ce

**Hbase Lab Manual**

**1. General commands**

In Hbase, general commands are categorized into following commands

1. Status **hbase(main):001:0**> status /version/WHOAMI/table\_help
2. Version
3. table\_help ( scan, drop, get, put, disable, etc.)
4. Whoami

**2. Tables Managements commands**

These commands will allow programmers to create tables and table schemas with rows and column families.The following are Table Management commands

1. Create
2. List
3. Describe
4. Disable
5. Disable\_all
6. Enable
7. Enable\_all
8. Drop
9. Drop\_all
10. Show\_filters
11. Alter
12. Alter\_statuscreae

1.**Create- To Create a table under Hbase**

**Syntax:**  create <tablename>, <columnfamilyname>

**Ex- hbase(main):007:0>** create 'sunil','personaldetail','educationaldetails'

**lIVE:** create 'cdac1','course','type'

### **2.List - Will display all the tables that are present or created in HBase**

**Syntax:** list

3.**Describe** **will give more information about column families present in the mentioned table**

**Syntax:** describe <table name>

**hbase(main):010:0>** describe 'cdac' / TABLE NAME IS "**cdac"**

4.**disable This command will start disabling the named table**

If table needs to be deleted or dropped, it has to disable first

**Syntax: disable <tablename>**

**hbase(main):011:0> disable 'education'**

**5.disable\_all This command will disable all the tables matching the given regex.**

The implementation is same as delete command (Except adding regex for matching)

Once the table gets disable the user can able to delete the table from HBase

Before delete or dropping table, it should be disabled first

**Syntax: disable\_all<"matching regex"**

### **6.Enable This command will start enabling the named table**

Whichever table is disabled, to retrieve back to its previous state we use this command

**Syntax: enable <tablename>**

**7.show\_filters** This command displays all the filters present in HBase like ColumnPrefix Filter, TimestampsFilter, PageFilter, FamilyFilter, etc.

**Syntax: show\_filters**

**8.drop** To delete/DROP the table present in HBase, first we have to disable it.

So either table to drop or delete first the table should be disable using disable command

**Syntax:** drop <table name>

**9.drop\_all** This command will drop all the tables matching the given regex

**Syntax:** drop\_all<"regex">

**10.is\_enabled** This command will verify whether the named table is enabled or not.

* Suppose a table is disabled, to use that table we have to enable it by using enable command
* is\_enabled command will check either the table is enabled or not

**Syntax:** is\_enabled 'education'

### 11..alter adding a column family in table

**Syntax: alter <tablename>, NAME=><column familyname>, VERSIONS=>5**

**Altering single, multiple column family names**

**Syntax: alter ‘<table\_name>’, ‘reference of column\_family’ , {** NAME => 'New \_column\_family name', IN\_MEMORY => true, VERSIONS => 5}

Ex- alter 'student','Personal\_details', {NAME => 'Admission\_details', IN\_MEMORY => true, VERSIONS => 5}

Before alter command table structure in Hbase

| Personal\_details | | | Education\_details | | |
| --- | --- | --- | --- | --- | --- |
| Name | Age | Address | Course | Year | Grade |

After alter command

| Personal\_details | | | Admission\_details | | | Education\_details | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Age | Address |  |  |  | Course | Year | Grade |

* **Deleting column family names from table**

alter 'sun', 'delete' => 'weather' // only column family

* alter 'education', 'delete' =>’waether'

12. **alter\_status - can get the status of the alter command**

* Which indicates the number of regions of the table that have received the updated schema pass table name

Syntax: alter\_status 'education'(Table Name)

## **3.Data manipulation commands**

These commands will work on the table related to data manipulations such as putting data into a table, retrieving data from a table and deleting schema, etc.

The commands come under these are

1. Count
2. Put
3. Get
4. Delete
5. Delete all
6. Truncate
7. Scan

**1.Count** The command will retrieve the count of a number of rows in a table. The value returned by this one is the number of rows.

**Syntax:**  count <'tablename'>, CACHE =>1000

Ex- hbase> count 'guru99', CACHE=>1000

* Current count is shown per every 1000 rows by default.
* Count interval may be optionally specified.
* Default cache size is 10 rows.

### **2. Put -** It will put a cell ‘value’ at defined or specified table or row or column. It will optionally coordinate time stamp

Syntax: put <'tablename'>,<'rowname'>,<'columnvalue'>,<'value'> **OR**

**put '<HBase\_table\_name>', 'row\_key', '<colfamily:colname>', '<value>'**

**eX- hbase(main):018:0> put 'cdac1',1,'course:Bridgecourse','type:paid'**

**hbase(main):018:0>**put 'cdac1',1,'course:online','type:free'

**hbase(main):020:0> scan 'cdac1'**

**Example:** Here we are placing values into table “guru99” under row r1 and column c1

hbase> put 'guru99', 'r1', 'c1', 'value', 10 **/where r1 is row and c1 is column**

**3.Get-** By using this command, you will get a row or cell contents present in the table.

Syntax: get <'tablename'>, <'rowname'>, {< A

}

eX-hbase> get 'guru99', 'r1'

**Ex - get 'cdac','A1', 'course:Diploma'**

### **4.Delete** This command will delete cell value at defined table of row or column.

Syntax:delete <'tablename'>,<'row name'>,<'column name'>

* Delete must and should match the deleted cells coordinates exactly.
* When scanning, delete cell suppresses older versions of values.

eX-hbase(main):)020:0> delete 'guru99', 'r1', 'c1''.

* The above execution will delete row r1 from column family c1 in table “guru99.”

**5.deleteall** This Command will delete all cells in a given row.

Syntax: deleteall <'tablename'>, <'rowname'>

eX- hbase>deleteall 'guru99', 'r1', 'c1'

6.**Truncate**

Syntax: truncate <tablename>

After truncate of an hbase table, the schema will present but not the records. This command performs 3 functions; those are listed below

* **Disables table if it already presents**
* **Drops table if it already presents**
* **Recreates the mentioned table**

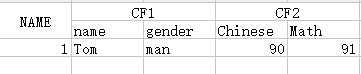
**7.Scan-** Display the Content of HBase Table

Syntax: scan <'tablename'>, {Optional parameters}

* Scanner specifications may include one or more of the following attributes.
* These are TIMERANGE, FILTER, TIMESTAMP, LIMIT, MAXLENGTH, COLUMNS, CACHE, STARTROW and STOPROW.

eX-scan 'guru99'

**Create a table who has more than 01 column families-**



**1. There are two column families CF1 and CF2 in creating table Hbase**

**\*you can only add one column and one column, not multiple columns at the same time.**

This table has two column families, CF1 and CF2. Under CF1 and CF2, there are two columns, name and gender, Chinese and Math

**\*.** {NAME=>'cf1'} / name should be in capital i.e **NAME**

**Example:**

hbase(main):041:0> create 'hbase\_1102', {NAME=>'cf1'}, {NAME=>'cf2'}

**2. Add data to the table. When you want to add data to the table of HBase,**

hbase(main):042:0> put'hbase\_1102', '001','cf1:name','Sumit'

hbase(main):043:0> put'hbase\_1102', '001','cf1:gender','male'

hbase(main):044:0> put'hbase\_1102', '001','cf2:chinese','90'

hbase(main):045:0> put'hbase\_1102', '001','cf2:math','91'

**To Create a table with 03 colun family in Hbase**

**1: create 'student', {NAME=>'Academic'}, {NAME=>'Personal'},{NAME=>'placement'}**

**A.insert data under Academic column family**

a. **hbase(main):4:0>**put'student', '001','Academic:Grade','First'

b. **hbase(main):5:0>**put'student', '001','Academic:Attendance','Good'

c. **hbase(main):6:0>**put'student', '001','Academic:Eassywriting','Average'

d**. hbase(main):7:0>**put'student', '001','Academic:Computerproficency','Yes'

**scan 'student' to check all the fields data inserted**

**B**.**insert data under Personal column family**

**a.** hbase(main):010:0> put'student', '001','Personal:Name','Tarun'

b. hbase(main):011:0> put'student', '001','Personal:mobno','8589658965'

c. hbase(main):012:0> put'student', '001','Personal:height','5'

**C.insert data under Placement column family**

a. hbase(main):004:0> put 'student','001','placement:Company','HCL'

b. hbase(main):005:0> put 'student','001','placement:Year','2018'

c. hbase(main):006:0> put 'student','001','placement:Package','6 Lakh P.A'

**output of the Table:**

**Student:**

| **R**  **o**  **w** | **Academic** | | | | **Personal** | | | | **Placement** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Grade** | **Attendance** | **Essay**  **writing** | **Computer**  **proficency** | **Name** | **Mob**  **No.** | **Height** | **Weight** | **Company** | **Year** | **Package** |  |
| 1 | First | Good | Average | Yes | Tarun | 8589 | 5" | 58 | HCL | 2018 | 6 LPA |  |

**hbase(main):007:0> scan 'student'**

ROW COLUMN+CELL

001 column=Academic:Attendance, timestamp=1660106971926, value=Good

001 column=Academic:Computerproficency, timestamp=1660107016304, value=Yes

001 column=Academic:Eassywriting, timestamp=1660106995630, value=Average

001 column=Academic:Grade, timestamp=1660106942707, value=First

001 column=Personal:Name, timestamp=1660107080883, value=Tarun

001 column=Personal:height, timestamp=1660107113454, value=5

001 column=Personal:mobno, timestamp=1660107098959, value=8589658965

001 column=placement:Company, timestamp=1660107453461, value=HCL

001 column=placement:Package, timestamp=1660107499919, value=6 Lakh P.A

001 column=placement:Year, timestamp=1660107478876, value=2018

## **4. Cluster Replication Commands**

* These commands work on cluster set up mode of HBase.
* For adding and removing peers to cluster and to start and stop replication these commands are used in general.

| **Command** | **Functionality** |
| --- | --- |
| add\_peer | Add peers to cluster to replicate  hbase> add\_peer ‘3’, zk1,zk2,zk3:2182:/hbase-prod |
| remove\_peer | Stops the defined replication stream.  Deletes all the metadata information about the peer  hbase> remove\_peer ‘1’ |
| start\_replication | Restarts all the replication features  hbase> start\_replication |
| stop\_replication | Stops all the replication features  hbase>stop\_replication |

HBase architecture always has “**Single Point Of Failure**” feature, and there is no exception handling mechanism associated with it

## **Performance Bottlenecks in HBase**

* In any production environment, HBase is running with a cluster of more than 5000 nodes, only Hmaster acts as the master to all the slaves Region servers. If Hmaster goes down, it can be only be recovered after a long time. Even though the client is able to connect region server. Having another master is possible but only one will be active. It will take a long time to activate the second Hmaster if the main Hmaster goes down. So, Hmaster is a performance bottleneck.
* In HBase, we cannot implement any cross data operations and joining operations, of course, we can implement the joining operations using [MapReduce](https://www.guru99.com/introduction-to-mapreduce.html), which would take a lot of time to designing and development. Tables join operations are difficult to perform in HBase. In some use case, its impossible to create join operations that related to tables that are present in HBase
* HBase would require new design when we want to migrate data from RDBMS external sources to HBase servers. However, this process takes a lot of time.
* HBase is really tough for querying. We may have to integrate HBase with some[SQL](https://www.guru99.com/sql.html) layers like[Apache](https://www.guru99.com/apache.html)phoenix where we can write queries to trigger the data in the HBase. It’s really good to have Apache Phoenix on top of HBase.
* Another drawback with HBase is that, we cannot have more than one indexing in the table, only row key column acts as a primary key. So, the performance would be slow when we wanted to search on more than one field or other than Row key. This problem we can overcome by writing MapReduce code, integrating with [Apache SOLR](https://www.guru99.com/apache-solr-tutorial.html) and with Apache Phoenix.
* Slow improvements in the security for the different users to access the data from HBase.
* HBase doesn’t support partial keys completely
* HBase allows only one default sort per table
* It’s very difficult to store large size of binary files in HBase
* The storage of HBase will limit real-time queries and sorting
* Key lookup and Range lookup in terms of searching table contents using key values, it will limit queries that perform on real time
* Default indexing is not present in HBase. Programmers have to define several lines of code or script to perform indexing functionality in HBase
* Expensive in terms of Hardware requirements and memory blocks allocations.
  + More servers should be installed for distributed cluster environments (like each server for NameNode, DataNodes, [ZooKeeper](https://www.guru99.com/zookeeper-tutorial.html), and Region Servers)
  + Performance wise it require high memory machines
  + Costing and maintenance wise it is also higher

# **Advantages of HBase**

Here, we will learn what are the pros/benefits of HBase:

* Can store large data sets on top of HDFS file storage and will aggregate and analyze billions of rows present in the HBase tables
* In HBase, the database can be shared
* Operations such as data reading and processing will take small amount of time as compared to traditional relational models
* Random read and write operations
* For online analytical operations, HBase is used extensively.
* For example: In banking applications such as real-time data updates in ATM machines, HBase can be used.

# **Here are the important cons/limitations of HBase:**

* We cannot expect completely to use HBase as a replacement for traditional models. Some of the traditional models features cannot support by HBase
* HBase cannot perform functions like SQL. It doesn’t support SQL structure, so it does not contain any query optimizer
* HBase is CPU and Memory intensive with large sequential input or output access while as Map Reduce jobs are primarily input or output bound with fixed memory. HBase integrated with Map-reduce jobs will result in unpredictable latencies
* HBase integrated with pig and[Hive](https://www.guru99.com/hive-tutorials.html)jobs results in some time memory issues on cluster
* In a shared cluster environment, the set up requires fewer task slots per node to allocate for HBase CPU