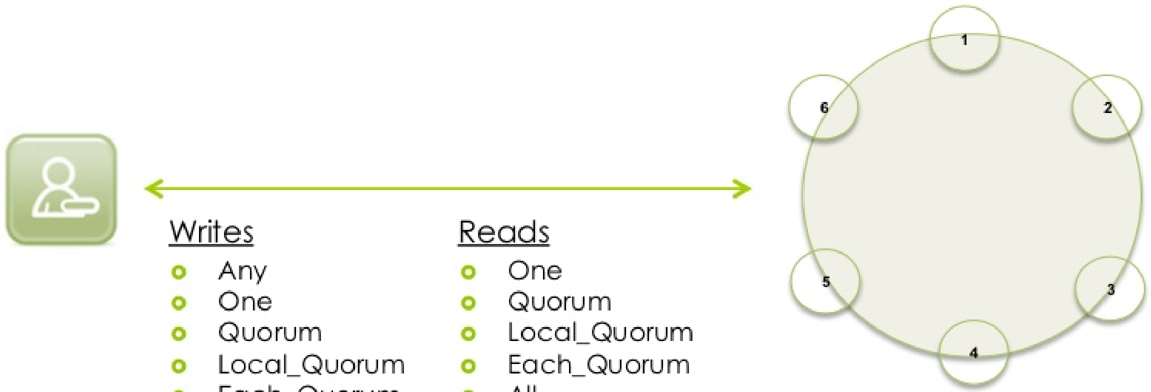


Cassandra is known for being the fastest database in the industry where write operations are concerned.

www.datastax.com

Tunable Data Consistency

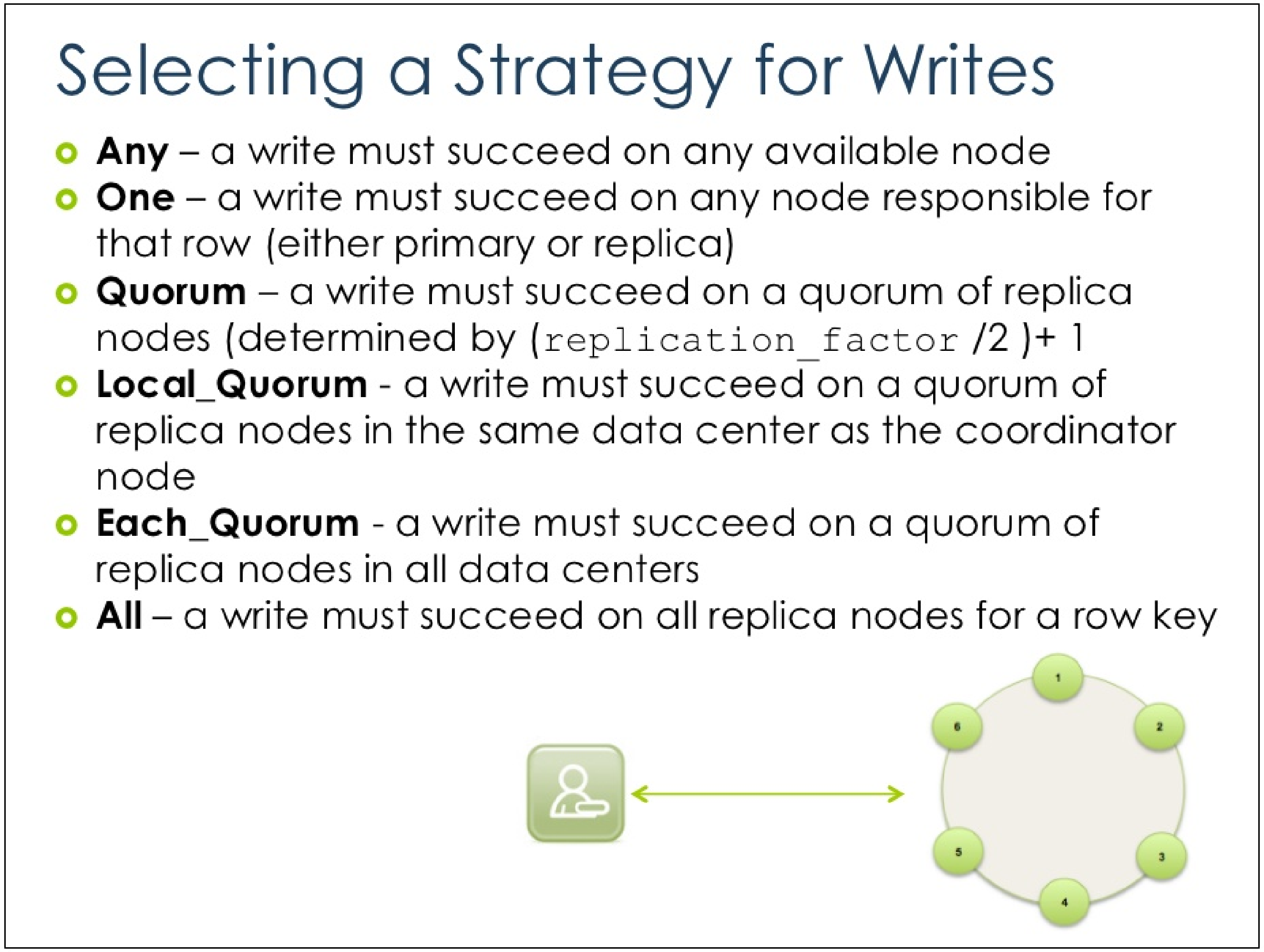
- Choose between strong and eventual consistency (All to any node responding) depending on the need
- Can be done on a per-operation basis, and for both reads and writes
- Handles Multi-data center operations



Writes
- Any
- One
- Quorum
- Local_Quorum
- Each_Quorum
- All

Reads
- One
- Quorum
- Local_Quorum
- Each_Quorum
- All

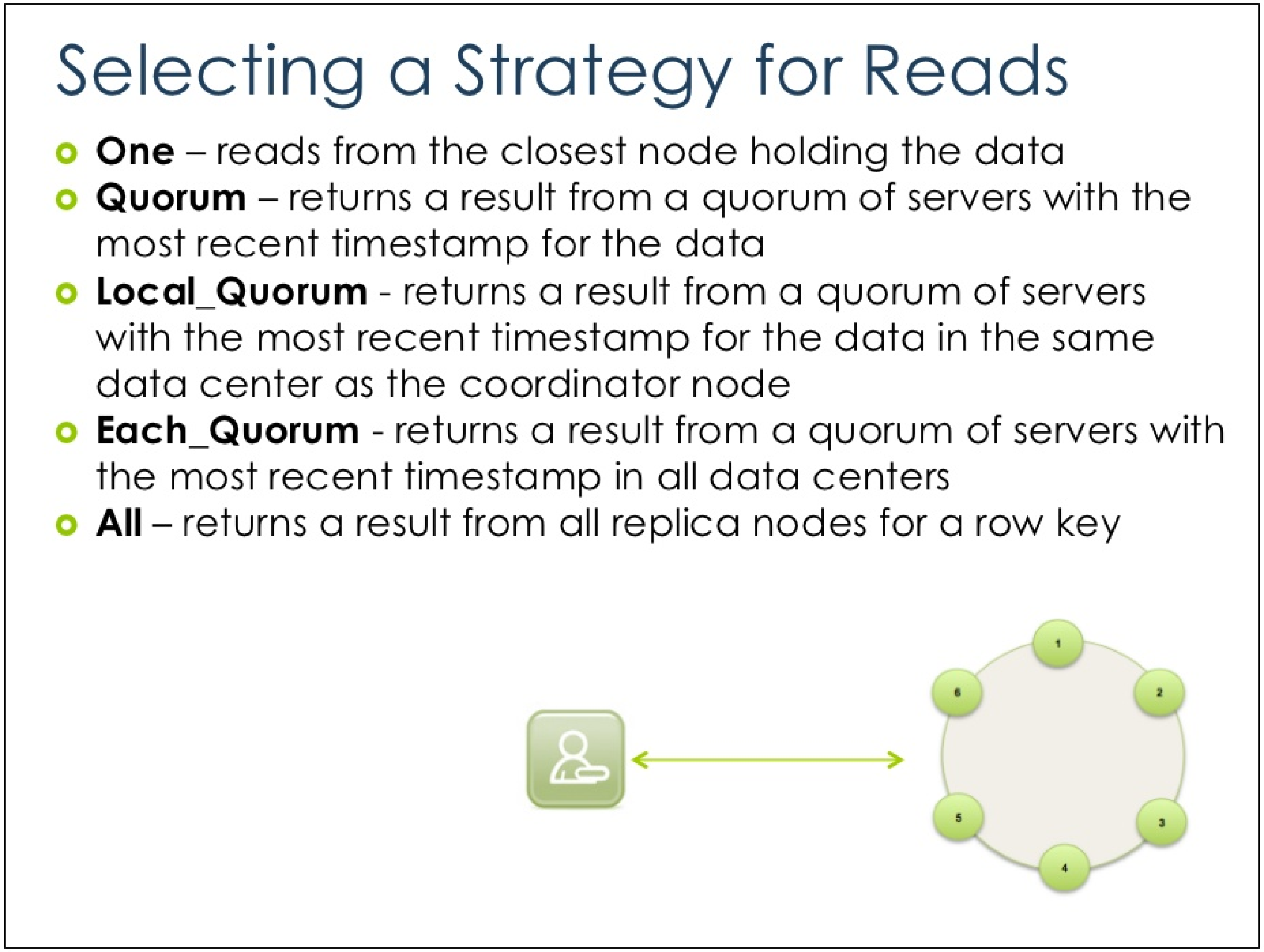
www.datastax.com



Selecting a Strategy for Writes

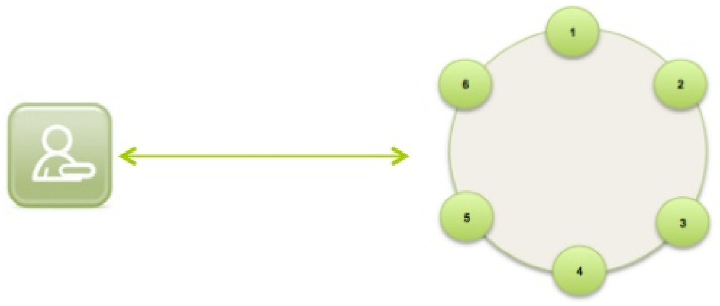
- **Any** – a write must succeed on any available node
- **One** – a write must succeed on any node responsible for that row (either primary or replica)
- **Quorum** – a write must succeed on a quorum of replica nodes (determined by (`replication_factor` /2)+ 1
- **Local_Quorum** - a write must succeed on a quorum of replica nodes in the same data center as the coordinator node
- **Each_Quorum** - a write must succeed on a quorum of replica nodes in all data centers
- **All** – a write must succeed on all replica nodes for a row key

www.datastax.com



Selecting a Strategy for Reads

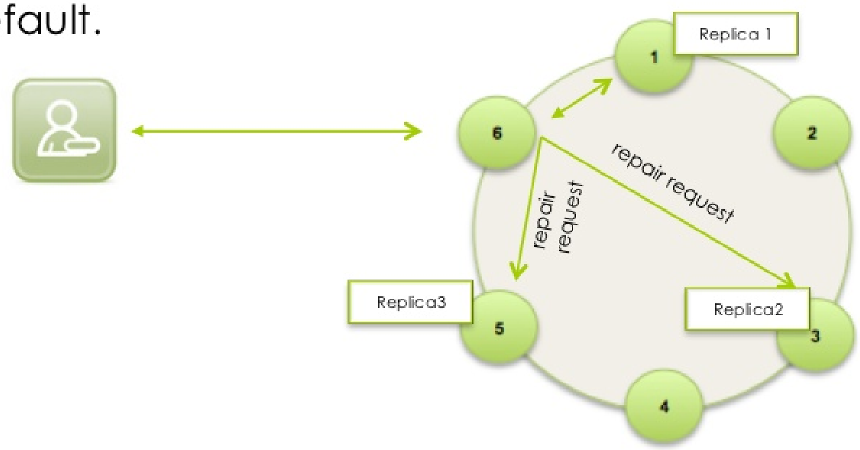
- **One** – reads from the closest node holding the data
- **Quorum** – returns a result from a quorum of servers with the most recent timestamp for the data
- **Local_Quorum** - returns a result from a quorum of servers with the most recent timestamp for the data in the same data center as the coordinator node
- **Each_Quorum** - returns a result from a quorum of servers with the most recent timestamp in all data centers
- **All** – returns a result from all replica nodes for a row key



www.datastax.com

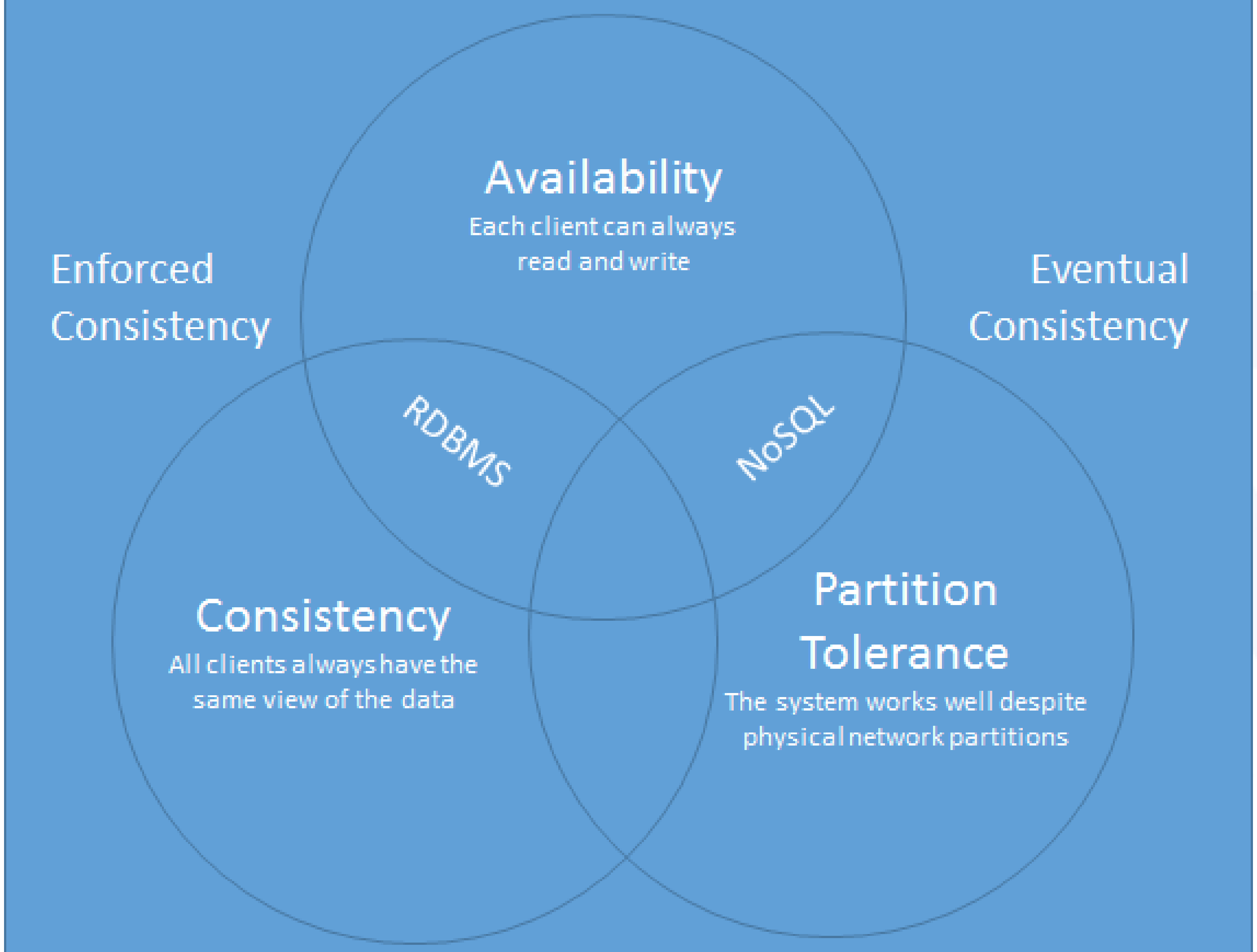
Read Repair

- Cassandra ensures that frequently-read data remains consistent
- When a read is done, the coordinator node compares the data from all the remaining replicas that own the row in the background, and if they are inconsistent, issues writes to the out-of-date replicas to update the row to reflect the most recently written values.
- Read repair can be configured per column family and is enabled by default.



www.datastax.com

CAP Theorem


Availability
Each client can always read and write
Enforced Consistency
Eventual Consistency
RDBMS
NoSQL
Consistency
All clients always have the same view of the data
Partition Tolerance
The system works well despite physical network partitions

WYDEN EDUCACIONAL

Faci facid FACIMP FBV fmf Faculdade Martha Falcão ISL UNIFAVIP UNI METROCAMP RUY BARBOSA ÁREA1 UniFBV UniFanor