HBase

Learning Outcomes

At the end of this topic, You should be able to

- Demonstrate the key concepts of Hbase for Data processing
- Apply appropriate Hbase model in data storage
- Evaluate Hbase architecture.

Topic & Structure of The Lesson

- HBASE - Introduction
- Conceptual data model
- Physical data storage
- Data operations
- Architecture
- Summary

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

Introduction to HBase

- NoSQL database built on top of Hadoop.
- Stores large amounts of unstructured data.
- Column-oriented storage model.
- Designed for scalability and flexibility.
- Works well with real-time applications.

HBase: Overview

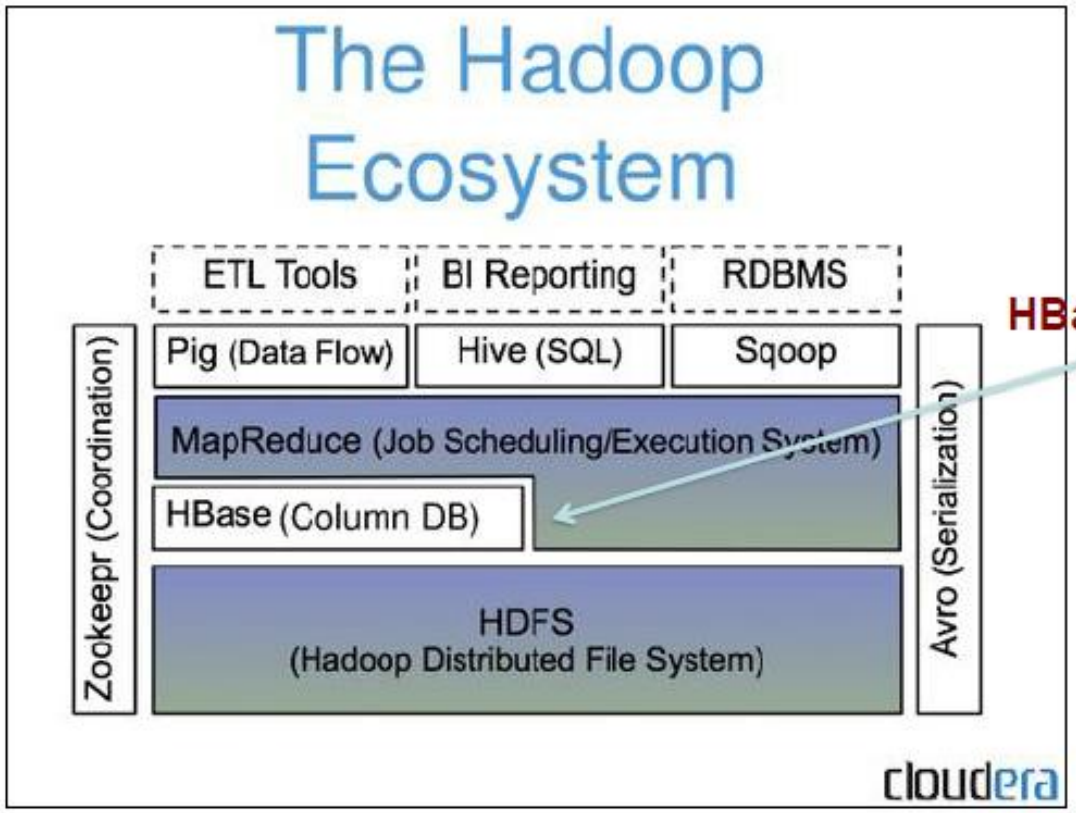
"HBase™ is the Hadoop database, a distributed, scalable, big data store. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System."

(https://hbase.apache.org)

HBase: Overview

- HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data.
- It leverages the fault tolerance provided by the Hadoop File System (HDFS).

HBase: Part of Hadoop's Ecosystem



HBase is built on top of HDFS

HBase files are internally stored in HDFS

Comparison of HBase and RDBMS

Feature	HBase	RDBMS
Data Model	NoSQL, schema-less	Relational, schema-based
Data Structure	Wide tables, column-oriented	Structured tables, row-oriented
Scalability	Horizontally scalable	Usually vertically scalable
Query Language	No standard SQL support	SQL for querying
Consistency Model	Strong consistency	ACID compliance
Data Types Supported	Semi-structured, unstructured	Structured
Use Case	Real-time, big data	Transactional processing

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

Comparison of HDFS and HBase

Feature	HDFS (Hadoop Distributed File System)	HBase
Purpose	Distributed file storage	NoSQL database built on HDFS
Data Access	Sequential access, high latency	Random access, low latency
Data Lookups	Does not support fast individual lookups	Provides fast lookups for large tables
Data Structure	File-based, stores large files	Column-oriented, structured as tables
Schema	No schema, unstructured storage	Schema-less with column families
Use Case	Large-scale data storage, analytics	Real-time applications, big data handling
Consistency Model	Eventually consistent	Strong consistency
Optimization	Optimized for batch processing	Optimized for random read/write access
Data Retrieval	Sequential access only	Indexed HDFS files for faster lookups

HBase is ..

- A distributed data store that can scale horizontally to 1,000s of commodity servers and petabytes of indexed storage.
- Designed to operate on top of the Hadoop distributed file system (HDFS) for scalability, fault tolerance, and high availability.

Column Oriented vs. Row Oriented

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.

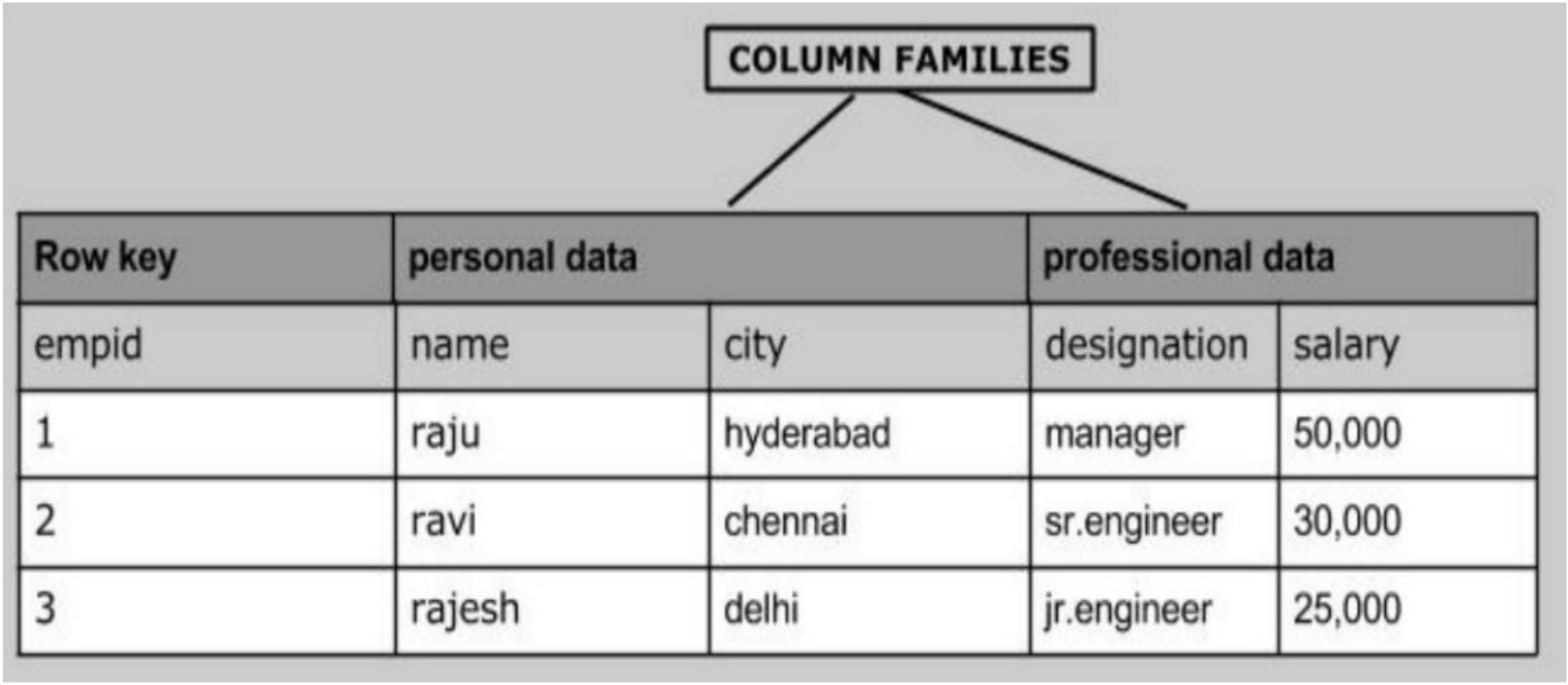
Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.

Storage Mechanism in HBase

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

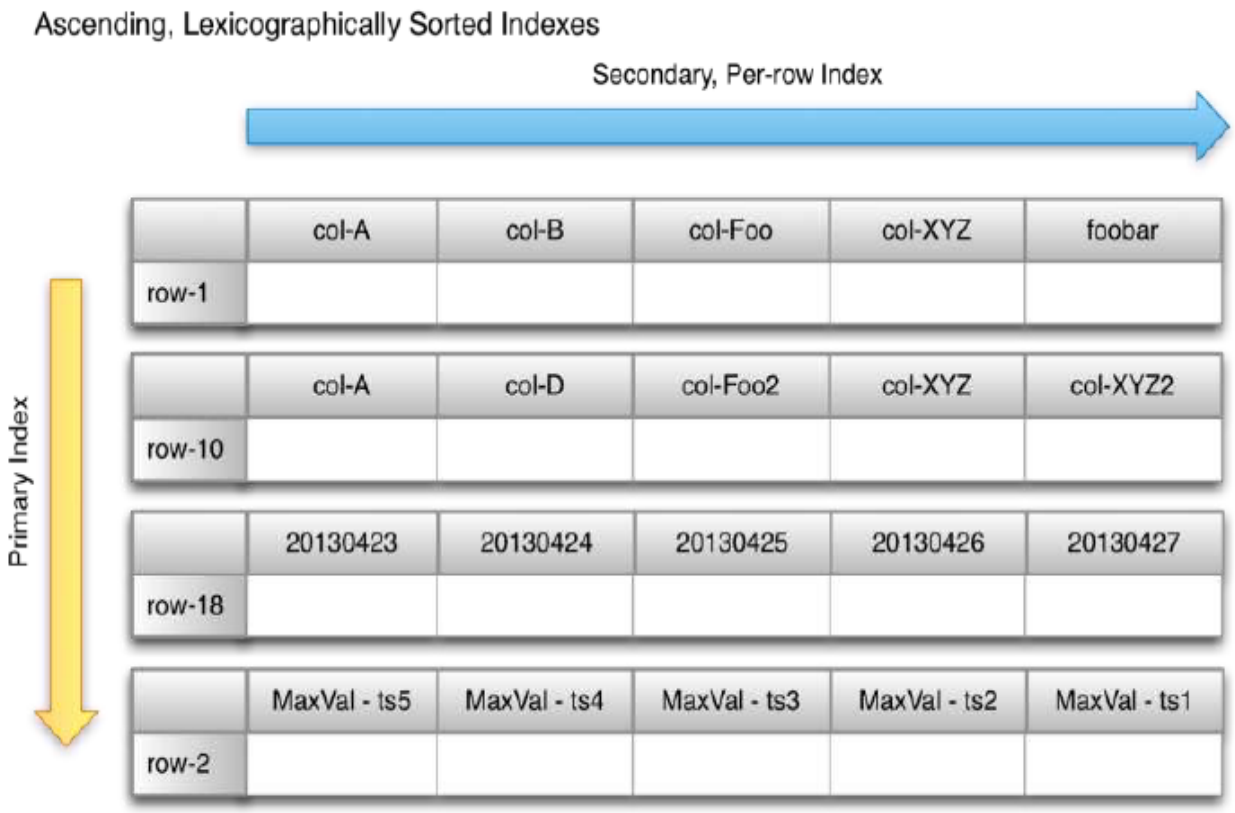
Rowid	Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3
1									
2									
3									



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

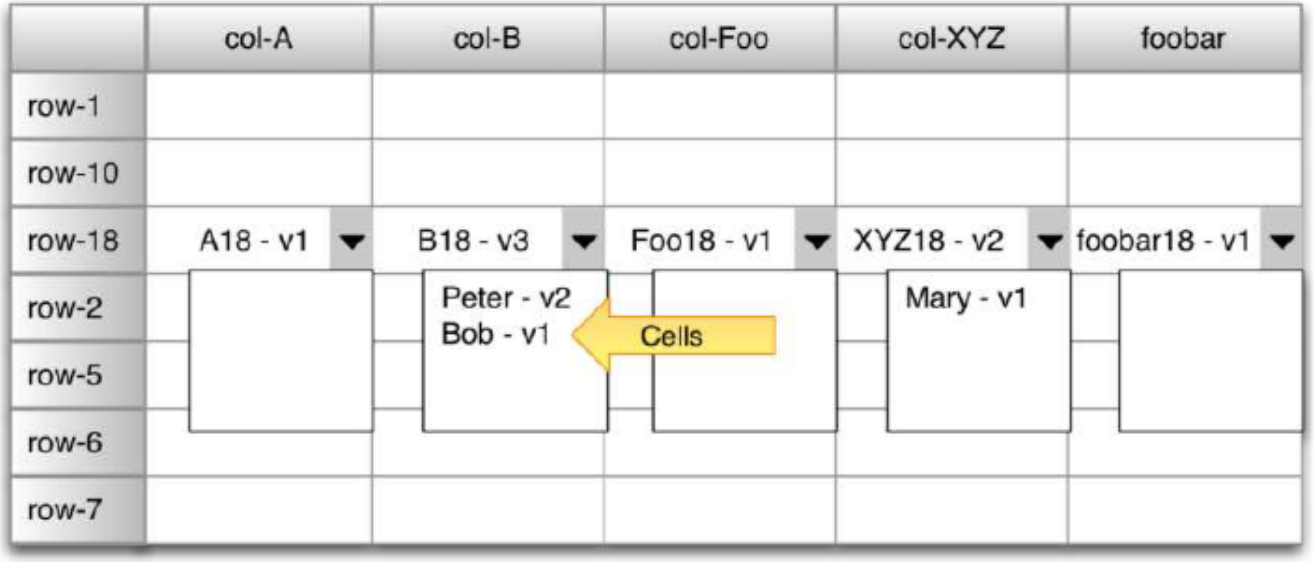
HBase Tables

- HBase can have a **different** schema per row
- Could be called **schema-less**
- **Primary** access by the user given **row key** and **column name**
- **Sorting** of **rows** and **columns** by their **key** (aka names)



HBase Tables

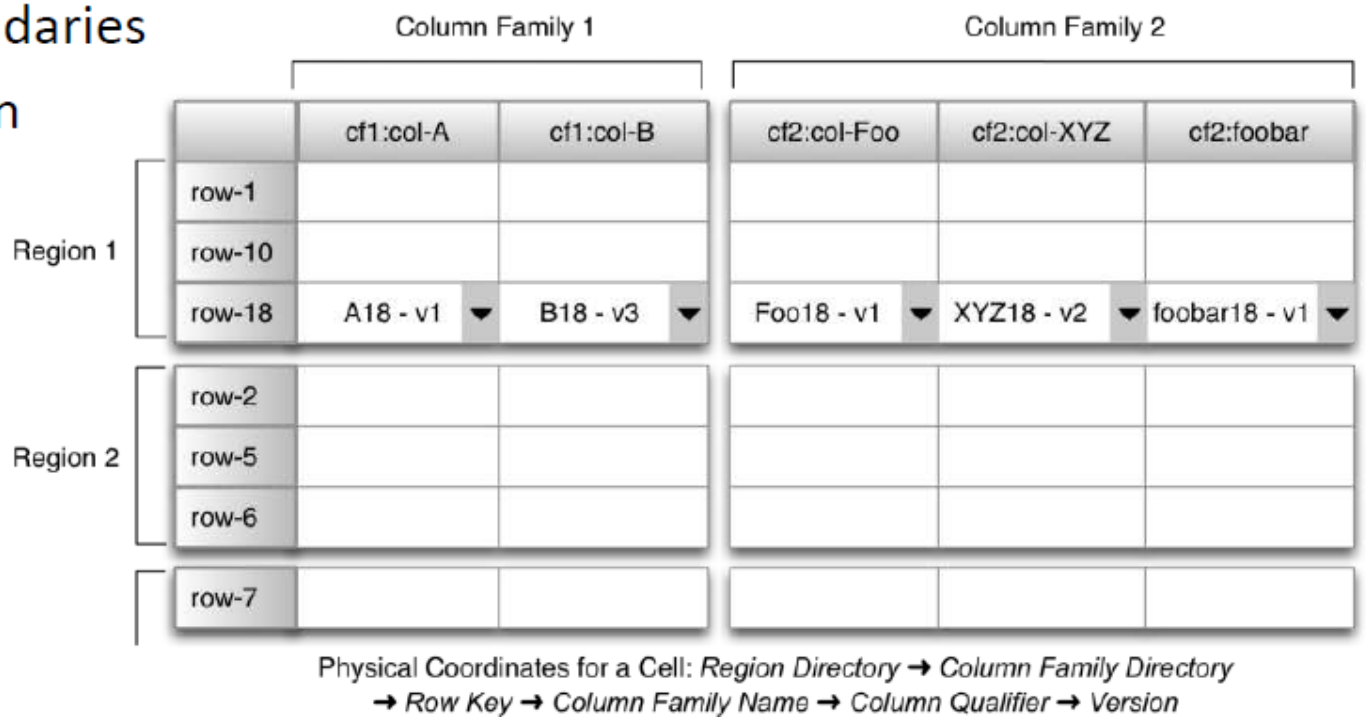
- Each row/column coordinate is tagged with a **version** number, allowing multi-versioned values
- Version is usually the current **time** (as epoch)
- API lets user **ask** for versions (specific, by count, or by ranges)
- Up to **2B** versions



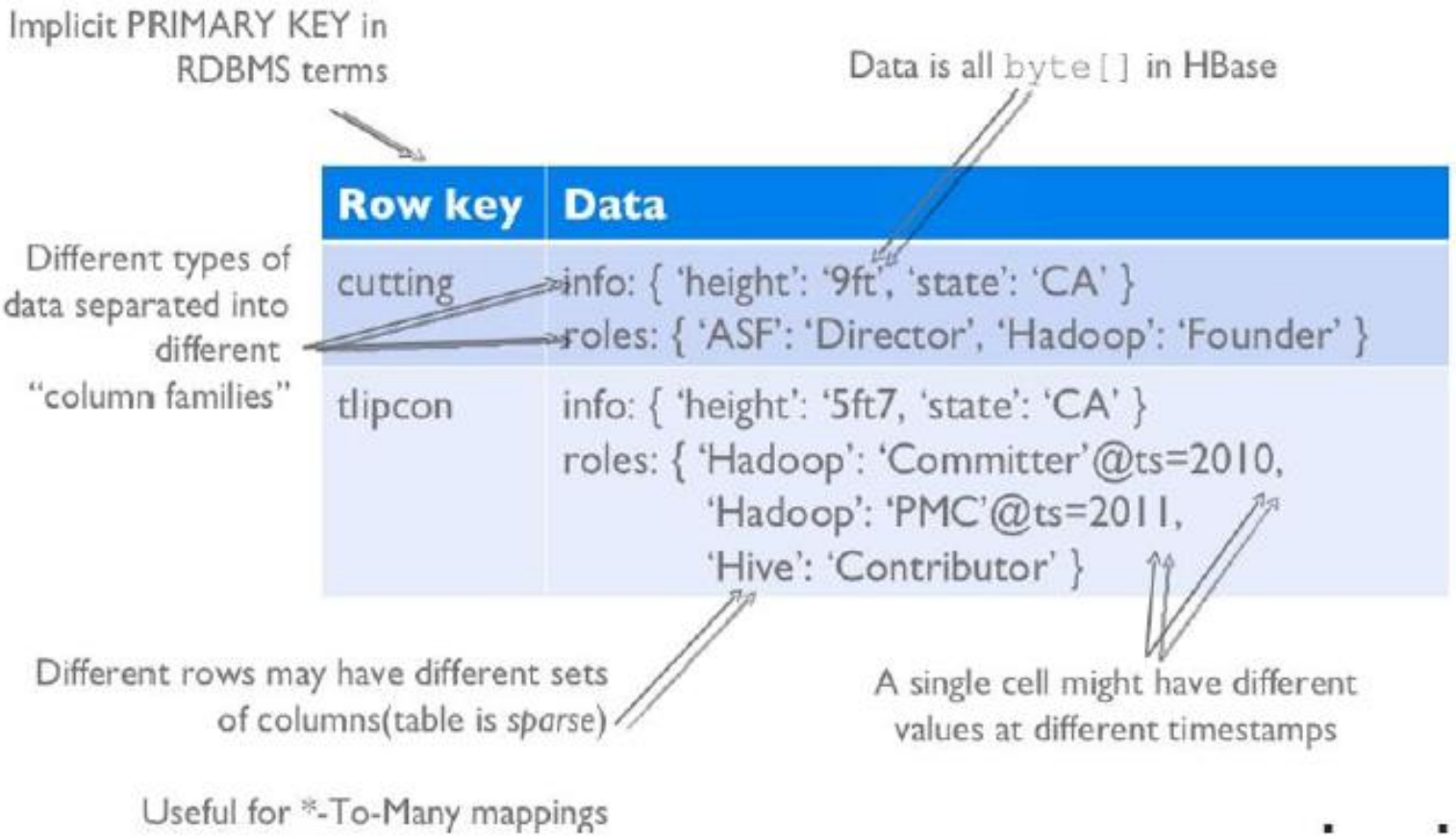
Coordinates for a Cell: *Row Key* → *Column Name* → *Version*

HBase Tables

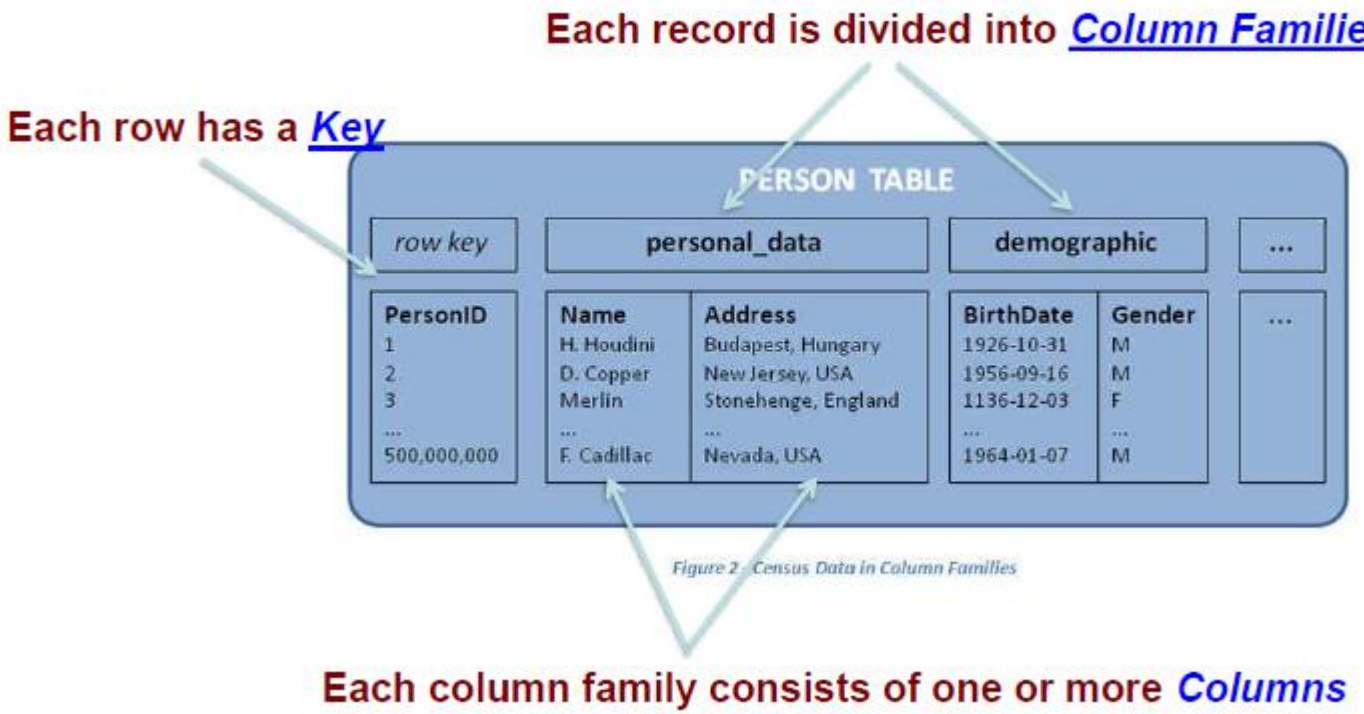
- Table data is **cut** into pieces to **distribute** over cluster
- **Regions** split table into **shards** at size boundaries
- **Families** split within regions to **group** sets of columns together
- At least **one** of each is needed



HBase Logical View



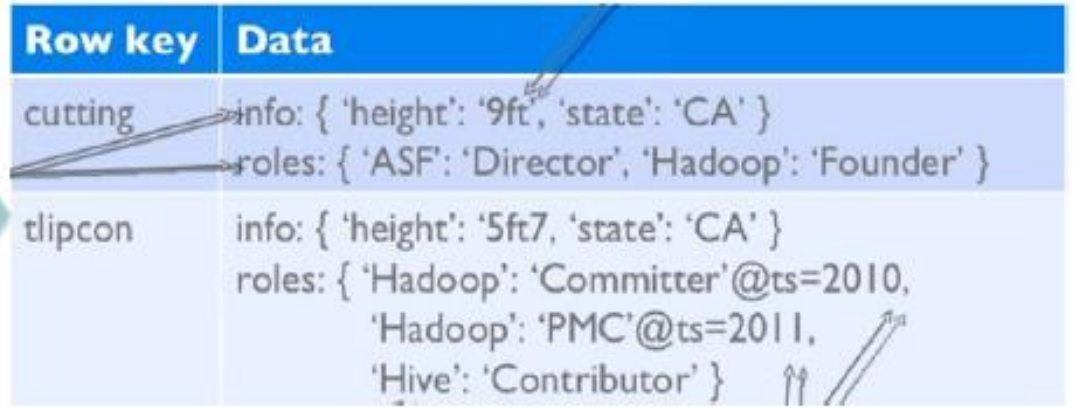
HBase: Keys and Column Families



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' }

Storage Mechanism in HBase

- 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.
- Each cell value of the table has a timestamp. In short, in an HBase:
  - ✓Table is a collection of rows.
  - ✓Row is a collection of column families.
  - ✓Column family is a collection of columns.
  - ✓Column is a collection of key value pairs.

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"

HBase Regions

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

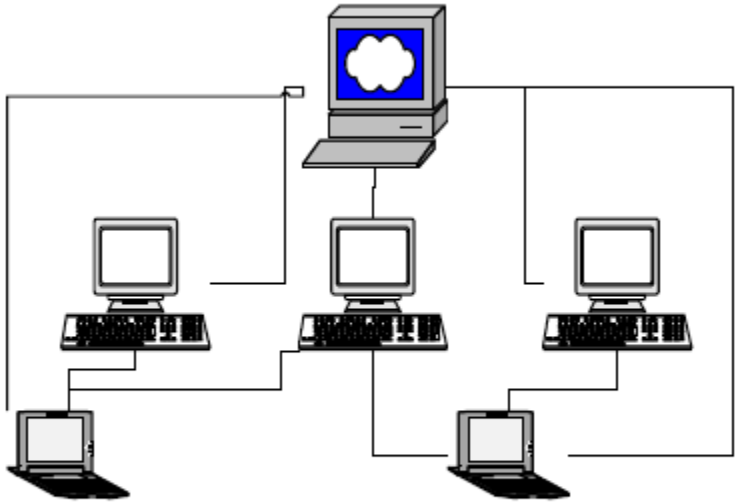
Table 5.3. ColumnFamily `contents`

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

Three Major Components

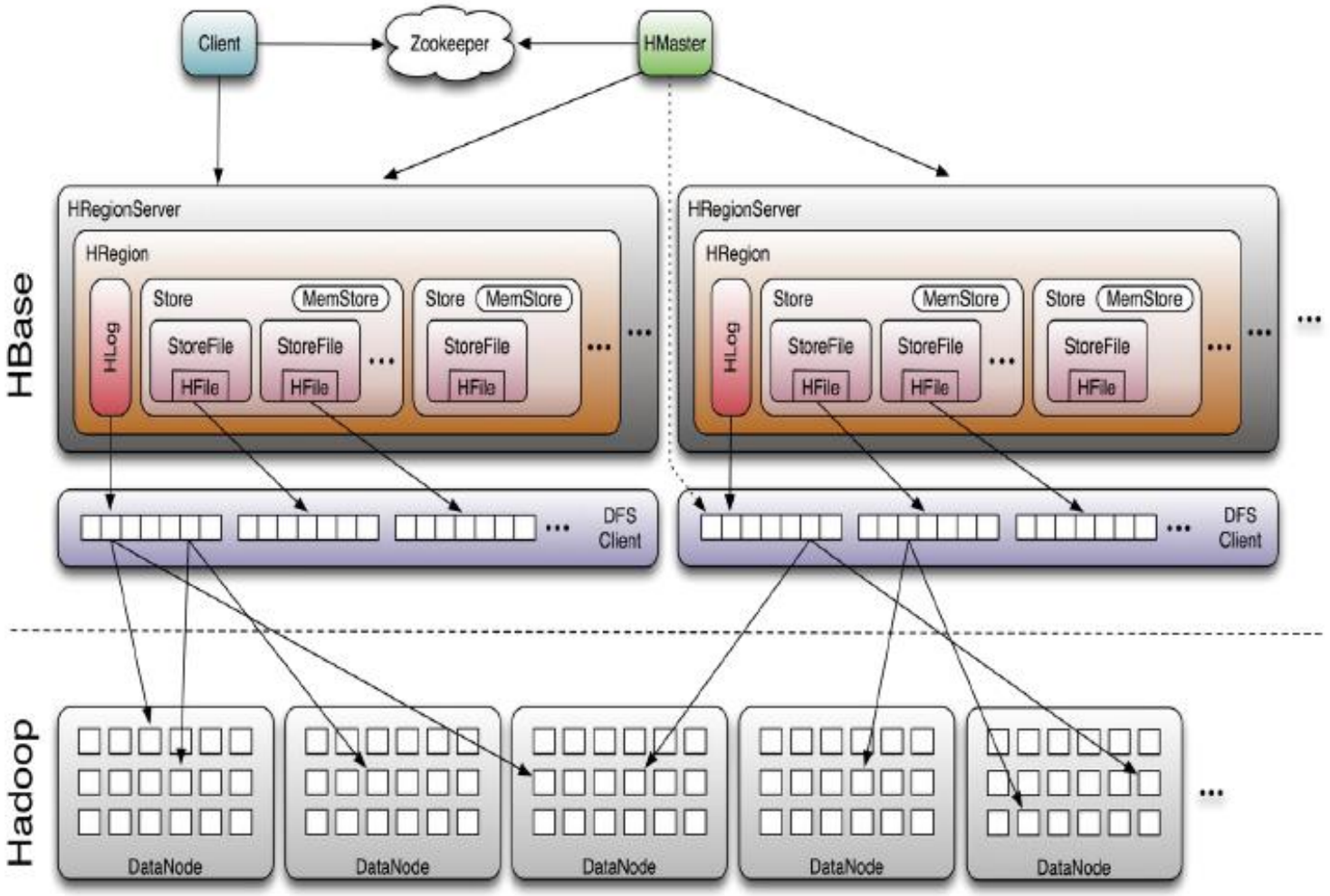
- The HBaseMaster
  - One master
- The HRegionServer
  - Many region servers
- The HBase client



HBase Components

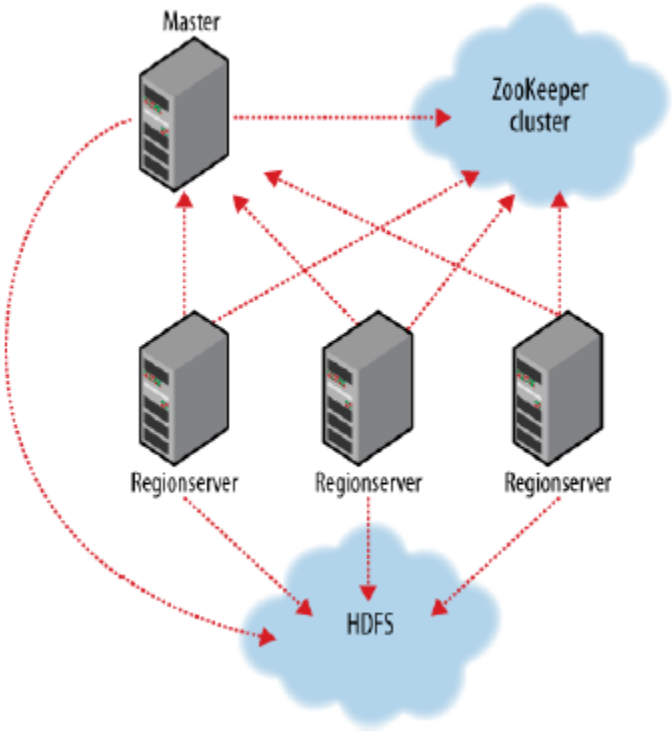
- **Region**
  - A subset of a table's rows, like horizontal range partitioning
  - Automatically done
- **RegionServer (many slaves)**
  - Manages data regions
  - Serves data for reads and writes (***using a log***)
- **Master**
  - Responsible for coordinating the slaves
  - Assigns regions, detects failures
  - Admin functions

Architecture



ZooKeeper

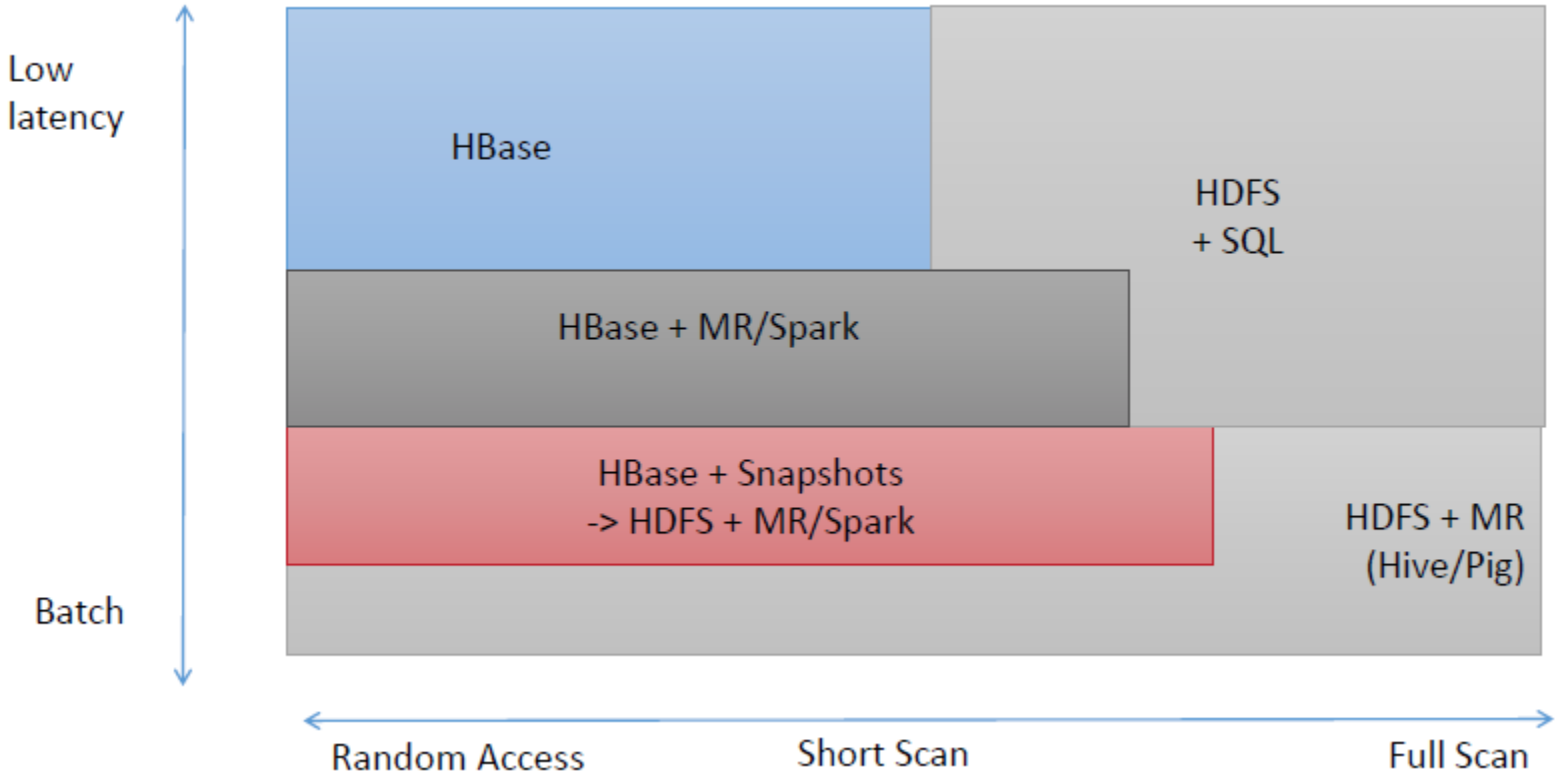
- Zookeeper acts like as a coordinator inside HBase distributed environment. It helps in maintaining server state inside the cluster by communicating through sessions.



Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

Data Workloads

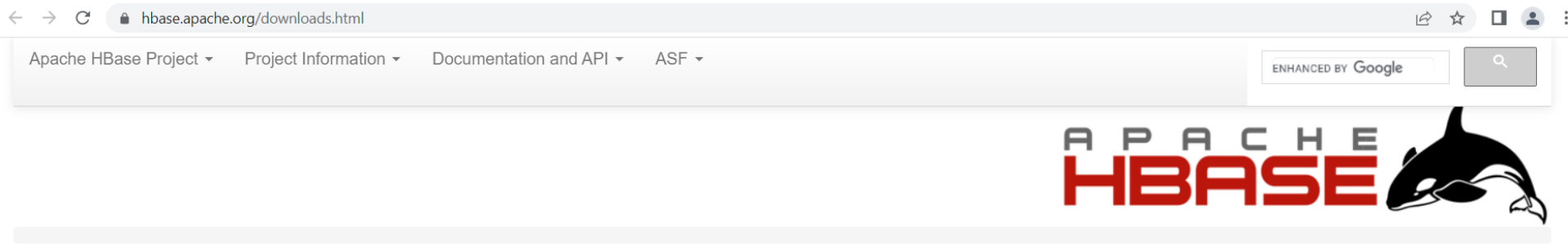


DDL Commands

- create: Creates a table.
- list: Lists all the tables in HBase.
- disable: Disables a table.
- is_disabled: Verifies whether a table is disabled.
- enable: Enables a table.
- is_enabled: Verifies whether a table is enabled.
- describe: Provides the description of a table.
- alter: Alters a table.
- exists: Verifies whether a table exists.
- drop: Drops a table from HBase.

DML Commands

- put: Puts a cell value at a specified column in a specified row in a particular table.
- get: Fetches the contents of row or a cell.
- delete: Deletes a cell value in a table.
- deleteall: Deletes all the cells in a given row.
- scan: Scans and returns the table data.
- count: Counts and returns the number of rows in a table.
- truncate: Disables, drops, and recreates a specified table.



Downloads

The below table lists mirrored release artifacts and their associated hashes and signatures available ONLY at apache.org. The keys used to sign releases can be found in our published KEYS file. See Verify The Integrity Of The Files for how to verify your mirrored downloads.

Releases

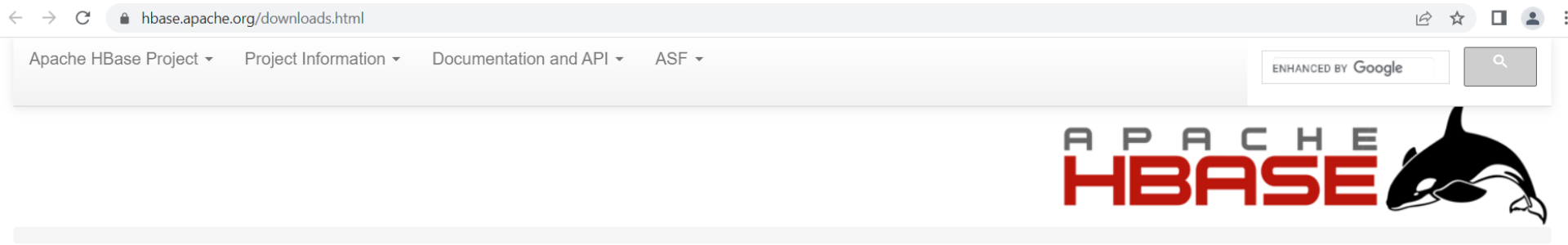
Version	Release Date	Compatibility Report	Changes	Release Notes	Download	Notices
3.0.0-alpha-3	2022/06/27	3.0.0-alpha-3 vs 2.0.0	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	Testing only, not production ready
2.5.1	2022/10/28	2.5.1 vs 2.5.0	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	
2.4.15	2022/10/28	2.4.15 vs 2.4.14	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	*stable release*

Downloads

The below table lists mirrored release artifacts and their associated hashes and signatures available ONLY at apache.org. The keys used to sign releases can be found in our published KEYS file. See Verify The Integrity Of The Files for how to verify your mirrored downloads.

Releases

Version	Release Date	Compatibility Report	Changes	Release Notes	Download	Notices
3.0.0-beta-1	2024/01/14	3.0.0-beta-1 vs 2.0.0	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	Feature freeze, passed a 10B ITBLL run, use with caution
2.6.1	2024/10/18	2.6.0 vs 2.6.1	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc) hadoop3-bin (sha512 asc) hadoop3-client-bin (sha512 asc)	
2.5.10	2024/07/24	2.5.10 vs 2.5.9	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc) hadoop3-bin (sha512 asc) hadoop3-client-bin (sha512 asc)	*stable release*



Downloads

The below table lists mirrored release artifacts and their associated hashes and signatures available ONLY at apache.org. The keys used to sign releases can be found in our published KEYS file. See Verify The Integrity Of The Files for how to verify your mirrored downloads.

Releases

Version	Release Date	Compatibility Report	Changes	Release Notes	Download	Notices
3.0.0-alpha-3	2022/06/27	3.0.0-alpha-3 vs 2.0.0	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	Testing only, not production ready
2.5.1	2022/10/28	2.5.1 vs 2.5.0	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	
2.4.15	2022/10/28	2.4.15 vs 2.4.14	Changes	Release Notes	src (sha512 asc) bin (sha512 asc) client-bin (sha512 asc)	*stable release*

HBase Use Cases

- **Data Volume**
  HBase is ideal for handling large-scale data where the volume is substantial, requiring distributed and scalable storage.

- **Application Types**
  HBase suits applications that need fast access to large datasets, especially in big data analytics, real-time applications, and IoT.

- **Hardware Environment**
  HBase is designed for environments with distributed hardware setups, typically involving clusters of commodity servers.

- **No Requirement of Relational Features**
  Applications that don't need relational database features like complex joins or foreign keys can benefit from HBase's simplified structure.

- **Quick Access to Data**
  HBase excels in scenarios where fast and random access to large amounts of data is essential, such as real-time processing systems.

Top 10 Companies Using HBase for Big Data Solutions

- **Facebook** – Uses HBase for its messaging infrastructure, including real-time analytics and data storage.
- **Twitter** – Utilizes HBase for storing and serving user timelines and large-scale social media data.
- **Yahoo!** – One of the largest users of HBase for real-time analytics and search indexing.
- **Salesforce** – Uses HBase for handling customer data and processing vast amounts of real-time information.
- **Adobe** – Leverages HBase in its marketing cloud platform for real-time data storage and analytics.
- **Pinterest** – Employs HBase to manage and process its massive data sets related to user-generated content.
- **Netflix** – Uses HBase for real-time streaming analytics and recommendation engine data.
- **eBay** – Utilizes HBase for search, analytics, and personalization applications, along with its large-scale data storage.
- **Spotify** – Uses HBase for storing and managing its large metadata and user-generated content.
- **Cloudera** – Provides enterprise-level data solutions and uses HBase as part of its distributed data platform for real-time operations.

Quick Review Question

- What are the key components of HBase?
- What is the role of Zookeeper in Hbase?

Summary

- Overview and the history of Hbase
- Introduce Hbase Architecture

Question and Answer Session

Q & A