Big Data Training

Jnaneshwar Bohara

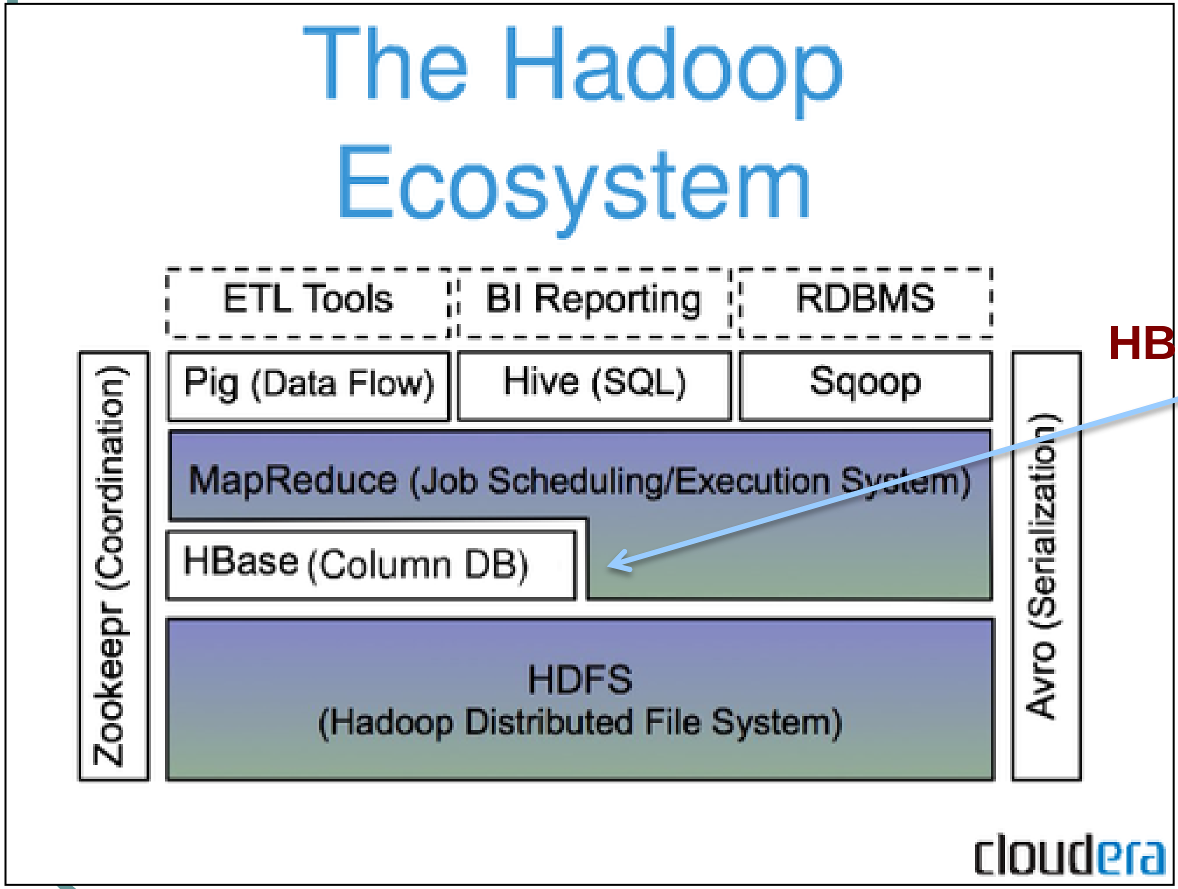
HBase

APACHE HBASE

HBase: Overview

- HBase is a distributed column-oriented data store built on top of HDFS
- HBase is an Apache open source project whose goal is to provide storage for the Hadoop Distributed Computing
- Data is logically organized into tables, rows and columns

HBase: Part of Hadoop's Ecosystem



HBase is built on top of HDFS

HBase files are internally stored in HDFS

HBase vs. HDFS

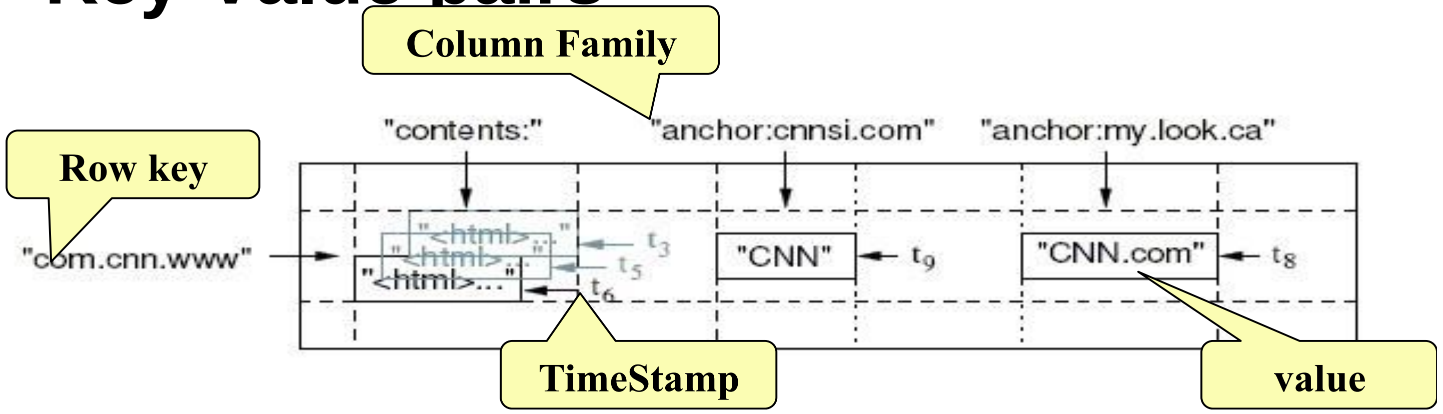
- Both are distributed systems that scale to hundreds or thousands of nodes
- ***HDFS*** is good for batch processing (scans over big files)
    - Not good for record lookup
    - Not good for incremental addition of small batches
    - Not good for updates

HBase vs. HDFS (Cont'd)

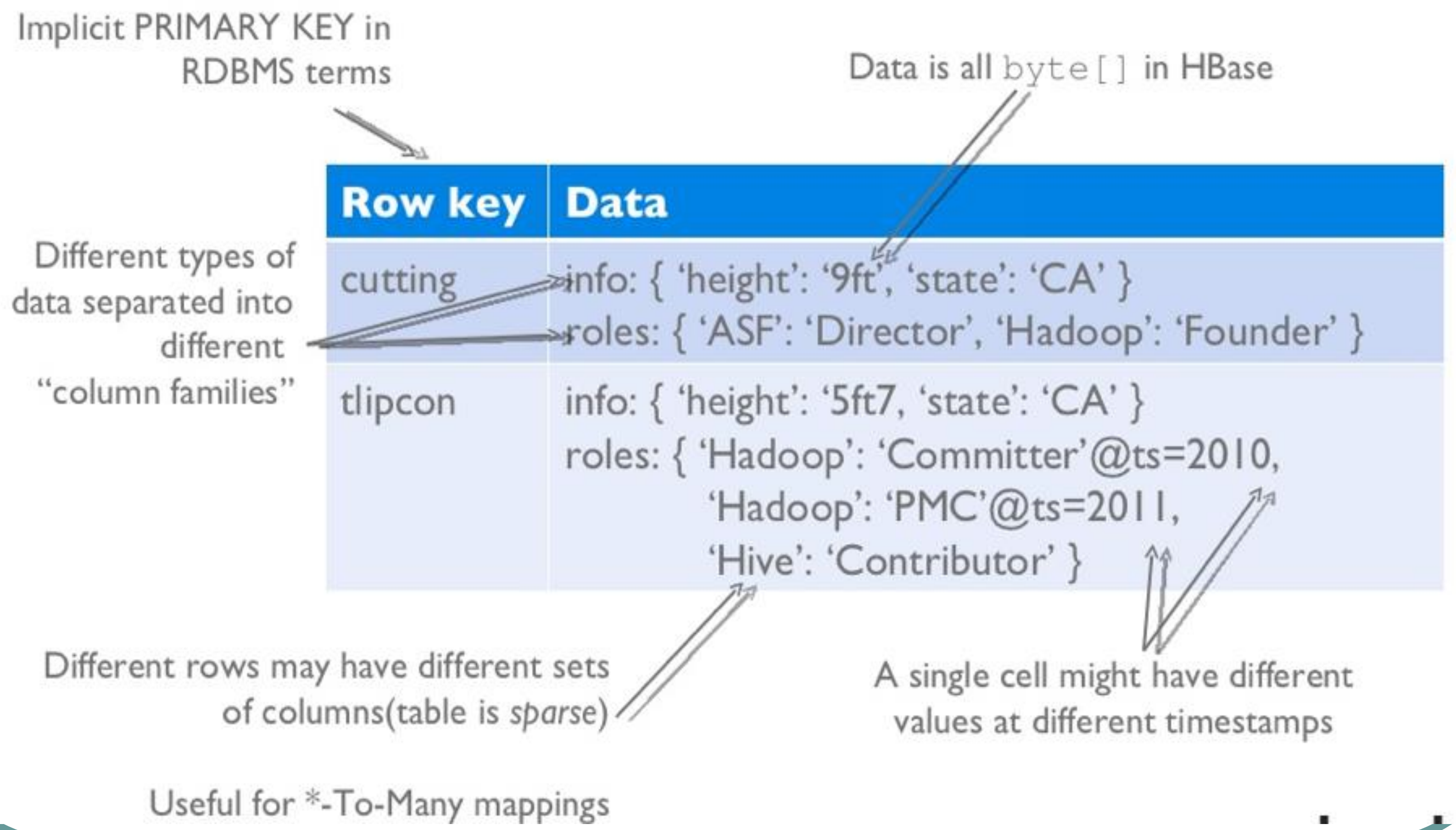
- ***HBase*** is designed to efficiently address the above points
  - Fast record lookup
  - Support for record-level insertion
  - Support for updates (not in place)
- HBase updates are done by creating new versions of values

HBase Data Model

- **HBase is based on Google's Bigtable model**
  - **Key-Value pairs**



HBase Logical View


Implicit PRIMARY KEY in RDBMS terms
Data is all byte[] in HBase
Different types of data separated into different "column families"
Different rows may have different sets of columns(table is *sparse*)
A single cell might have different values at different timestamps
Useful for *-To-Many mappings

Row key	Data

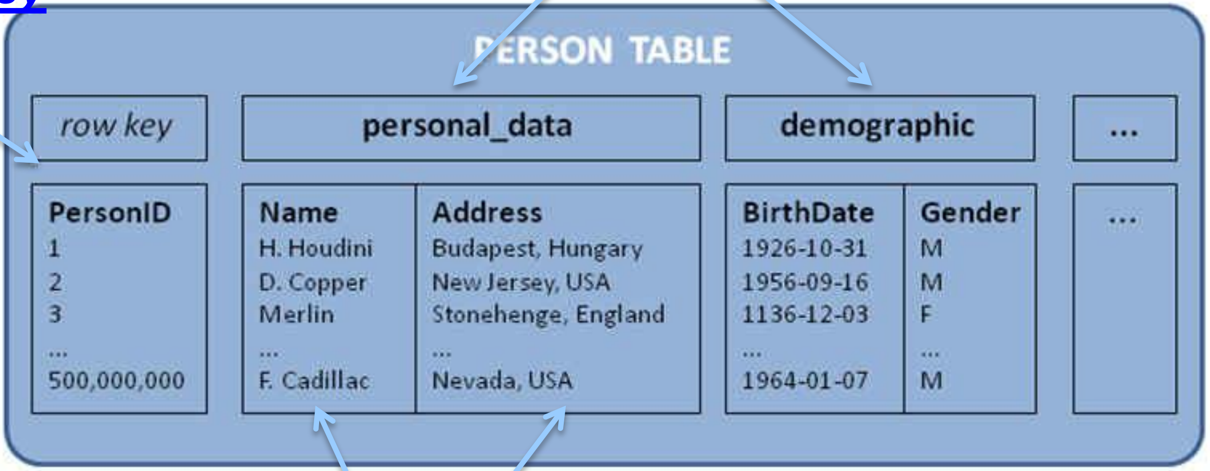
cutting	info: { 'height': '9ft', 'state': 'CA' }
	roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }

tlipcon	info: { 'height': '5ft7, 'state': 'CA' }
	roles: { 'Hadoop': 'Committer'@ts=2010,
	'Hadoop': 'PMC'@ts=2011,
	'Hive': 'Contributor' }

HBase: Keys and Column Families

Each record is divided into *Column Families*

Each row has a *Key*



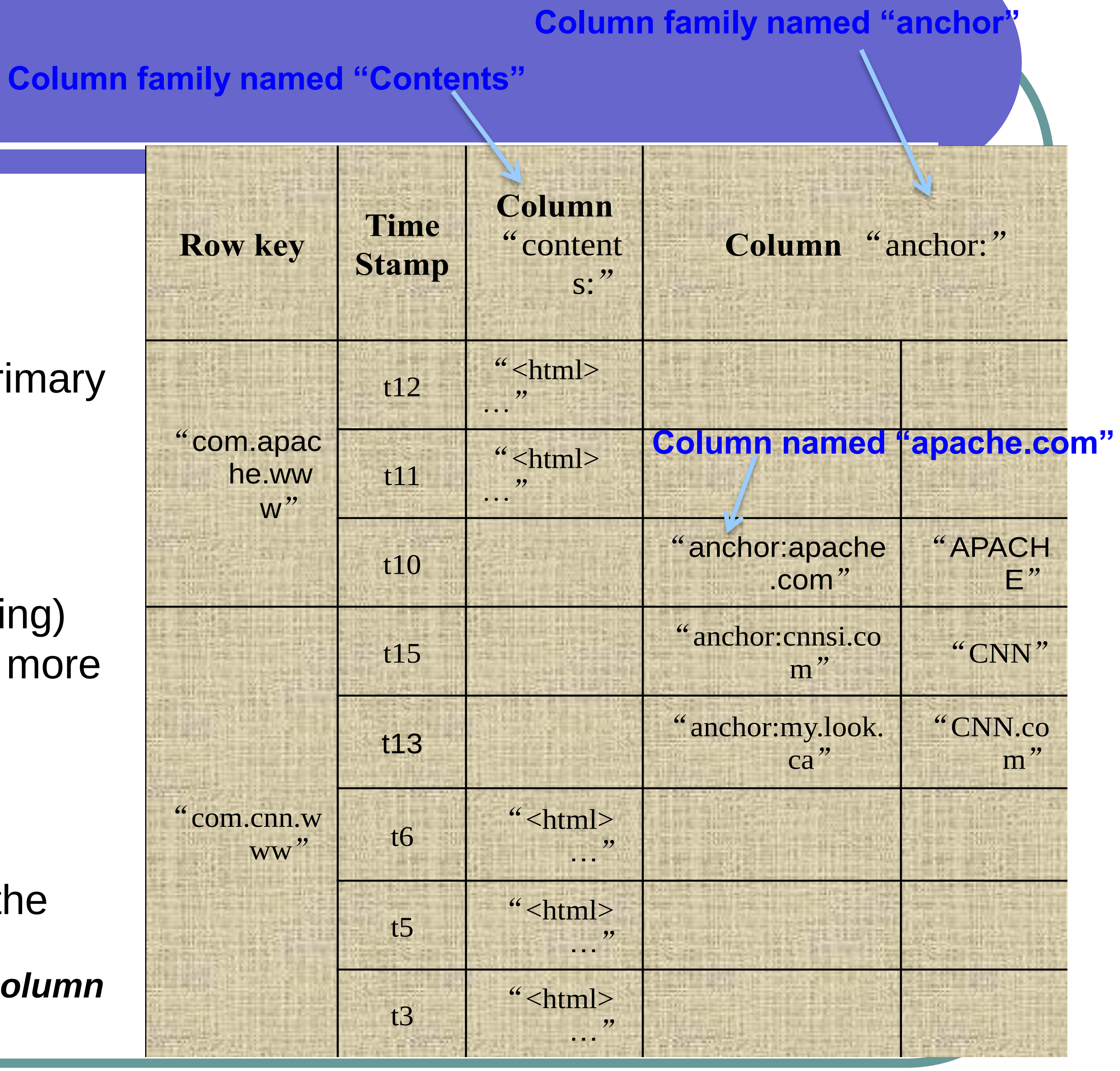
PERSON TABLE

row key	personal_data		demographic		...
PersonID	**Name**	**Address**	**BirthDate**	**Gender**	...
1	H. Houdini	Budapest, Hungary	1926-10-31	M	
2	D. Copper	New Jersey, USA	1956-09-16	M	
3	Merlin	Stonehenge, England	1136-12-03	F	
...	...	...	...	...	
500,000,000	F. Cadillac	Nevada, USA	1964-01-07	M	

Figure 2 - Census Data in Column Families

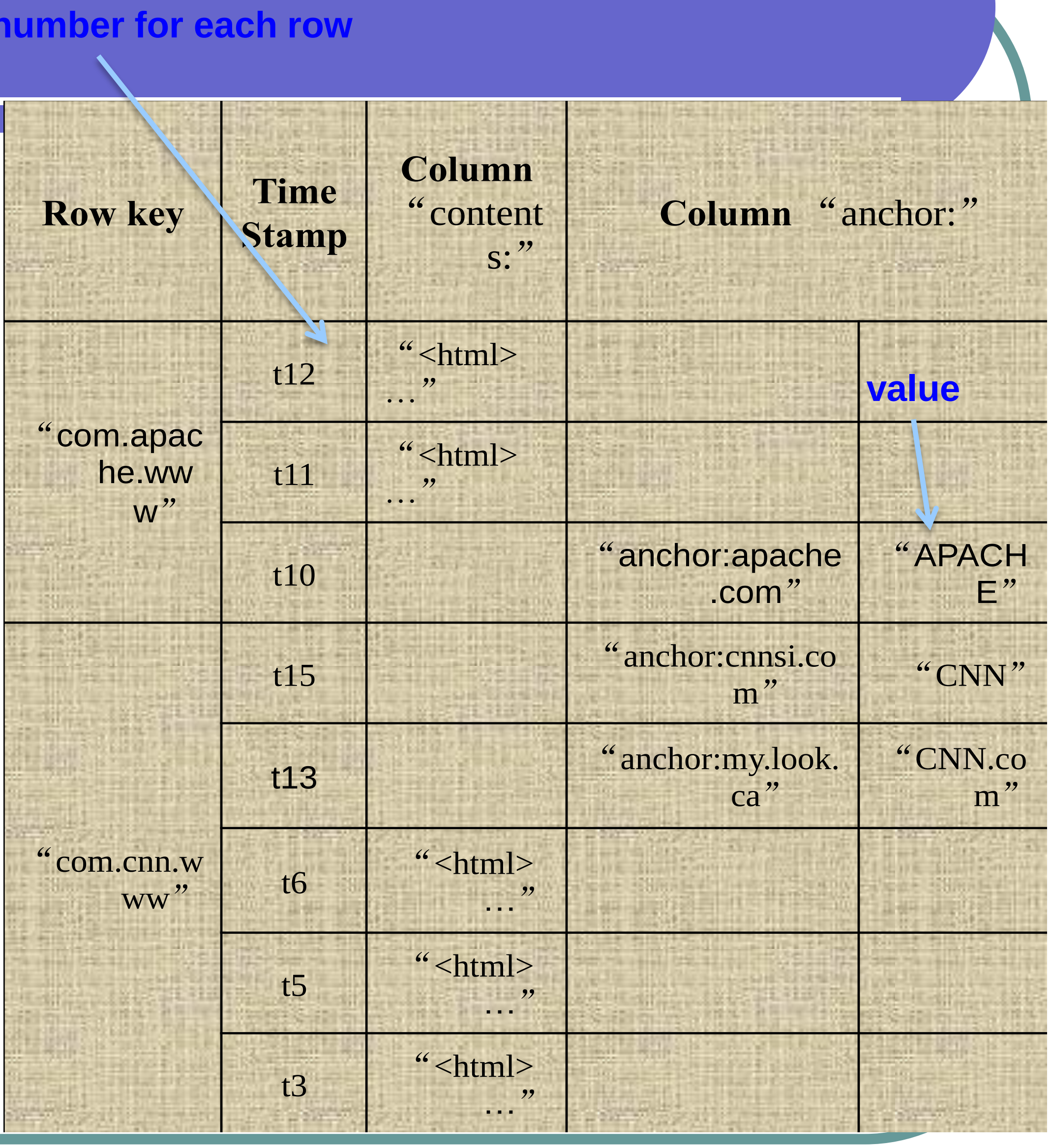
Each column family consists of one or more *Columns*

- **Key**
  - Byte array
  - Serves as the primary key for the table
  - Indexed far 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***


Column family named “anchor”
Column family named “Contents”
Column named “apache.com”

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>…”		

- **Version Number**
  - Unique within each key
  - By default→ System’s timestamp
  - Data type is Long
- **Value (Cell)**
  - Byte array

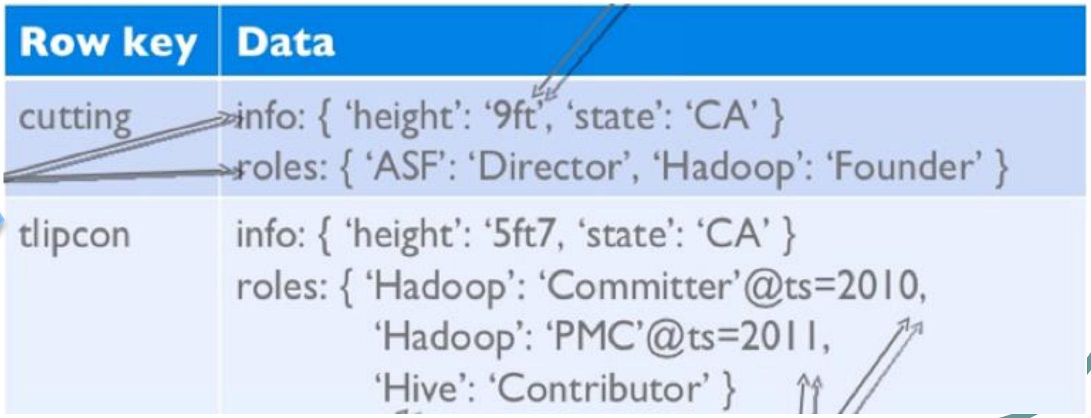

Version number for each row
value

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>…”		

Notes on Data Model

- 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

"Roles" column family has different columns in different cells



Row key	Data

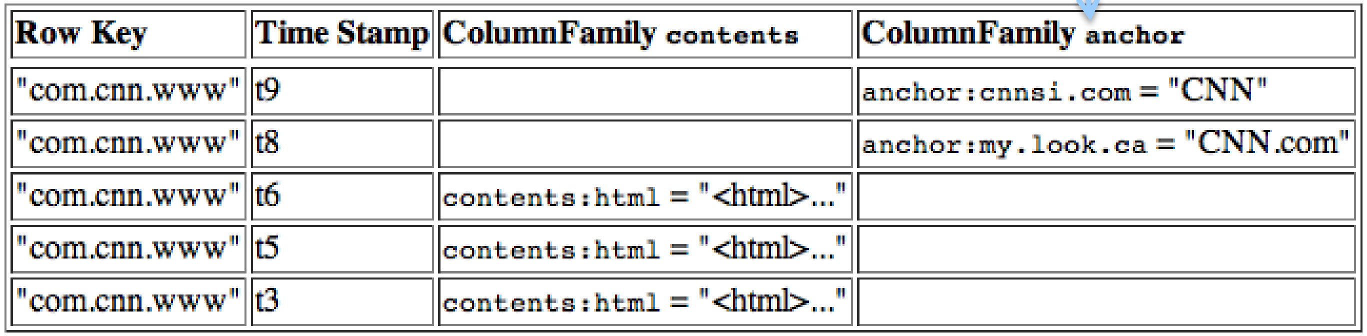
cutting	info: { 'height': '9ft', 'state': 'CA' }
	roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }

tlipcon	info: { 'height': '5ft7, 'state': 'CA' }
	roles: { 'Hadoop': 'Committer'@ts=2010,
	'Hadoop': 'PMC'@ts=2011,
	'Hive': 'Contributor' }

Notes on Data Model (Cont'd)

- The ***version number*** can be user-supplied
  - Even does not have to be inserted in increasing order
  - Version number are unique within each key
- Table can be very sparse
  - Many cells are empty
- ***Keys*** are indexed as the primary key

Has two columns
[cnnsi.com & my.look.ca]



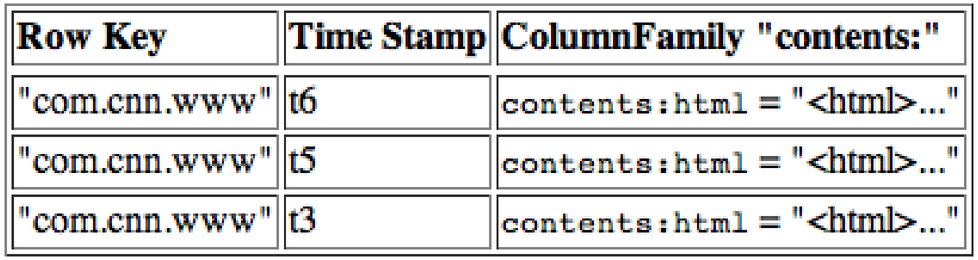
Row Key	Time Stamp	ColumnFamily contents	ColumnFamily anchor
"com.cnn.www"	t9		anchor:cnnsi.com = "CNN"
"com.cnn.www"	t8		anchor:my.look.ca = "CNN.com"
"com.cnn.www"	t6	contents:html = "<html>..."	
"com.cnn.www"	t5	contents:html = "<html>..."	
"com.cnn.www"	t3	contents:html = "<html>..."	

HBase Physical Model

- Each column family is stored in a separate file (called ***HTables***)
- Key & Version numbers are replicated with each column family
- Empty cells are not stored

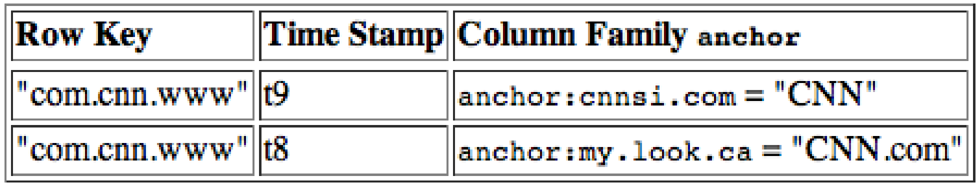
HBase maintains a multi-level index on values:
<key, column family, column name, timestamp>

Table 5.3. ColumnFamily `contents`



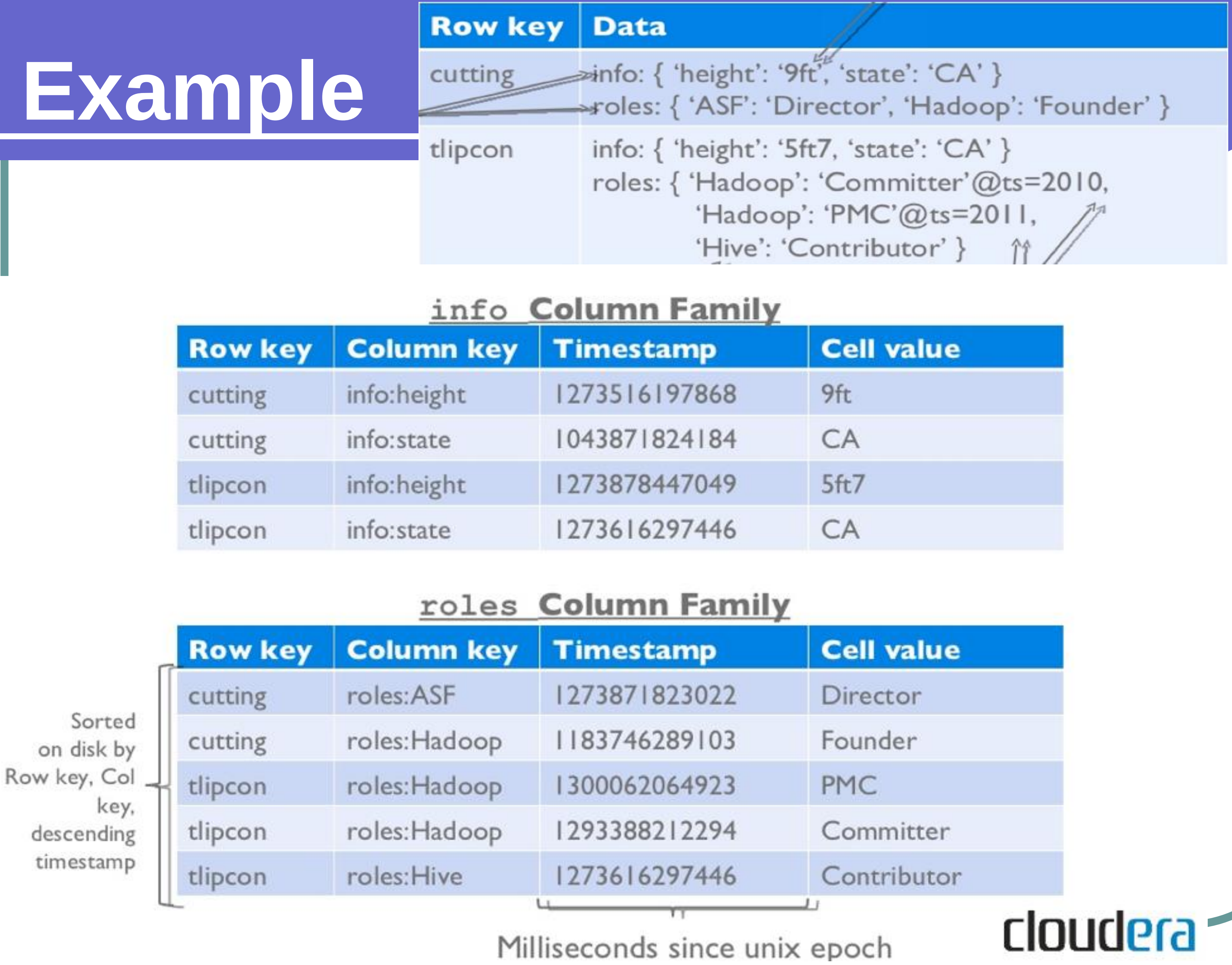
Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	`contents:html` = "<html>..."
"com.cnn.www"	t5	`contents:html` = "<html>..."
"com.cnn.www"	t3	`contents:html` = "<html>..."

Table 5.2. ColumnFamily `anchor`



Row Key	Time Stamp	Column Family `anchor`
"com.cnn.www"	t9	`anchor:cnnsi.com` = "CNN"
"com.cnn.www"	t8	`anchor:my.look.ca` = "CNN.com"

Example



Row key	Data

cutting	info: { 'height': '9ft', 'state': 'CA' }
	roles: { 'ASF': 'Director', 'Hadoop': 'Founder' }

tlipcon	info: { 'height': '5ft7, 'state': 'CA' }
	roles: { 'Hadoop': 'Committer'@ts=2010,
	'Hadoop': 'PMC'@ts=2011,
	'Hive': 'Contributor' }

info Column Family

Row key	Column key	Timestamp	Cell value
cutting	info:height	1273516197868	9ft
cutting	info:state	1043871824184	CA
tlipcon	info:height	1273878447049	5ft7
tlipcon	info:state	1273616297446	CA

roles Column Family

Row key	Column key	Timestamp	Cell value
cutting	roles:ASF	1273871823022	Director
cutting	roles:Hadoop	1183746289103	Founder
tlipcon	roles:Hadoop	1300062064923	PMC
tlipcon	roles:Hadoop	1293388212294	Committer
tlipcon	roles:Hive	1273616297446	Contributor

Sorted on disk by Row key, Col key, descending timestamp

Milliseconds since unix epoch

cloudera

HBase Regions

- Each HTable (column family) is partitioned horizontally into ***regions***
  - Regions are counterpart to HDFS blocks

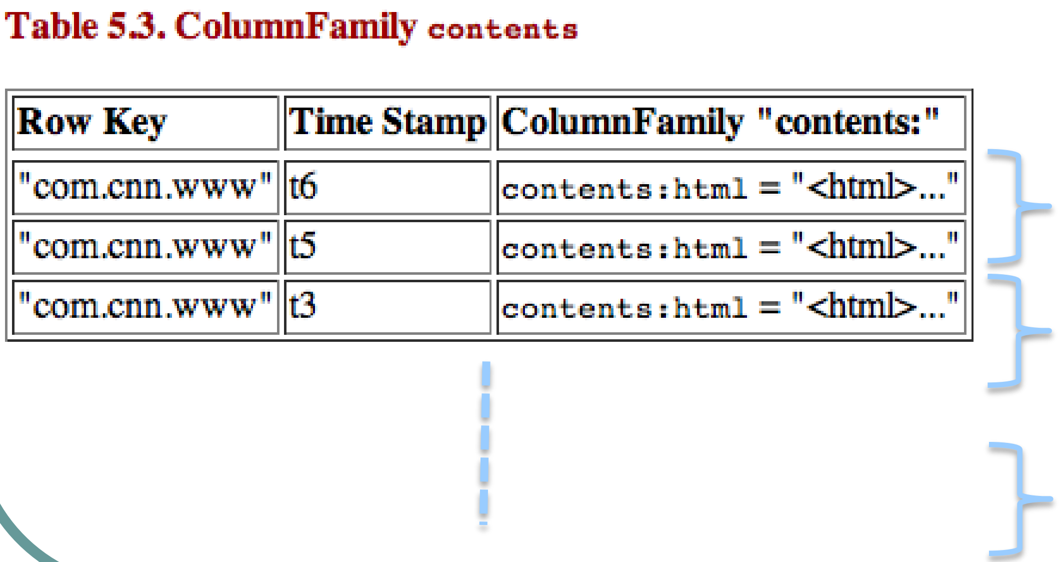


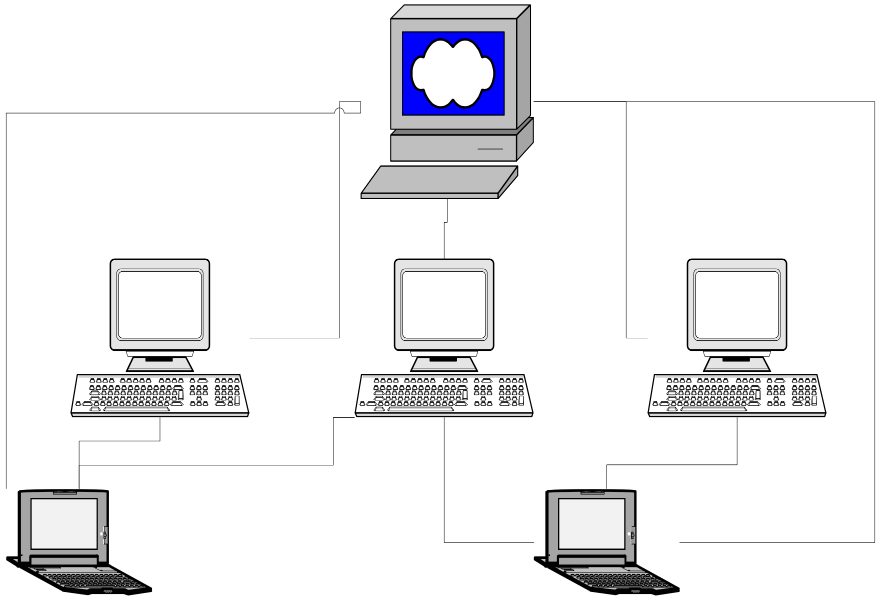
Table 5.3. ColumnFamily `contents`

Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	`contents:html` = "<html>..."
"com.cnn.www"	t5	`contents:html` = "<html>..."
"com.cnn.www"	t3	`contents:html` = "<html>..."

Each will be one region

Hbase Architecture

- The HBaseMaster
  - One master
- The HRegionServer
  - Many region servers
- The HBase client

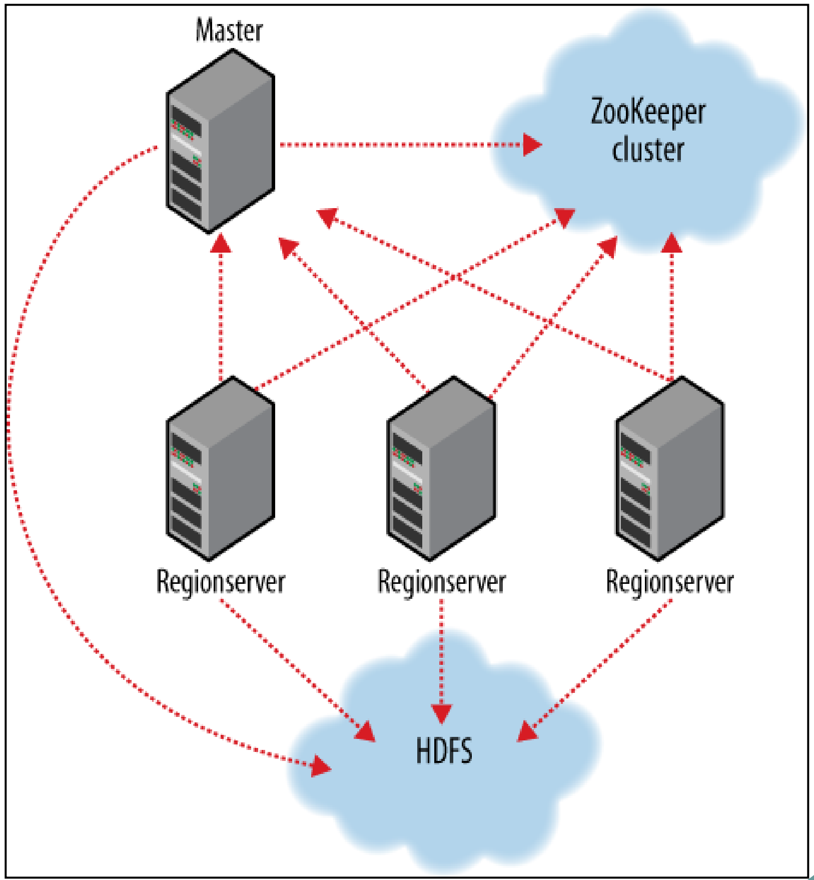


HBase Components

- **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***)
- **Master**
  - Responsible for coordinating the slaves
  - Assigns regions, detects failures
  - Admin functions

ZooKeeper

- HBase depends on ZooKeeper
- By default HBase manages the ZooKeeper instance
  - E.g., starts and stops ZooKeeper
- HMaster and HRegionServers register themselves with ZooKeeper



Thank You !