HBase Tutorial

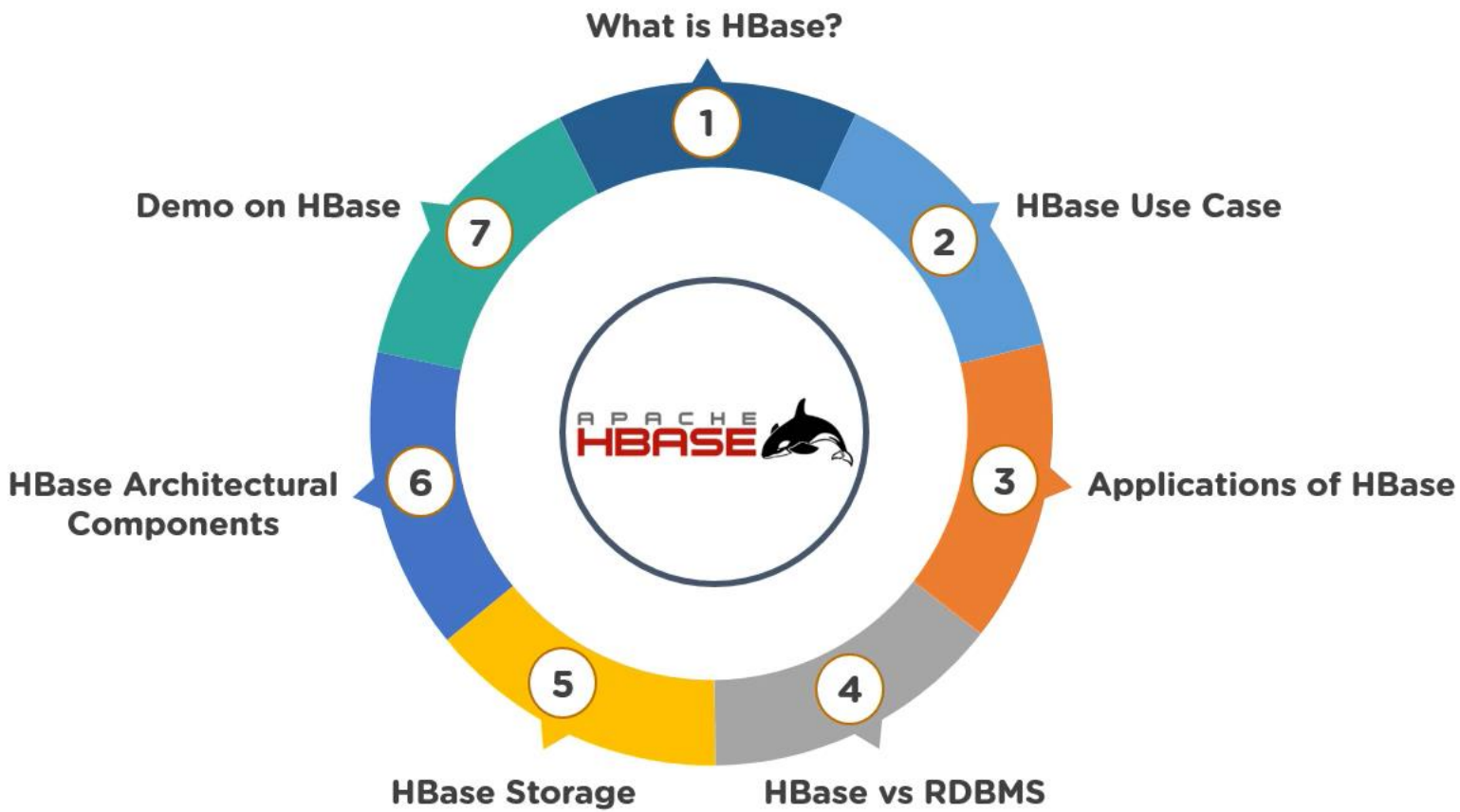
NoSQL

Database System

Data

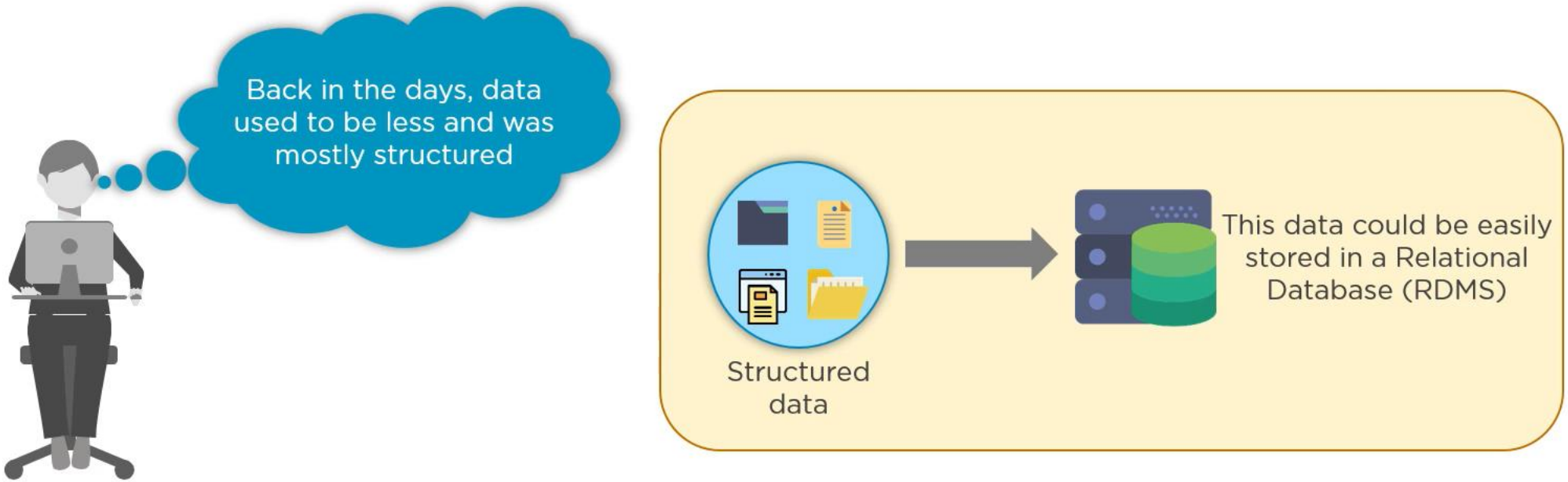
Bulk Data Load

What's in it for you?

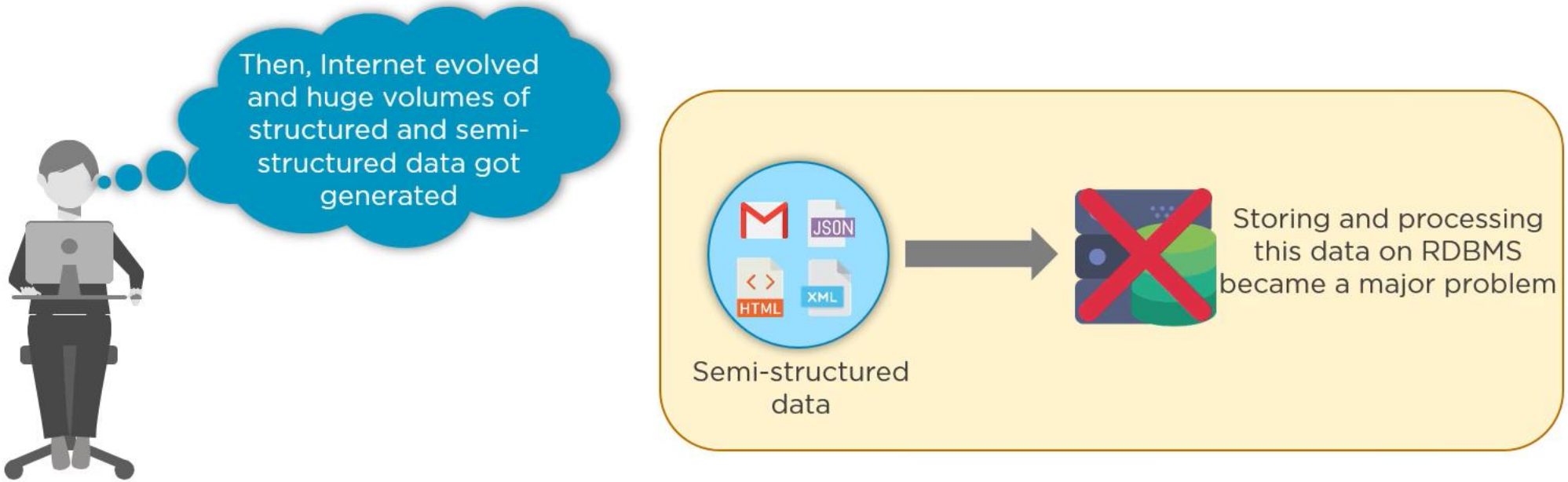


Introduction to HBase

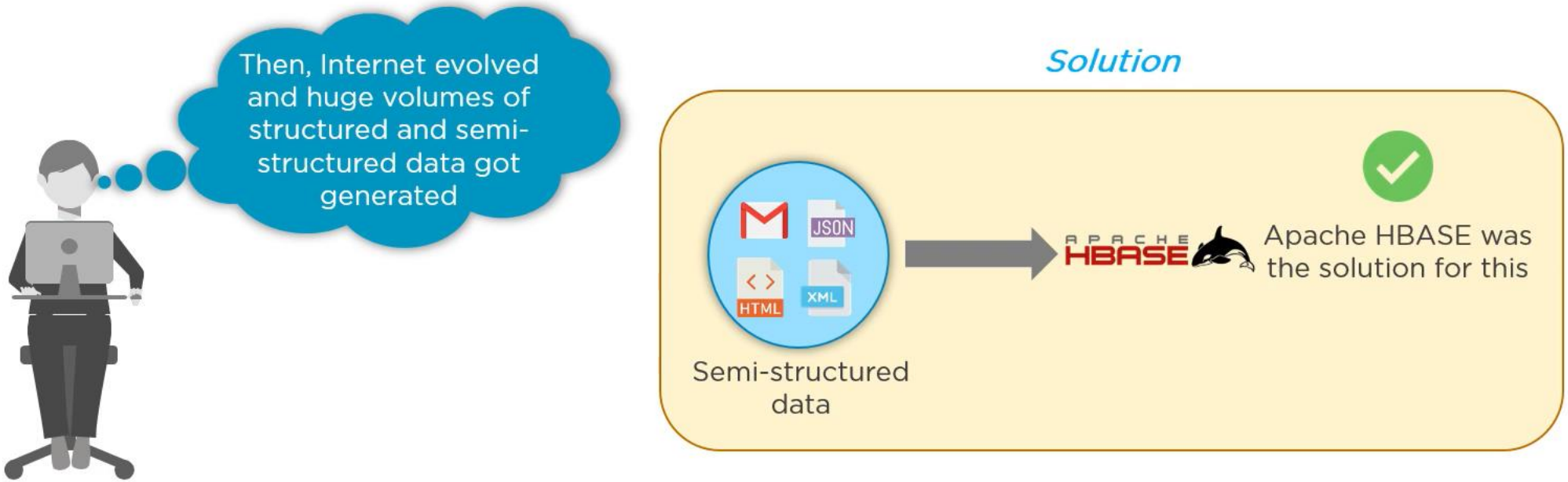
Introduction to HBase



Introduction to HBase

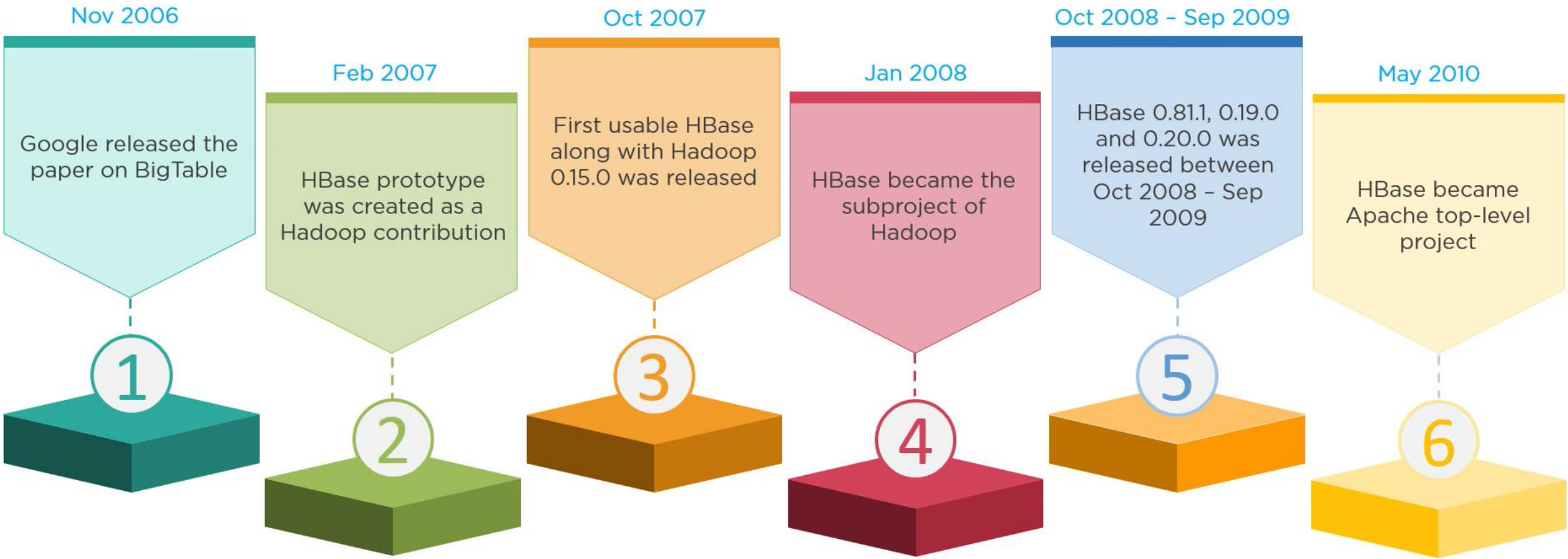


Introduction to HBase



HBase History

HBase History



What is HBase?

What is HBase?

HBase is a column oriented database management system derived from Google's NoSQL database BigTable that runs on top of HDFS

APACHE HBASE

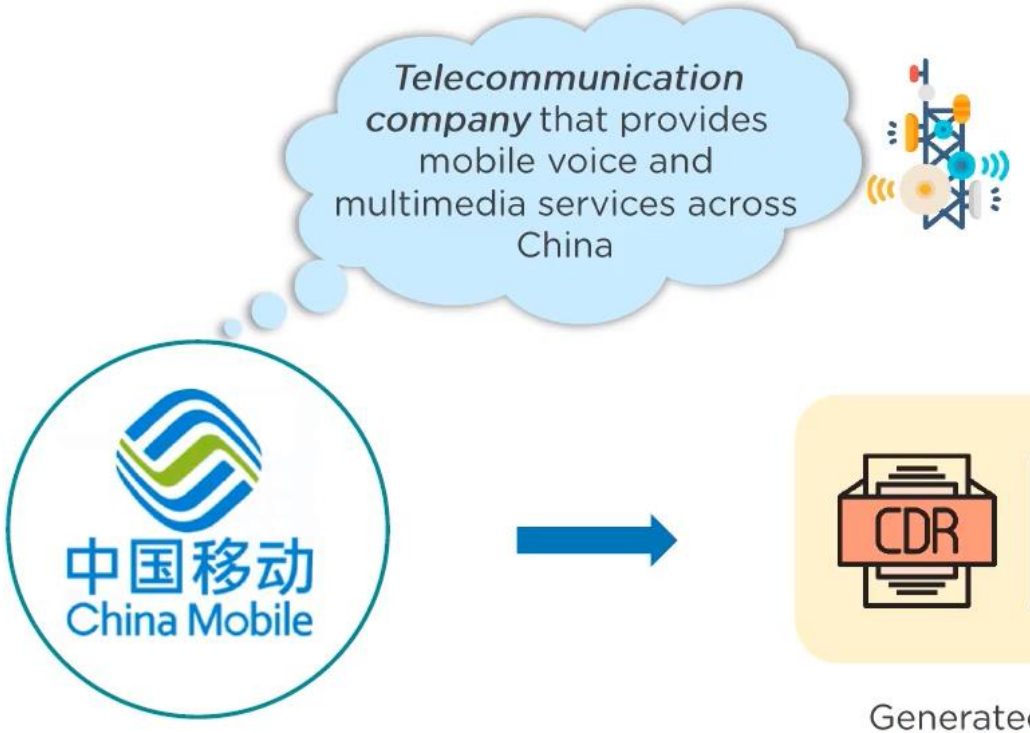
1. Open source project that is horizontally scalable
2. NoSQL database written in JAVA which performs faster querying
3. Well suited for sparse data sets (can contain missing or NA values)

Companies using HBase

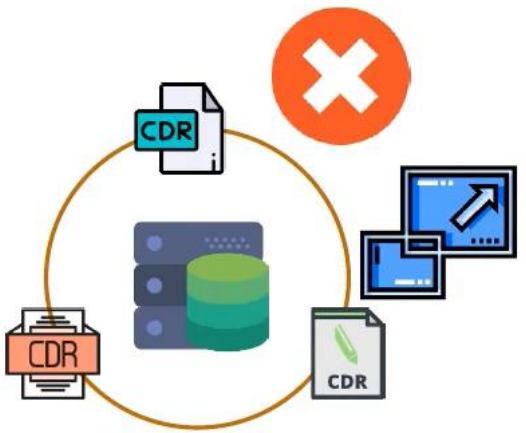
Companies using HBase

HBase Use Case

HBase Use Case



Generated billions of Call Detail Records (CDR)



Traditional database systems were unable to scale up to the vast volumes of data and provide a cost-effective solution

HBase Use Case

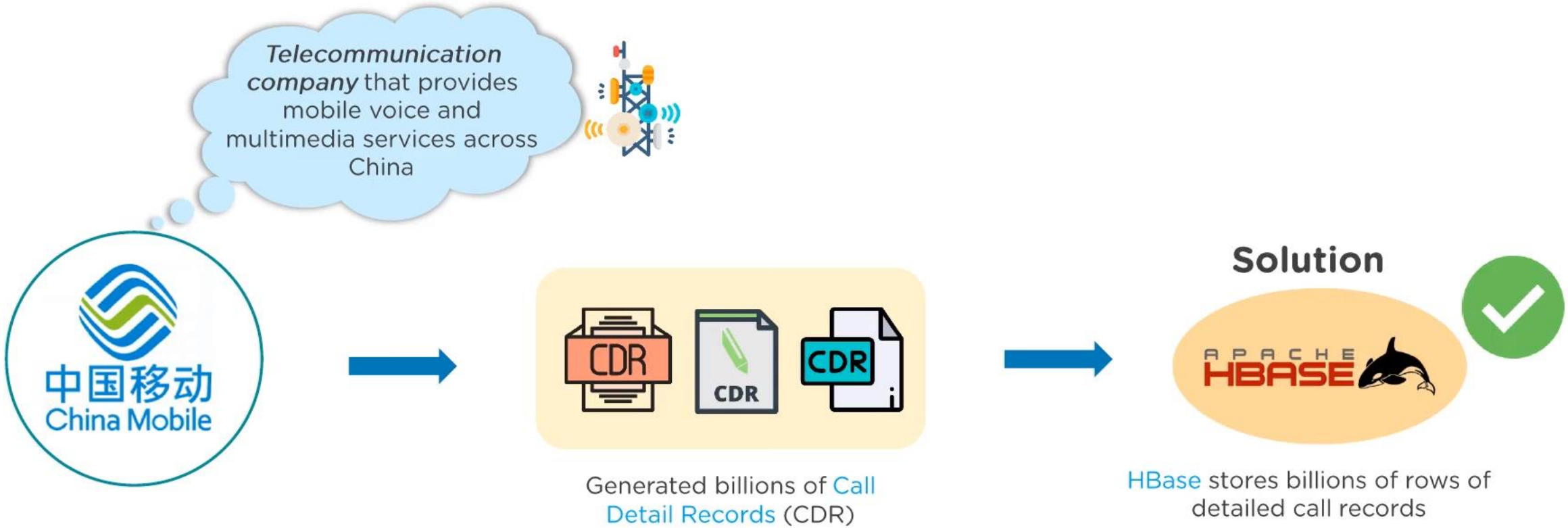
Telecommunication company that provides mobile voice and multimedia services across China

中国移动
China Mobile

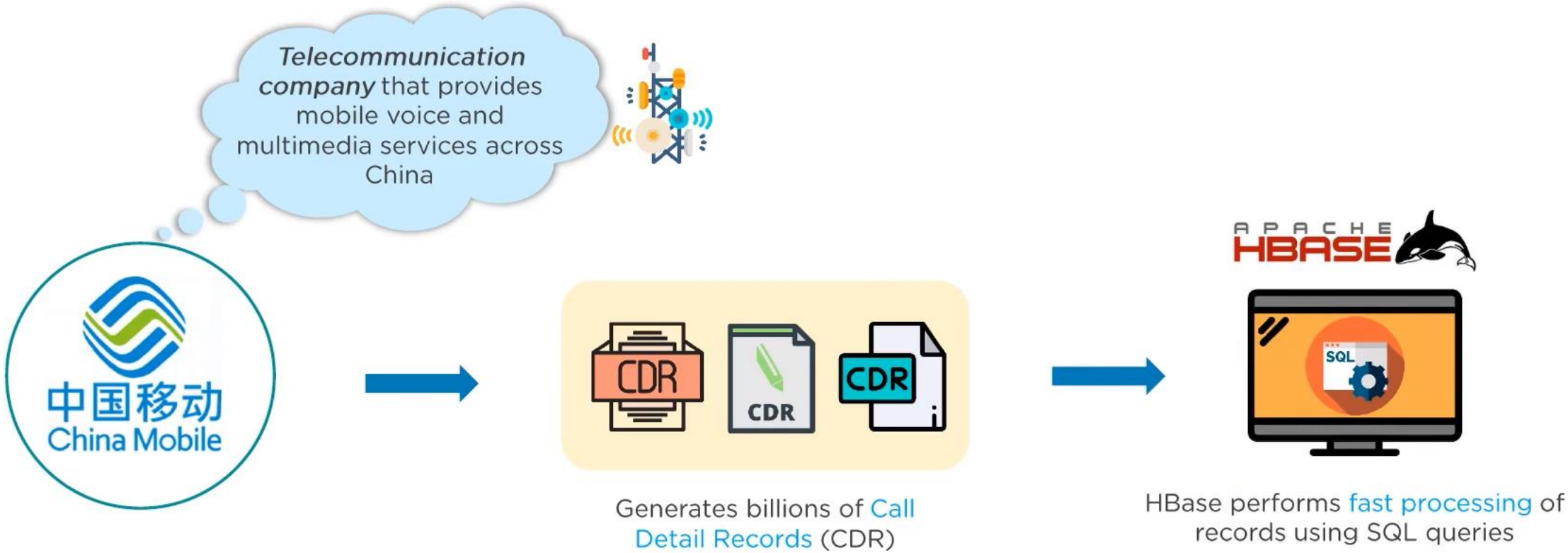
Generated billions of Call Detail Records (CDR)

Storing and real-time analysis of billions of call records was a major problem

HBase Use Case



HBase Use Case



Applications of HBase

Applications of HBase

Medical

HBase is used for storing genome sequences

Storing disease history of people or an area

E-Commerce

HBase is used for storing logs about customer search history

Performs analytics and target advertisement for better business insights

Sports

HBase stores match details and history of each match

Uses this data for better prediction

HBase vs RDBMS

HBase vs RDBMS

HBase	RDBMS
Does not have a fixed schema (schema-less). Defines only column families	Has a fixed schema which describes the structure of the tables
Works well with structured and semi-structured data	Works well with structured data
It can have de-normalized data (can contain missing or NA values)	RDBMS can store only normalized data
Built for wide tables that can be scaled horizontally	Built for thin tables that is hard to scale

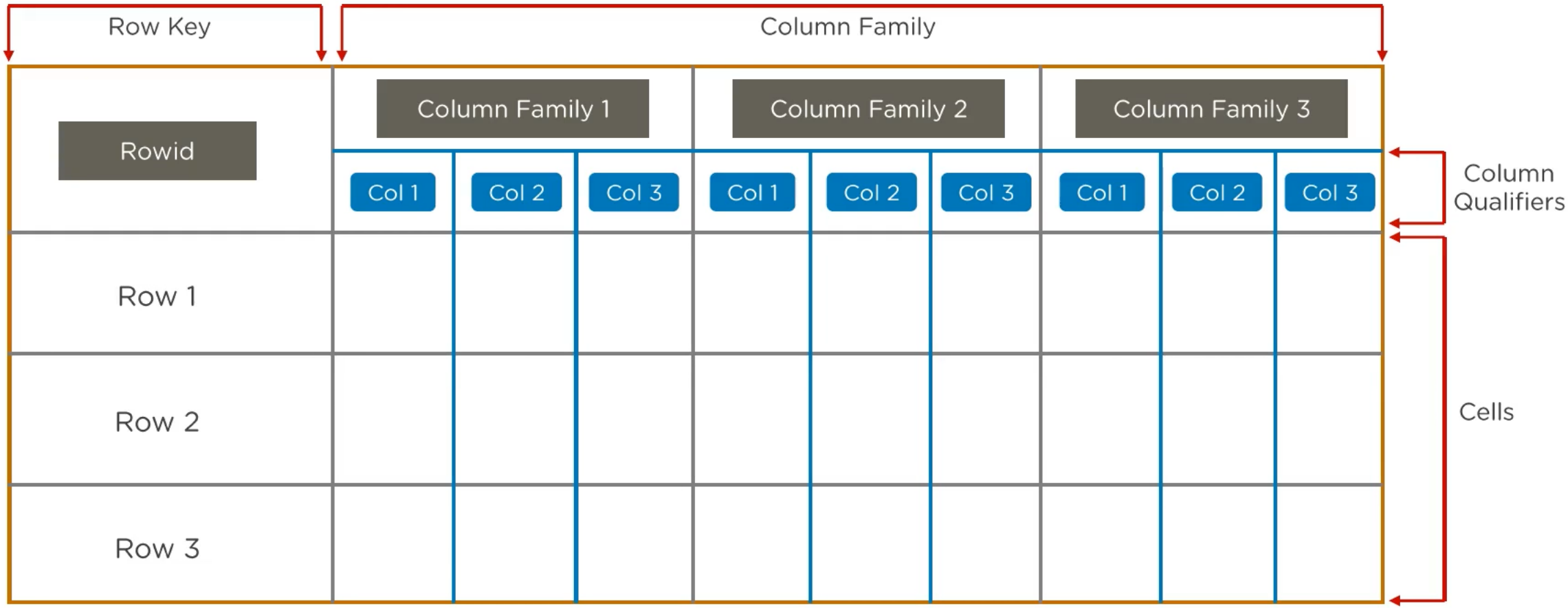
Features of HBase

Features of HBase

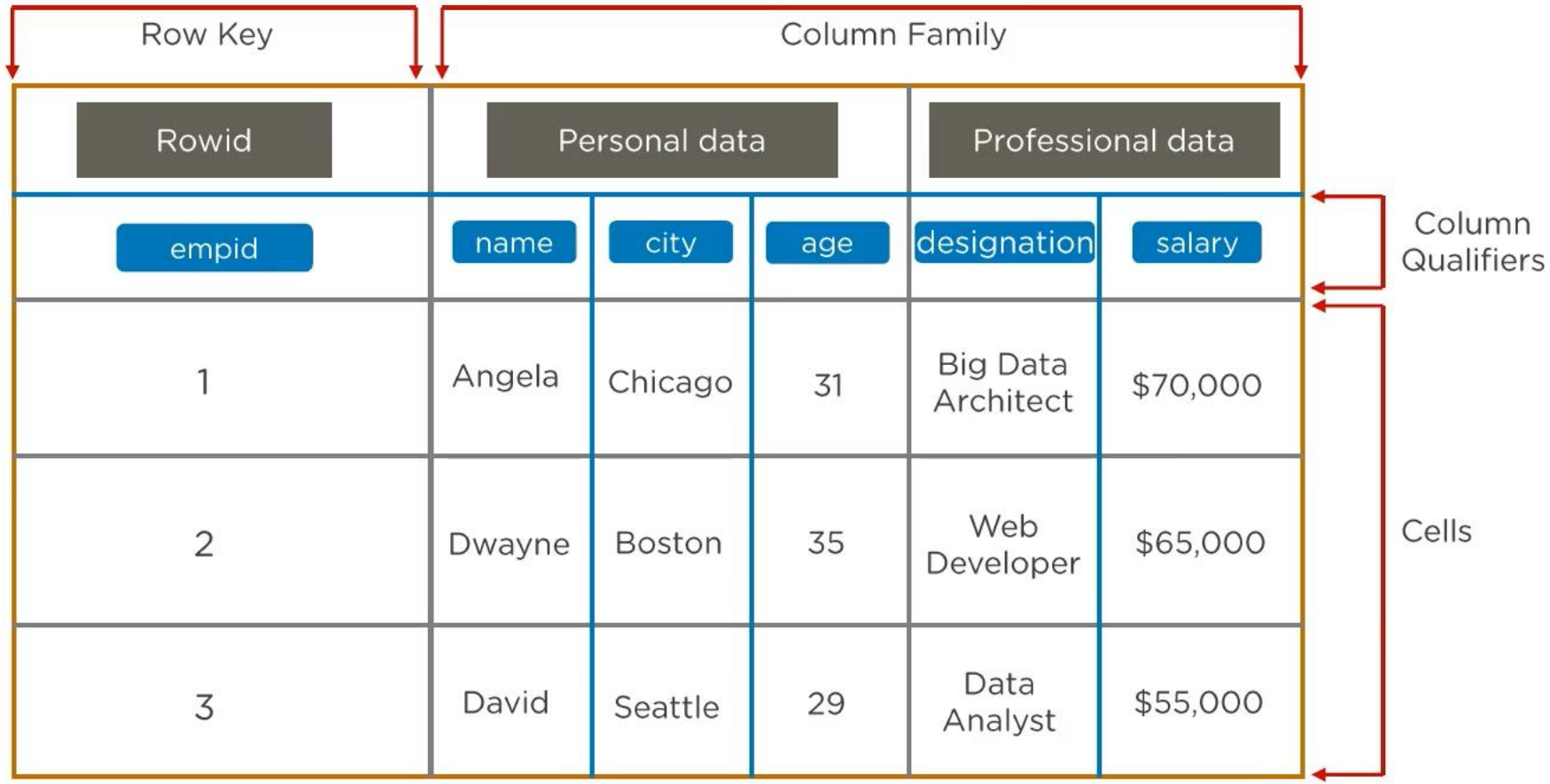
Scalable	Automatic failure support	Consistent read and write	JAVA API for client access	Block cache and bloom filters
				
Data can be scaled across various nodes as it is stored in HDFS	Write Ahead Log across clusters which provides automatic support against failure	HBase provides consistent read and write of data	Provides easy to use JAVA API for clients	Supports block cache and bloom filters for high volume query optimization

HBase Storage

HBase column oriented storage

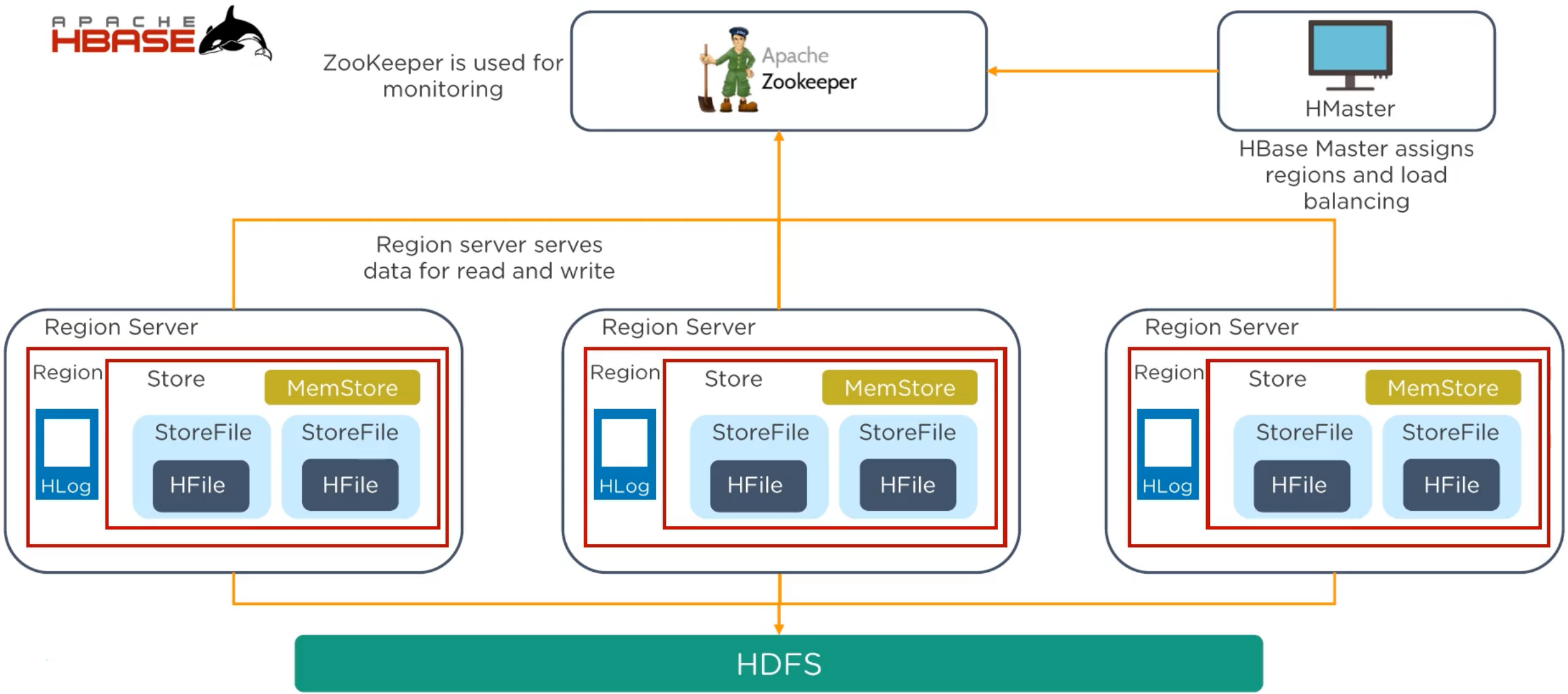


HBase column oriented storage



HBase Architecture

HBase Architectural Components



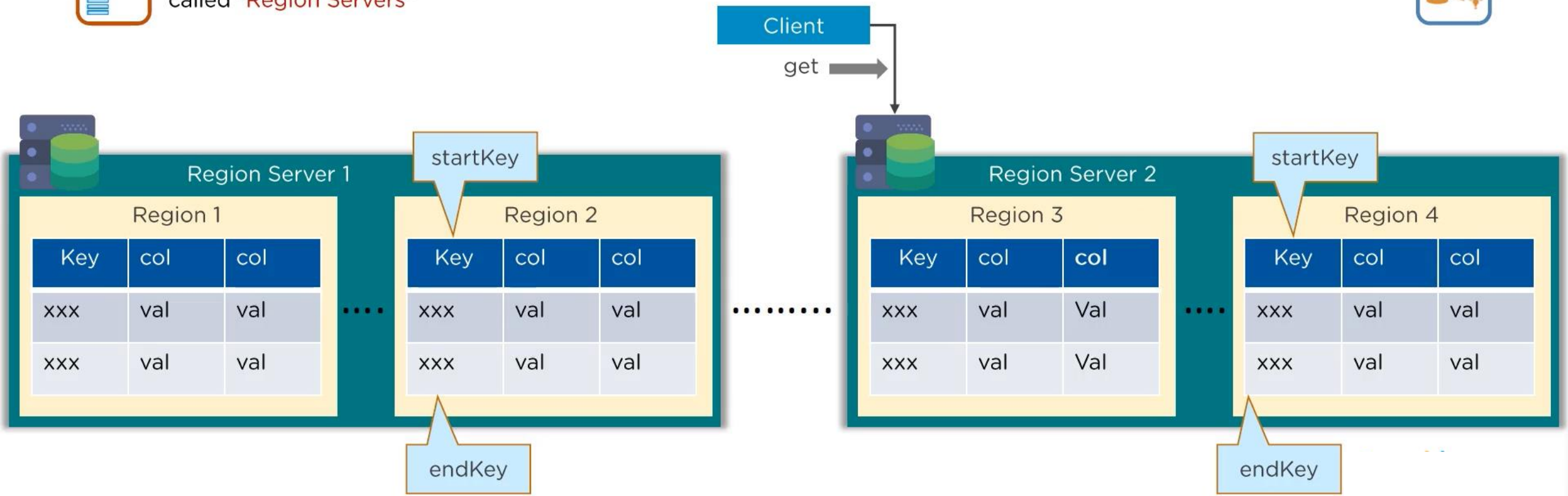
HBase Architectural Components - Regions

HBase tables are divided horizontally by row key range into "Regions"

A region contains all rows in the table between the region's start key and end key

Regions are assigned to the nodes in the cluster, called "Region Servers"

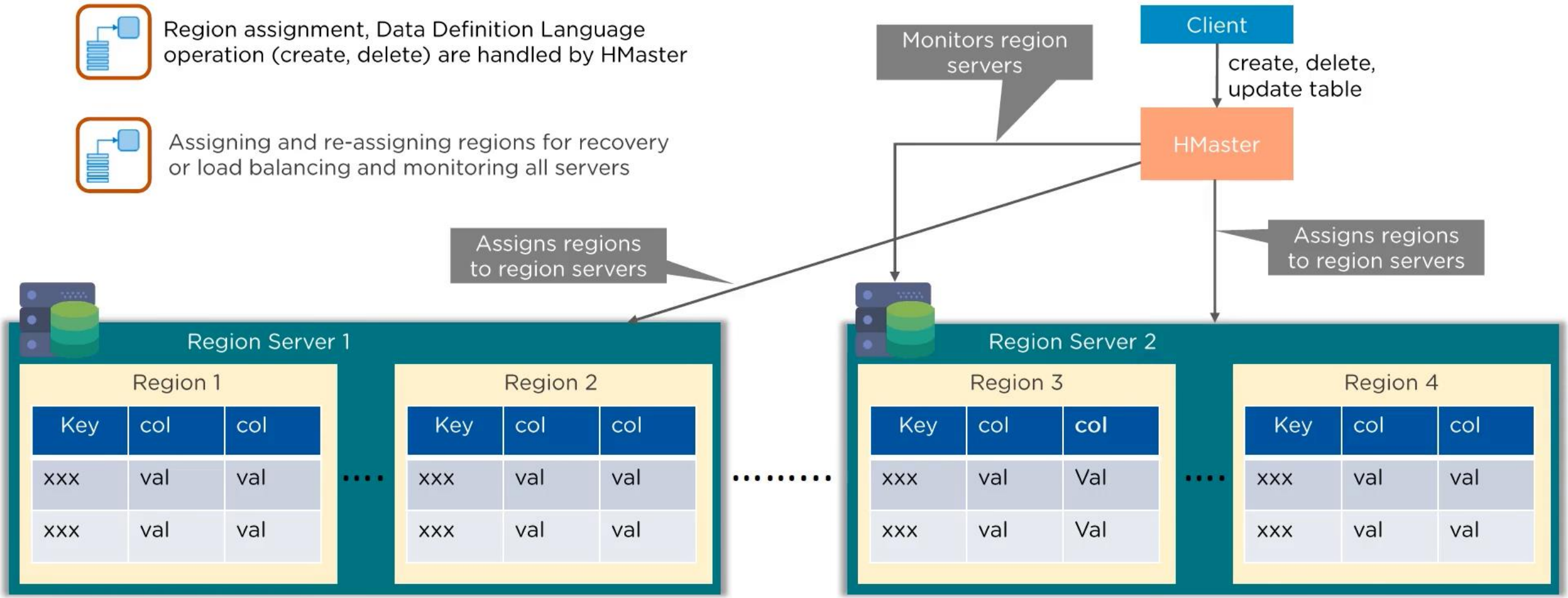
These servers serve data for read and write



HBase Architectural Components - HMaster

Region assignment, Data Definition Language operation (create, delete) are handled by HMaster

Assigning and re-assigning regions for recovery or load balancing and monitoring all servers

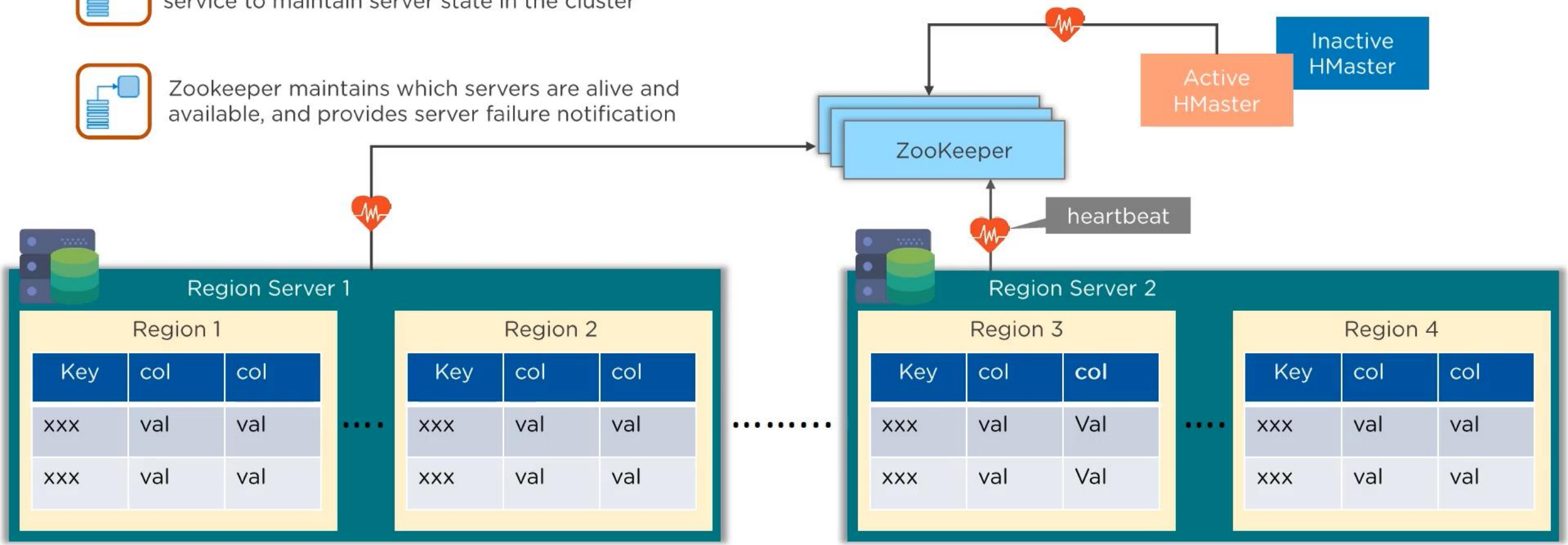


HBase has a distributed environment where HMaster alone is not sufficient to manage everything. Hence, *ZooKeeper* was introduced

HBase Architectural Components - ZooKeeper

ZooKeeper is a distributed coordination service to maintain server state in the cluster

Zookeeper maintains which servers are alive and available, and provides server failure notification

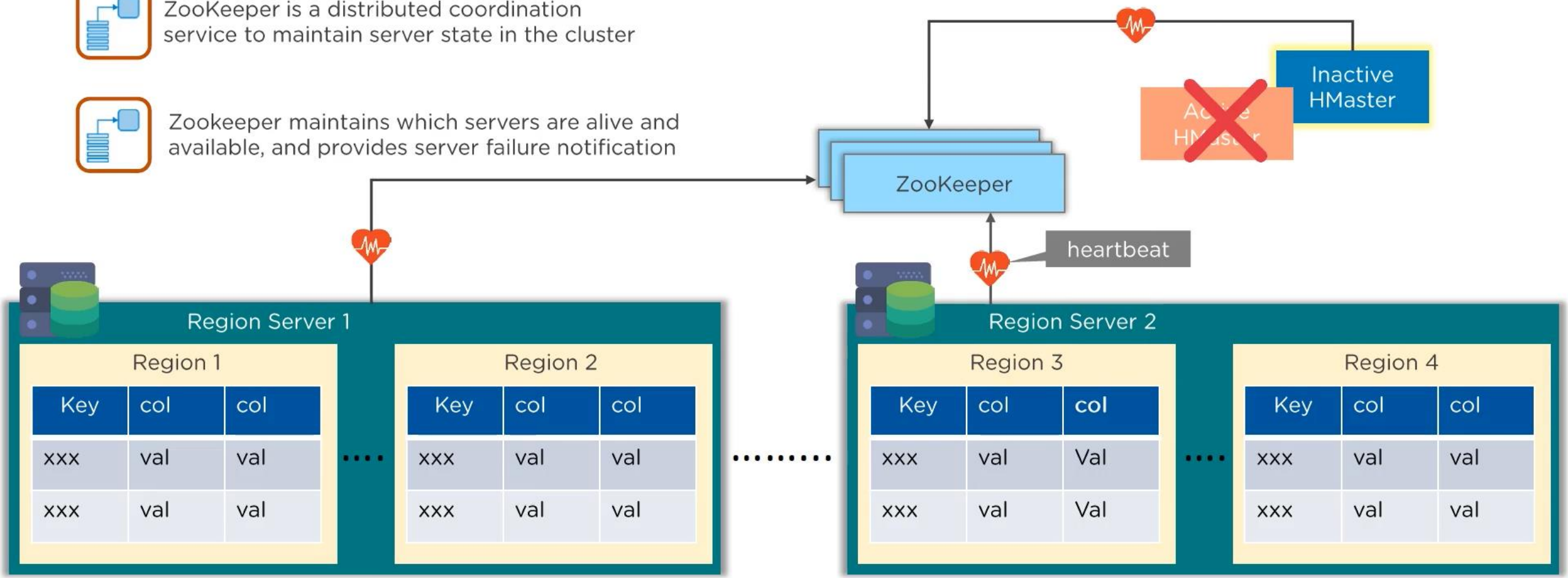


Region servers send their status to ZooKeeper indicating they are ready for read and write operation

HBase Architectural Components - ZooKeeper

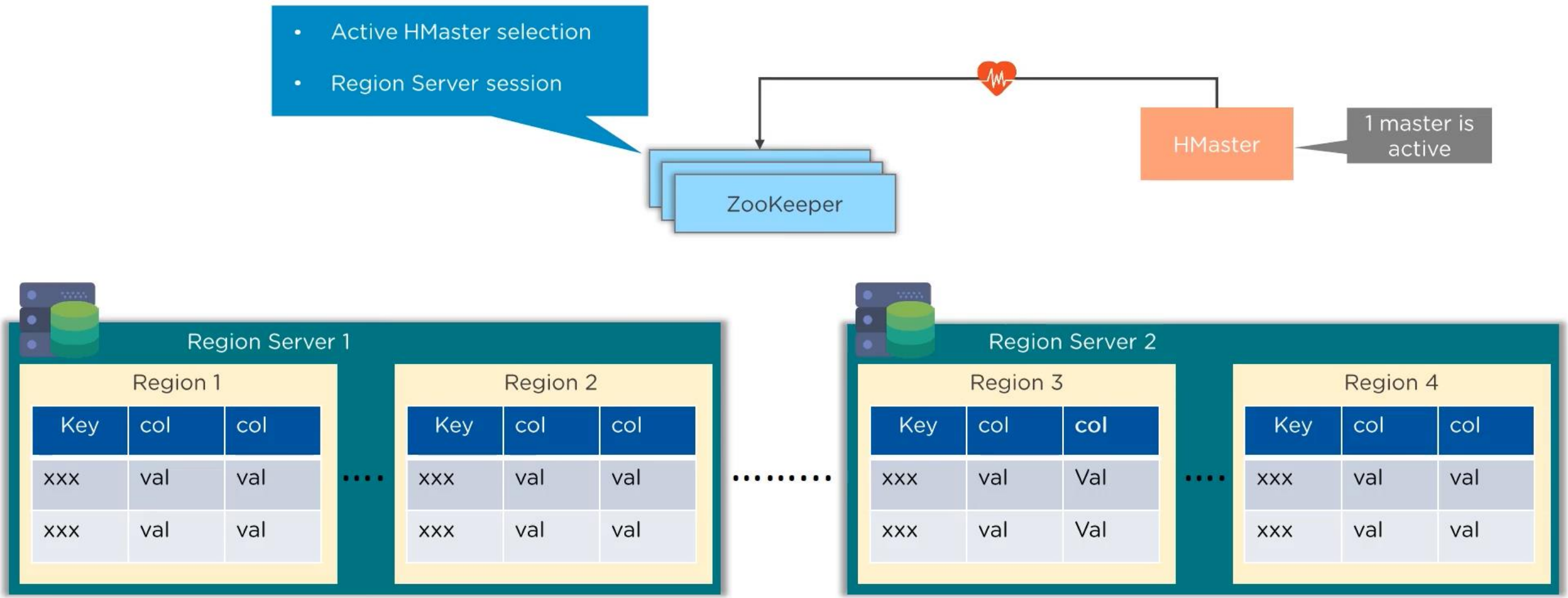
ZooKeeper is a distributed coordination service to maintain server state in the cluster

Zookeeper maintains which servers are alive and available, and provides server failure notification



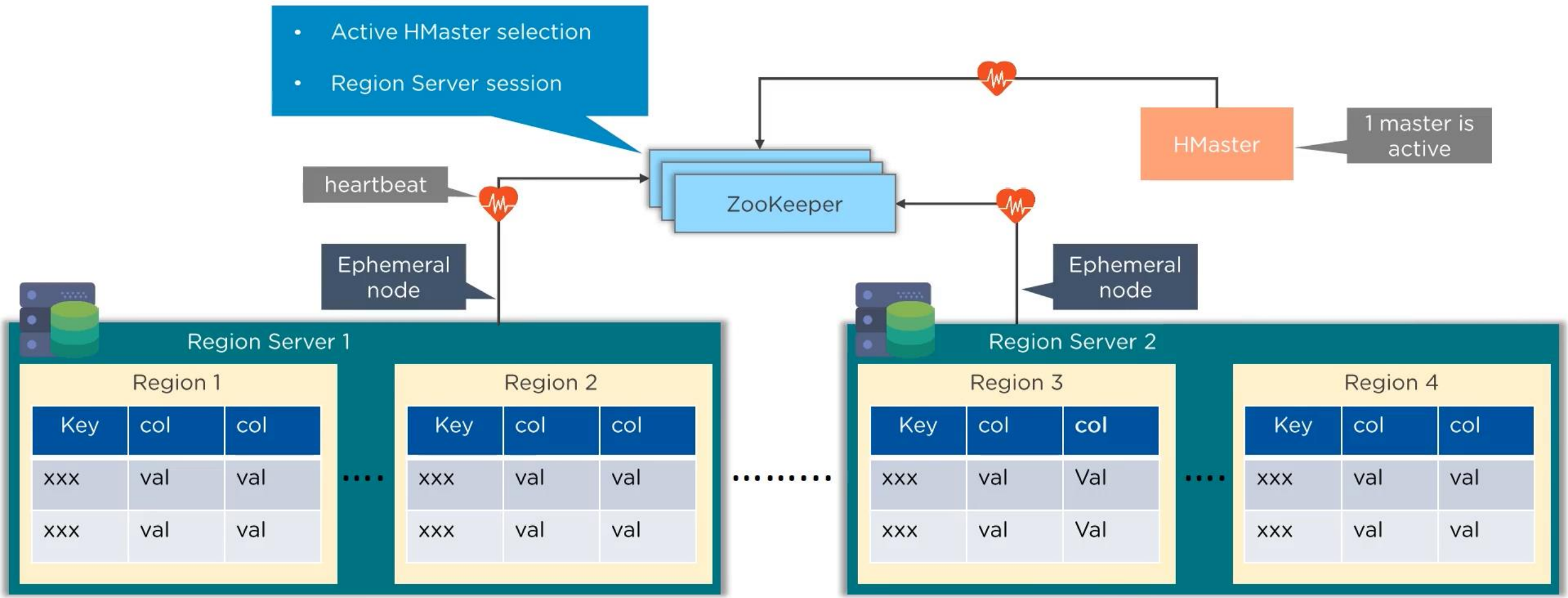
Inactive server acts as a backup. If the active HMaster fails, it will come to rescue

How the components work together?



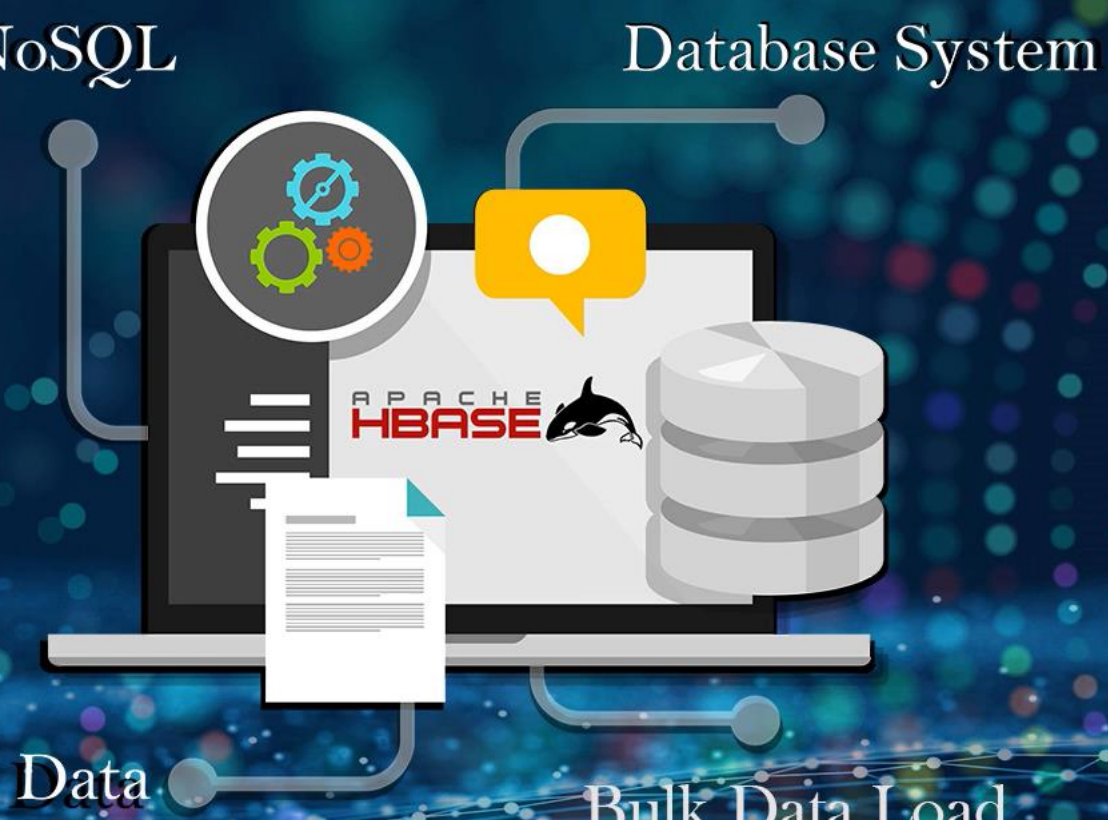
Active HMaster and Region Servers connect with a session to ZooKeeper

How the components work together?



ZooKeeper maintains ephemeral nodes for active sessions via heartbeats to indicate that region servers are up and running

HBase Read or Write

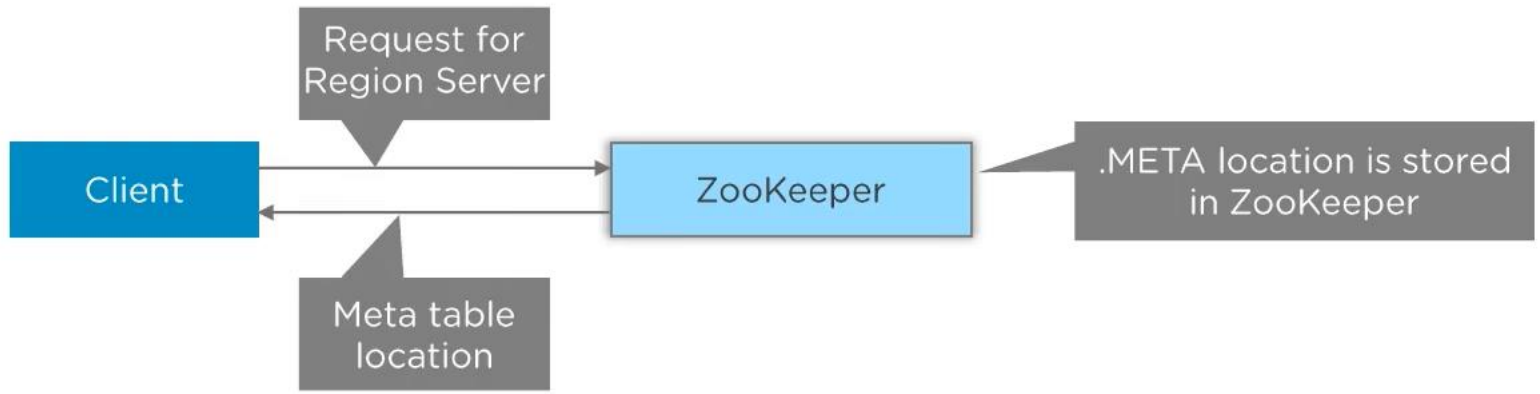


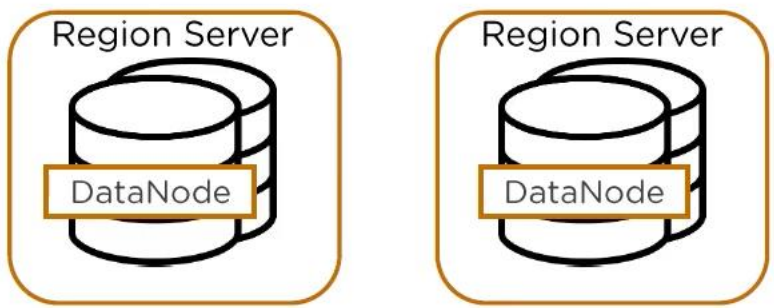
HBase Read or Write

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster

Here is what happens the first time a client reads or writes data to HBase

The client gets the Region Server that hosts the META table from ZooKeeper





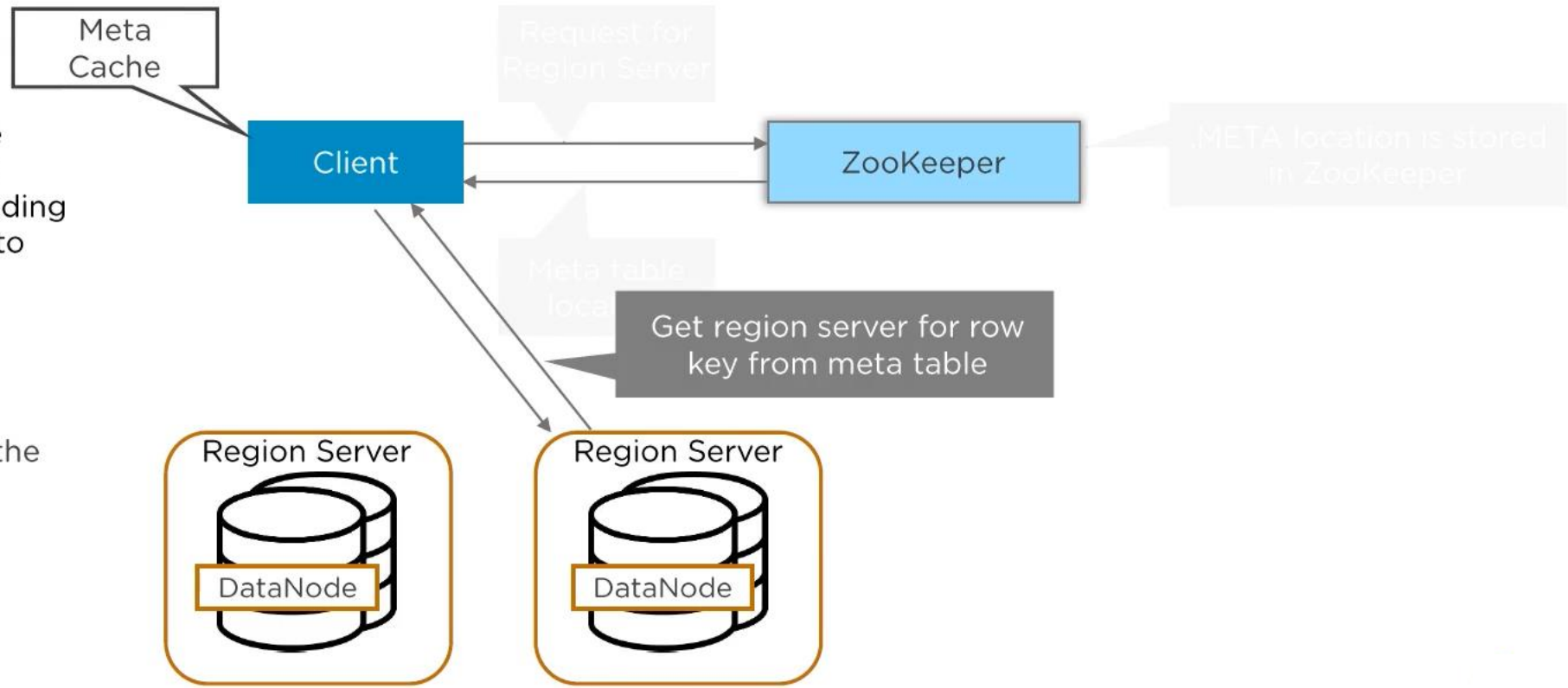
HBase Read or Write

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster

Here is what happens the first time a client reads or writes data to HBase

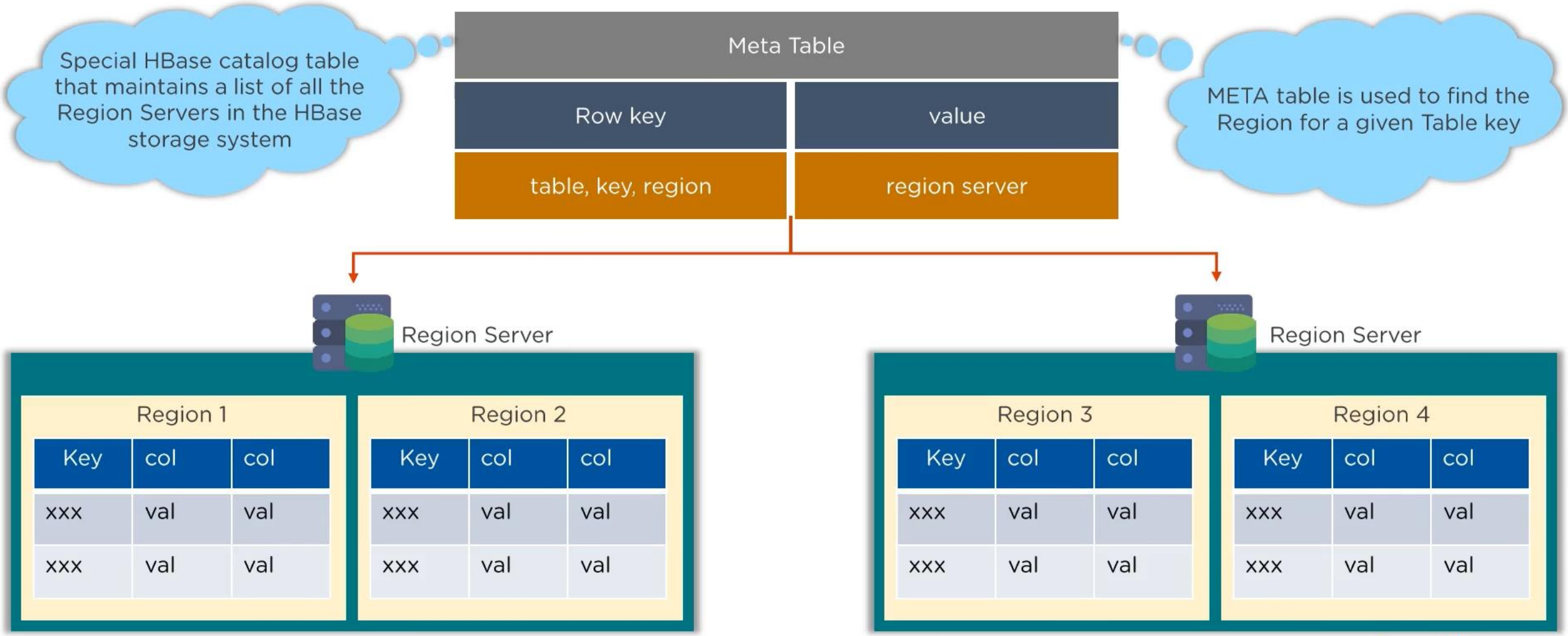
The client will query the .META server to get the region server corresponding to the row key it wants to access

The client caches this information along with the META table location



HBase Meta Table

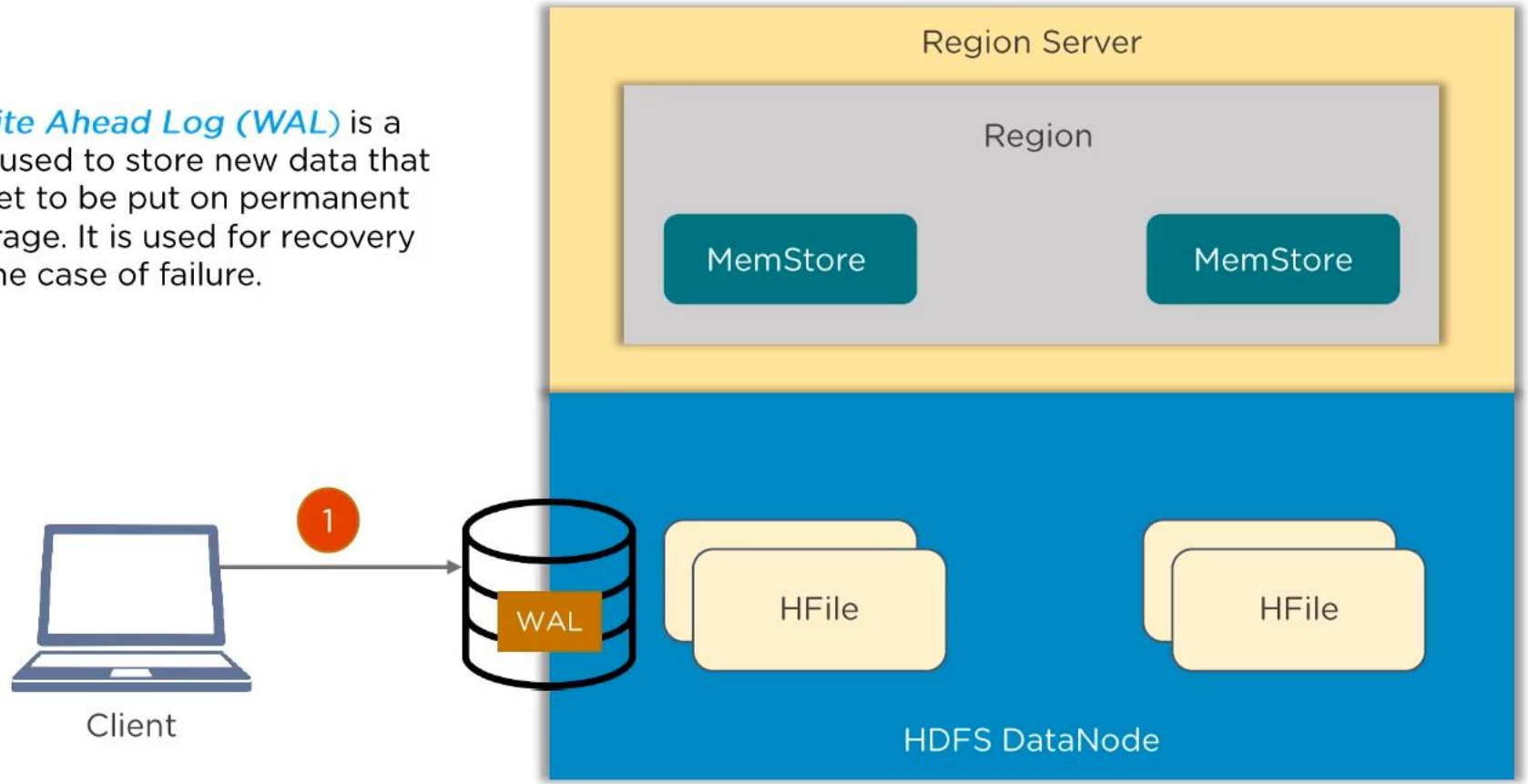
HBase Meta Table



HBase Write Mechanism

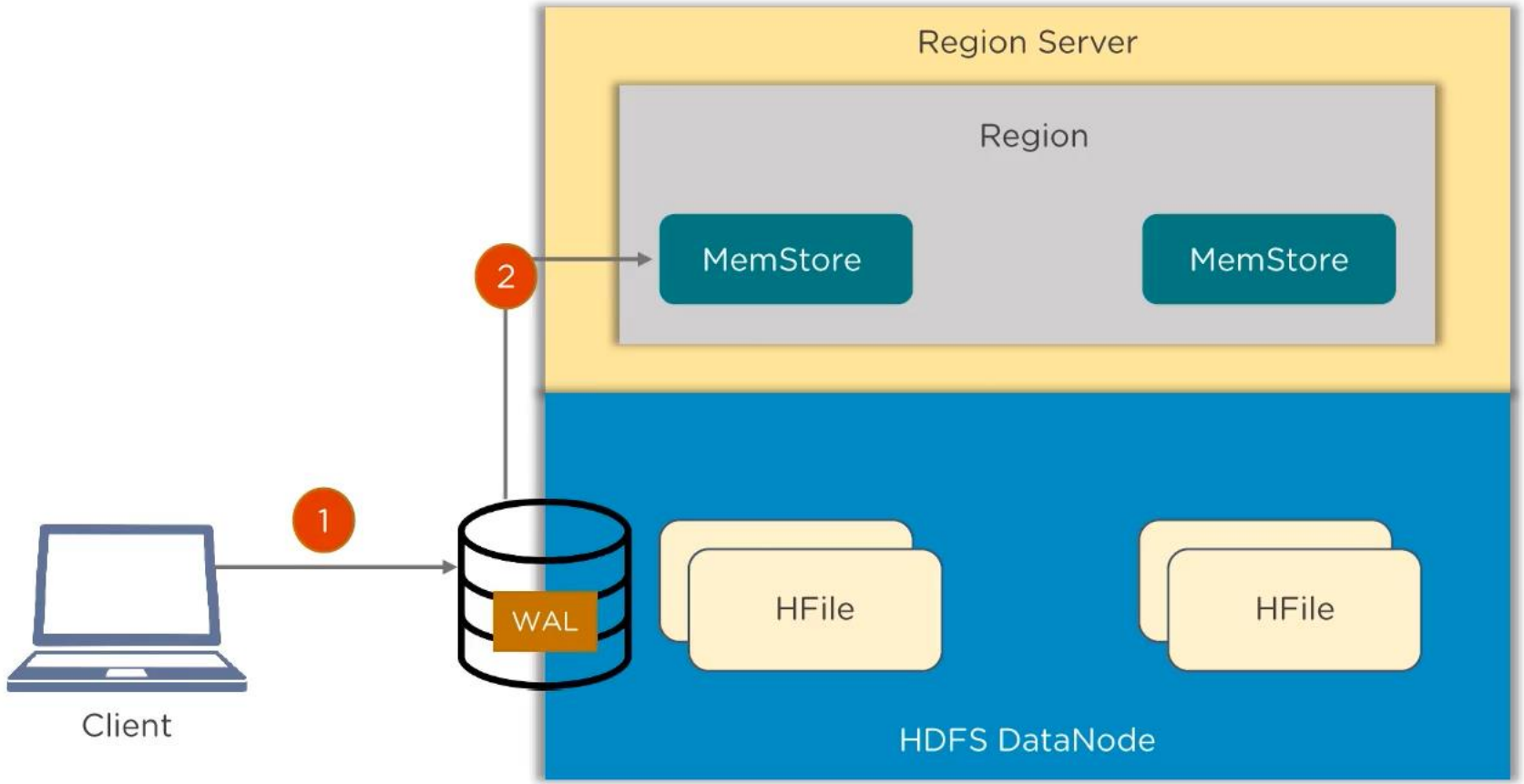
HBase Write Mechanism

Write Ahead Log (WAL) is a file used to store new data that is yet to be put on permanent storage. It is used for recovery is the case of failure.



1 When client issues a put request, it will write the data to the write-ahead log (WAL)

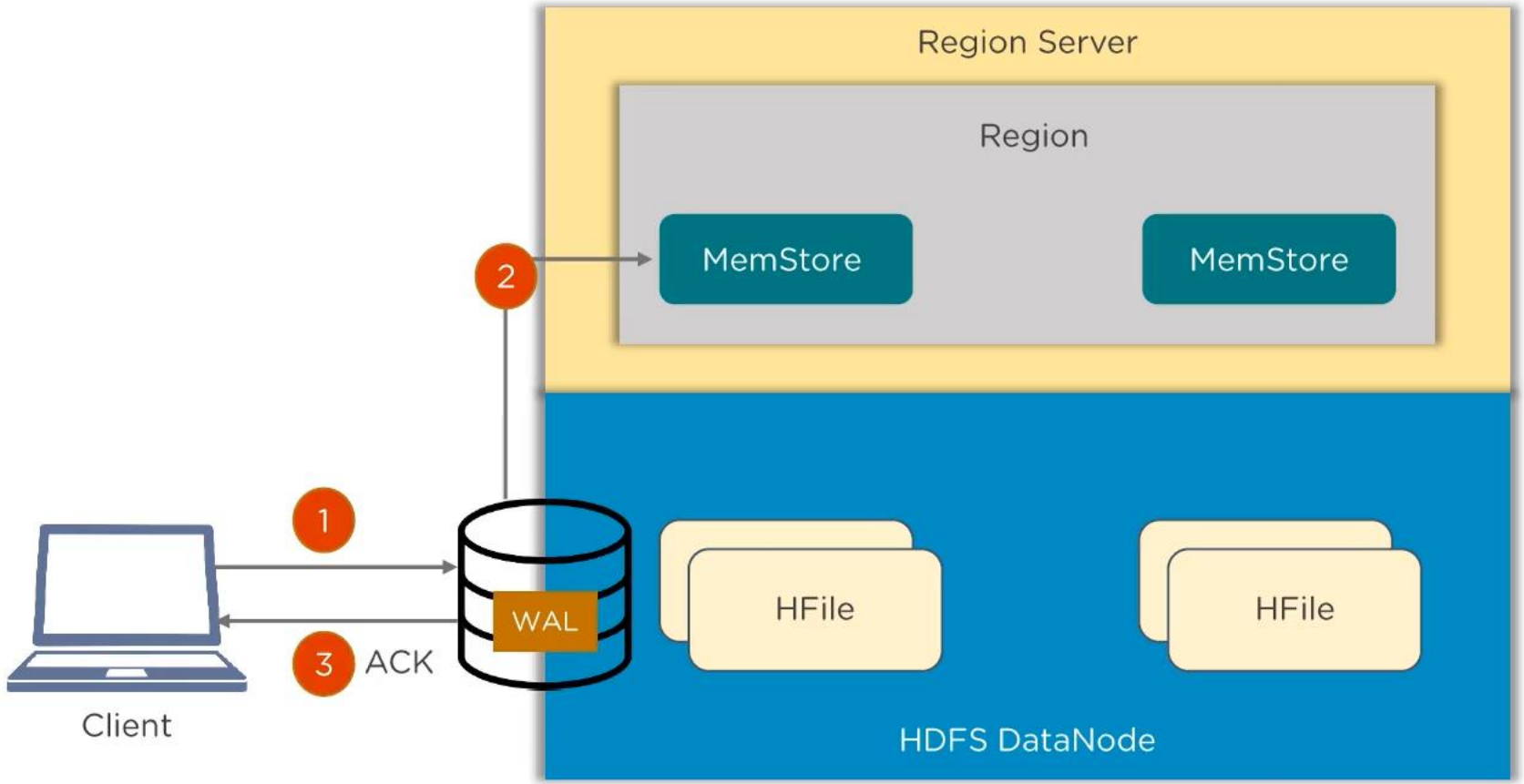
HBase Write Mechanism



MemStore is the write cache that stores new data that has not yet been written to disk. There is one MemStore per column family per region.

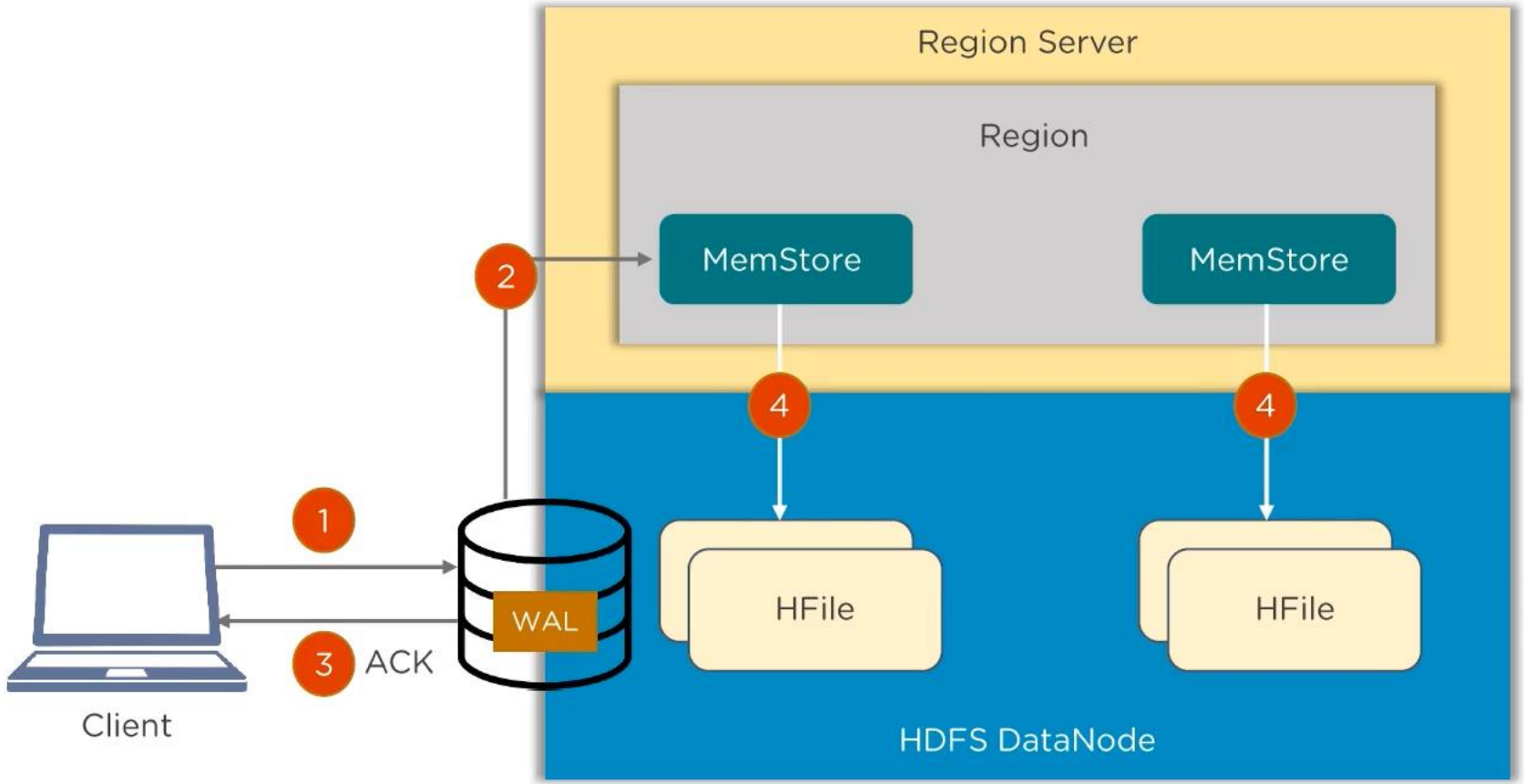
2 Once data is written to the WAL, it is then copied to the MemStore

HBase Write Mechanism



3 Once the data is placed in MemStore, the client then receives the acknowledgment

HBase Write Mechanism



Hfiles store the rows of data as sorted KeyValue on disk

4 When the MemStore reaches the threshold, it dumps or commits the data into a HFile

Demo on HBase

HBase - General Commands

- **Status**
  This command returns the status of the system including the details of the servers running on the system.
- **Version**
  This command returns the version of HBase used in your system
- **Table_help**
  This command guides you what and how to use table-referenced commands.
- **Whoami**
  This command returns the user details of HBase. If you execute this command, returns the current HBase user

Demo on HBase

Open cmd as Admin and write next commands:

```
cd \
cd hbase-2.3.7\bin
start-hbase.cmd
hbase shell
```

Demo on HBase

After a couple of seconds, you'll be inside the HBase shell where you can type the HBase commands. You can start off by typing the following commands:

```
list  // Lists down all the tables present in HBase
create 'newtbl', 'knowledge' // Creates a new table
list // Lists down all the tables present in HBase
describe 'newtbl' // Checks if the table was created
status 'summary' // Checks the status of HBase
```

Demo on HBase

Now that we have created a new table, let's put some data into it.

```
put  'newtbl', 'r1', 'knowledge:sports', 'cricket'
put  'newtbl', 'r1', 'knowledge:sciense', 'chemistry'
put  'newtbl', 'r1', 'knowledge:sciense', 'physics'
put  'newtbl', 'r2', 'knowledge:economics', 'macro economics'
put  'newtbl', 'r2', 'knowledge:music', 'hard rock'
```

Let's now list the contents of the table by typing:

```
scan 'newtbl' //
```

We cannot see "chemistry" as it will only display the latest update which in this case is "physics"

Demo on HBase

Now we can type the following commands:

```
is_enabled 'newtbl' // Checks if the table is enabled
disable 'newtbl' // Disables the table
scan 'newtbl' // Lists the contents of the table.
 Note that this will throw an error as the table is disabled.
```

Now before we move and enable the table, let’s do an alteration on it.

```
alter 'newtbl', 'test_info' // Updates column family in the table
enable 'newtbl' // Enables the table
describe 'newtbl' //Checks the column families after updating
scan 'newtbl'
```

Demo on HBase

Then extract values for one particular row and also see how to add new information to a row by using the following commands:

```
get 'newtbl', 'r1' // Extracts the values for r1 in the table


put  'newtbl', 'r1', 'knowledge:economics', 'macro economics' // /Adds
new                                                    information to r1 for
economics.
Note that this will update the table but will not override the information


get 'newtbl', 'r1' // Displays the results for r1
```

Demo on HBase

Using the **delete** command, you can delete a specific cell in a table.

```
scan 'newtbl'
delete 'newtbl', 'r2', 'knowledge:music' // Here is an example to delete a specific cell
deleteall 'newtbl', 'r1' // You can delete all the cells in a row
scan 'newtbl'
```