HBase Programming

Compiled by Amit S Khedkar

Training kickoff

Course Structure

- Ability to use the command line

Pre-requisites

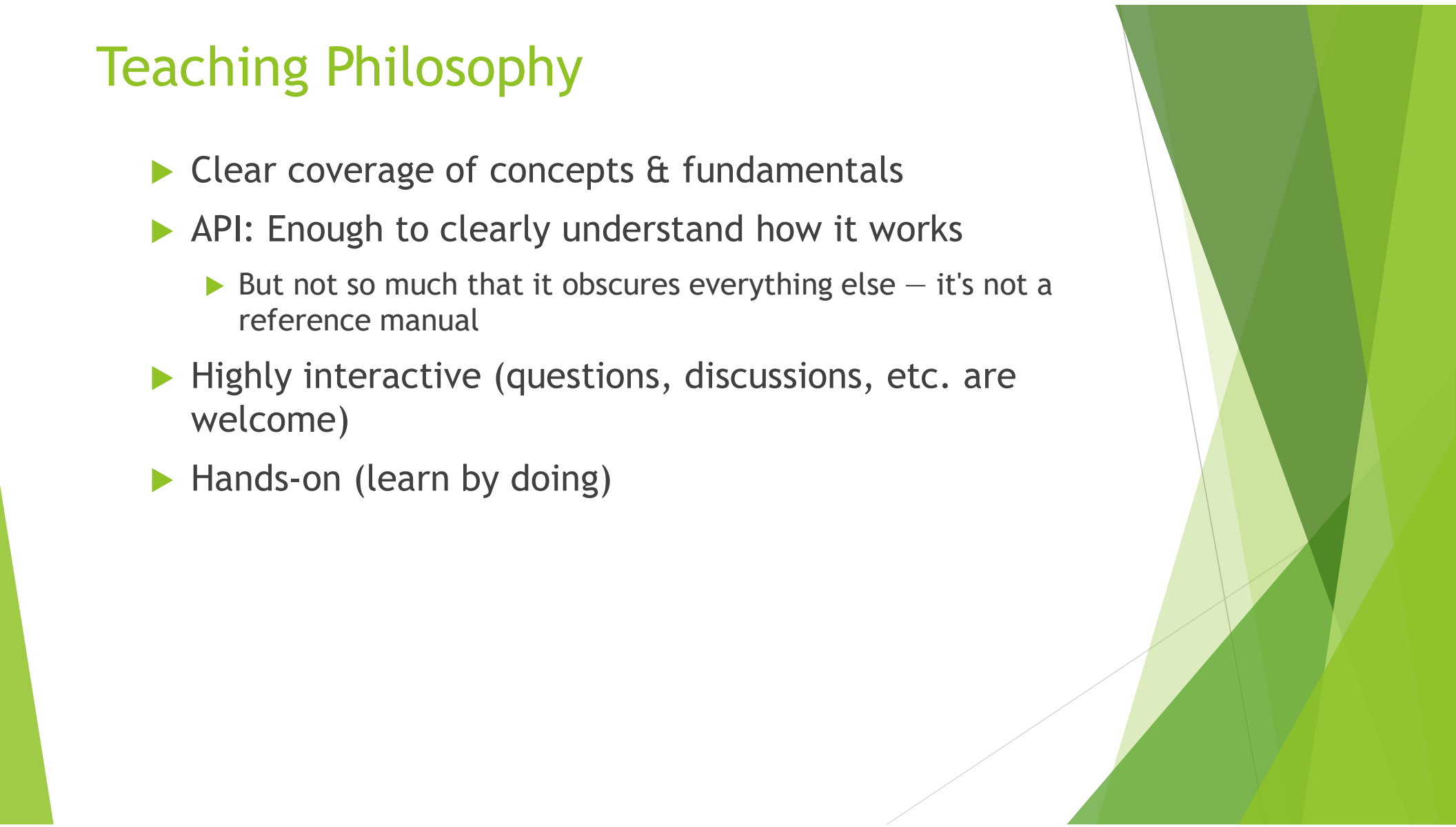
- Ability to use the command line
- Some knowledge of Java, or programming in general (We are going to use Java 8)
- Linux and Mac are strongly preferred
- Willingness to learn an awesome technology 

Who is this course for?

- You could be a **developer** and you want to learn how to write an application using Kafka
- You may be an **architect** who understand the role of Kafka in the Enterprise pipeline, some different architectures
- Or finally, you may be a **devops** who wants to understand how Kafka works with regards to topics, partitions, and multi broker setup

Teaching Philosophy

- Clear coverage of concepts & fundamentals
- API: Enough to clearly understand how it works
  - But not so much that it obscures everything else — it's not a reference manual
- Highly interactive (questions, discussions, etc. are welcome)
- Hands-on (learn by doing)



About You And Me

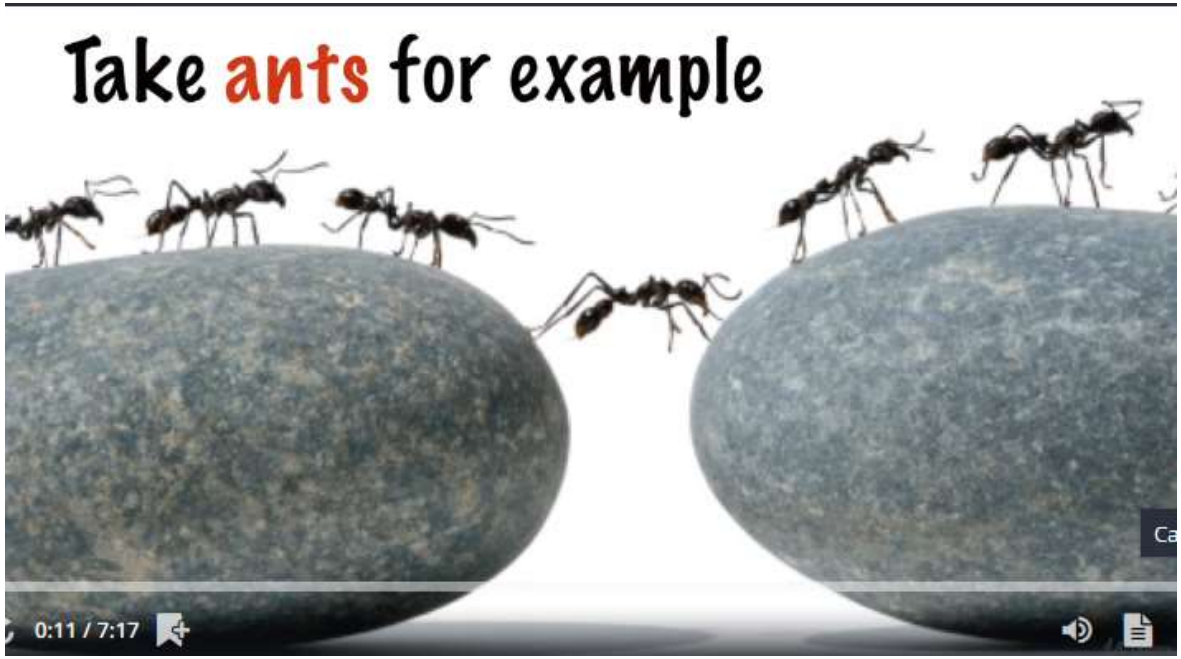
- About you
  - Your Name
  - Your background (developer, admin, manager …etc)
  - Technologies you are familiar with
  - Familiarity with Kafka (scale of 1 – 4 ; 1 – new, 4 – expert)
  - **Something non-technical about you!**
    (favorite ice cream flavor / hobby…etc)

HBase Introduction

HBase

Many technological innovations are inspired by nature

HBase



HBase

Take ants for example

Ants are fascinating
in many ways

HBase

Individually,
each ant is
seemingly
inconsequential

HBase

Collectively, they work together to accomplish complicated things

They rarely come alone. They march single file through miniscule cracks around windows or under doors, looking for crumbs, water or a warm place to make a new home. Often you'll see them trooping up your walls or across your counter, organized and on a mission. You have an ant invasion.

MARY JO DILONARDO, "What kind of ants are in my house?", *Mother Nature Network*, August 10, 2015

HBase

Together, ants behave
like a single entity

In pursuit of a
common goal

HBase


a single entity
common goal
Take one ant down;
Another comes up to
take it's place!

HBase


Distributed computing
is the idea of putting
many small and cheap
computers together

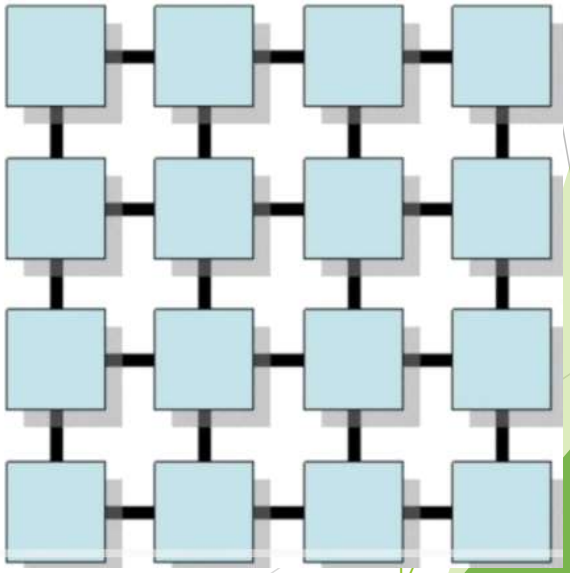
HBase

roblem with distributed computing

Distributed computing

is the idea of putting many small and cheap computers together

to accomplish **really complex** tasks

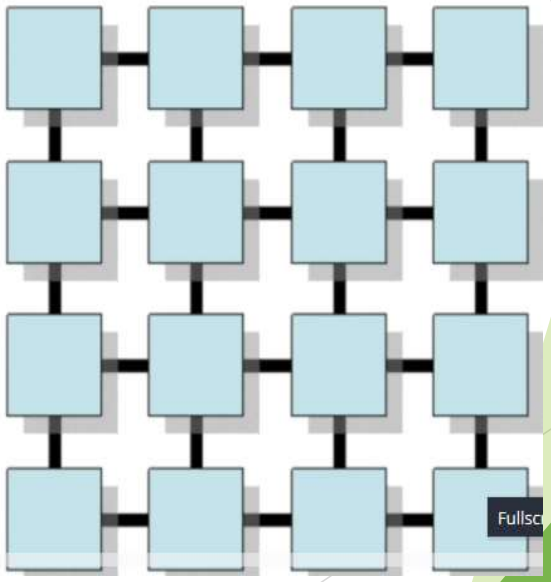


HBase

problem with distributed computing

Distributed computing

Each individual computer is called a node

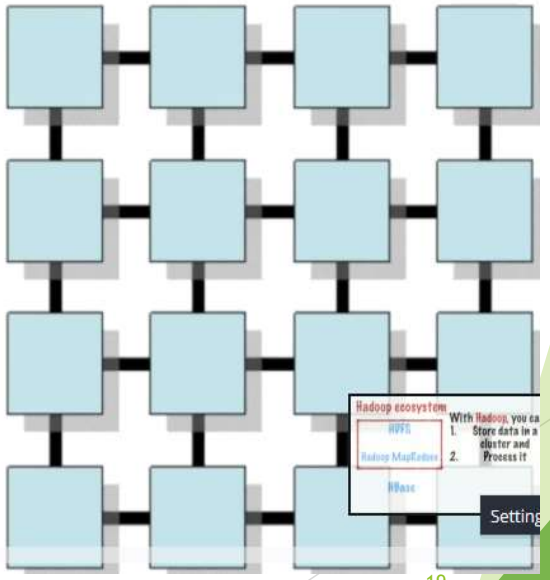


HBase

problem with distributed computing

Distributed computing

Together, all the nodes form a cluster



HBase

3. The problem with distributed computing

Distributed computing

Like ants,

Each individual node is pretty inconsequential

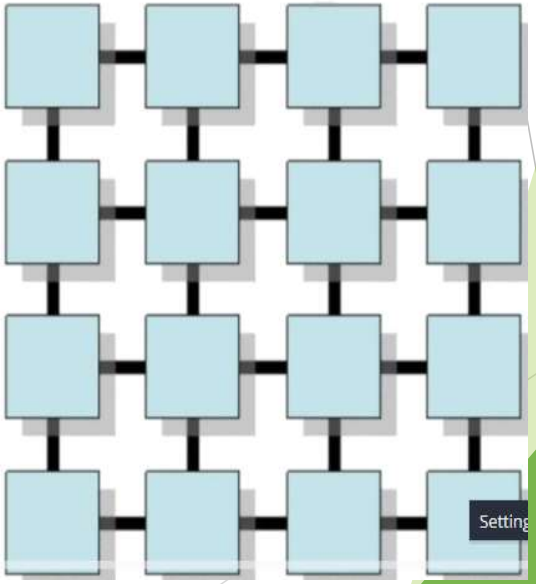
HBase

The problem with distributed computing

Distributed computing

Like ants,

Together these nodes act like a single entity with a common goal

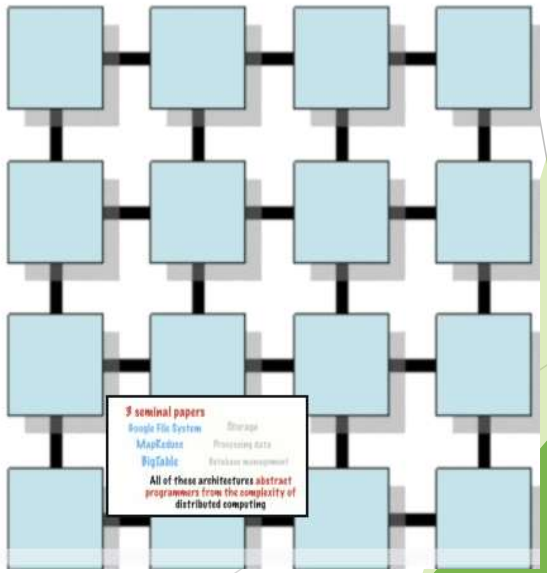


HBase

he problem with distributed computing

Vistributed computing

Why is this so cool?



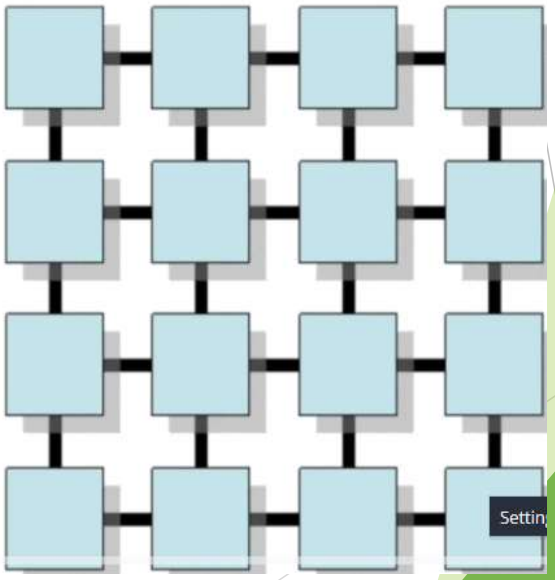
ind 5s

HBase

e problem with distributed computing

Distributed computing

The performance of this system scales linearly

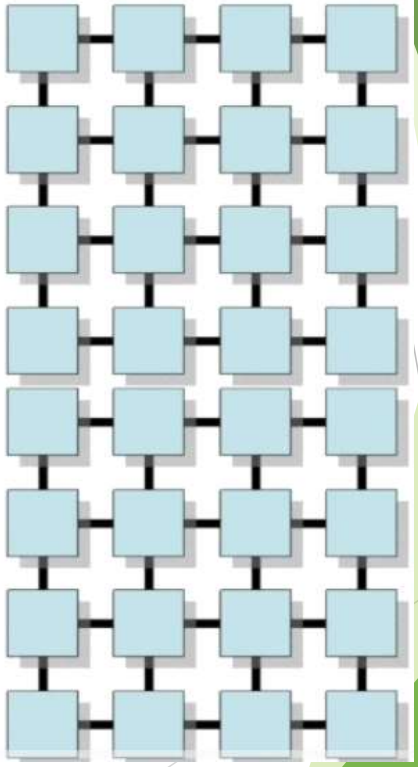


HBase

problem with distributed computing

Distributed computing

To double the perfomance, just double the number of nodes



HBase

e problem with distributed computing

Distributed computing

This is not true of individual computers

HBase

problem with distributed computing

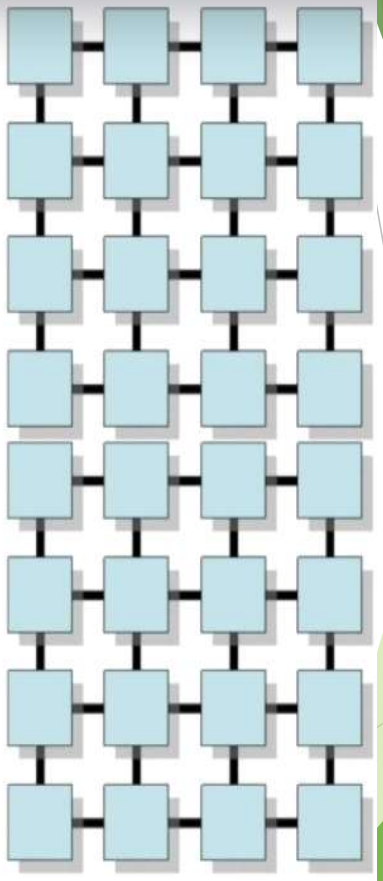
Distributed computing

A computer that's twice as expensive, will not necessarily give you twice the performance

HBase

e problem with distributed computing

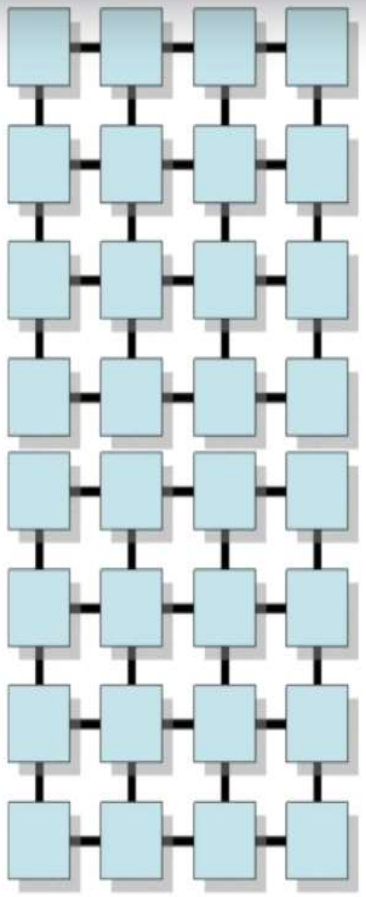
Distributed computing can get very complicated



HBase

Distributed computing can get very complicated

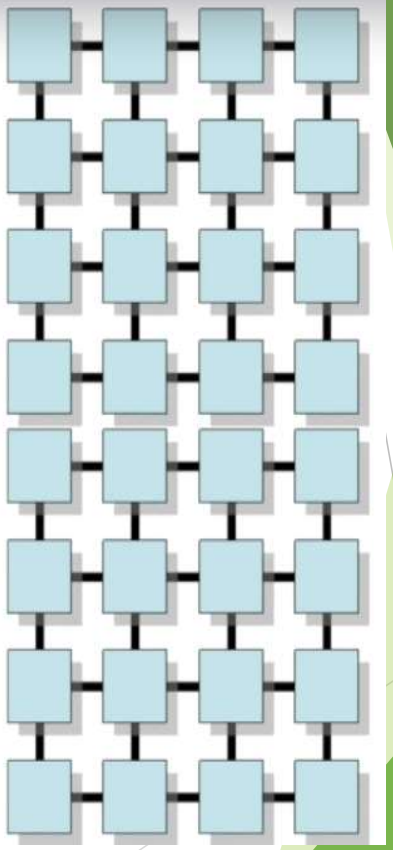
1. You have to manage resources and memory across multiple nodes



HBase

Distributed computing can get very complicated

manage resources and memory



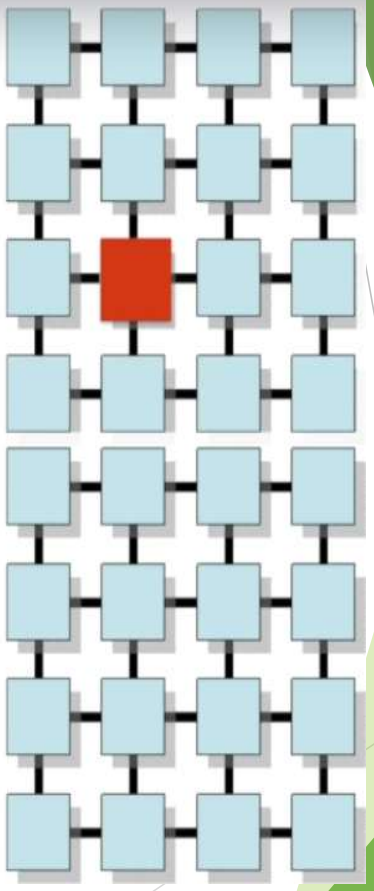
2. You have to co-ordinate and schedule tasks

HBase

Distributed computing can get very complicated

manage resources and memory

co-ordinate and schedule tasks



3. If one node goes down, the system should not be affected

(Just like with ants)

HBase

Distributed computing can get very complicated

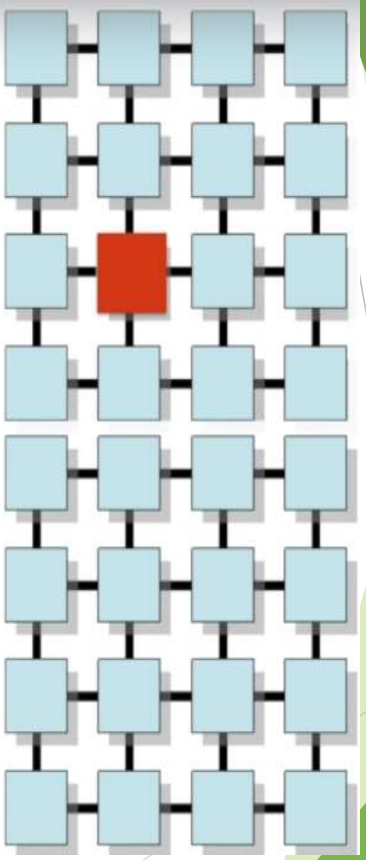
manage resources and memory

co-ordinate and schedule tasks

Fault Tolerance

3. If one node goes down, the system should not be affected

(Just like with ants)



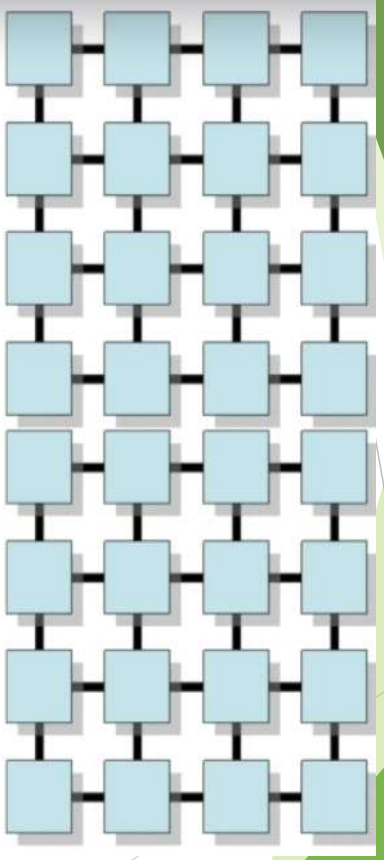
HBase

Distributed computing can get very complicated

manage resources and memory

co-ordinate and schedule tasks

Fault tolerance



Before the 2000s, all of these problems had to be taken care of by the programmer

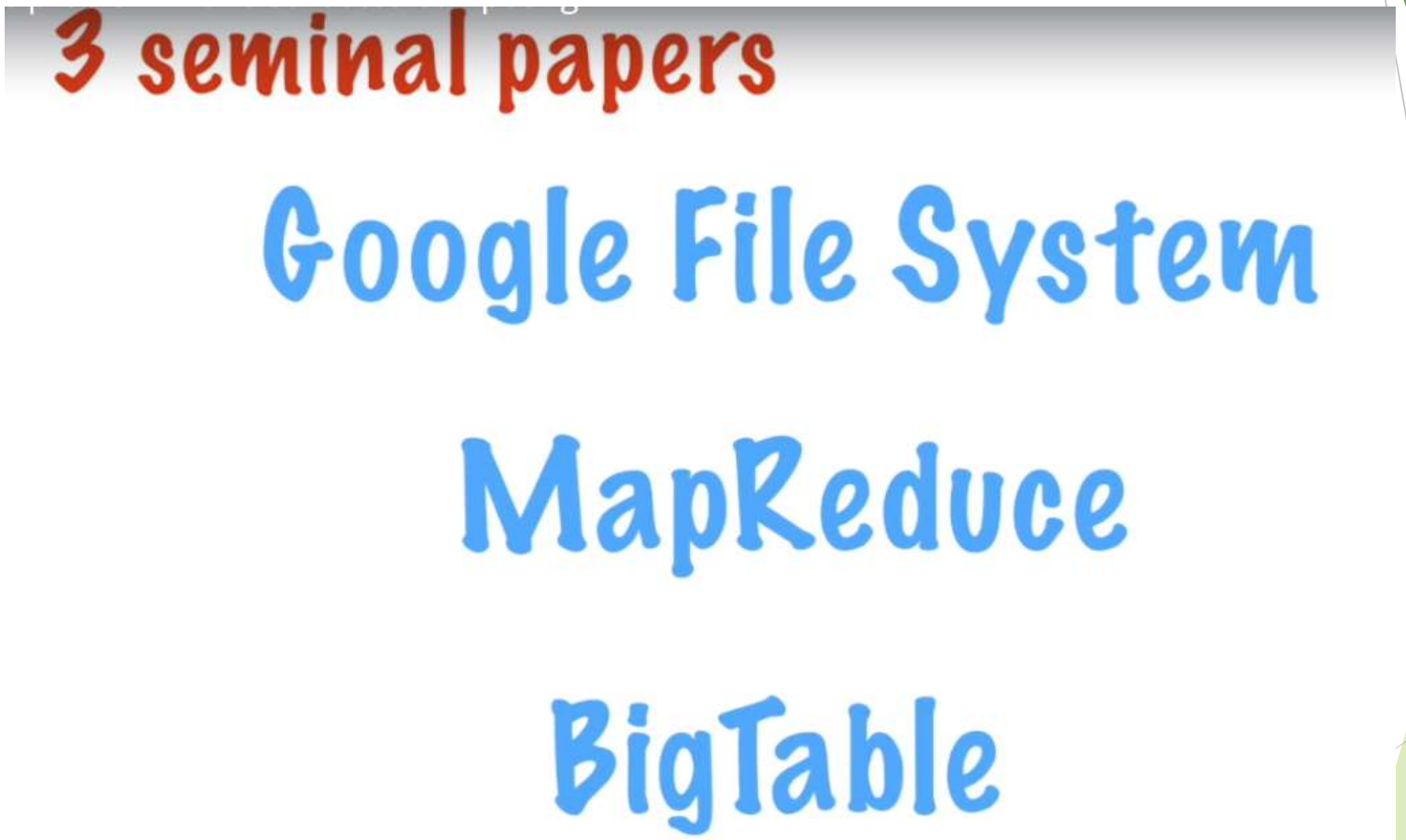
HBase

Between 2003 and 2006

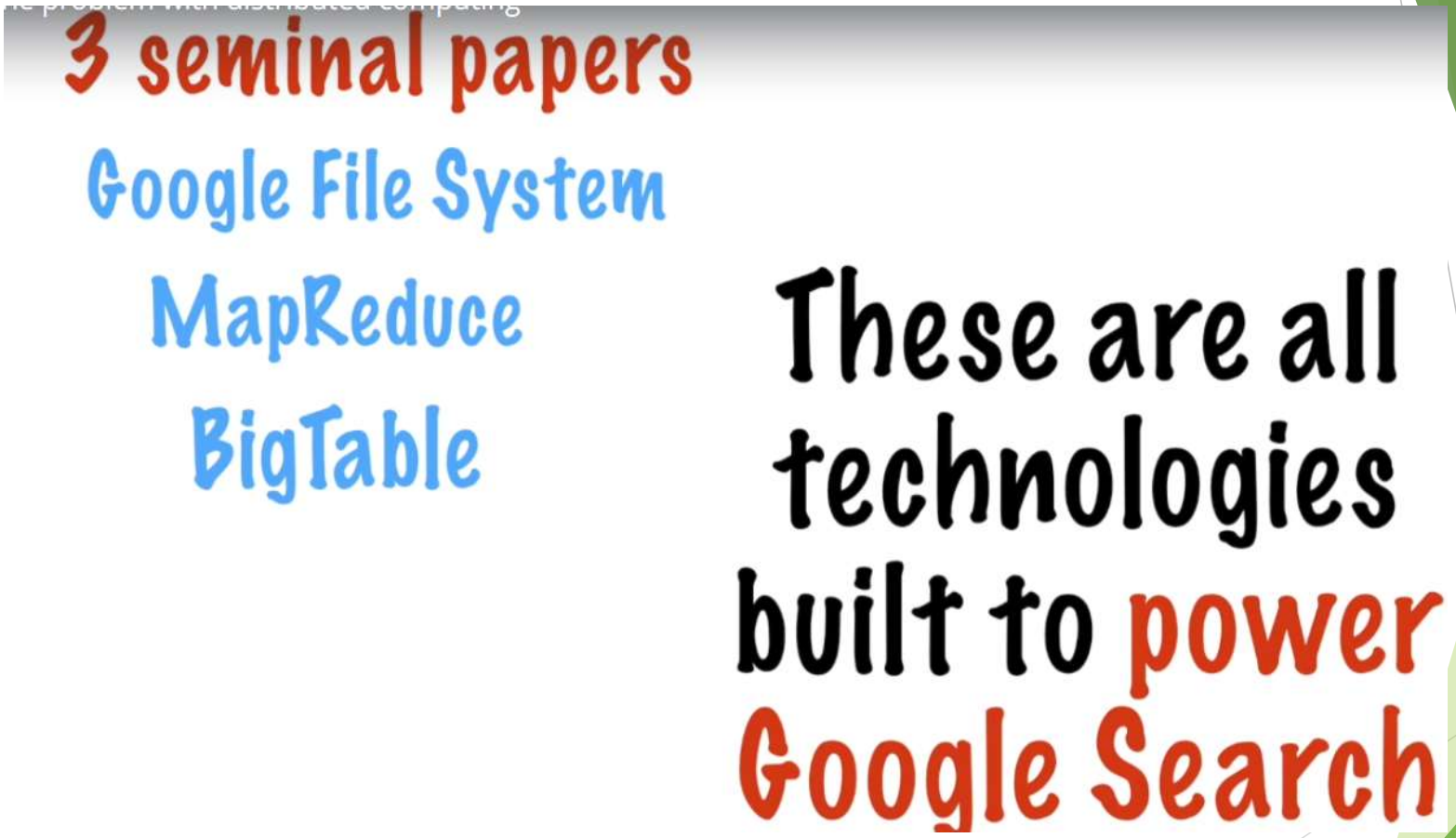
Google published 3 seminal papers

that completely changed the world of distributed computing

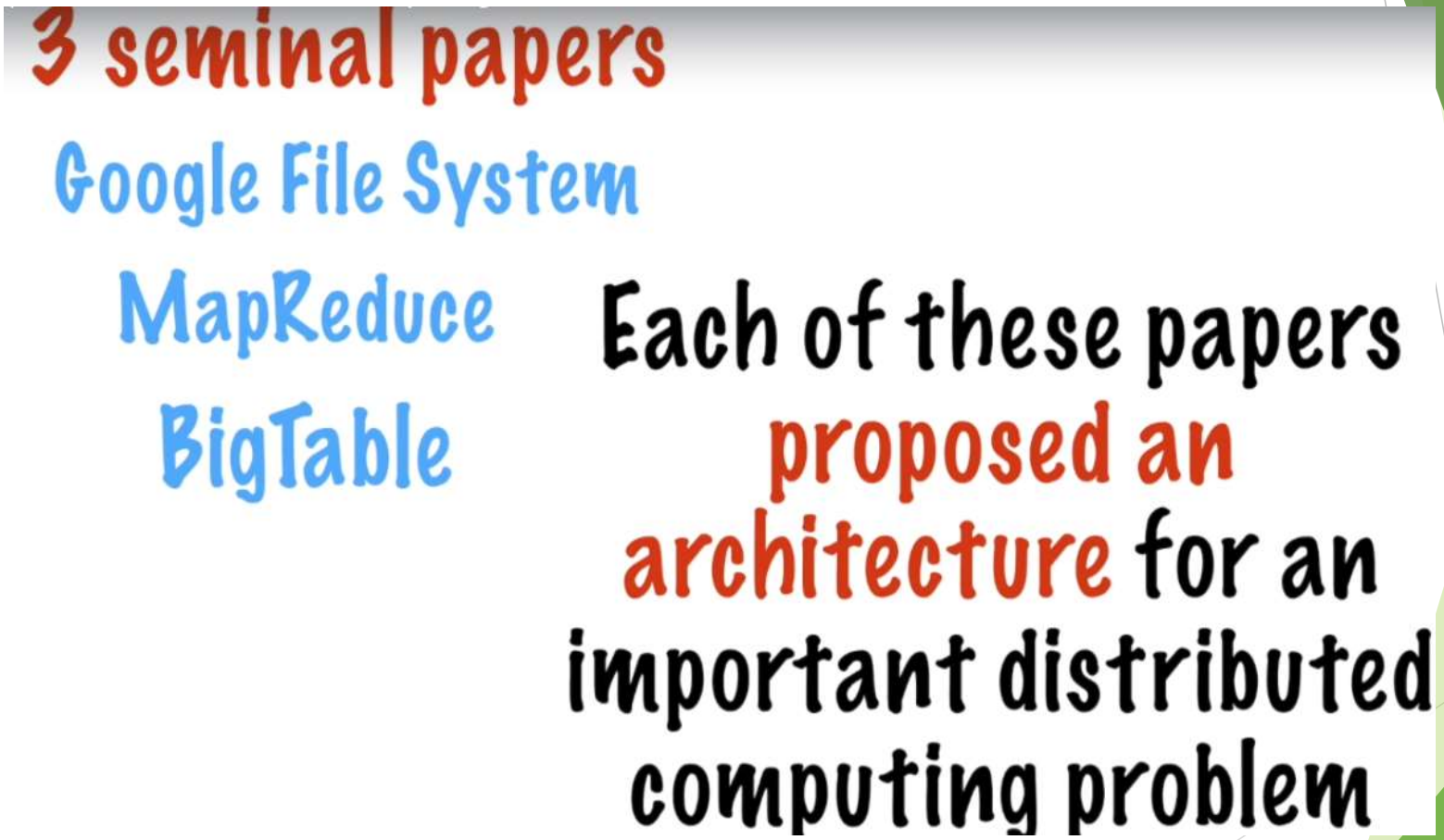
HBase



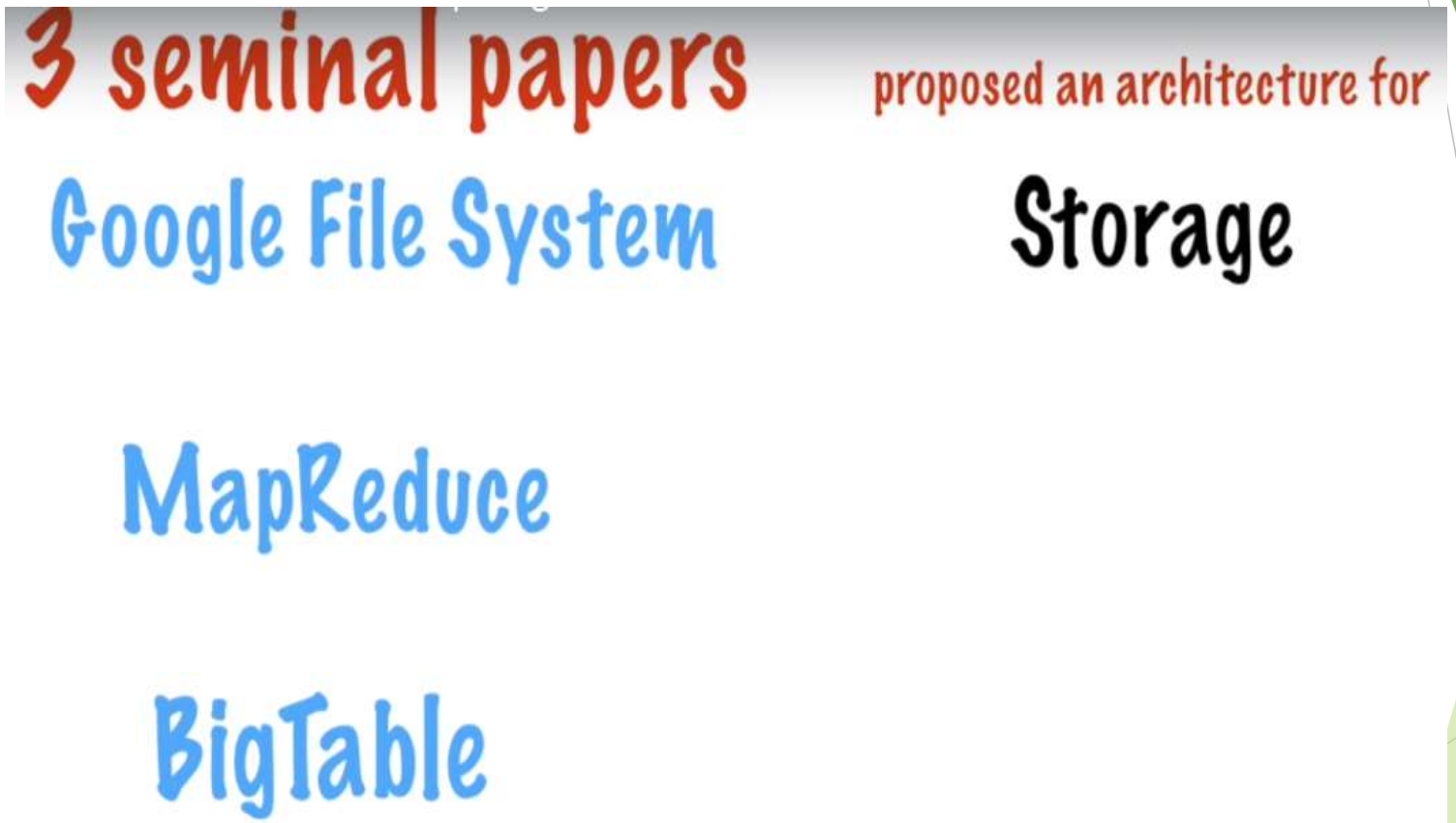
HBase


3 seminal papers
Google File System
MapReduce
BigTable
These are all technologies built to power Google Search

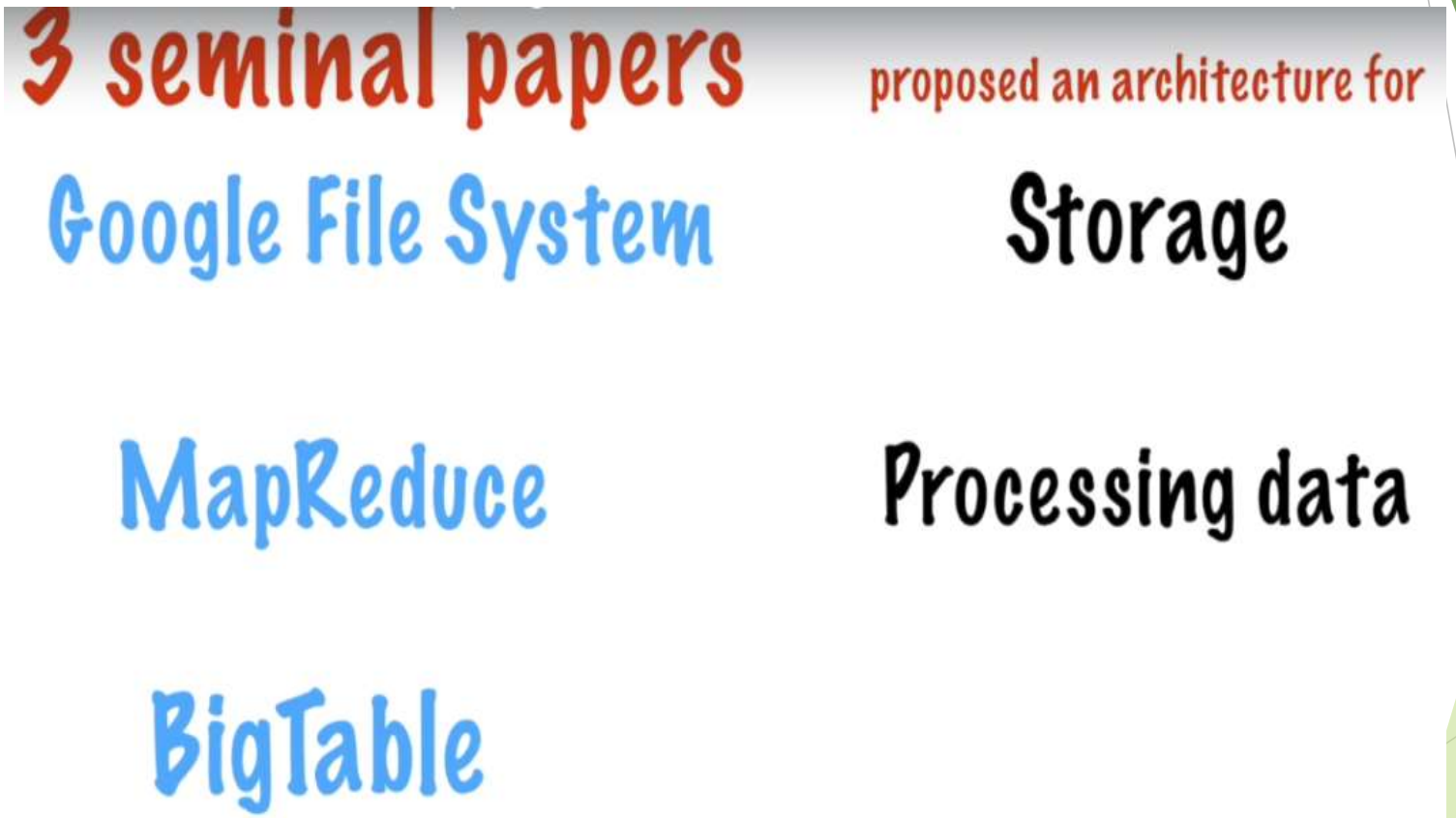
HBase


3 seminal papers
Google File System
MapReduce
BigTable
Each of these papers proposed an architecture for an important distributed computing problem

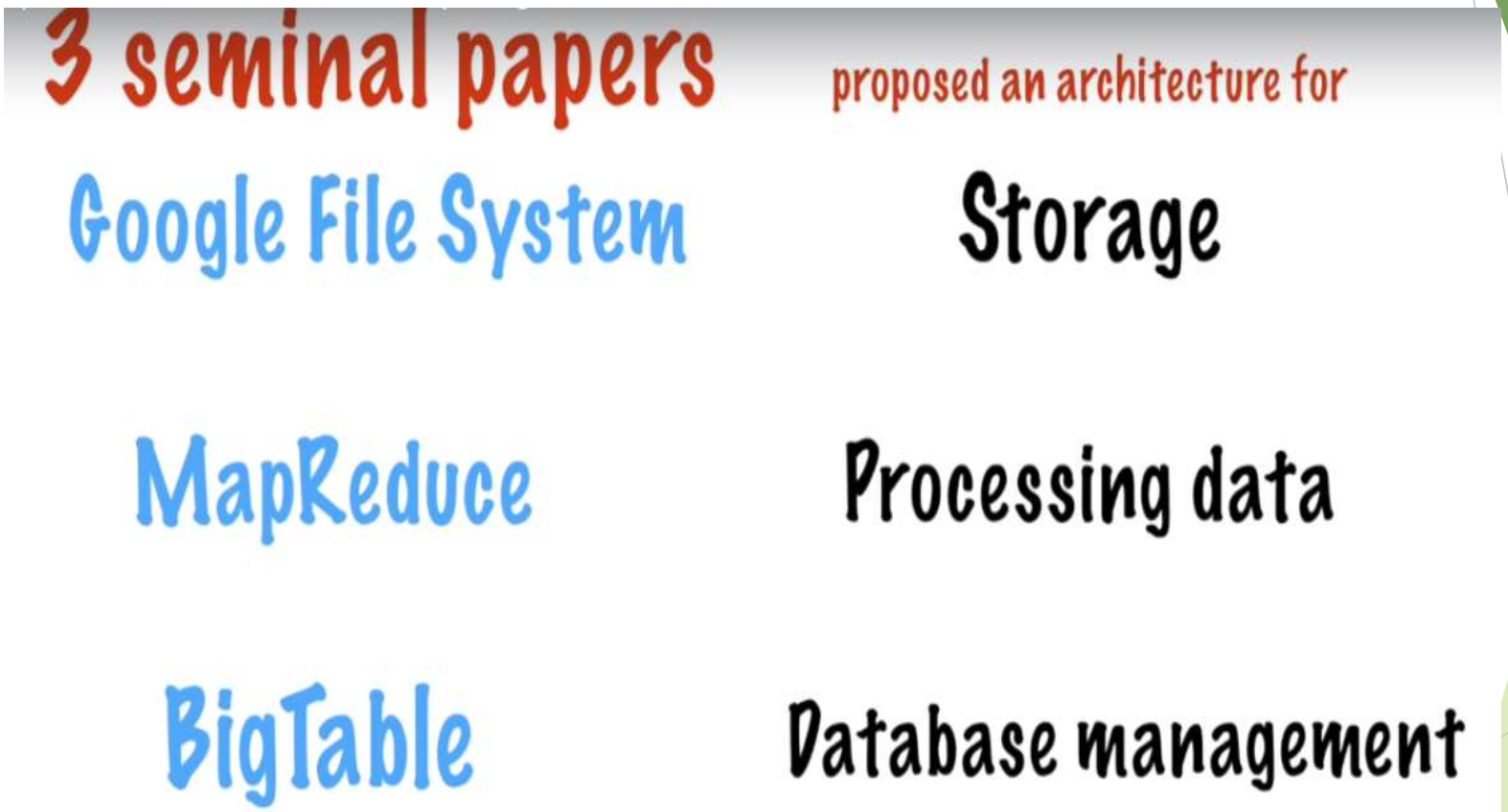
HBase


3 seminal papers
proposed an architecture for
Google File System
Storage
MapReduce
BigTable

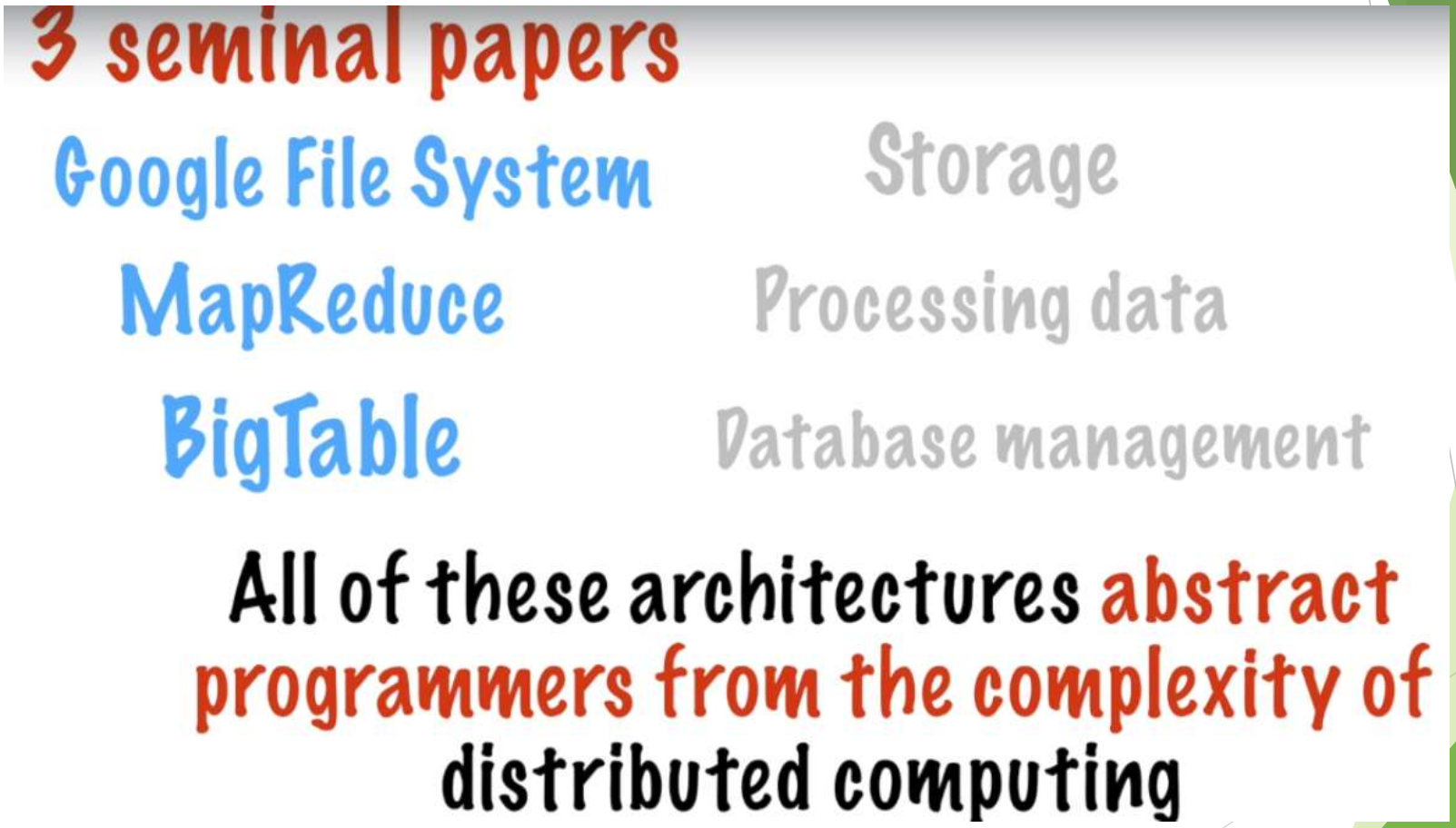
HBase


3 seminal papers
proposed an architecture for
Google File System
Storage
MapReduce
Processing data
BigTable

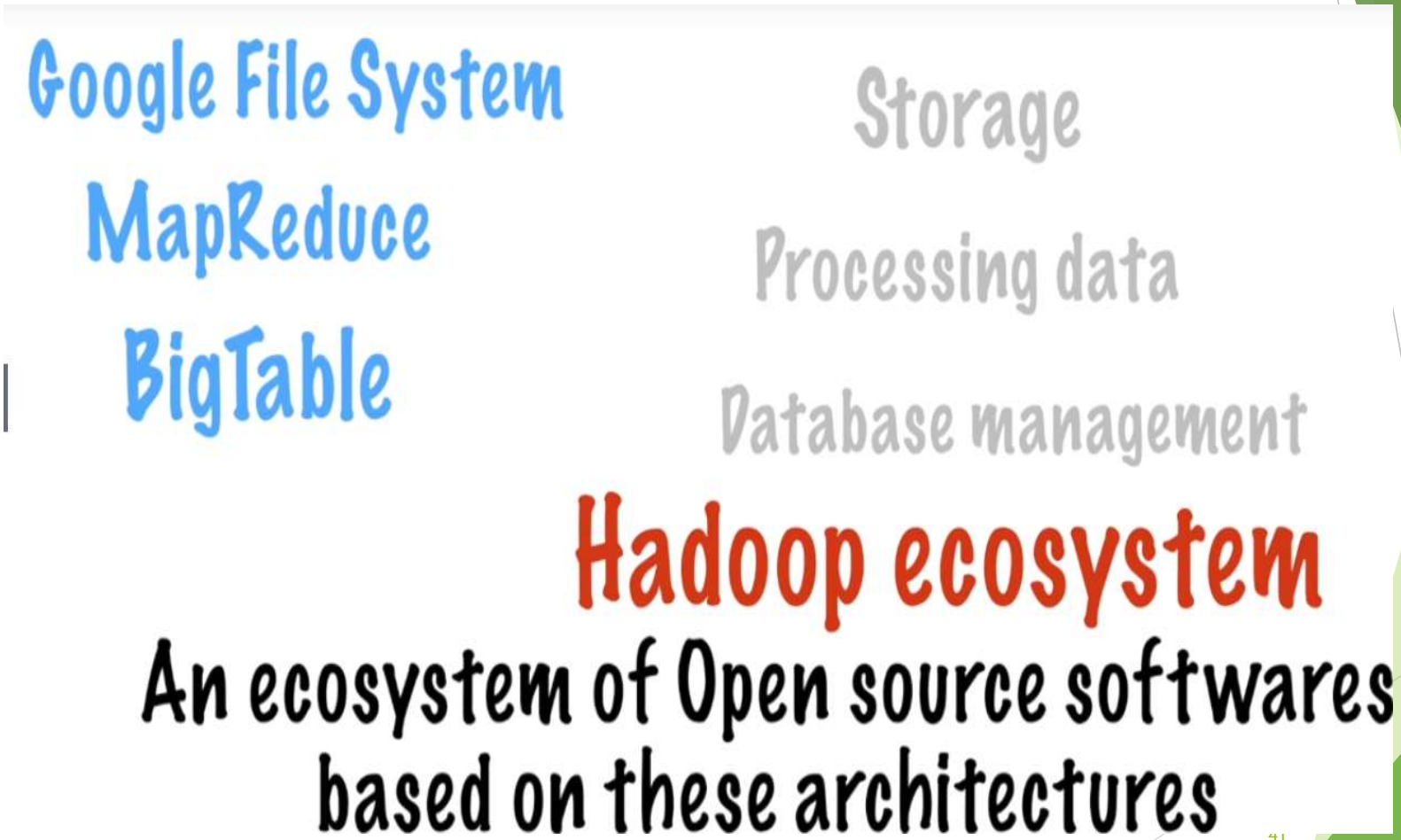
HBase


3 seminal papers
proposed an architecture for
Google File System
Storage
MapReduce
Processing data
BigTable
Database management

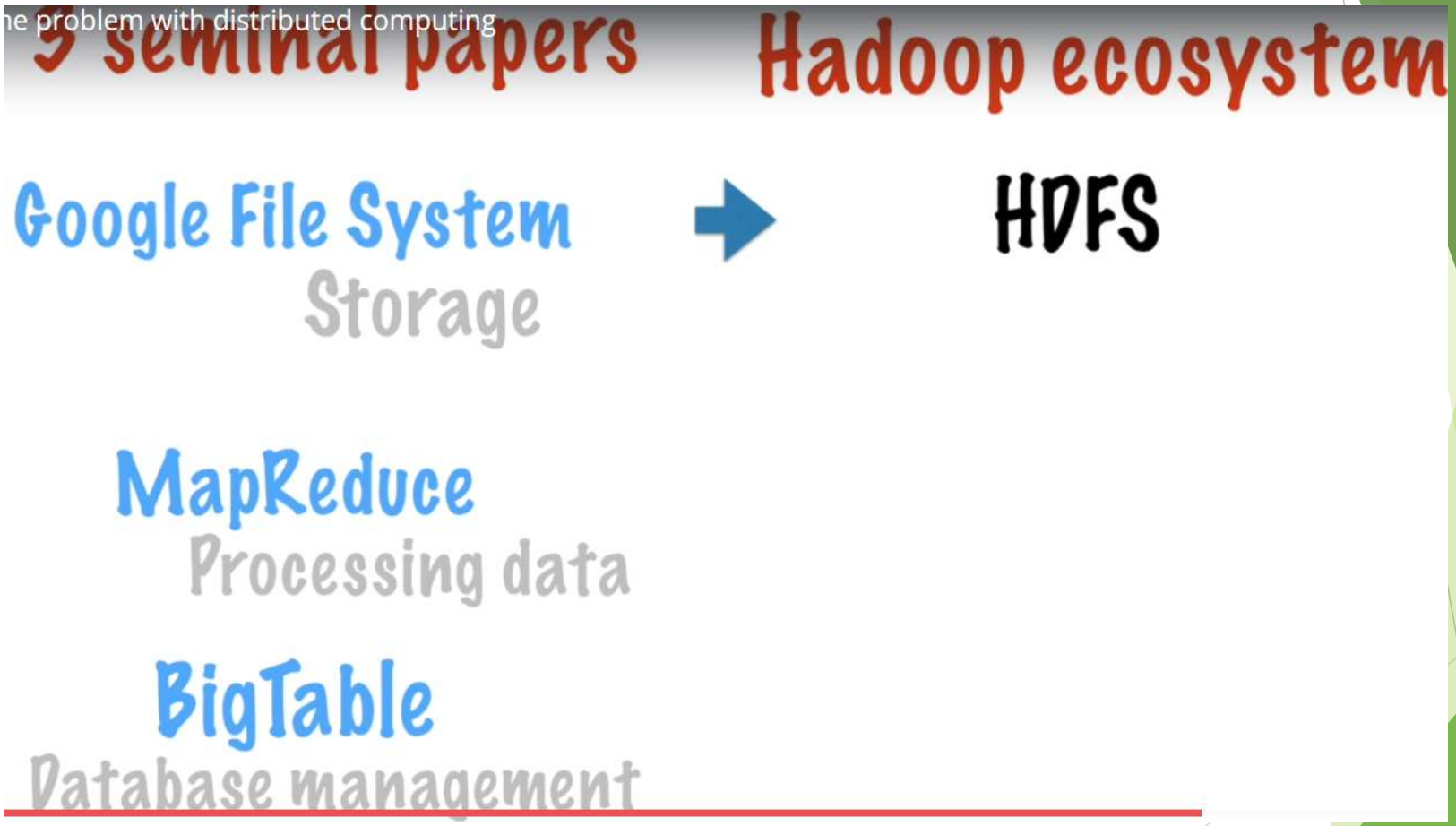
HBase


3 seminal papers
Google File System
Storage
MapReduce
Processing data
BigTable
Database management
All of these architectures abstract programmers from the complexity of distributed computing

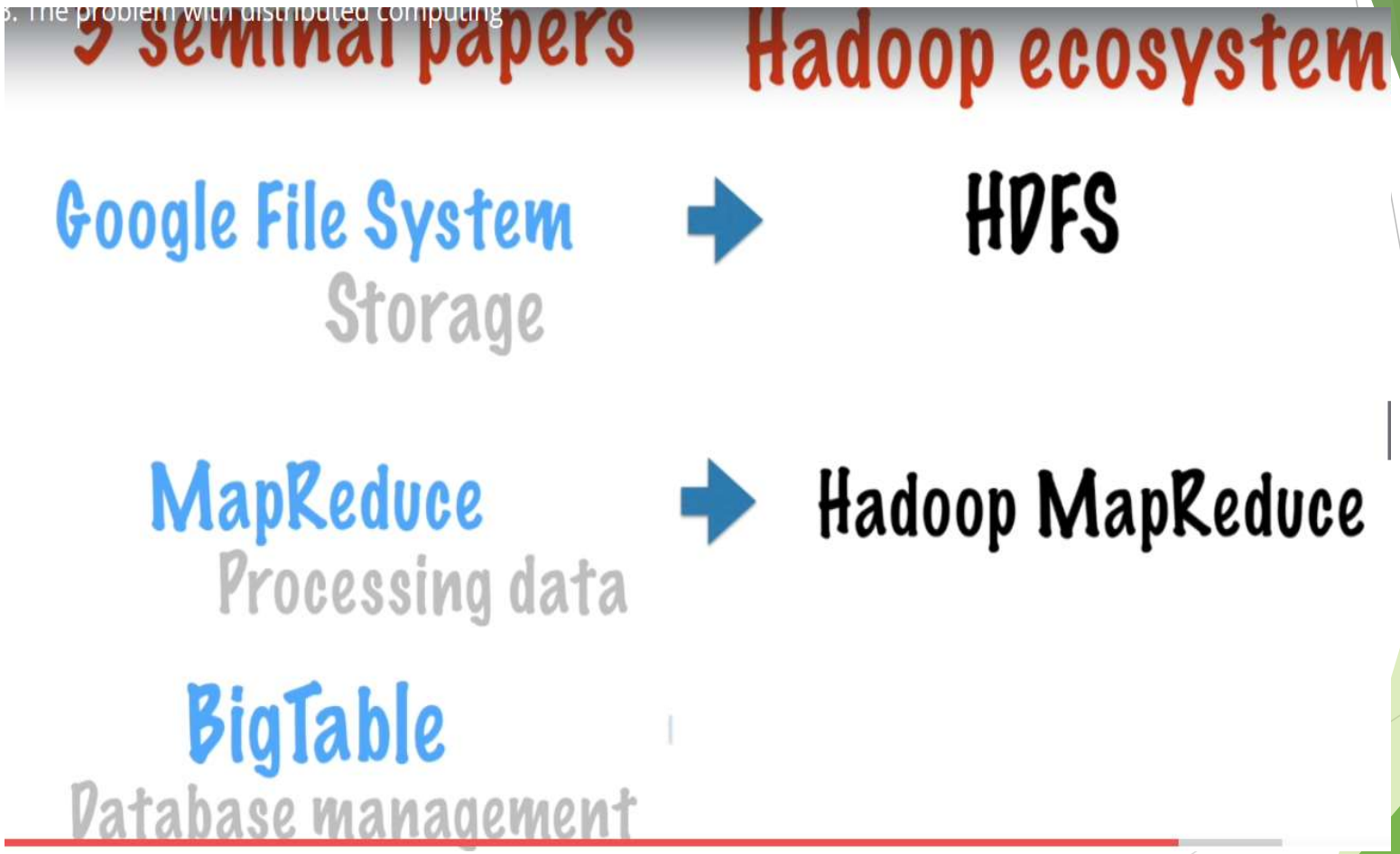
HBase


Google File System
MapReduce
BigTable
Storage
Processing data
Database management
Hadoop ecosystem
An ecosystem of Open source softwares
based on these architectures

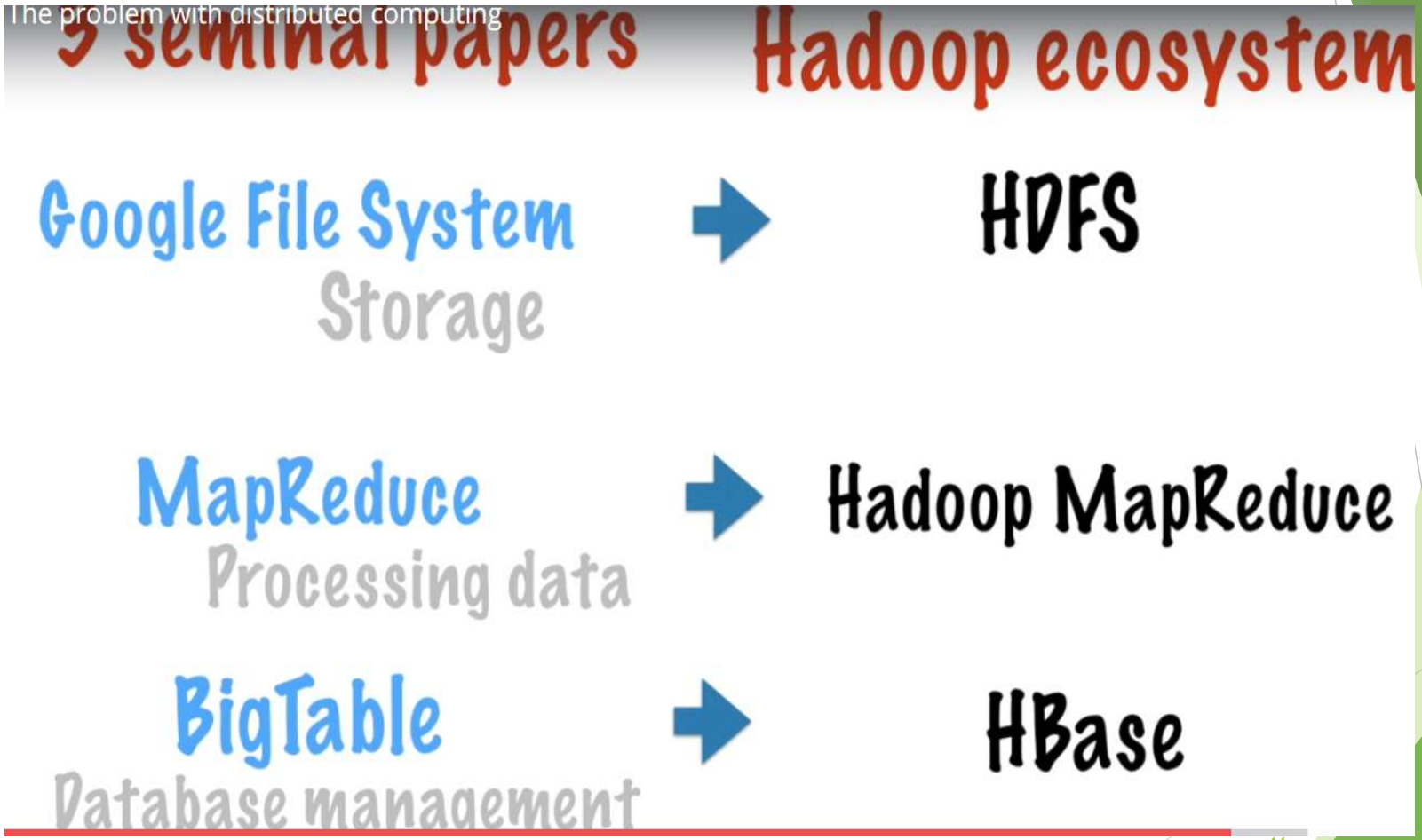
HBase


he problem with distributed computing
9 seminal papers
Hadoop ecosystem
Google File System
Storage
HDFS
MapReduce
Processing data
BigTable
Database management

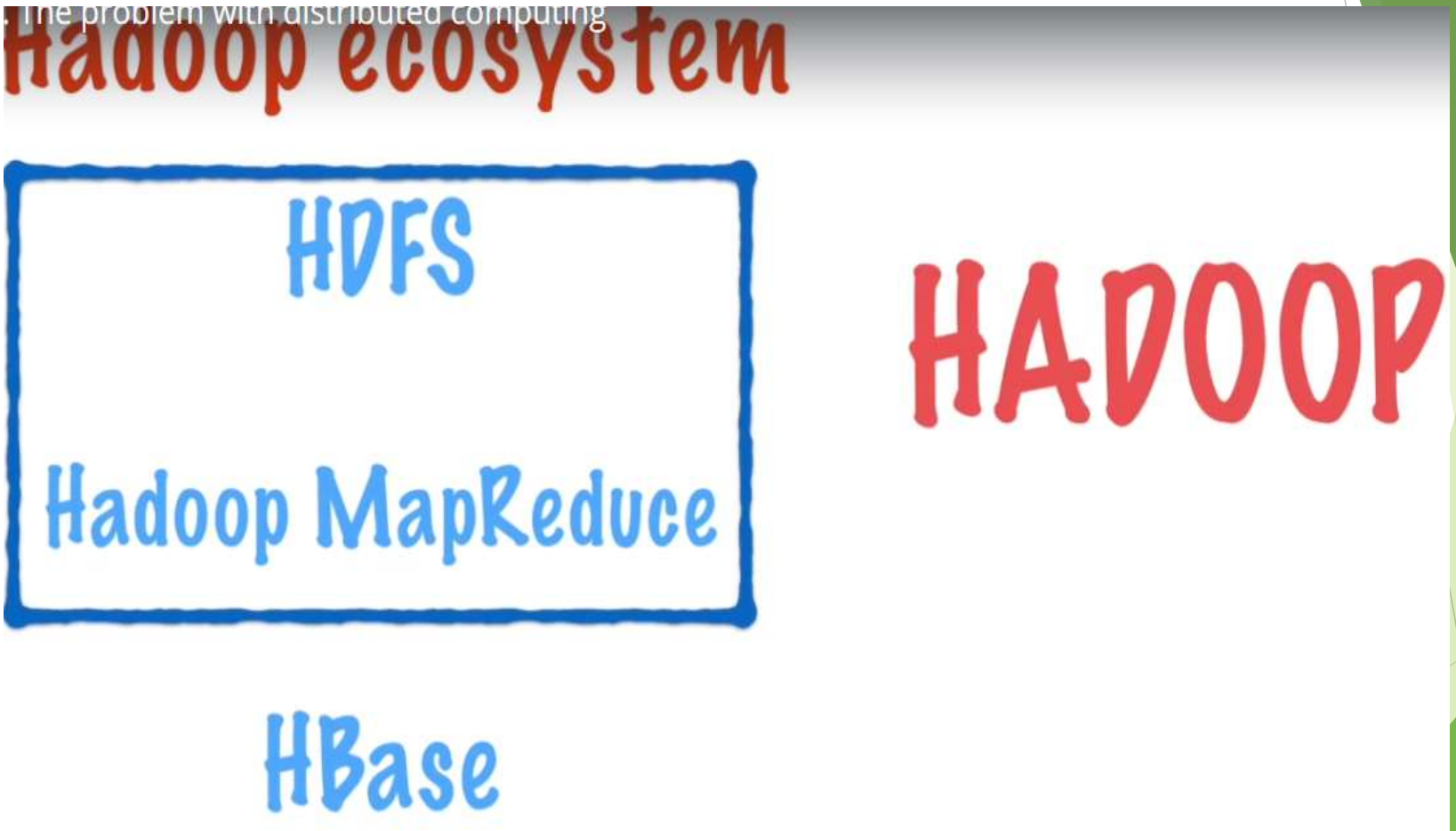
HBase


9 seminal papers
Hadoop ecosystem
Google File System
Storage
HDFS
MapReduce
Processing data
Hadoop MapReduce
BigTable
Database management

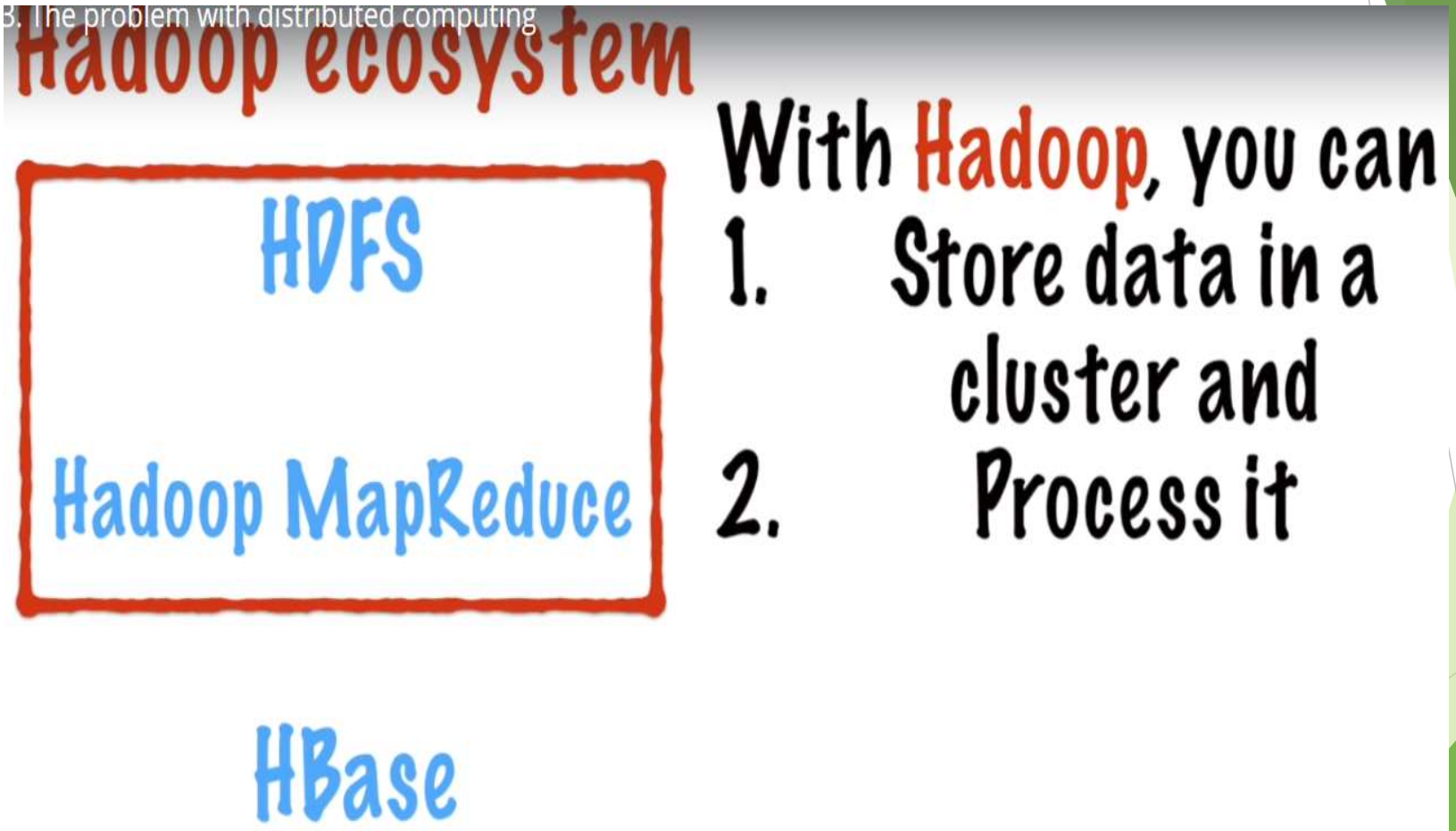
HBase


The problem with distributed computing
5 seminal papers
Hadoop ecosystem
Google File System
Storage
HDFS
MapReduce
Processing data
Hadoop MapReduce
BigTable
Database management
HBase

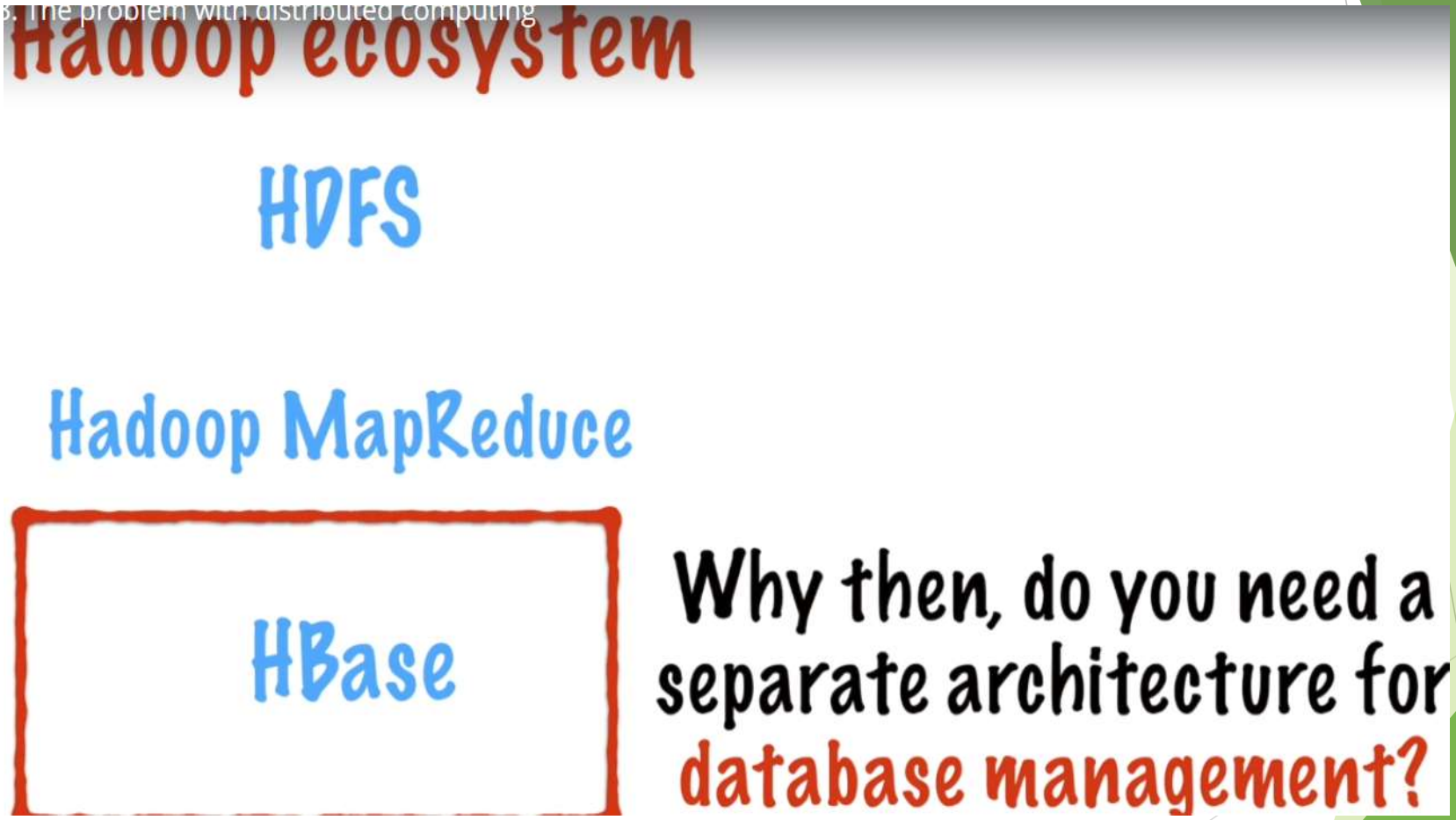
HBase


The problem with distributed computing
Hadoop ecosystem
HDFS
Hadoop MapReduce
HBase
HADOOP

HBase


3. The problem with distributed computing
Hadoop ecosystem
HDFS
Hadoop MapReduce
HBase
With Hadoop, you can
1. Store data in a cluster and
2. Process it

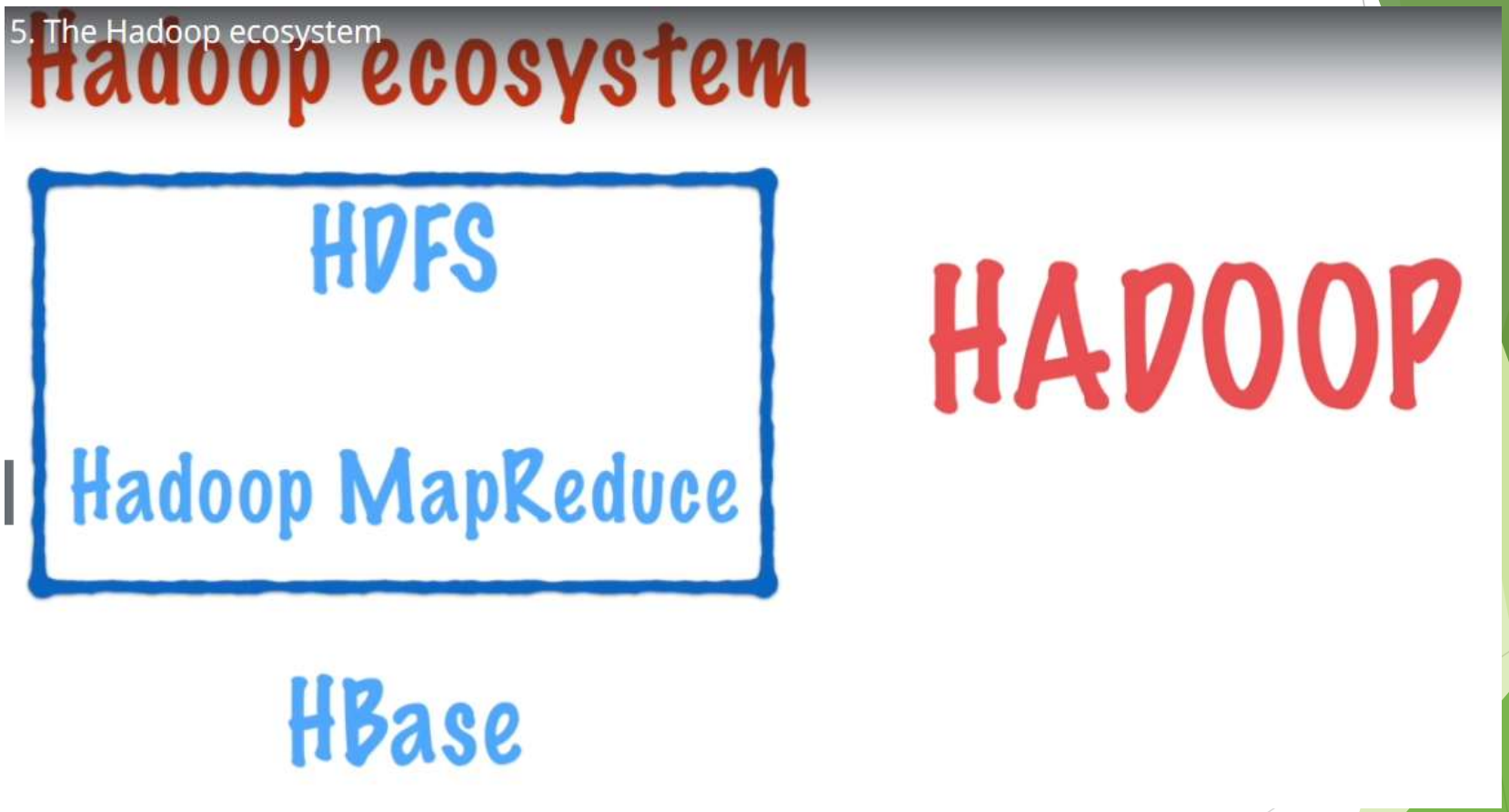
HBase


3. The problem with distributed computing
Hadoop ecosystem
HDFS
Hadoop MapReduce
HBase
Why then, do you need a separate architecture for database management?

Hbase - Installation

- Please refer HBaseInstall.txt for Hbase installation

HBase


5. The Hadoop ecosystem
Hadoop ecosystem
HDFS
Hadoop MapReduce
HBase
HADOOP

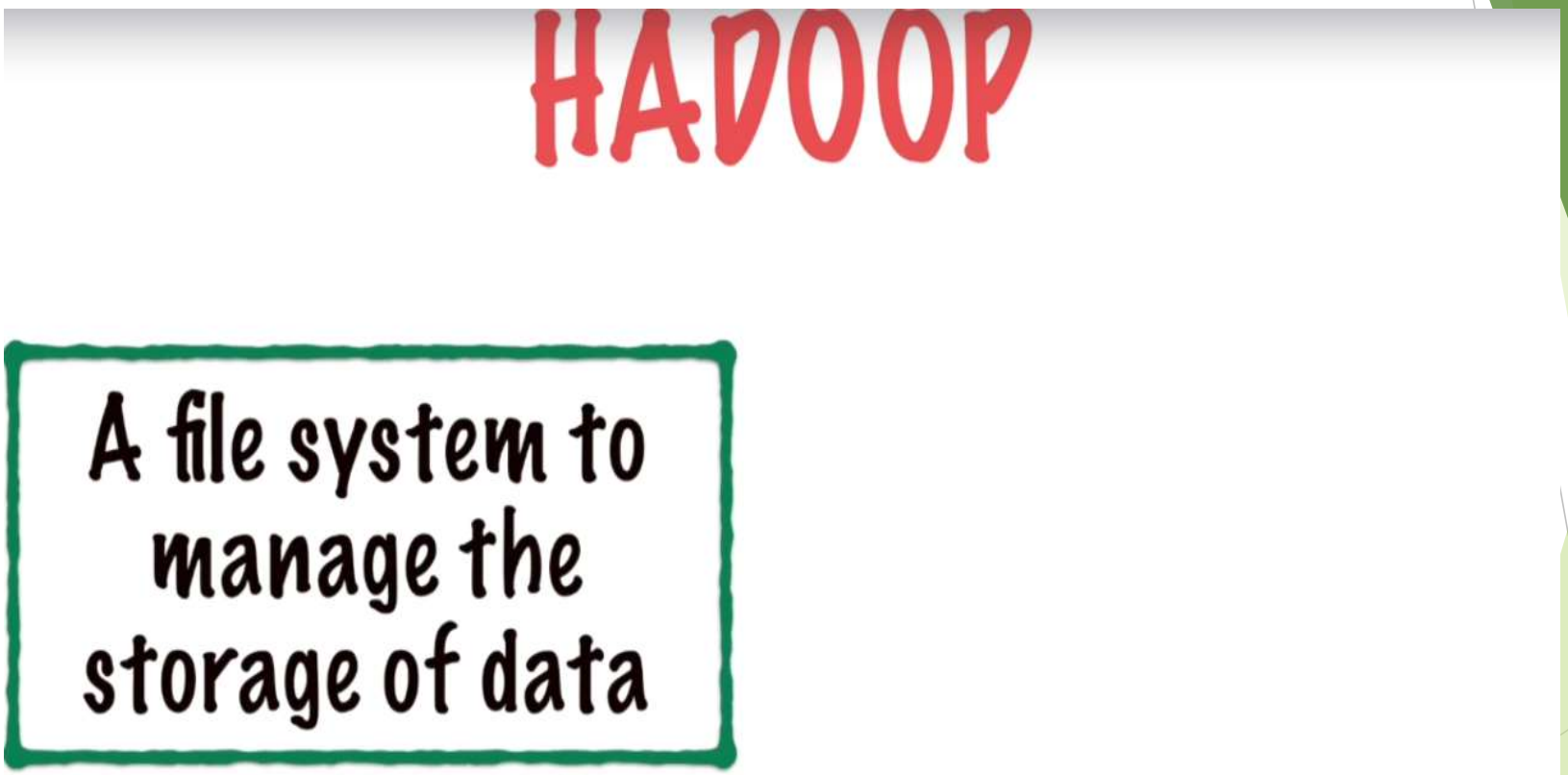
HBase


HADOOP
is a distributed computing framework
developed and maintained by
THE APACHE SOFTWARE FOUNDATION

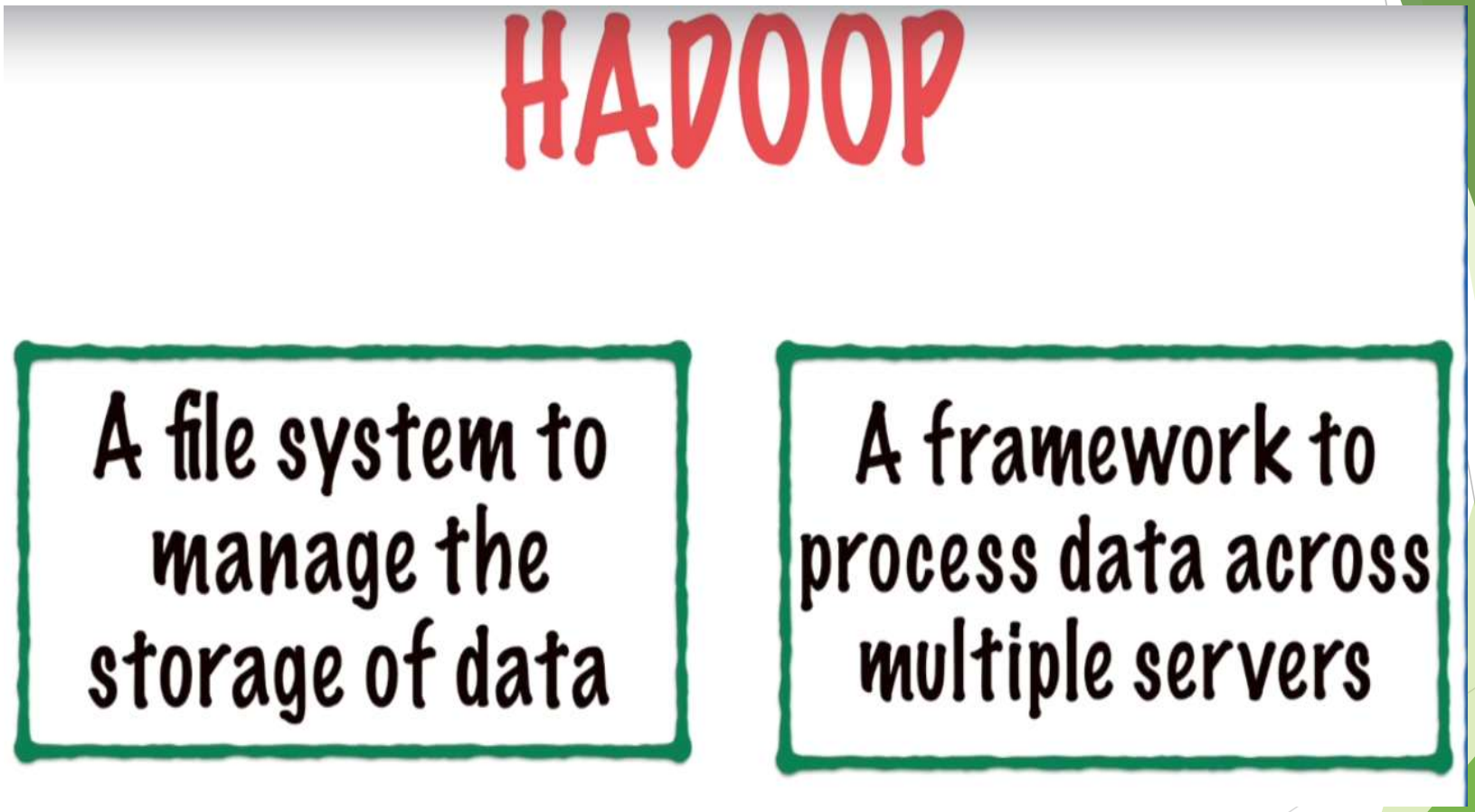
HBase


HADOOP
is a distributed computing framework
developed and maintained by
THE APACHE SOFTWARE FOUNDATION
written in Java

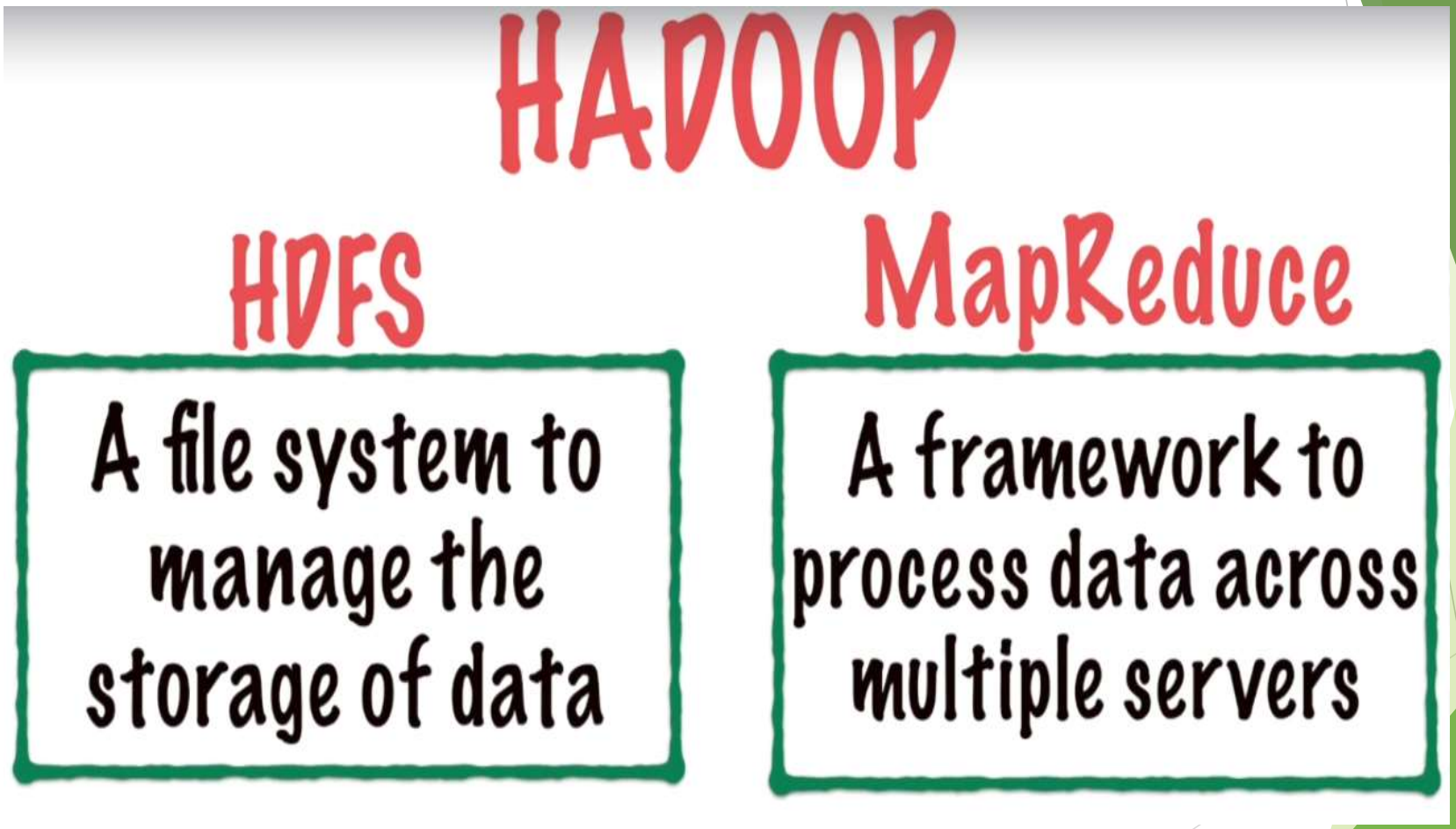
HBase



HBase


HADOOP
A file system to manage the storage of data
A framework to process data across multiple servers

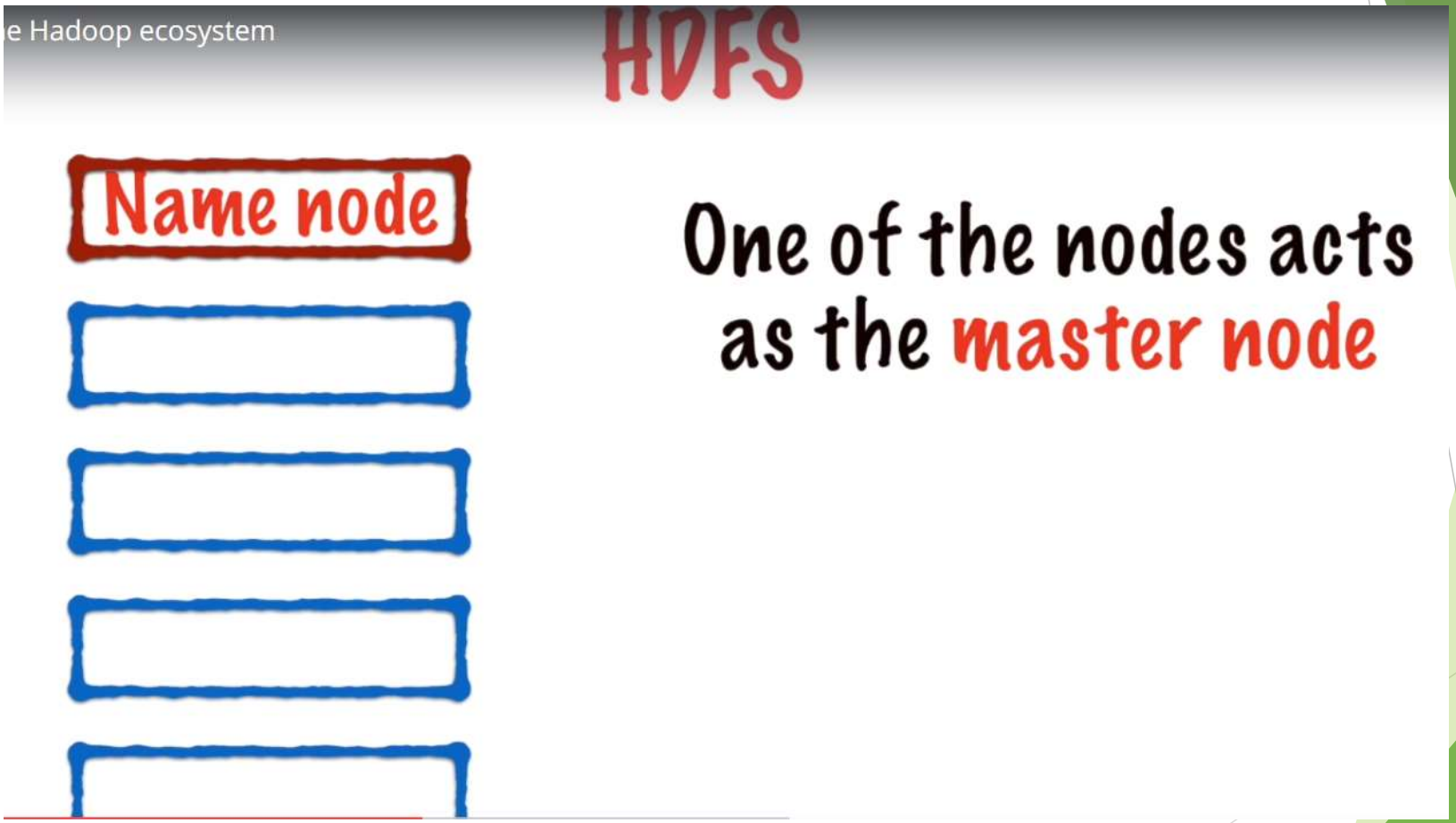
HBase


HADOOP
HDFS
A file system to manage the storage of data
MapReduce
A framework to process data across multiple servers

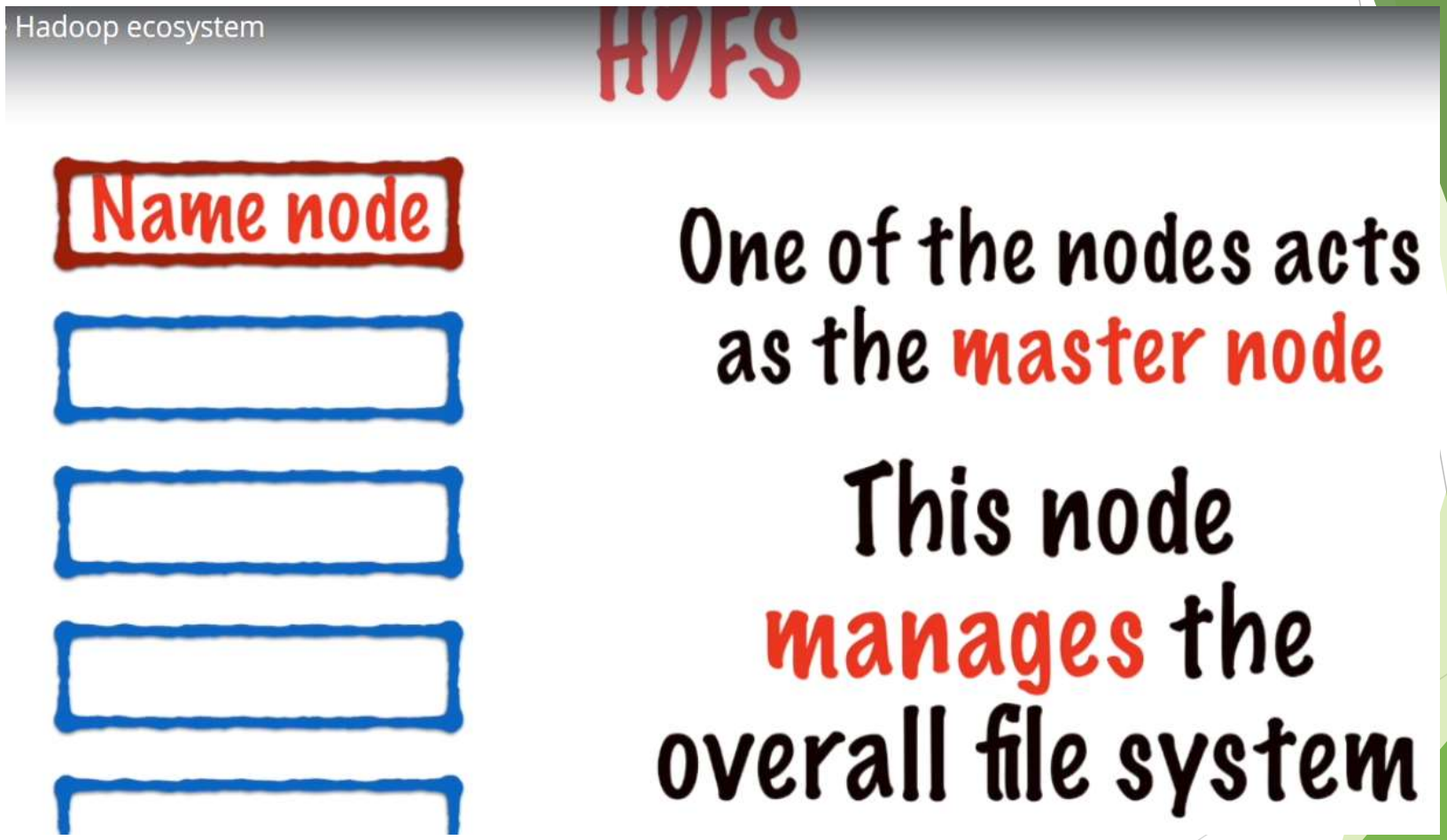
HBase


HDFS
The Hadoop Distributed File System
Hadoop uses this to store
data across multiple disks

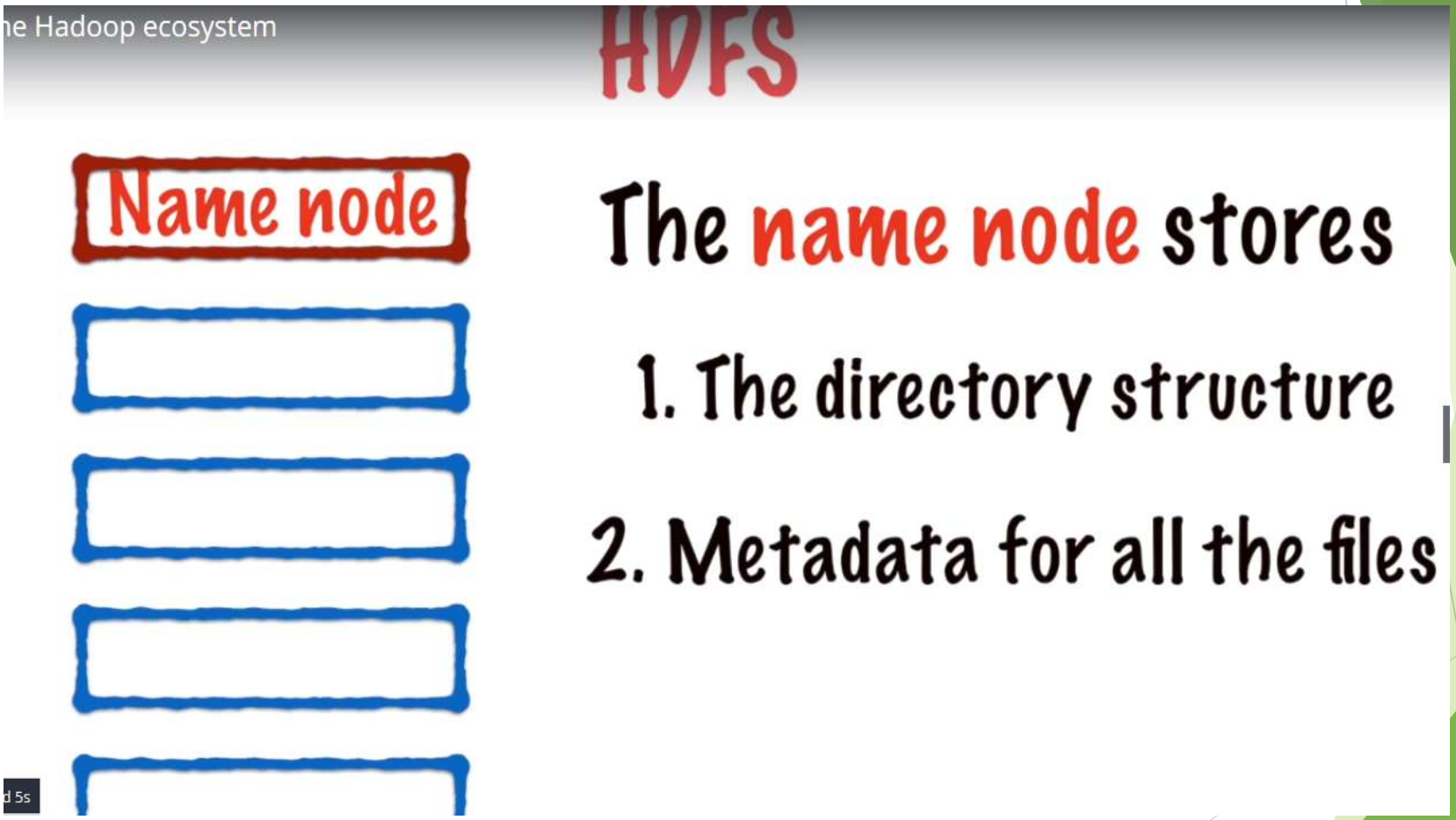
HBase


e Hadoop ecosystem
HDFS
Name node
One of the nodes acts
as the master node

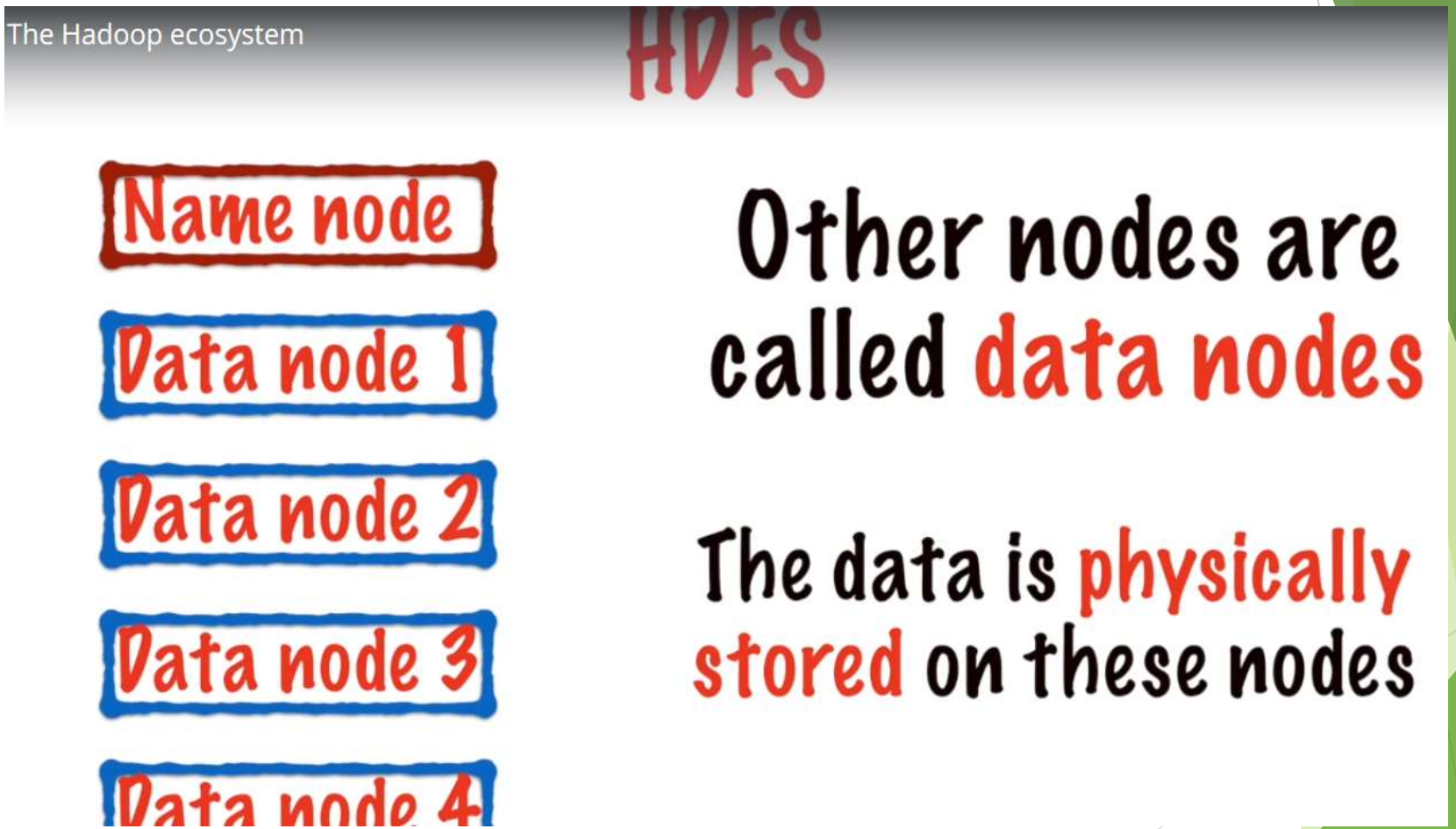
HBase


Hadoop ecosystem
HDFS
Name node
One of the nodes acts as the master node
This node manages the overall file system

HBase


he Hadoop ecosystem
HDFS
Name node
The name node stores
1. The directory structure
2. Metadata for all the files
d 5s

HBase


The Hadoop ecosystem
HDFS
Name node
Data node 1
Data node 2
Data node 3
Data node 4
Other nodes are called data nodes
The data is physically stored on these nodes

HBase

5. The Hadoop ecosystem

HDFS

Here is a large text file

next up previous contents index
Next: Dynamic indexing Up: Index construction Previous: Single-pass in-memory indexing Contents Index

Distributed indexing

Collections are often so large that we cannot perform index construction efficiently on a single machine. This is particularly true of the World Wide Web for which we need large computer clusters [*]to construct any reasonably sized web index. Web search engines, therefore, use distributed indexing algorithms for index construction. The result of the construction process is a distributed index that is partitioned across several machines – either according to term or according to document. In this section, we describe distributed indexing for a
term-partitioned index . Most large search engines prefer a document-partitioned index (which can be easily generated from a term-partitioned index). We discuss this topic further in Section 20.3 (page [*]).

The distributed index construction method we describe in this section is an application of MapReduce , a general architecture for distributed computing. MapReduce is designed for large computer clusters. The point of a cluster is to solve large computing problems on cheap commodity machines or nodes that are built from standard parts (processor, memory, disk) as opposed to on a supercomputer with specialized hardware. Although hundreds or thousands of machines are available in such clusters, individual machines can fail at any time. One requirement for robust distributed indexing is, therefore, that we divide the work up into chunks that we can easily assign and – in case of failure – reassign. A master node directs the process of assigning and reassigning tasks to individual worker nodes.

The map and reduce phases of MapReduce split up the computing job into chunks that standard machines can process in a short time. The various steps of MapReduce are shown in Figure 4.5 and an example on a collection consisting of two documents is shown in Figure 4.6 . First, the input data, in our case a collection of web pages, are split into n splits where the size of the split is chosen to ensure that the work can be distributed evenly (chunks should not be too large) and efficiently (the total number of chunks we need to manage should not be too large); 16 or 64 MB are good sizes in distributed indexing. Splits are not preassigned to machines, but are instead assigned by the master node on an ongoing basis: As a machine finishes processing one split, it is assigned the next one. If a machine dies or becomes a laggard due to hardware problems, the split it is working on is simply reassigned to another machine.

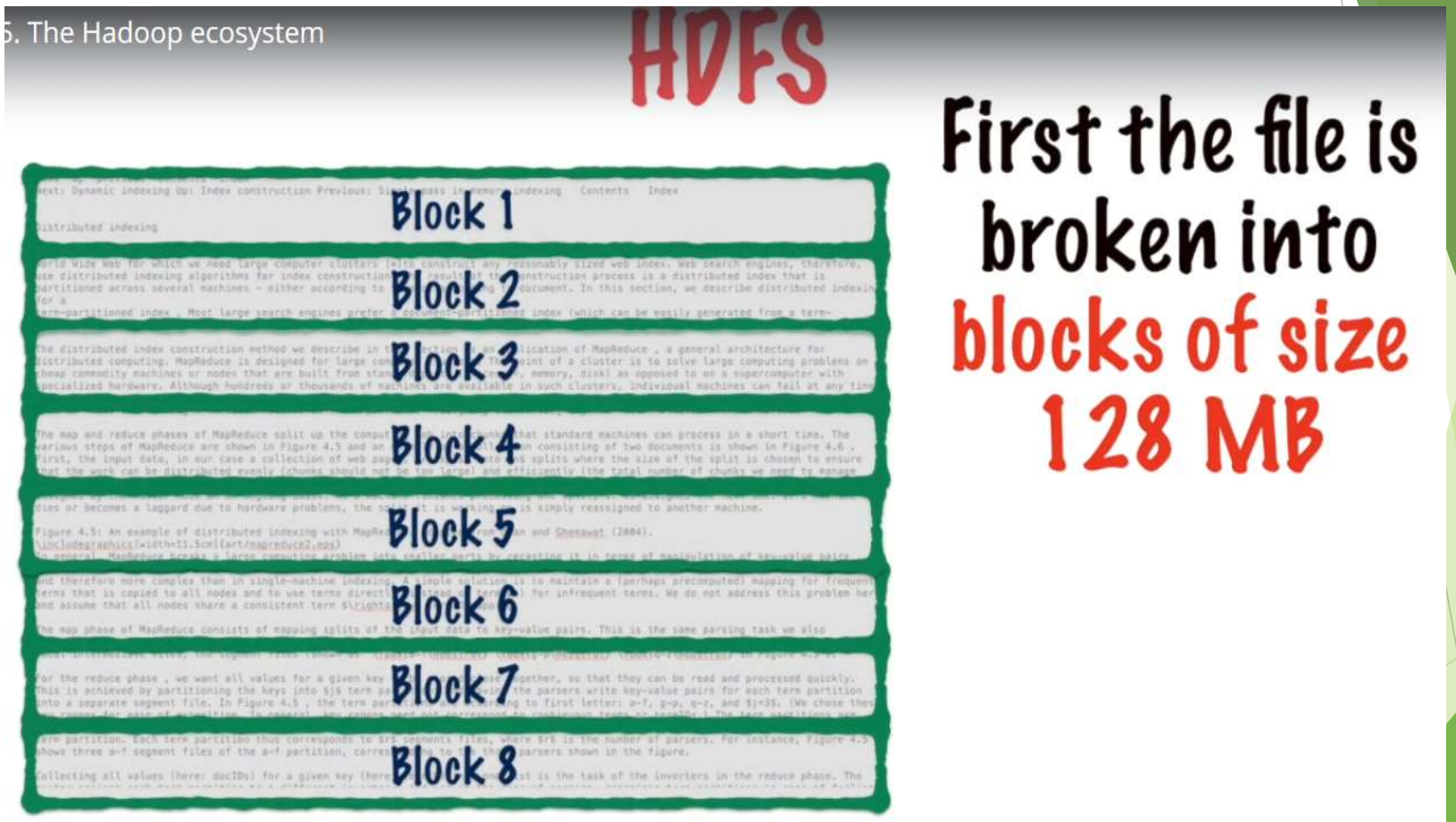
Figure 4.5: An example of distributed indexing with MapReduce. Adapted from Dean and Ghemawat (2004).
\includegraphics[width=11.5cm]{art/mapreduce2.eps}
In general, MapReduce breaks a large computing problem into smaller parts by recasting it in terms of manipulation of key-value pairs . For indexing, a key-value pair has the form (termID,docID). In distributed indexing, the mapping from terms to termIDs is also distributed and therefore more complex than in single-machine indexing. A simple solution is to maintain a (perhaps precomputed) mapping for frequent terms that is copied to all nodes and to use terms directly (instead of termIDs) for infrequent terms. We do not address this problem here and assume that all nodes share a consistent term $\rightarrow$ termID mapping.

The map phase of MapReduce consists of mapping splits of the input data to key-value pairs. This is the same parsing task we also encountered in BSBI and SPIMI, and we therefore call the machines that execute the map phase parsers . Each parser writes its output to local intermediate files, the segment files (shown as \fbox{a-f\medstrut} \fbox{g-p\medstrut} \fbox{q-z\medstrut} in Figure 4.5).

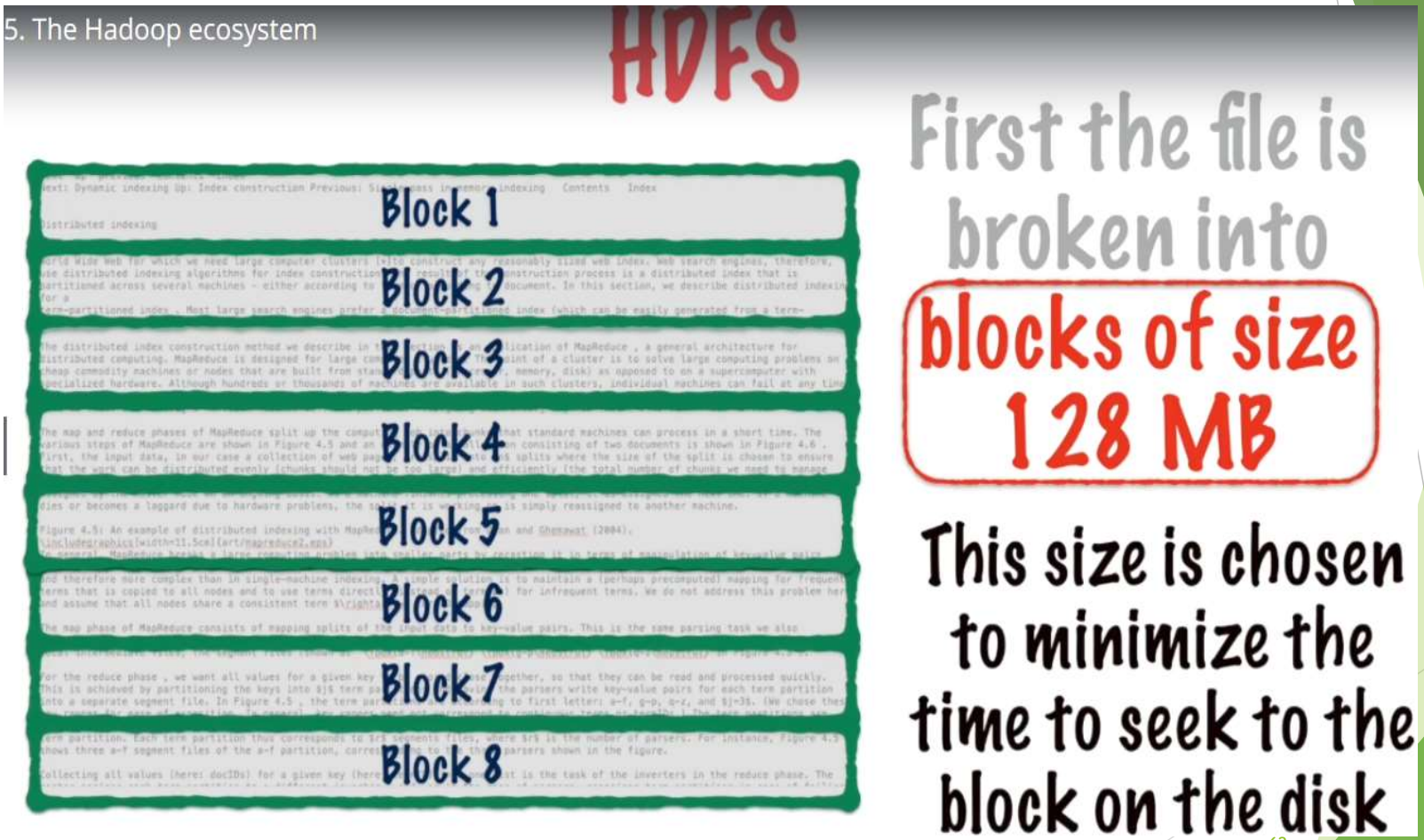
For the reduce phase , we want all values for a given key to be stored close together, so that they can be read and processed quickly. This is achieved by partitioning the keys into j term partitions and having the parsers write key-value pairs for each term partition into a separate segment file. In Figure 4.5 , the term partitions are according to first letter: a–f, g–p, q–z, and $j=3$. (We chose these key ranges for ease of exposition. In general, key ranges need not correspond to contiguous terms or termIDs.) The term partitions are defined by the person who operates the indexing system (Exercise 4.6). The parsers then write corresponding segment files, one for each term partition. Each term partition thus corresponds to r segments files, where r is the number of parsers. For instance, Figure 4.5 shows three a–f segment files of the a–f partition, corresponding to the three parsers shown in the figure.

Let's see how this file is stored in HDFS

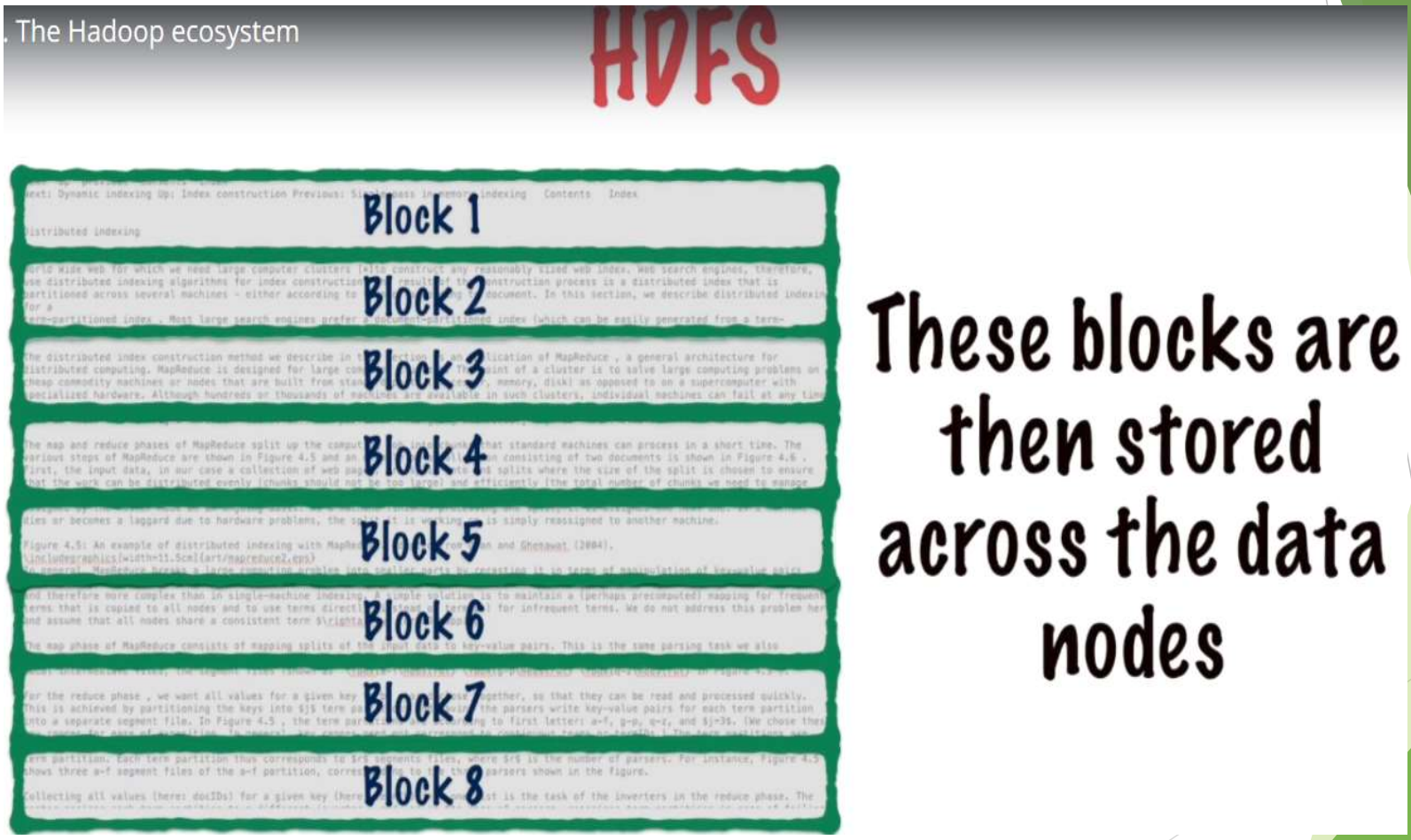
HBase


5. The Hadoop ecosystem
HDFS
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
First the file is broken into blocks of size 128 MB

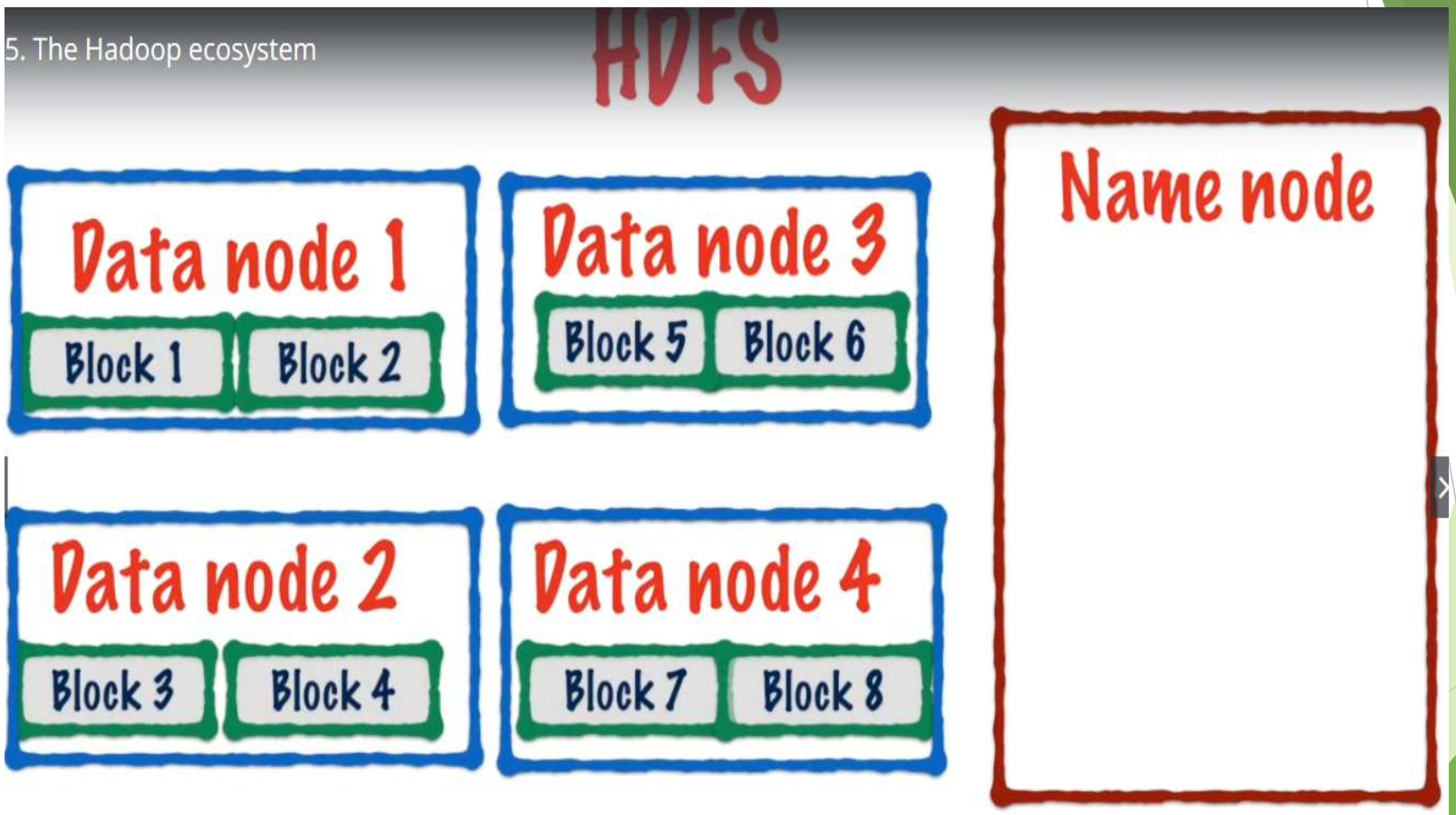
HBase


5. The Hadoop ecosystem
HDFS
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
First the file is broken into
blocks of size 128 MB
This size is chosen to minimize the time to seek to the block on the disk

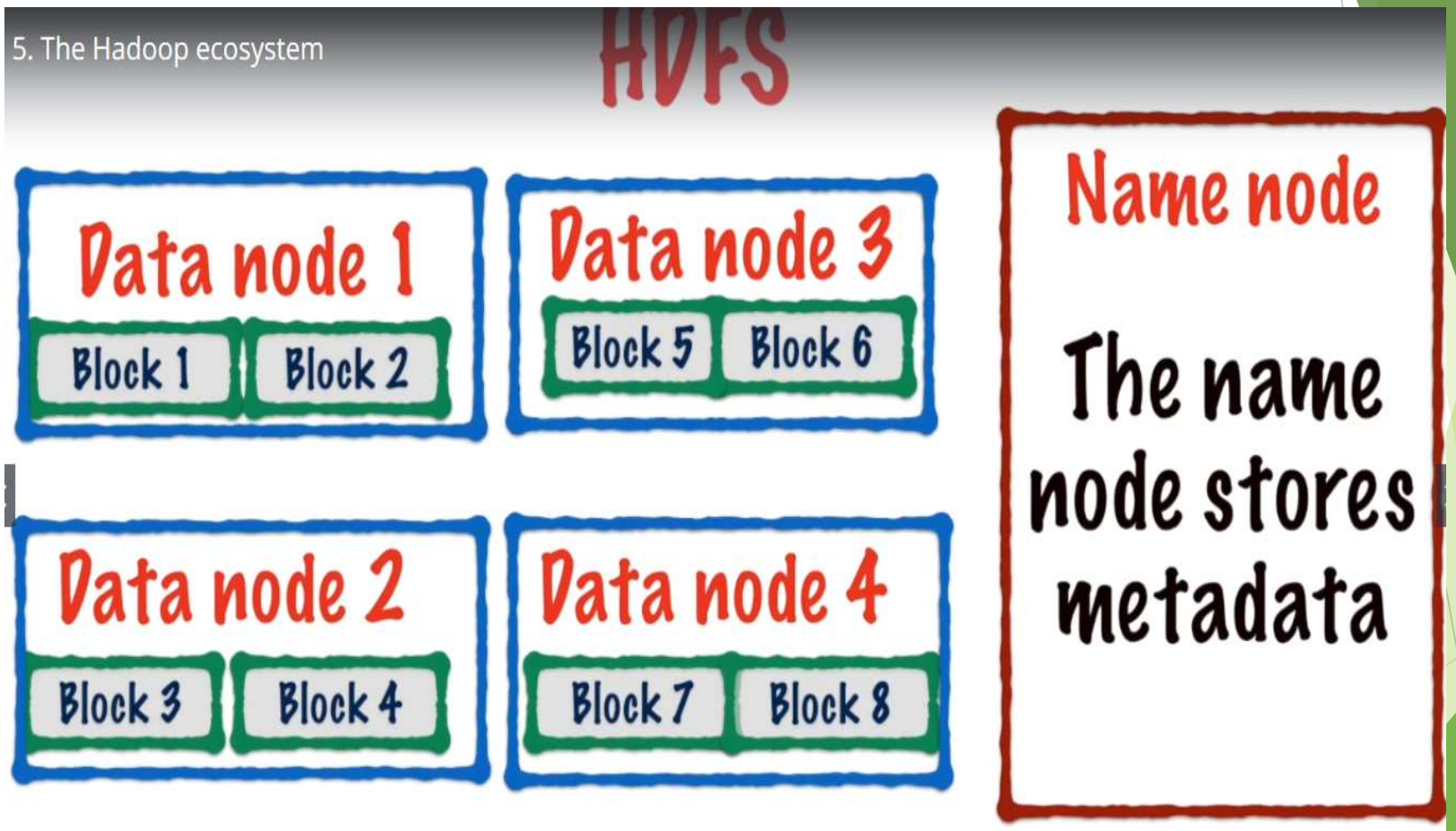
HBase


. The Hadoop ecosystem
HDFS
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
Block 7
Block 8
These blocks are then stored across the data nodes

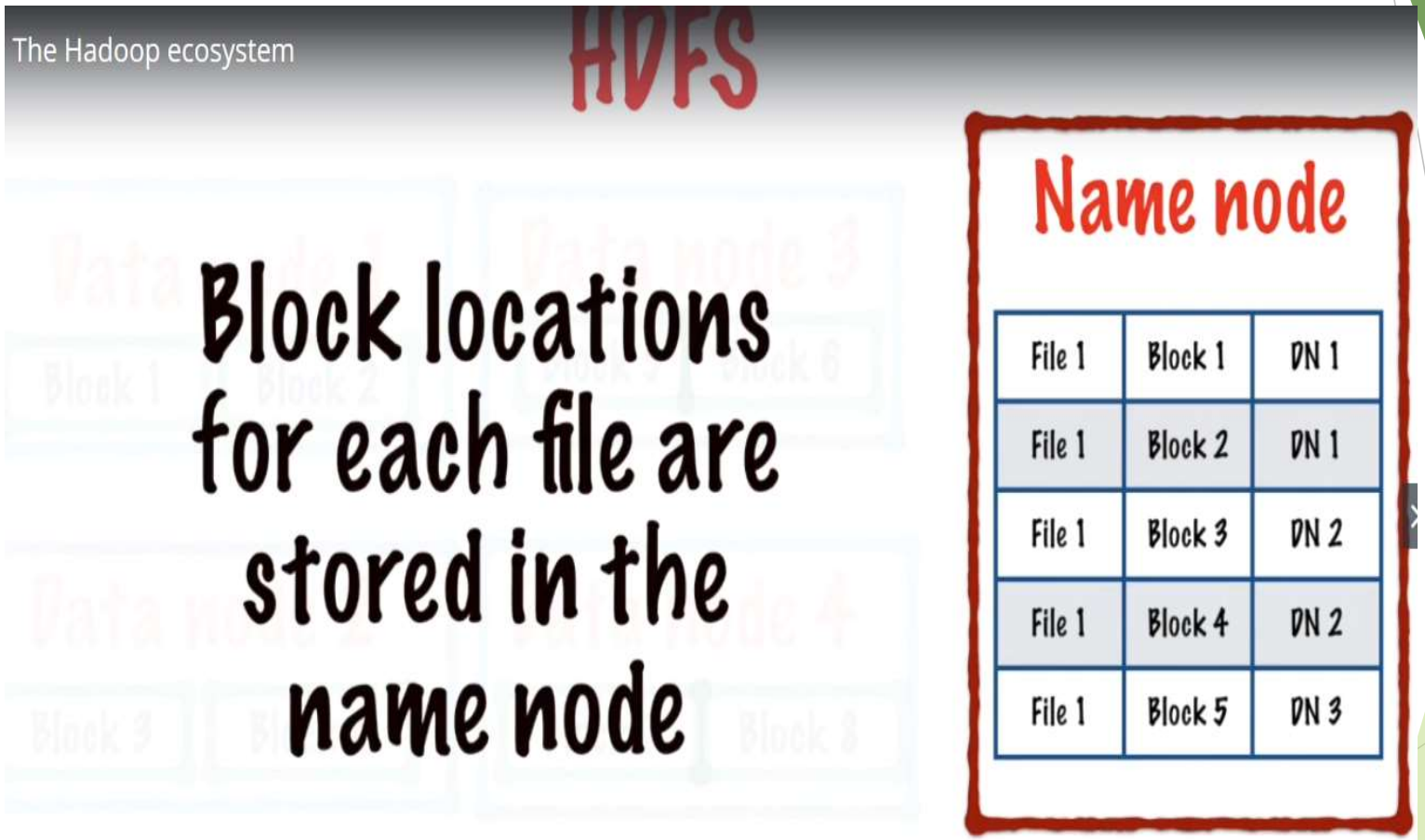
HBase


5. The Hadoop ecosystem
HDFS
Data node 1
Block 1
Block 2
Data node 3
Block 5
Block 6
Name node
Data node 2
Block 3
Block 4
Data node 4
Block 7
Block 8

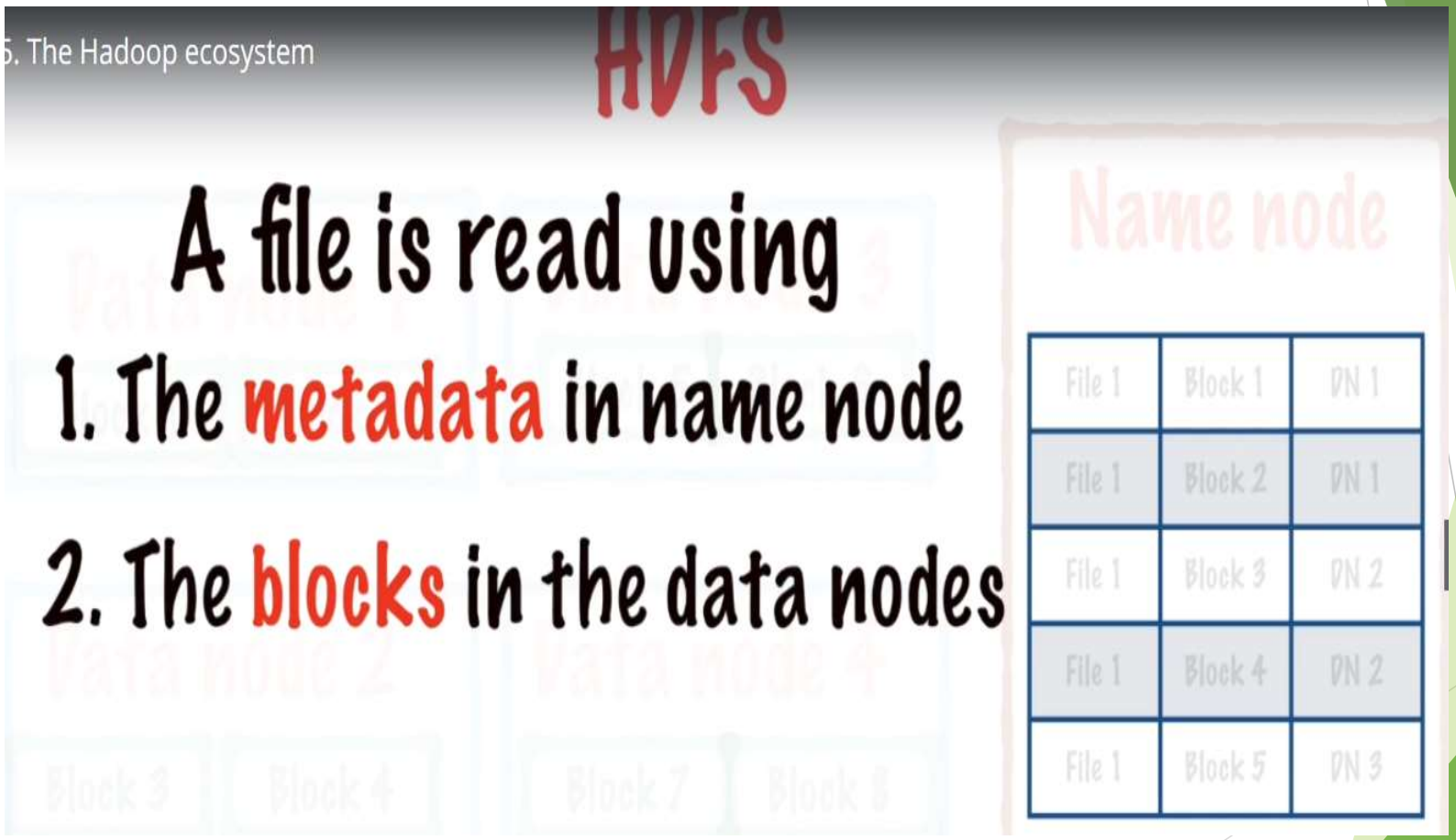
HBase


5. The Hadoop ecosystem
HDFS
Data node 1
Block 1
Block 2
Data node 3
Block 5
Block 6
Data node 2
Block 3
Block 4
Data node 4
Block 7
Block 8
Name node
The name node stores metadata

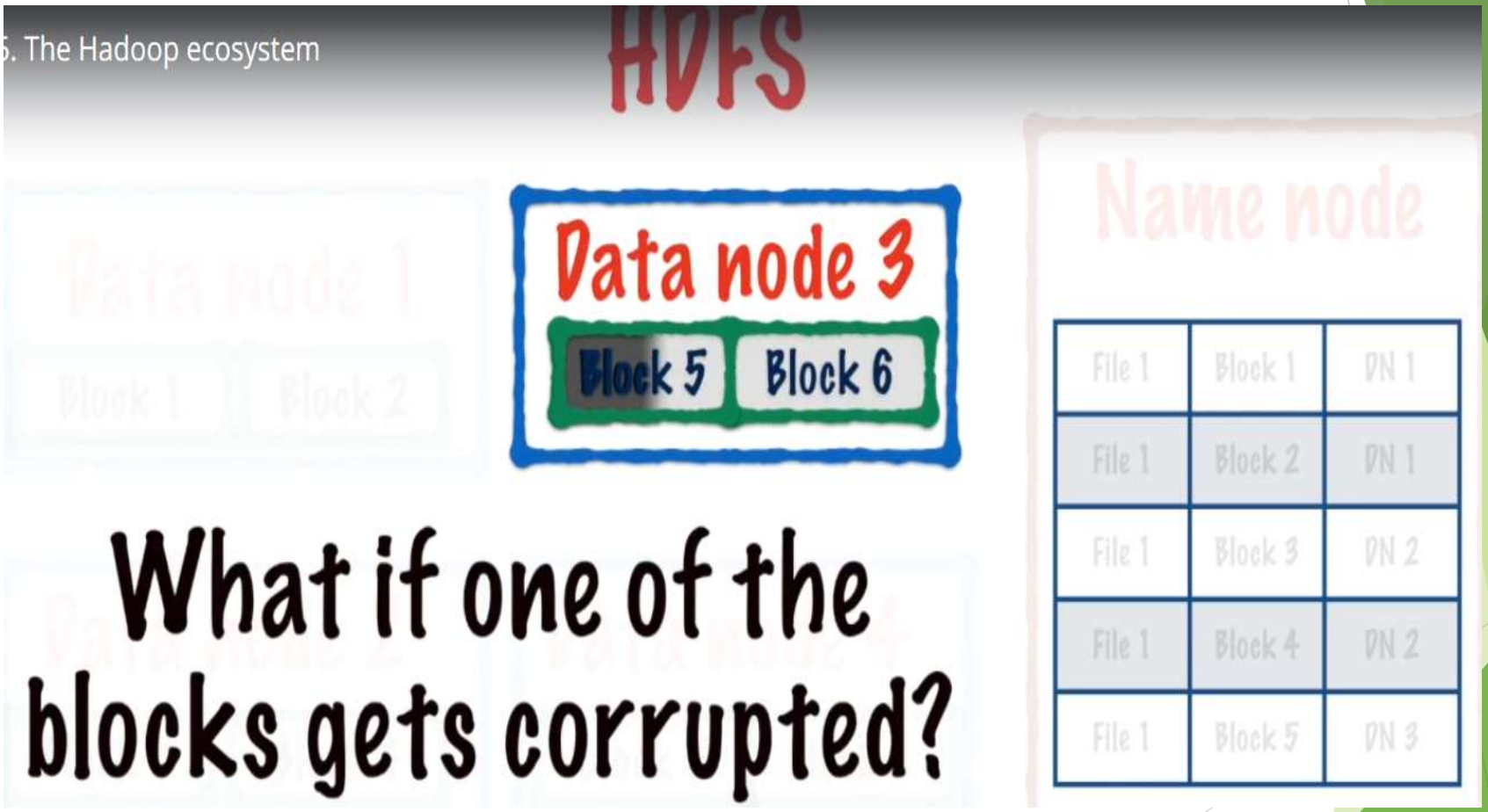
HBase


The Hadoop ecosystem
HDFS
Block locations for each file are stored in the name node
Name node
File 1 Block 1 DN 1
File 1 Block 2 DN 1
File 1 Block 3 DN 2
File 1 Block 4 DN 2
File 1 Block 5 DN 3

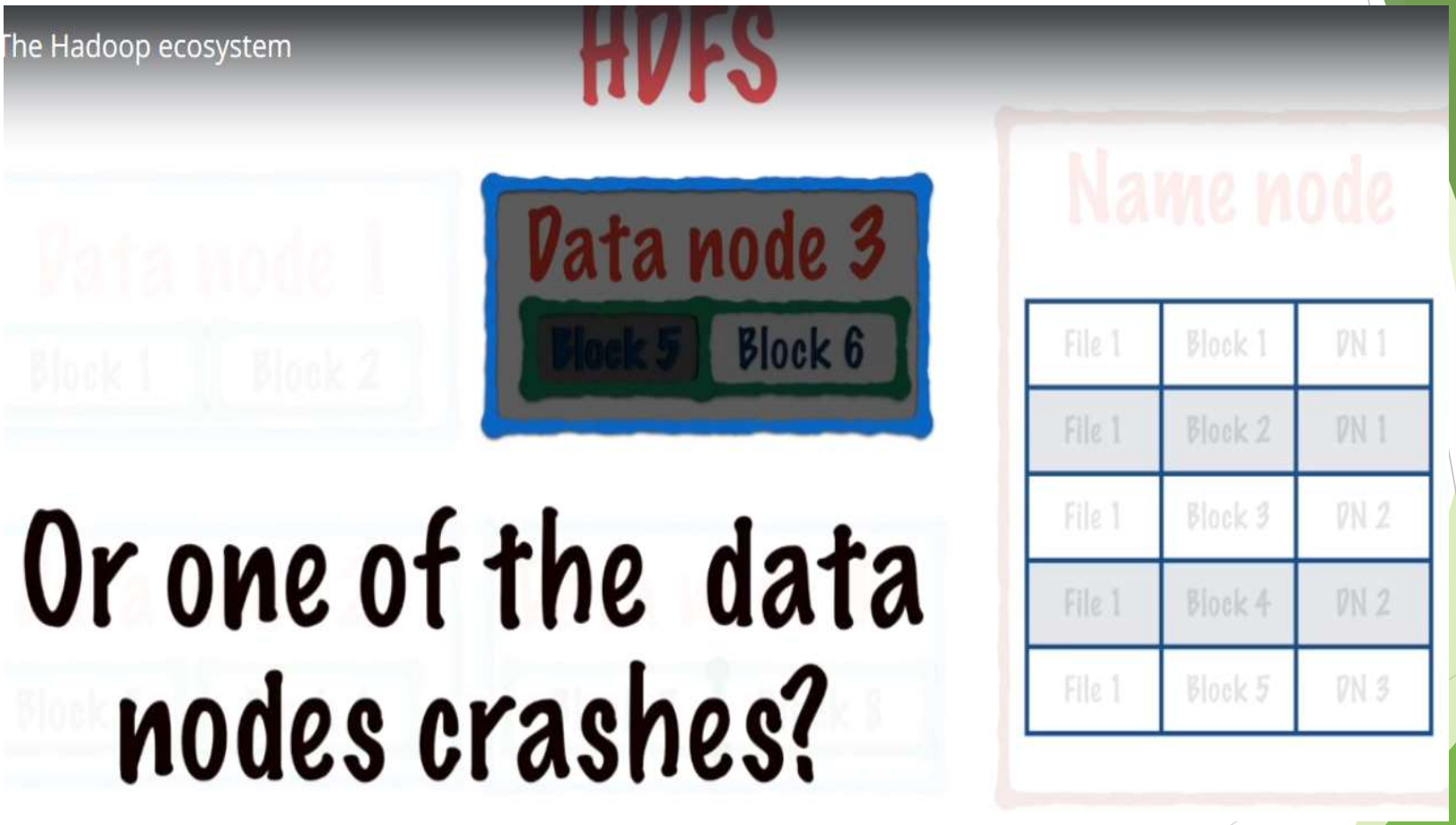
HBase


5. The Hadoop ecosystem
HDFS
A file is read using
1. The metadata in name node
2. The blocks in the data nodes
Name node
File 1 Block 1 DN 1
File 1 Block 2 DN 1
File 1 Block 3 DN 2
File 1 Block 4 DN 2
File 1 Block 5 DN 3

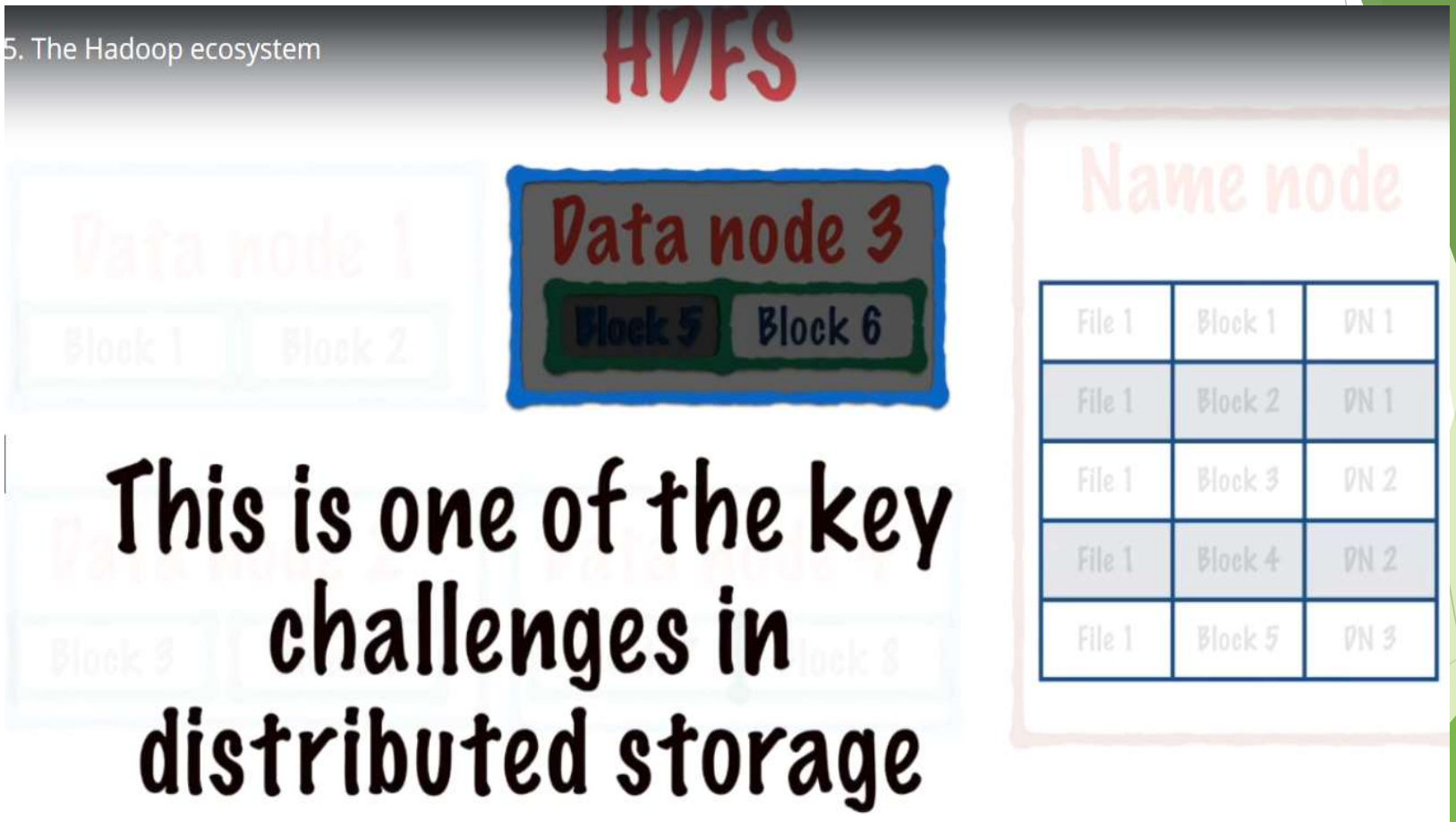
HBase


. The Hadoop ecosystem
HDFS
Data node 3
Block 5
Block 6
What if one of the blocks gets corrupted?
Name node
File 1 Block 1 DN 1
File 1 Block 2 DN 1
File 1 Block 3 DN 2
File 1 Block 4 DN 2
File 1 Block 5 DN 3

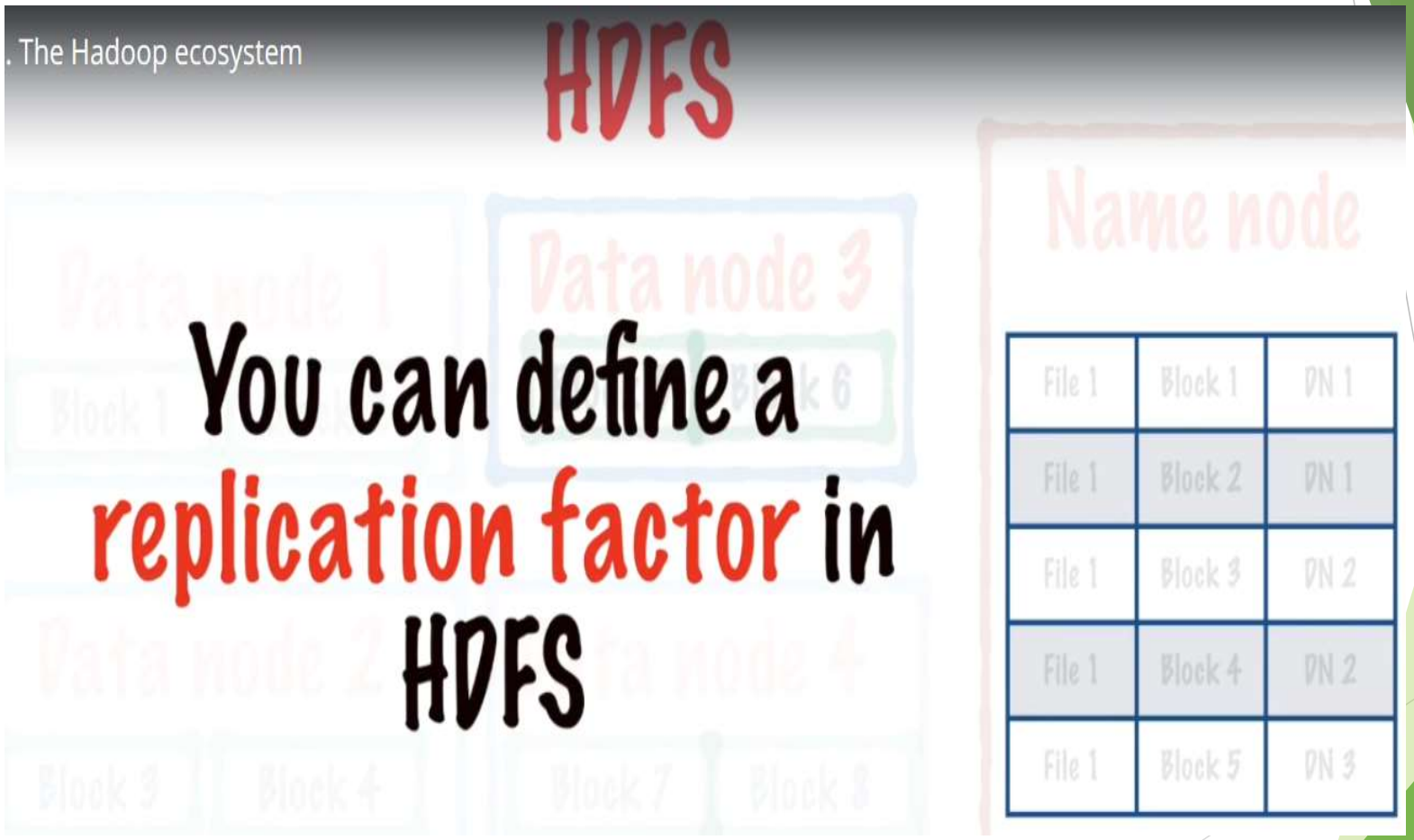
HBase


The Hadoop ecosystem
HDFS
Data node 3
Block 5
Block 6
Name node
File 1
Block 1
DN 1
File 1
Block 2
DN 1
File 1
Block 3
DN 2
File 1
Block 4
DN 2
File 1
Block 5
DN 3
Or one of the data nodes crashes?

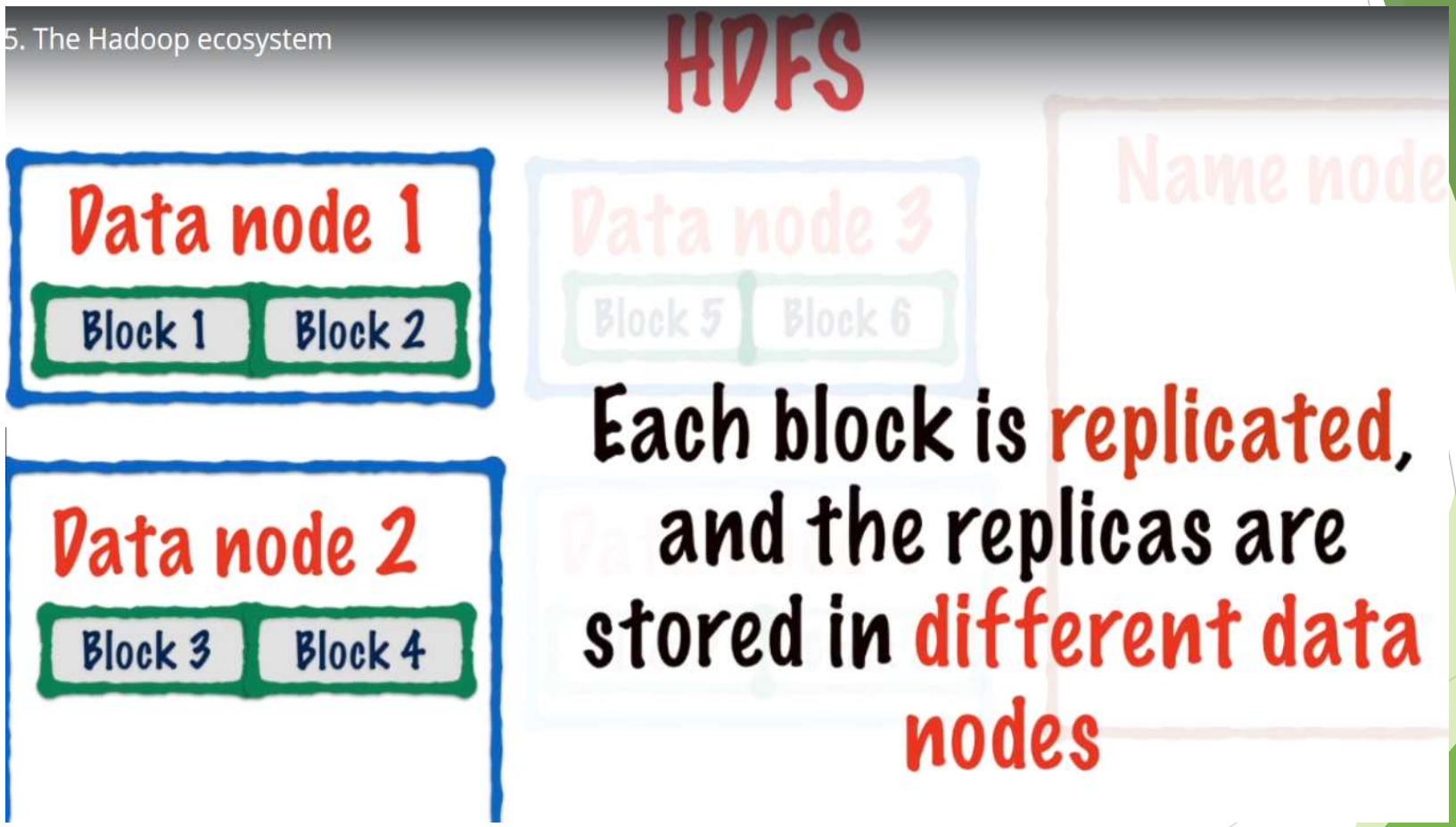
HBase


5. The Hadoop ecosystem
HDFS
Data node 3
Block 5
Block 6
Name node
File 1 Block 1 DN 1
File 1 Block 2 DN 1
File 1 Block 3 DN 2
File 1 Block 4 DN 2
File 1 Block 5 DN 3
This is one of the key challenges in distributed storage

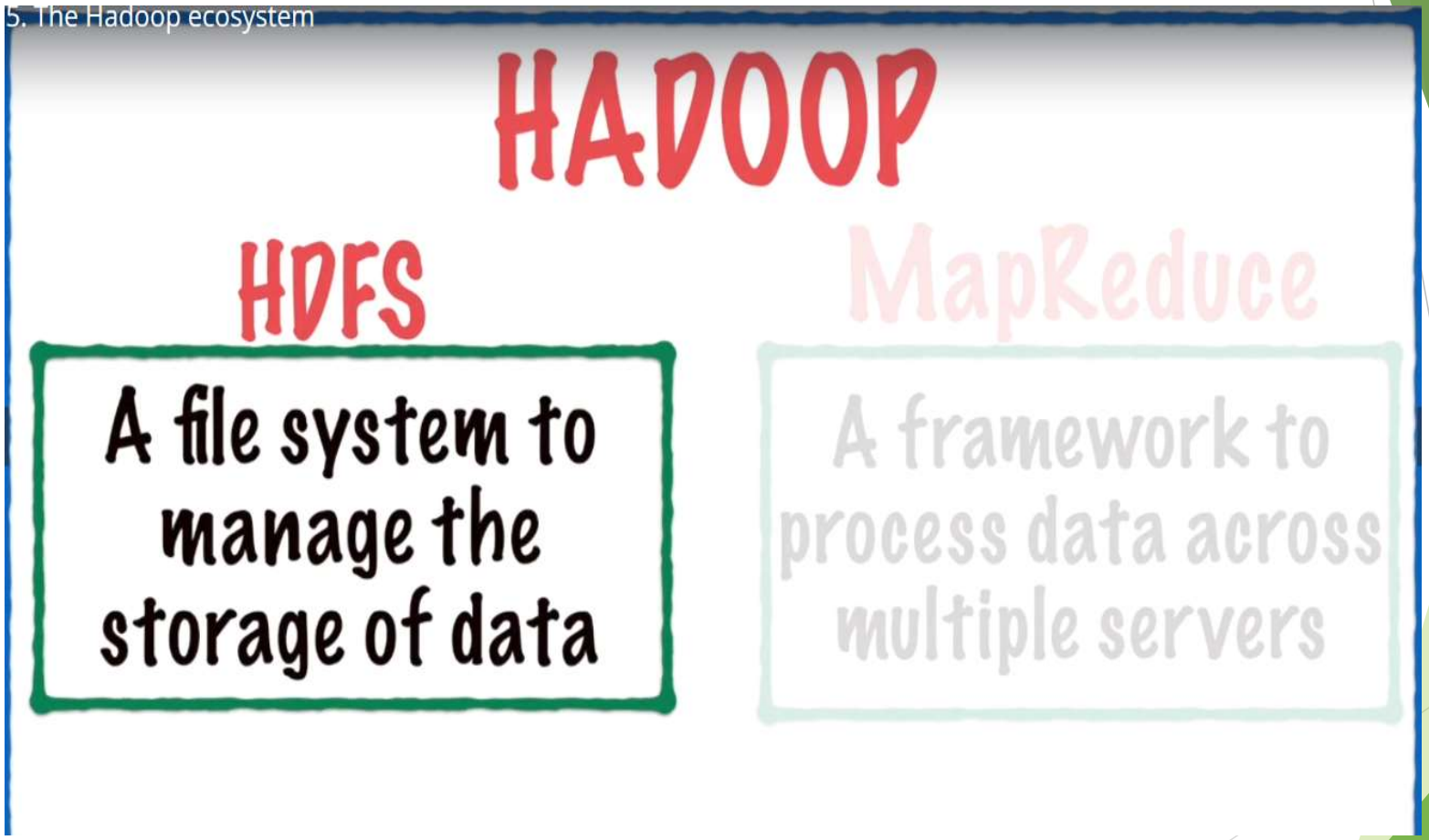
HBase


. The Hadoop ecosystem
HDFS
You can define a replication factor in HDFS
Name node
File 1 Block 1 DN 1
File 1 Block 2 DN 1
File 1 Block 3 DN 2
File 1 Block 4 DN 2
File 1 Block 5 DN 3

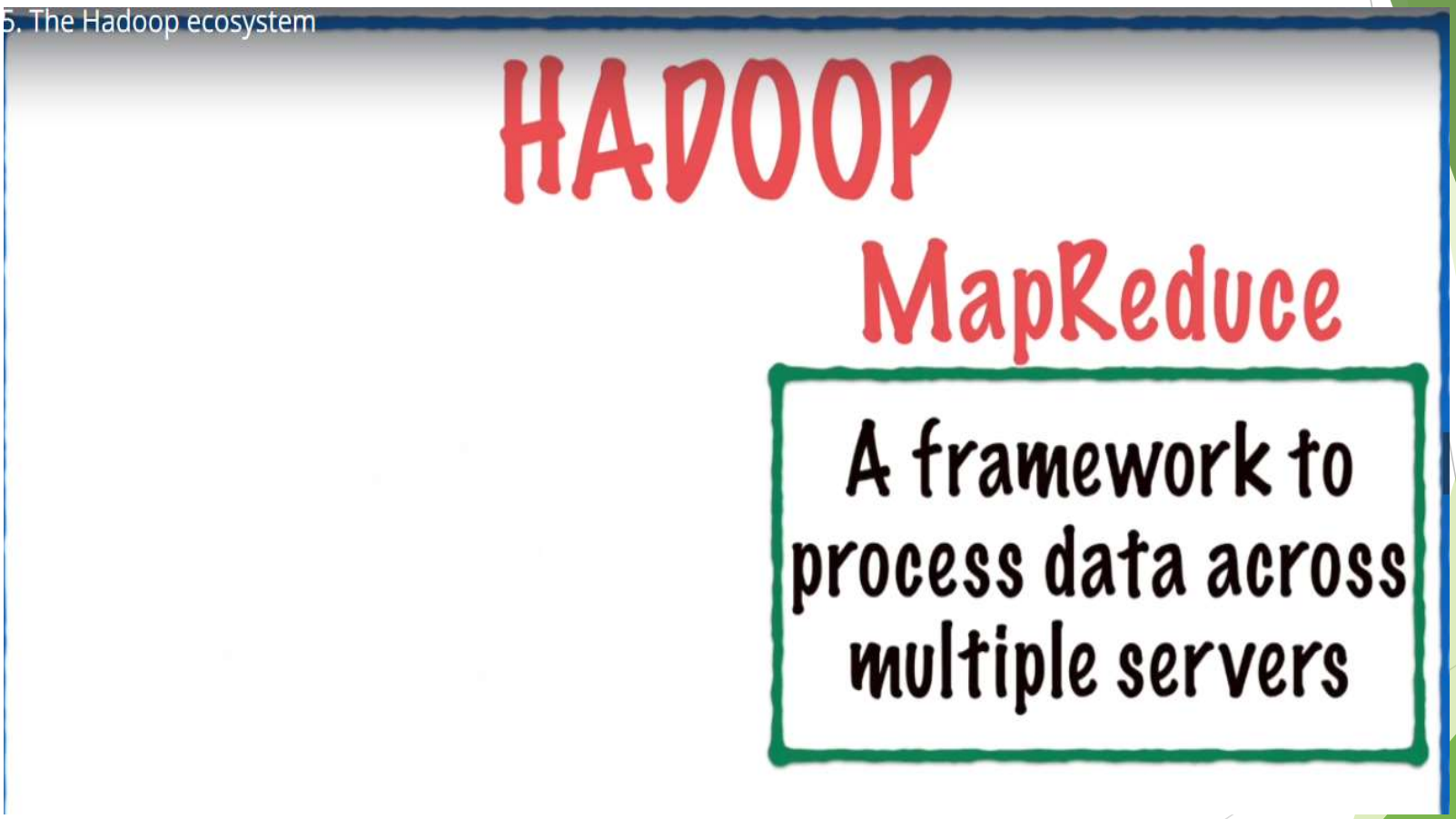
HBase


5. The Hadoop ecosystem
HDFS
Data node 1
Block 1
Block 2
Data node 2
Block 3
Block 4
Each block is replicated, and the replicas are stored in different data nodes

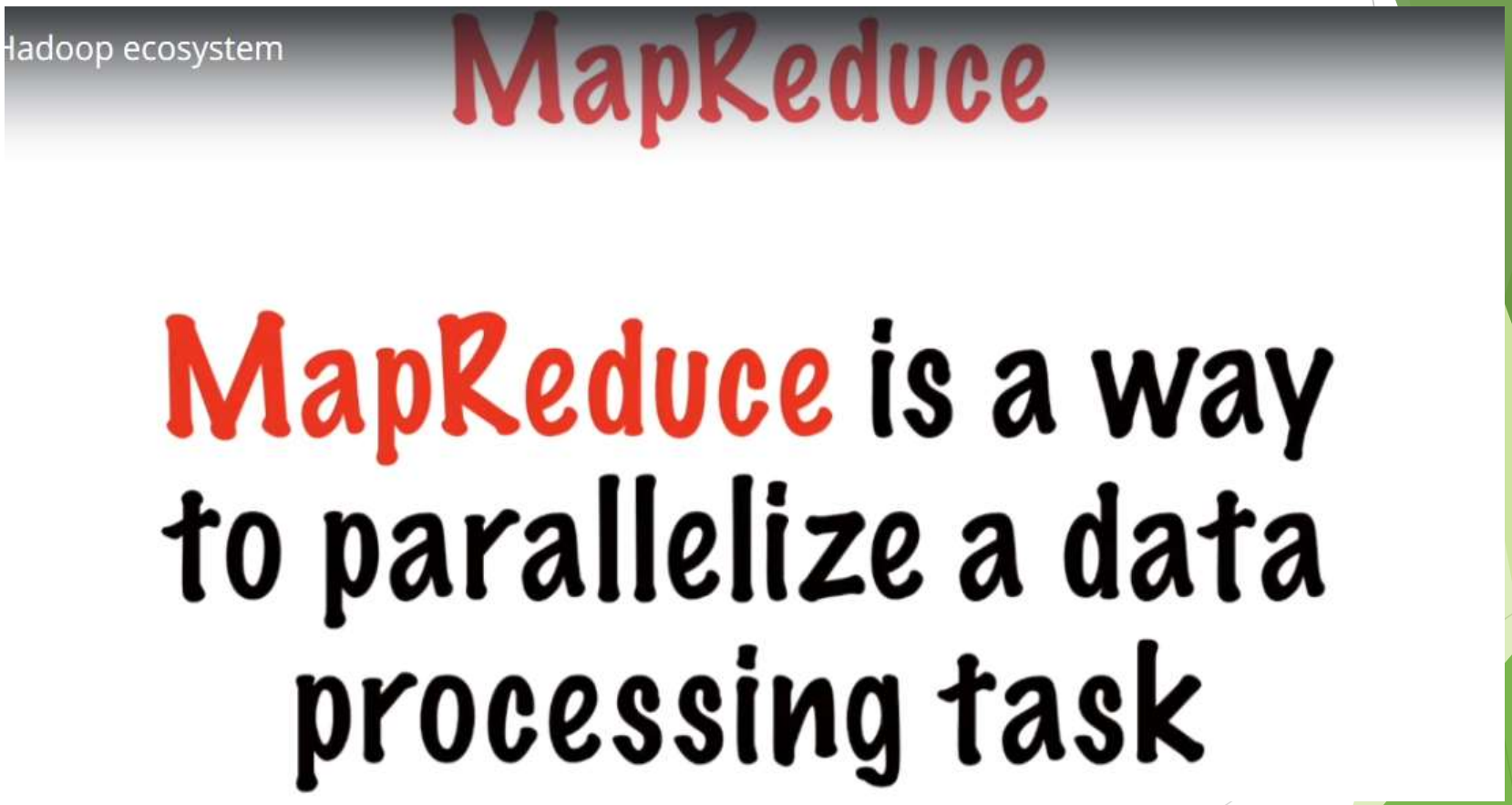
HBase


5. The Hadoop ecosystem
HADOOP
HDFS
A file system to manage the storage of data
MapReduce
A framework to process data across multiple servers

HBase


5. The Hadoop ecosystem
HADOOP
MapReduce
A framework to process data across multiple servers

HBase


Hadoop ecosystem
MapReduce
MapReduce is a way to parallelize a data processing task

HBase

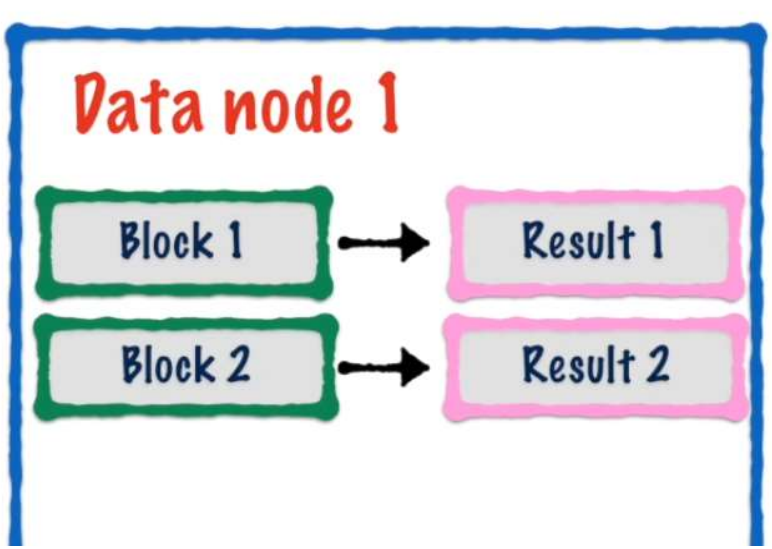
MapReduce tasks have 2 phases

HBase

e Hadoop ecosystem

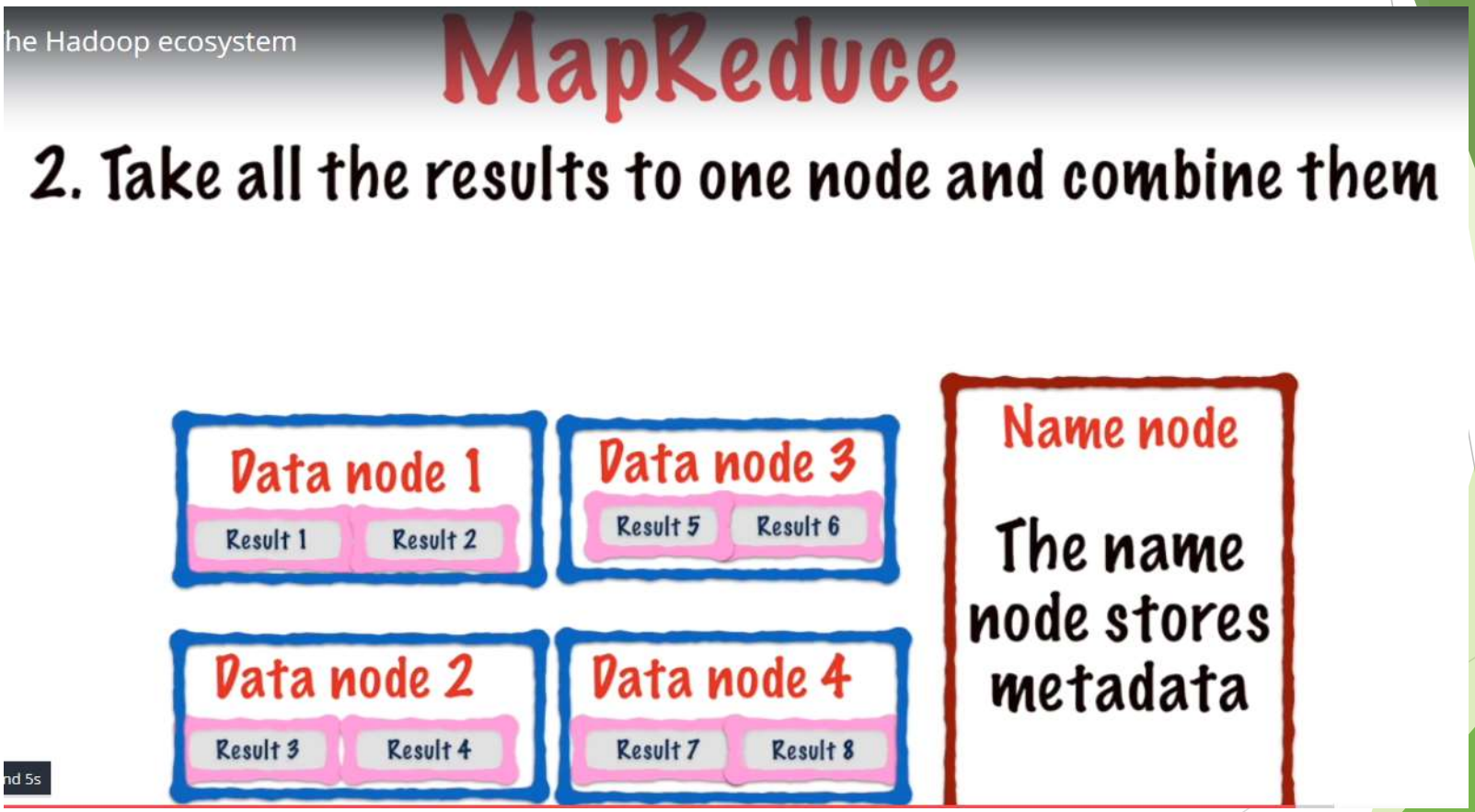
MapReduce

1. Process each block in the node it is stored in

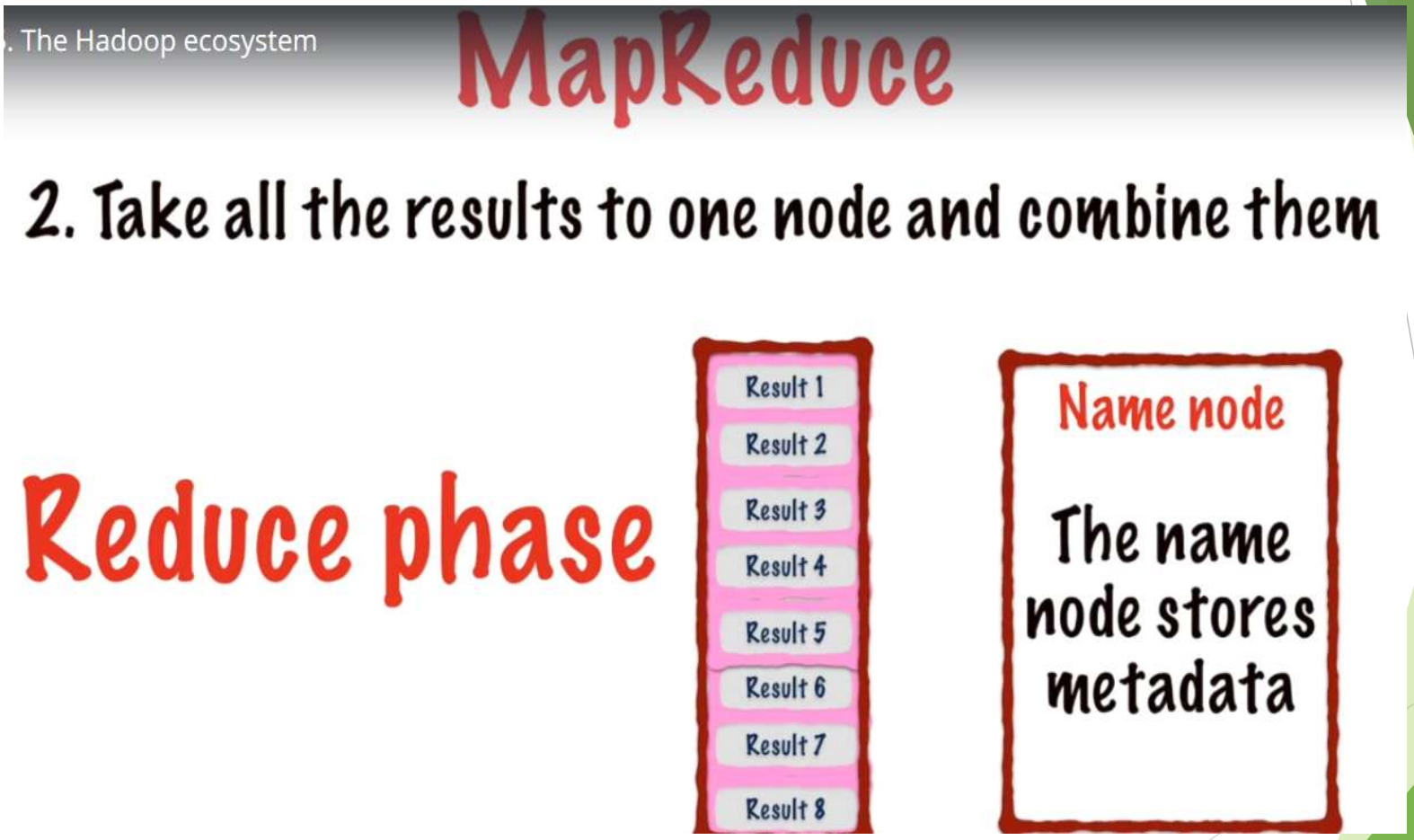


Map phase

HBase


he Hadoop ecosystem
MapReduce
2. Take all the results to one node and combine them
Data node 1
Result 1
Result 2
Data node 3
Result 5
Result 6
Data node 2
Result 3
Result 4
Data node 4
Result 7
Result 8
Name node
The name node stores metadata
nd 5s

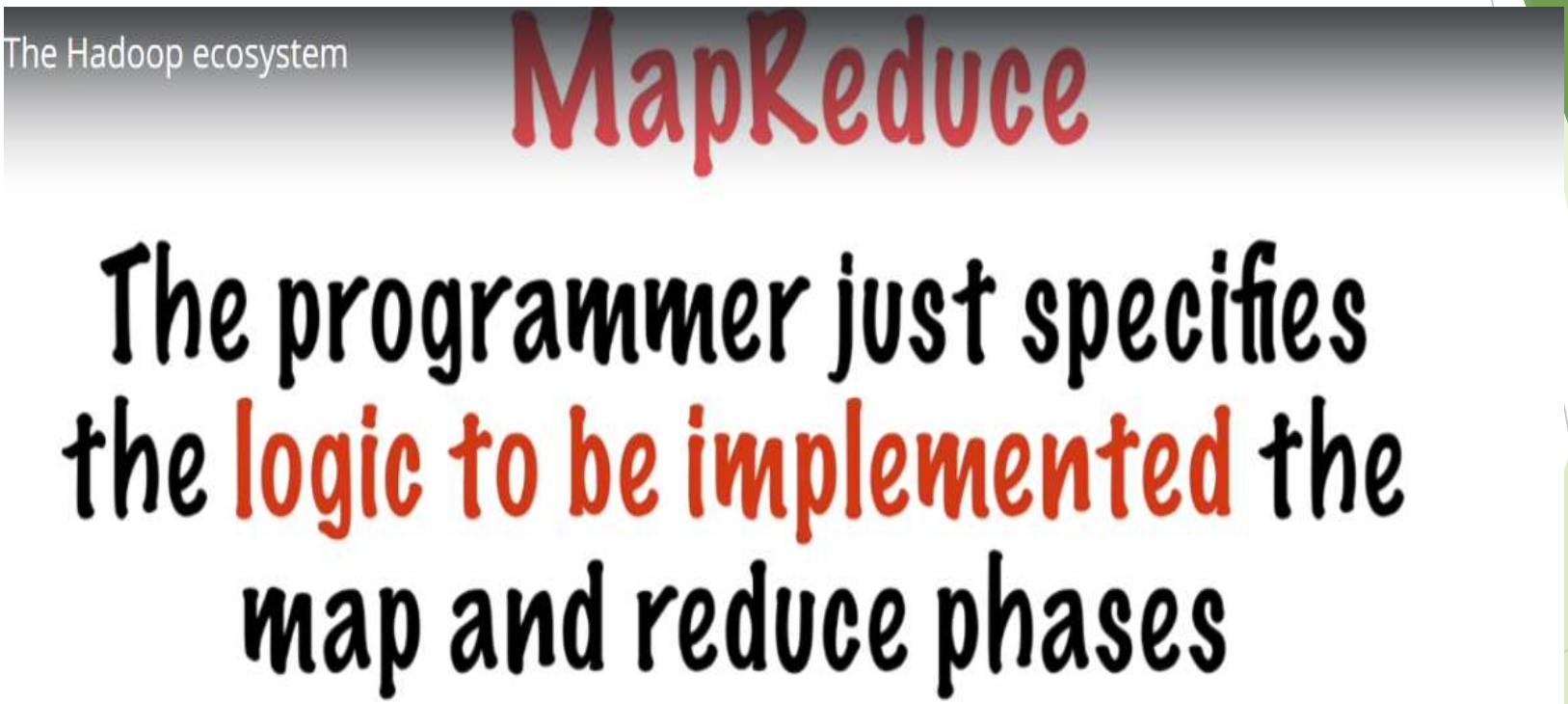
HBase


. The Hadoop ecosystem
MapReduce
2. Take all the results to one node and combine them
Reduce phase
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Result 8
Name node
The name node stores metadata

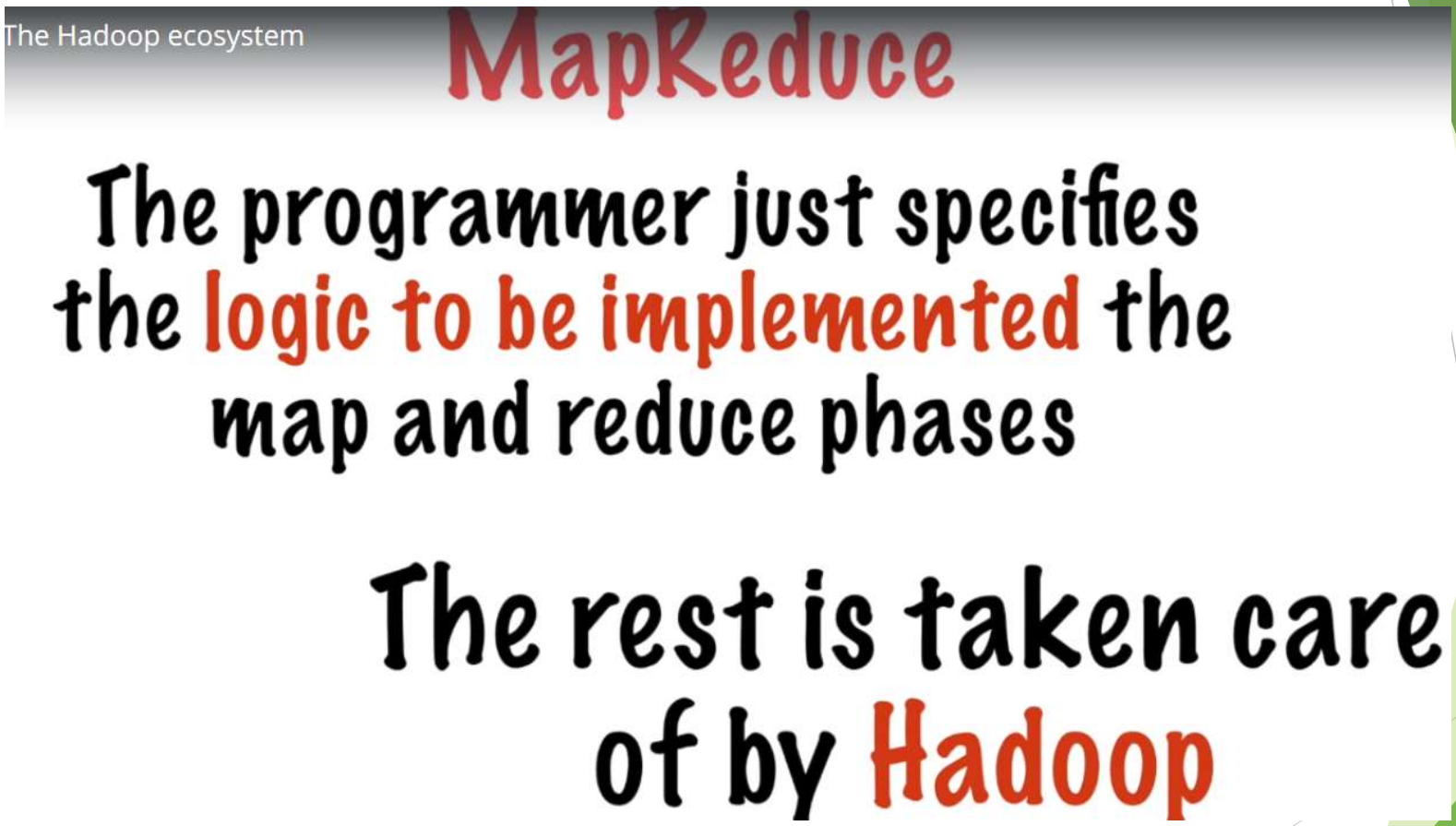
HBase


he Hadoop ecosystem
MapReduce
Any data processing task can be expressed as a chain of map reduce operations

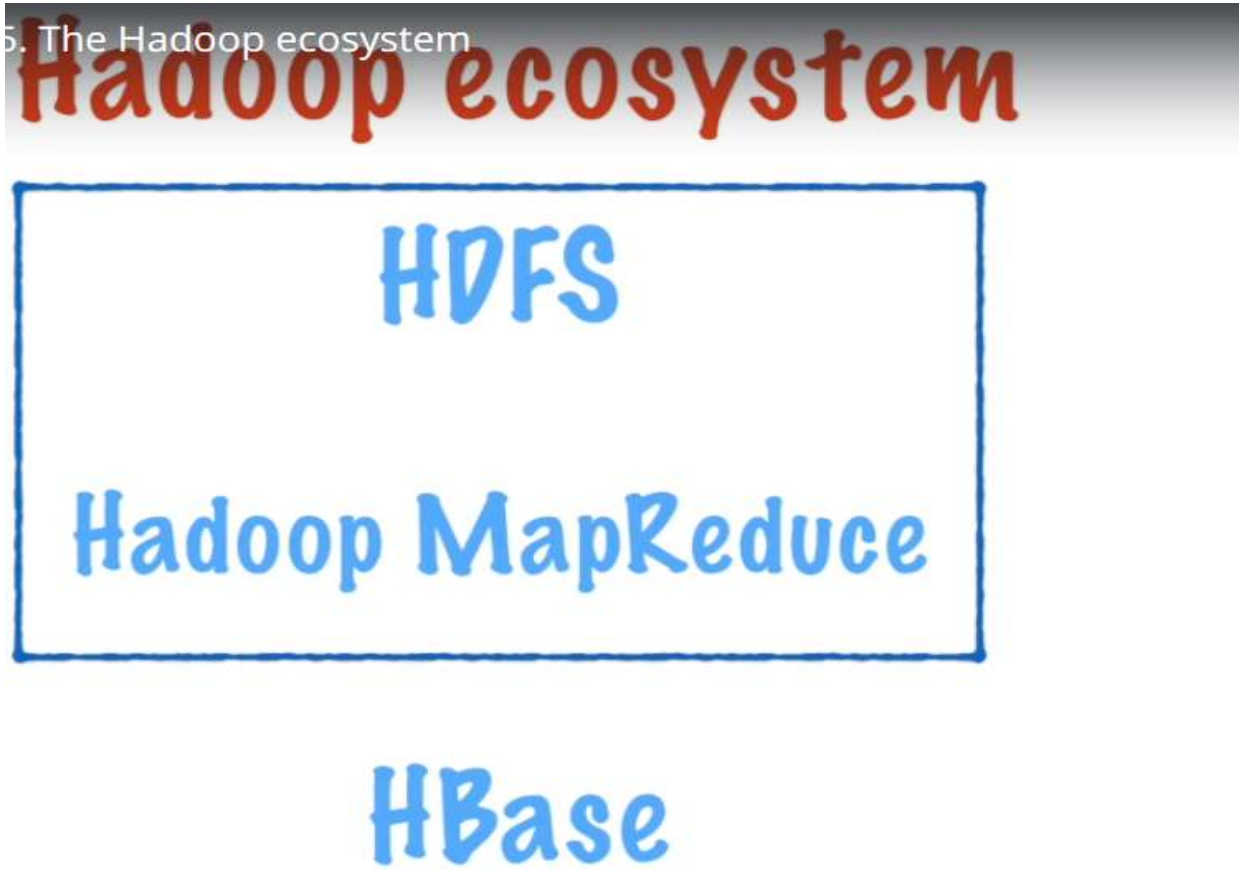
HBase


The Hadoop ecosystem
MapReduce
The programmer just specifies the logic to be implemented the map and reduce phases

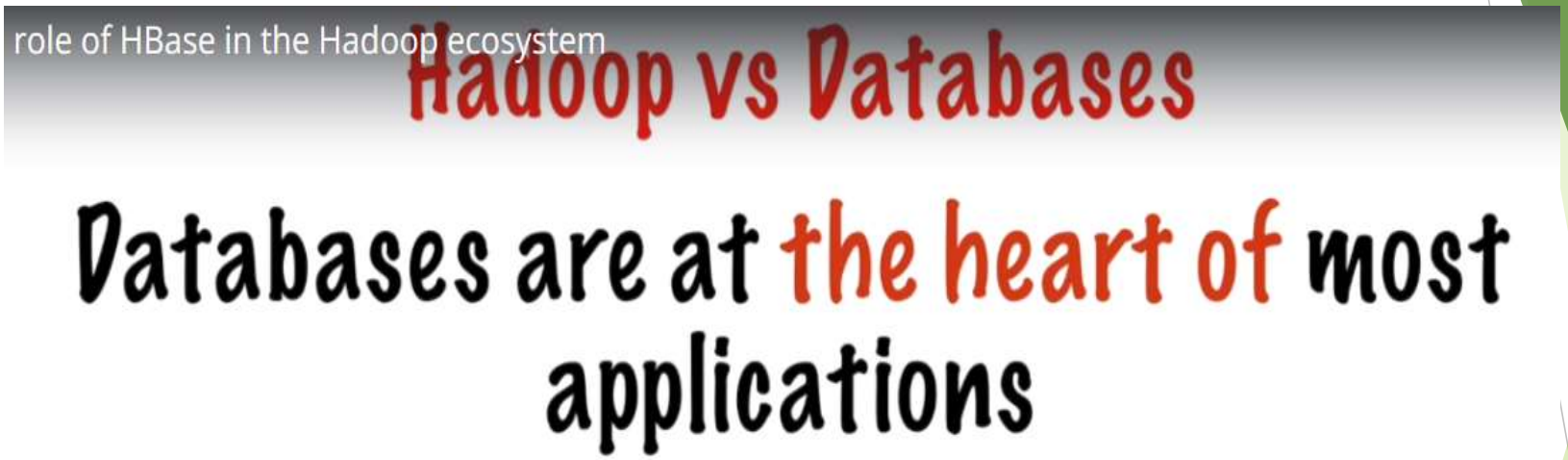
HBase


The Hadoop ecosystem
MapReduce
The programmer just specifies the logic to be implemented the map and reduce phases
The rest is taken care of by Hadoop

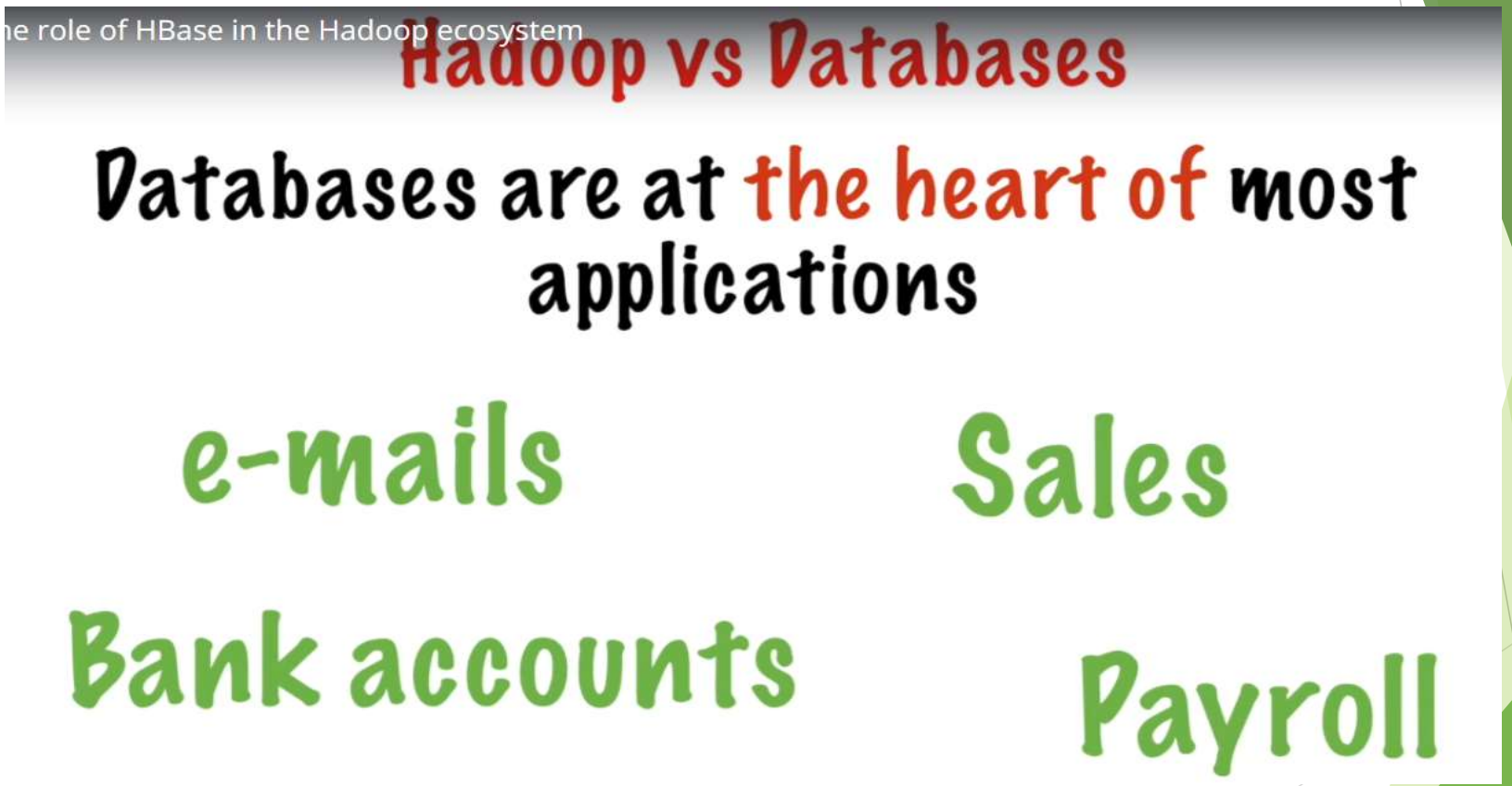
HBase


5. The Hadoop ecosystem
Hadoop ecosystem
HDFS
Hadoop MapReduce
HBase

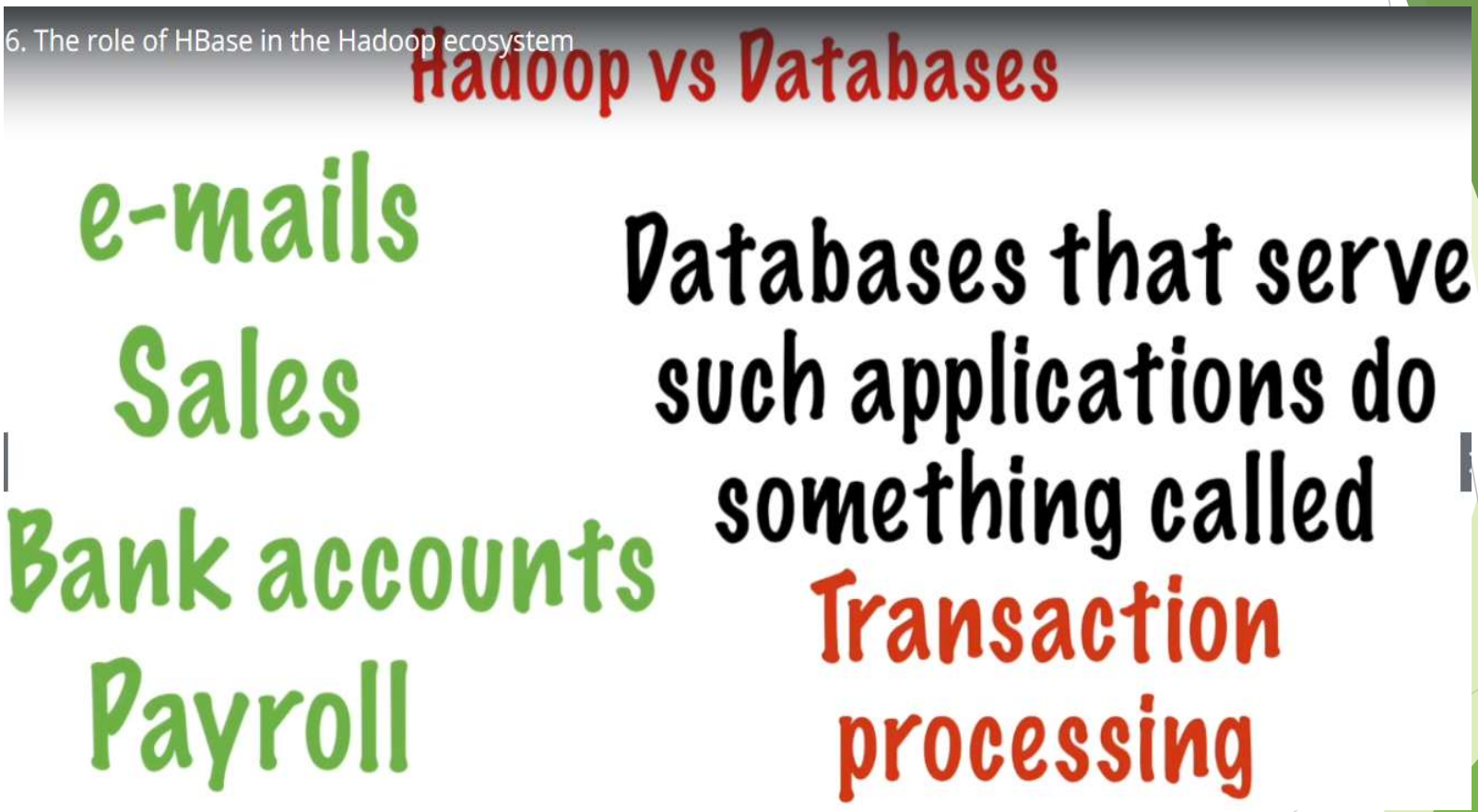
HBase


role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Databases are at the heart of most applications

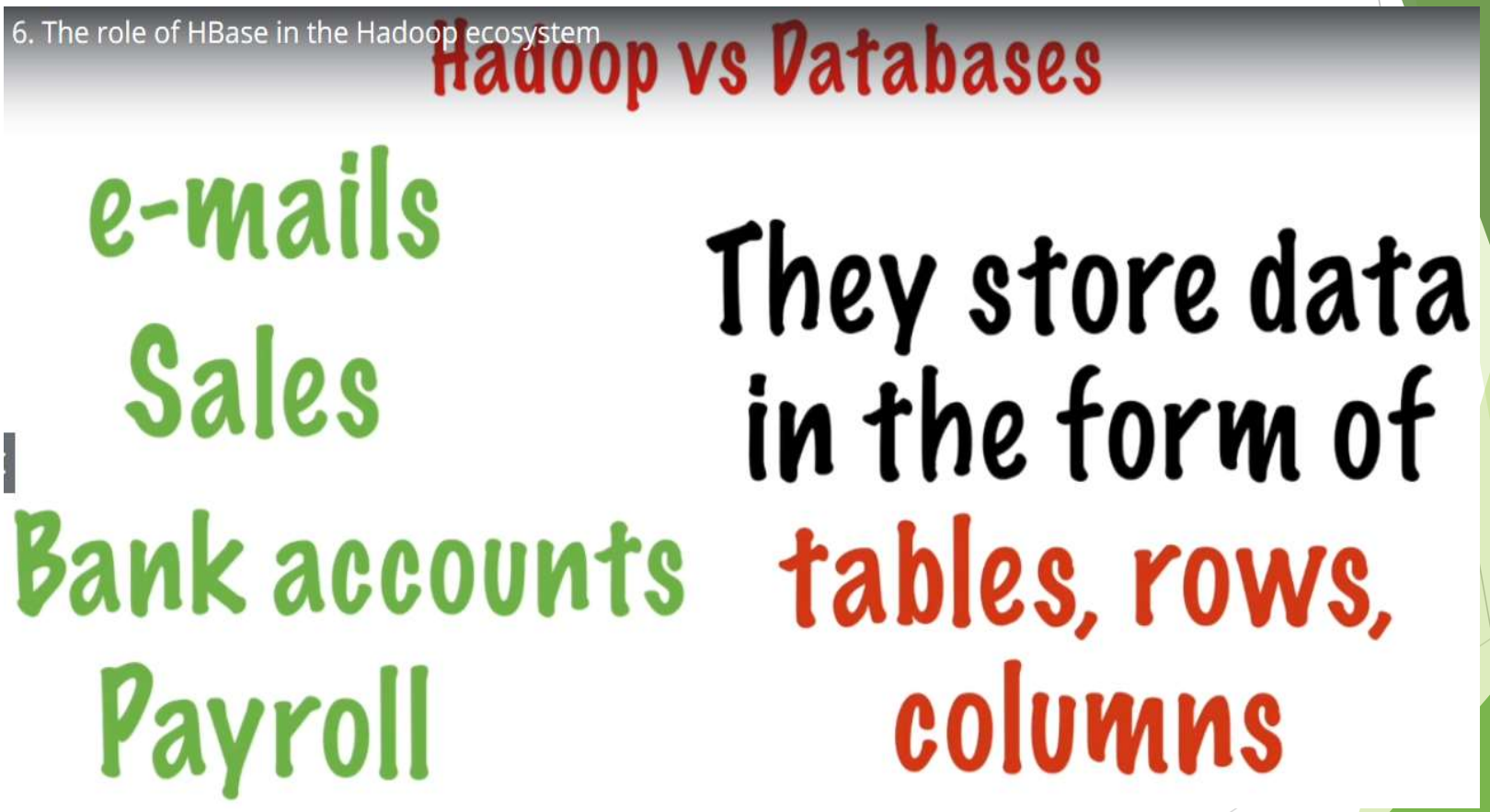
HBase


e role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Databases are at the heart of most applications
e-mails
Sales
Bank accounts
Payroll

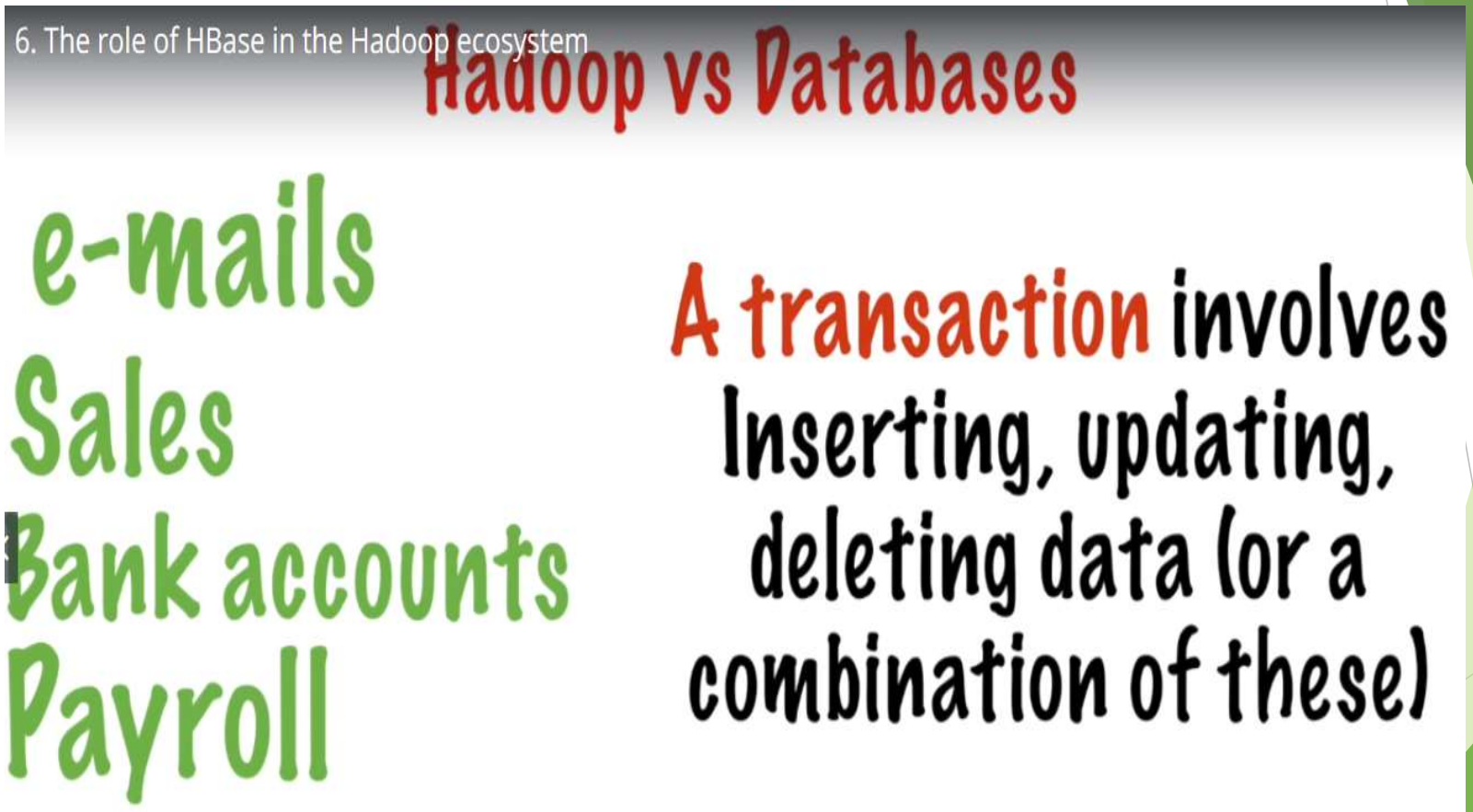
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
e-mails
Sales
Bank accounts
Payroll
Databases that serve such applications do something called
Transaction processing

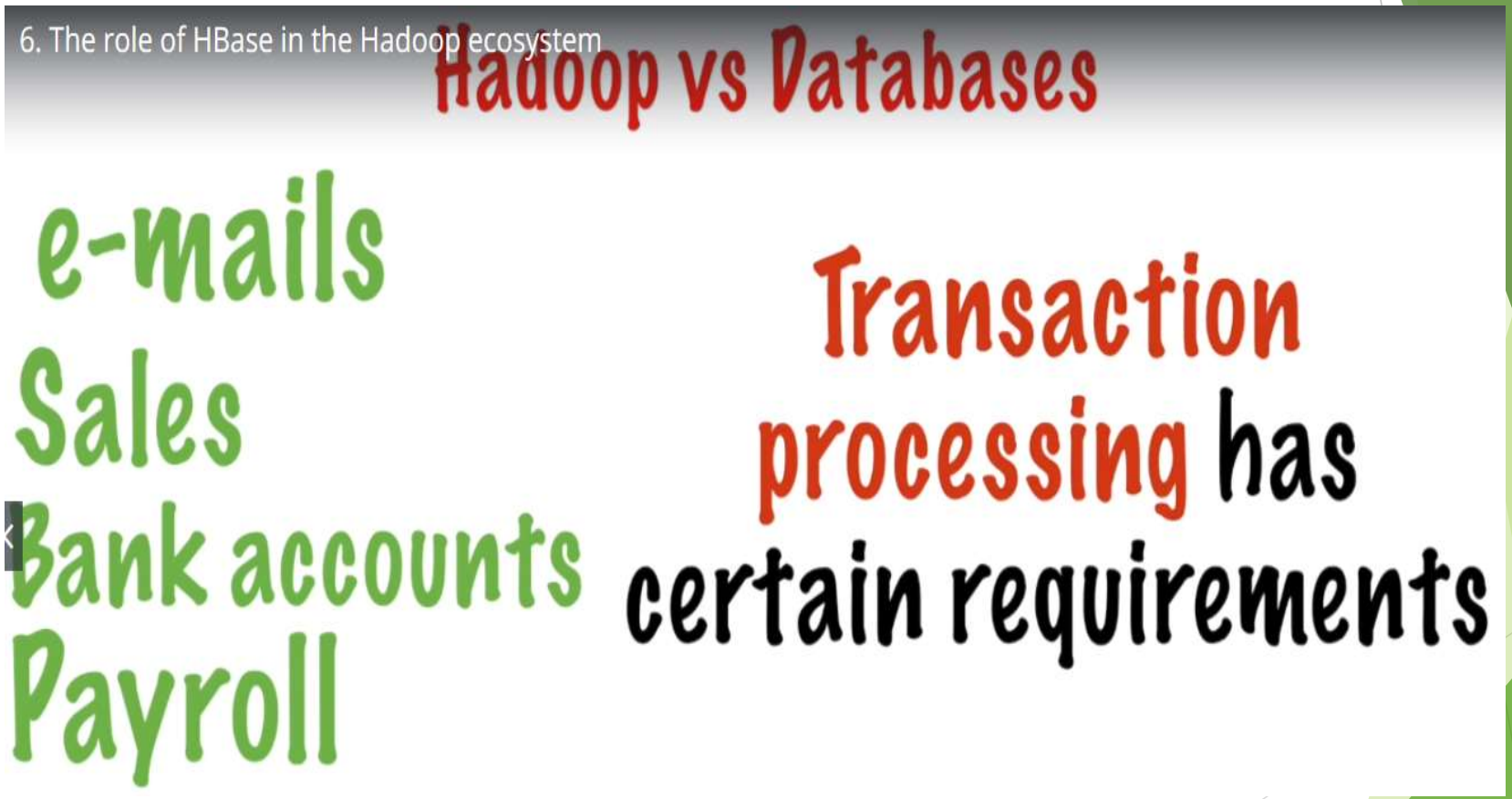
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
e-mails
Sales
Bank accounts
Payroll
They store data in the form of tables, rows, columns

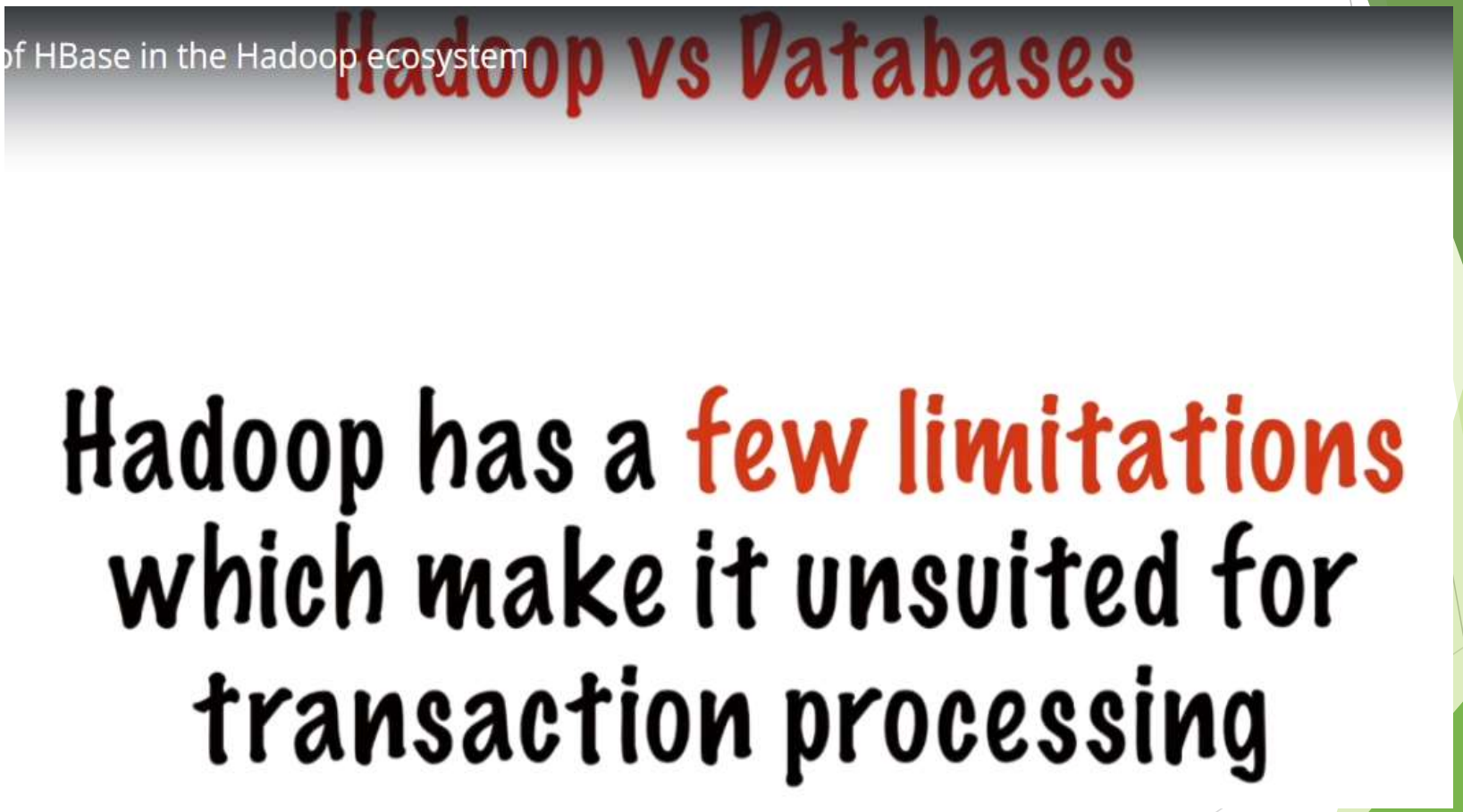
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
e-mails
Sales
Bank accounts
Payroll
A transaction involves Inserting, updating, deleting data (or a combination of these)

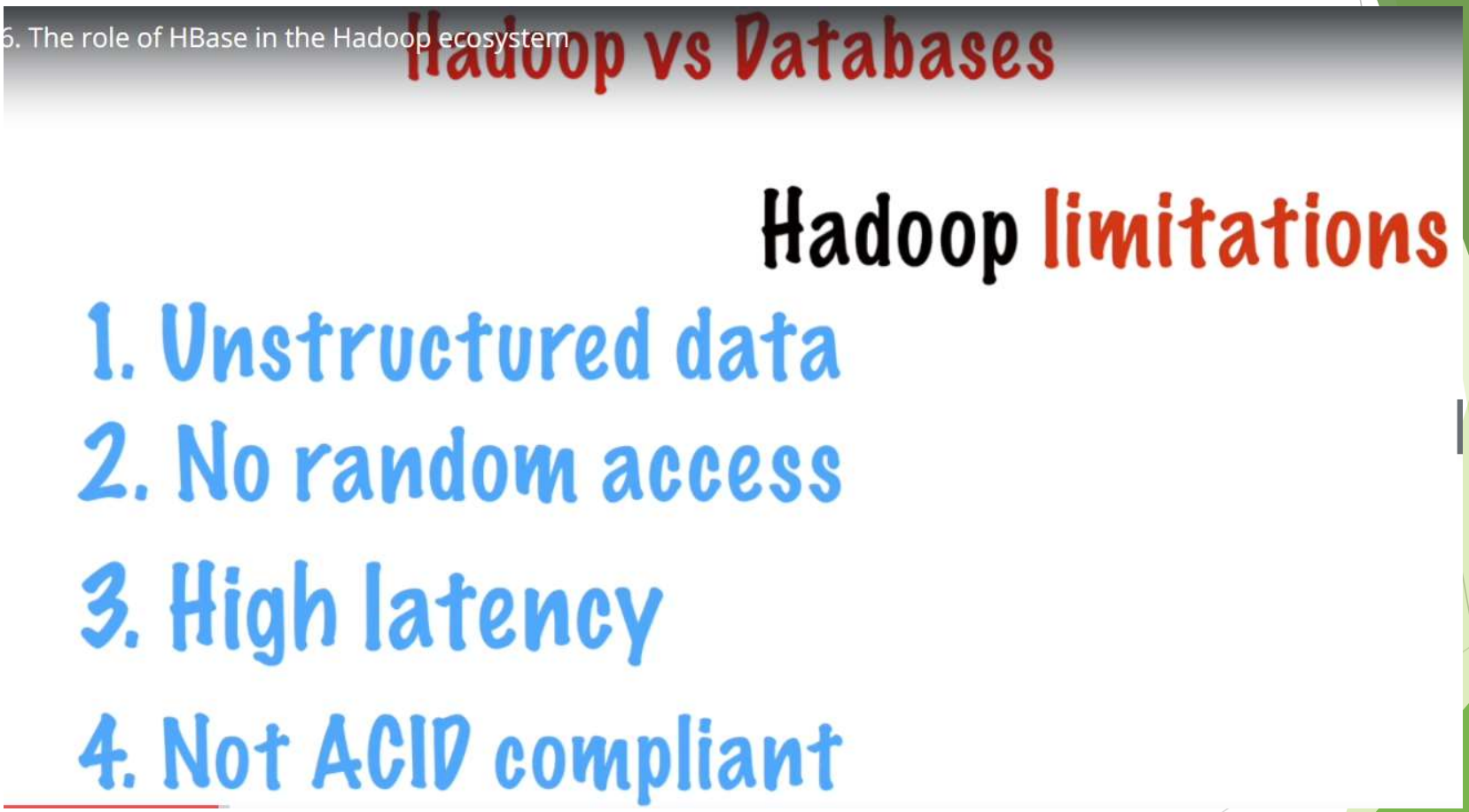
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
e-mails
Sales
Bank accounts
Payroll
Transaction processing has certain requirements

HBase


of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop has a few limitations
which make it unsuited for
transaction processing

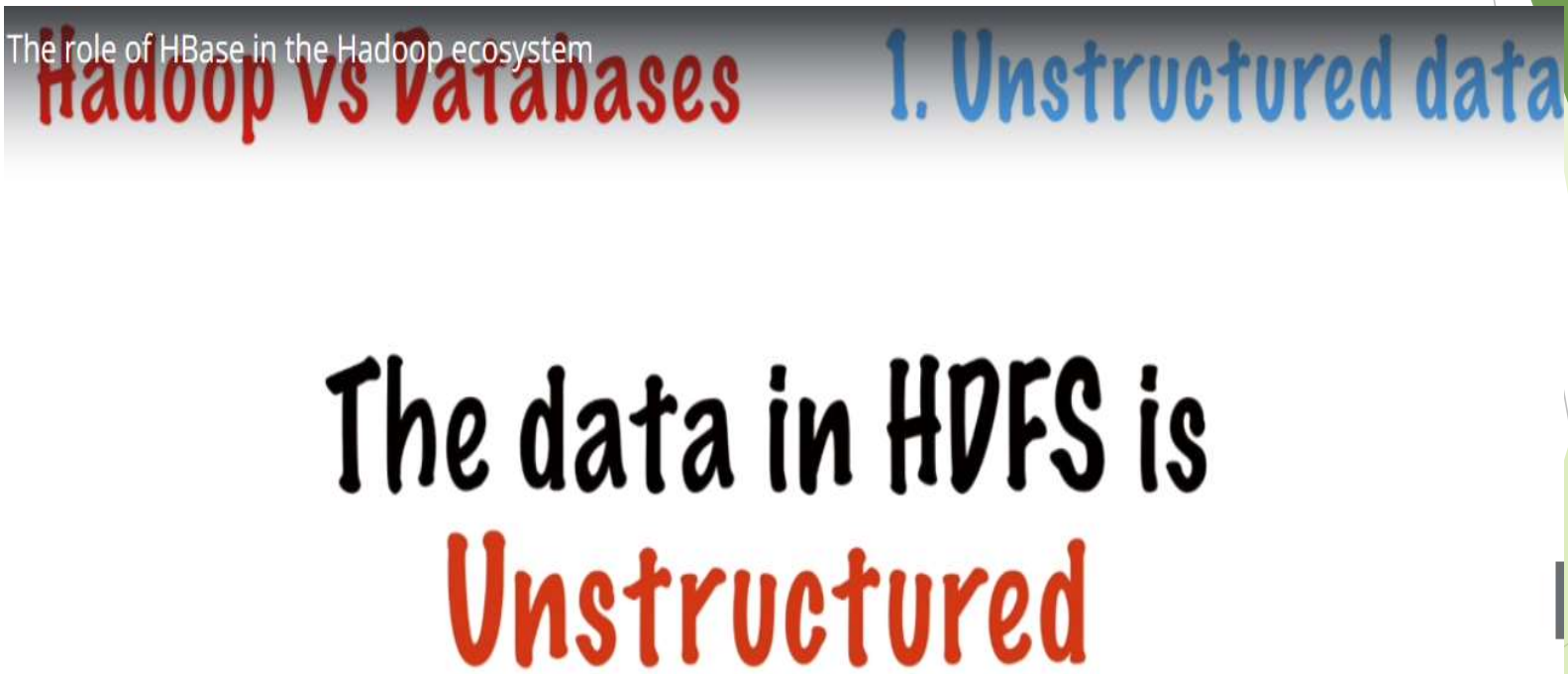
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1. Unstructured data
2. No random access
3. High latency
4. Not ACID compliant

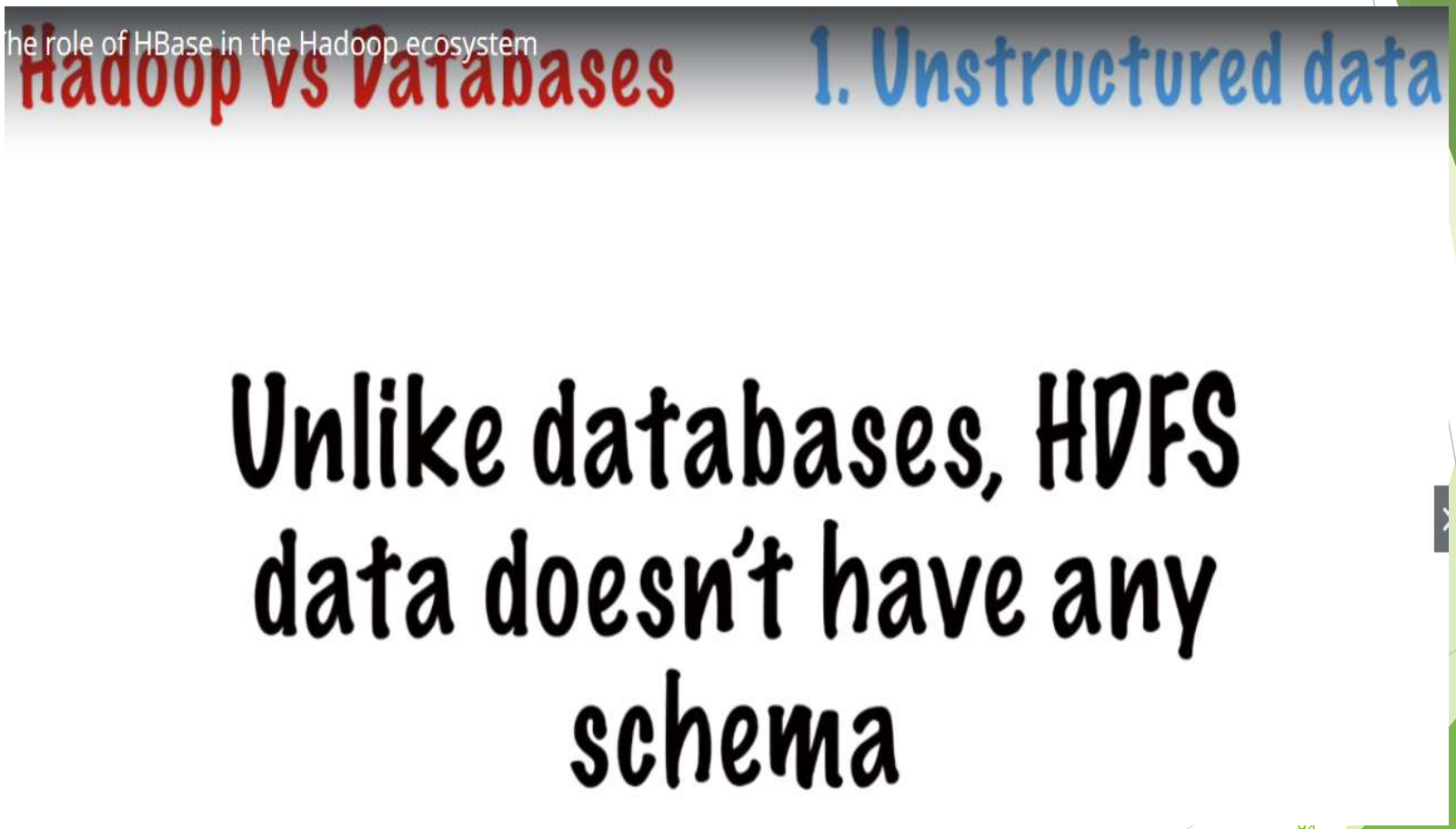
HBase

Hadoop stores data in HDFS

HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
The data in HDFS is
Unstructured

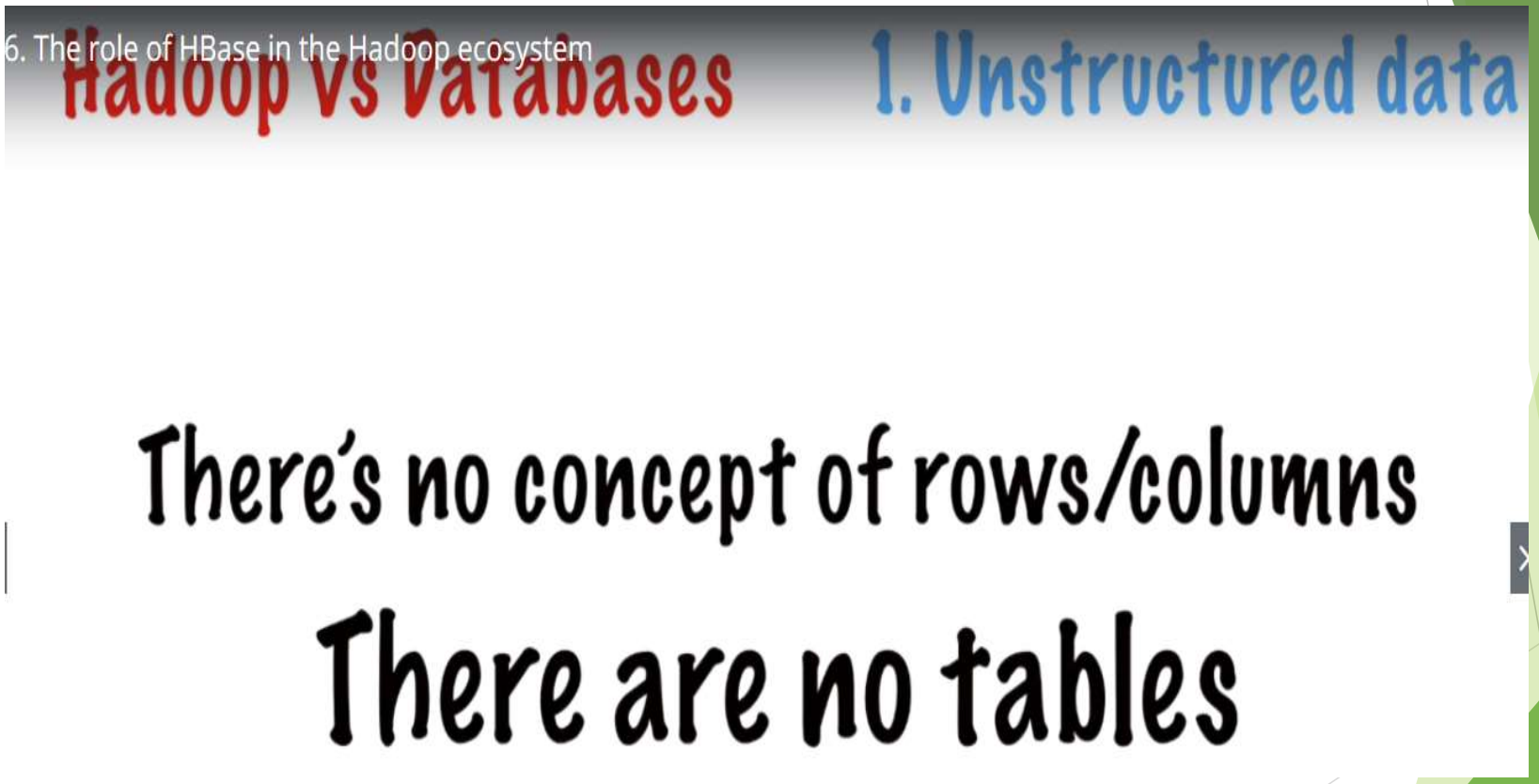
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
Unlike databases, HDFS data doesn't have any schema

HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
It's basically in the form of files
Text files
Log files
Video/Audio files
wind 5s

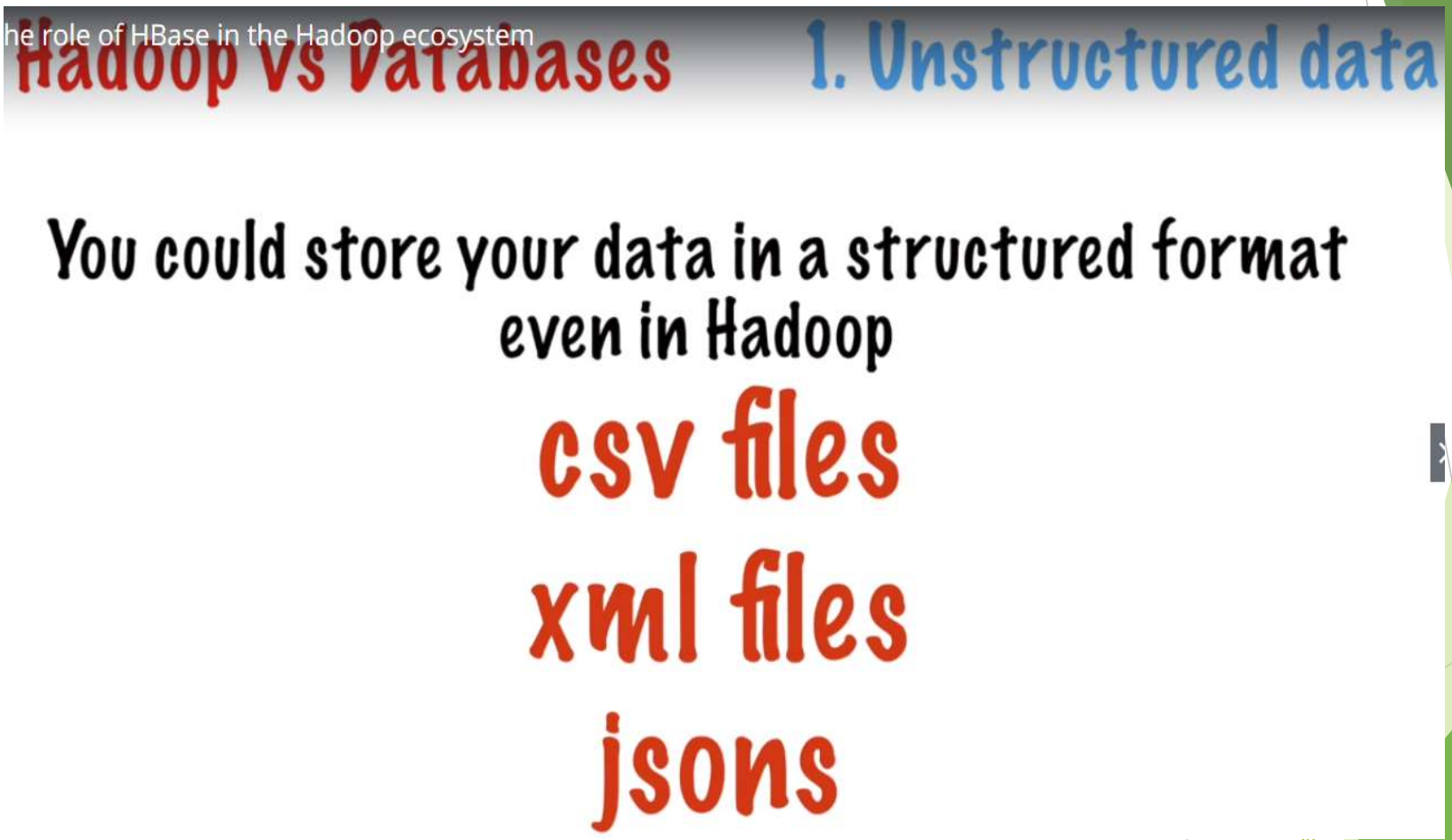
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
There's no concept of rows/columns
There are no tables

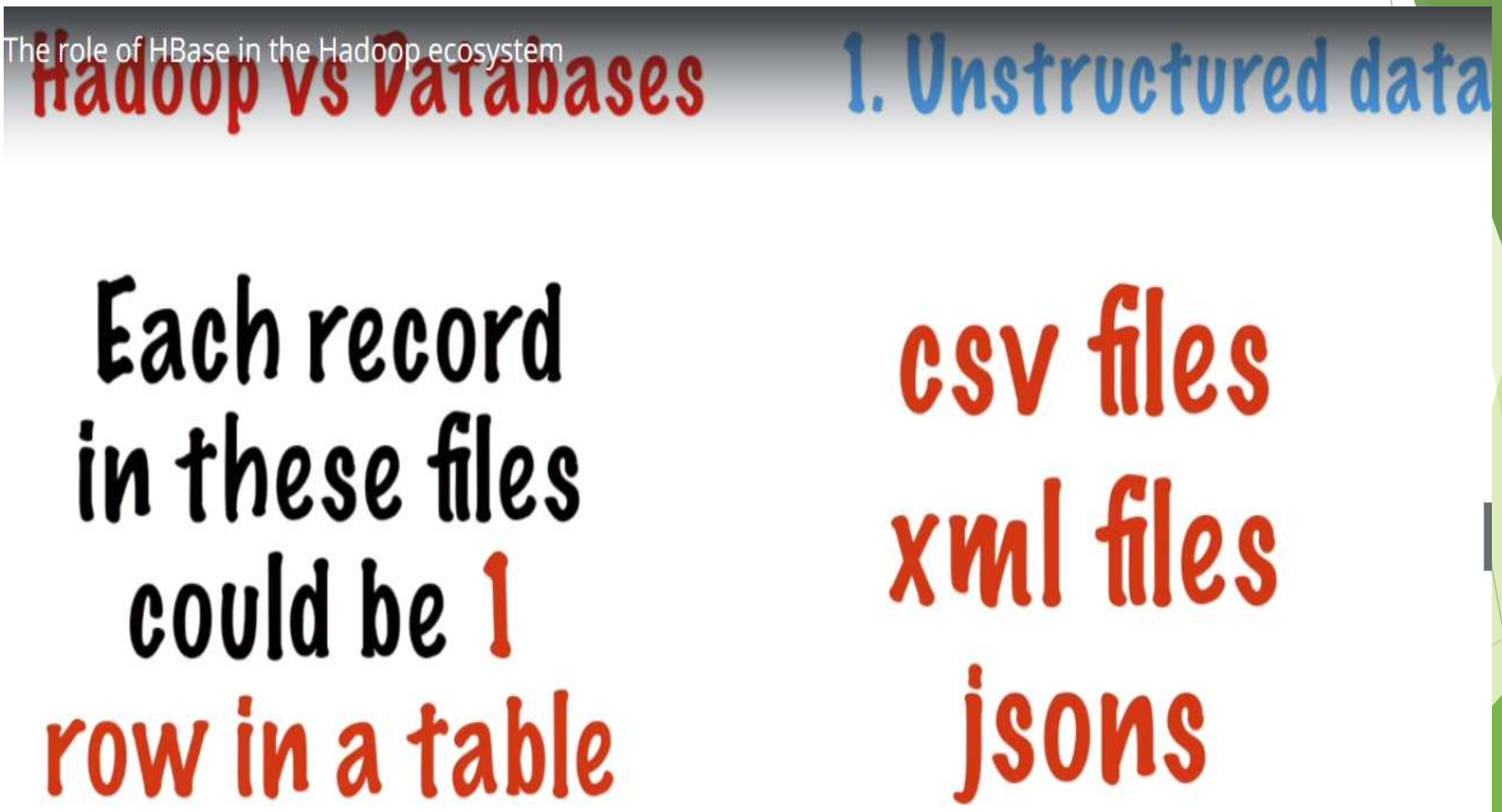
HBase

This is not to say that Hadoop can't be used to store structured data

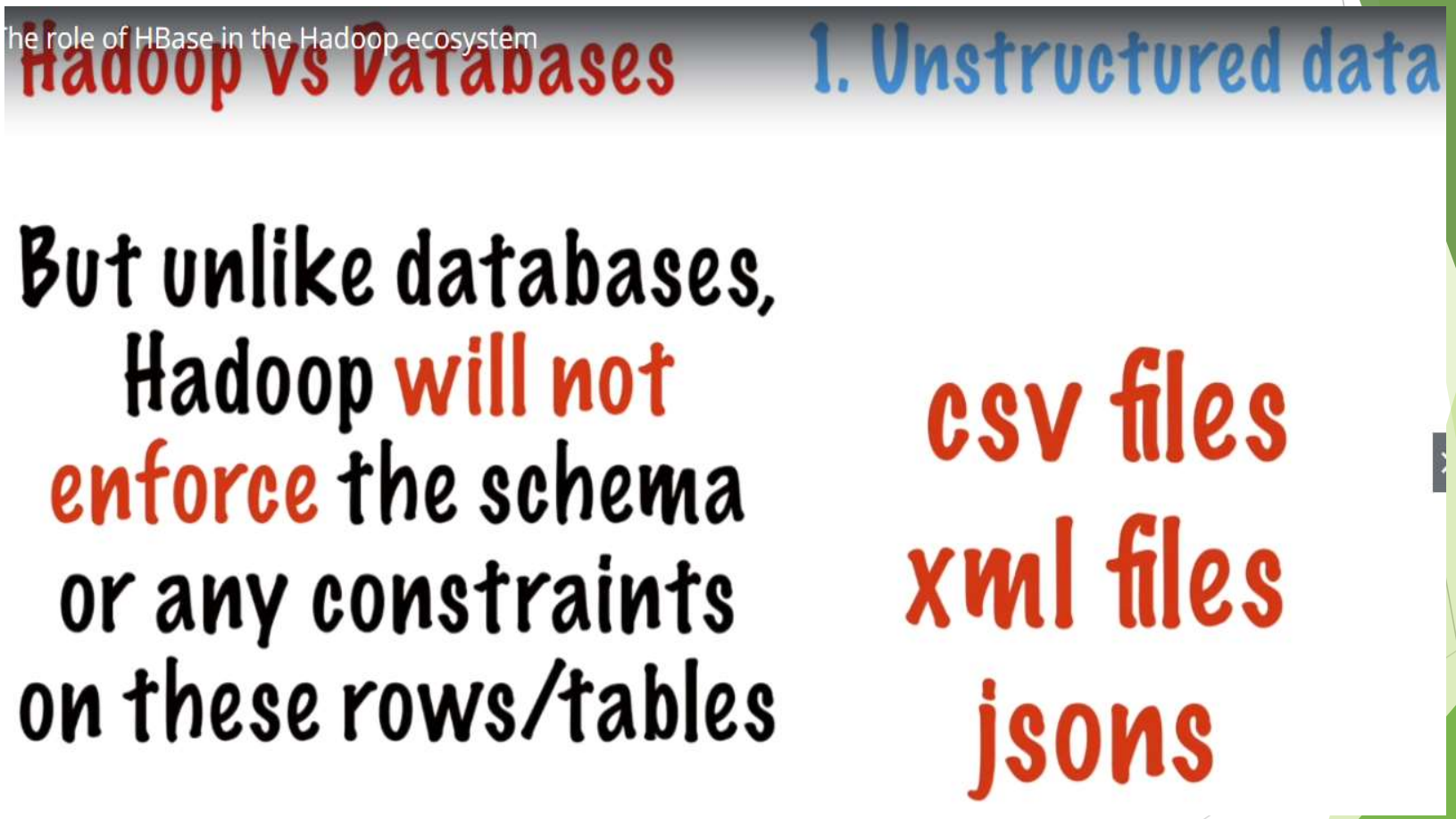
HBase


he role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
You could store your data in a structured format even in Hadoop
csv files
xml files
jsons

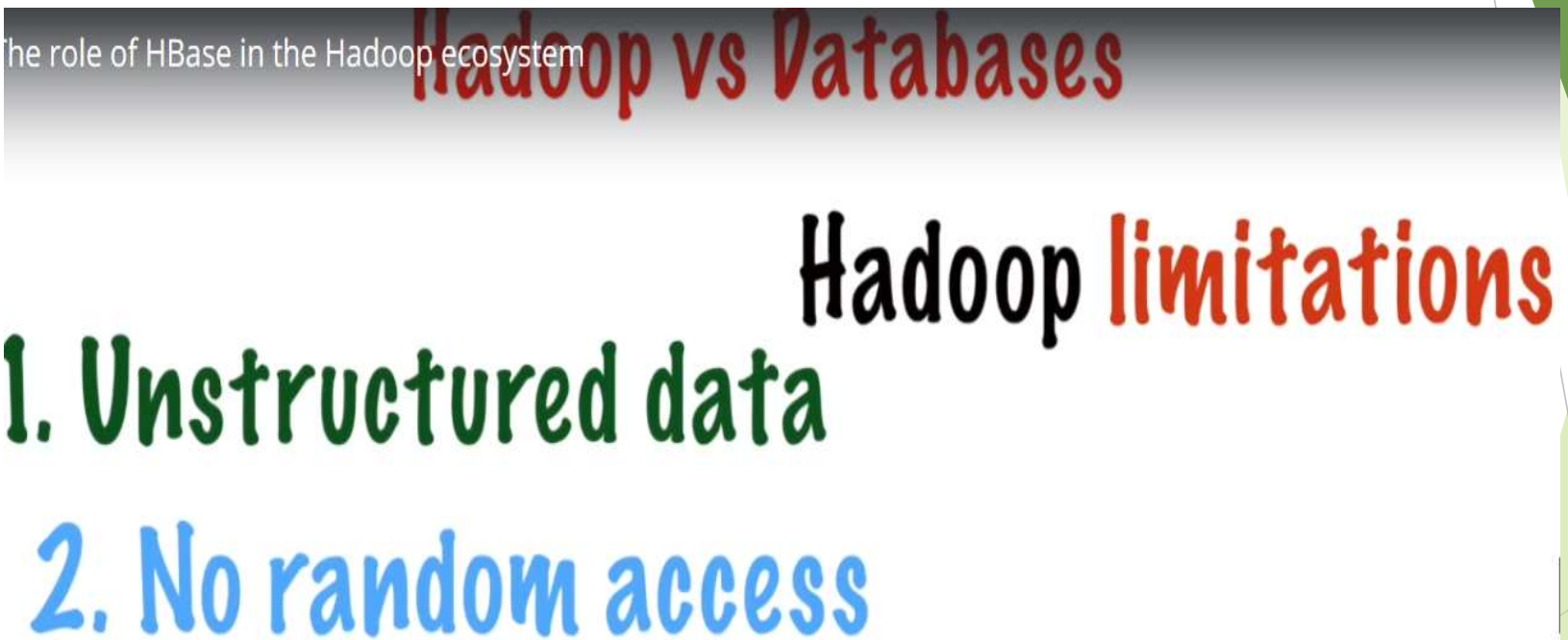
HBase


The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
1. Unstructured data
Each record in these files could be 1 row in a table
csv files
xml files
jsons

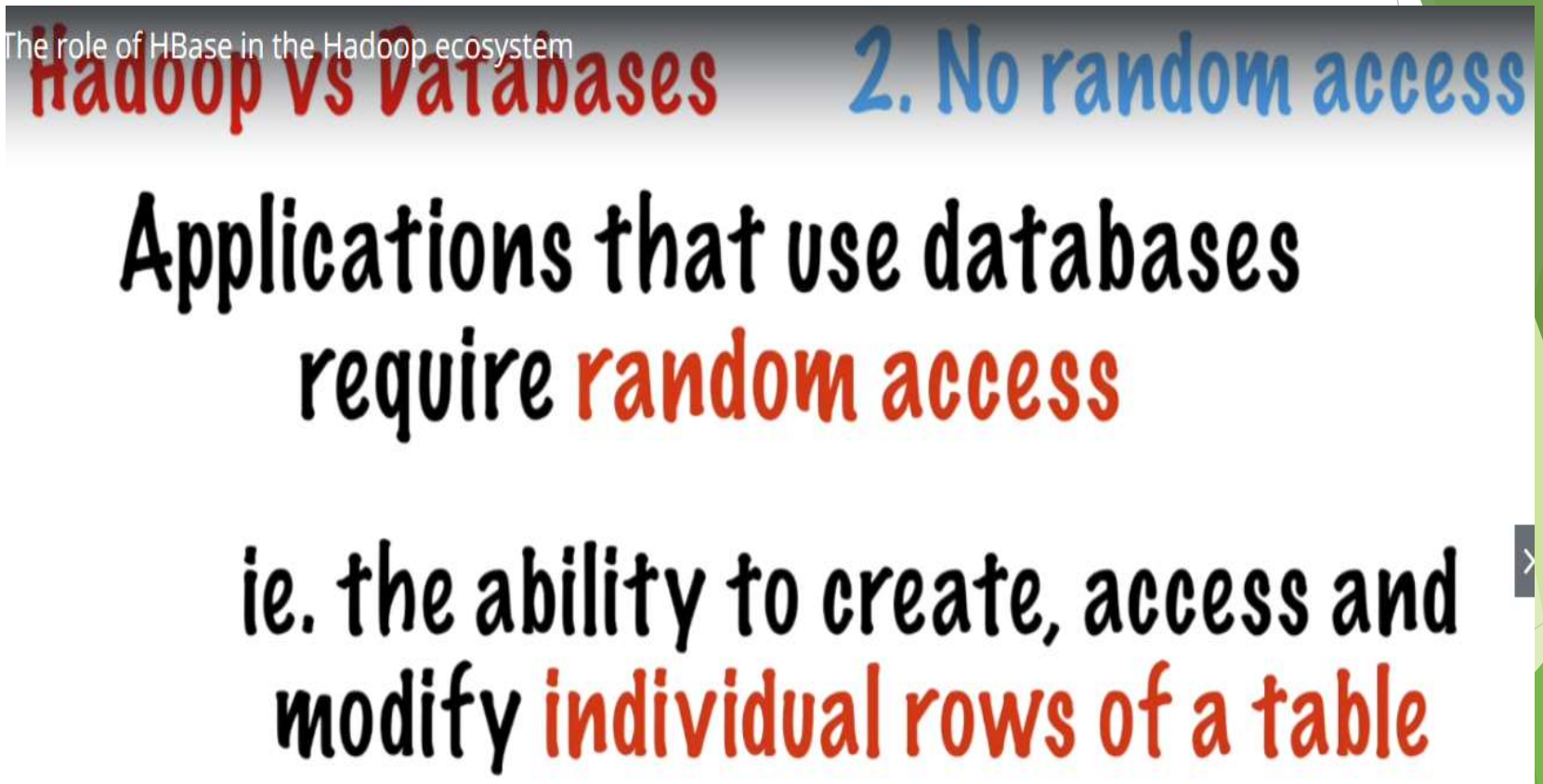
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
1. Unstructured data
But unlike databases,
Hadoop will not
enforce the schema
or any constraints
on these rows/tables
csv files
xml files
jsons

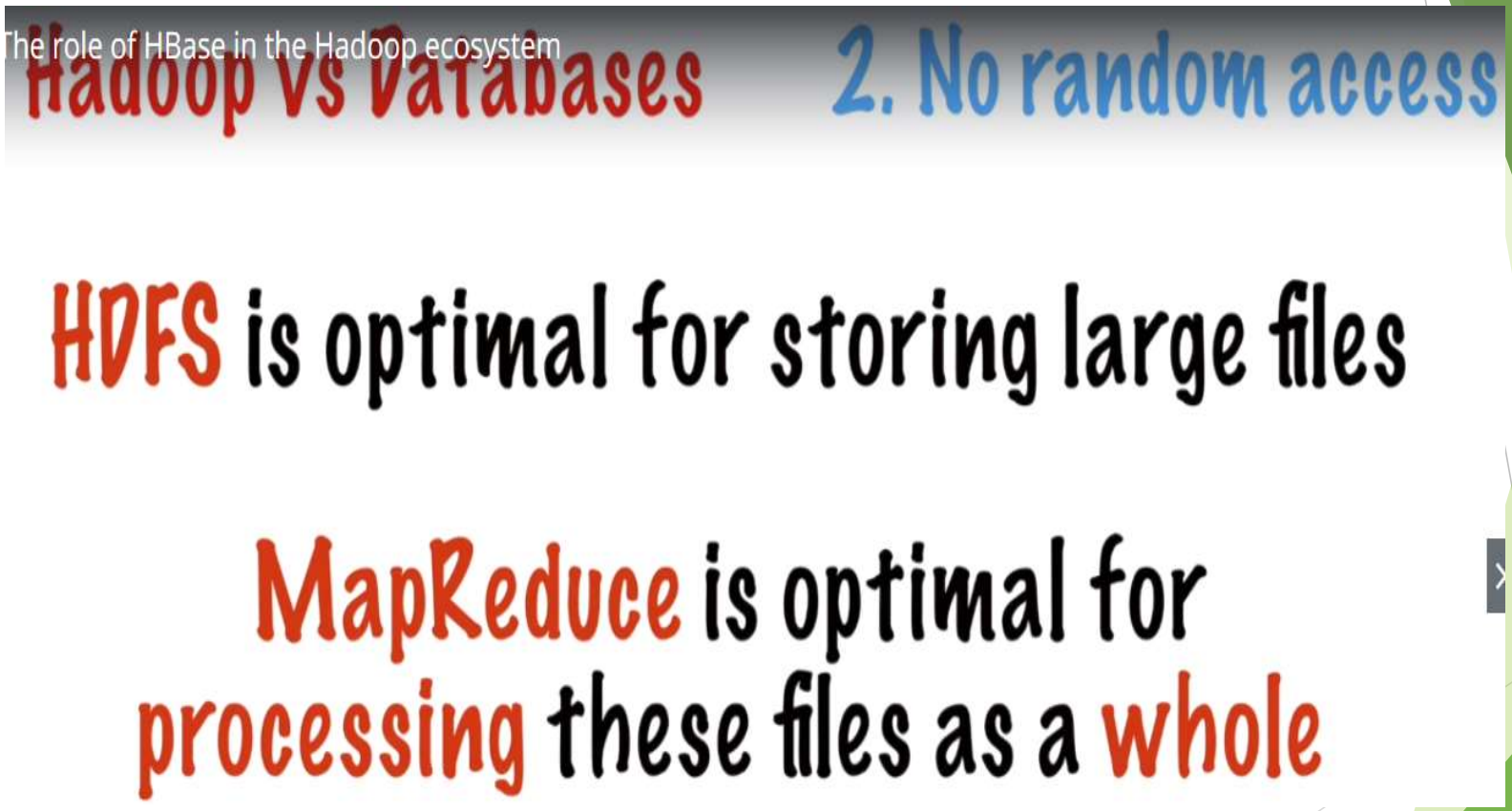
HBase


The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1. Unstructured data
2. No random access

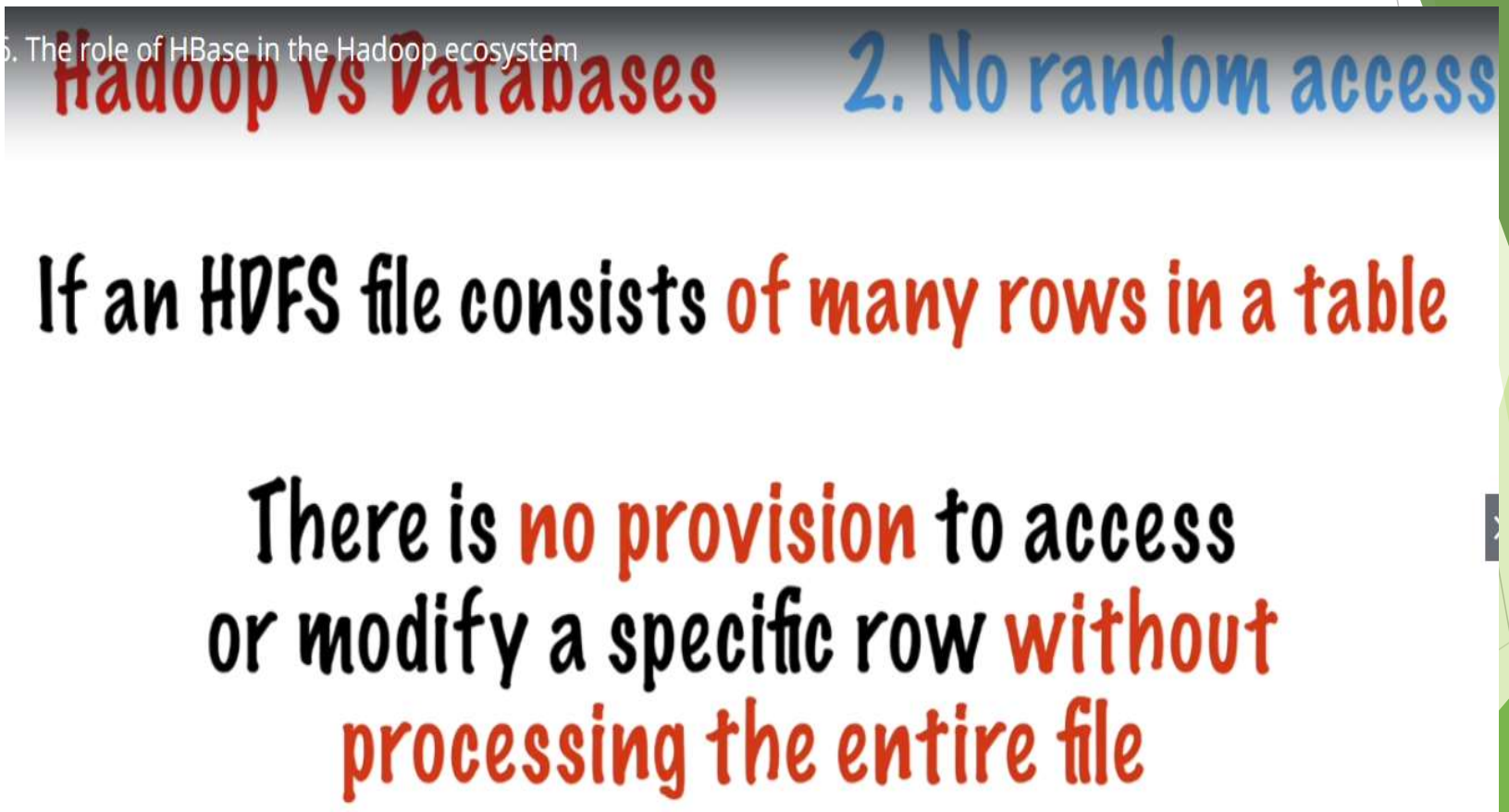
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
2. No random access
Applications that use databases require random access
ie. the ability to create, access and modify individual rows of a table

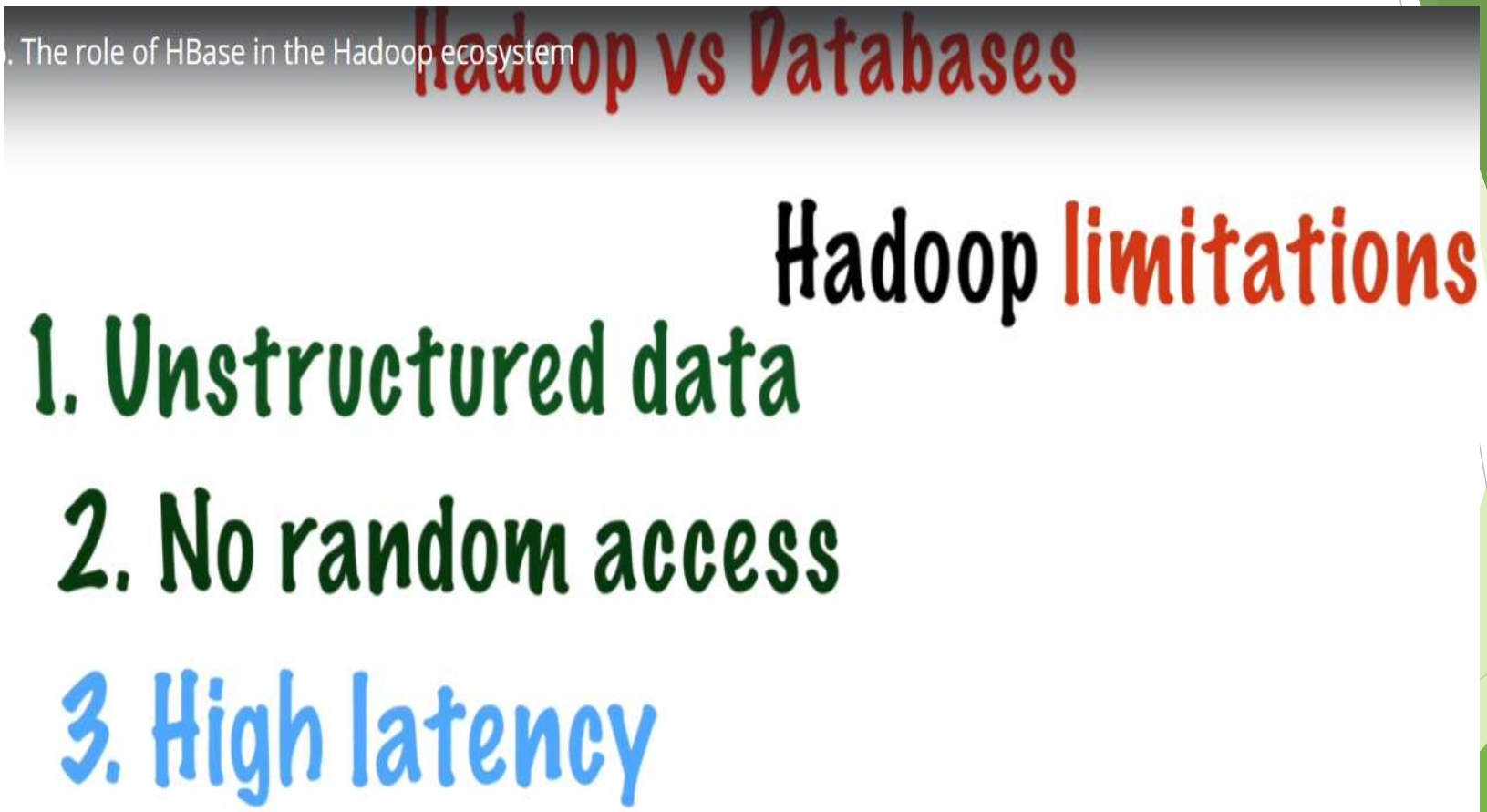
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
2. No random access
HDFS is optimal for storing large files
MapReduce is optimal for processing these files as a whole

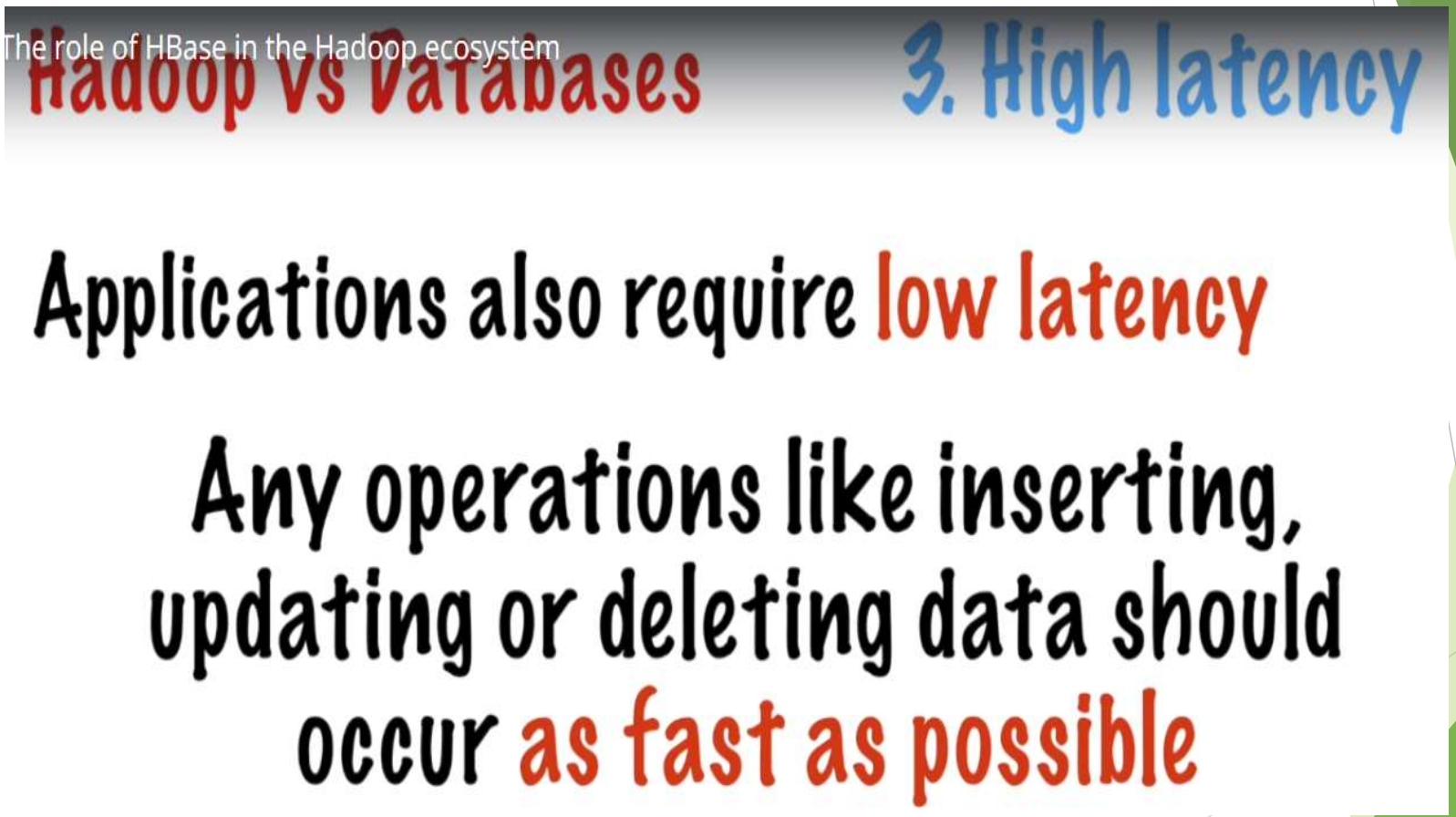
HBase


5. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
2. No random access
If an HDFS file consists of many rows in a table
There is no provision to access
or modify a specific row without
processing the entire file

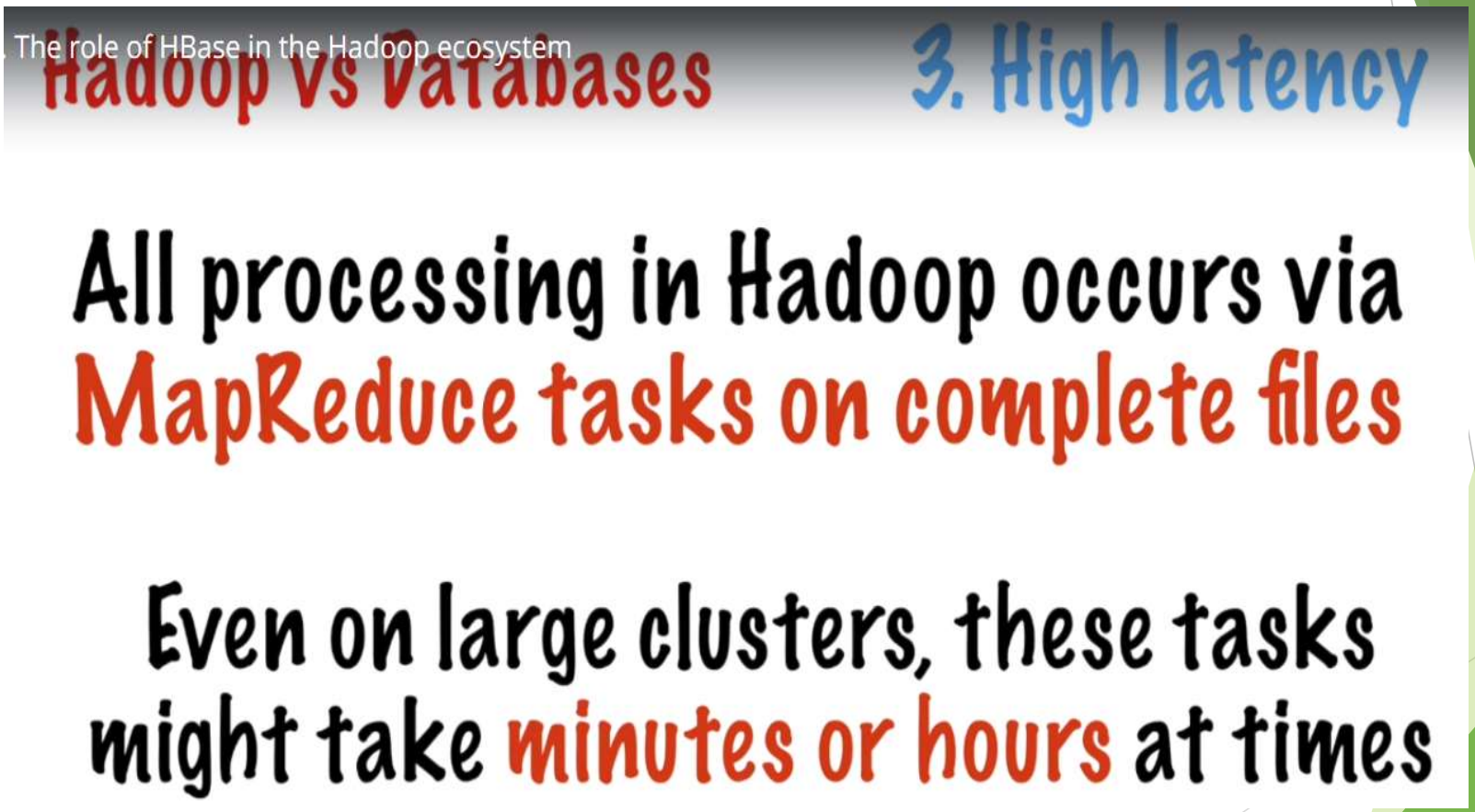
HBase


. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1. Unstructured data
2. No random access
3. High latency

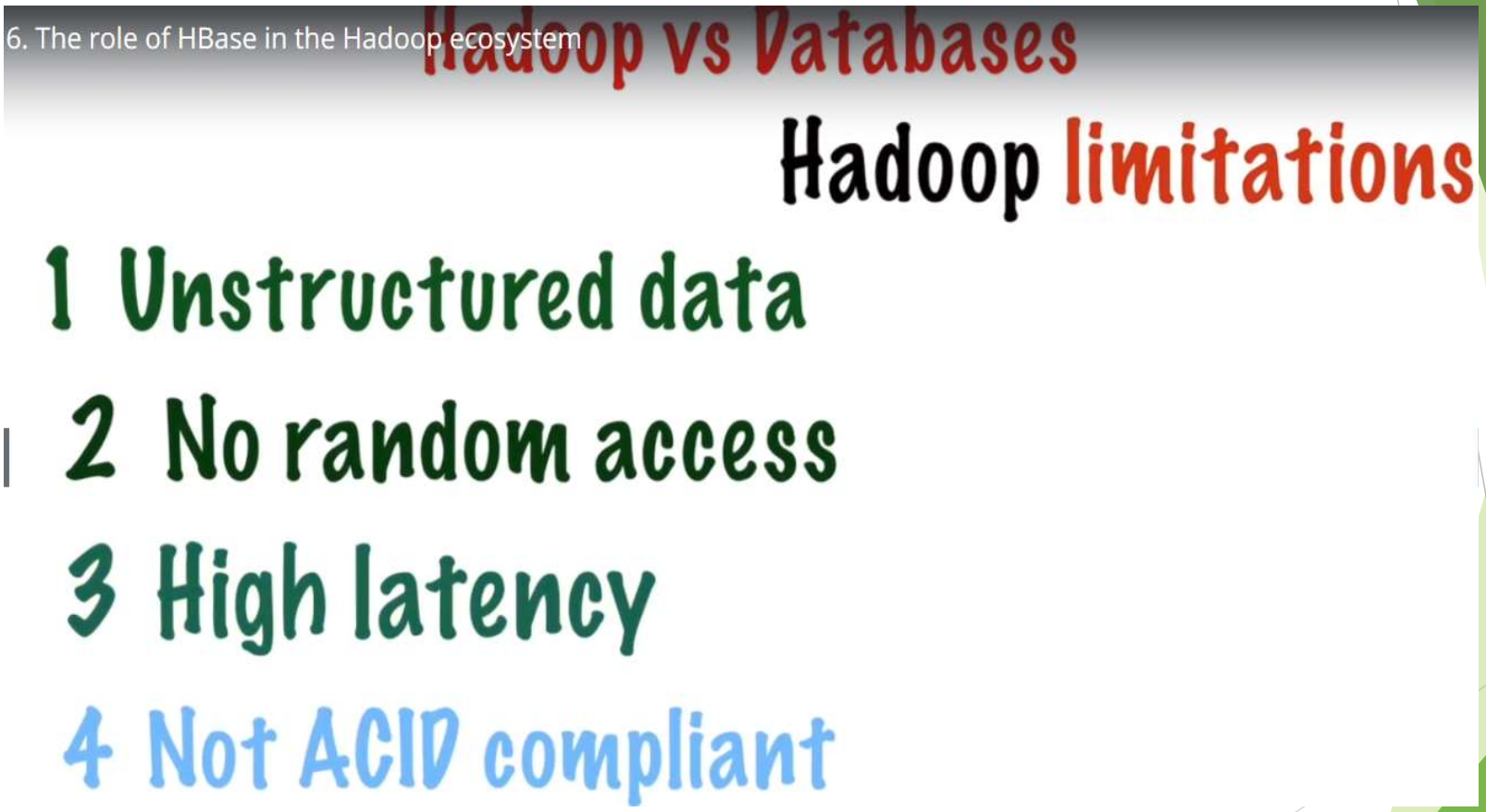
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
3. High latency
Applications also require low latency
Any operations like inserting, updating or deleting data should occur as fast as possible

HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
3. High latency
All processing in Hadoop occurs via
MapReduce tasks on complete files
Even on large clusters, these tasks
might take minutes or hours at times

HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1 Unstructured data
2 No random access
3 High latency
4 Not ACID compliant

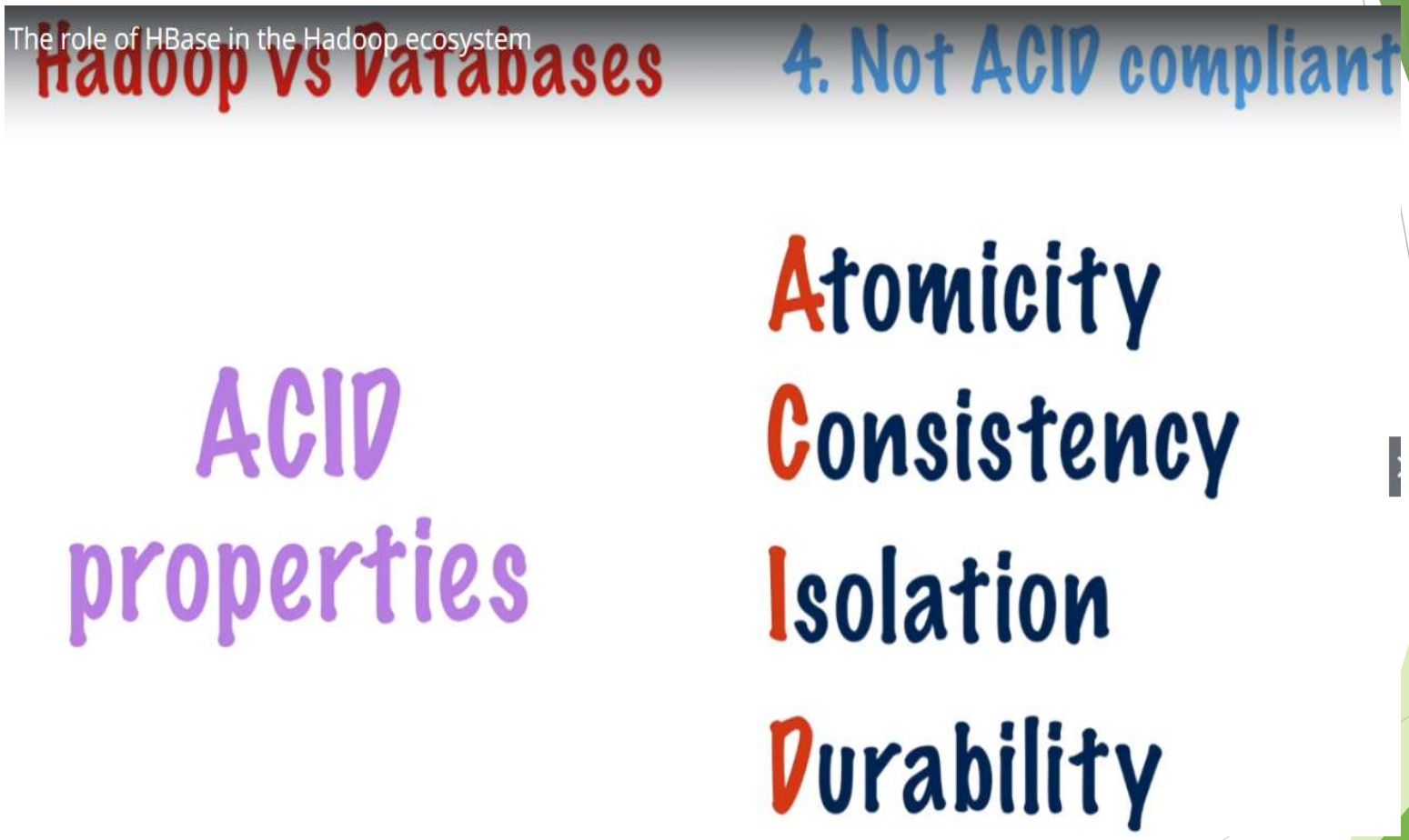
HBase

Databases are the source of truth for the data that they store

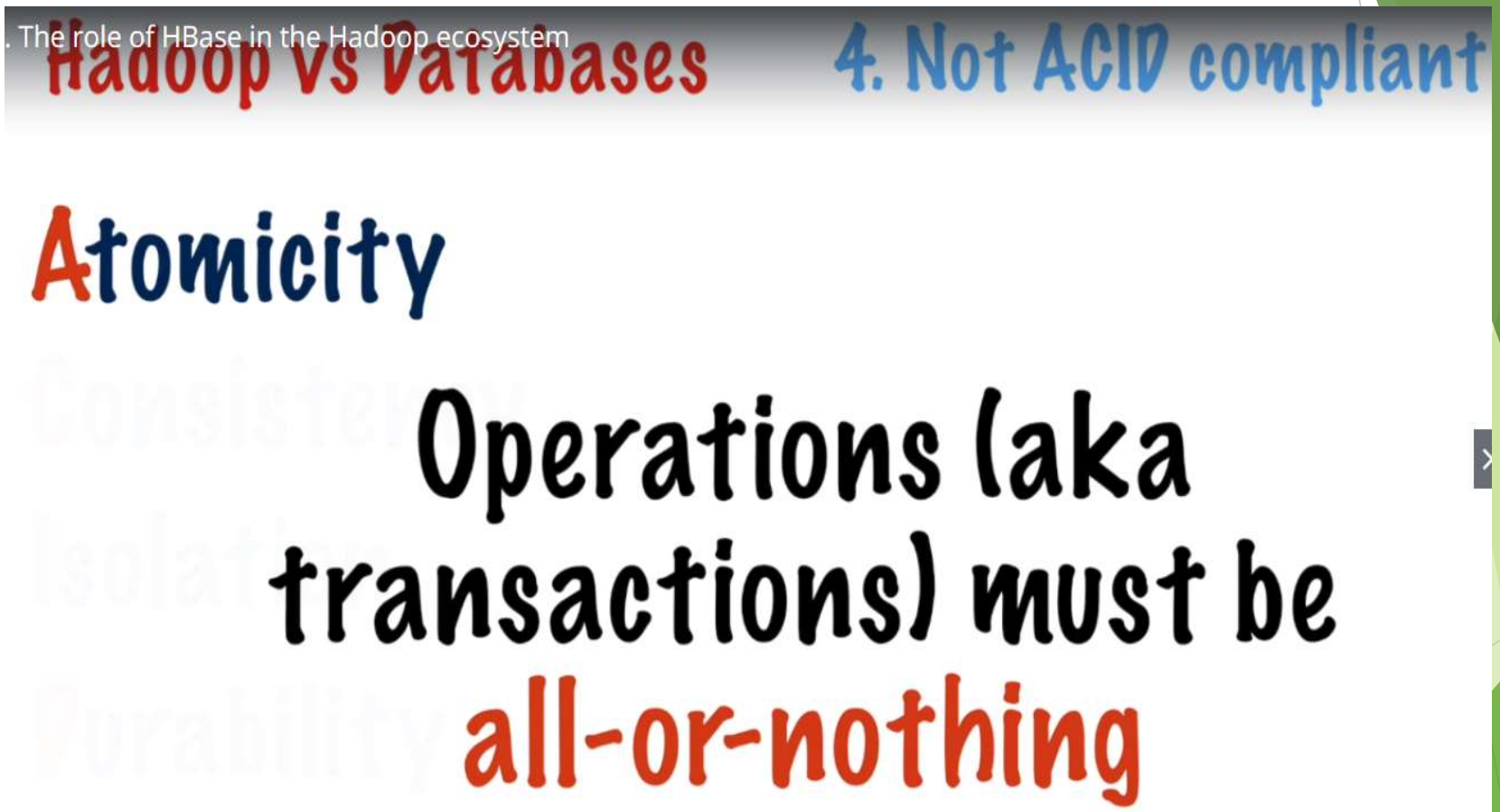
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Databases guarantee ACID properties to maintain the integrity of their data

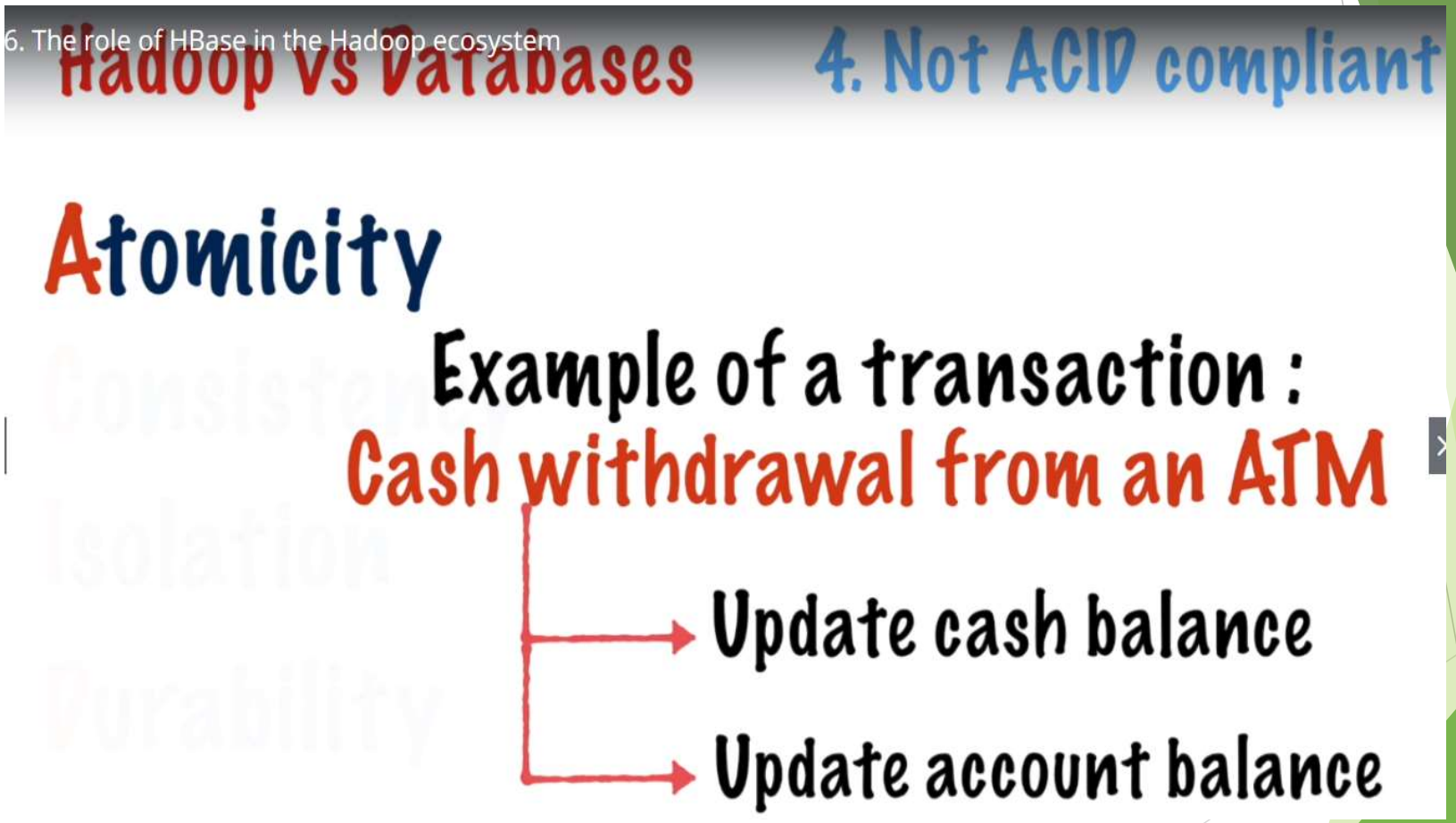
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
ACID properties
Atomicity
Consistency
Isolation
Durability

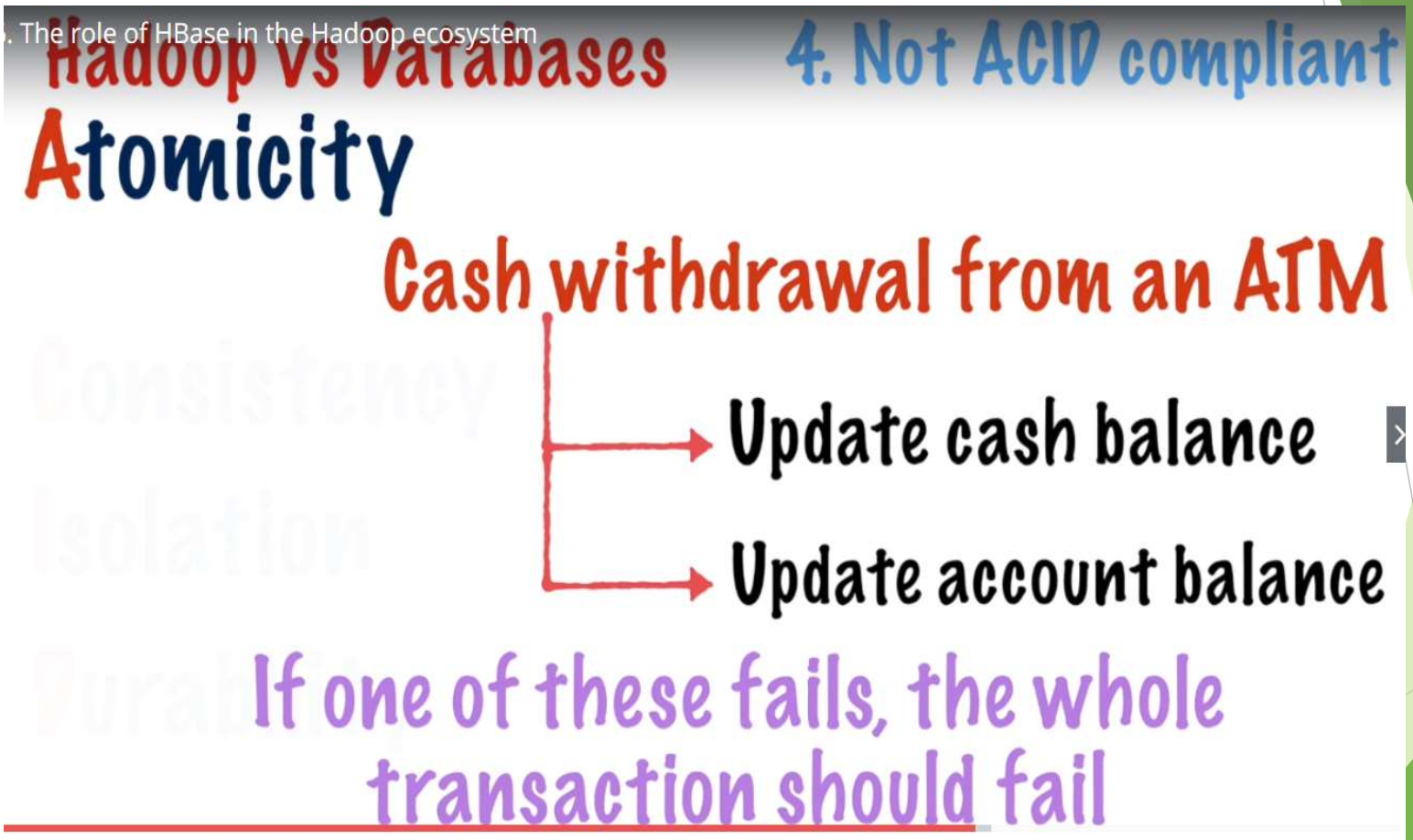
HBase


. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Operations (aka transactions) must be all-or-nothing

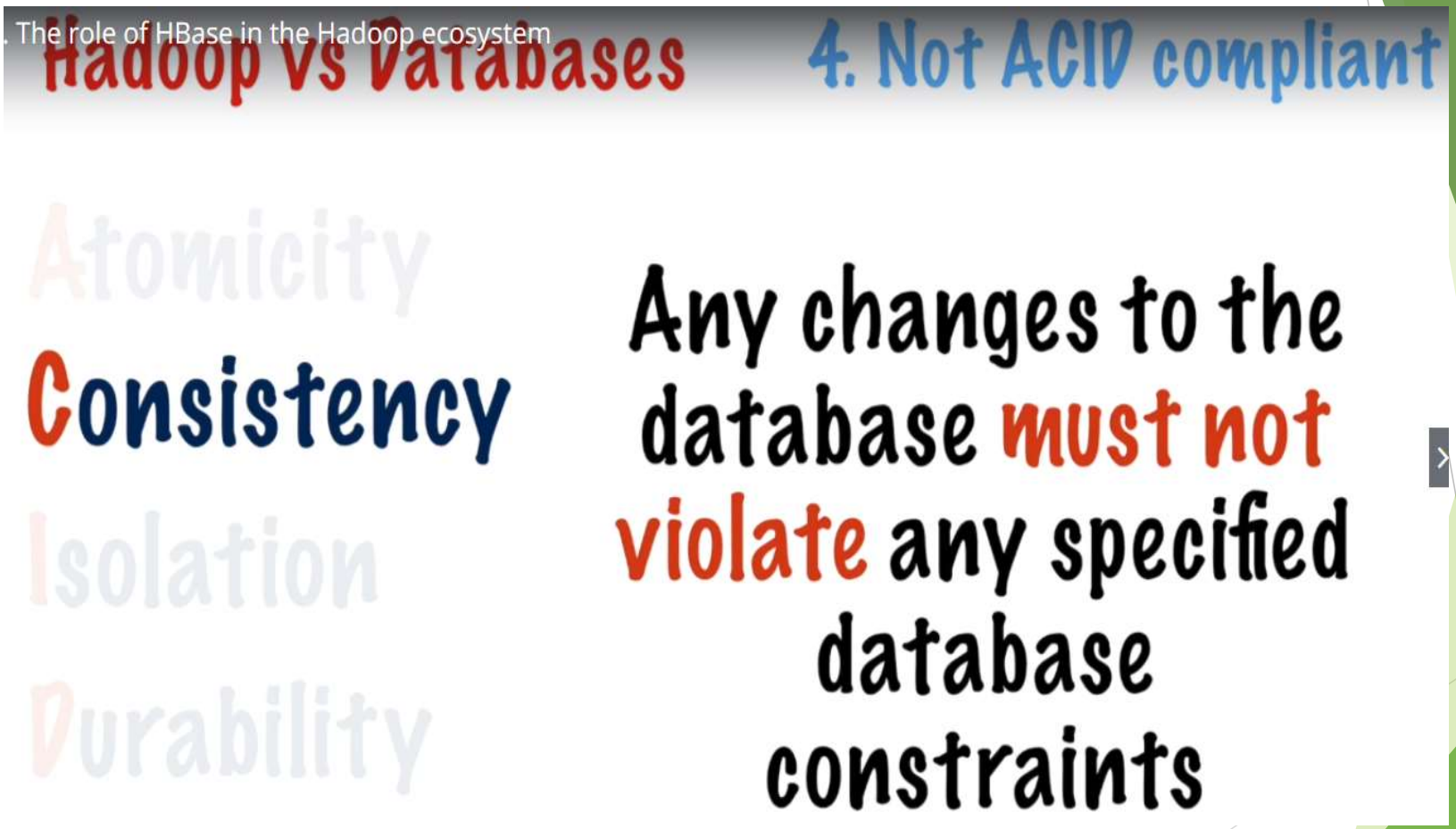
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Example of a transaction :
Cash withdrawal from an ATM
Update cash balance
Update account balance

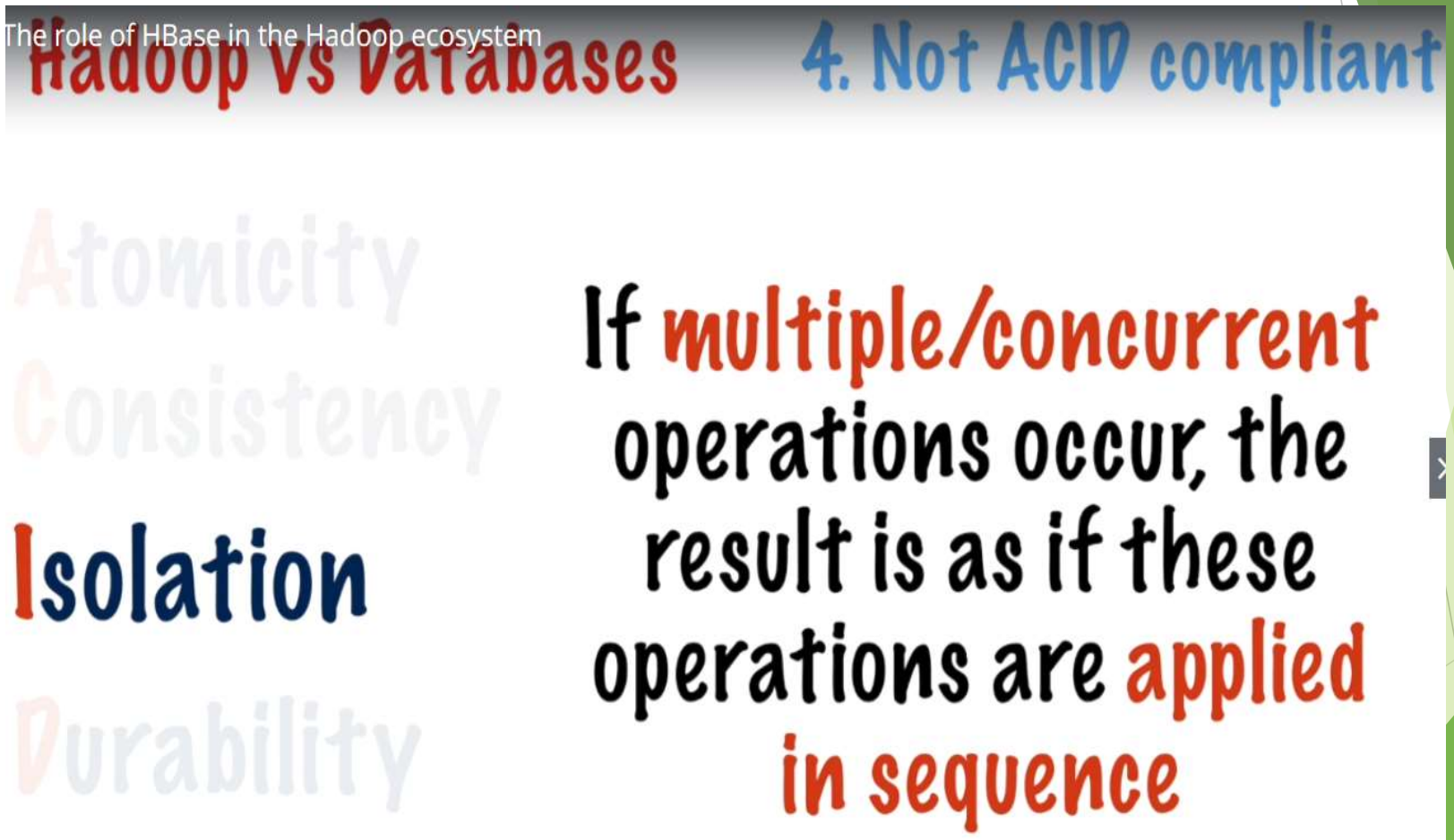
HBase


. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Cash withdrawal from an ATM
Update cash balance
Update account balance
If one of these fails, the whole transaction should fail

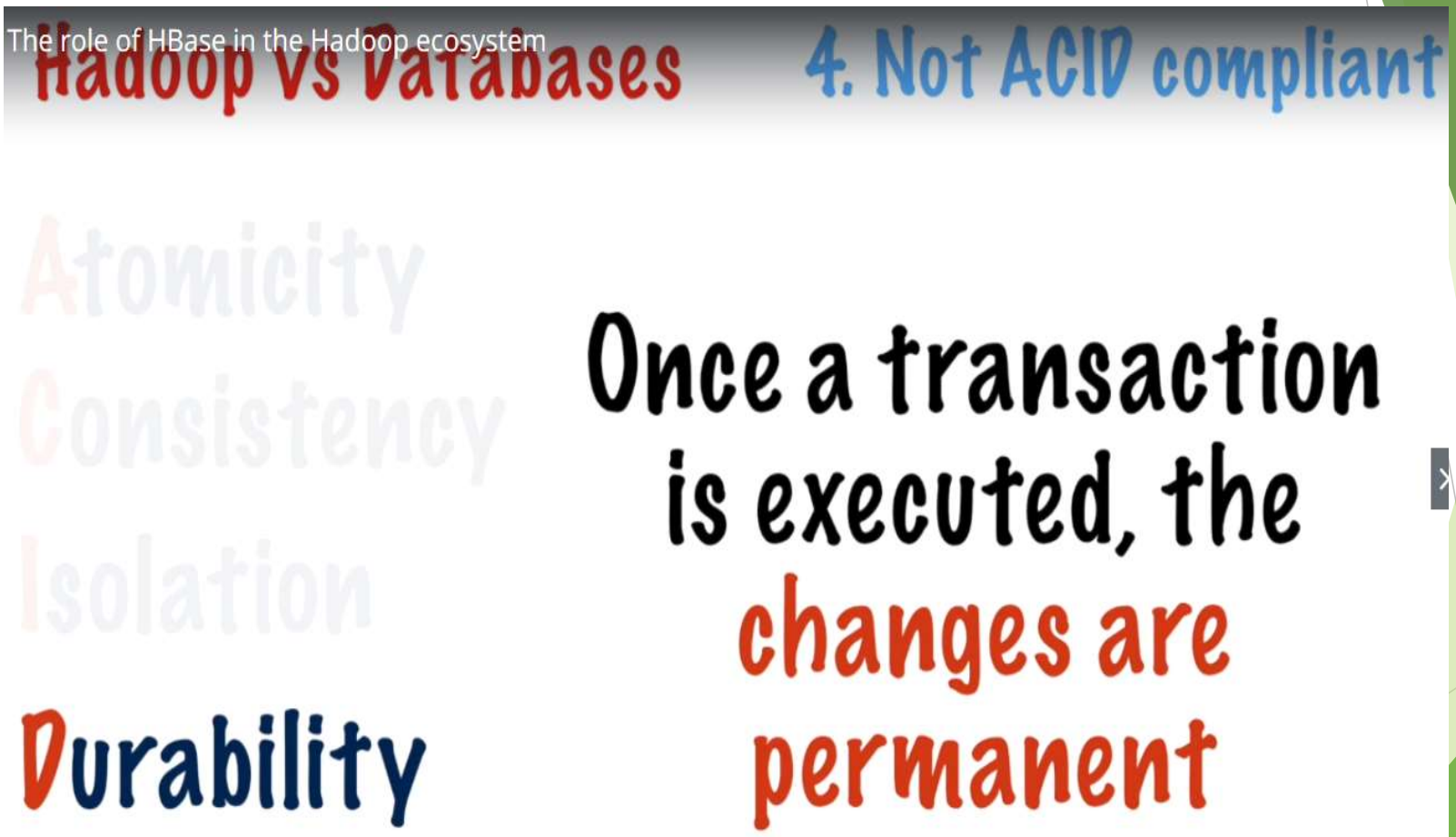
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Consistency
Isolation
Durability
Any changes to the database must not violate any specified database constraints

HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Consistency
Isolation
Durability
If multiple/concurrent operations occur, the result is as if these operations are applied in sequence

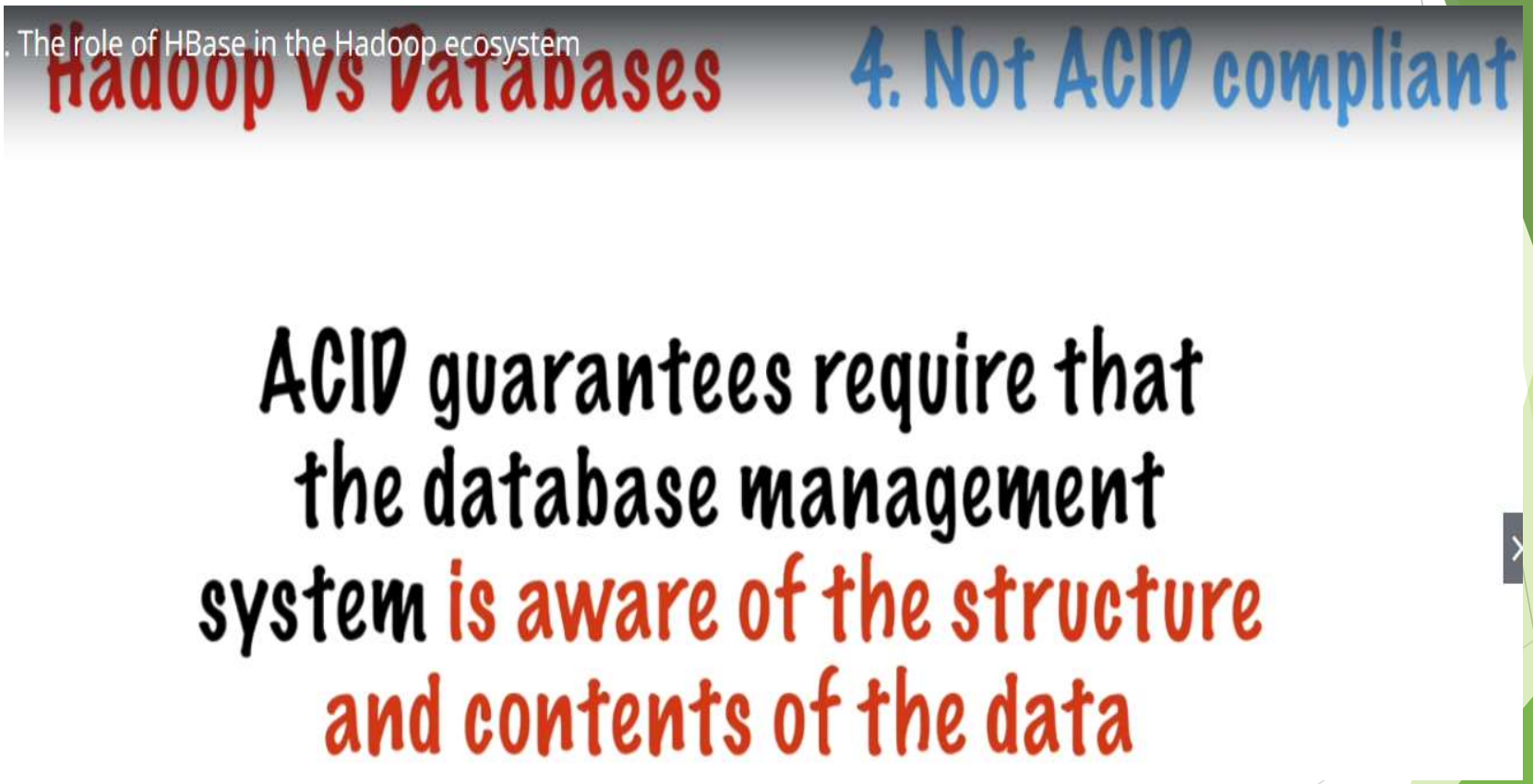
HBase


The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Consistency
Isolation
Durability
Once a transaction is executed, the changes are permanent

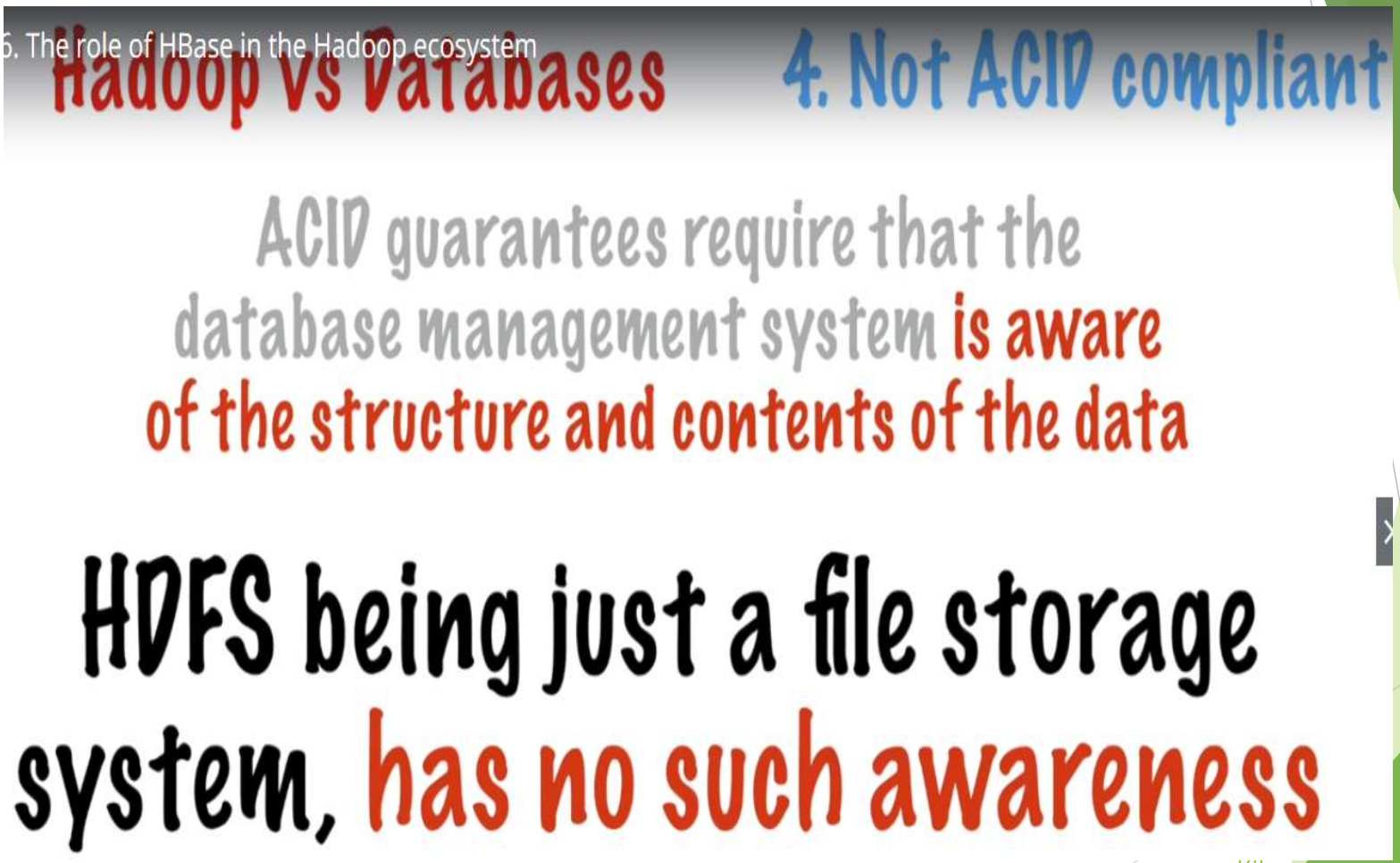
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
Atomicity
Consistency
Isolation
Durability
Traditional databases are designed to guarantee all of these properties

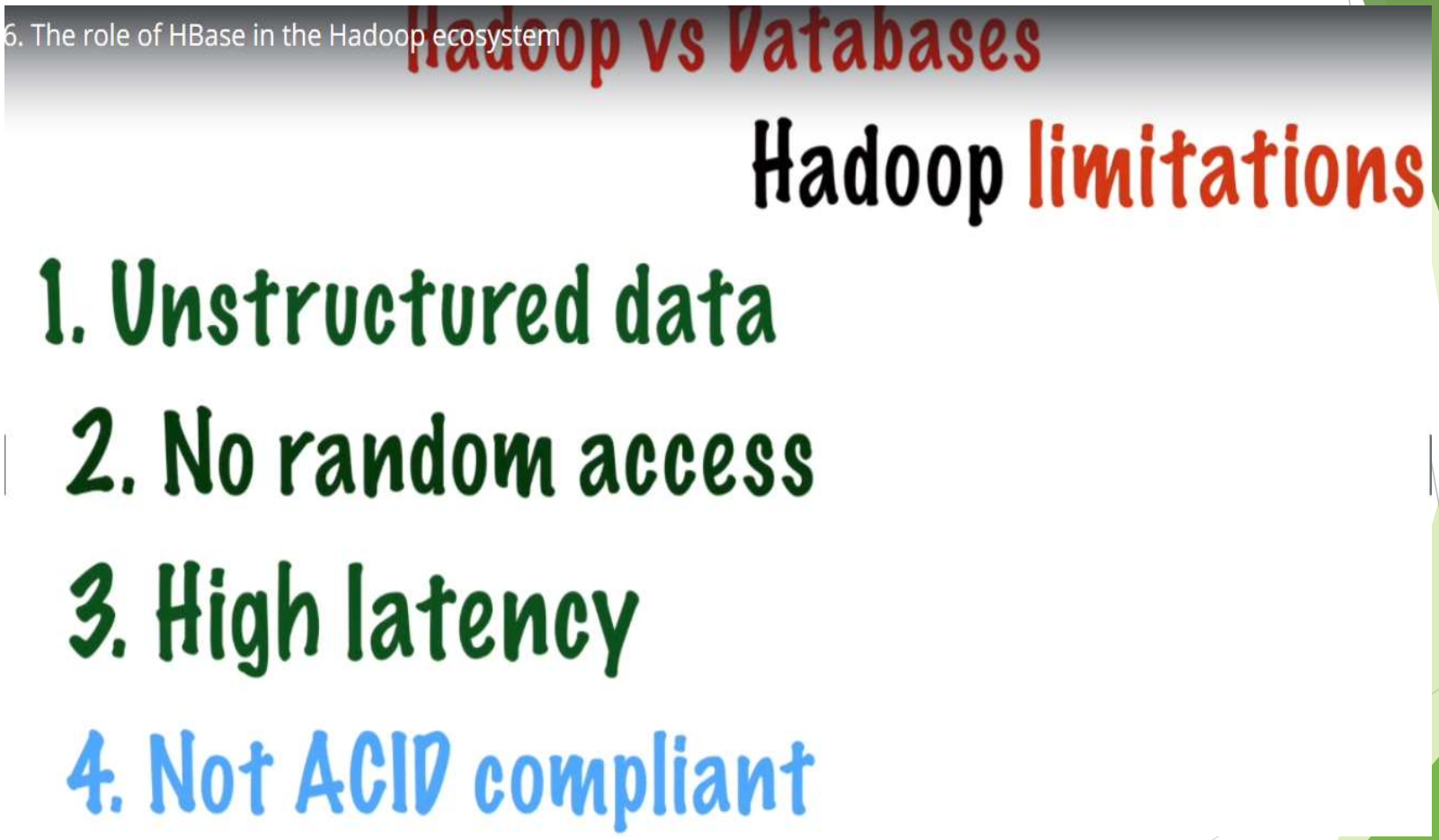
HBase


. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
ACID guarantees require that the database management system is aware of the structure and contents of the data

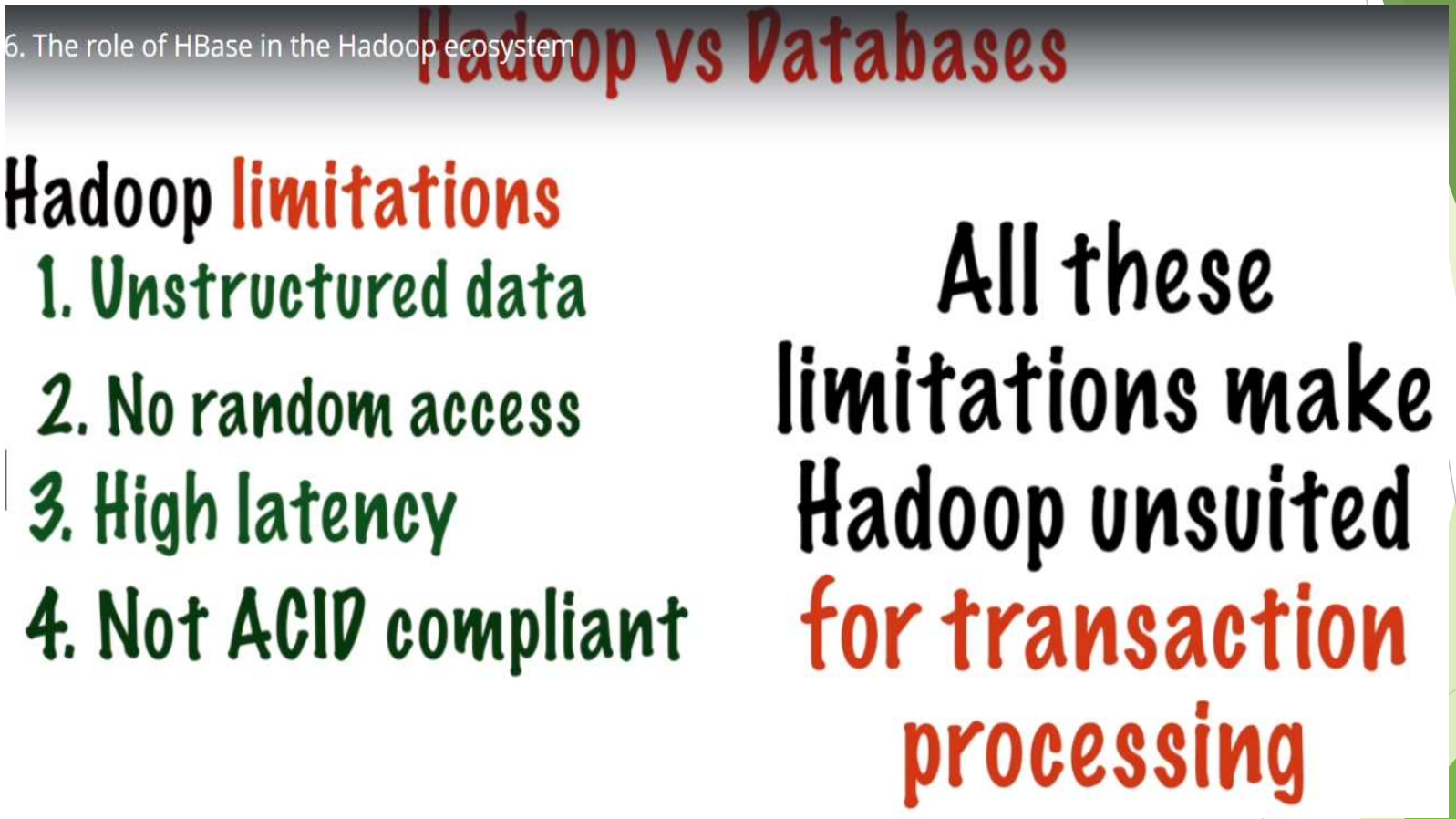
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop Vs Databases
4. Not ACID compliant
ACID guarantees require that the database management system is aware of the structure and contents of the data
HDFS being just a file storage system, has no such awareness

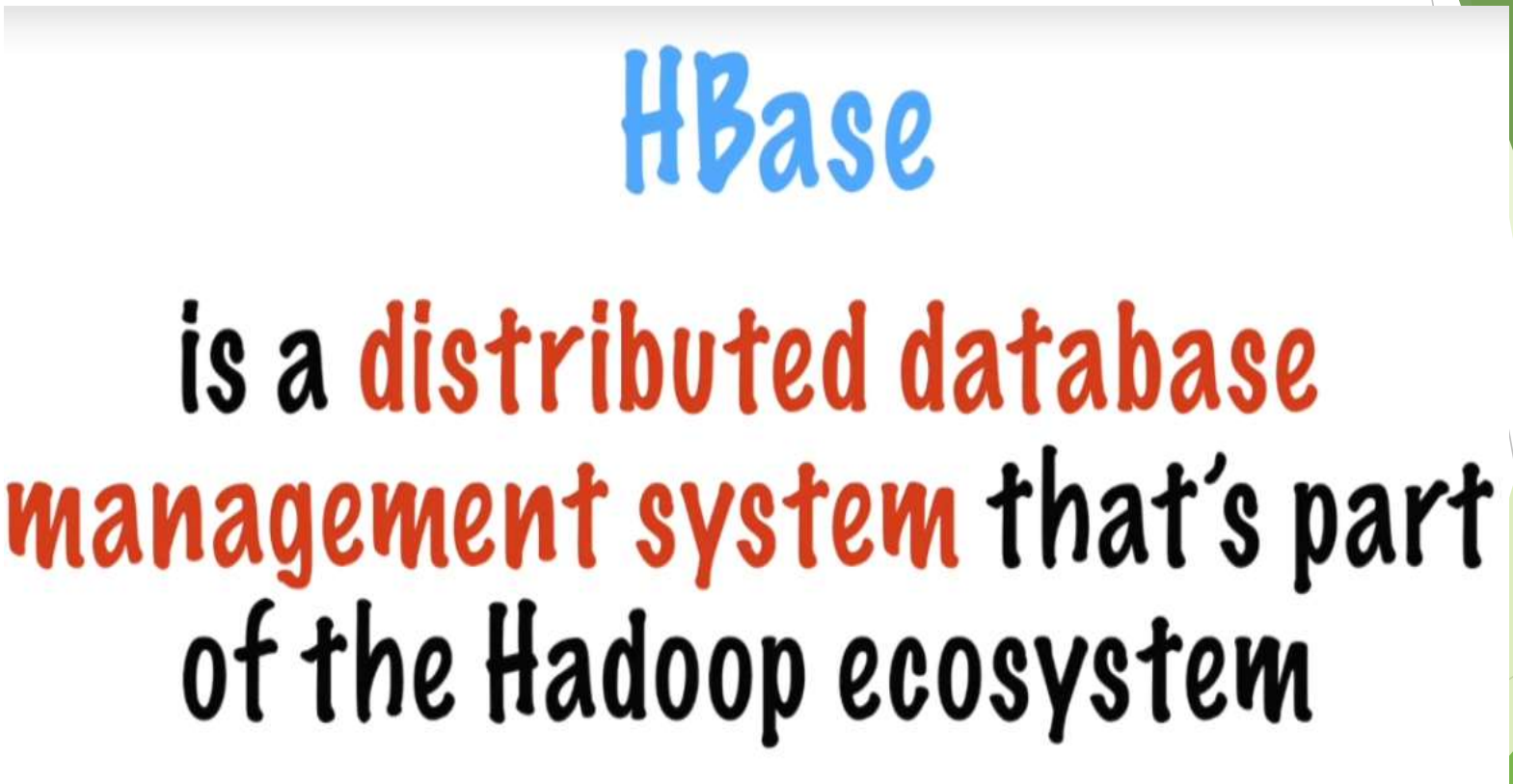
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1. Unstructured data
2. No random access
3. High latency
4. Not ACID compliant

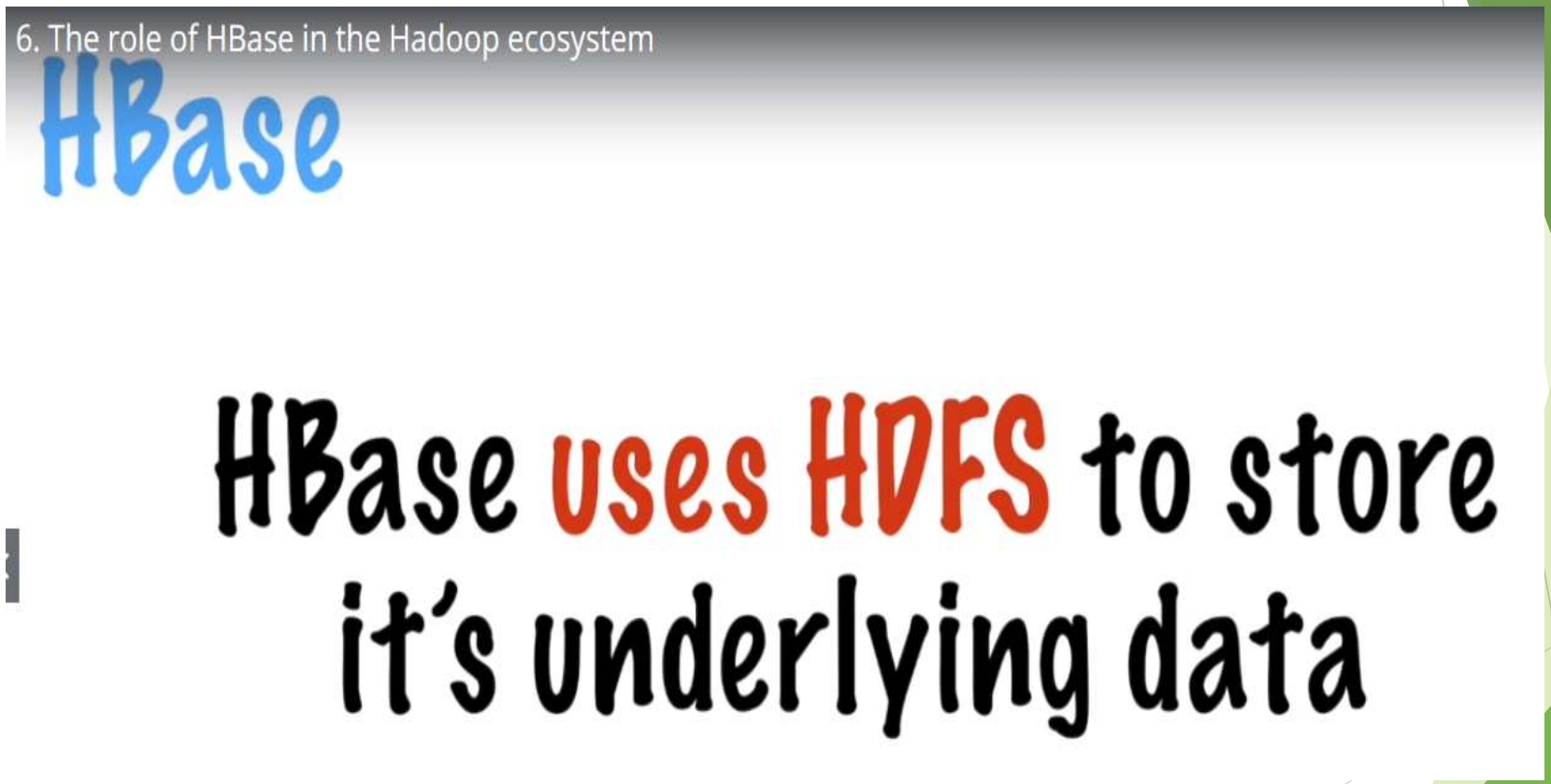
HBase


6. The role of HBase in the Hadoop ecosystem
Hadoop vs Databases
Hadoop limitations
1. Unstructured data
2. No random access
3. High latency
4. Not ACID compliant
All these limitations make Hadoop unsuited for transaction processing

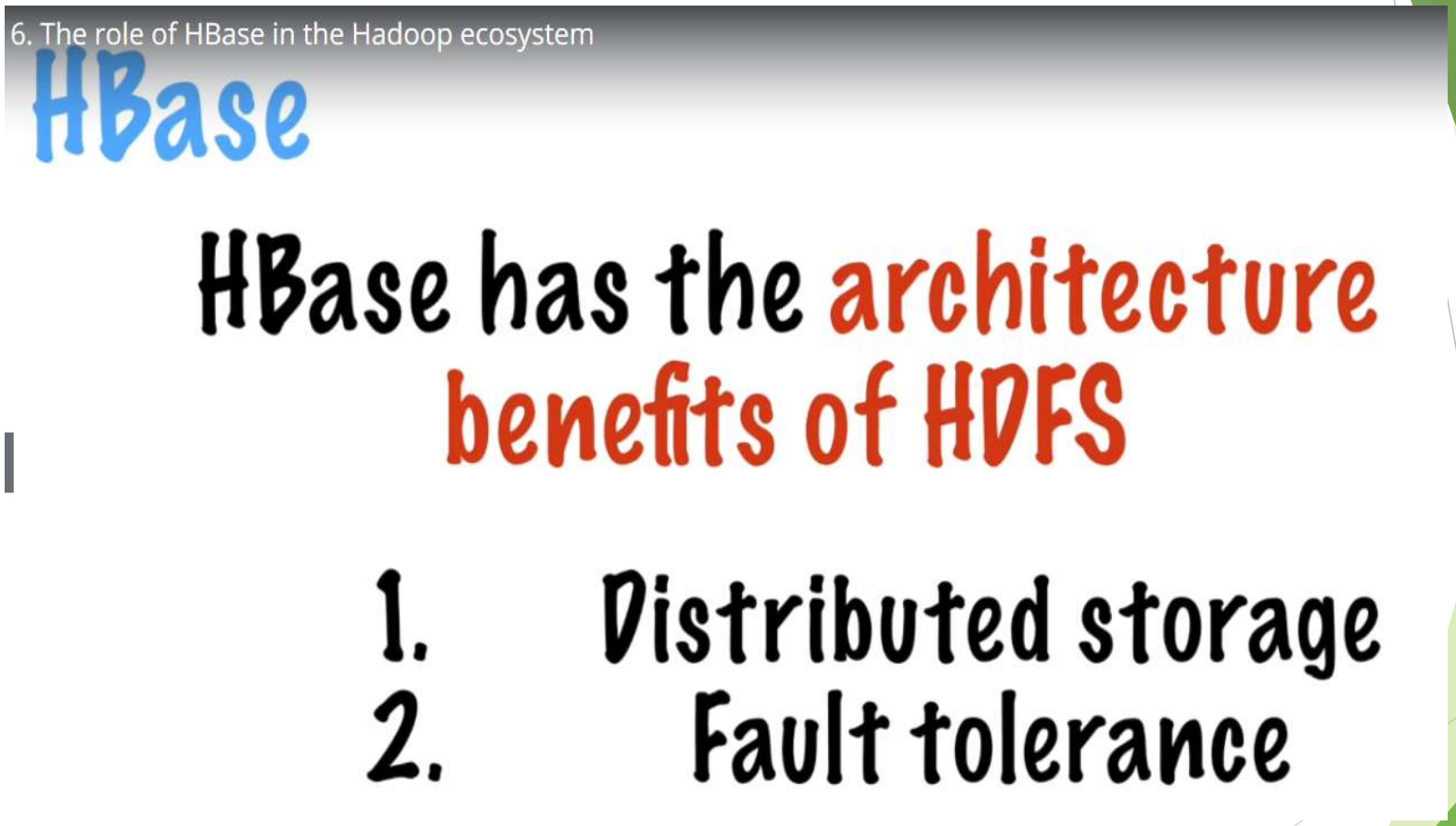
HBase


HBase
is a distributed database
management system that's part
of the Hadoop ecosystem

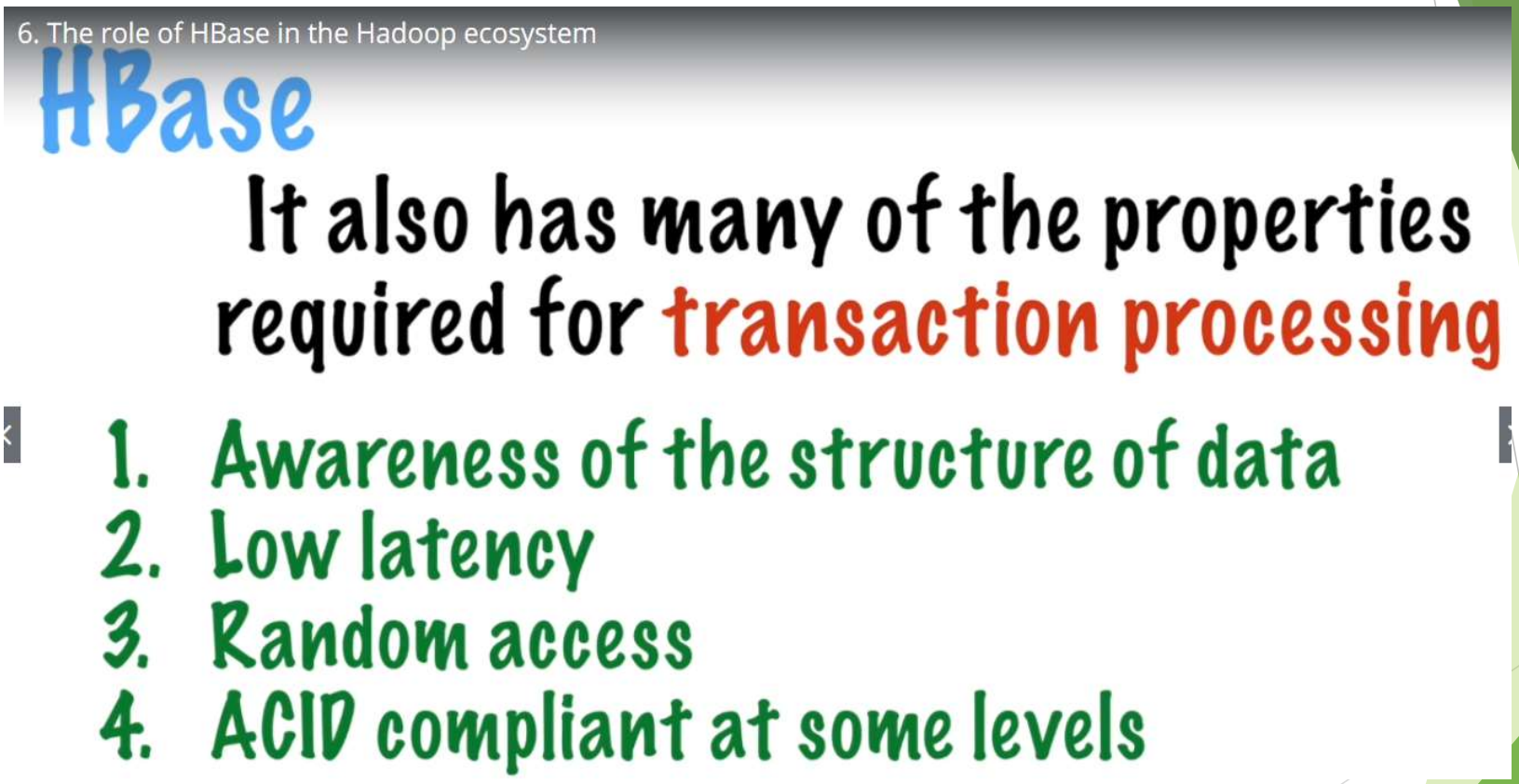
HBase


6. The role of HBase in the Hadoop ecosystem
HBase
HBase uses HDFS to store
it's underlying data

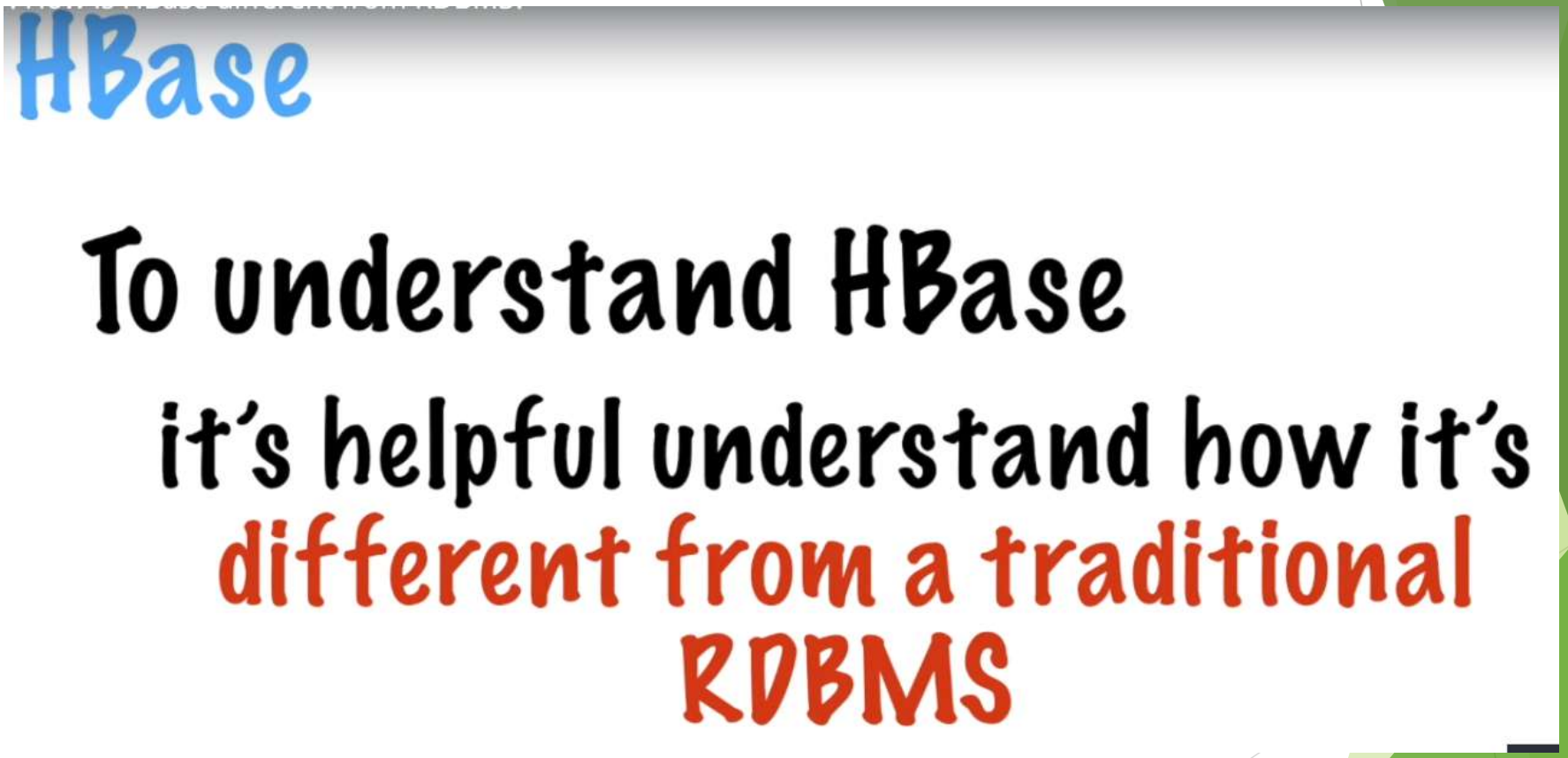
HBase


6. The role of HBase in the Hadoop ecosystem
HBase
HBase has the architecture benefits of HDFS
1. Distributed storage
2. Fault tolerance

HBase


6. The role of HBase in the Hadoop ecosystem
HBase
It also has many of the properties required for transaction processing
1. Awareness of the structure of data
2. Low latency
3. Random access
4. ACID compliant at some levels

HBase


HBase
To understand HBase
it's helpful understand how it's
different from a traditional
RDBMS

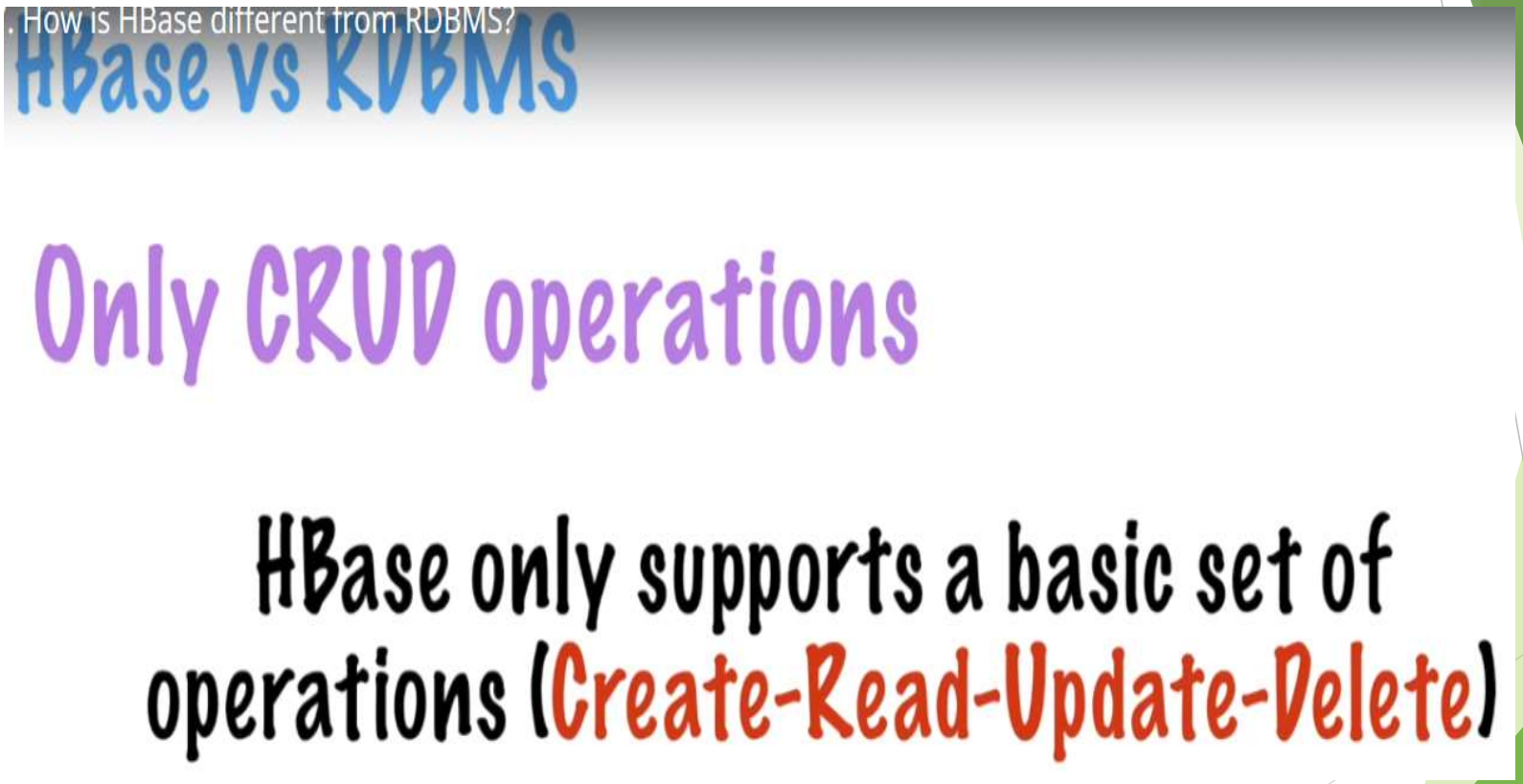
HBase


How is HBase different from RDBMS?
HBase vs RDBMS
NoSQL
HBase does not support SQL

HBase


How is HBase different from RDBMS?
HBase vs RDBMS
In a traditional RDBMS, all operations like creating, inserting, updating rows are done using SQL
HBase does not support SQL

HBase


. How is HBase different from RDBMS?
HBase vs RDBMS
Only CRUD operations
HBase only supports a basic set of operations (Create-Read-Update-Delete)

HBase


. How is HBase different from RDBMS?
HBase vs RDBMS
(Create-Read-Update-Delete)
Only CRUD operations
All these operations have to be applied at a row level

HBase

. How is HBase different from RDBMS?

HBase

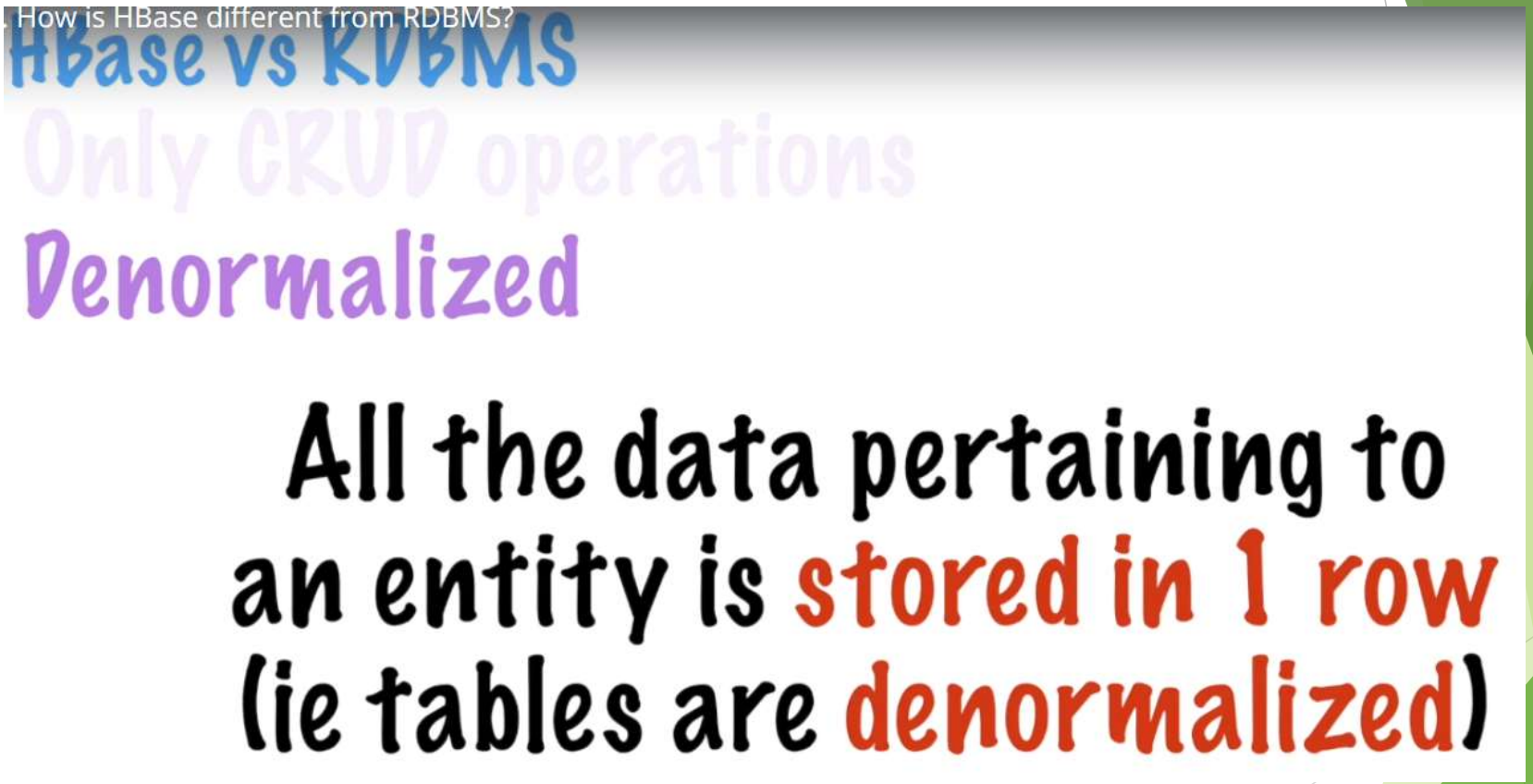

. How is HBase different from RDBMS?
HBase vs RDBMS
(Create-Read-Update-Delete)
Only CRUD operations
This means that you cannot perform operations like
Joins, Group by etc

HBase


7. How is HBase different from RDBMS?
HBase vs RDBMS
Only CRUD operations
Denormalized
HBase tables are not designed
using a relational data model

HBase

How is HBase different from RDBMS?

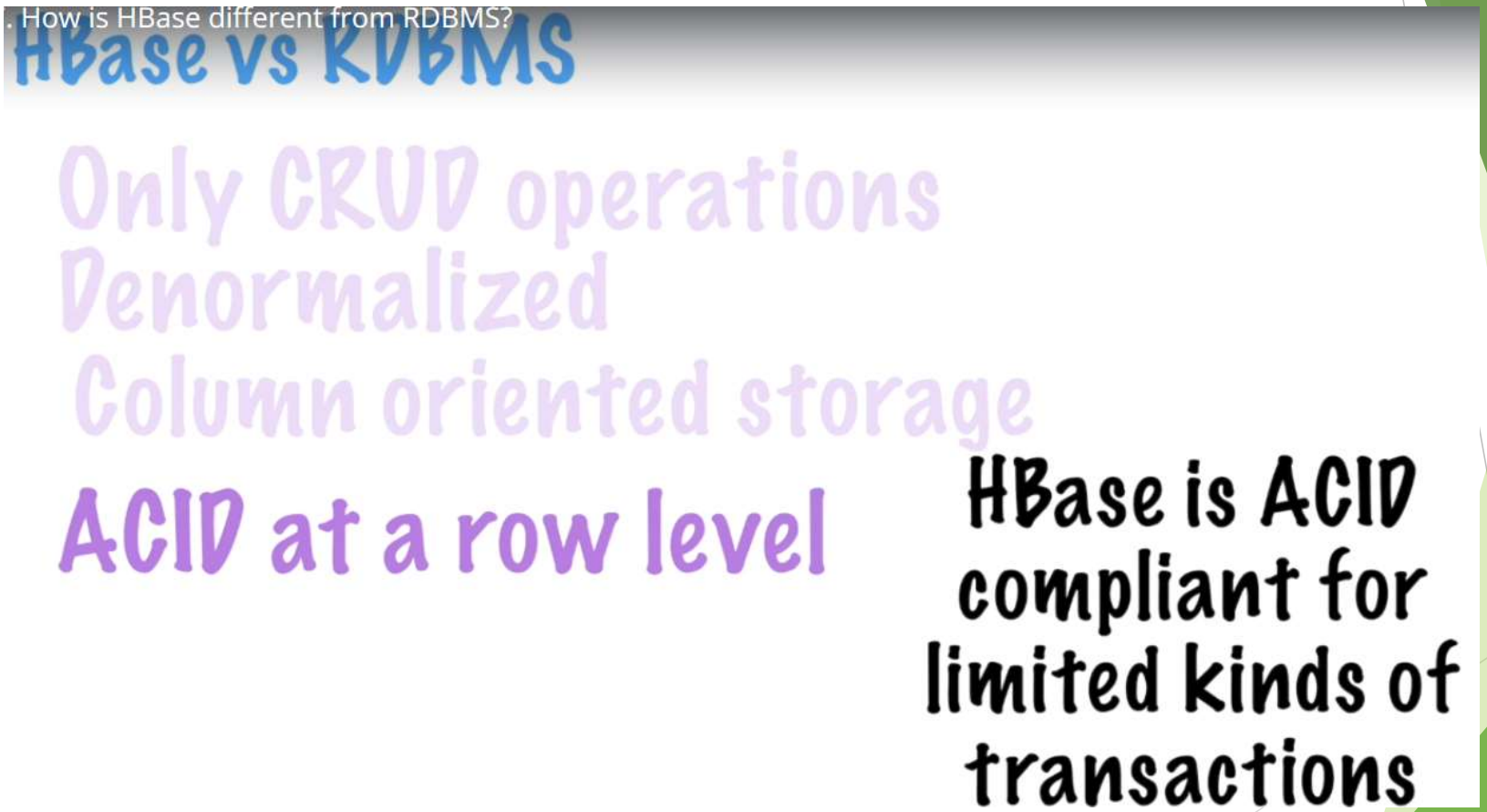


HBase

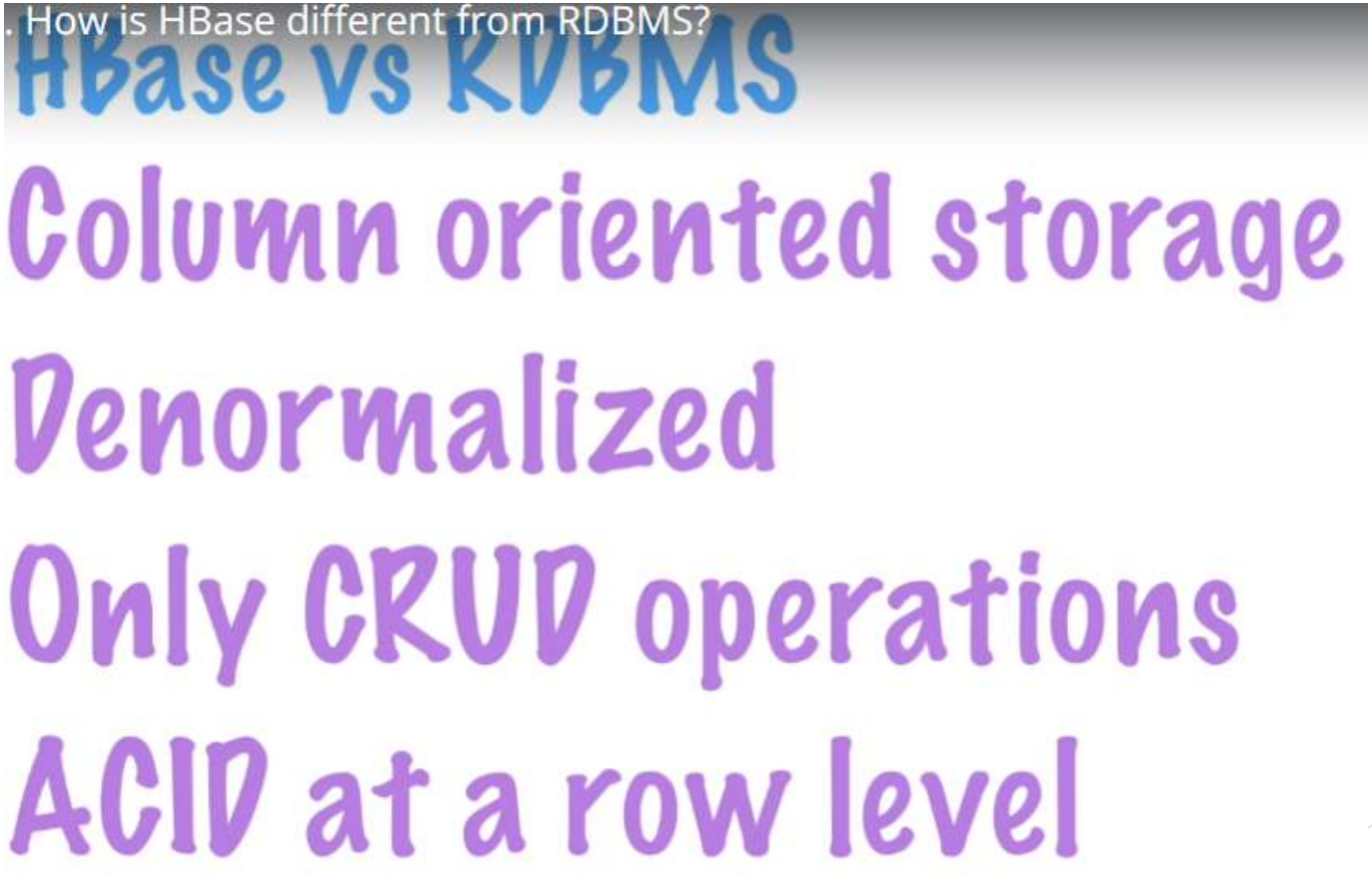
7. How is HBase different from RDBMS?

HBase

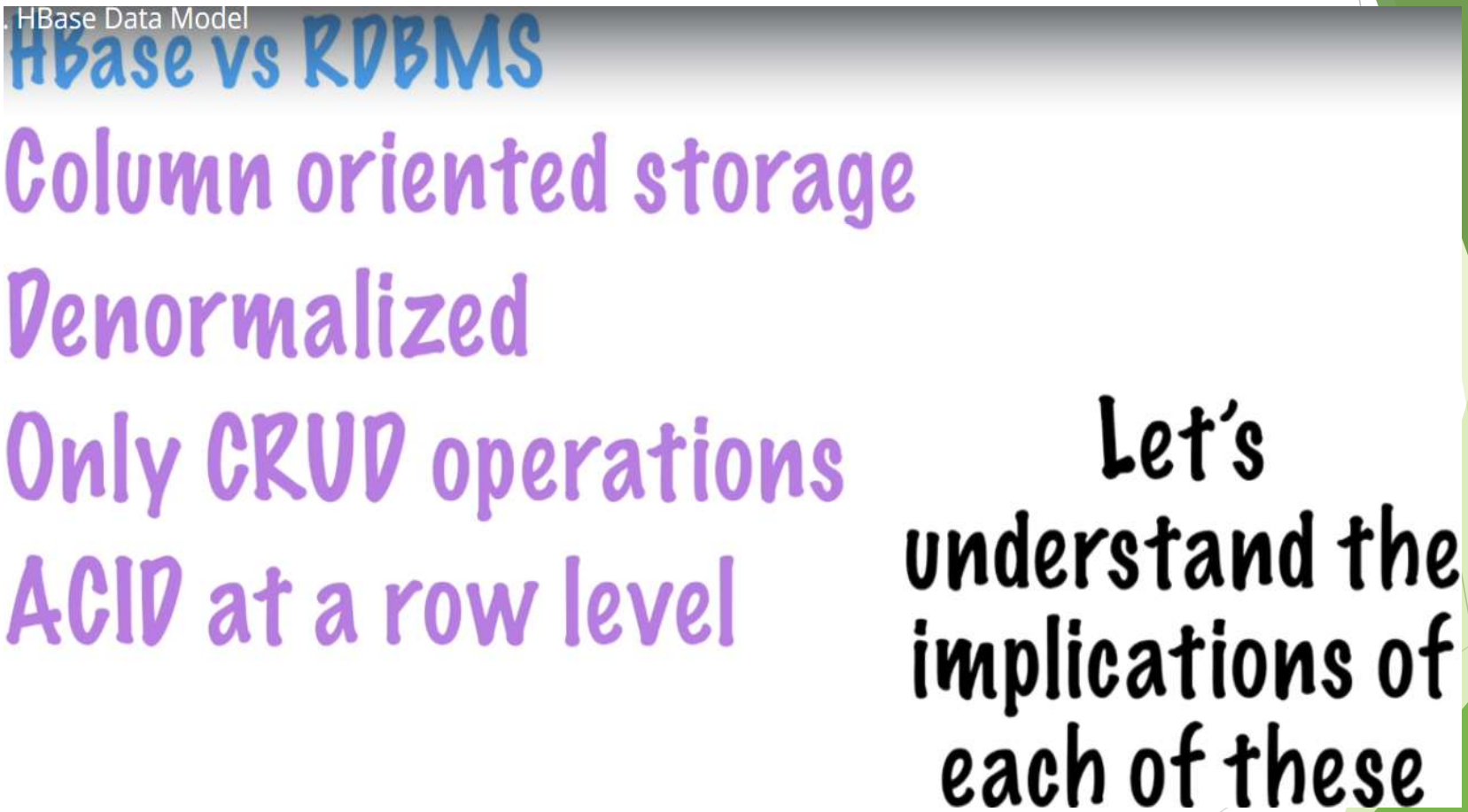
. How is HBase different from RDBMS?



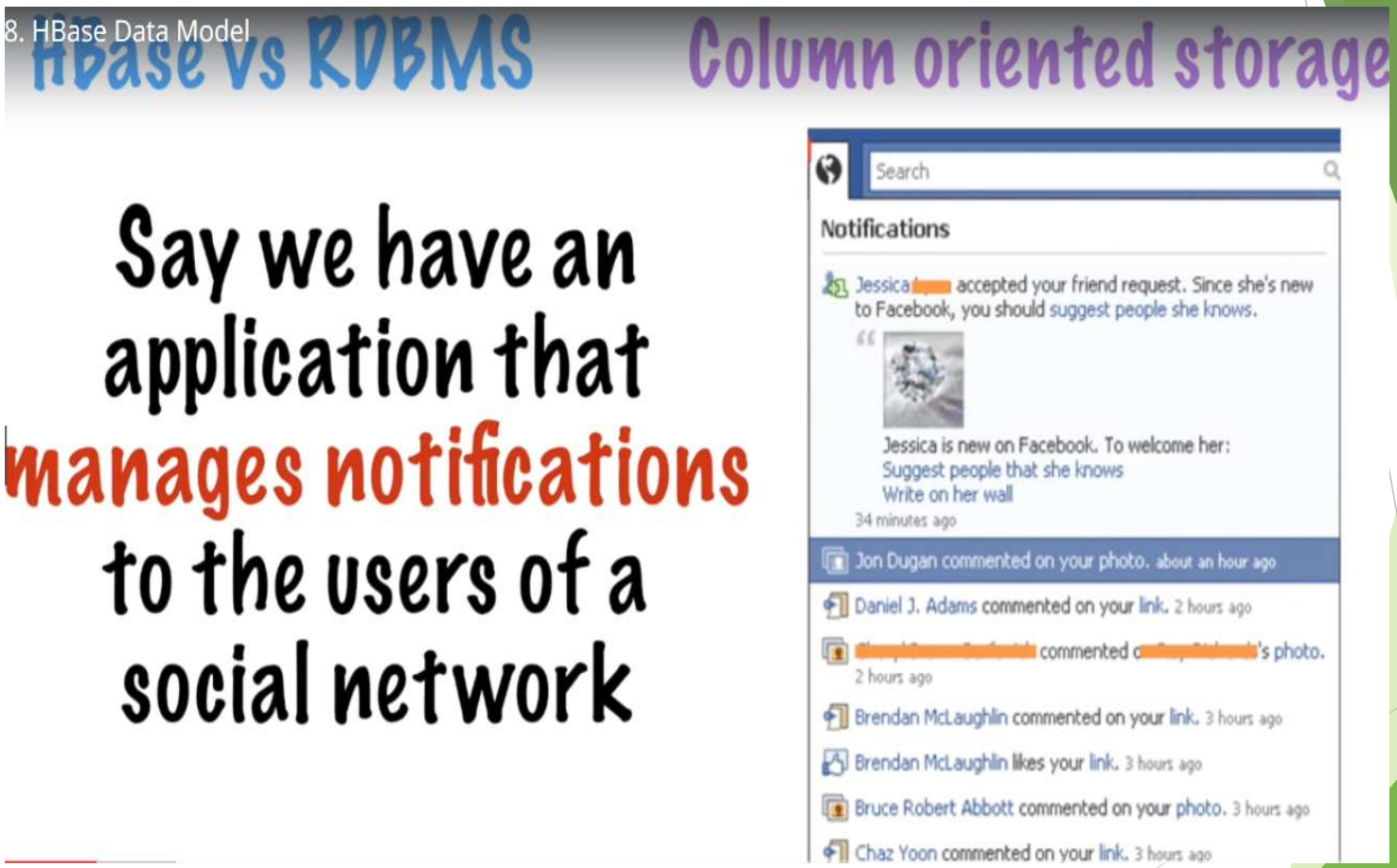
HBase


How is HBase different from RDBMS?
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level

HBase


HBase Data Model
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level
Let's understand the implications of each of these

HBase


8. HBase Data Model
HBase Vs RDBMS
Column oriented storage
Say we have an application that manages notifications to the users of a social network
Search
Notifications
Jessica accepted your friend request. Since she's new to Facebook, you should suggest people she knows.
Jessica is new on Facebook. To welcome her:
Suggest people that she knows
Write on her wall
34 minutes ago
Jon Dugan commented on your photo. about an hour ago
Daniel J. Adams commented on your link. 2 hours ago
commented on 's photo.
2 hours ago
Brendan McLaughlin commented on your link. 3 hours ago
Brendan McLaughlin likes your link. 3 hours ago
Bruce Robert Abbott commented on your photo. 3 hours ago
Chaz Yoon commented on your link. 3 hours ago

HBase Vs RDBMS

Column oriented storage

Here is a table that stores some notification related data

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

HBase Vs RDBMS

Column oriented storage

This is how data is stored in traditional databases

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

HBase Vs RDBMS

Column oriented storage

A table with a fixed schema is defined

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

HBase Vs RDBMS

Column oriented storage

Each row represents a data point

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

HBase Vs RDBMS

Column oriented storage

In a column oriented store, each cell represents a datapoint

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

HBase

. HBase Data Model

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Data is stored in a map

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Data is stored in a map

Key = <Row id, Col id>

Value = <data>

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Data is stored in a map

Key = 2, for_user

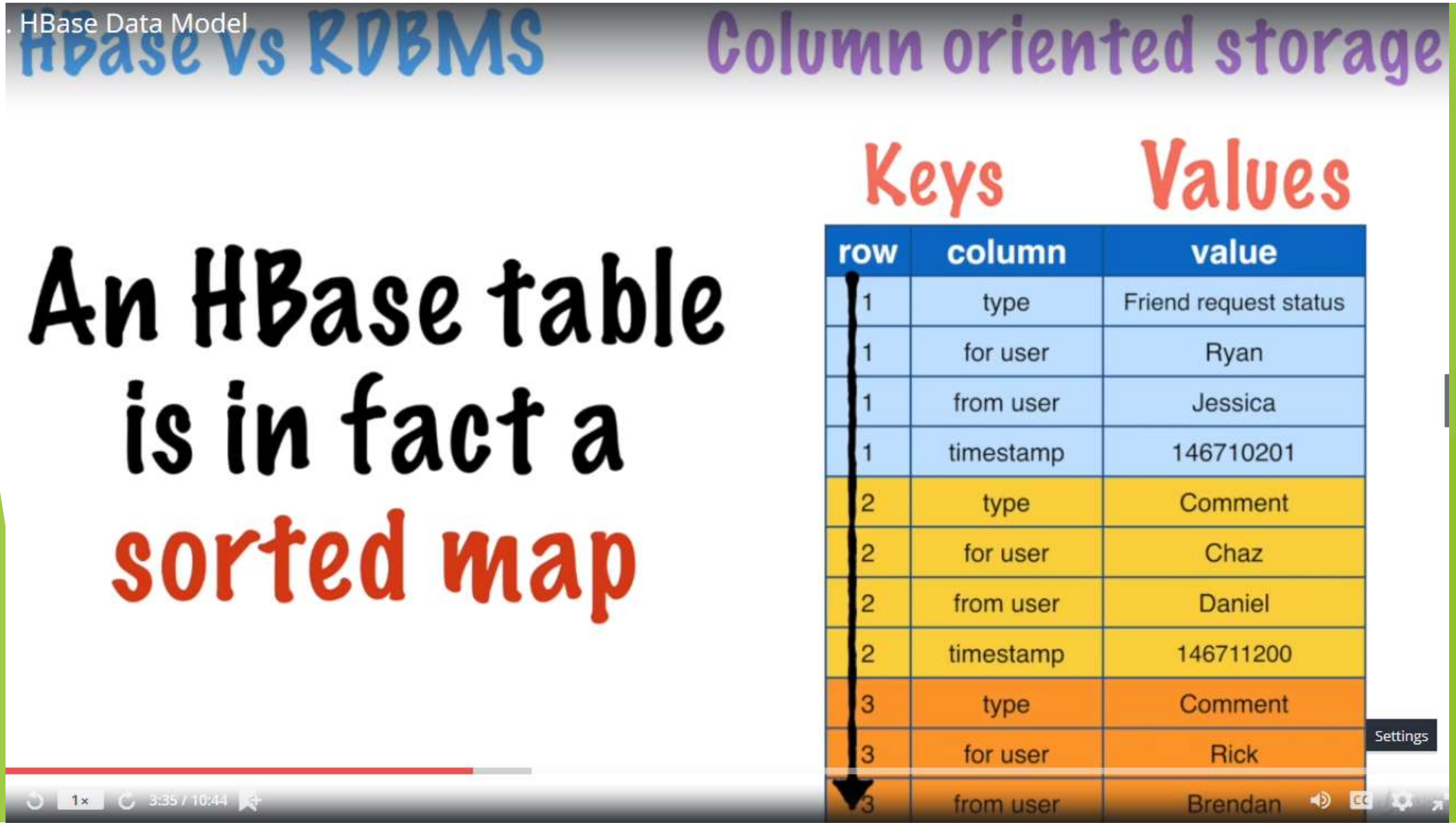
Value = Chaz

HBase Vs RDBMS

Column oriented storage

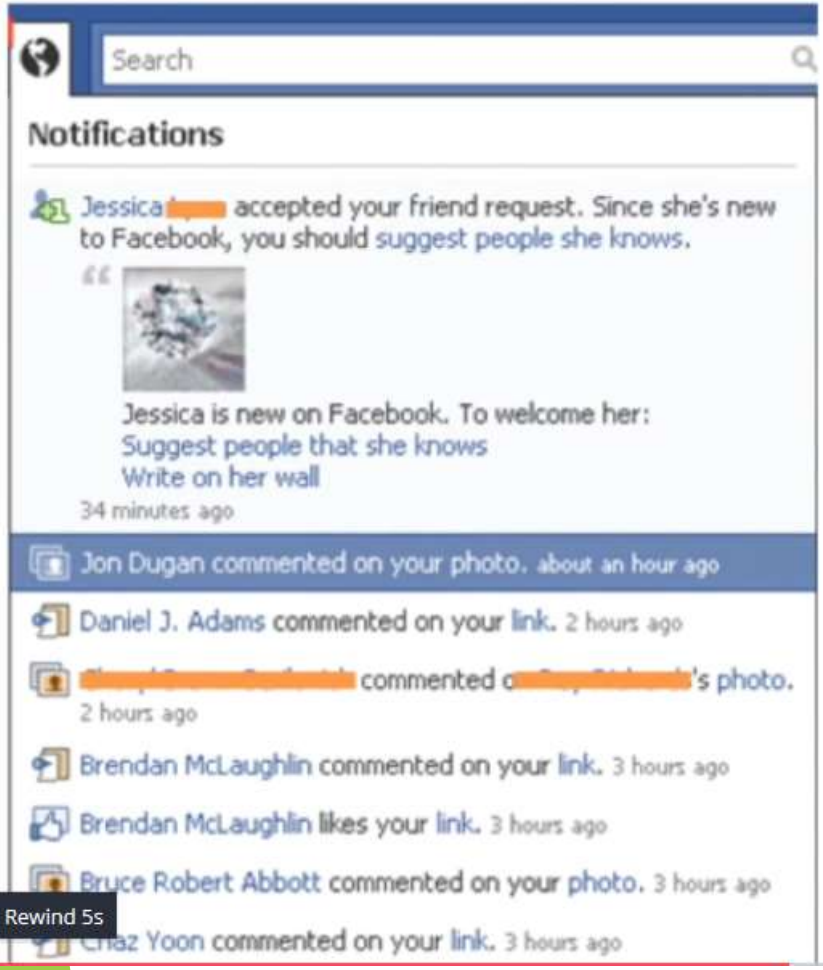
id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

row	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
2	type	Comment
2	for user	Chaz
2	from user	Daniel
2	timestamp	146711200
3	type	Comment
3	for user	Rick
3	from user	Brendan
3	timestamp	1467112205


. HBase Data Model
HBase Vs RDBMS
Column oriented storage
An HBase table is in fact a sorted map
Keys
Values
row
column
value
1
type
Friend request status
1
for user
Ryan
1
from user
Jessica
1
timestamp
146710201
2
type
Comment
2
for user
Chaz
2
from user
Daniel
2
timestamp
146711200
3
type
Comment
3
for user
Rick
3
from user
Brendan
Settings
1×
3:35 / 10:44

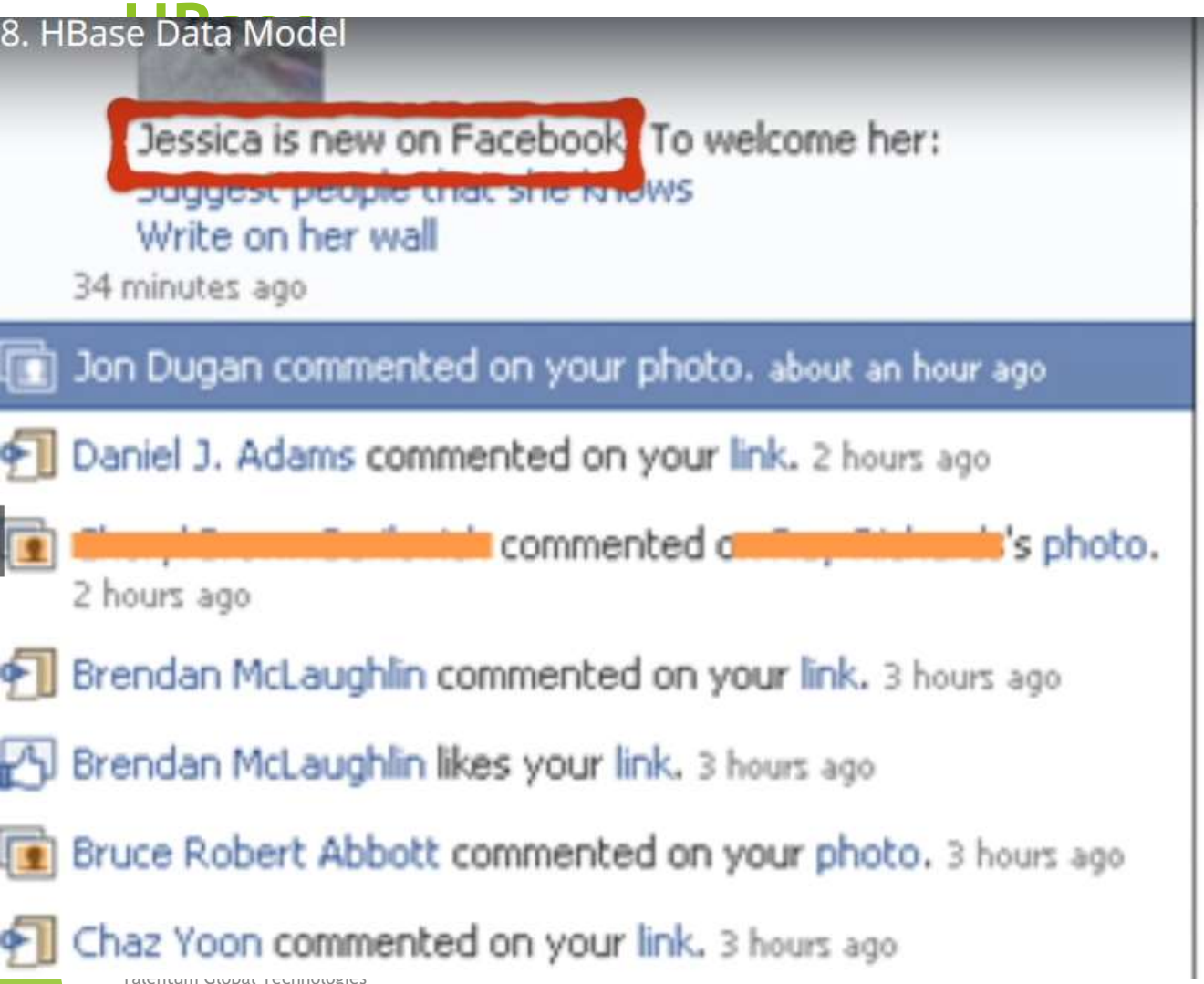
HBase Vs RDBMS

Column oriented storage



Rewind 5s

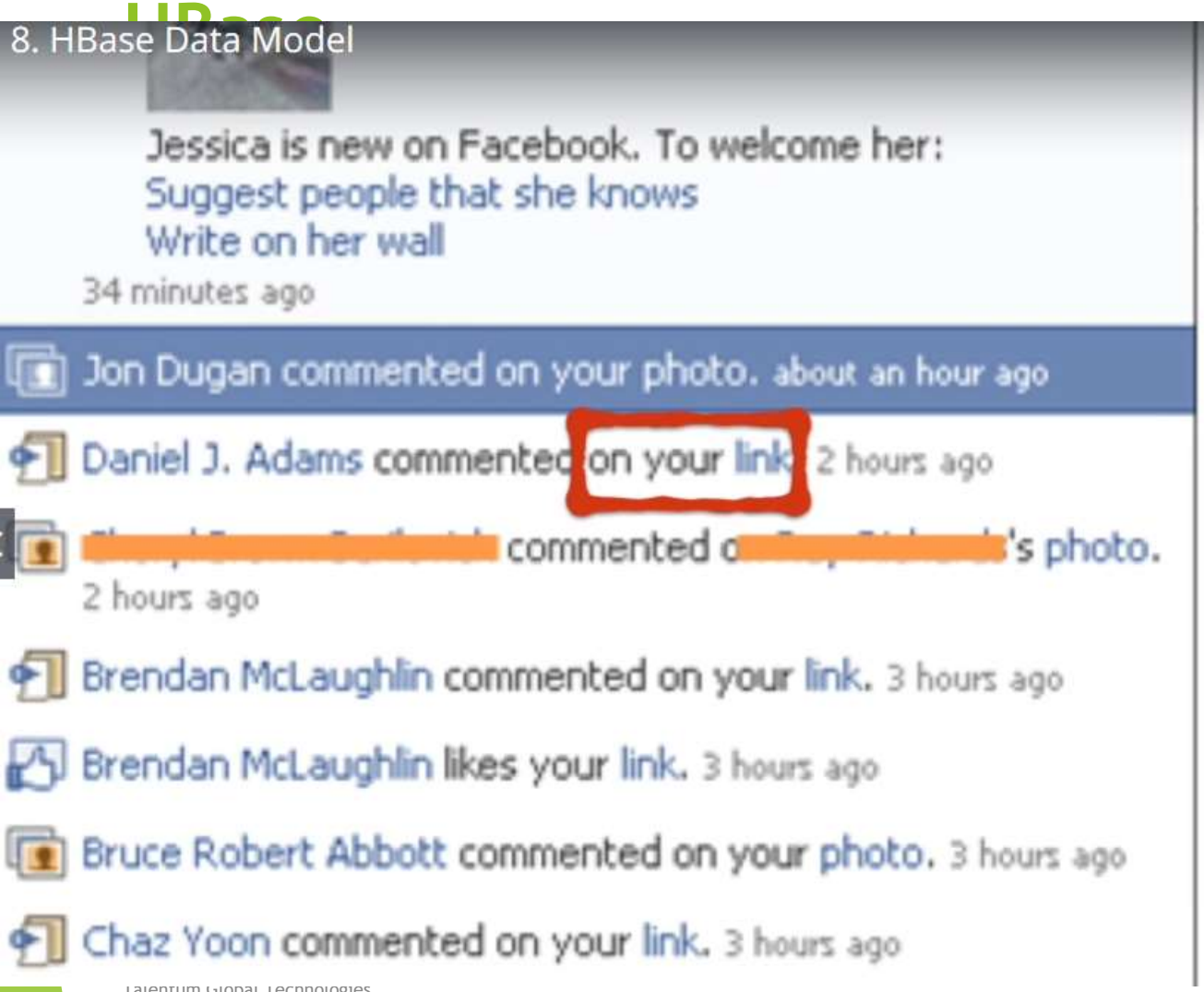
Let's say some notifications have special attributes depending on their type



Column oriented storage

Friend request notifications might have information about the friend

8. HBase Data Model



Column oriented storage

Comments and likes have information about a link or photo that prompted them

HBase

HBase Data Model

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type
1	Friend request	Ryan	Jessica	146710201	new
2	Comment	Chaz	Daniel	146711200	-
3	Comment	Rick	Brendan	1467112205	-
4	Like	Rick	Brendan	1467112213	-

HBase

8. HBase Data Model

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request	Ryan	Jessica	146710201	new	-
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request	Ryan	Jessica	146710201	new	-
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

In the RDBMS table, each of these attributes becomes a new column

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request	Ryan	Jessica	146710201	new	-
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

This results in tables that are very sparse

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request	Ryan	Jessica	146710201	new	-
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

In an RDBMS, Sparse tables utilize disk space even for these empty cells

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request	Ryan	Jessica	146710201	new	-
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

In a column-oriented store, these cells can be skipped completely

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request status	Ryan	Jessica	146710201	new	
2	Comment	Chaz	Daniel	146711200	-	link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

row id	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
1	friend type	new

HBase Vs RDBMS

Column oriented storage

id	type	for user	from user	timestamp	friend type	commented on
1	Friend request status	Ryan	Jessica	146710201	new	
2	Comment	Chaz	Daniel	146711200		link
3	Comment	Rick	Brendan	1467112205	-	photo
4	Like	Rick	Brendan	1467112213	-	-

row id	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
1	friend type	new
2	type	Friend request status
2	for user	Ryan
2	from user	Jessica
2	timestamp	146710201
2	commented on	link

HBase


3. HBase Data Model
HBase Vs RDBMS
Column oriented storage
Column oriented storage has some
powerful advantages

HBase

HBase Vs RDBMS

Column oriented storage

Column oriented storage has some powerful advantages

1. You can store really sparse tables very efficiently

HBase Vs RDBMS — Column oriented storage

Column oriented storage has some powerful advantages

1. You can store really sparse tables very efficiently

2. You can accommodate dynamically changing attributes

HBase Vs RDBMS

Column oriented storage

Each row id can have a different set of col ids

2. You can accommodate dynamically changing attributes

HBase Vs RDBMS

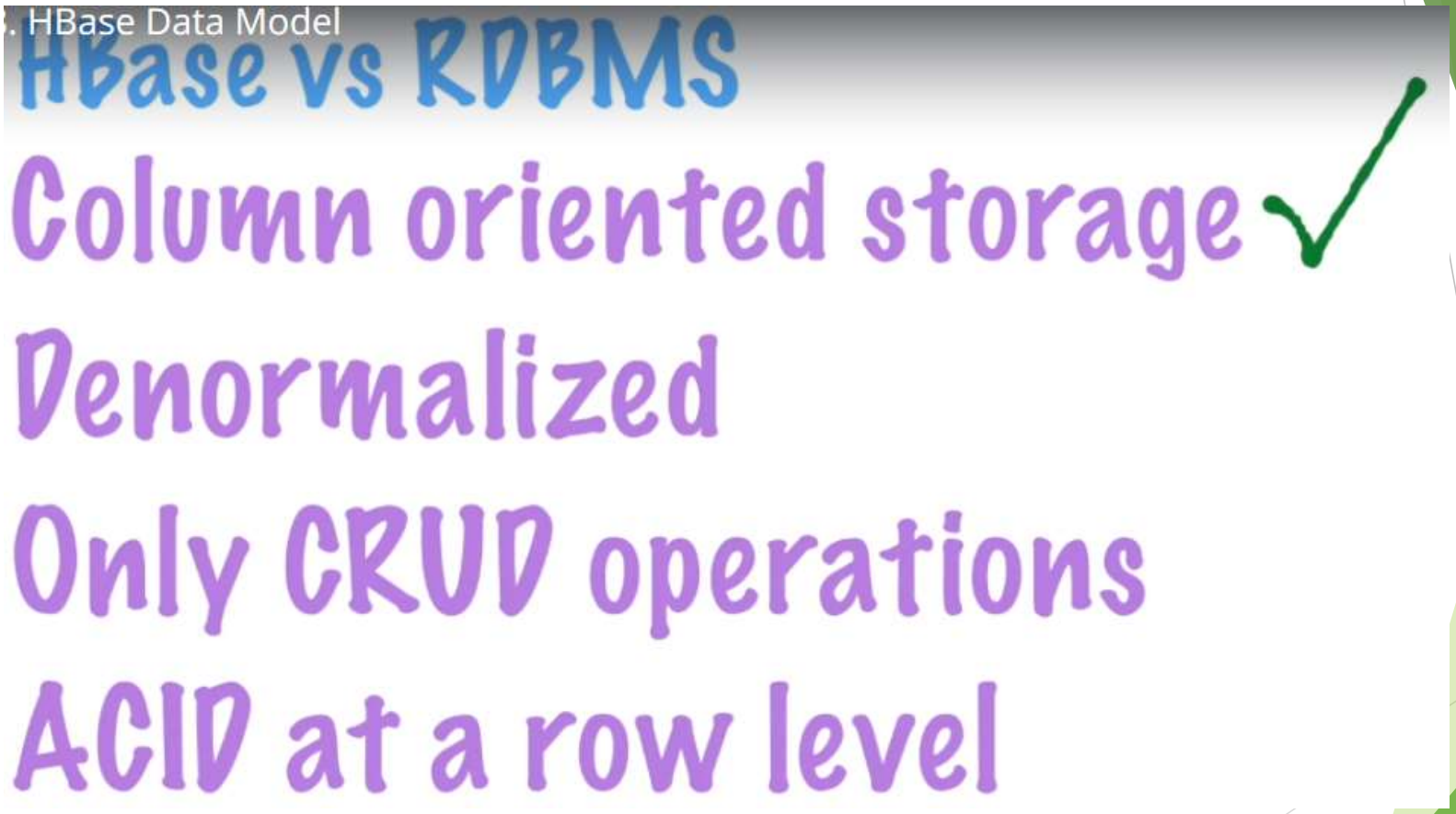
Column oriented storage

The schema for a row id is not fixed, you can keep changing it

ie, Add or remove new col ids

2. You can accommodate dynamically changing attributes

HBase


HBase Data Model
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level

HBase


3. HBase Data Model
HBase vs RDBMS
Denormalized
LET'S SAY WE HAVE AN EMPLOYEES DATABASE
WE WANT TO CAPTURE EMPLOYEE NAME, ADDRESS, SUBORDINATES

HBase vs RDBMS

Denormalized

A TRADITIONAL RDBMS WOULD MODEL IT AS 3 TABLES

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

HBase vs RDBMS

Denormalized

A TRADITIONAL RDBMS WOULD MODEL IT AS 3 TABLES

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

THIS KIND OF DESIGN MINIMIZES REDUNDANT STORAGE OF DATA

HBase vs RDBMS

Denormalized

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

THESE STREET AND CITY NAMES ARE ONLY STORED ONCE

AND REFERRED TO BY AN INTEGER ID THEREAFTER

HBase vs RDBMS

Denormalized

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

NORMALIZATION OPTIMIZES FOR STORAGE

HBase vs RDBMS

Denormalized

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

IN A DISTRIBUTED SYSTEM,
STORAGE IS CHEAP

HBase vs RDBMS

Denormalized

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

IN A DISTRIBUTED SYSTEM,
STORAGE IS CHEAP

INSTEAD YOU NEED TO
OPTIMIZE DISK SEEKS

HBase vs RDBMS

Denormalized

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

IF YOU STORE DATA ACROSS DIFFERENT TABLES

YOU HAVE TO PERFORM DISK SEEKS FOR EACH TABLE

HBase vs RDBMS

Denormalized

INSTEAD WE CAN EMBED ALL 3 TABLES INTO A SINGLE TABLE

EmpID	EmpName	Address	Subordinates
1	Vitthal	<STRUCT>	<ARRAY>

```
"Street":"Bellandur",
"City":"Bangalore"
```

```
("Anuradha",
"Arun",
"Swetha")
```

HBase vs RDBMS

Denormalized

EmpID	EmpName	Address	Subordinates
1	Vitthal	<STRUCT>	<ARRAY>

THIS IS A DENORMALIZED DESIGN

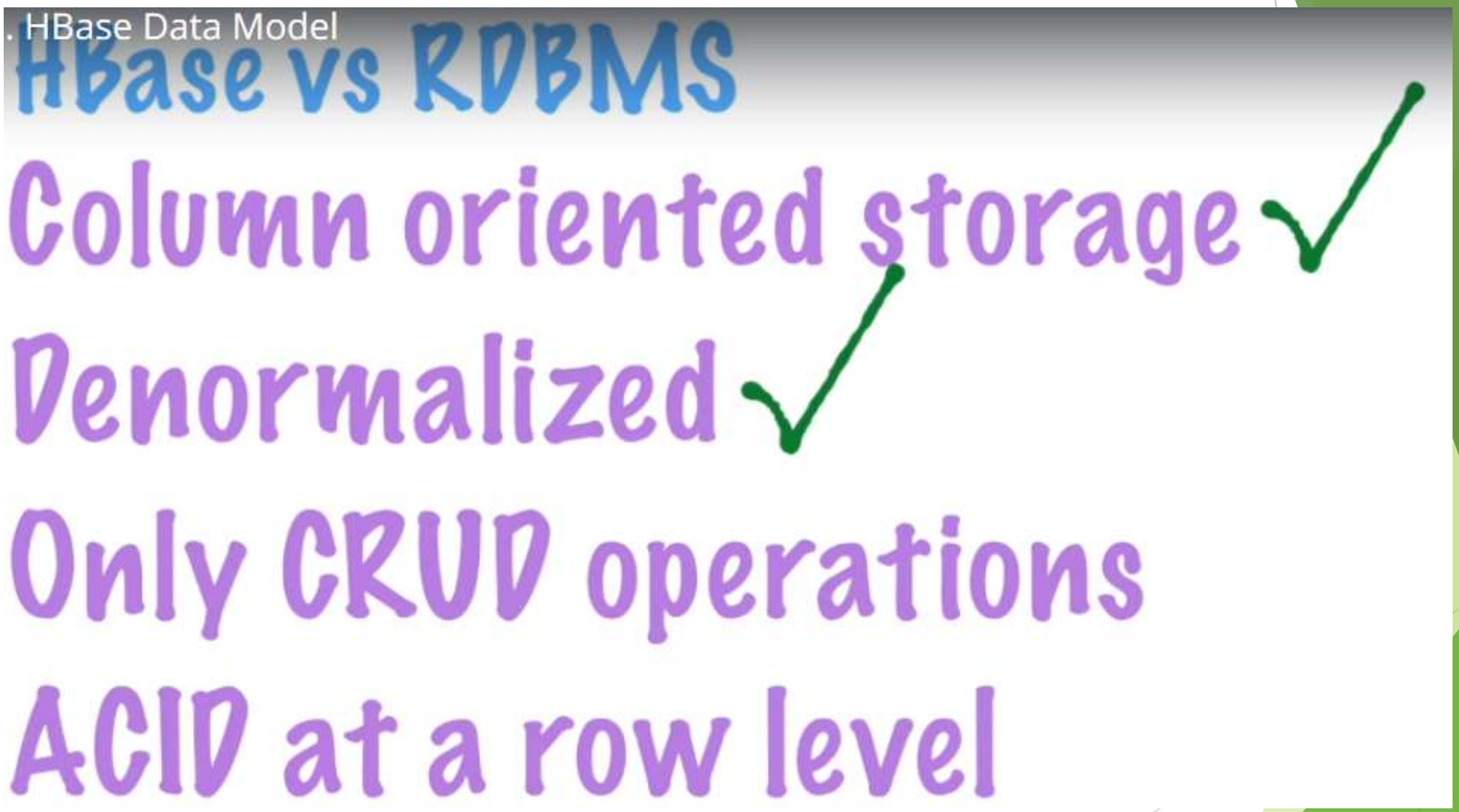
HBase vs RDBMS

Denormalized

EmpID	EmpName	Address	Subordinates
1	Vitthal	<STRUCT>	<ARRAY>

ALL THE DATA CORRESPONDING TO AN EMPLOYEE IS STORED IN A SINGLE TABLE

HBase


HBase Data Model
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level

HBase

HBase vs RDBMS Only CRUD operations

HBase architecture is designed such that you can get random read-write access to a specific row

HBase

HBase vs RDBMS

Only CRUD operations

Unlike, traditional RDBMS, HBase does not support SQL

HBase vs RDBMS

Only CRUD operations

Unlike, traditional RDBMS, HBase does not support SQL

NoSQL

HBase

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

Create

Read

Update

Delete

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

Create — Add a new value to the table

Read

Update

Delete

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

Create — Add a new value to the table

Read — Read the value for a specific row id, col id

Update

Delete

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

Create — Add a new value to the table

Read — Read the value for a specific row id, col id

Update — Update the value for a specific row id, col id

Delete

HBase vs RDBMS Only CRUD operations

HBase only supports a limited set of operations

Create — Add a new value to the table

Read — Read the value for a specific row id, col id

Update — Update the value for a specific row id, col id

Delete — Delete the value for a specific row id, col id

HBase vs RDBMS

Only CRUD operations

Create

Read

Update

Delete

All HBase operations deal with a specific row

HBase vs RDBMS

Only CRUD operations

Create

Read

Update

Delete

HBase does not support any operations across tables

No Joins

HBase vs RDBMS

Only CRUD operations

Create

Read

Update

Delete

HBase does not support any operations across row ids

No Grouping/Aggregation

HBase vs RDBMS Only CRUD operations

Create

Read

Update

Delete

This is another reason why denormalization is important in HBase

HBase vs KVBMS

Only CRUD operations

Create

Read

Update

Delete

All the data needed to describe an entity should be self-contained within its row id

HBase vs RDBMS Only CRUD operations

LET'S GO BACK TO THE EMPLOYEE EXAMPLE

A TRADITIONAL RDBMS WOULD MODEL IT AS 3 TABLES

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

EmpID	SubordinateEmpID
1	3
1	4
1	8

HBase vs RDBMS Only CRUD operations

WHEN AN APPLICATION ASKS FOR AN EMPLOYEE'S DETAILS

EmpID	EmpName	AddressId
1	Vitthal	1

AddressId	Street	City
1	Bellandur	Bangalore

YOU WOULD NEED TO JOIN 2 TABLES TO FETCH THE ADDRESS

HBase vs RDBMS

Only CRUD operations

WHEN AN APPLICATION ASKS FOR AN EMPLOYEE'S DETAILS

EmpID	EmpName	AddressId
1	Vitthal	1

EmpID	SubordinateEmpID
1	3
1	4
1	8

YOU WOULD NEED TO JOIN THESE 2 TABLES TWICE TO GET THE LIST OF SUBORDINATES FOR AN EMPLOYEE

HBase

Introducing CRUD operations

HBase vs RDBMS Only CRUD operations

IN AN RDBMS THESE JOINS CAN BE MADE EFFICIENT WITH THE ADDITION OF INDICES

HBase

Introducing CRUD operations

HBase vs RDBMS

Only CRUD operations

IN HBASE, THERE IS NO SUPPORT FOR JOINING TABLES ON THE FLY WHILE FETCHING THE DETAILS FOR 1 ROW

HBase

. Introducing CRUD operations

YOU COULD USE AN EXTERNAL APPLICATION LIKE MAPREDUCE TO PERFORM JOINS

10. HBase is different from Hive

HBase vs RDBMS

Only CRUD operations

YOU COULD USE AN EXTERNAL APPLICATION LIKE MAPREDUCE TO PERFORM JOINS

WHILE THIS IS FINE FOR ANALYTICAL QUERIES, IT WOULD NOT BE SUITABLE FOR TRANSACTION PROCESSING

HBase vs RDBMS

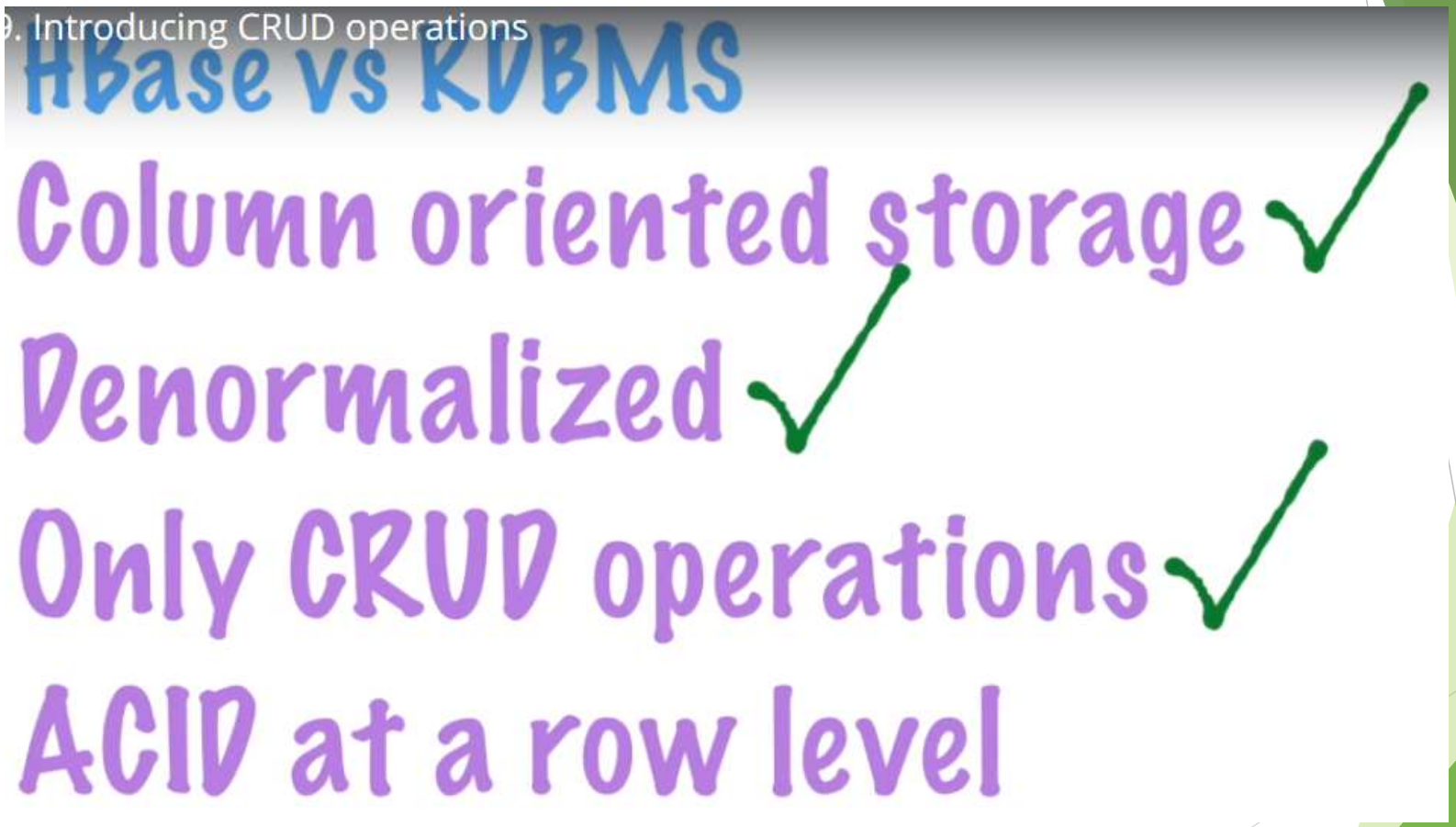
Only CRUD operations

IN A DENORMALIZED DESIGN

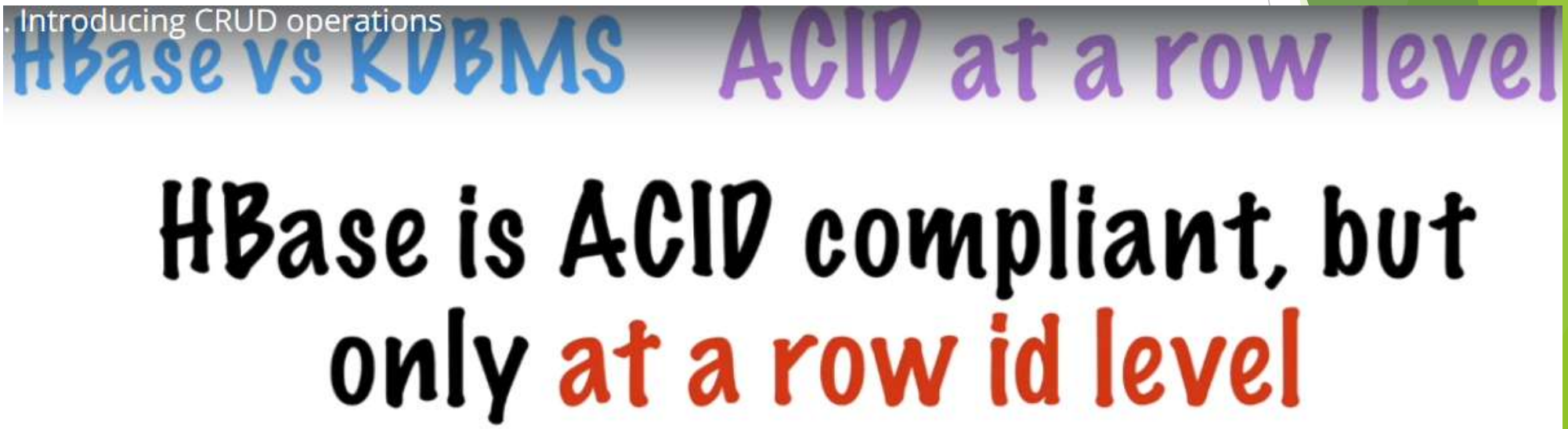
EmpID	EmpName	Address	Subordinates
1	Vitthal	<STRUCT>	<ARRAY>

YOU CAN USE THE HBASE SUPPORTED READ OPERATION TO READ THE ROW AND FETCH ALL THE DATA

HBase


. Introducing CRUD operations
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level

HBase


Introducing CRUD operations
HBase vs KVBMS
ACID at a row level
HBase is ACID compliant, but
only at a row id level

HBase vs RDBMS ACID at a row level

HBase is ACID compliant, but only at a row id level

For example, let's look at Atomicity

HBase vs RDBMS

ACID at a row level

Atomic

Transaction 1:

Update values for 2 col ids within 1 row

Atomicity

If one col id update fails, the entire transaction fails

HBase vs RDBMS

ACID at a row level

Atomicity

Not Atomic

Transaction 2:

Update values
for 2 col ids
for 10 row ids

. Introducing CRUD operations

HBase vs RDBMS

ACID at a row level

Atomicity

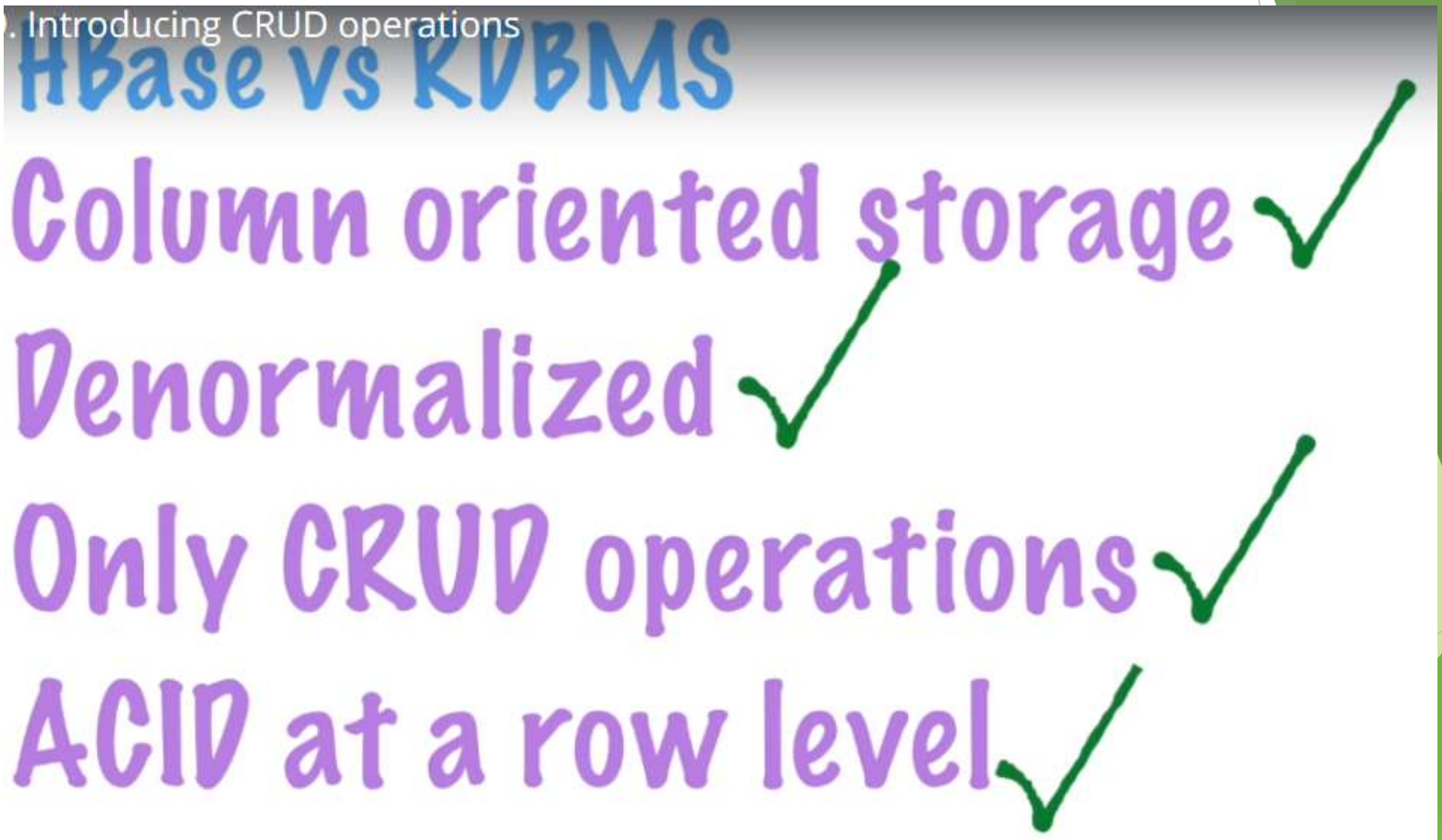
If the operation fails after 5 row ids are updated, the row ids which are updated remain updated

Not Atomic

Transaction 2:

Update values for 2 col ids for 10 row ids

HBase


Introducing CRUD operations
HBase vs RDBMS
Column oriented storage
Denormalized
Only CRUD operations
ACID at a row level

HBase

If you are familiar with the Hadoop ecosystem, you might know of other technologies which seem similar to HBase

HIVE FOR INSTANCE

HBase


HBase is a database management system
HIVE IS A DATA WAREHOUSE

HBase

HBase is a database management system

Used for both transaction processing and analytical processing

HIVE IS A DATA WAREHOUSE

HBase

HBase is a database management system

Used for both transaction processing and analytical processing

HIVE IS A DATA WAREHOUSE

Used only for analytical processing

HBase

HBase is a database management system

Provides low latency and random access for some supported operations

HIVE IS A DATA WAREHOUSE

HBase

HBase is a database management system

Provides low latency and random access for some supported operations

HIVE IS A DATA WAREHOUSE

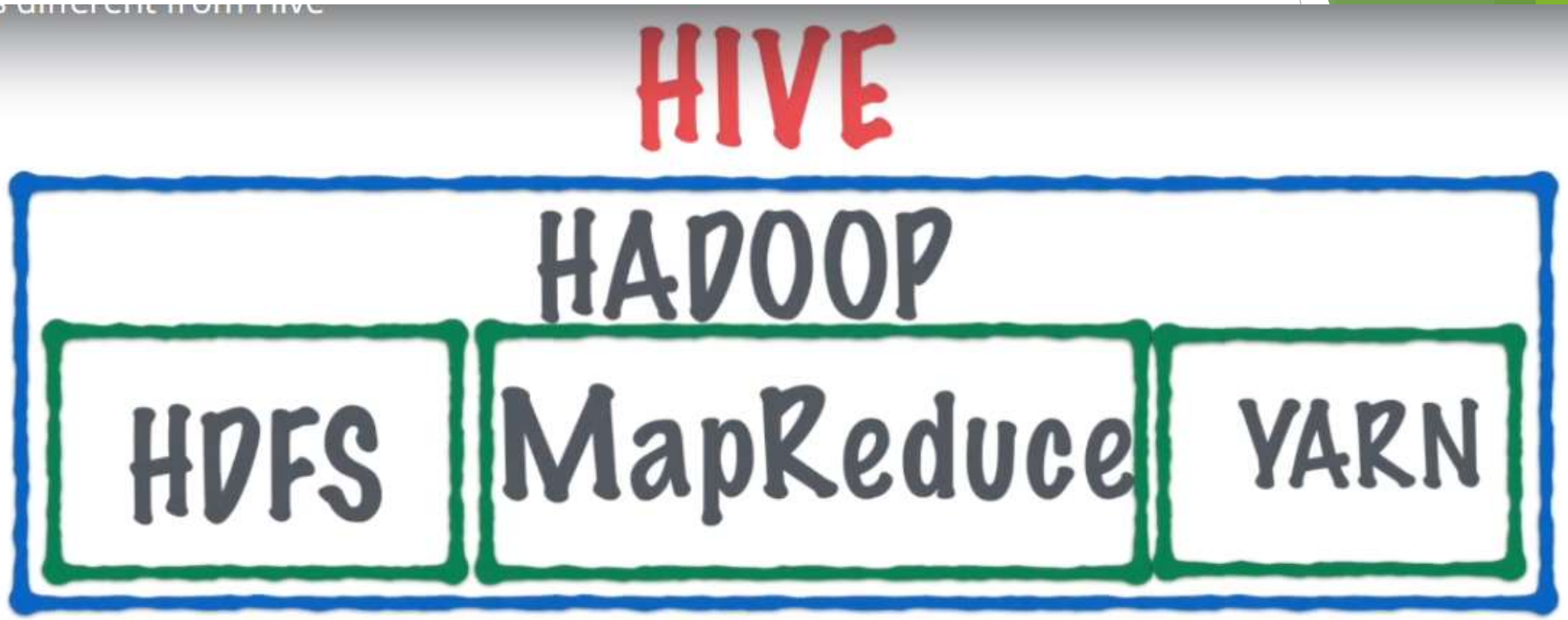
Only suitable for batch processing jobs that can tolerate high latency

HBase

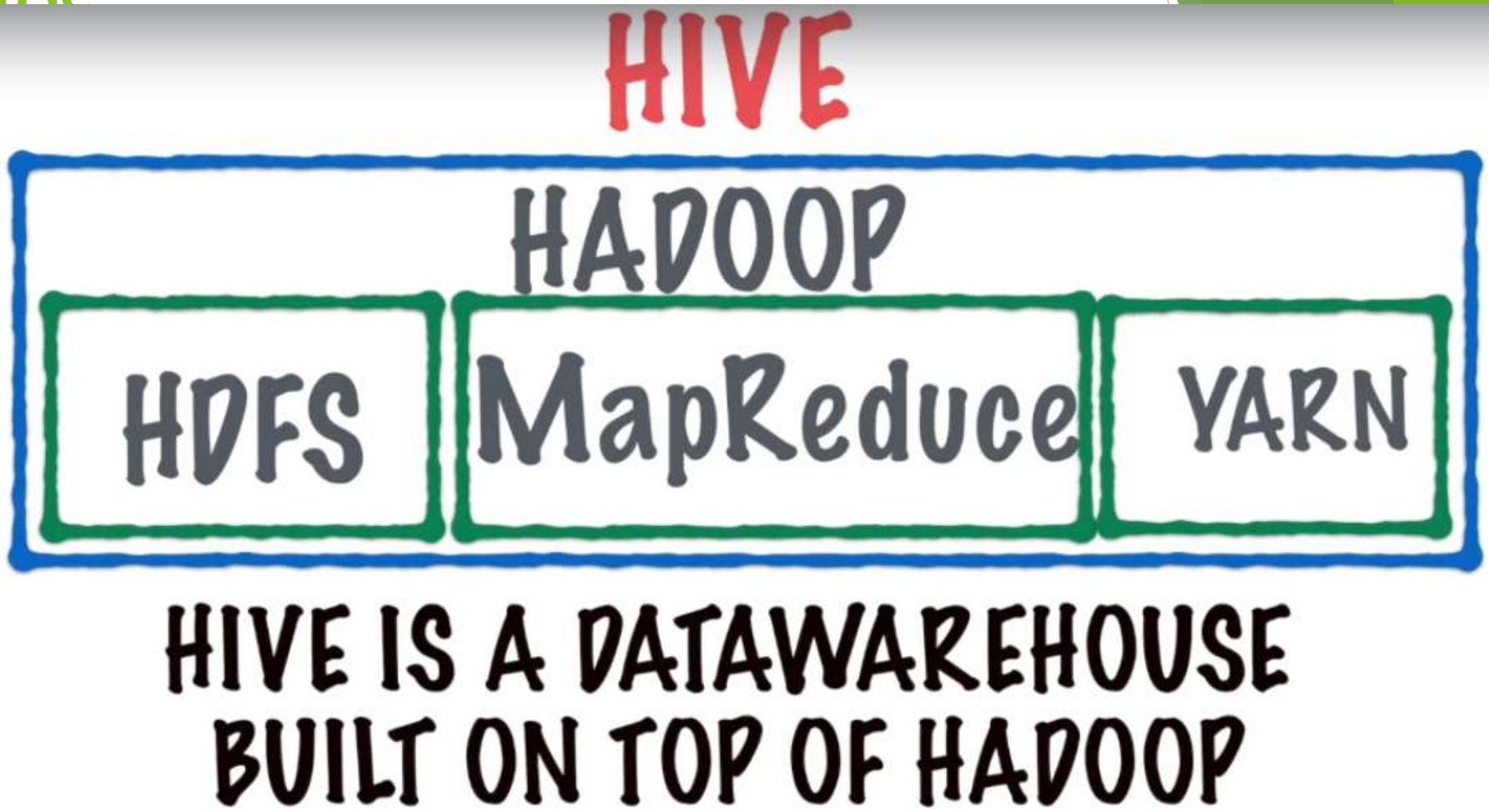
HBase does not provide
any SQL interface

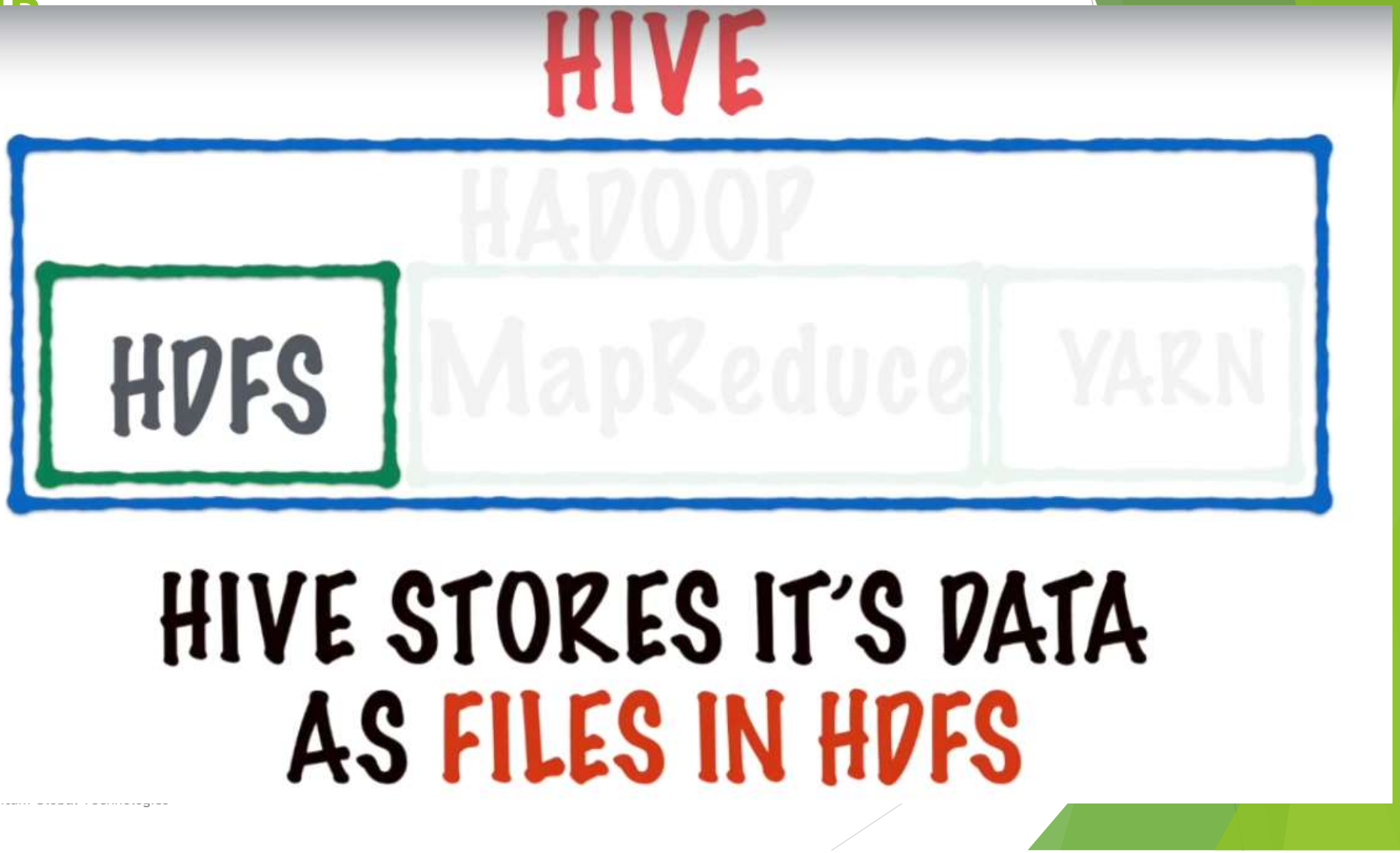
Hive does!

HBase

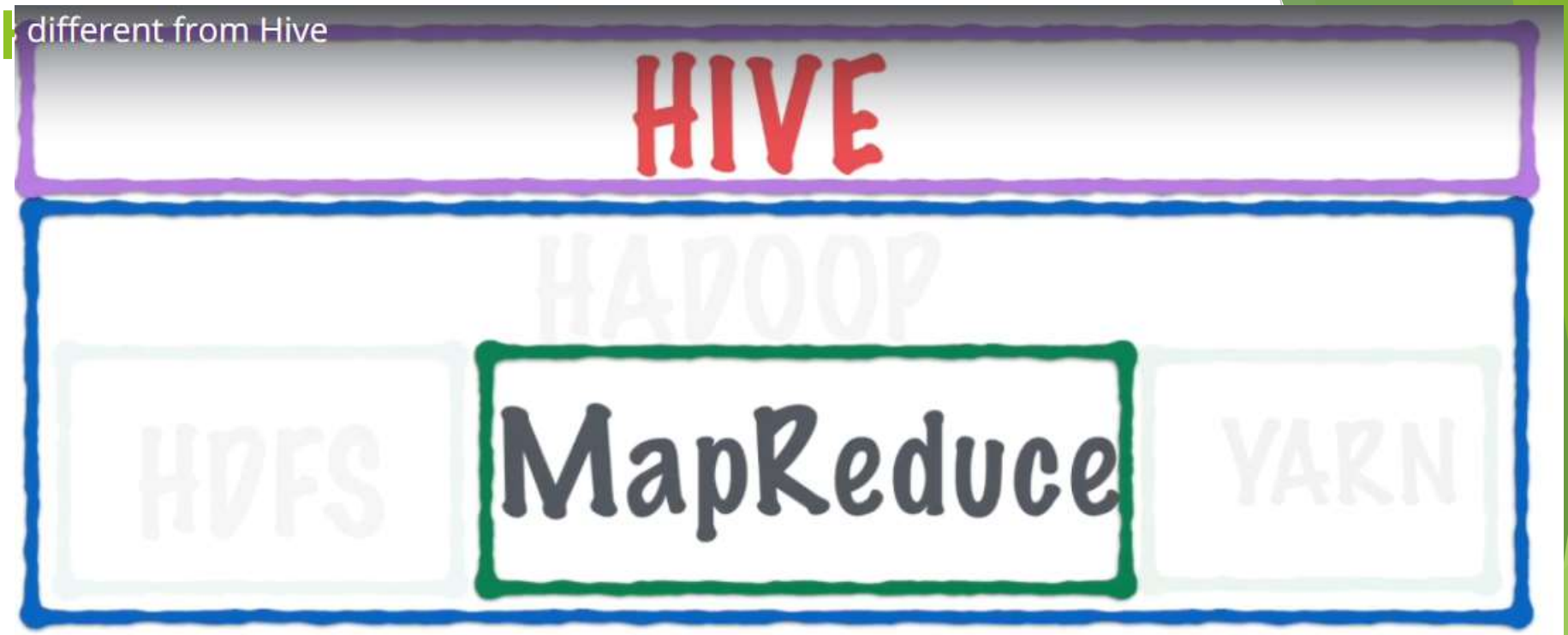


HBase


HIVE
HADOOP
HDFS
MapReduce
YARN
HIVE IS A DATAWAREHOUSE
BUILT ON TOP OF HADOOP


HIVE
HADOOP
HDFS
MapReduce
YARN
HIVE STORES IT'S DATA
AS FILES IN HDFS

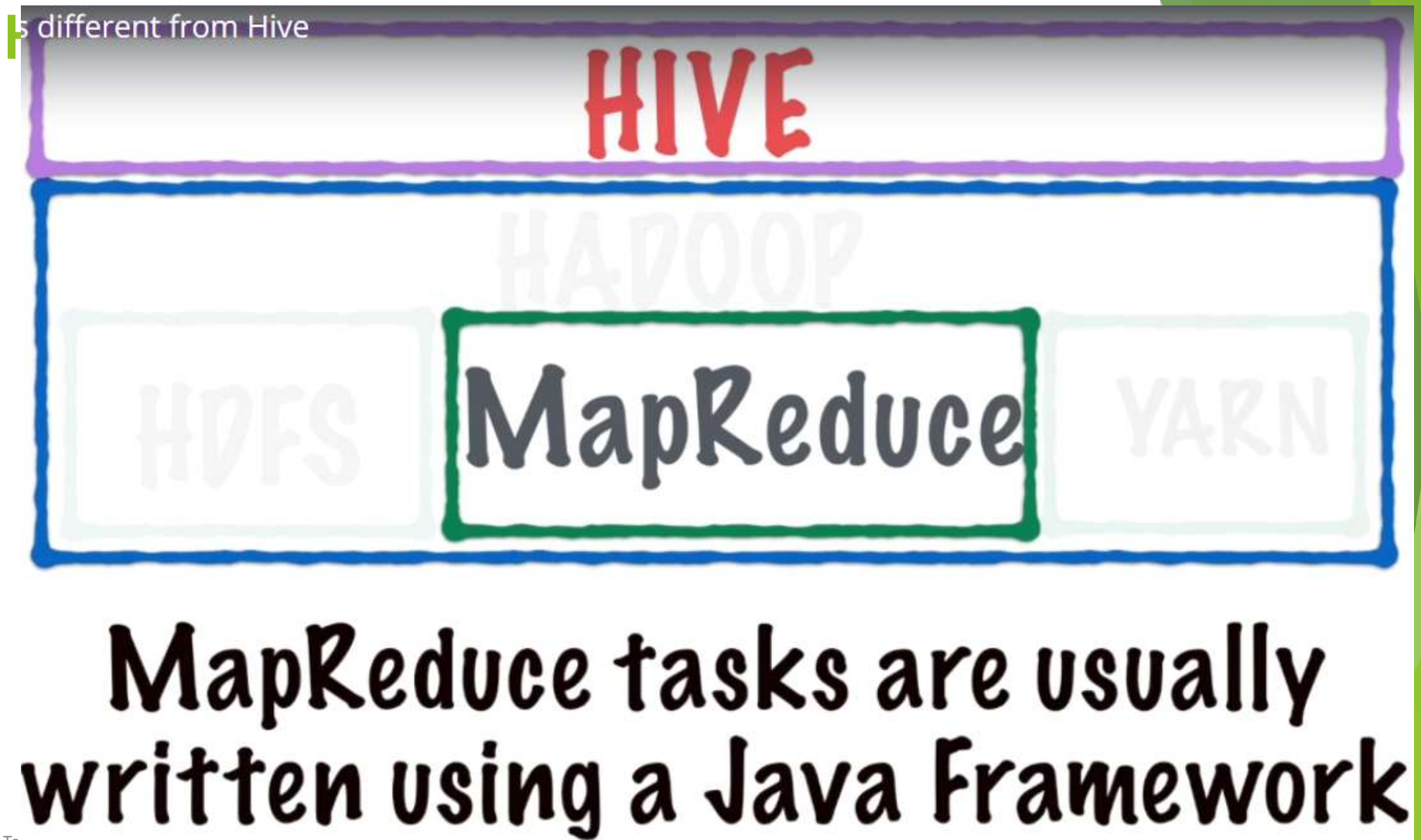
different from Hive



All processing tasks in Hadoop are run using MapReduce tasks

Ta

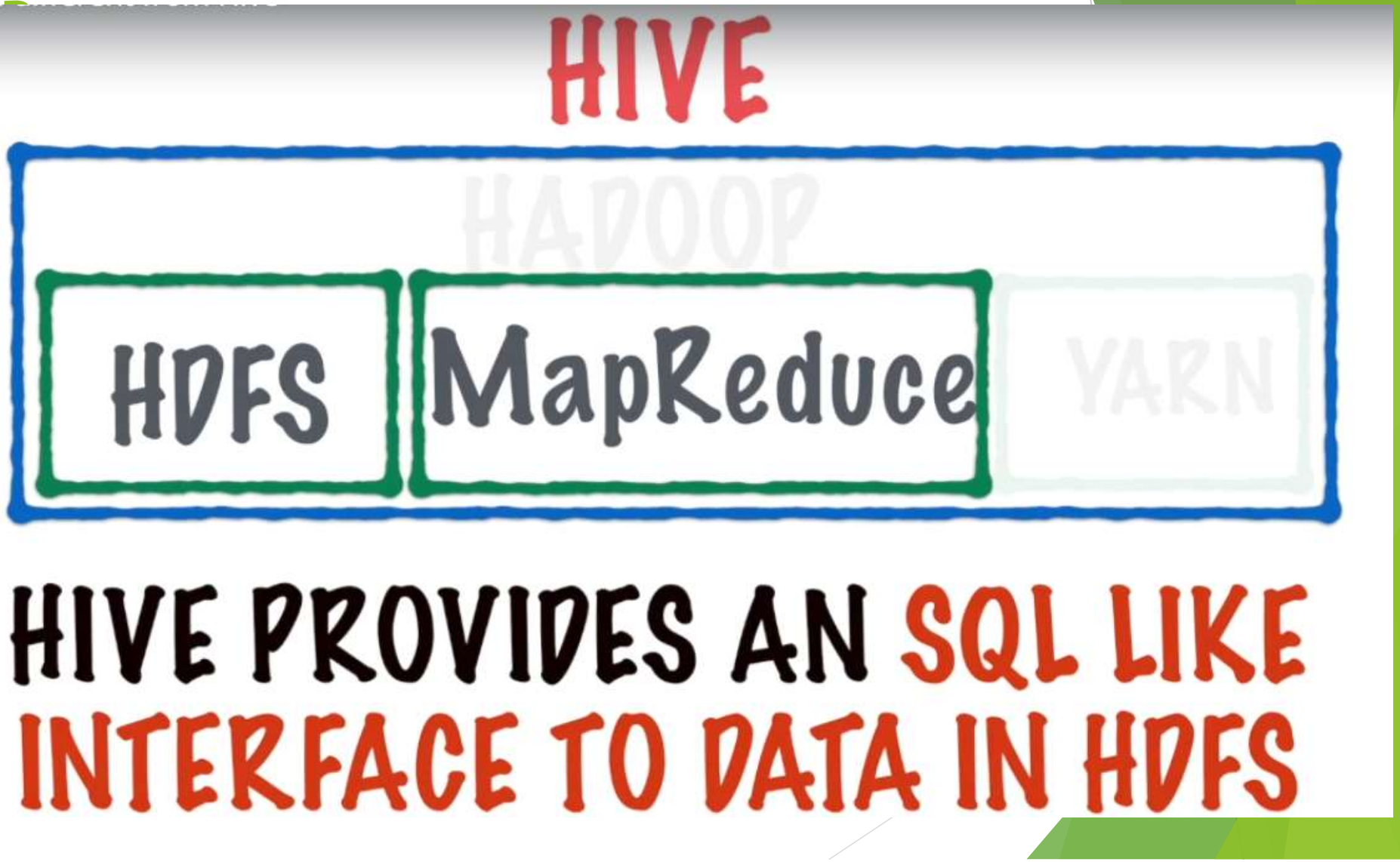
s different from Hive

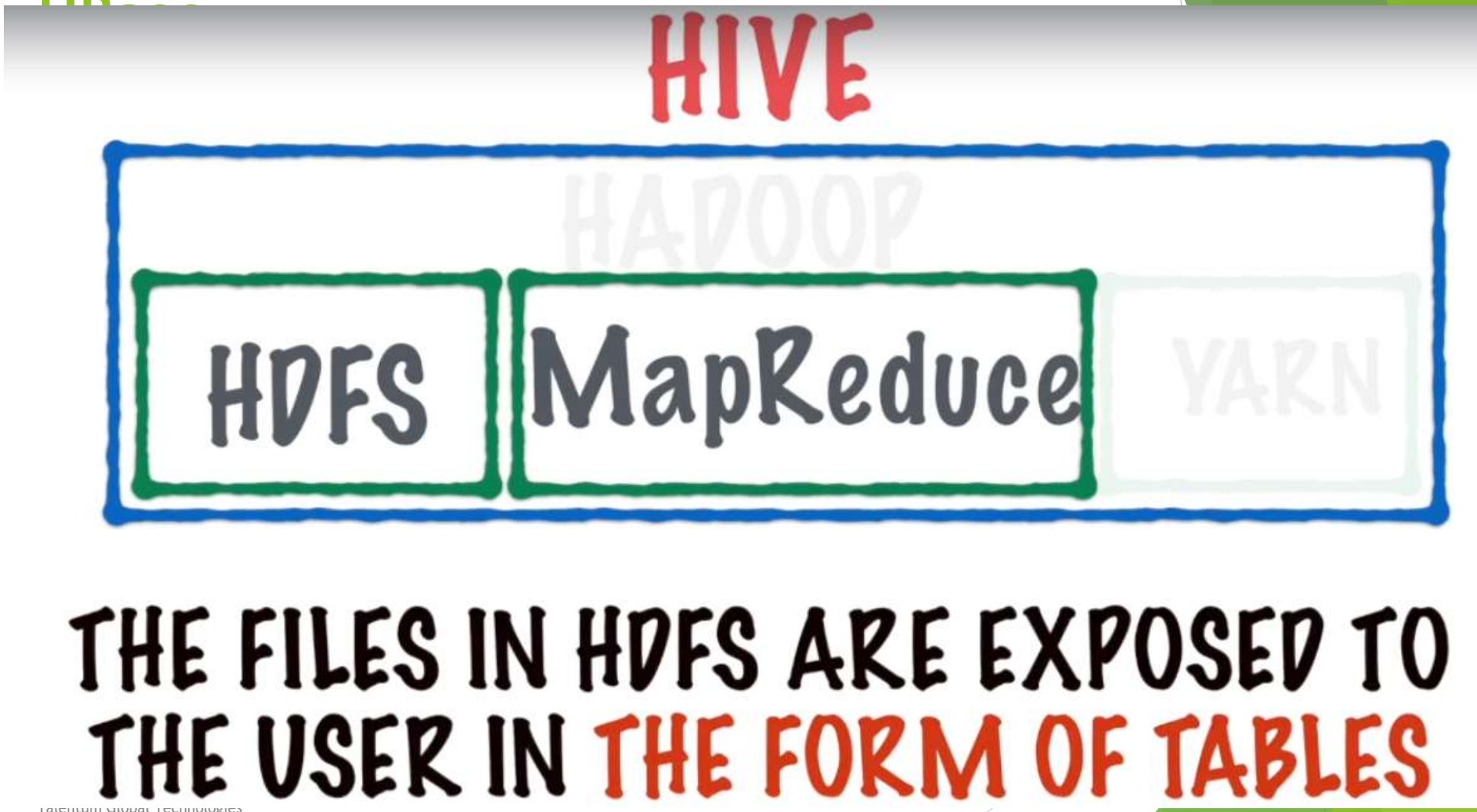


MapReduce tasks are usually written using a Java Framework

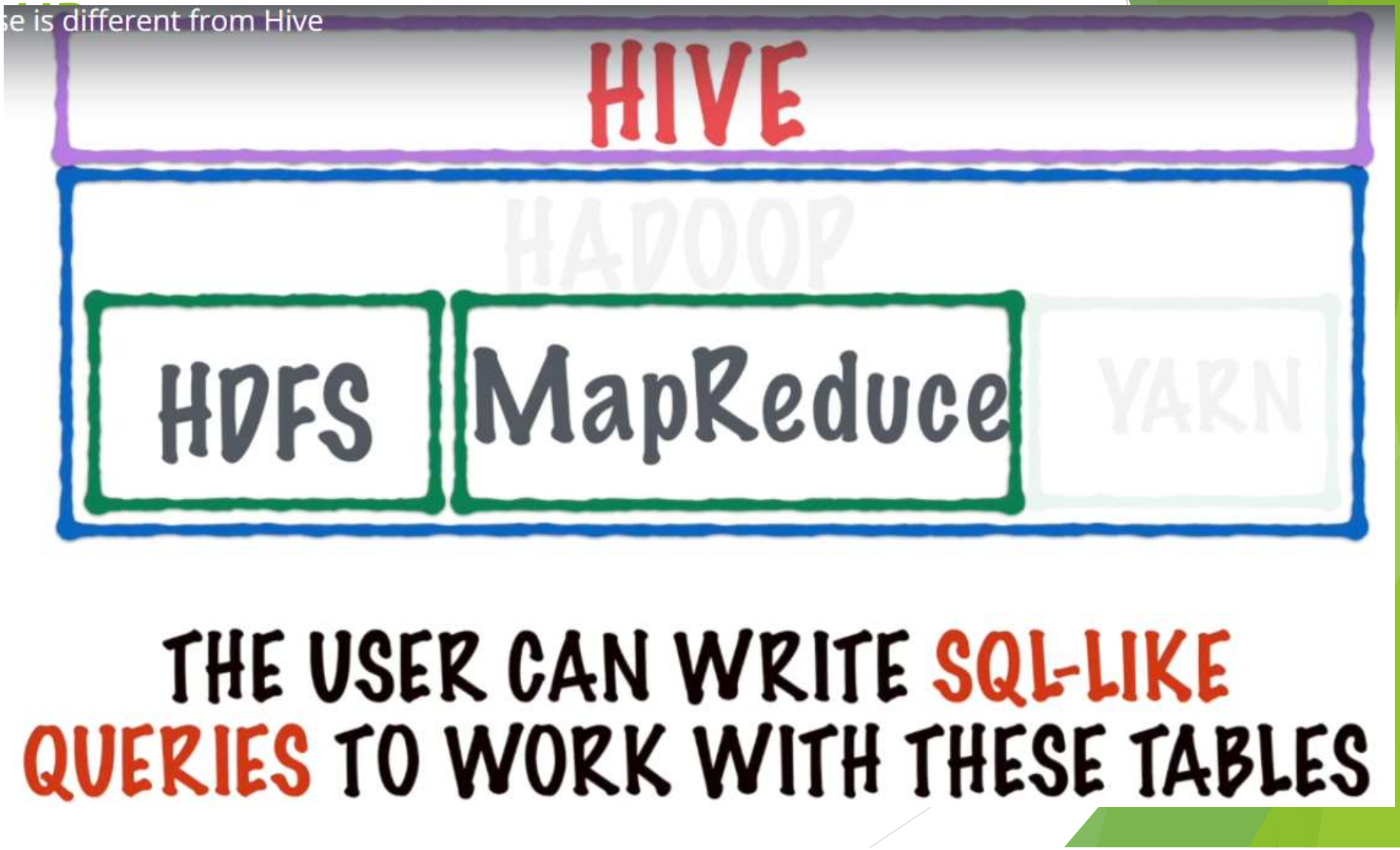
Ta


s different from Hive
HIVE
HADOOP
HDFS
MapReduce
YARN
Writing these MapReduce tasks can be pretty daunting

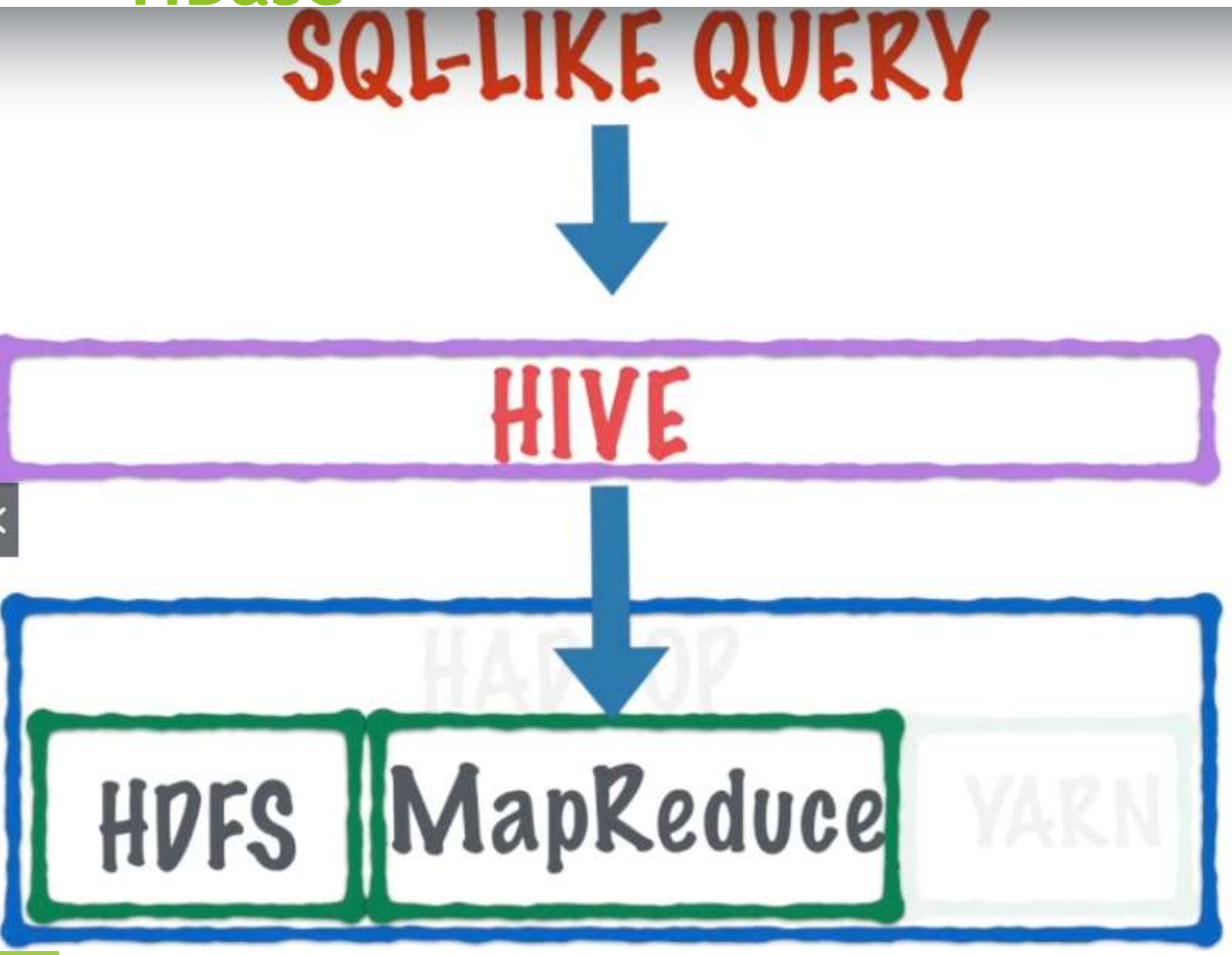

HIVE
HADOOP
HDFS
MapReduce
YARN
HIVE PROVIDES AN SQL LIKE
INTERFACE TO DATA IN HDFS


HIVE
HADOOP
HDFS
MapReduce
YARN
THE FILES IN HDFS ARE EXPOSED TO
THE USER IN THE FORM OF TABLES

se is different from Hive



HBase



HIVE WILL TRANSLATE THE QUERY INTO 1/MORE MAPREDUCE TASKS

HBase


Base is different from Hive
HIVE
USED FOR BATCH PROCESSING
HBASE
USED FOR BOTH BATCH AND TRANSACTION PROCESSING

HBase


HIVE
USED FOR BATCH PROCESSING
PROVIDES AN SQL SKIN FOR HADOOP
HBASE
USED FOR BOTH BATCH AND TRANSACTION PROCESSING
NO SQL INTERFACE

HBase


HIVE
USED FOR BATCH PROCESSING
PROVIDES AN SQL SKIN FOR HADOOP
USES BOTH HDFS AND THE MAPREDUCE ENGINE
HBASE
USED FOR BOTH BATCH AND TRANSACTION PROCESSING
NO SQL INTERFACE
USES HDFS BUT HAS IT'S OWN ARCHITECTURE

HBase

HIVE

USED FOR BATCH PROCESSING

PROVIDES AN SQL SKIN FOR HADOOP

USES BOTH HDFS AND THE MAPREDUCE ENGINE

DATA MODEL IS SIMILAR TO DATABASES (TABLES WITH FIXED SCHEMA)

HBASE

USED FOR BOTH BATCH AND TRANSACTION PROCESSING

NO SQL INTERFACE

USES HDFS BUT HAS IT'S OWN ARCHITECTURE

DATA MODEL IS COLUMN ORIENTED STORAGE

Hbase Architecture

Hbase Architecture

Hbase's Architecture is inspired by

Hbase Architecture

Recap

HBase vs RDBMS

This is how data is stored in traditional databases

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Hbase Architecture

Recap

Column oriented storage

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Data is stored in a map

Key = <Row id, Col id>

Value = <data>

Hbase Architecture

Recap

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Column oriented storage

Data is stored in a map

Key = 2, for_user

Value = Chaz

Hbase Architecture

Recap

id	type	for user	from user	timestamp
1	Friend request status	Ryan	Jessica	146710201
2	Comment	Chaz	Daniel	146711200
3	Comment	Rick	Brendan	1467112205
4	Like	Rick	Brendan	1467112213

Column oriented storage

row	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
2	type	Comment
2	for user	Chaz
2	from user	Daniel
2	timestamp	146711200
3	type	Comment
3	for user	Rick
3	from user	Brendan
3	timestamp	1467112205

Hbase Architecture

Recap

An HBase table is in fact a sorted map

Column oriented storage

Keys Values

row	column	value
1	type	Friend request status
1	for user	Ryan
1	from user	Jessica
1	timestamp	146710201
2	type	Comment
2	for user	Chaz
2	from user	Daniel
2	timestamp	146711200
3	type	Comment
3	for user	Rick
3	from user	Brendan

Hbase Architecture

A sorted nested map

<Row id,
ColumnFamily,
<Column,
<Timestamp,Value>>>

Hbase Architecture

<Row id,

A sorted nested map

When you read data from HBase, it performs a lookup for the specified row id

Hbase Architecture

<Row id,

A sorted nested map

When you write data to HBase, it needs to insert the row id in the right place, so the rows are sorted

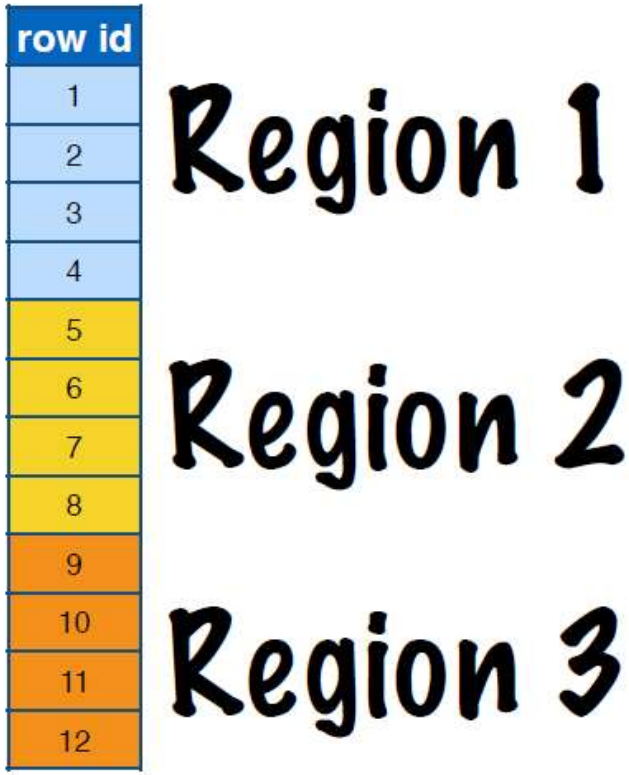
Hbase Architecture

<Row id,

A sorted nested map

HBase does this using Region Servers

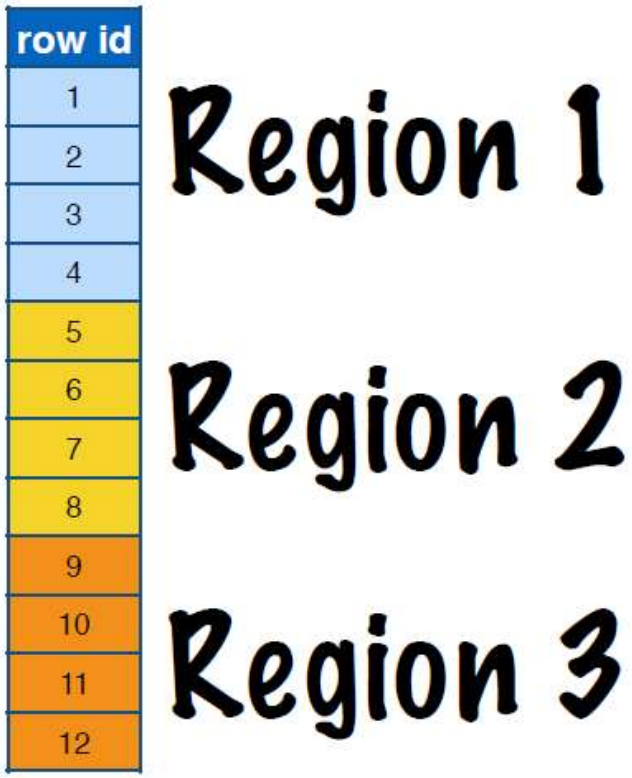
Hbase Architecture



Region Servers

Row ids in a table are divided into ranges called regions

Hbase Architecture



Region Servers

Each region is handled by a Region Server

Hbase Architecture

Region Servers

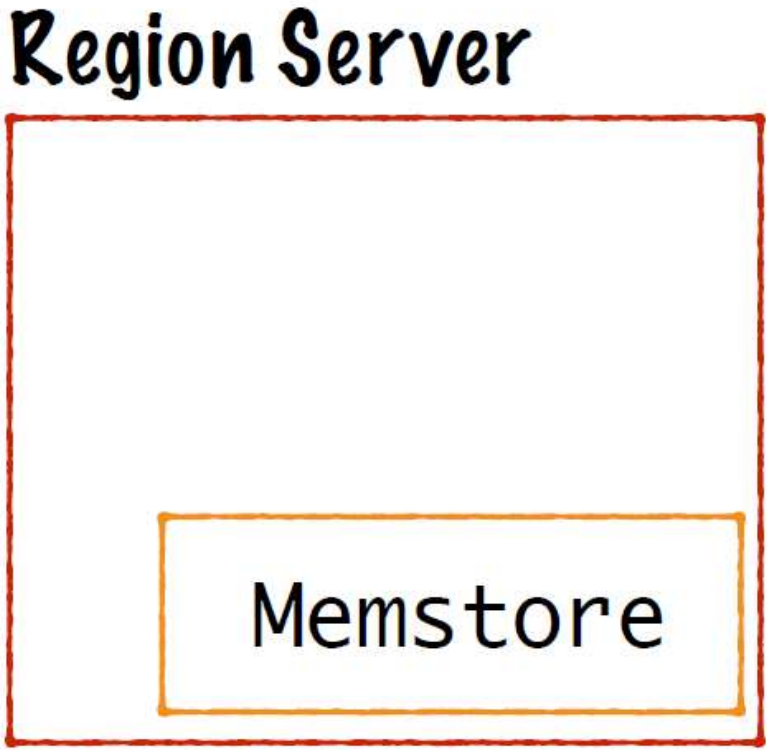
Regions serve as an index to perform fast lookup for where a row key belongs

Hbase Architecture

Region Servers

A region server handles all read-write operations to Regions that are allotted to it

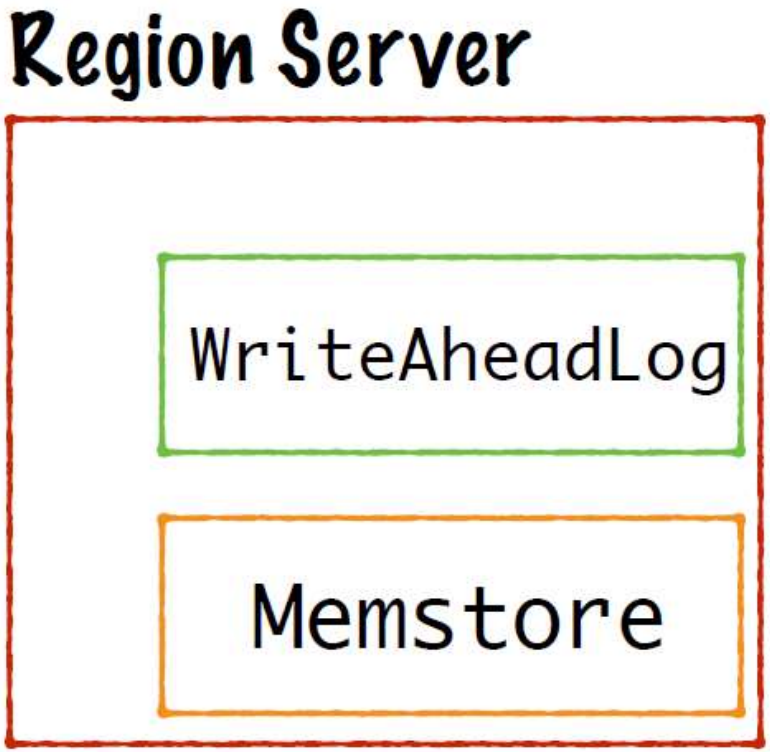
Hbase Architecture



Region Servers

Initially all writes are stored in memory

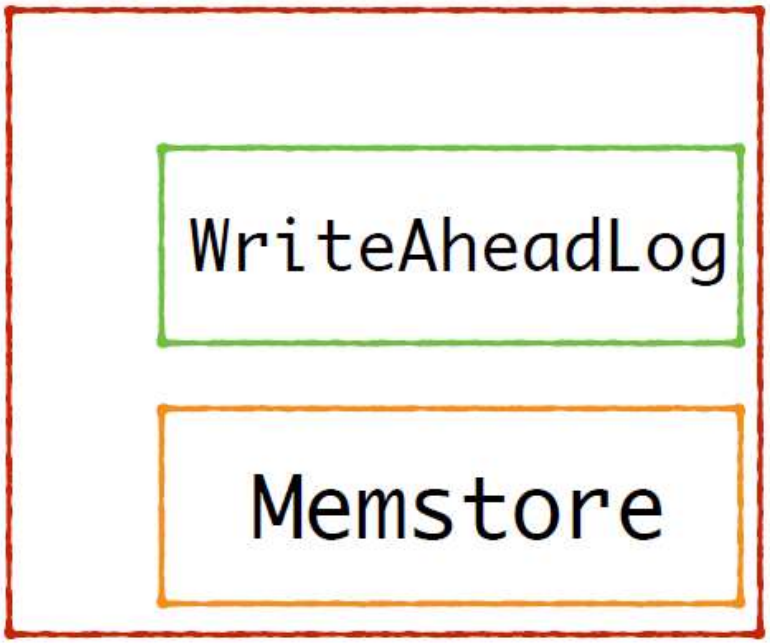
Hbase Architecture



Region Servers

Whenever there is a new change, the data is updated in the Memstore and a change log is written to disk

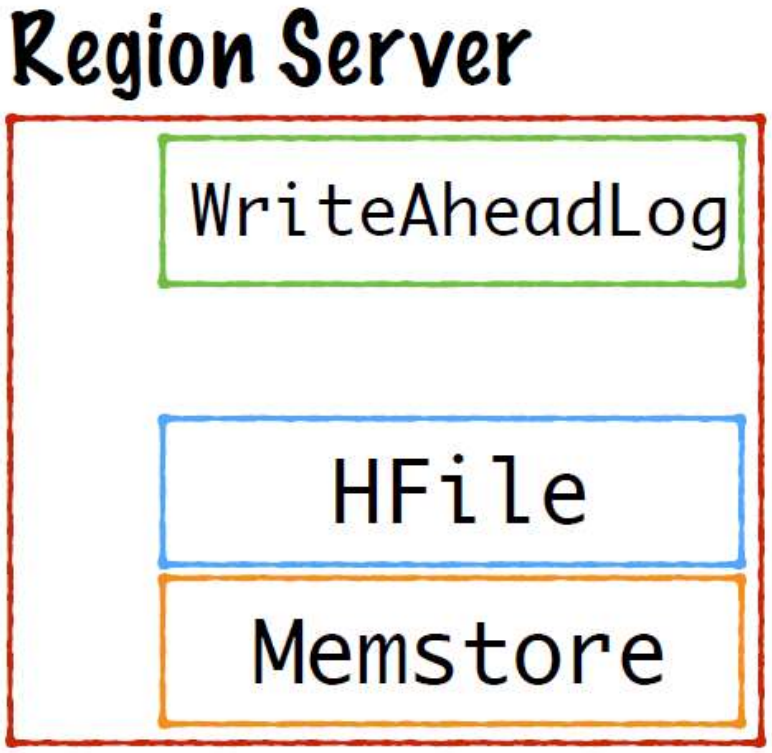
Hbase Architecture



Region Servers

The WriteAheadLog is created for recovery in case the Region Server crashes

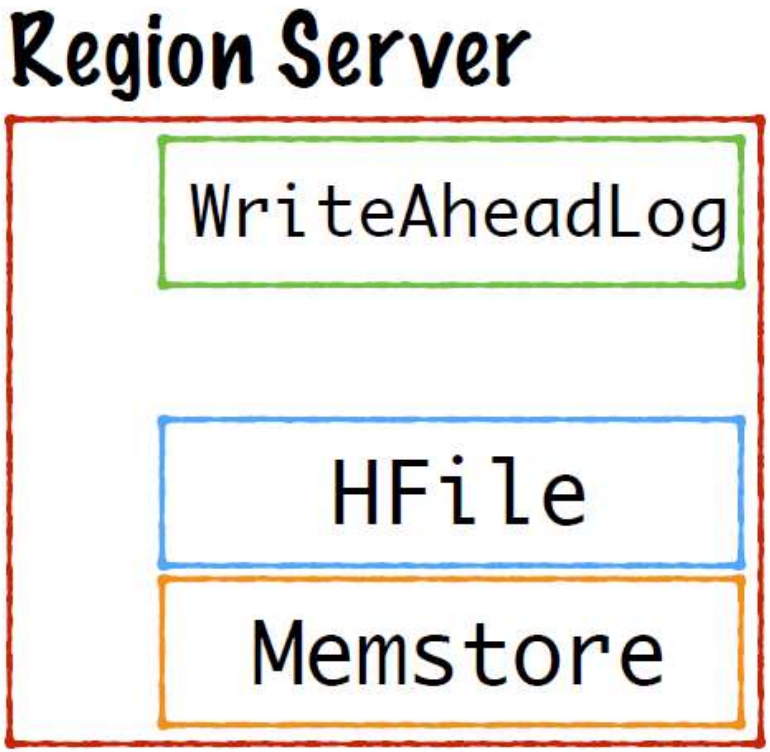
Hbase Architecture



Region Servers

Periodically the Memstore gets full, and the data in Memstore is flushed to disk

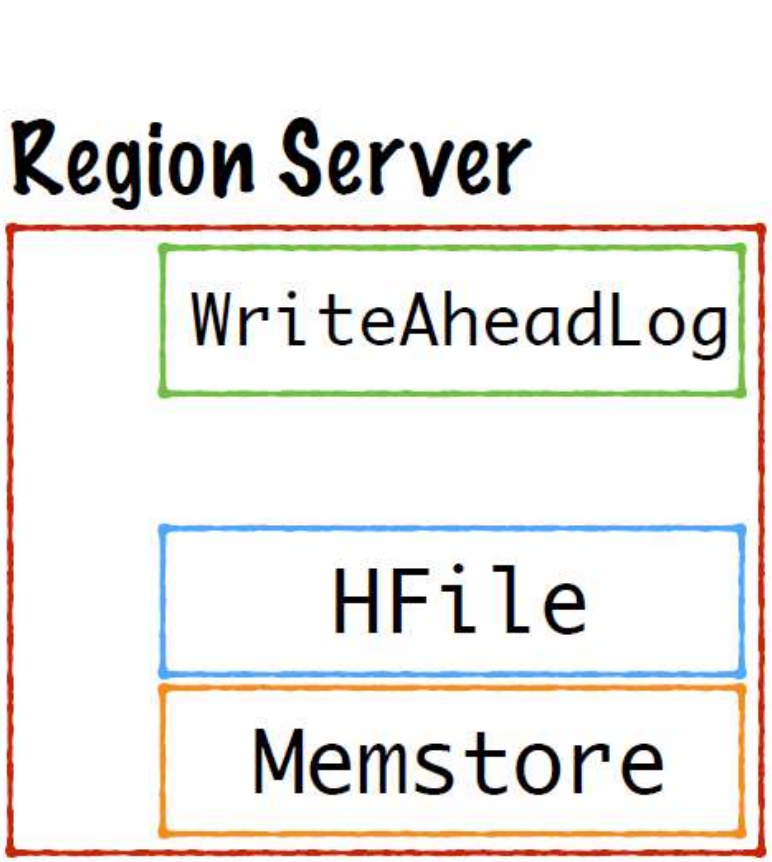
Hbase Architecture



Region Servers

The data for a row key is either in the Memstore or in a HFile

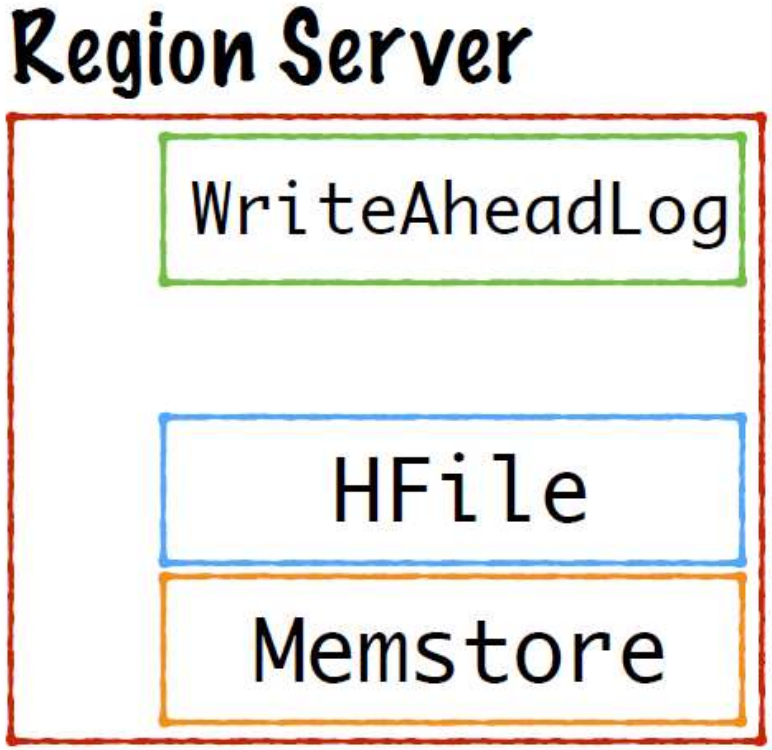
Hbase Architecture



Region Servers

HFiles are stored in HDFS

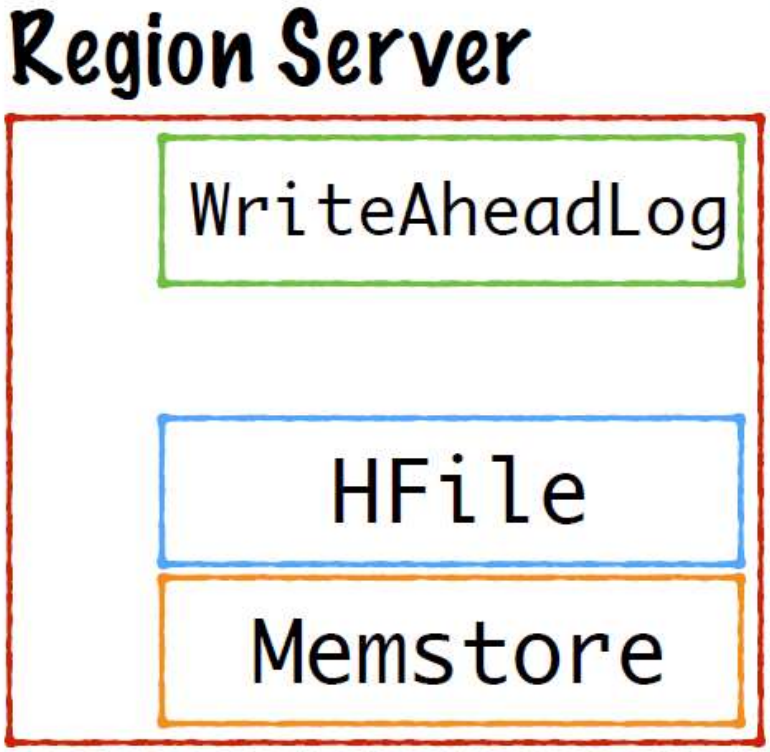
Hbase Architecture



Region Servers

HDFS will break up the HFile into blocks and store it on different nodes

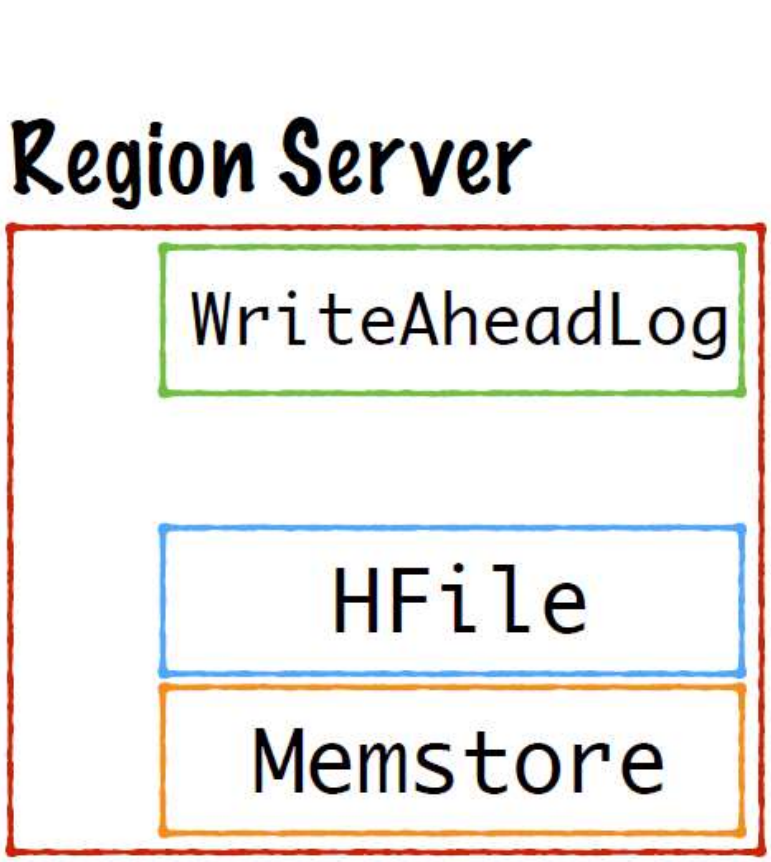
Hbase Architecture



Region Servers

To minimize disk seeks, the region server keeps an index of row key to HFile block in memory

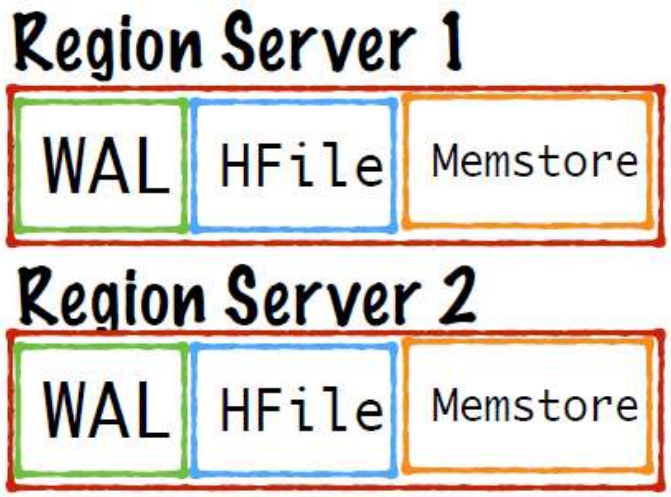
Hbase Architecture



Region Servers

It only performs 1 disk seek for finding a row key

Hbase Architecture

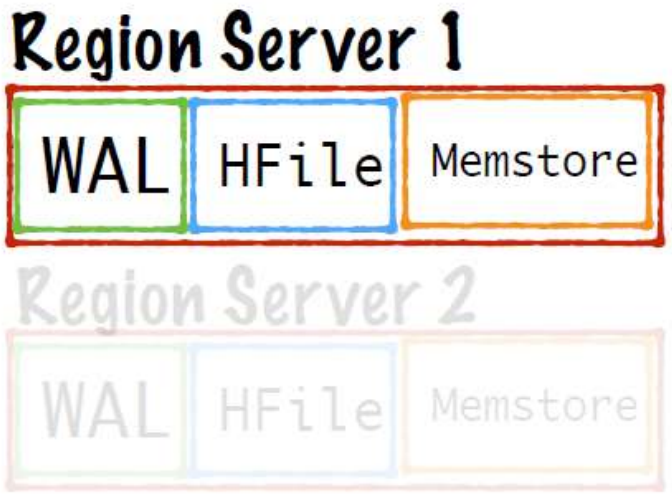


Region Servers

When you try to read/insert data

1. The region server containing the row key is identified

Hbase Architecture



Region Servers

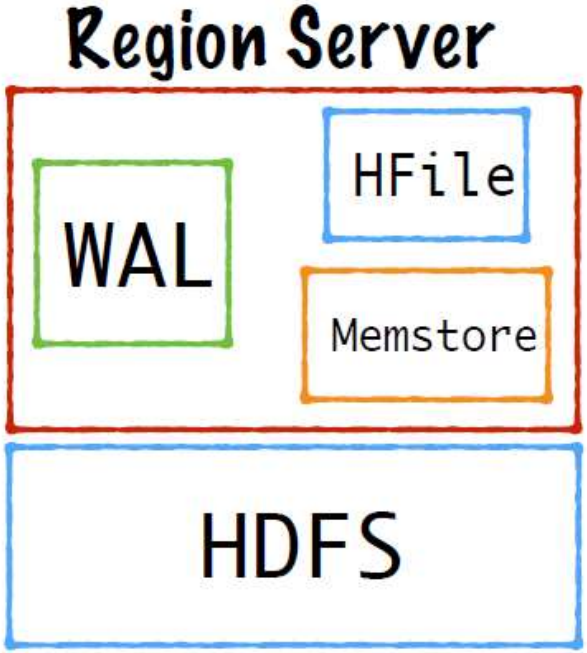
When you try to read/ insert data

1. The region server containing the row key is identified

2. The region server will lookup the Memstore or the HFile and do the needful

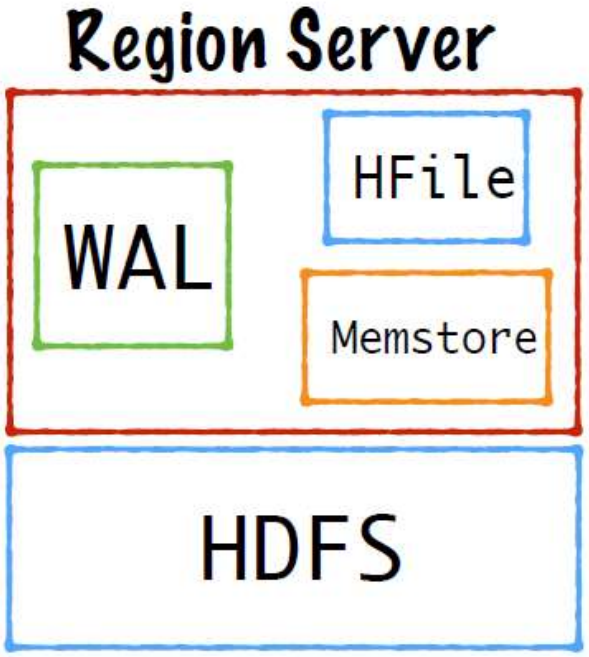
Hbase Architecture

Clients interact directly with a Region server handling the relevant row keys



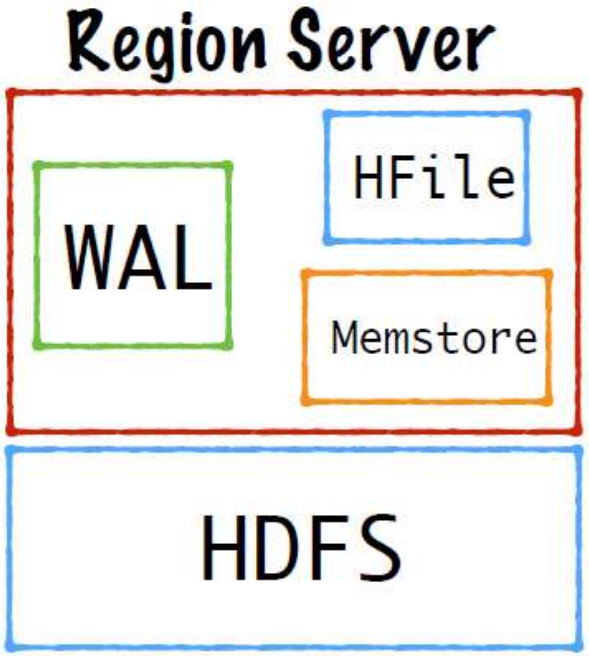
Hbase Architecture

They need to know which region server their row key is being handled by

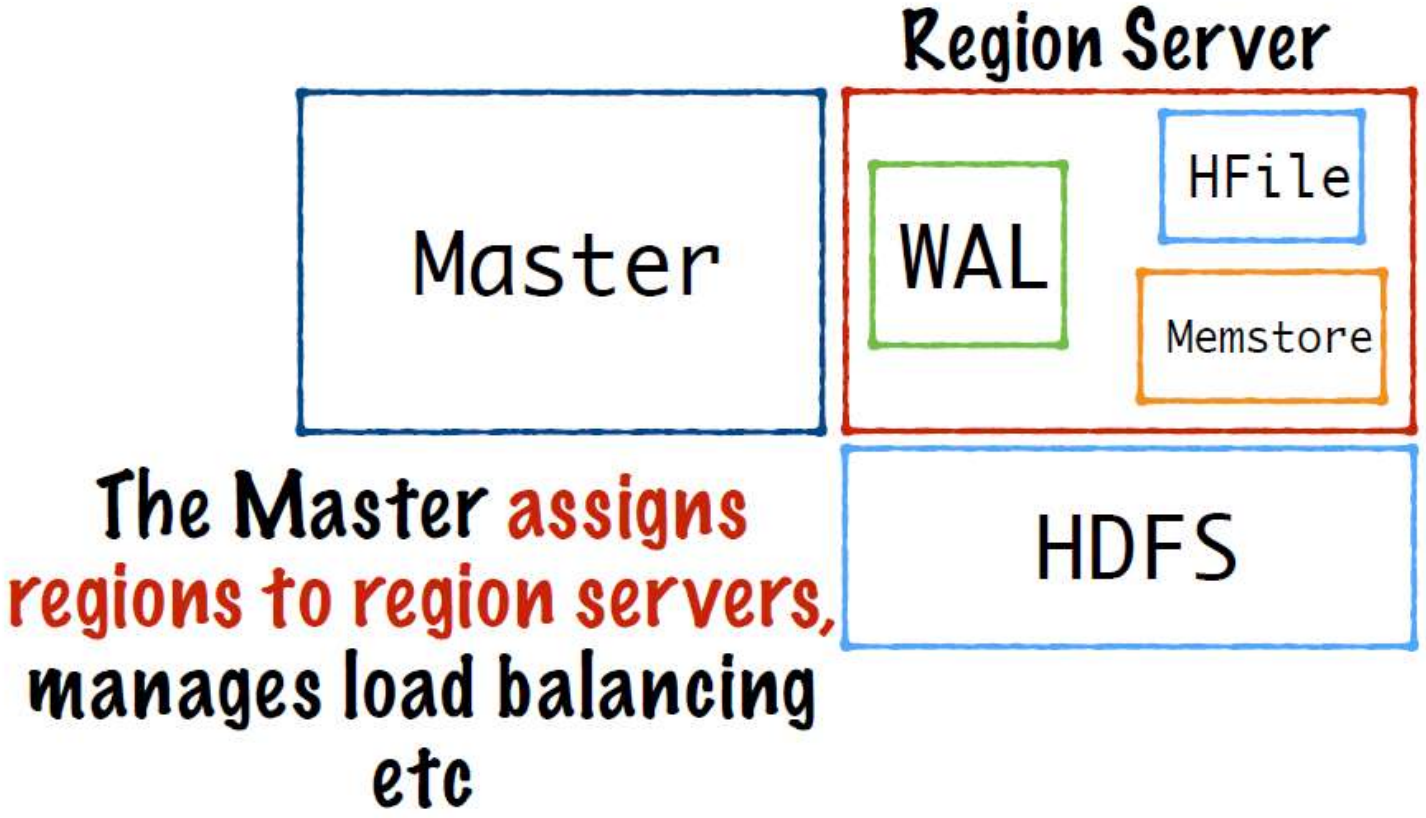


Hbase Architecture

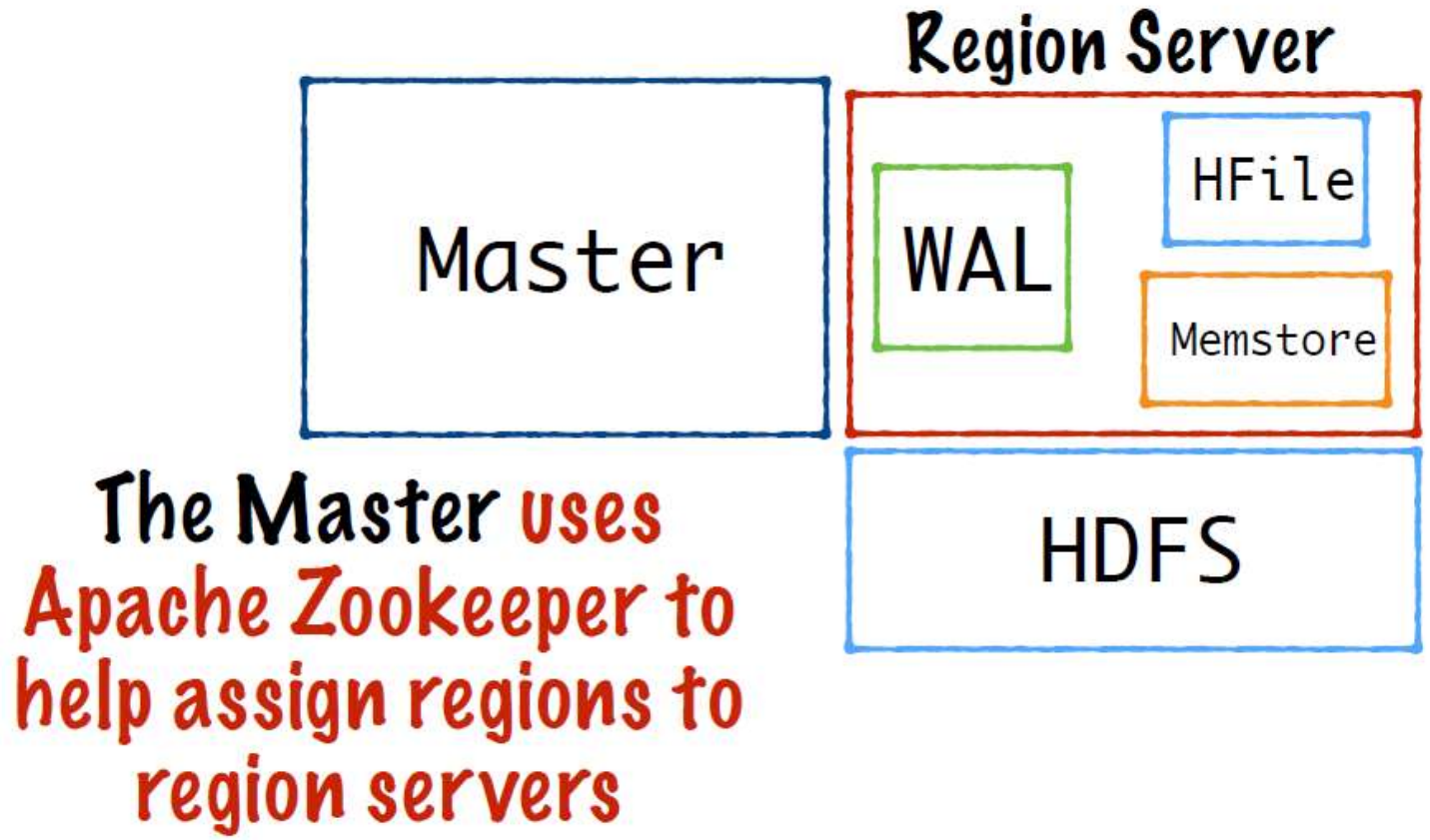
HBase uses a Master server to manage Regions and RegionServers



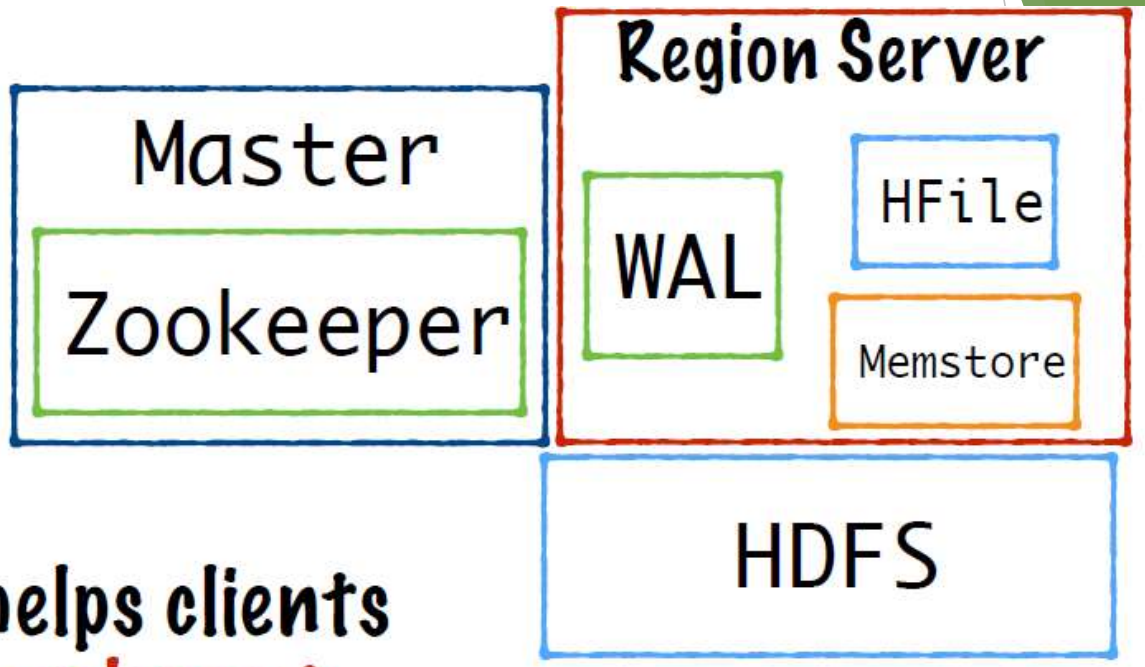
Hbase Architecture



Hbase Architecture



Hbase Architecture



Zookeeper helps clients lookup the relevant region server for a specific row id

Hbase Architecture