



Module 4_1

HBase

Thanachart Numnonda, Executive Director, IMC Institute

Thanisa Numnonda, Faculty of Information Technology,
King Mongkut's Institute of Technology Ladkrabang



Introduction

An open source, non-relational, distributed database

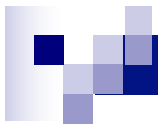


HBase is an open source, non-relational, distributed database modeled after Google's BigTable and is written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS, providing BigTable-like capabilities for Hadoop. That is, it provides a fault-tolerant way of storing large quantities of sparse data.



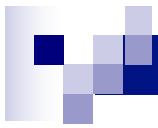
HBase Features

- Hadoop database modelled after Google's BigTable
- Column oriented data store, known as Hadoop Database
- Support random real-time CRUD operations (unlike HDFS)
- NoSQL Database
- Open source, written in Java
- Run on a cluster of commodity hardware



When to use HBase?

- When you need high volume data to be stored
- Un-structured data
- Sparse data
- Column-oriented data
- Versioned data (same data template, captured at various time, time-elapse data)
- When you need high scalability



Which one to use?

- **HDFS**
 - Only append dataset (no random write)
 - Read the whole dataset (no random read)
- **HBase**
 - Need random write and/or read
 - Has thousands of operation per second on TB+ of data
- **RDBMS**
 - Data fits on one big node
 - Need full transaction support
 - Need real-time query capabilities

HBase vs. RDBMS

	HBase	RDBMS
Hardware architecture	Similar to Hadoop. Clustered commodity hardware. Very affordable.	Typically large scalable multiprocessor systems. Very expensive.
Fault Tolerance	Built into the architecture. Lots of nodes means each is relatively insignificant. No need to worry about individual node downtime.	Requires configuration of the HW and the RDBMS with the appropriate high availability options.
Typical Database Size	Terabytes to Petabytes - hundred of millions to billions of rows.	Gigabytes to Terabytes – hundred of thousands to millions of rows.
Data Layout	A sparse, distributed, persistent, multidimensional sorted map.	Rows or column oriented.
Data Types	Bytes only.	Rich data type support.
Transactions	ACID support on a single row only	Full ACID compliance across rows and tables
Query Language	API primitive commands only, unless combined with Hive or other technology	SQL
Indexes	Row-Key only unless combined with other technologies such as Hive or IBM's BigSQL	Yes
Throughput	Millions of queries per second	Thousands of queries per second



- Given this RDBMS:

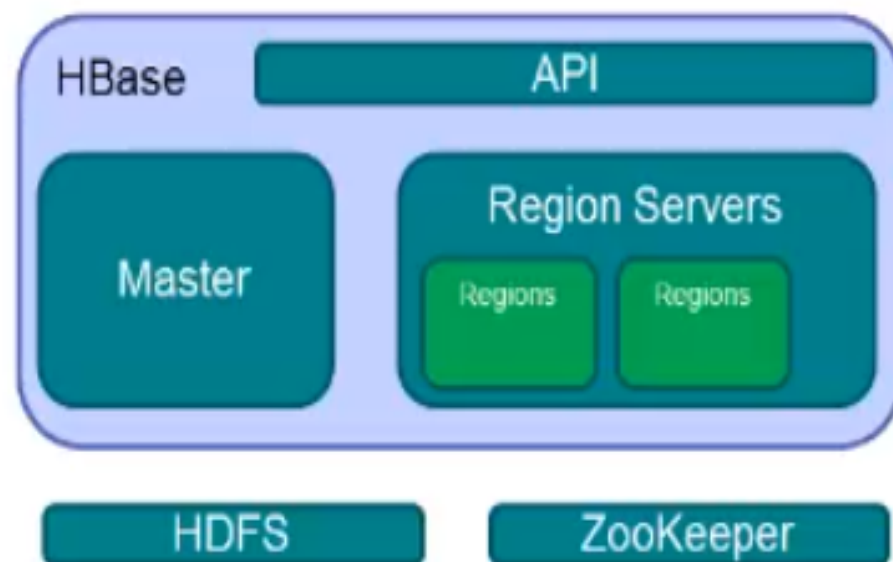
ID (Primary key)	Last name	First name	Password	Timestamp
1234	Smith	John	Hello, world!	20130710
5678	Cooper	Joyce	wysiwyg	20120825
5678	Cooper	Joyce	wisiwig	20130916

- Logical view in HBase:

Row-Key	Value (CF, Qualifier, Version)
1234	info {'lastName': 'Smith', 'firstName': 'John'} pwd {'password': 'Hello, world!'}
5678	info {'lastName': 'Cooper', 'firstName': 'Joyce'} pwd {'password': 'wysiwyg'@ts 20130916, 'password': 'wisiwig'@ts 20120825}

HBase Components

- **Region**
 - Row of table are stores
- **Region Server**
 - Hosts the tables
- **Master**
 - Coordinating the Region Servers
- **API**
 - The Java Client API
- **ZooKeeper**
- **HDFS**





HBase Shell Commands

- See the list of the tables

```
list
```

- Create a table:

```
create 'testTable', 'cf'
```

- Insert data into a table:

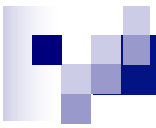
Insert at rowA, column "cf:columnName" with a value of "val1"

```
put 'testTable', 'rowA', 'cf:columnName', 'val1'
```



HBase Shell Commands (Cont.)

- Retrieve data from a table:
Retrive "rowA" from the table "testTable"
`get 'testTable', 'rowA'`
- Iterate through a table:
– `scan 'testTable'`
- Delete a table:
`disable 'testTable'`
`drop 'testTable'`



Hands-On: Running HBase



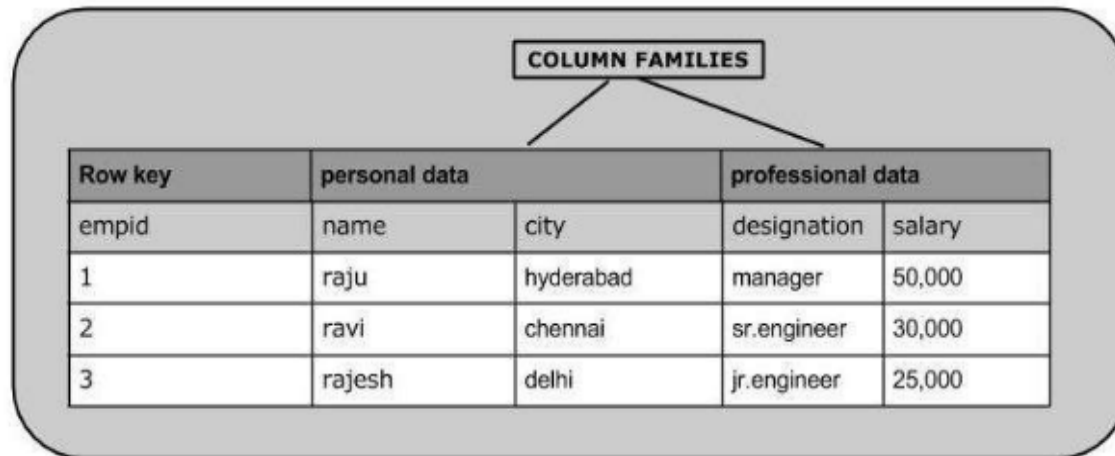
HBase shell

Row key	personal data	professional data

```
$hbase shell  
hbase(main):001:0> create 'employee', 'personal data',  
'professional data'  
hbase(main):002:0> list
```

```
TABLE  
employee  
1 row(s) in 0.0310 seconds
```

Create Data



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

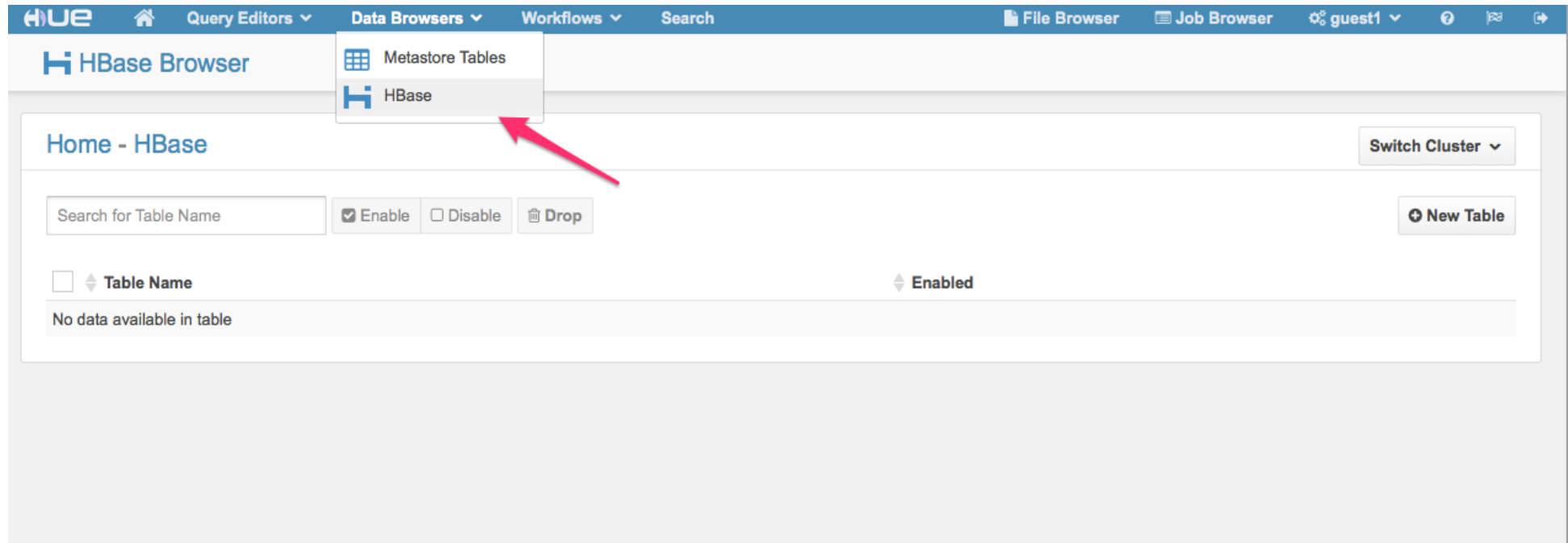
```
hbase(main):010:0> put 'employee','1','personal data:name','raju'  
0 row(s) in 0.1720 seconds
```

```
hbase(main):011:0> put 'employee','1','personal data:city','hyderabad'  
0 row(s) in 0.0140 seconds
```

```
hbase(main):018:0> put 'employee','1','professional data:designation','manager'  
0 row(s) in 0.0110 seconds
```

```
hbase(main):019:0> put 'employee','1','professional data:salary','50000'  
0 row(s) in 0.0070 seconds
```

Running HBase Browser



The screenshot displays the Hue web interface for the HBase Browser. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, File Browser, Job Browser, and a user profile for guest1. The 'Data Browsers' menu is expanded, showing 'Metastore Tables' and 'HBase'. A red arrow points to the 'HBase' option. The main content area is titled 'Home - HBase' and features a search bar labeled 'Search for Table Name', buttons for 'Enable', 'Disable', and 'Drop', and a 'New Table' button. Below these controls, a table is shown with one row containing 'Table Name' and 'Enabled'. The text 'No data available in table' is displayed at the bottom of the table area.

Viewing Employee Table

HUE **Query Editors** **Data Browsers** **Workflows** **Search** **Security**

HBase Browser

Home - Cluster

Switch Cluster

Search for Table Name

☒ Enable

☐ Disable

Drop

New Table

☐ Table Name

Enabled

☐ employee

☒

HBase Browser

Home - Cluster / employee

Switch Cluster

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefi



personal data:

professional data:



Filter Columns/Families

☒ All

Sort By ASC

1

personal data: city	personal data: name	professional data: designation	professional data: salary
hyderabad	raju	manager	50000

Create a table in HBase

HBase Browser

Home - HBase Switch Cluster ▾

☒ Enable ☐ Disable

☐ Table Name Enabled

Create New Table

Table Name:

Student

Column Families:

✕ cf:name

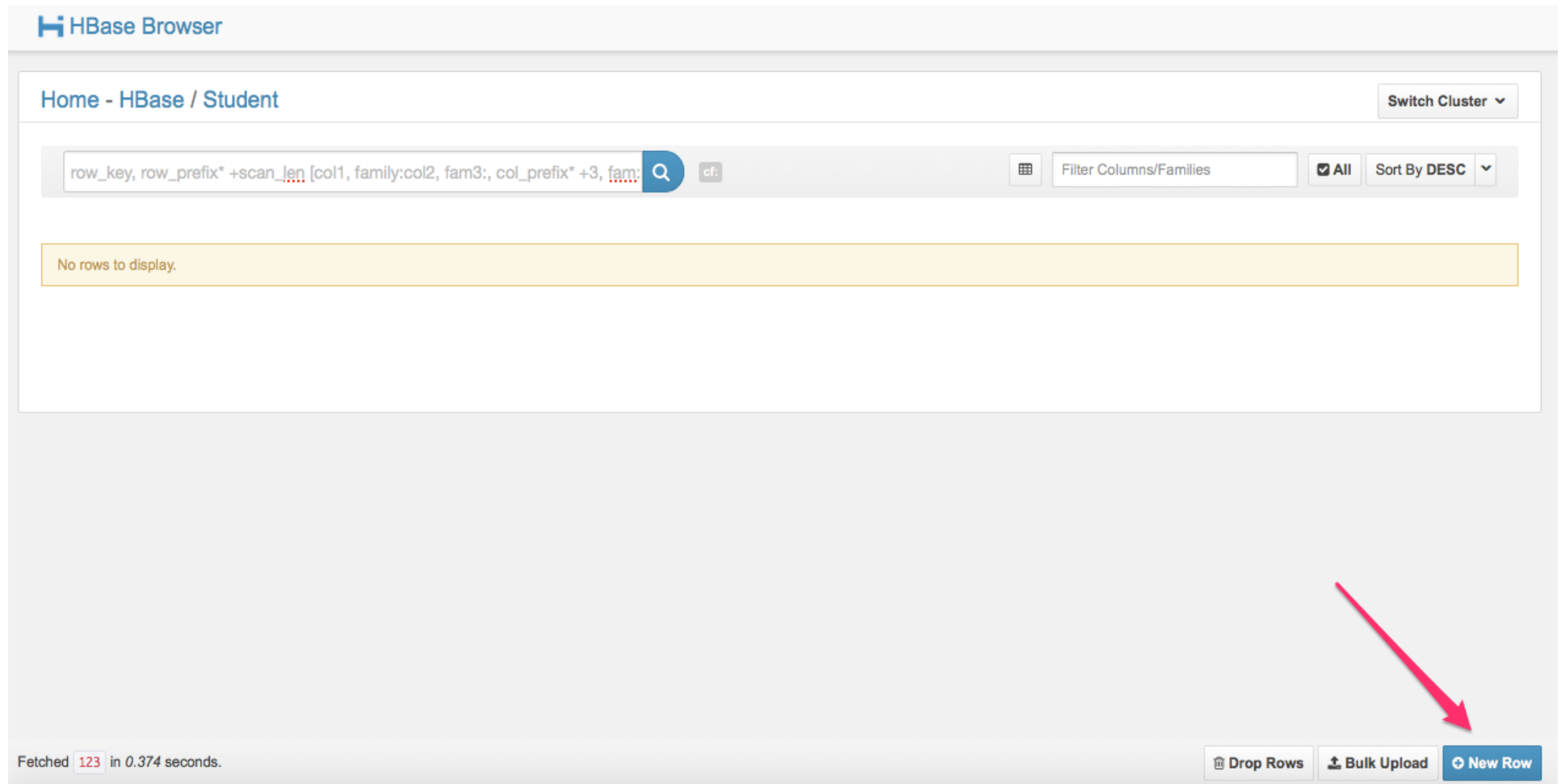
+ Add a column property

+ Add an additional column family

Cancel

Submit

Insert a new row in a table



The image shows the HBase Browser interface. At the top, there's a header with the HBase Browser logo and the text "Home - HBase / Student". Below this, there's a search bar containing the text "row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam:". To the right of the search bar are buttons for "Filter Columns/Families", "All", and "Sort By DESC". Below the search bar, there's a yellow box with the text "No rows to display." At the bottom of the interface, there's a status bar showing "Fetched 123 in 0.374 seconds." and three buttons: "Drop Rows", "Bulk Upload", and "New Row". A red arrow points to the "New Row" button.

HBase Browser

Home - HBase / Student

Switch Cluster ▾

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: Sort By DESC ▾

No rows to display.

Fetched 123 in 0.374 seconds.

Add field into a new row

Insert New Row

Row Key

123

cf:firstname

Thanachart

cf:lastname

Numnonda

+ Add Field

Cancel

Submit

Insert New Row

Row Key

124

cf:firstname

Somchai

+ Add Field

Cancel

Submit

Home - HBase / Student

Switch Cluster ▾

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: cf:



Filter Columns/Families

☑ All

Sort By DESC ▾

124

cf: firstname

Somchai

123

cf: firstname

cf: lastname

Thanachart

Numnonda