BIG DATA

Columnar Databases for Analytics

K V Subramaniam
Computer Science and Engineering

Hadoop Available Storage Types

HDFS and it's limitations

HDFS is good for batch processing (scans over big files)

Limitations

- Not good for record lookup
- Not good for incremental addition of small batches
- Not good for updates

BIG DATA

HIVE and it's limitations

- Doesn't store data
  - Uses HDFS or Hbase as actual store
- Provides an SQL Interface for querying data
- Data must be Structured: definite schema

Limitations

- Not good for record lookup
- Not good for incremental addition of small batches
- Not good for updates
- Not good for unstructured/semistructured data

BIG DATA

Hbase and Cassandra

- Built on the BigTable data model
- different in architecture

Advantages

- Fast record lookup
- Record level insertion
- Support for updates (Hbase creates new versions)
- Support for unstructured/semistructured data

Chang, Fay, et al. "Bigtable: A distributed storage system for structured data."*ACM Transactions on Computer Systems (TOCS)* 26.2 (2008): 4.

BIG DATA

Use case for different storage types

- HDFS
  - Unstructured data
  - Writes: no updates, only appends
  - Read entire file and analyze
- Hive
  - Structured data
  - Analytics via SQL
- HBase/Cassandra
  - Unstructured data
  - Arbitrary writes
  - Analytics

BIG DATA

Exercise

Which of these could be stored in HDFS, Hive or Hbase?

- Parsed transaction logs of user activity in a website where relevant fields from the log have been extracted
- Unparsed transaction logs of user activity
- Database of users and friends at a social website, which is periodically analyzed for social networking analysis

BIG DATA

Solution

Which of these could be stored in HDFS, Hive or Hbase?

- Parsed transaction logs of user activity in a website where relevant fields from the log have been extracted: **HIVE**
- Unparsed transaction logs of user activity : **HDFS**
- Database of users and friends at a social website, which is periodically analyzed for social networking analysis : **HBASE**

Columnar Storage

BIG DATA

Motivational Example – storage in DBMS

Row Key	Info:height	Info:age	School:House	School:sports
HarryPotter	4.5ft	11	Gryffindor	Quidditch
Voldemort	7ft	50	Slytherin	

- Row storage: DB is stored as a single file, one row per line
- Column storage: each column is a separate file, one value per line
- For using data, we need to perform an I/O to load data from disk
- Which method does less I/O for
  - Analyzing the relationship between age and earnings
    - Column storage
  - Adding a new row or read a row
    - Row storage

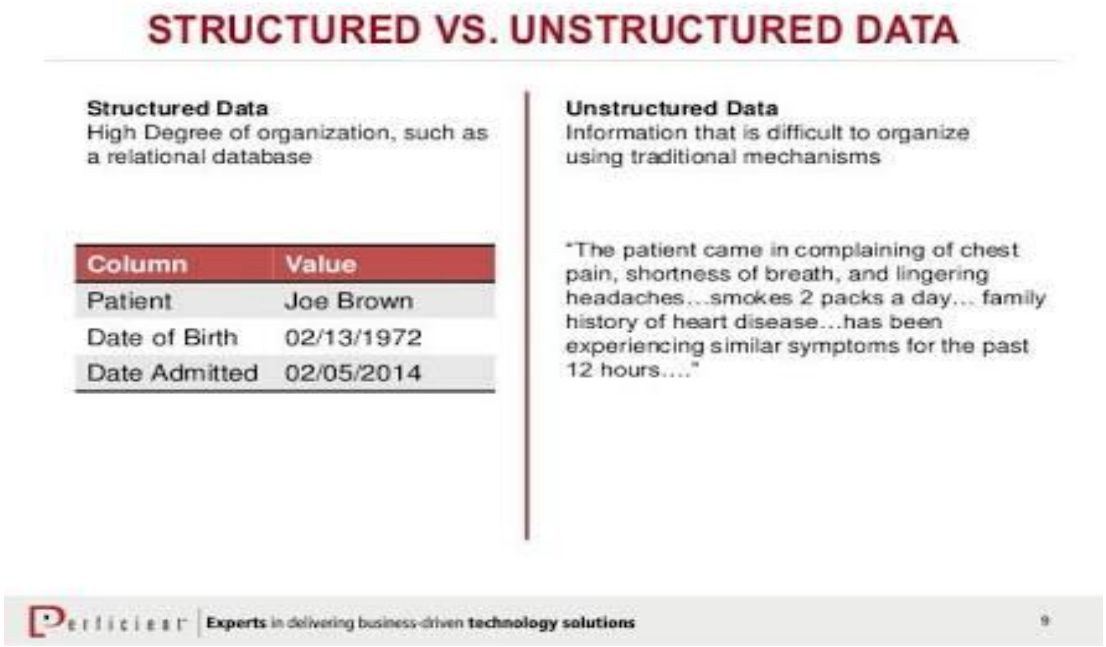
BIG DATA

History

- The first use of dbs were for transactions
  - Read a person's bank balance
  - Update bank balance
  - Row storage used since more efficient for transactions
- Column dbs
  - Became popular with Big Data systems
  - More efficient for analytics, particularly if db is large
  - To handle unstructured data

BIG DATA

Unstructured data



- How can the unstructured data above be stored in a structured relational db?
- How can it be stored in a unstructured db?

https://www.slideshare.net/perficientinc/ibm-watson-content-analytics-discover-hidden-value-in-your-unstructured-data

BIG DATA

Unstructured data

Customer id	Visit id	Date
		2-Oct 2017

Customer id	Visit id	Symptom id	Symptom
		1	Chest pain
		2	Headache

Structured db

Customer id	Visit id	Info
		Date: {2-Oct 2017} Symptoms: {Chest pain, Headache}

Unstructured db: simpler, more efficient

Hbase and Cassandra Data Model

BIG DATA

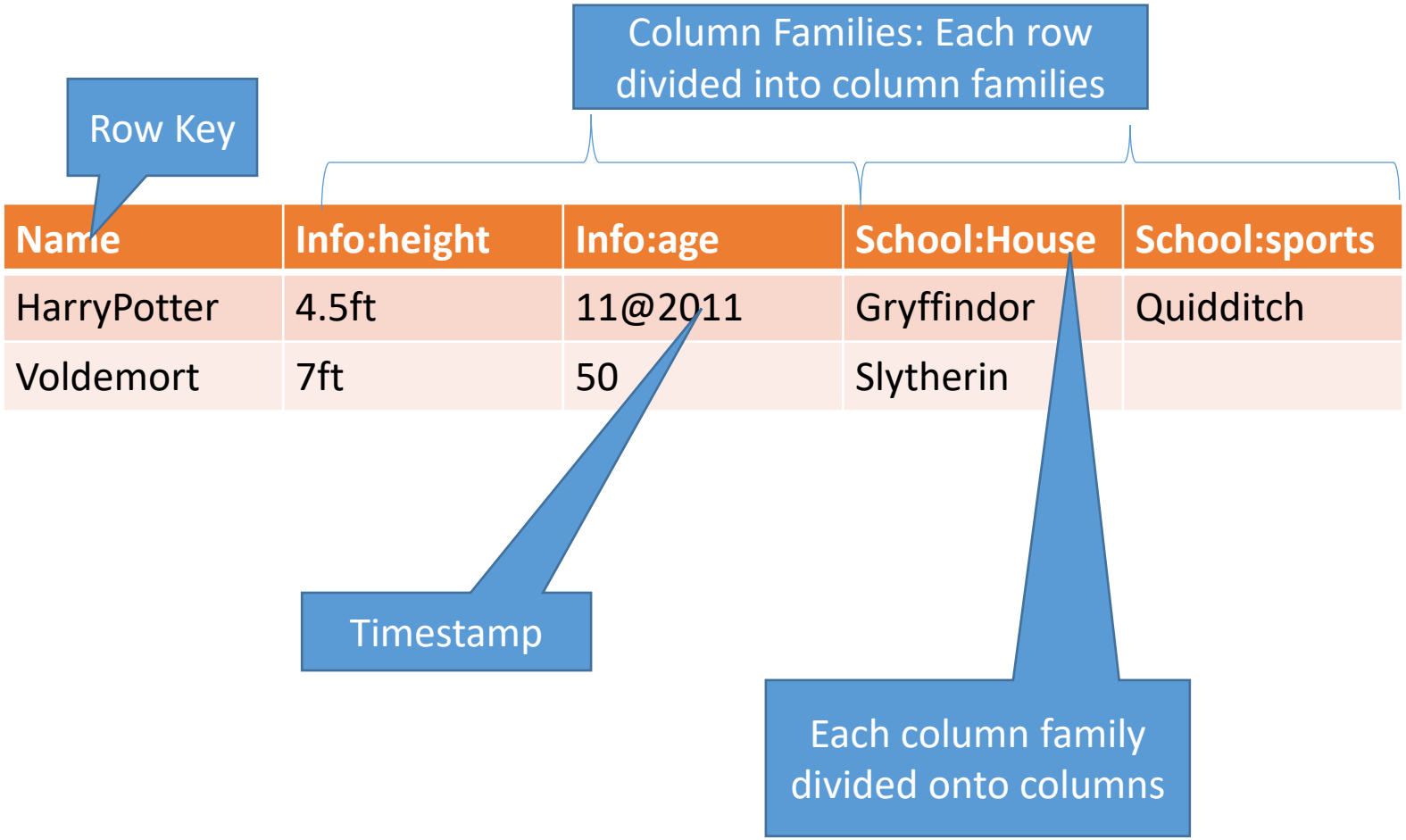
Hbase/Cassandra

- Hbase
  - Distributed column oriented database built on top of HDFS
  - Data is logically organized as rows/columns of a table
- Cassandra
  - Distributed database – peer to peer built by facebook
  - Inspired by Dynamo DB
  - Same data model as Hbase – inspired by BigTable

BIG DATA

Data Model – BigTable

- Key-Value pairs



BIG DATA

Column Families

HBase schema consists of several ***Tables***

Each table consists of a set of ***Column Families***

- Columns are not part of the schema

HBase has ***Dynamic Columns***

- Because column names are encoded inside the cells
- Different cells can have different columns

"School" column family has different columns in different cells

Name	Data
HarryPotter	Info:{height:"4.5ft", age: "11@2011"} School:{House:"Gryffindor", Sports:"Quidditch"}
Voldemort	Info:{height:"7ft", age: "50"} School:{House:"Sylitherin", Role:"Prefect"}

BIG DATA

Data Model – BigTable

Row Key	Data
HarryPotter	**Info**:{height:”4.5ft”, age: “11@2011”} **School**:{House:”Gryffindor”, Sports:”Quidditch”}
Voldemort	**Info**:{height:”7ft”, age: “50”} **School**:{House:”Sylytherin”, Role:”Prefect@1980, DarkLord@1995”}

Different types of data into different column families

Single column may have different values at different timestamps

BIG DATA

Data Model – BigTable

Column family named "anchor"

Column family named "Contents"

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>…"		
	t11	"<html>…"		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor:cnnsi.com"	"CNN"
	t13		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>…"		
	t5	"<html>…"		
	t3	"<html>…"		

Key

- Byte array
- Serves as the primary key for the table
- Indexed for fast lookup

Column Family

- Has a name (string)
- Contains one or more related columns

Column

- Belongs to one column family
- Included inside the row
  - ***familyName:columnName***

BIG DATA

Data Model – BigTable

Version Number

- Unique within each key
- By default→ System's timestamp
- Data type is Long

Value (Cell)

- Byte array (Hbase only)

Version number for each row

value

Row key	Time Stamp	Column "contents:"	Column "anchor:"	
"com.apache.www"	t12	"<html>…"		
	t11	"<html>…"		
	t10		"anchor:apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor:cnnsi.com"	"CNN"
	t13		"anchor:my.look.ca"	"CNN.com"
	t6	"<html>…"		
	t5	"<html>…"		
	t3	"<html>…"		

Hbase Architecture

Hbase Architecture – Master Slave

Region

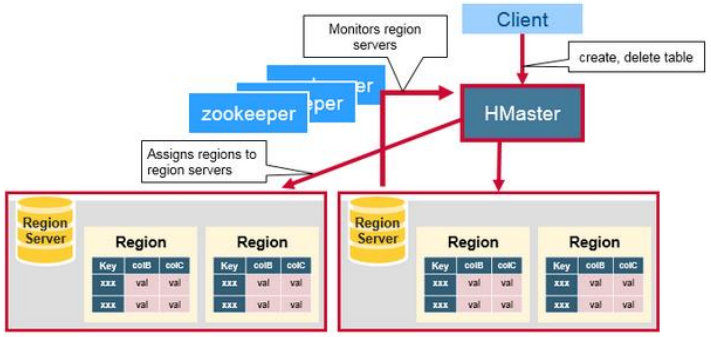
- A subset of a table's rows, like horizontal range partitioning
- Automatically done

RegionServer (many slaves)

- Manages data regions
- Serves data for reads and writes (***using a log***)
- Like datanode of HDFS

Master

- Responsible for coordinating the slaves
- Assigns regions, detects failures
- Admin functions
- Line namenode of HDFS



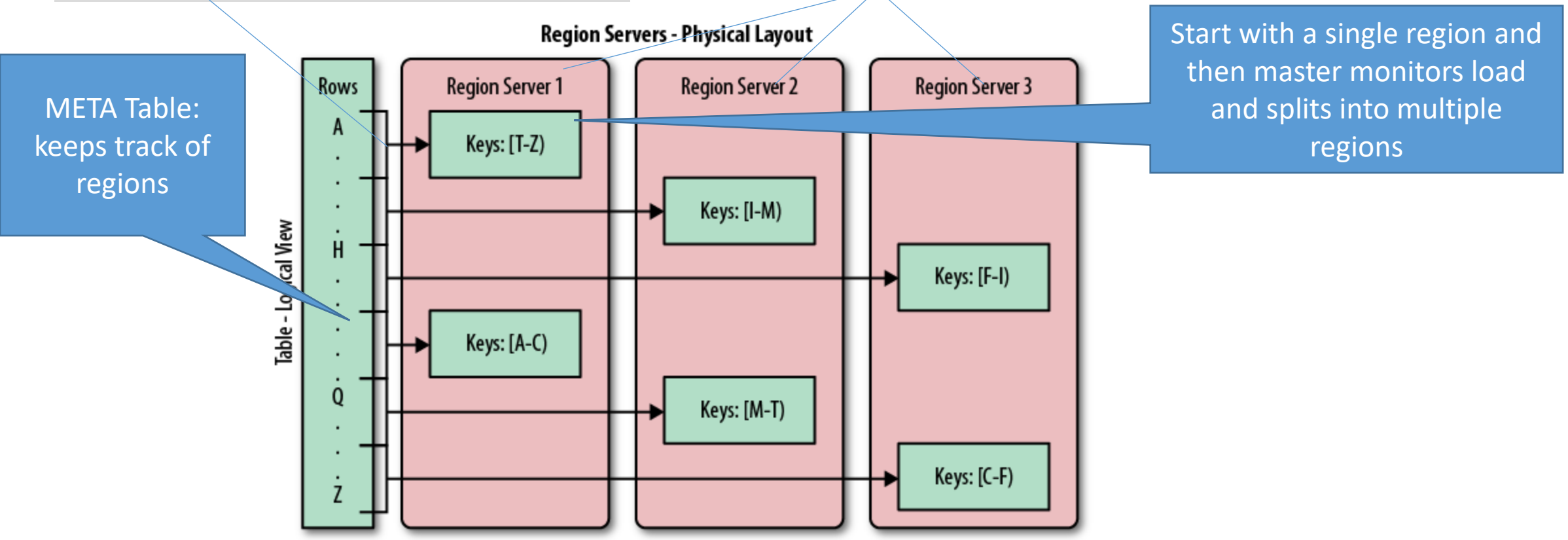
Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

BIG DATA

Regions and region servers

Tables horizontally partitioned into regions

Regions stored on region servers



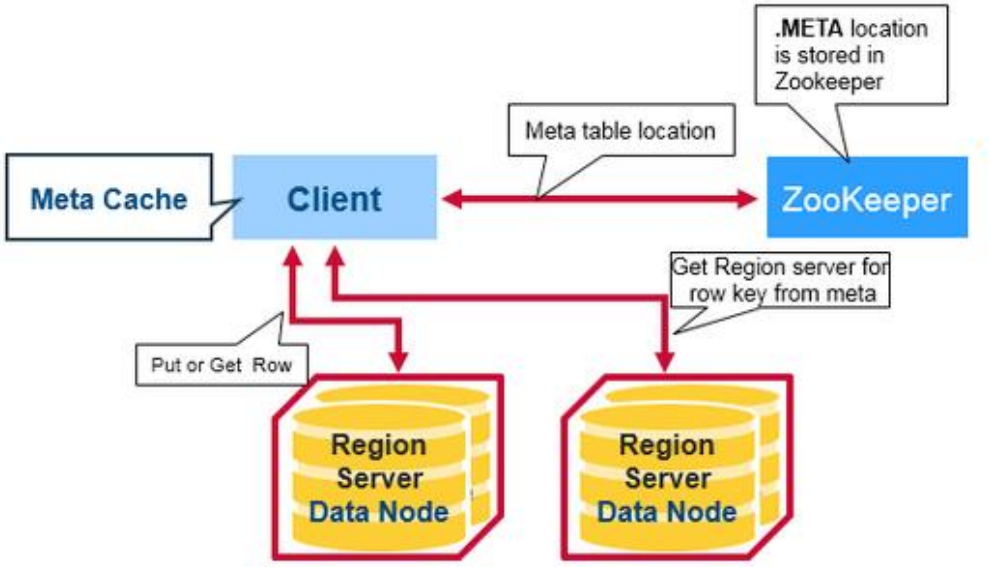
META Table: keeps track of regions

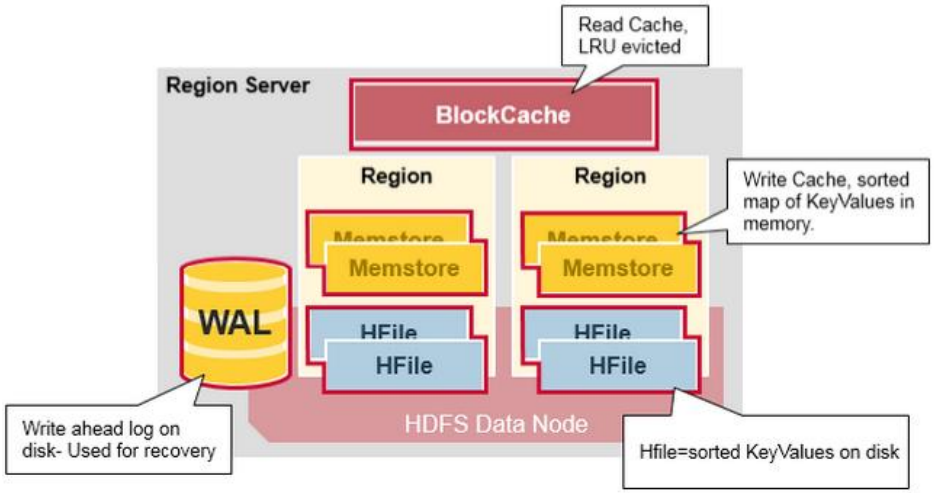
Start with a single region and then master monitors load and splits into multiple regions

Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

BIG DATA

Read/Write Operation





Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

Cassandra Architecture

Cassandra Architecture – peer to peer architecture

- Differences from Hbase
  - Request coordination over a partitioned dataset – no Master
  - Ring membership and failure detection – no Master
  - Local persistence (storage) engine – does not rely on HDFS
- Cqlsh – for performing queries

Refer https://www.mapr.com/blog/in-depth-look-hbase-architecture

Hbase usage

Hbase: Creating a new table

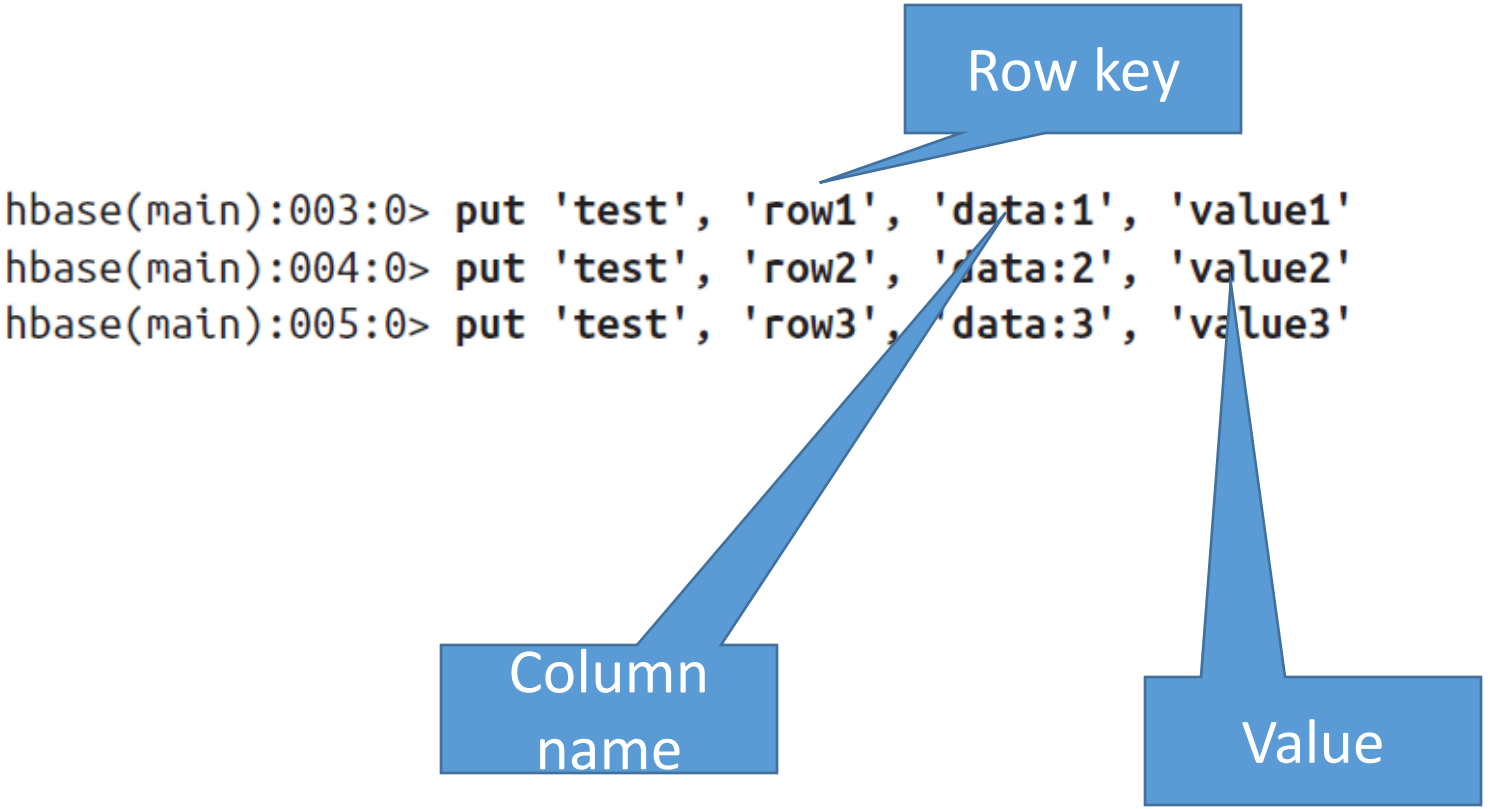
```
hbase(main):001:0> create 'test', 'data'
0 row(s) in 0.9810 seconds
```

Column family name

Table name

BIG DATA

Hbase inserting data



BIG DATA

Hbase: retrieving data

Getting a specific row

```
hbase(main):006:0> get 'test', 'row1'
COLUMN                          CELL
 data:1                         timestamp=1414927084811, value=value1
1 row(s) in 0.0240 seconds
hbase(main):007:0> scan 'test'
ROW                             COLUMN+CELL
 row1                           column=data:1, timestamp=1414927084811, value=value1
```

All rows

THANK YOU

K V Subramaniam
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu