Sohail Akhtar
CS Department
Bahria University, Islamabad Campus

Provided by:
Dr Muhammad Rashid Hussain

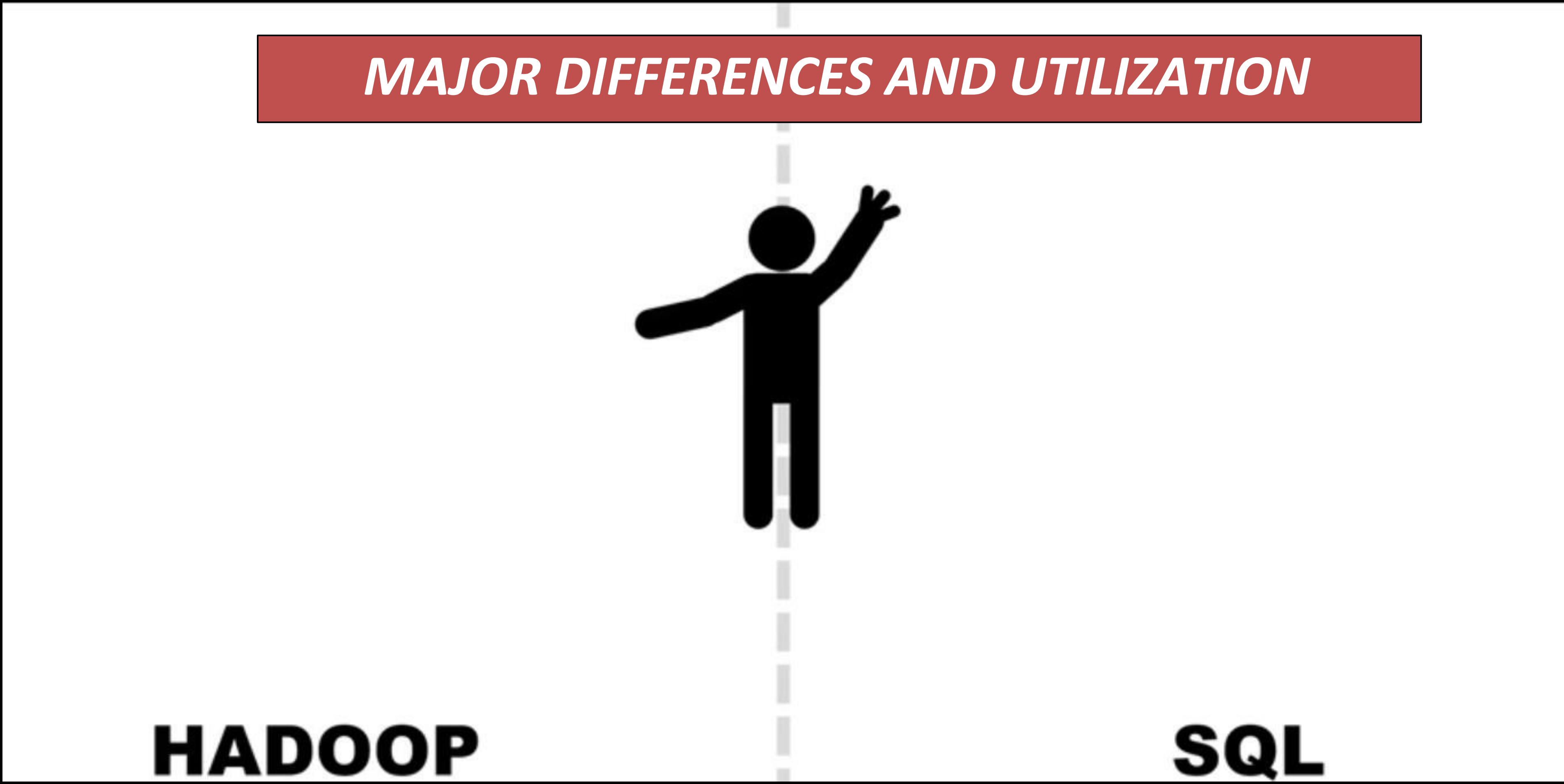
Agenda

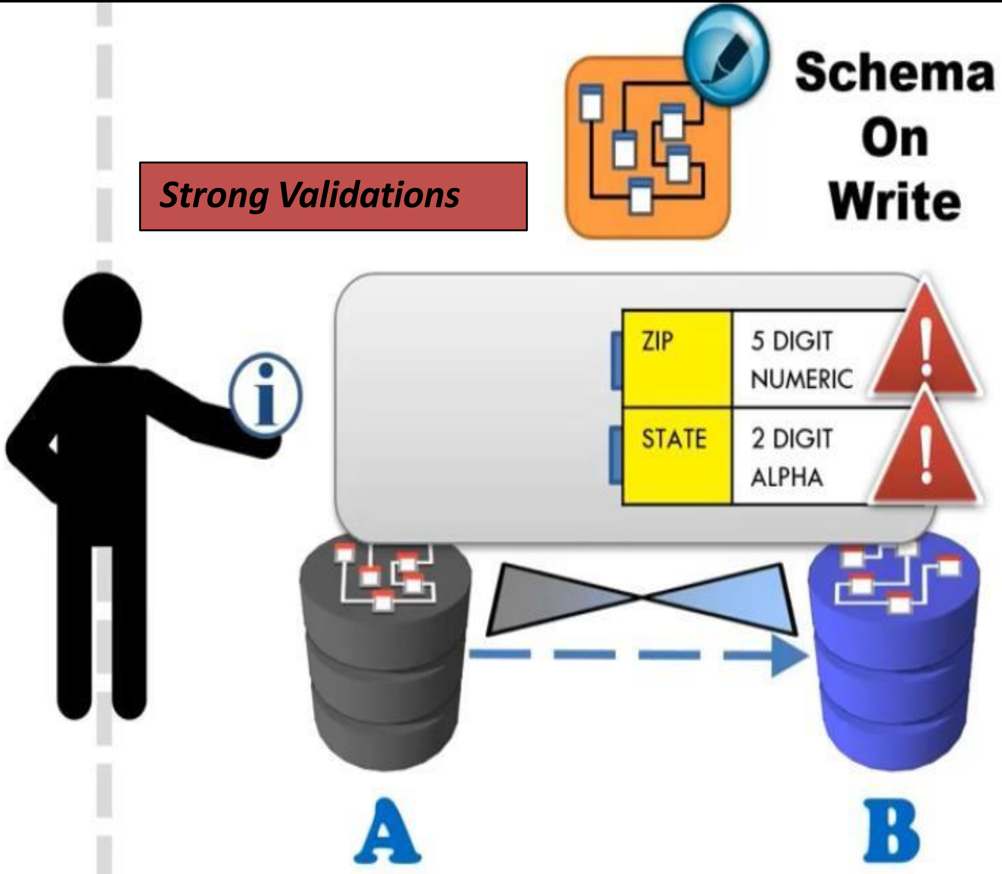
Differences between Hadoop and Sql DB

HBASE

COMPARISON BETWEEN HADOOP AND SQL

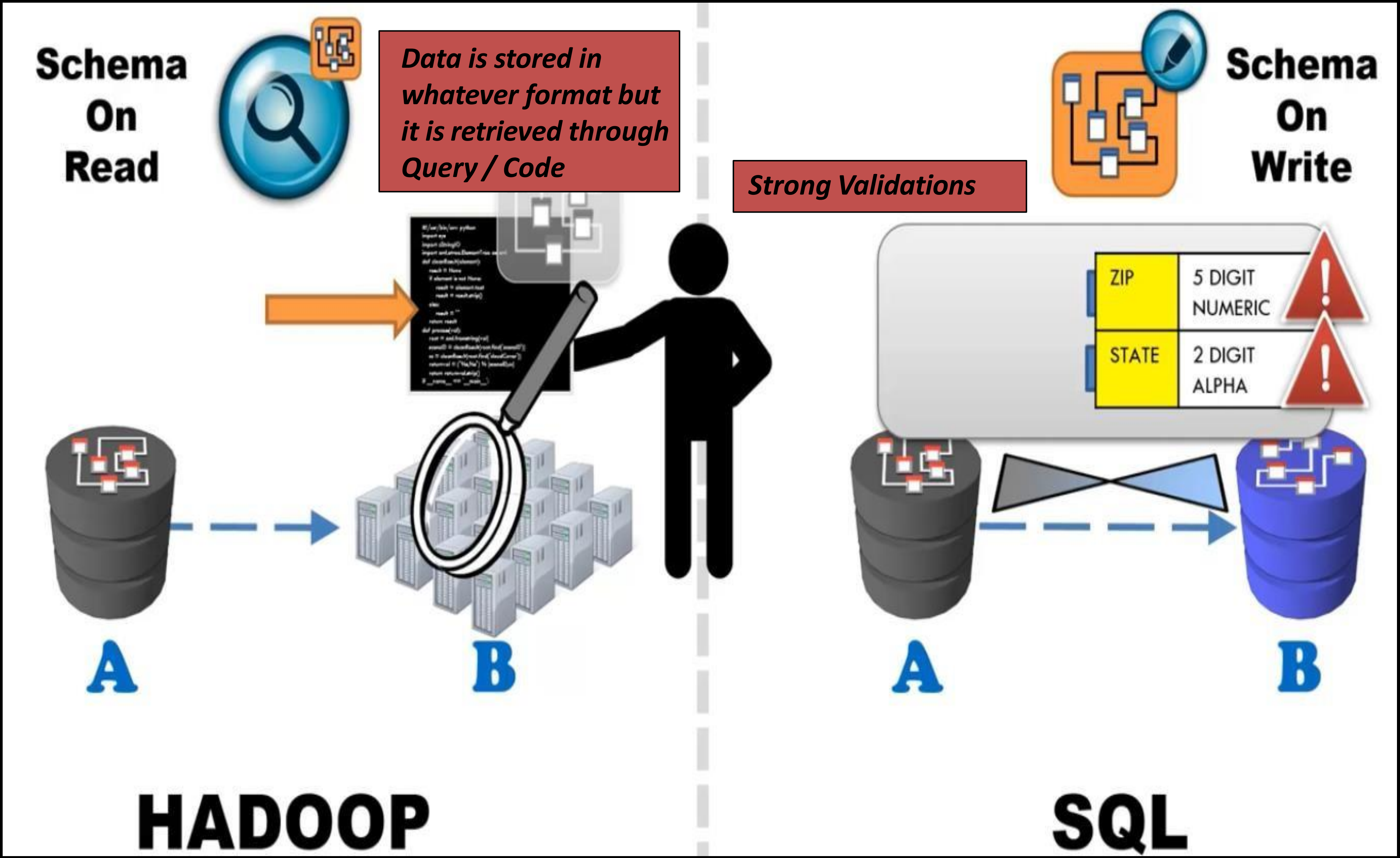
MAJOR DIFFERENCES AND UTILIZATION



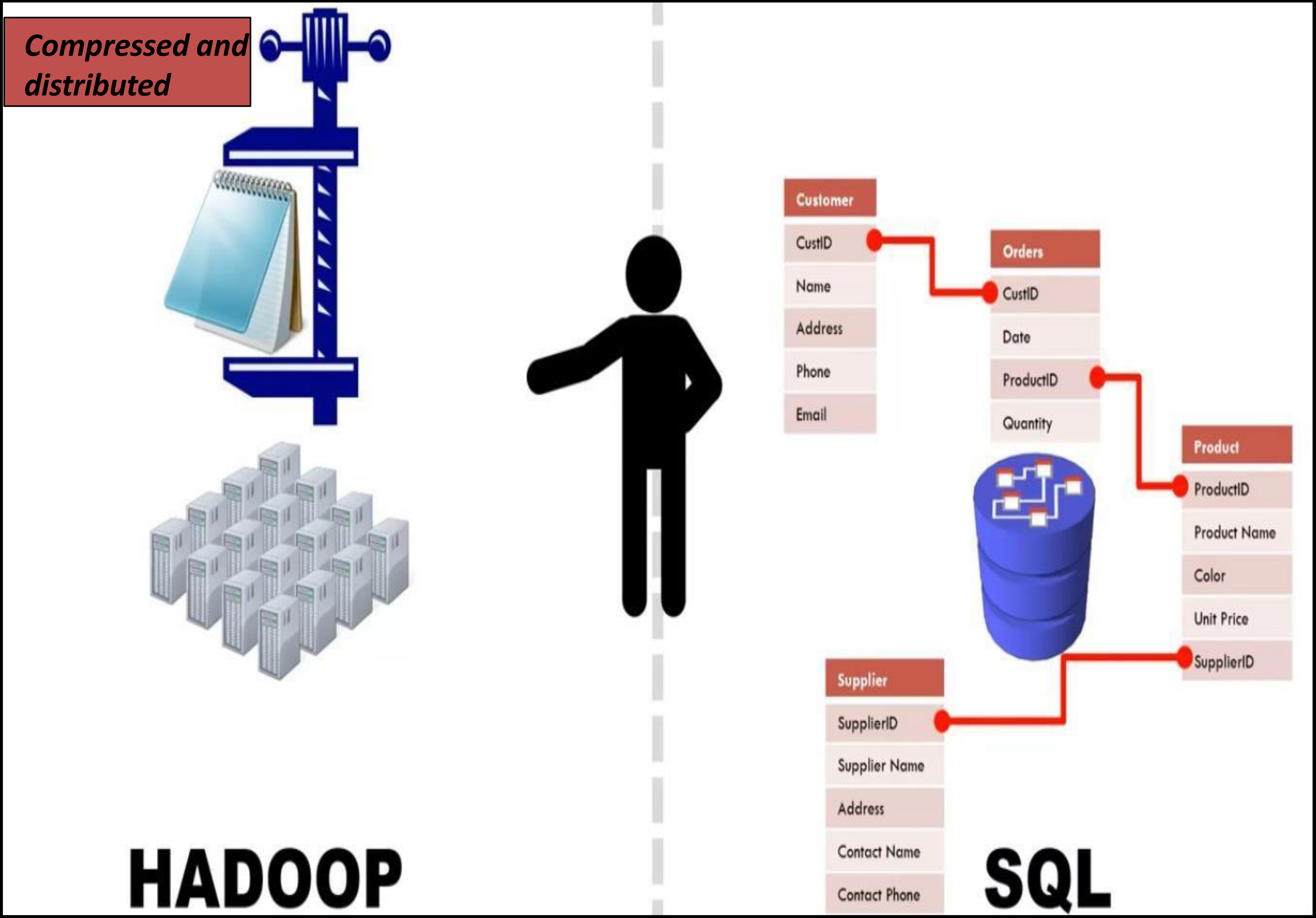

Strong Validations
Schema On Write
ZIP
5 DIGIT NUMERIC
STATE
2 DIGIT ALPHA
A
B

HADOOP

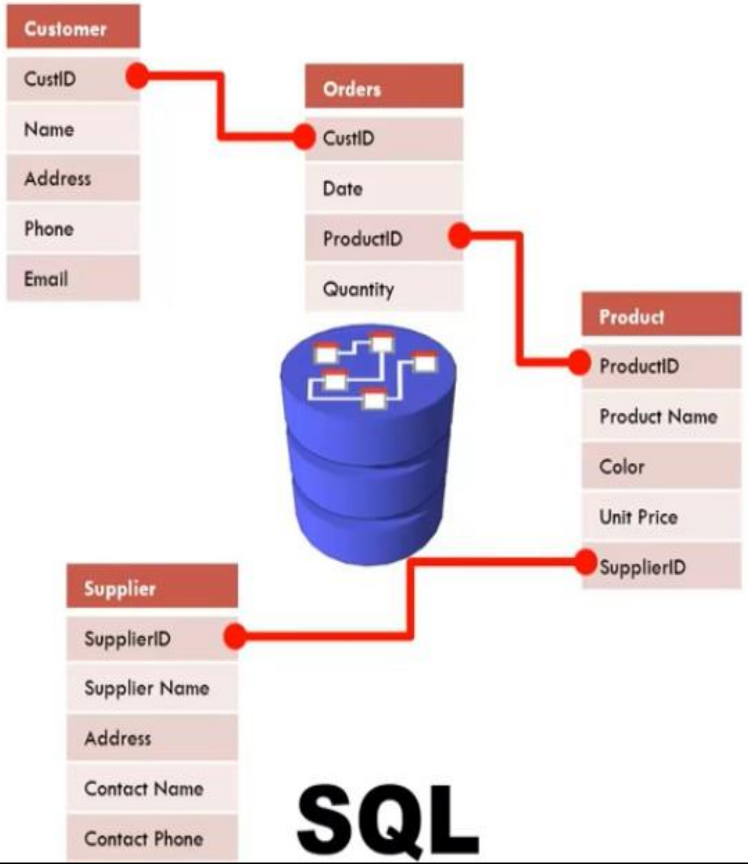
SQL


Schema On Read
Data is stored in whatever format but it is retrieved through Query / Code
Strong Validations
Schema On Write
ZIP
5 DIGIT NUMERIC
STATE
2 DIGIT ALPHA
A
B
A
B
HADOOP
SQL

Compressed and distributed

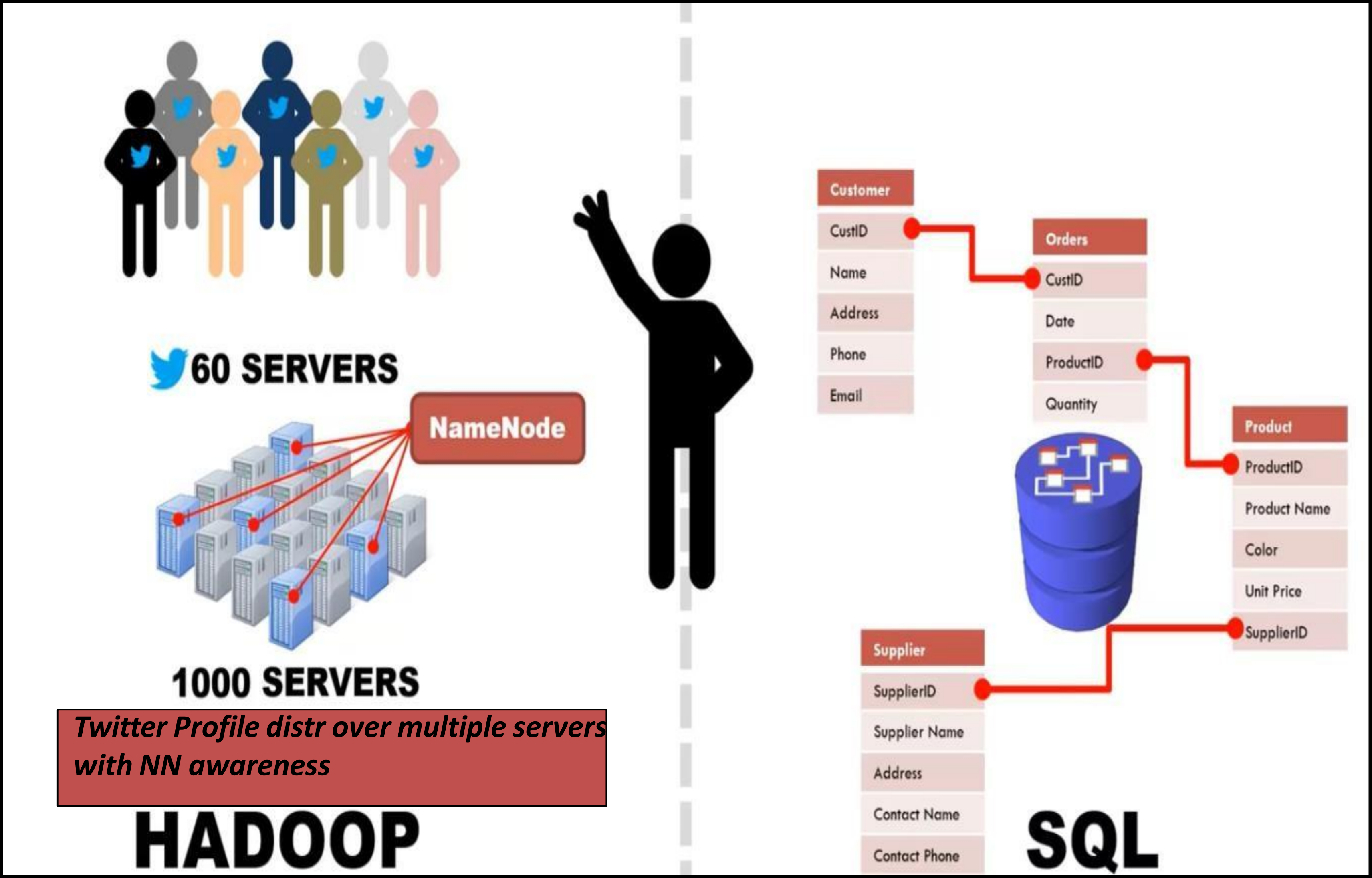


Compressed and distributed across multiple nodes



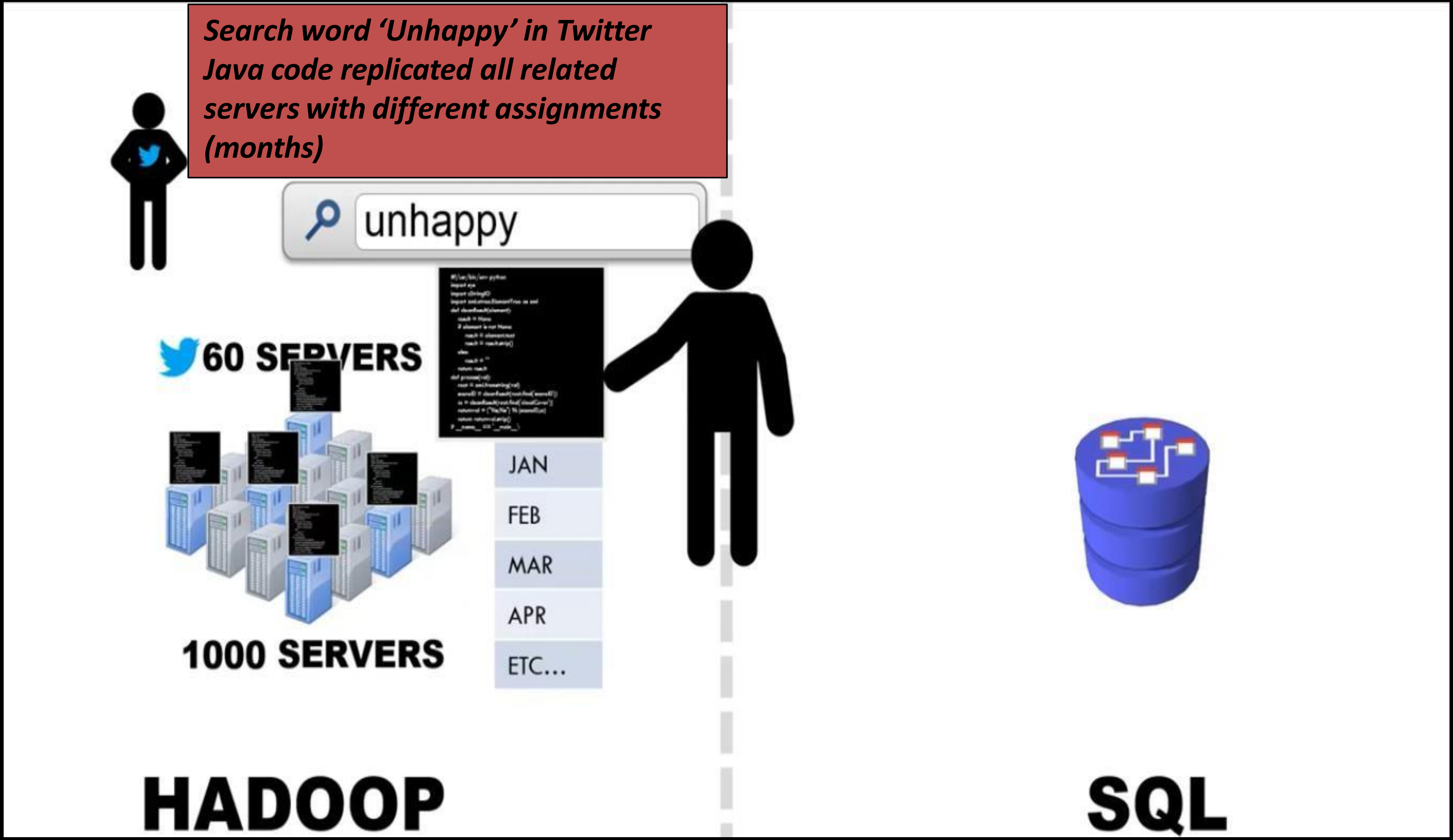
HADOOP

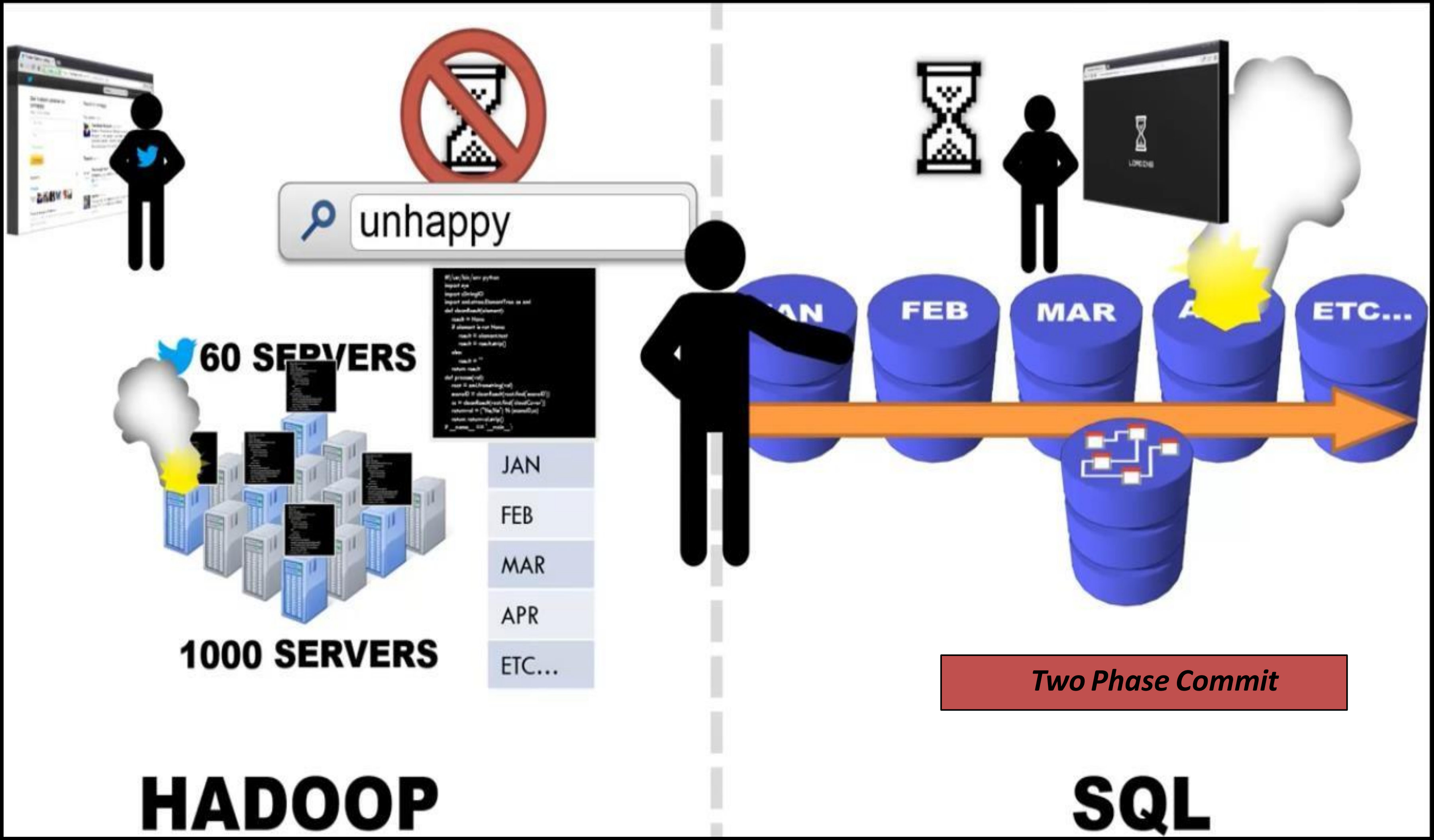
SQL



Twitter Profile distr over multiple servers with NN awareness

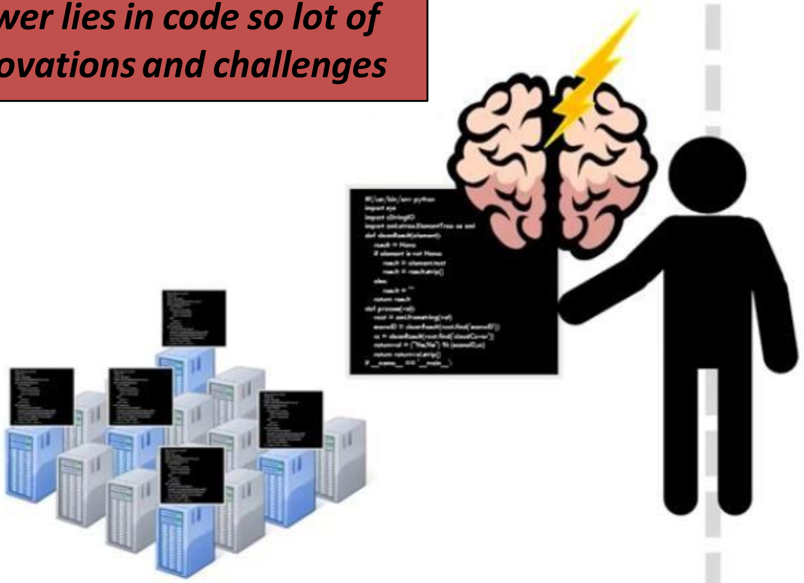
Search word 'Unhappy' in Twitter Java code replicated all related servers with different assignments (months)





In case of hadoop, result will be communicated despite one node going down, but in sql it will stop result till fully verified

Power lies in code so lot of innovations and challenges

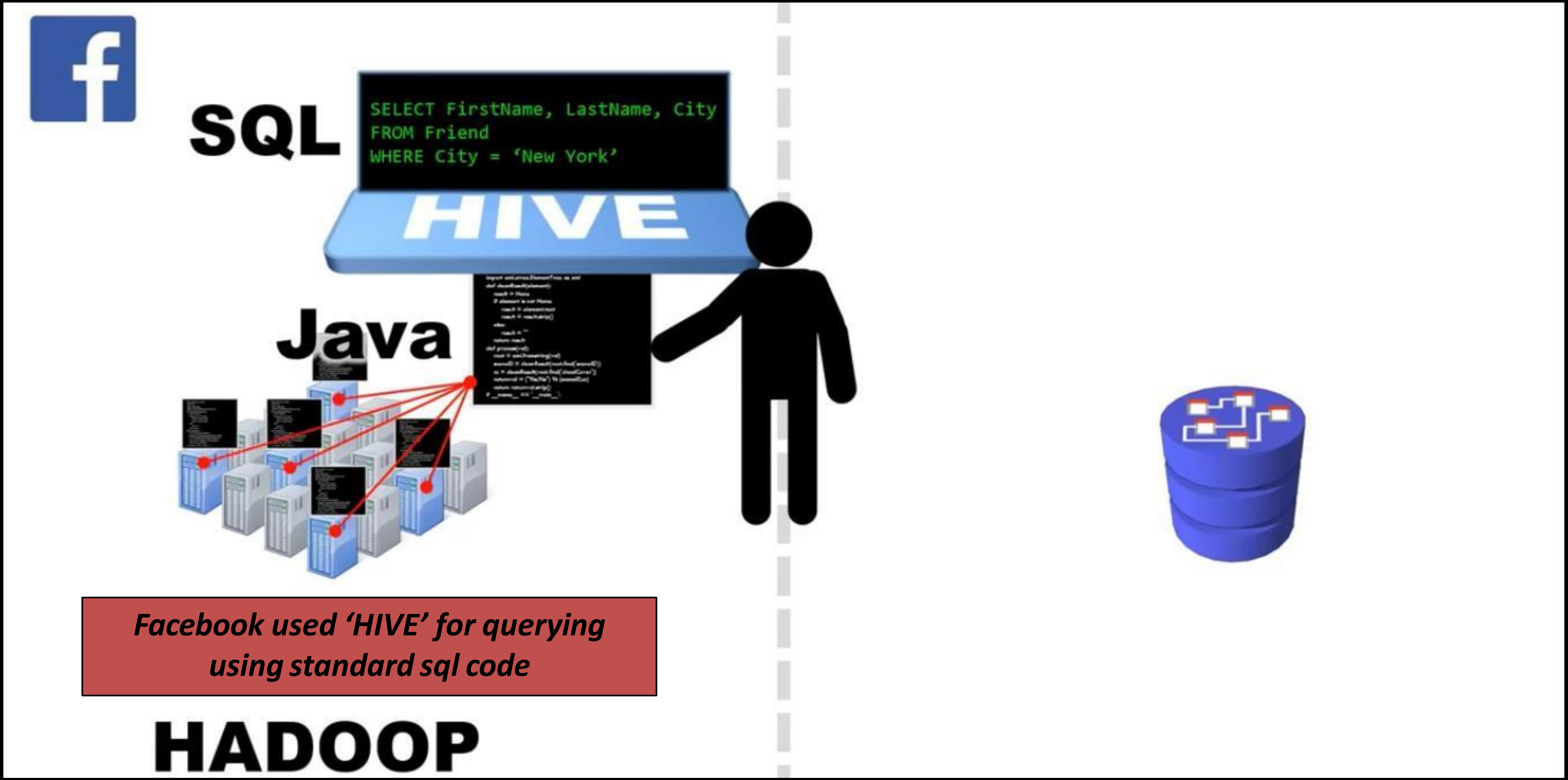


```
#!/usr/bin/env python
import sys
import cStringIO
import [illegible] as [illegible]
def [illegible](element):
    result = None
    if element is not None:
        result = element.text
        result = result.strip()
    else:
        result = ""
    return result
def process(val):
    root = [illegible](val)
    [illegible] = [illegible](root.find('[illegible]'))
    [illegible] = [illegible](root.find('[illegible]'))
    [illegible] = ("%s,%s") % ([illegible])
    return [illegible].strip()
if __name__ == '__main__':
```

HADOOP

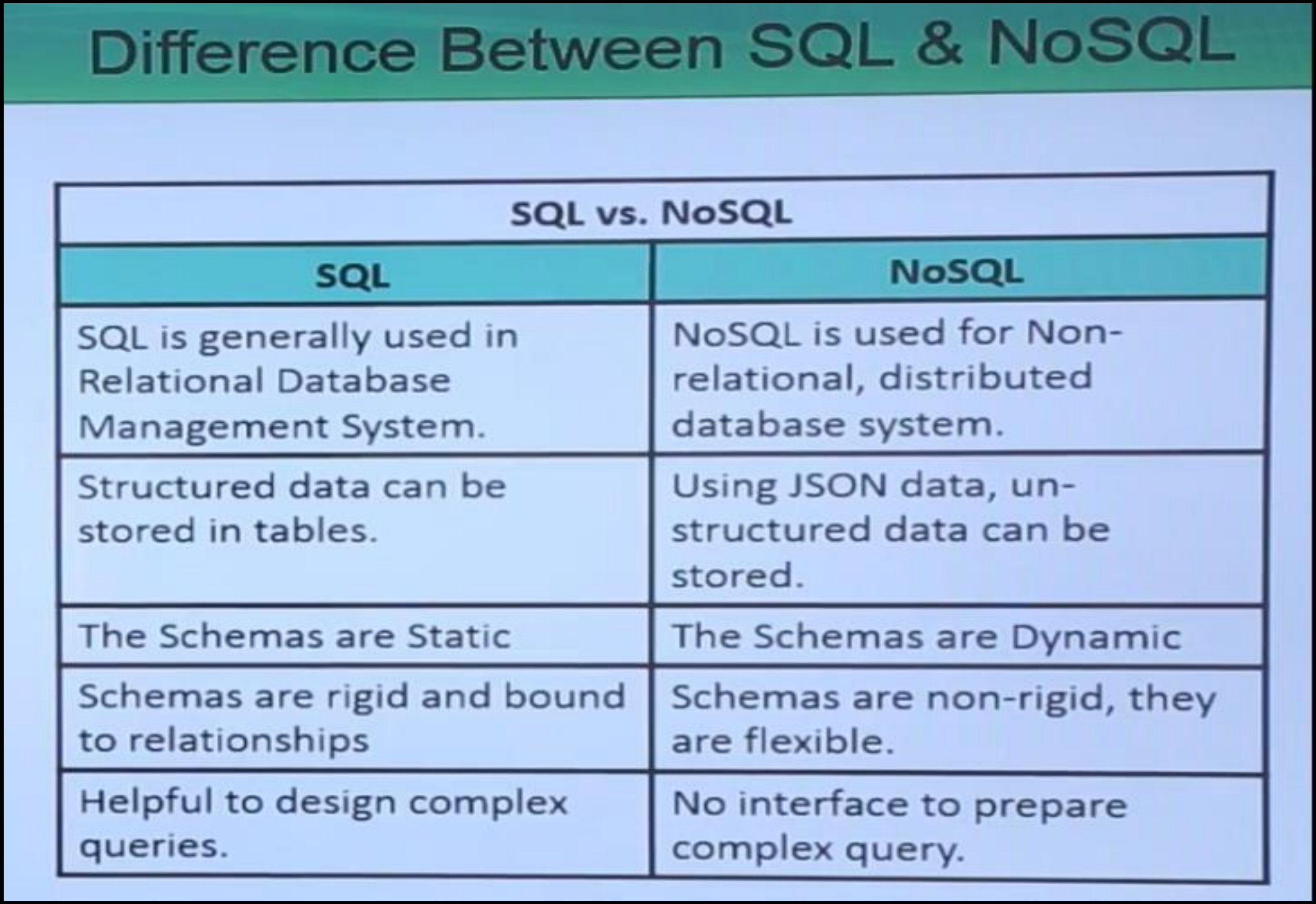
SQL

Flexibility



```
SELECT FirstName, LastName, City
FROM Friend
WHERE City = 'New York'
```

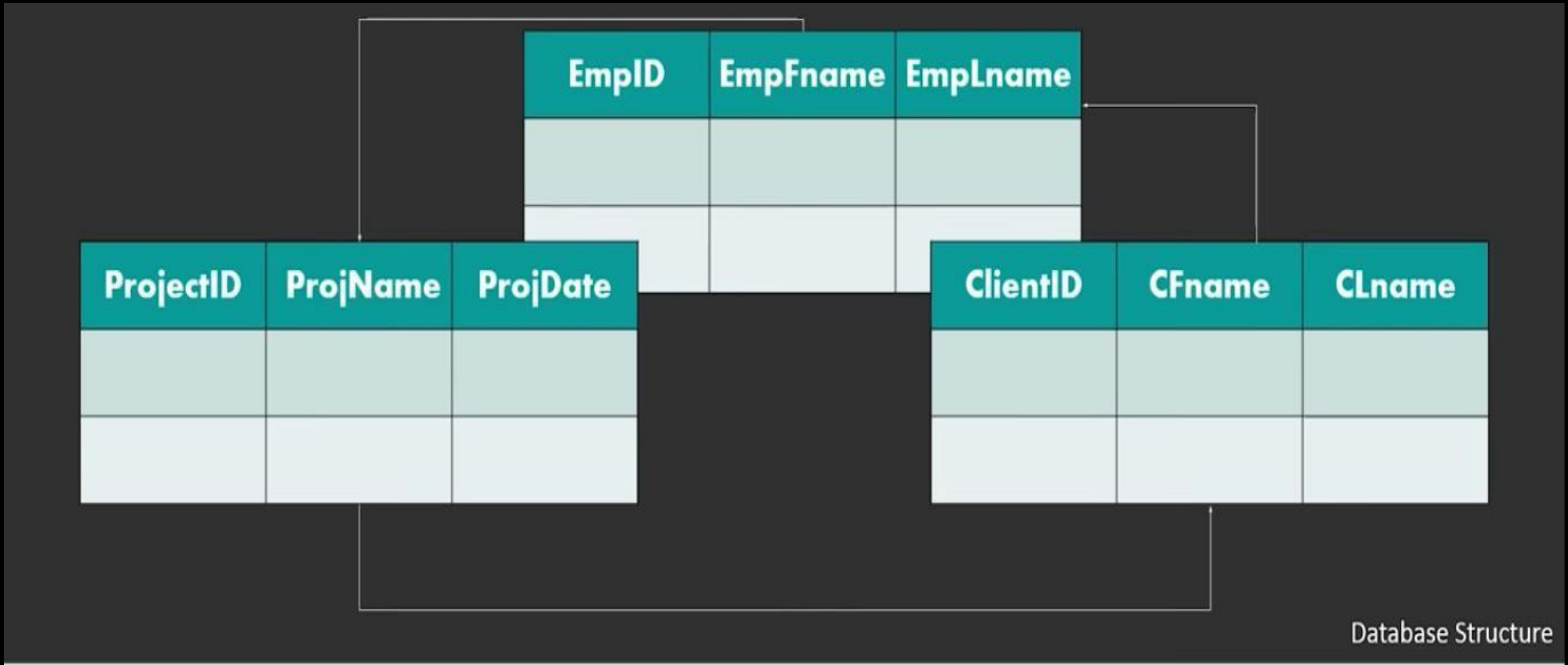
Facebook used 'HIVE' for querying using standard sql code



Difference Between SQL & NoSQL

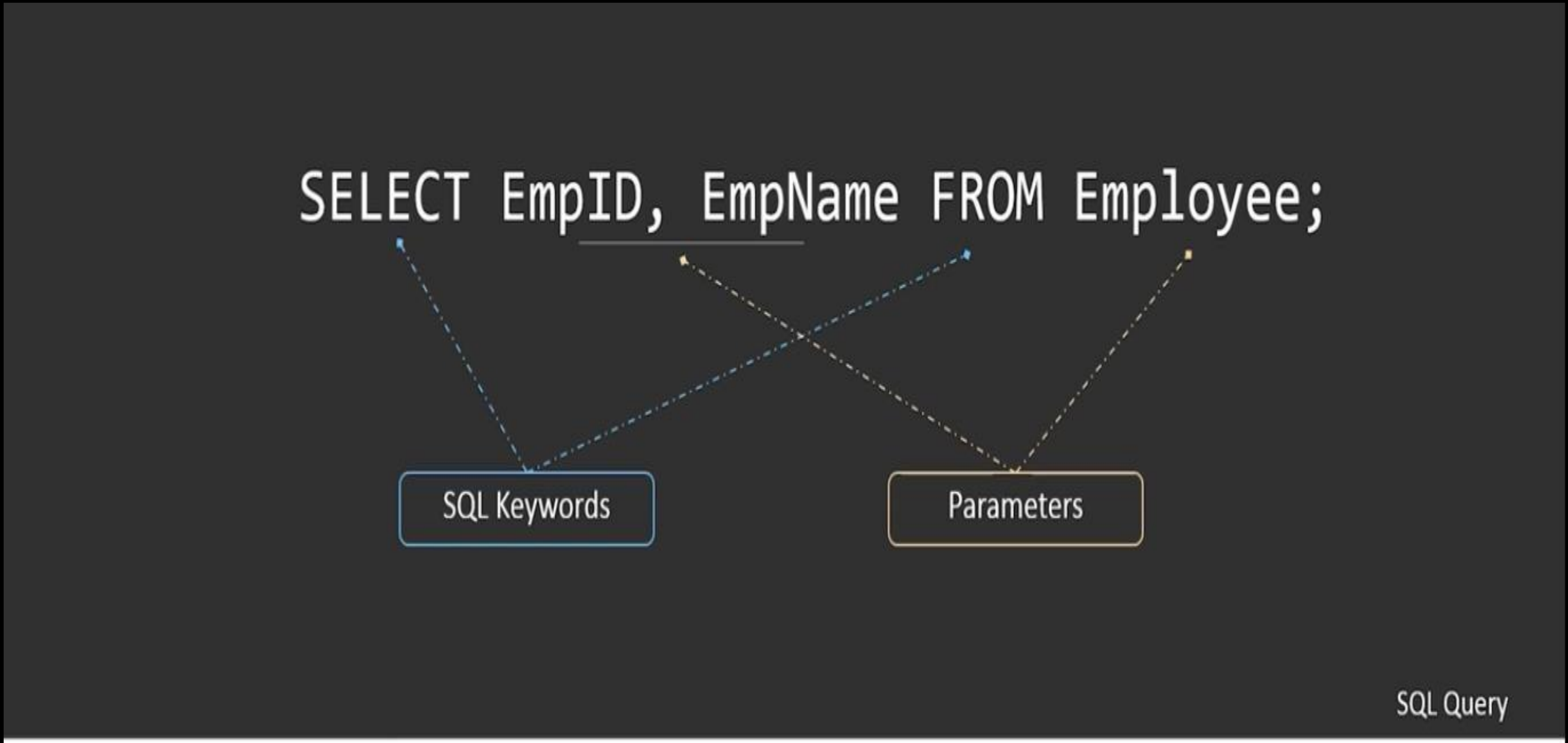
SQL vs. NoSQL	
SQL	**NoSQL**
SQL is generally used in Relational Database Management System.	NoSQL is used for Non-relational, distributed database system.
Structured data can be stored in tables.	Using JSON data, un-structured data can be stored.
The Schemas are Static	The Schemas are Dynamic
Schemas are rigid and bound to relationships	Schemas are non-rigid, they are flexible.
Helpful to design complex queries.	No interface to prepare complex query.

SQL



What is SQL?

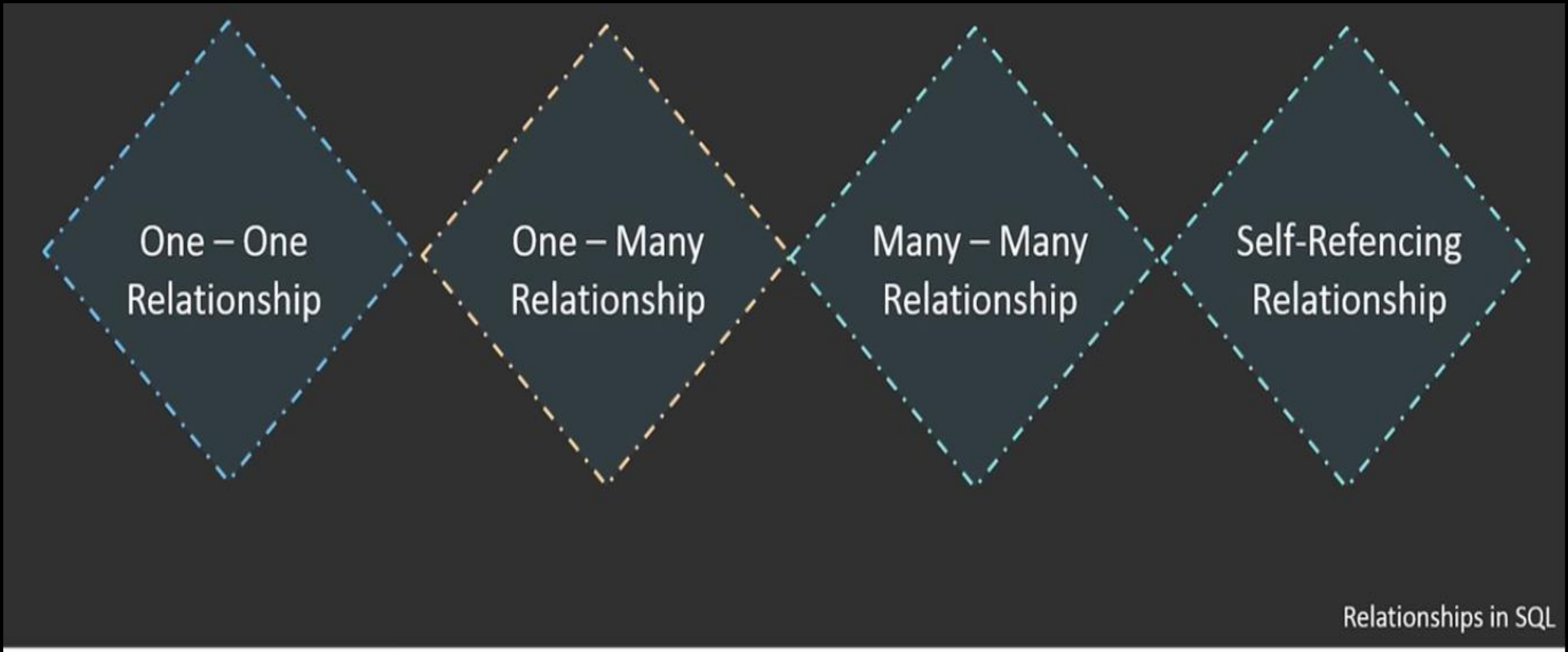
SQL is a standard language which stands for Structured Query Language. SQL is the core of relational database and is used for accessing and managing database.



```
SELECT EmpID, EmpName FROM Employee;
```

What is SQL?

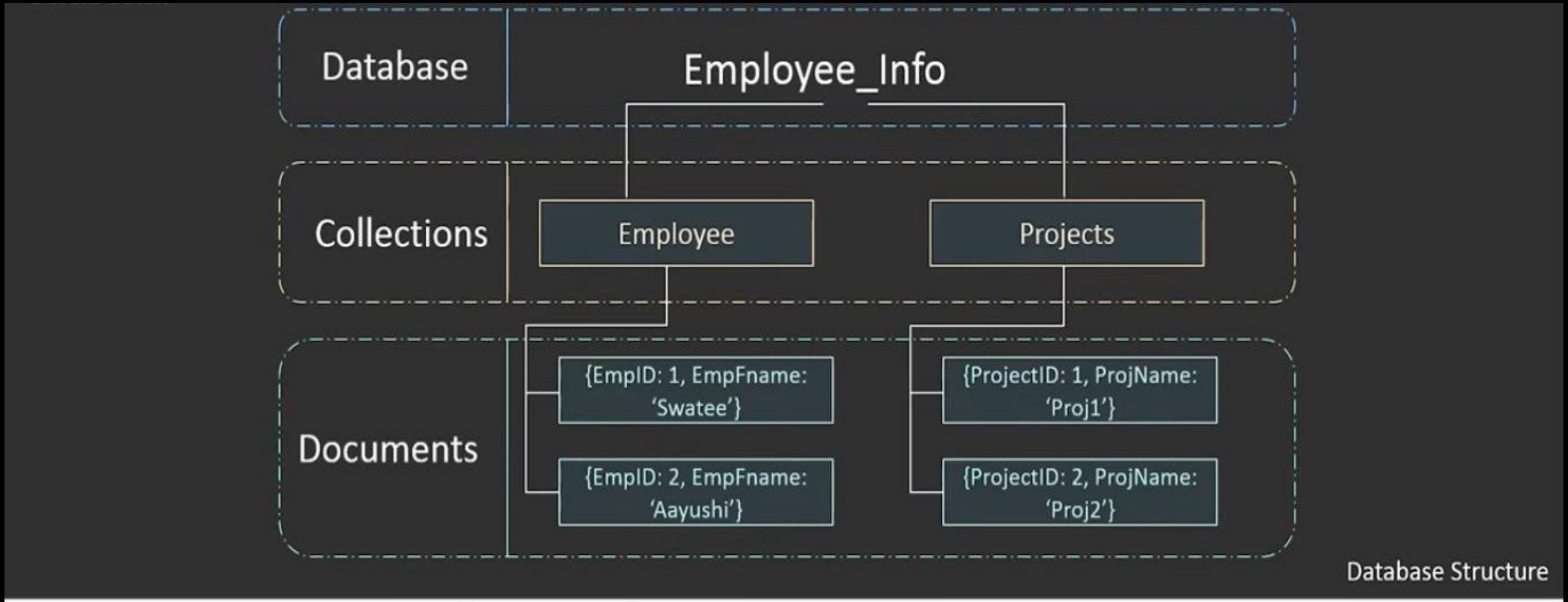
SQL is a standard language which stands for Structured Query Language. SQL is the core of relational database and is used for accessing and managing database.



What is SQL?

SQL is a standard language which stands for Structured Query Language. SQL is the core of relational database and is used for accessing and managing database.

NoSQL


Database
Employee_Info
Collections
Employee
Projects
Documents
{EmpID: 1, EmpFname: 'Swatee'}
{EmpID: 2, EmpFname: 'Aayushi'}
{ProjectID: 1, ProjName: 'Proj1'}
{ProjectID: 2, ProjName: 'Proj2'}

Database Structure

What is NoSQL?

NoSQL, known as Not only SQL database, provides a mechanism for storage and retrieval of data and is the next generation database . It has no specific schema and can handle huge amount of data.

No Specific Schema

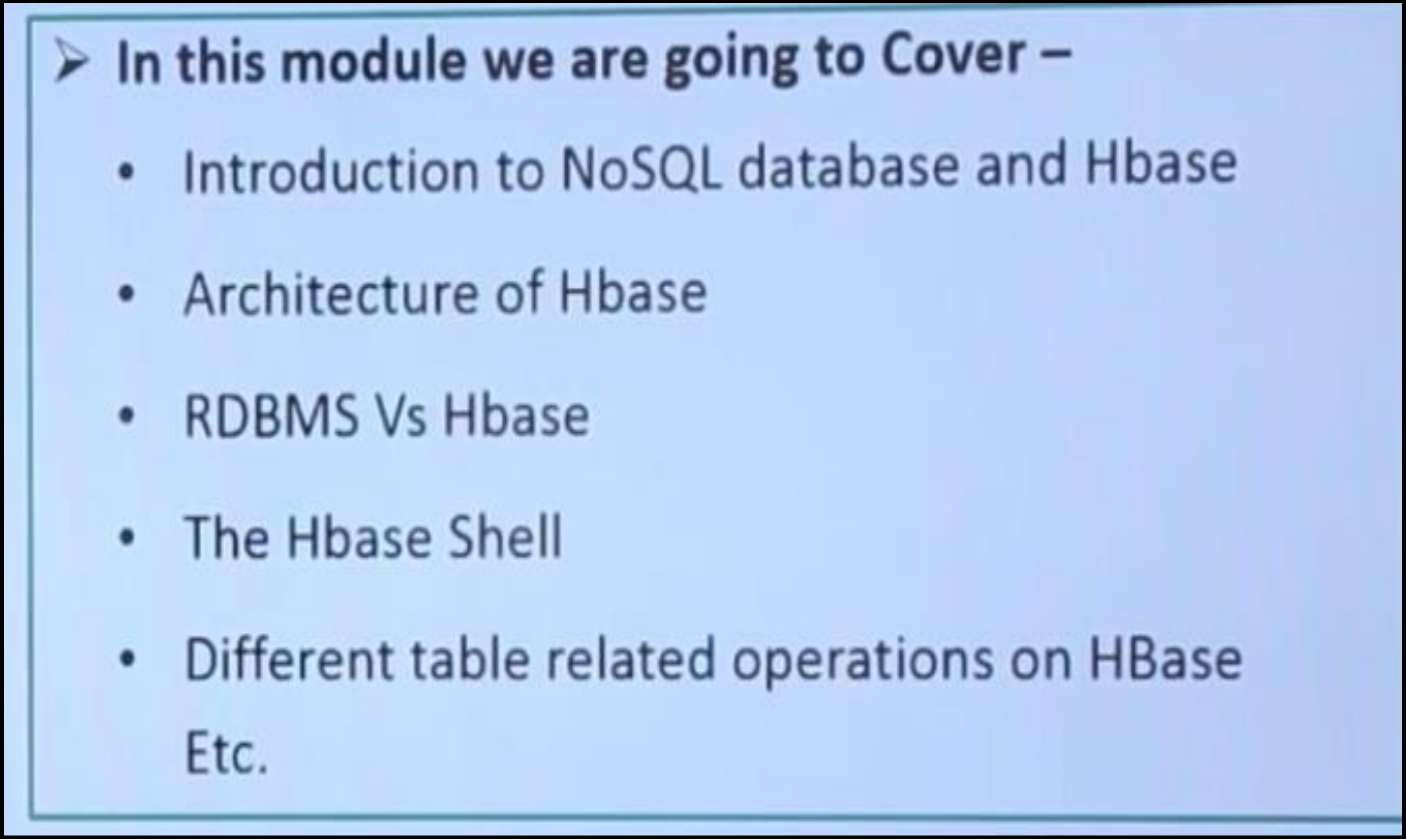
No Query Language

No or a Few Relationships

What is NoSQL?

NoSQL, known as Not only SQL database, provides a mechanism for storage and retrieval of data and is the next generation database . It has no specific schema and can handle huge amount of data.

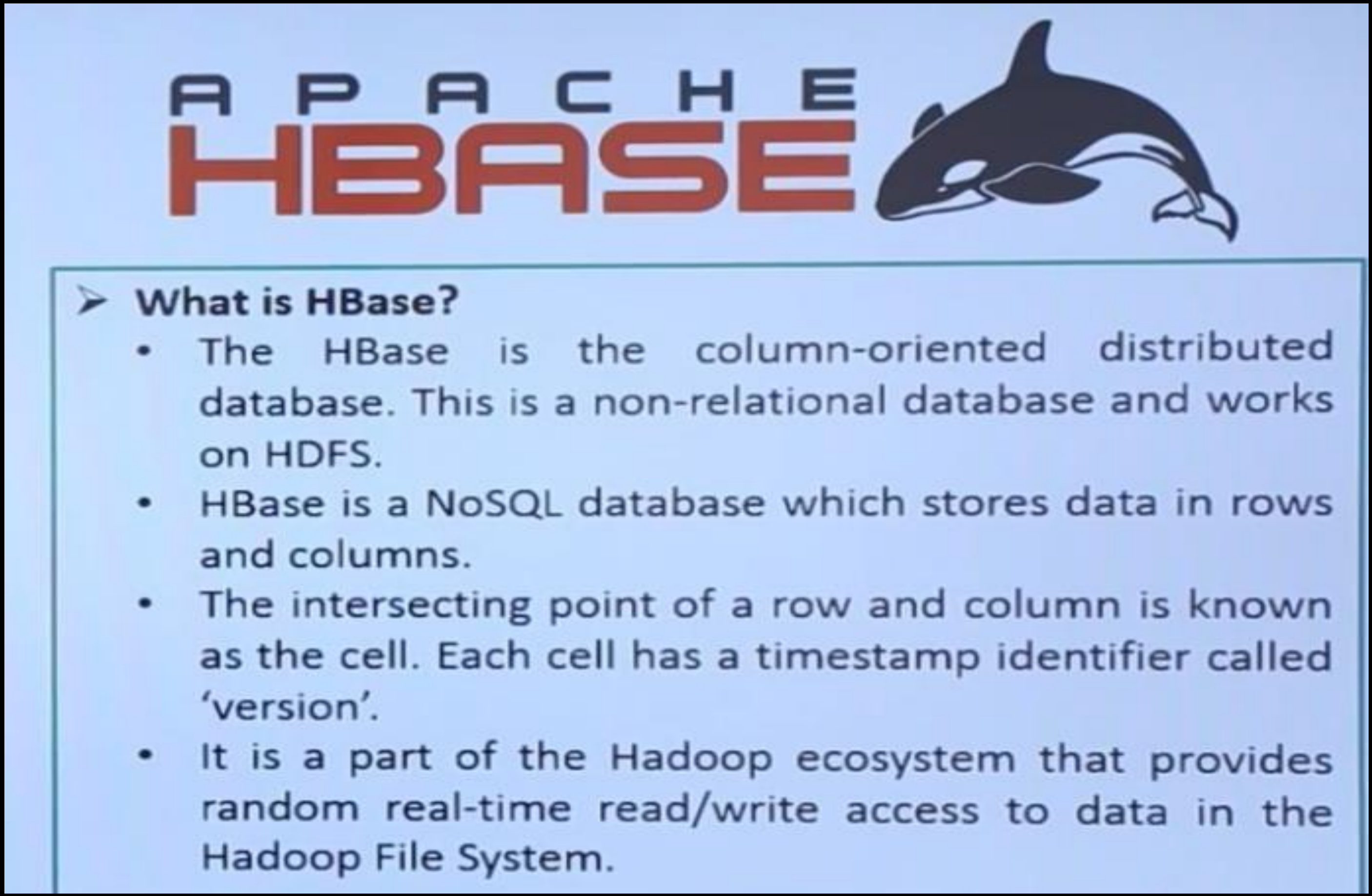
HBASE



- **In this module we are going to Cover –**
  - Introduction to NoSQL database and Hbase
  - Architecture of Hbase
  - RDBMS Vs Hbase
  - The Hbase Shell
  - Different table related operations on HBase

    Etc.

HBASE

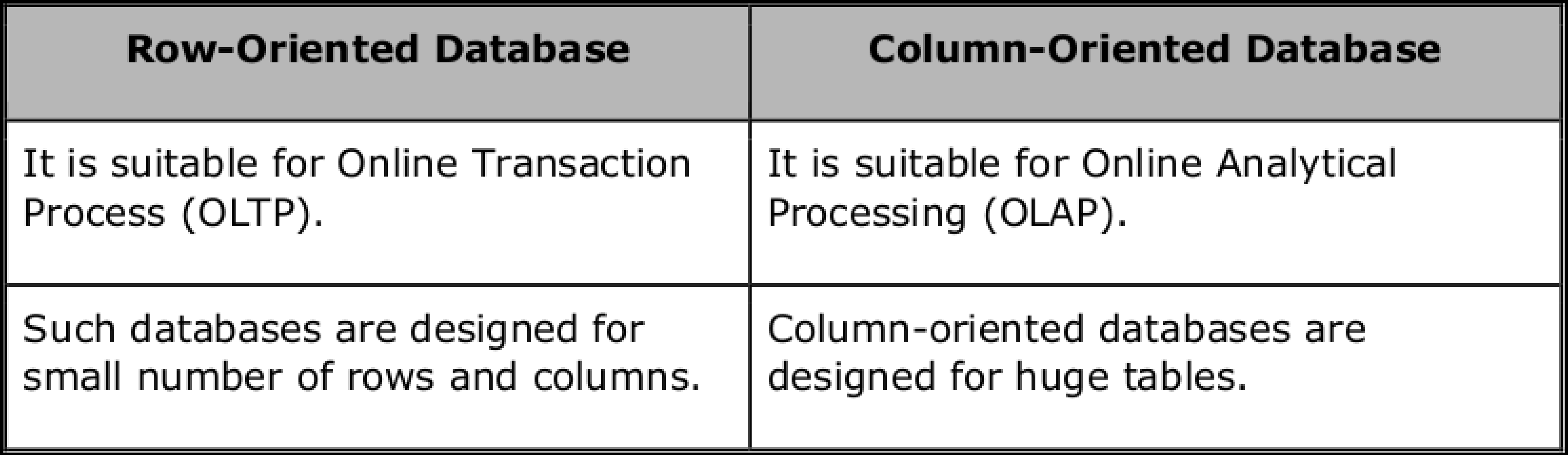


APACHE HBASE

➢ **What is HBase?**

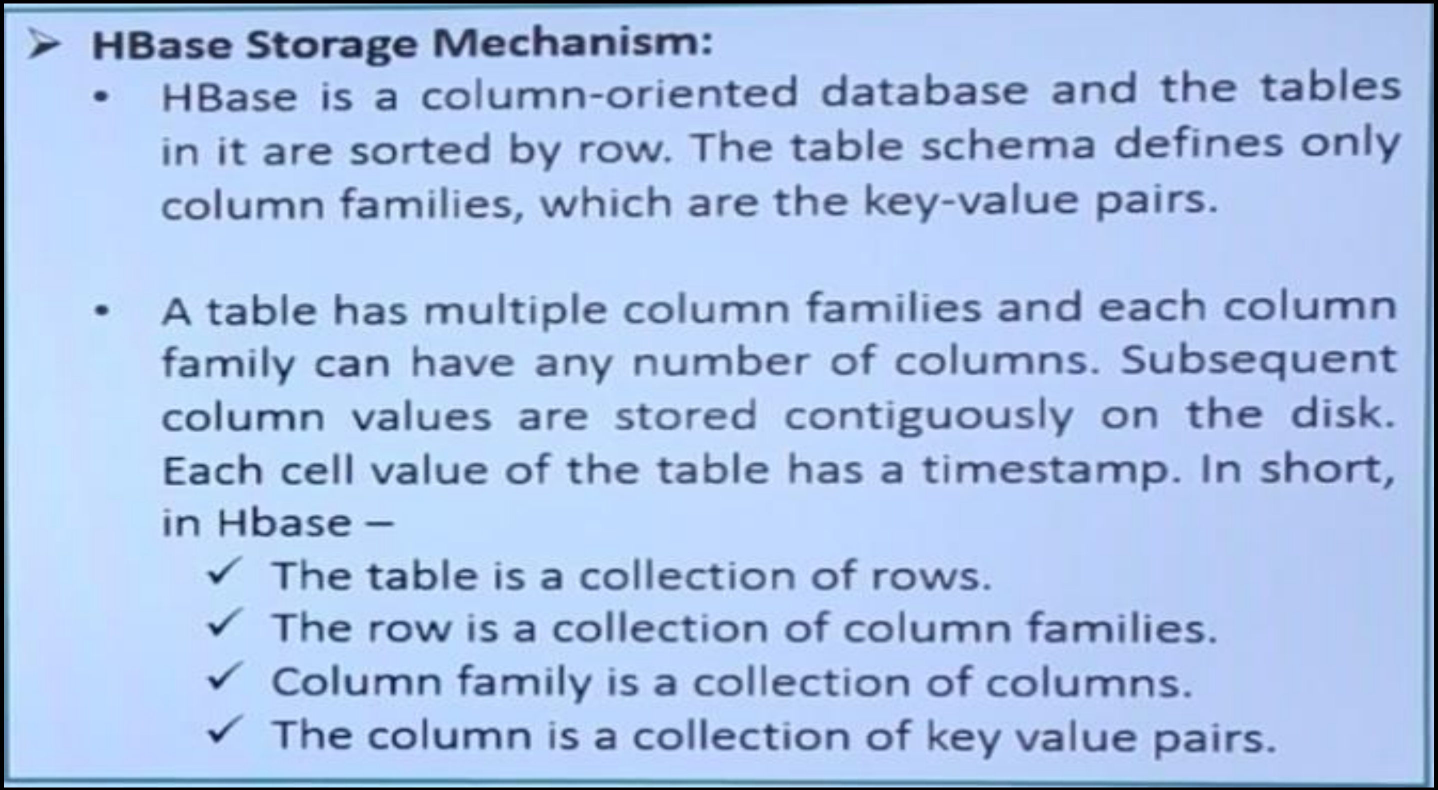
- The HBase is the column-oriented distributed database. This is a non-relational database and works on HDFS.
- HBase is a NoSQL database which stores data in rows and columns.
- The intersecting point of a row and column is known as the cell. Each cell has a timestamp identifier called 'version'.
- It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

HBASE



Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

HBASE



- **HBase Storage Mechanism:**
  - HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key-value pairs.
  - A table has multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in Hbase –
    - ✓ The table is a collection of rows.
    - ✓ The row is a collection of column families.
    - ✓ Column family is a collection of columns.
    - ✓ The column is a collection of key value pairs.

Hbase – Why / When use it ?

- Data in billions of rows
- Complex data
- High volume of I/O
- High level of data nodes, 5 +
- No need for extra RDBMS functions i.e. transactions

When To Use HBase

- **Have to have enough hardware!!**
  - At the minimum 5 nodes
    - There are multiple management daemon processes: Namenode, HBaseMaster, Zookeeper, etc....
    - HDFS won't do well on anything under 5 nodes anyway; particularly with a block replication of 3
    - HBase is memory and CPU intensive
- **Carefully evaluate HBase for mixed work loads**
  - Client Request vs. Batch processing (Map/Reduce)
  - HBase has intermittent but large IO access
    - May affect response latency!!!

HBase

- **Based on Google's Bigtable**
  - http://labs.google.com/papers/bigtable.html
- **Just like BigTable is built on top of Google File System (GFS), HBase is implemented on top of HDFS**
- **Horizontally scalable**
  - **Automatic sharding**
- **Strongly consistent reads and writes**
- **Automatic fail-over**
- **Simple Java API**
- **Integration with Map/Reduce framework**
- **Thrift, Avro and REST-ful Web-services**

Who Uses HBase?

- **Here is a very limited list of well known names**
  - Facebook
  - Adobe
  - Twitter
  - Yahoo!
  - Netflix
  - Meetup
  - Stumbleupon

HBase Data Model

- **Data is stored in Tables**
- **Tables contain rows**
  - Rows are referenced by a unique key
    - Key is an array of bytes – good news
    - Anything can be a key: string, long and your own serialized data structures
- **Rows made of columns which are grouped in column families**
- **Data is stored in cells**
  - Identified by row x column-family x column
  - Cell's content is also an array of bytes

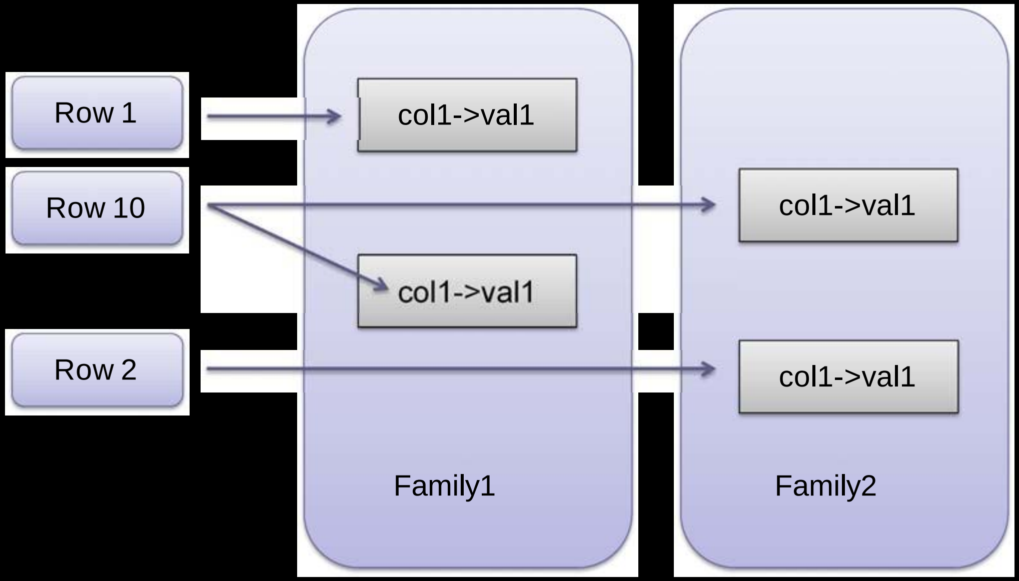
HBase Families

- **Rows are grouped into families**
  - Labeled as “family:column”
    - Example “user:first_name”
  - A way to organize your data
  - Various features are applied to families
    - Compression
    - In-memory option
    - Stored together - in a file called HFile/StoreFile
- **Family definitions are static**
  - Created with table, should be rarely added and changed
  - Limited to small number of families
    - unlike columns that you can have millions of

HBase Families

- **Family name must be composed of printable characters**
  - Not bytes, unlike keys and values
- **Think of family:column as a tag for a cell value and NOT as a spreadsheet**
- **Columns on the other hand are NOT static**
  - Create new columns at run-time
  - Can scale to millions for a family

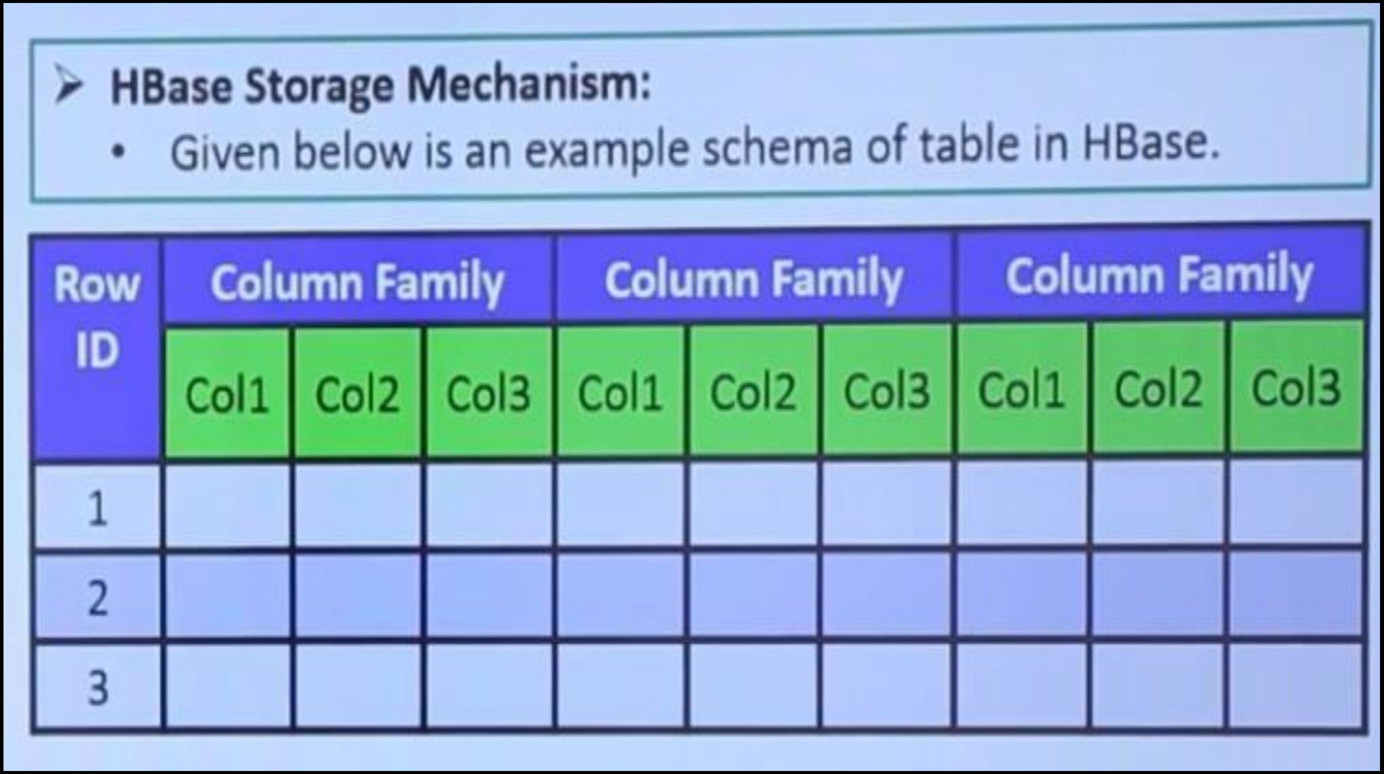
Rows Composed Of Cells Stored In Families:Columns



HBase Timestamps

- **Cells' values are versioned**
  - For each cell multiple versions are kept
    - 3 by default
  - Another dimension to identify your data
  - Either explicitly timestamped by region server or provided by the client
    - Versions are stored in decreasing timestamp order
    - Read the latest first – optimization to read the current value
- **You can specify how many versions are kept**

HBASE



➢ **HBase Storage Mechanism:**

- Given below is an example schema of table in HBase.

Row ID	Column Family			Column Family			Column Family		
	Col1	Col2	Col3	Col1	Col2	Col3	Col1	Col2	Col3
1									
2									
3									

HBase Cells

An example - Logical representation of how values are stored

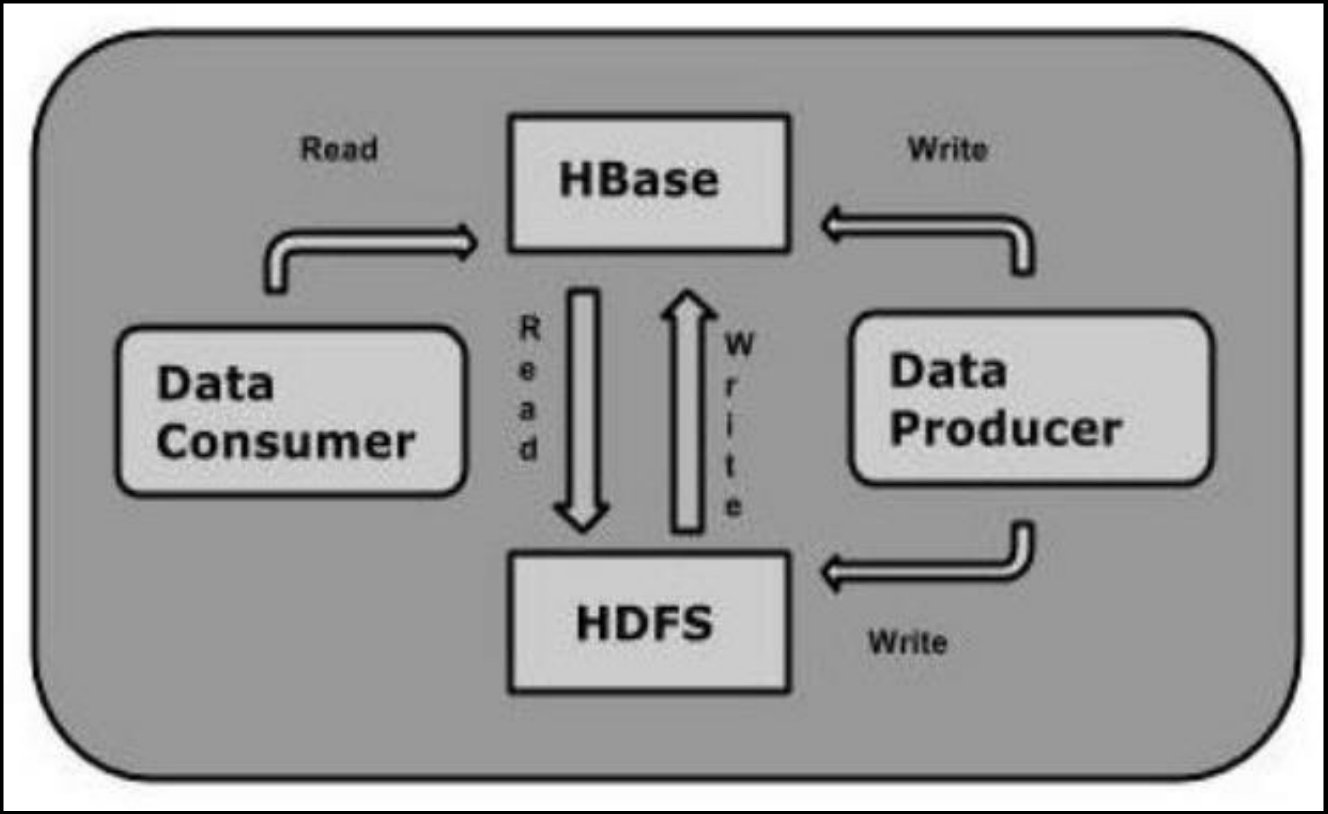
Row Key	Time stamp	Name Family		Address Family	
		first_name	last_name	number	address
row1	t1	**Bob**	**Smith**		
	t5			10	First Lane
	t10			30	Other Lane
	t15			**7**	**Last Street**
row2	t20	**Mary**	Tompson		
	t22			**77**	**One Street**
	t30		**Thompson**		

Source: Lars, George. HBase The Definitive Guide. O'Reilly Media. 2011

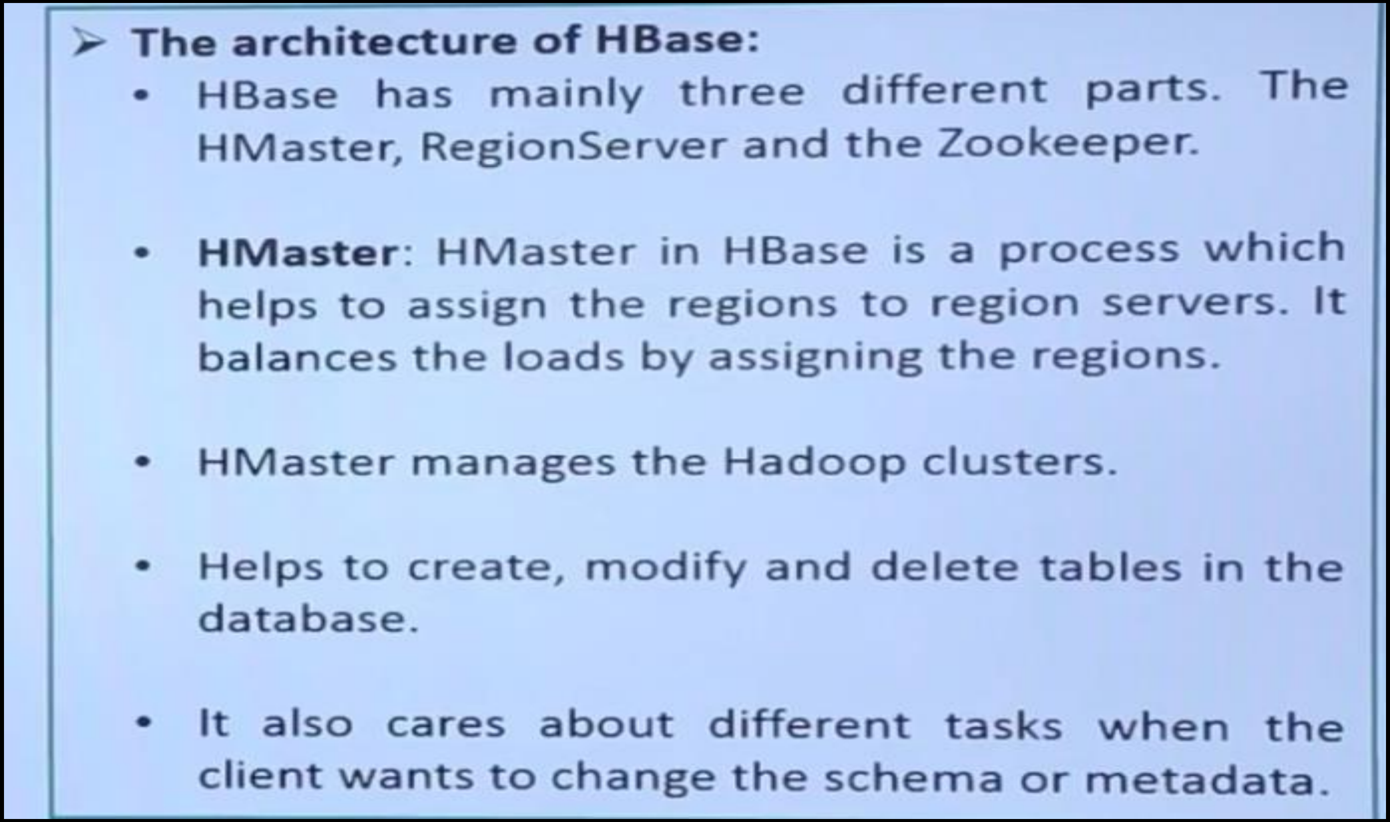
HBase Cells

- **Can ask for**
  - Most recent value (default)
  - Specific timestamp
  - Multiple values such as range of timestamps

Hbase Architecture



HBASE ARCHITECTURE



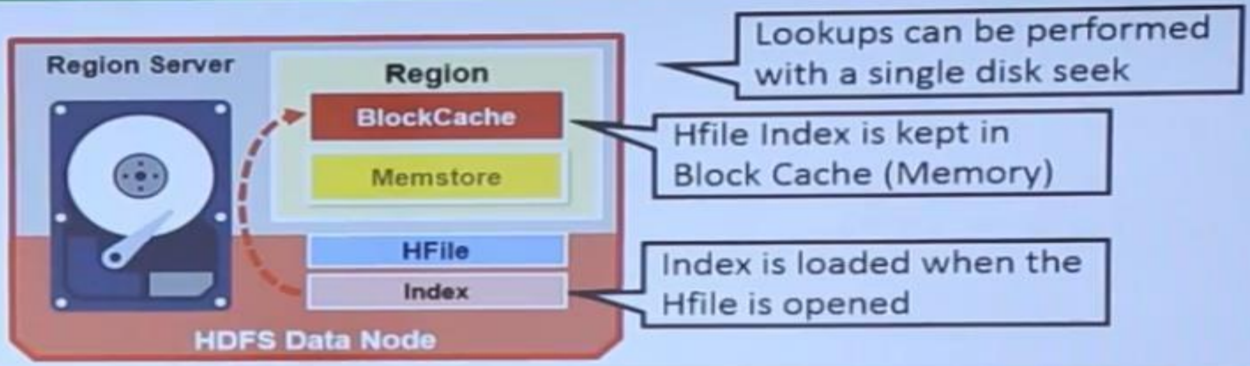
➢ **The architecture of HBase:**

- HBase has mainly three different parts. The HMaster, RegionServer and the Zookeeper.
- **HMaster**: HMaster in HBase is a process which helps to assign the regions to region servers. It balances the loads by assigning the regions.
- HMaster manages the Hadoop clusters.
- Helps to create, modify and delete tables in the database.
- It also cares about different tasks when the client wants to change the schema or metadata.

HMaster

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

HBASE ARCHITECTURE


Region Server
Region
BlockCache
Memstore
HFile
Index
HDFS Data Node
Lookups can be performed with a single disk seek
Hfile Index is kept in Block Cache (Memory)
Index is loaded when the Hfile is opened

➢ **The architecture of HBase:**

- **Region Server**: The Region Servers are the main working nodes. It handles the Read, write, modify requests from the clients. The Region Server runs on every node in the Hadoop Cluster.
- It has a read cache called **Block Cache**, read data are stored in the read cache, and when the cache is full, recently used data is removed.
- Another cache is present here called **MemStore**. It is the write cache. It stored new data that is not yet stored in the disks. Each Column-Family has different write cache in it.
- It has the actual storage file called HFile. It stores the actual data on a disk.

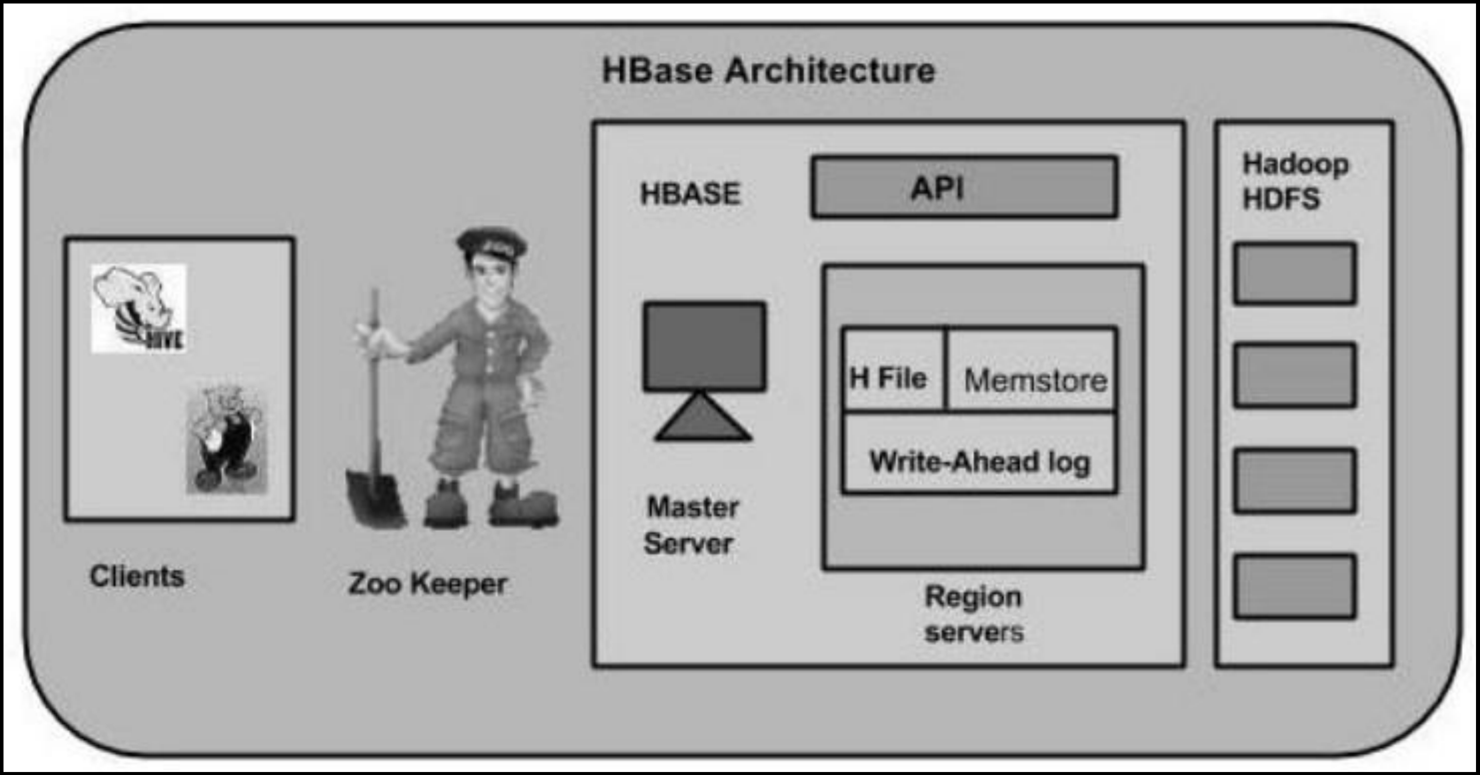
The RegionServer has 4 main components:

- **Write Ahead Log (WAL)** is a file on the DFS used to store data that hasn't yet been persisted to permanent storage; used for recovery in the case of failure.
- **BlockCache** is the read cache
  - Stores frequently read data in memory
  - Least Recently Used (LRU) data is evicted when full
- **MemStore** is the write cache
  - Stores new data which has not yet been written to disk
  - Data gets sorted before writing to disk
  - One MemStore per column family per region
- **HFiles** store the rows as sorted KeyValues on disk

Region server

- Regions
- Regions are nothing but tables that are split up and spread across the region servers.
- Region server
  - The region servers have regions that -
  - Communicate with the client and handle data- related operations. Handle read and write requests for all the regions under it.
  - Decide the size of the region by following the region size thresholds.

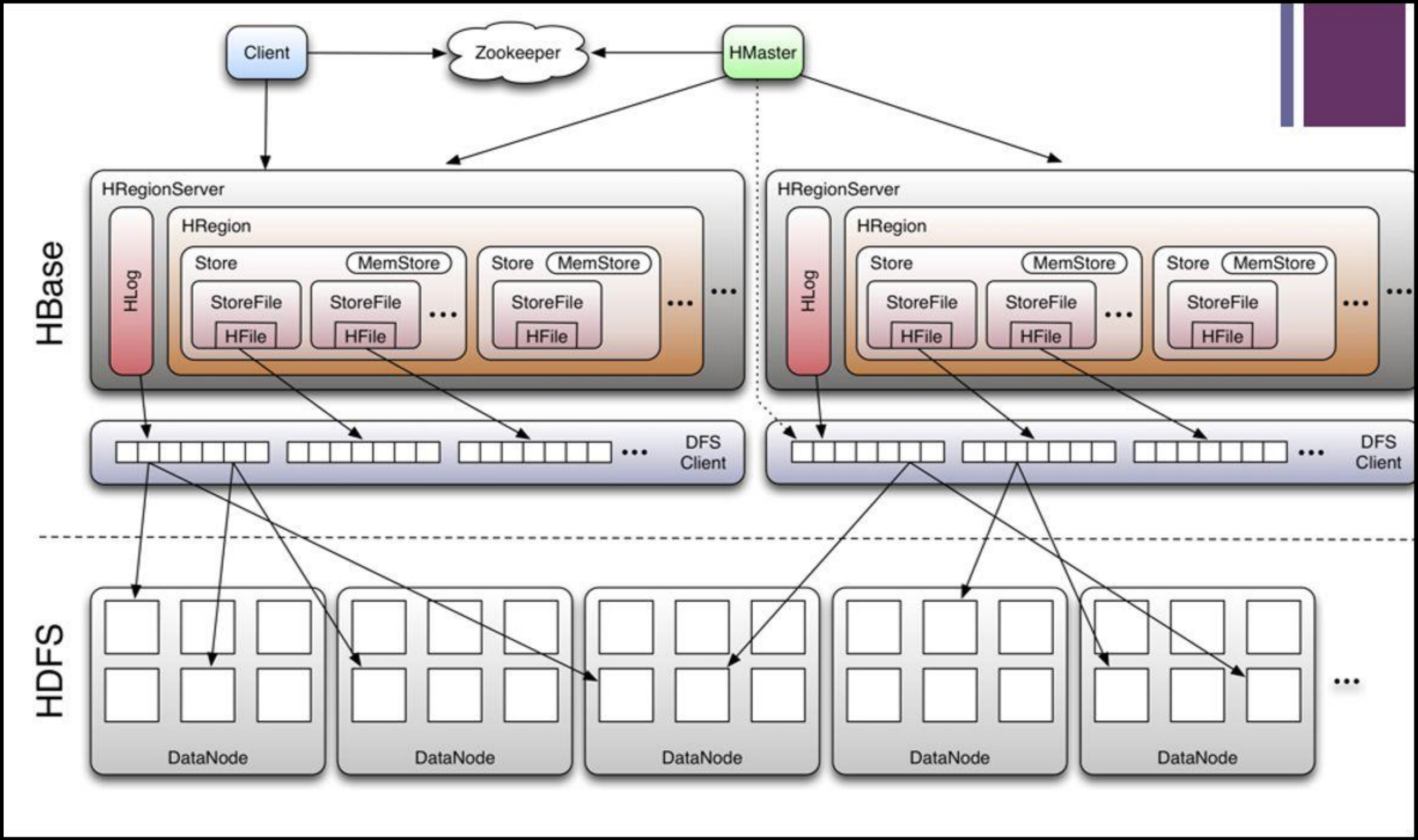
Architecture



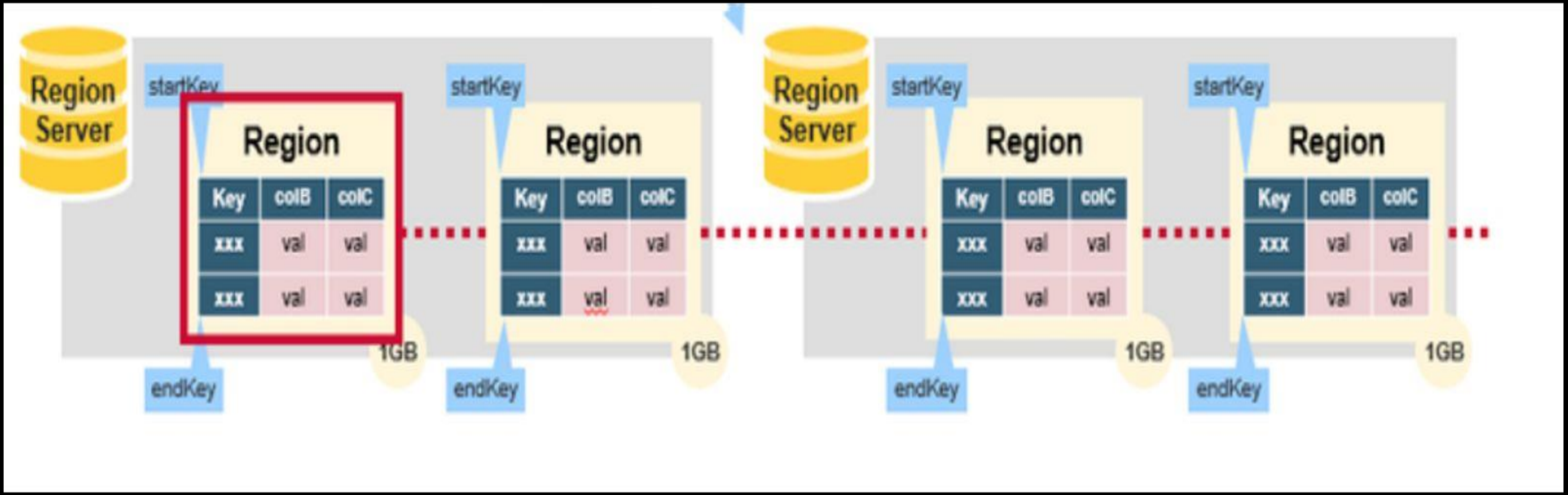
HBASE ARCHITECTURE

- Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
- Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.
- In pseudo and standalone modes, HBase itself will take care of zookeeper.

HBASE ARCHITECTURE

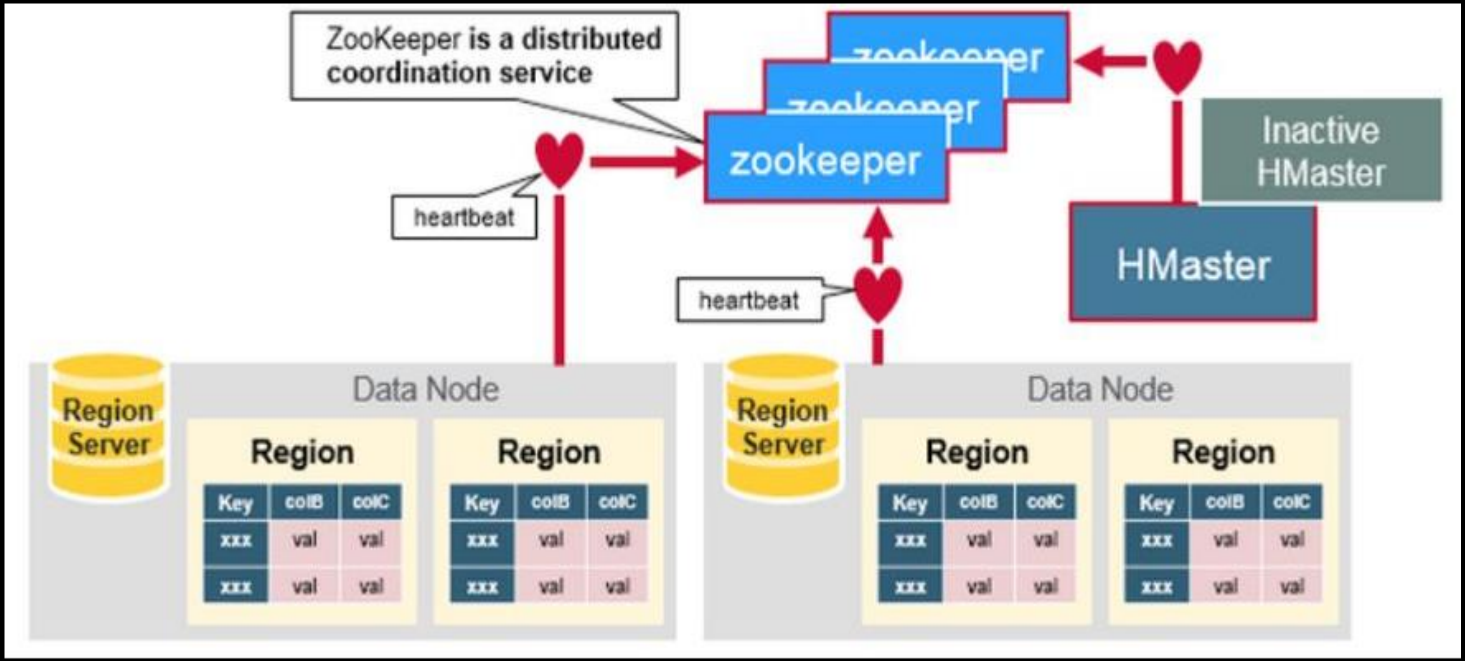


Architecture



- A **Region** has a default size of 256MB (can be configured based on need)
- A group of regions is served to the clients by a **Region Server**
- A Region Server can serve approximately 1000 regions to the client.

Architecture



- ZooKeeper maintains which servers are alive and available, and provides server failure notification.
- Zookeeper uses consensus to guarantee common shared state.

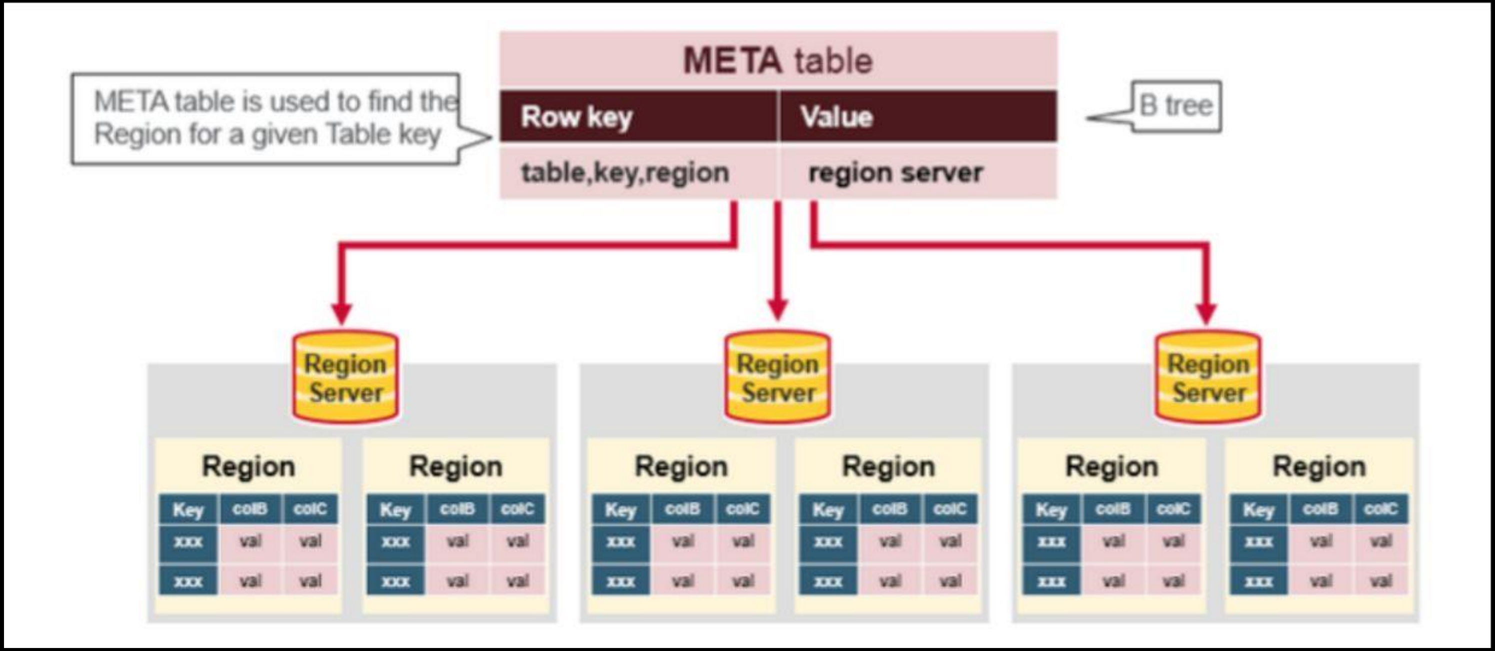
Architecture

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

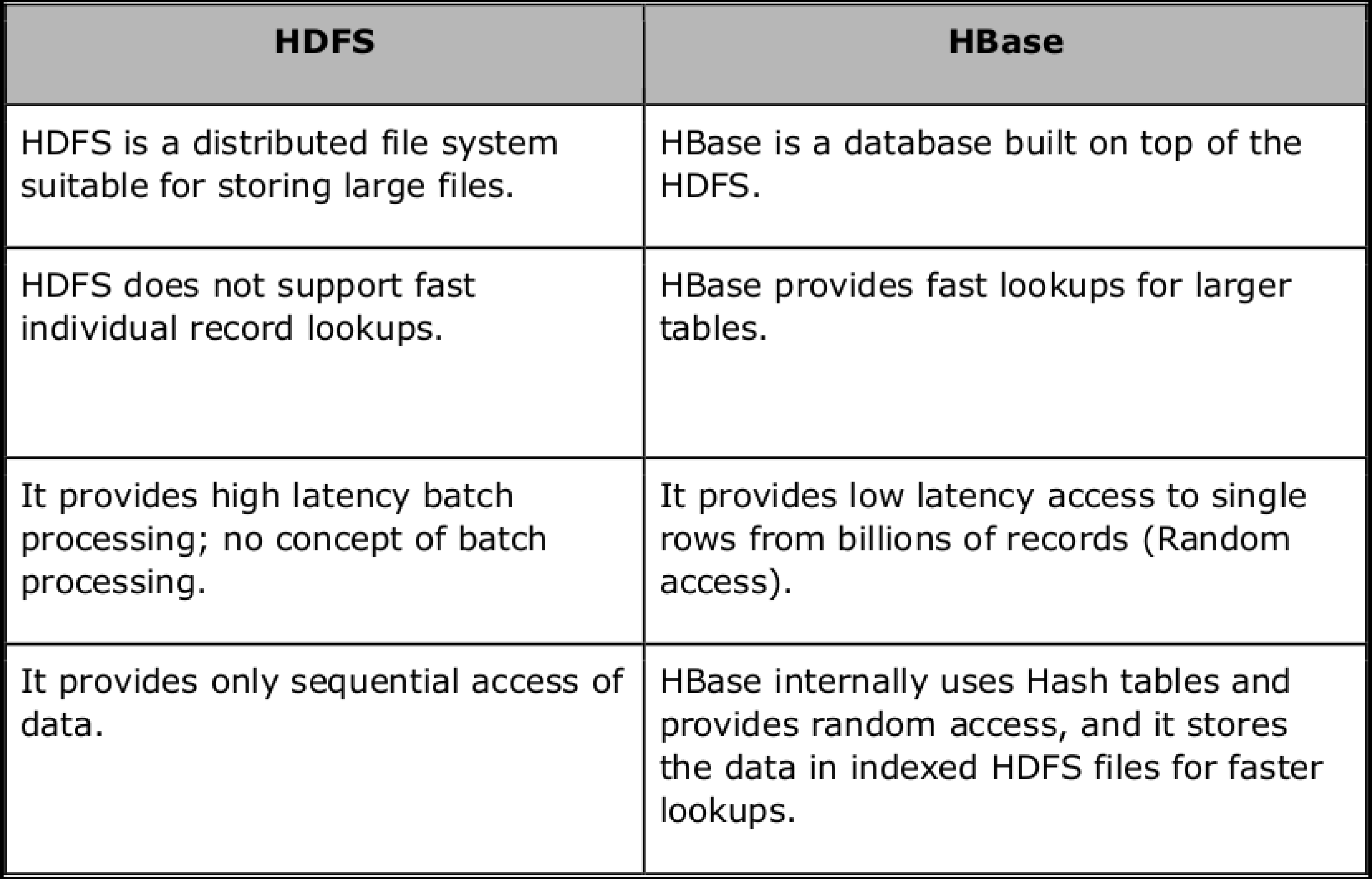
- ZooKeeper stores the location of the META table.
- First time a client reads or writes to HBase, the client gets the RegionServer that hosts the META table from ZooKeeper
- The client will query the META server to get the RegionServer corresponding to the row key it wants to access
    - The client caches this information along with the META table location
    - For future reads, the client uses the cache to retrieve the META location and previously read row keys
    - It will then get the row from the corresponding RegionServer

Architecture

The **META table** is an HBase table that keeps a list of all regions in the system

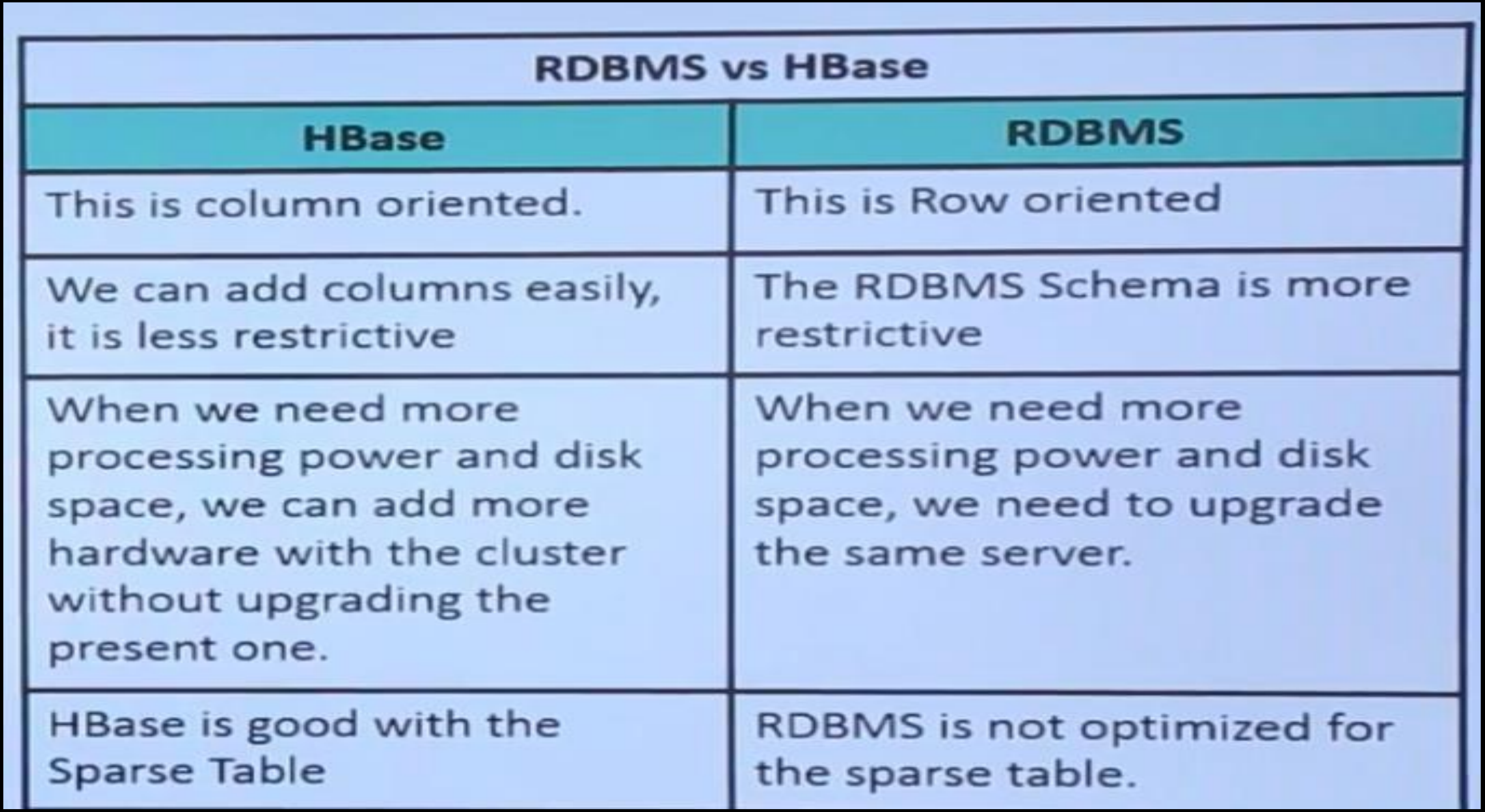


HDFS vs. HBase



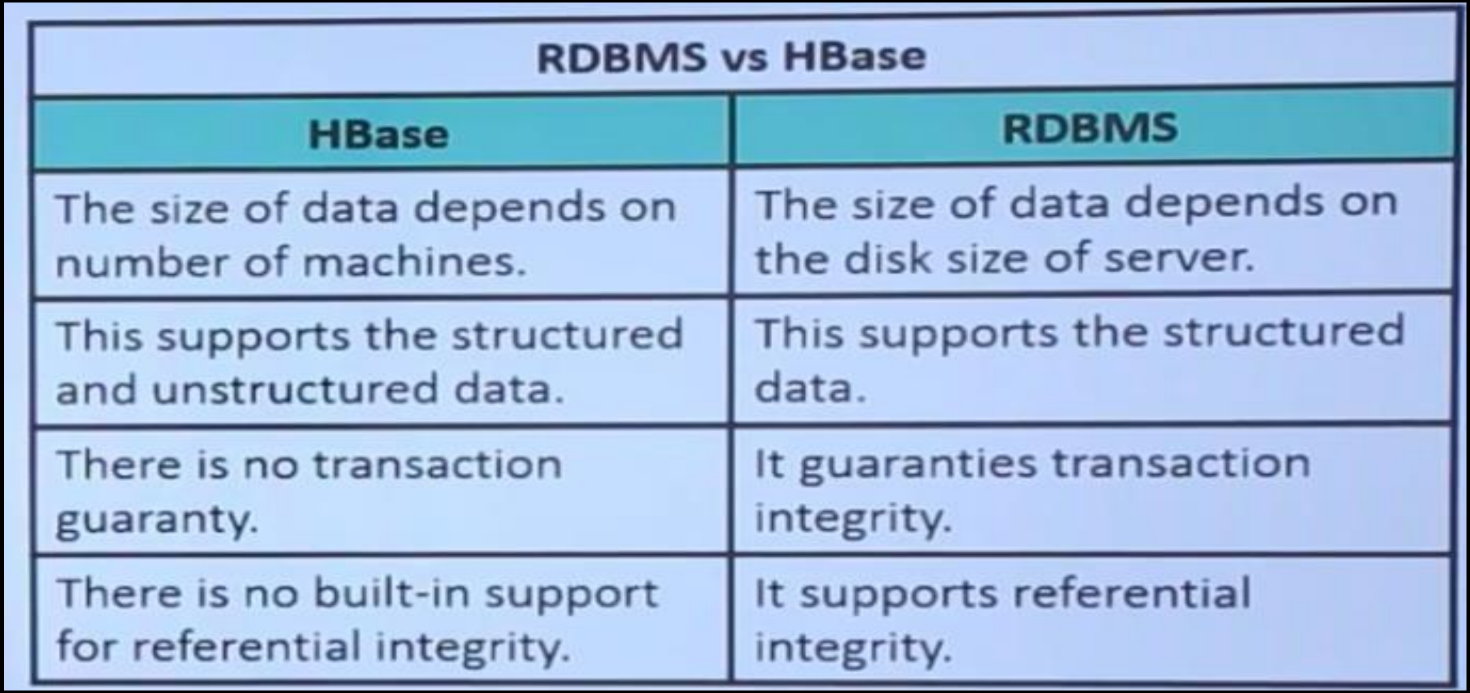
HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

DIFFERENCE BETWEEN RDBMS & HBase



RDBMS vs HBase	
HBase	**RDBMS**
This is column oriented.	This is Row oriented
We can add columns easily, it is less restrictive	The RDBMS Schema is more restrictive
When we need more processing power and disk space, we can add more hardware with the cluster without upgrading the present one.	When we need more processing power and disk space, we need to upgrade the same server.
HBase is good with the Sparse Table	RDBMS is not optimized for the sparse table.

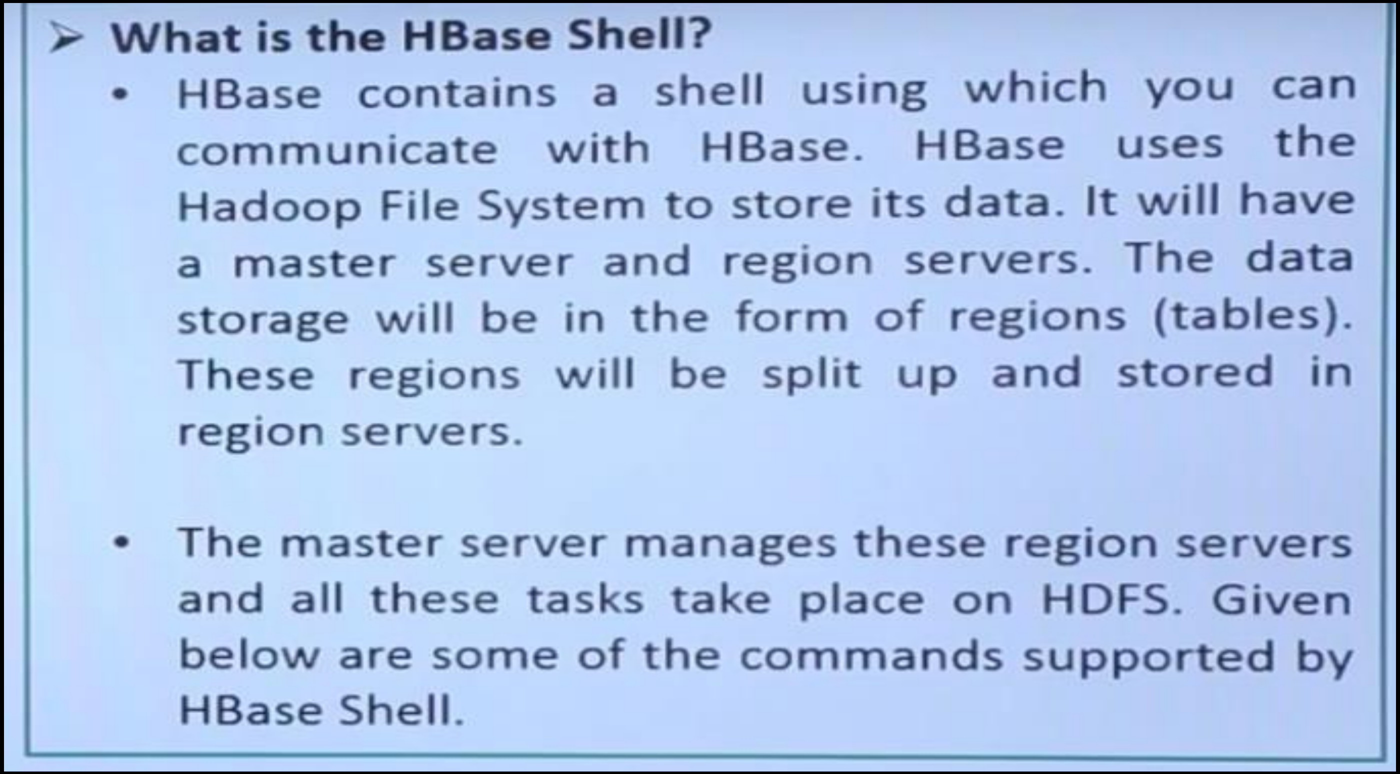
DIFFERENCE BETWEEN RDBMS & HBase



RDBMS vs HBase

HBase	RDBMS
The size of data depends on number of machines.	The size of data depends on the disk size of server.
This supports the structured and unstructured data.	This supports the structured data.
There is no transaction guaranty.	It guaranties transaction integrity.
There is no built-in support for referential integrity.	It supports referential integrity.

HBase Shell

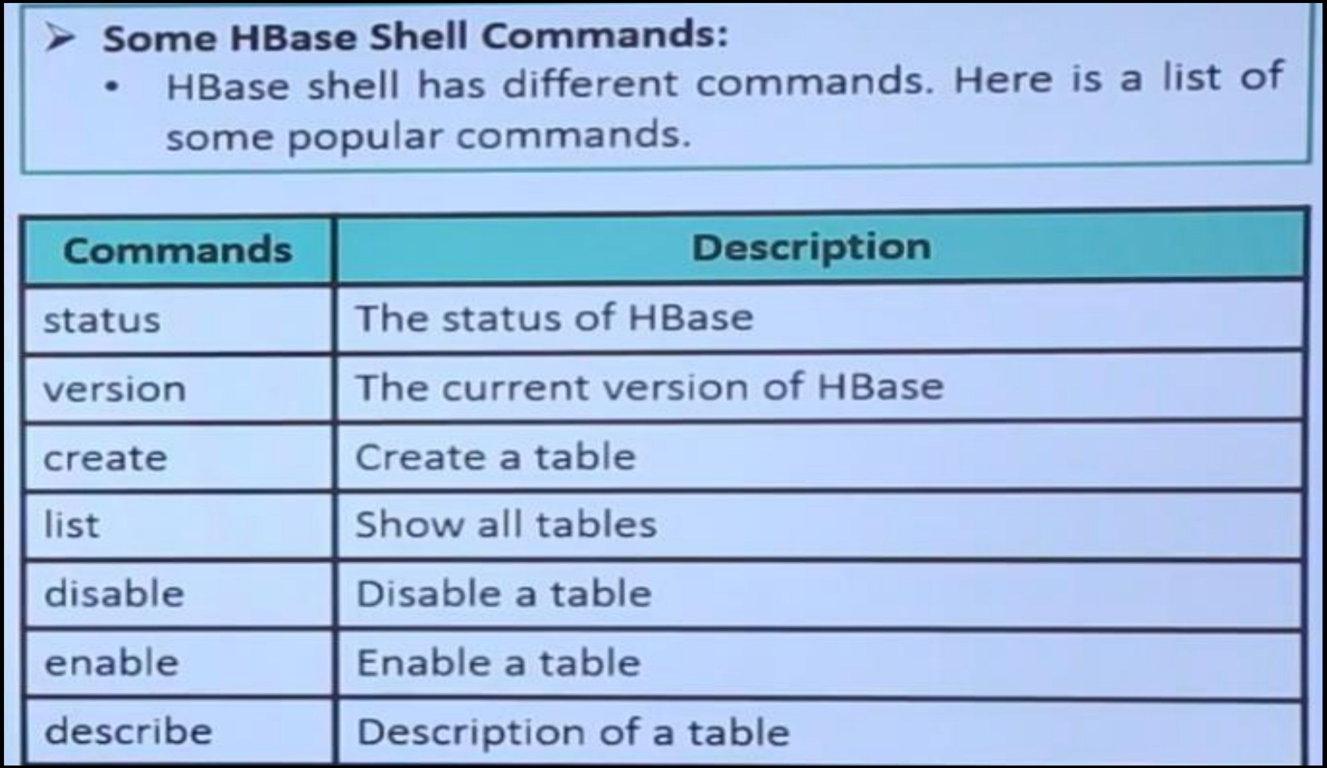


➢ **What is the HBase Shell?**

- HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

- The master server manages these region servers and all these tasks take place on HDFS. Given below are some of the commands supported by HBase Shell.

HBase Shell

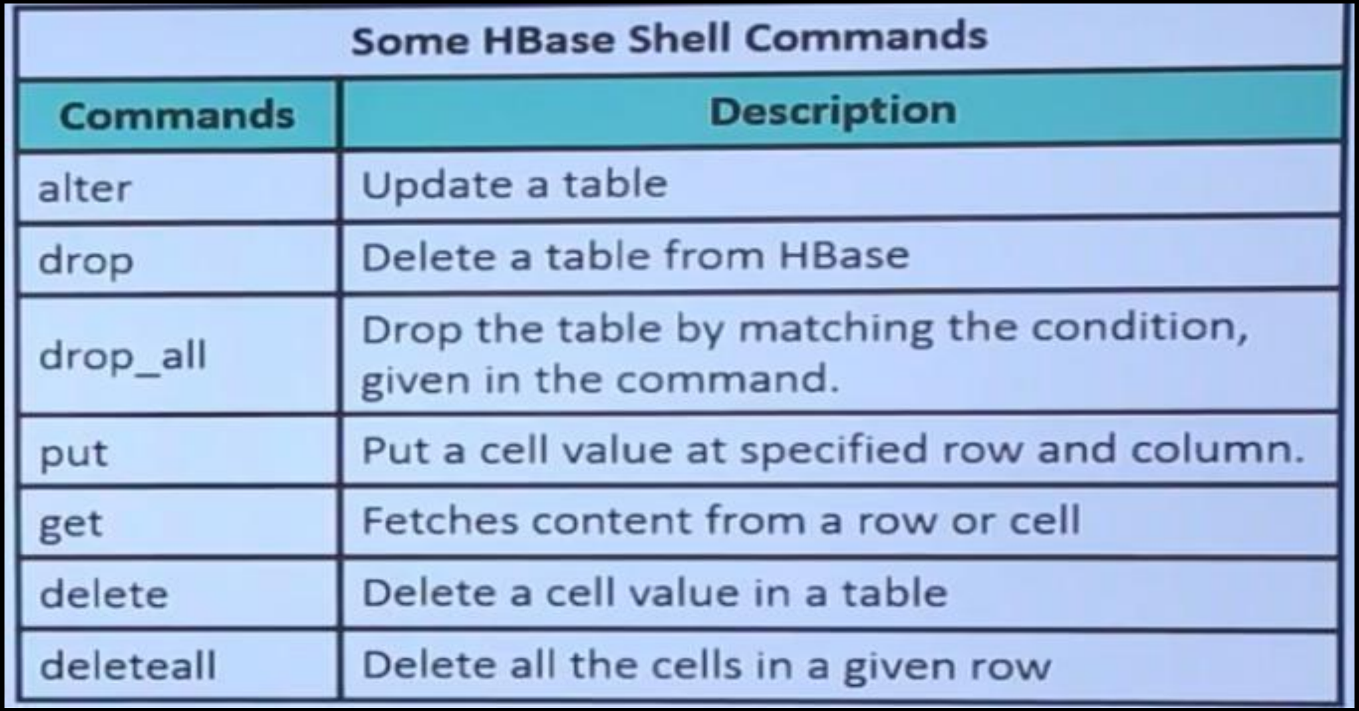


➢ **Some HBase Shell Commands:**

- HBase shell has different commands. Here is a list of some popular commands.

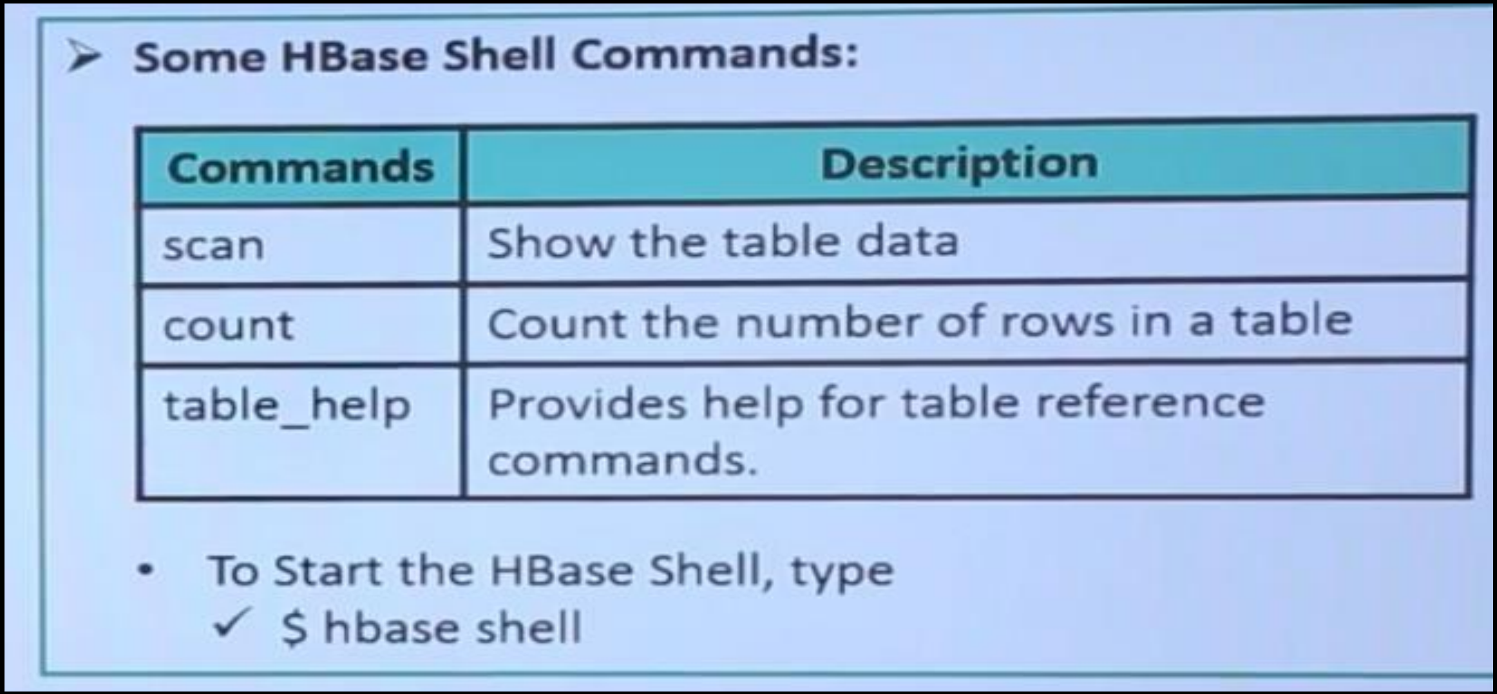
Commands	Description
status	The status of HBase
version	The current version of HBase
create	Create a table
list	Show all tables
disable	Disable a table
enable	Enable a table
describe	Description of a table

HBase Shell



Some HBase Shell Commands	
Commands	**Description**
alter	Update a table
drop	Delete a table from HBase
drop_all	Drop the table by matching the condition, given in the command.
put	Put a cell value at specified row and column.
get	Fetches content from a row or cell
delete	Delete a cell value in a table
deleteall	Delete all the cells in a given row

HBase Shell

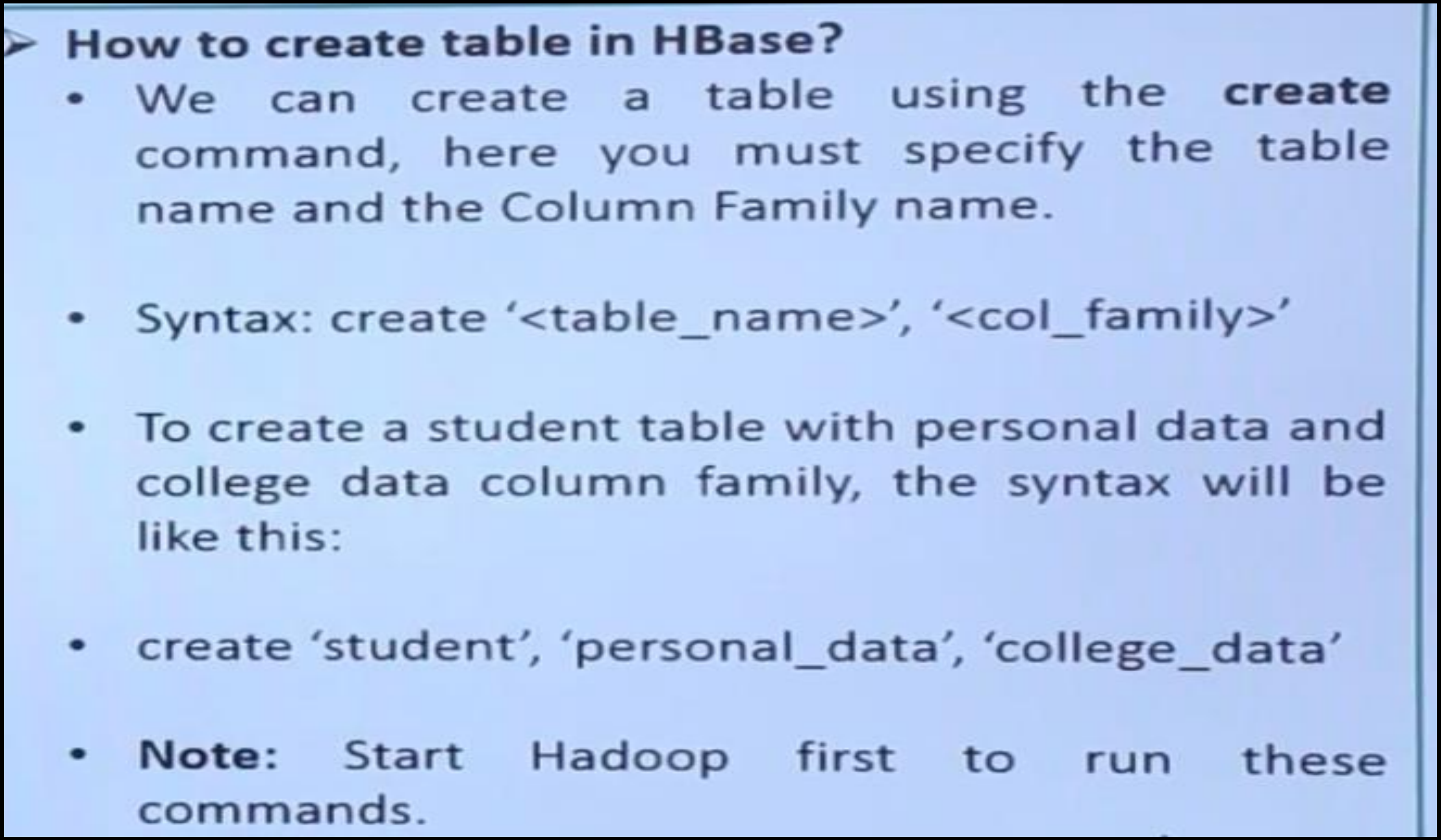


➢ **Some HBase Shell Commands:**

Commands	Description
scan	Show the table data
count	Count the number of rows in a table
table_help	Provides help for table reference commands.

- To Start the HBase Shell, type
  - ✓ $ hbase shell

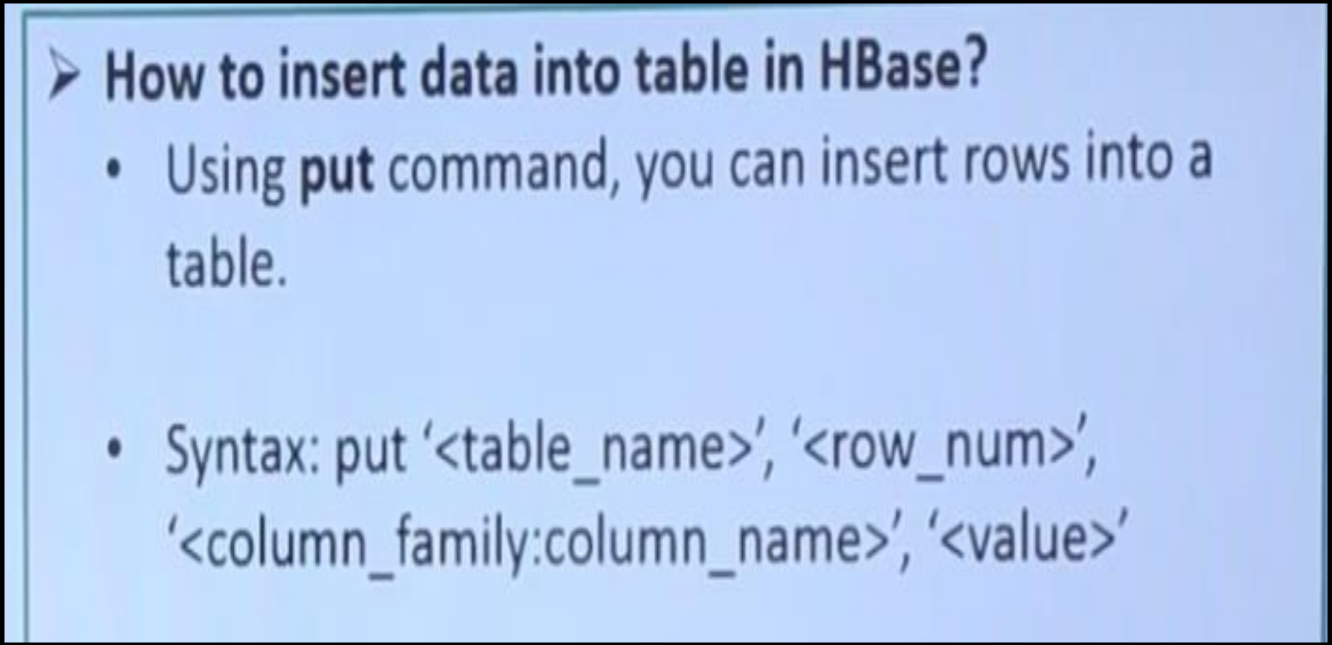
Table Creation and Data Insertion



➢ **How to create table in HBase?**

- We can create a table using the **create** command, here you must specify the table name and the Column Family name.
- Syntax: create '<table_name>', '<col_family>'
- To create a student table with personal data and college data column family, the syntax will be like this:
- create 'student', 'personal_data', 'college_data'
- **Note:** Start Hadoop first to run these commands.

Table Creation and Data Insertion

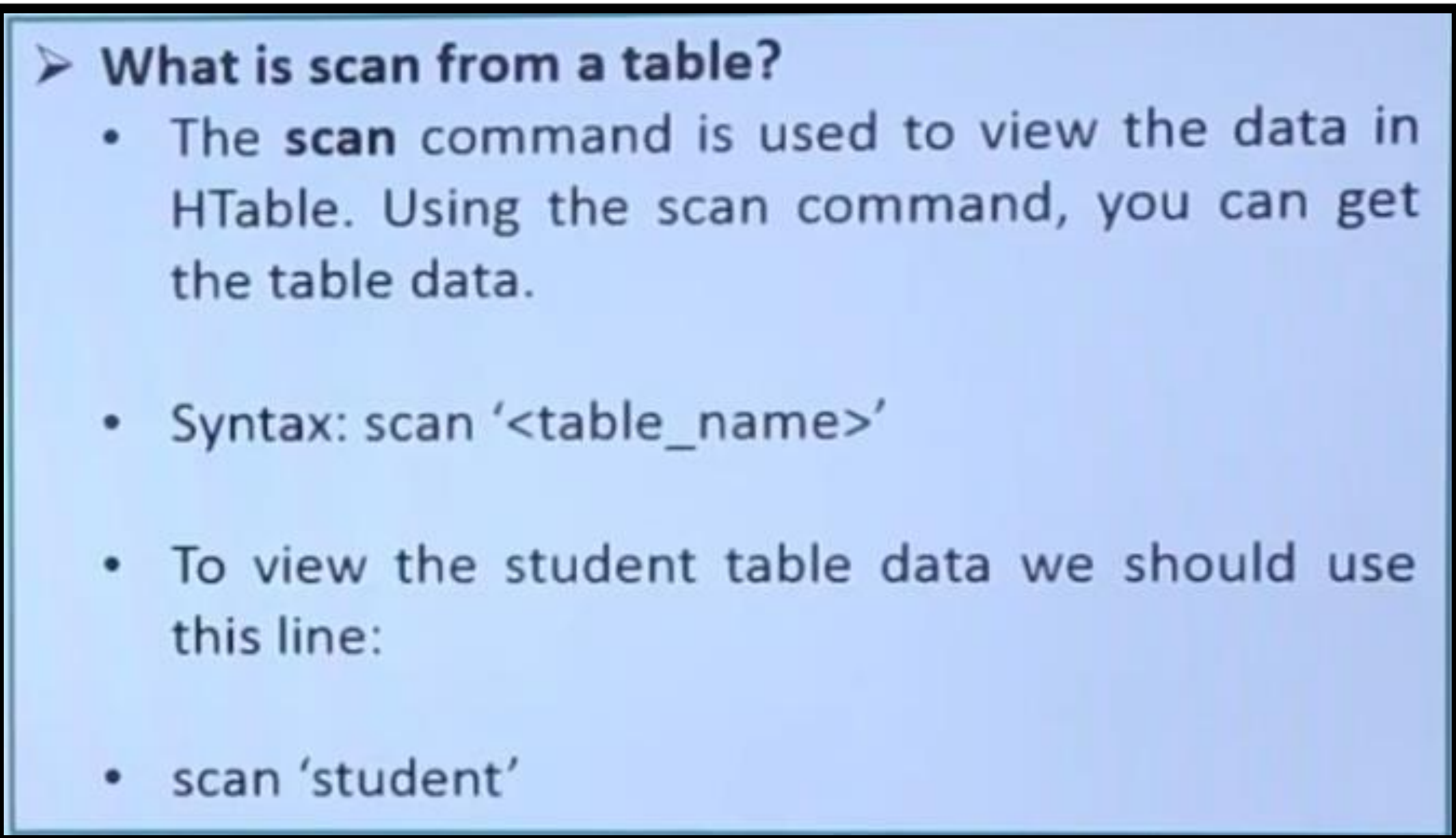


➢ **How to insert data into table in HBase?**

- Using **put** command, you can insert rows into a table.

- Syntax: put '<table_name>', '<row_num>', '<column_family:column_name>', '<value>'

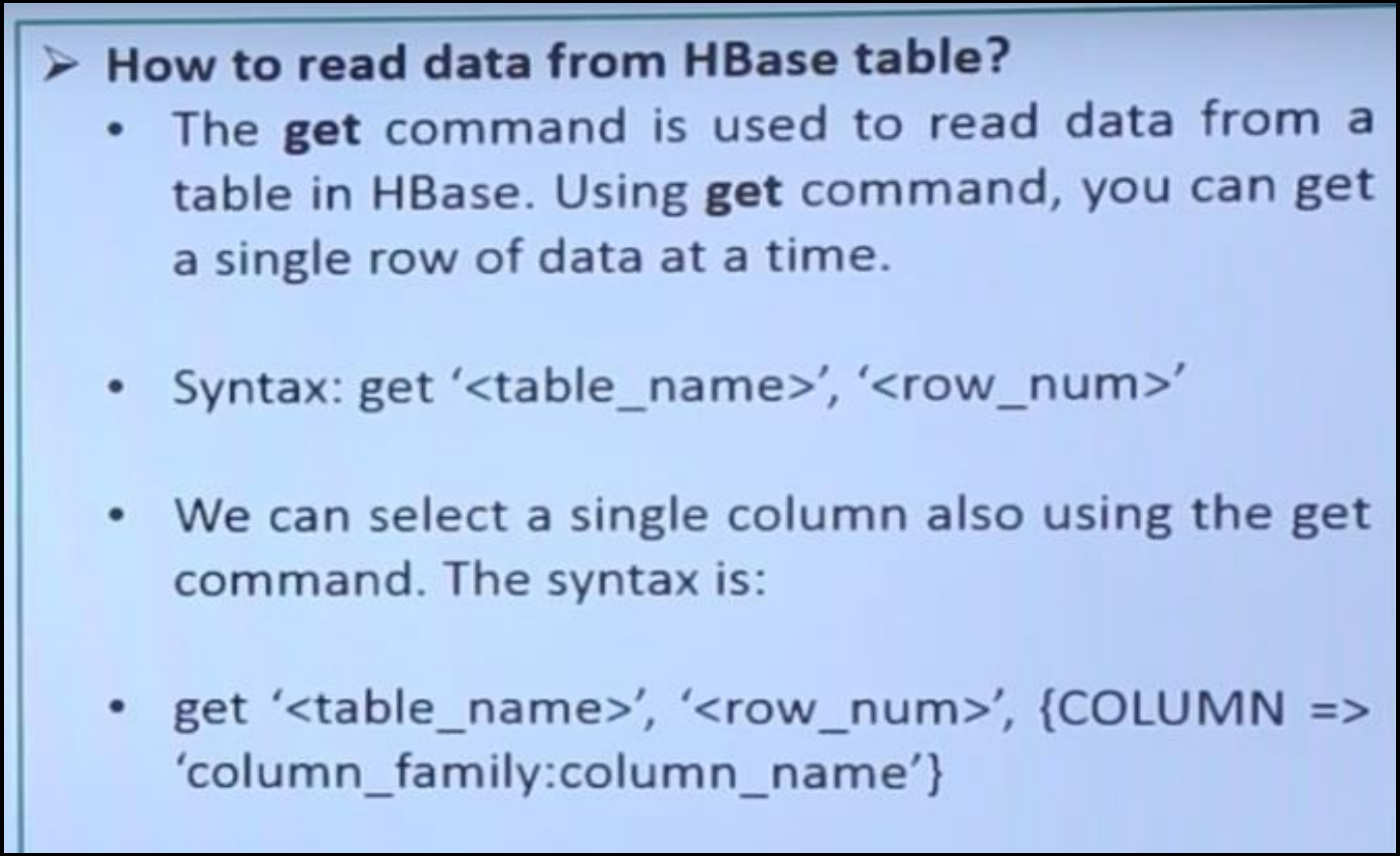
Data Scan and Read in HBase



➢ **What is scan from a table?**

- The **scan** command is used to view the data in HTable. Using the scan command, you can get the table data.
- Syntax: scan '<table_name>'
- To view the student table data we should use this line:
- scan 'student'

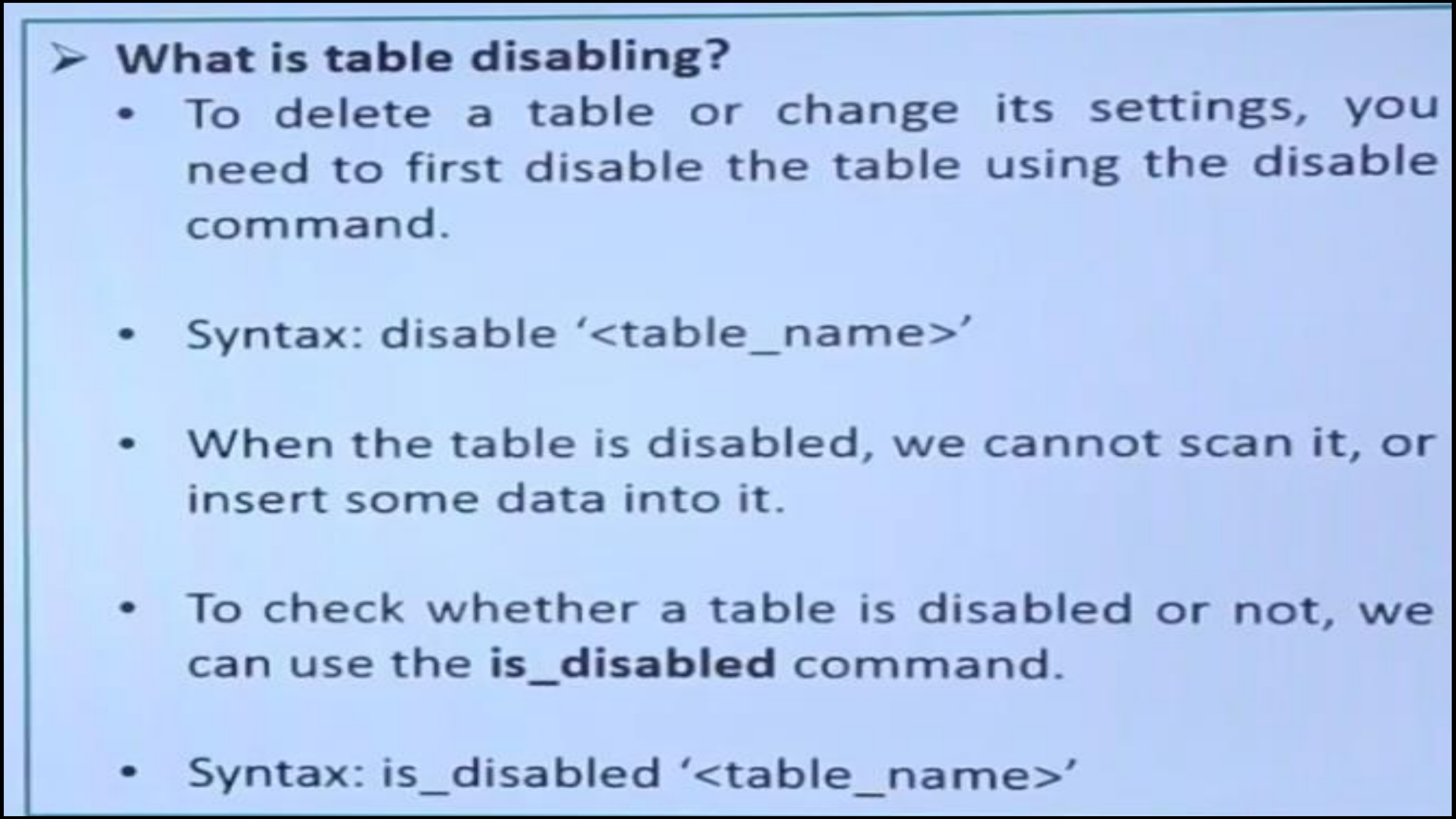
Data Scan and Read in HBase



➢ **How to read data from HBase table?**

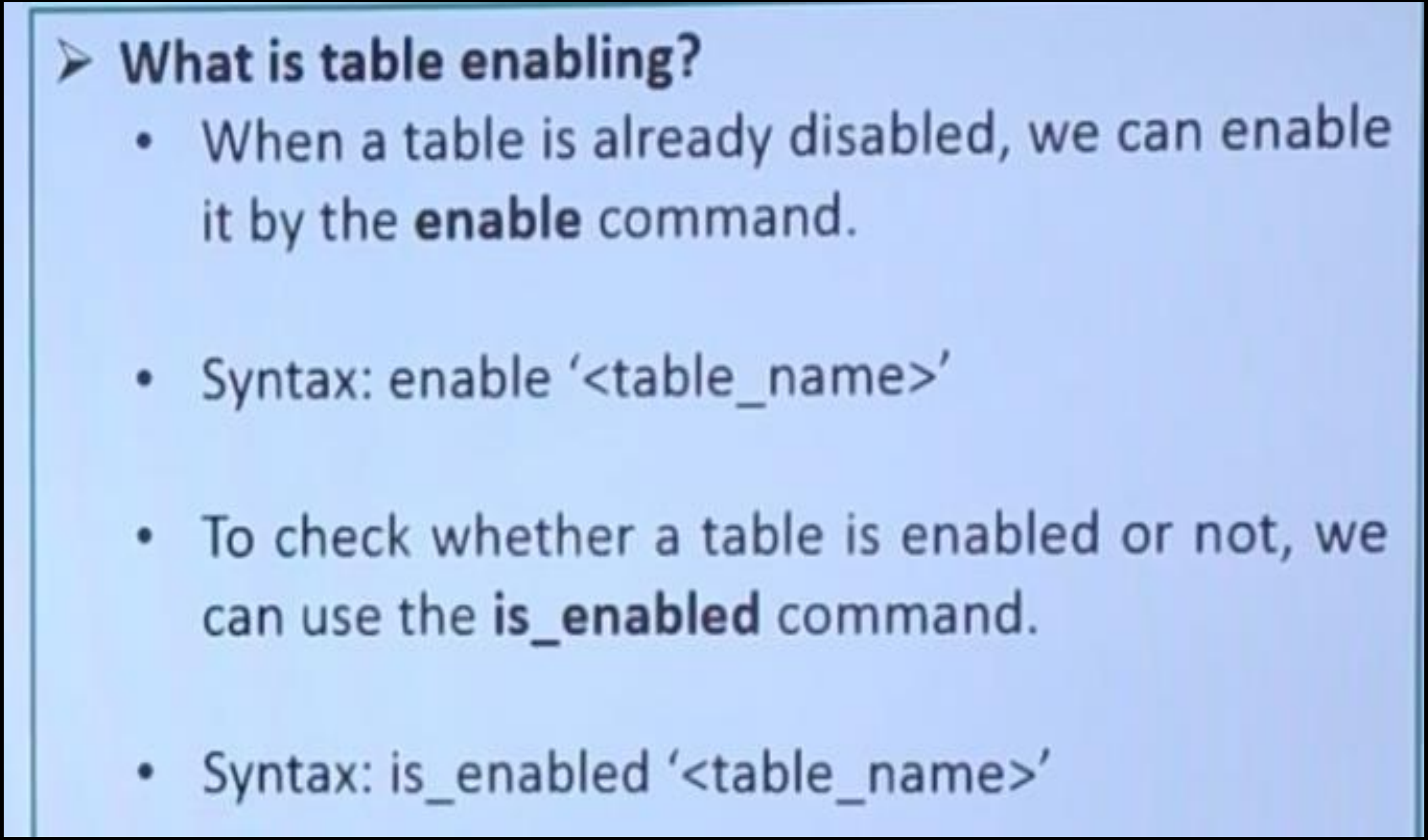
- The **get** command is used to read data from a table in HBase. Using **get** command, you can get a single row of data at a time.
- Syntax: get '<table_name>', '<row_num>'
- We can select a single column also using the get command. The syntax is:
- get '<table_name>', '<row_num>', {COLUMN => 'column_family:column_name'}

Disable and Disabling in HBase



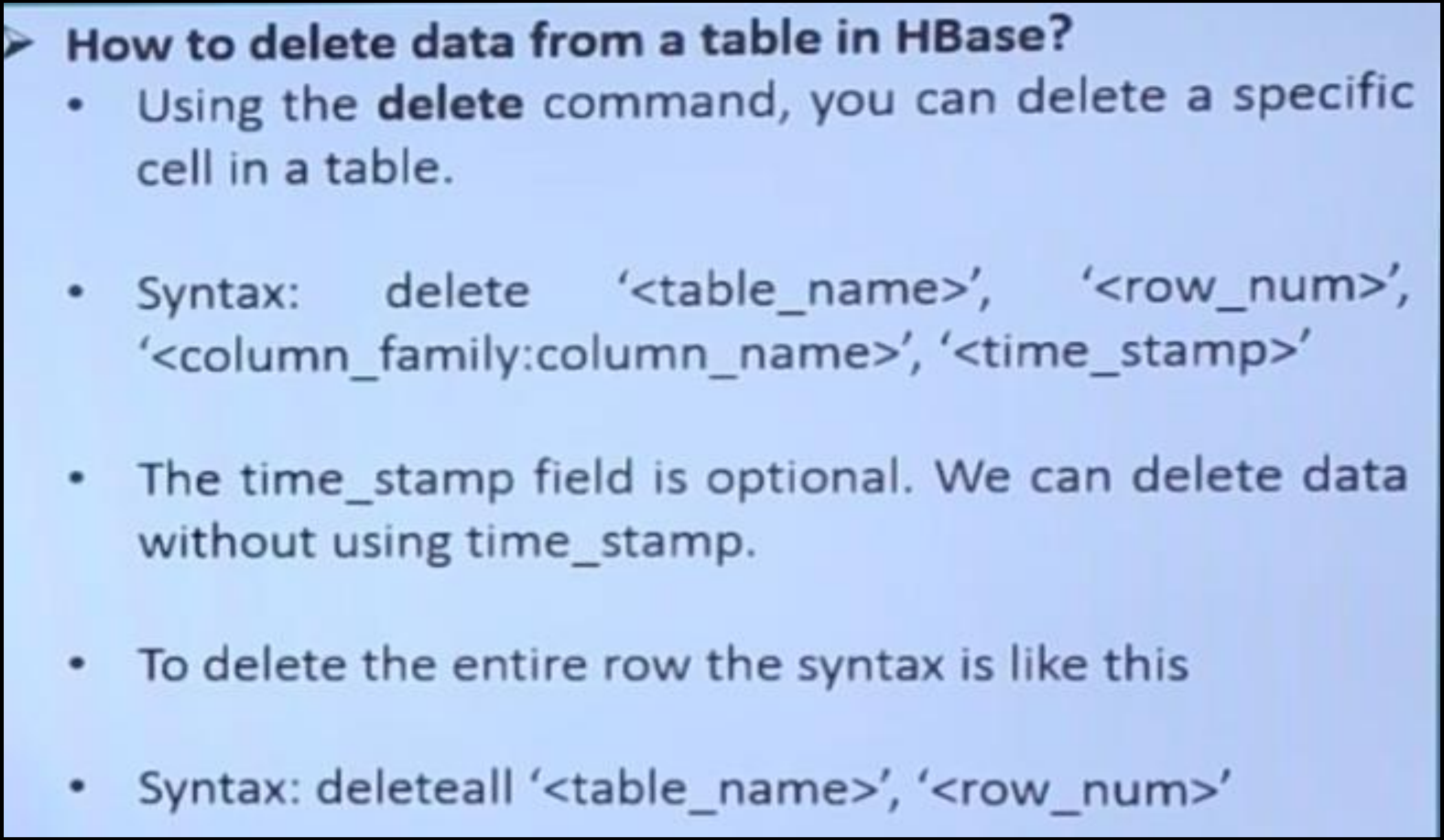
- **What is table disabling?**
  - To delete a table or change its settings, you need to first disable the table using the disable command.
  - Syntax: disable '<table_name>'
  - When the table is disabled, we cannot scan it, or insert some data into it.
  - To check whether a table is disabled or not, we can use the **is_disabled** command.
  - Syntax: is_disabled '<table_name>'

Disable and Disabling in HBase



- **What is table enabling?**
  - When a table is already disabled, we can enable it by the **enable** command.
  - Syntax: enable '<table_name>'
  - To check whether a table is enabled or not, we can use the **is_enabled** command.
  - Syntax: is_enabled '<table_name>'

Data Deletion in HBase Table



➢ **How to delete data from a table in HBase?**

- Using the **delete** command, you can delete a specific cell in a table.
- Syntax: delete '<table_name>', '<row_num>', '<column_family:column_name>', '<time_stamp>'
- The time_stamp field is optional. We can delete data without using time_stamp.
- To delete the entire row the syntax is like this
- Syntax: deleteall '<table_name>', '<row_num>'

THANK YOU