**Hive VS HBase**

Comparing Hive with HBase is like comparing Google with Facebook - although they compete over the same turf (our private information), they don’t provide the same functionality. But things can get confusing for the Big Data beginner when trying to understand what Hive and HBase do and when to use each one of them. Let’s try and clear it up.

[**Apache Hive**](https://hive.apache.org/) is a data warehouse infrastructure built on top of Hadoop. It allows for querying data stored on HDFS for analysis via HQL, an SQL-like language that gets translated to MapReduce jobs. Despite providing SQL functionality, Hive does not provide interactive querying yet - it only runs batch processes on Hadoop.

[**Apache HBase**](https://hbase.apache.org/) is a NoSQL key/value store which runs on top of HDFS. Unlike Hive, HBase operations run in real-time on its database rather than MapReduce jobs. HBase is partitioned to tables, and tables are further split into column families. Column families, which must be declared in the schema, group together a certain set of columns (columns don’t require schema definition). For example, the "message" column family may include the columns: "to", "from", "date", "subject", and "body". Each key/value pair in HBase is defined as a cell, and each key consists of row-key, column family, column, and time-stamp. A row in HBase is a grouping of key/value mappings identified by the row-key. HBase enjoys Hadoop’s infrastructure and scales horizontally using off the shelf servers.

**Features**

**Hive** can help the SQL savvy to run MapReduce jobs. Since it’s JDBC compliant, it also integrates with existing SQL based tools. Running Hive queries could take a while since they go over all of the data in the table by default. Nonetheless, the amount of data can be limited via Hive’s partitioning feature. Partitioning allows running a filter query over data that is stored in separate folders, and only read the data which matches the query. It could be used, for example, to only process files created between certain dates, if the files include the date format as part of their name.

**HBase** works by storing data as key/value. It supports four [primary operations](http://hbase.apache.org/book/data_model_operations.html): **put** to add or update rows, **scan** to retrieve a range of cells, **get** to return cells for a specified row, and **delete** to remove rows, columns or column versions from the table. Versioning is available so that previous values of the data can be fetched (the history can be deleted every now and then to clear space via [HBase compactions](http://blog.cloudera.com/blog/2013/12/what-are-hbase-compactions/)). Although HBase includes tables, a schema is only required for tables and column families, but not for columns, and it includes increment/counter functionality.

**Limitations**

**Hive** does not currently support update statements. Additionally, since it runs batch processing on Hadoop, it can take minutes or even hours to get back results for queries. Hive must also be provided with a predefined schema to map files and directories into columns and it is not ACID compliant.

**HBase** queries are written in a custom language that needs to be learned. SQL-like functionality can be achieved via [Apache Phoenix](http://phoenix.incubator.apache.org/), though it comes at the price of maintaining a schema. Furthermore, HBase isn’t fully ACID compliant, although it does support [certain properties](http://hbase.apache.org/acid-semantics.html). Last but not least - in order to run HBase, **[ZooKeeper](http://zookeeper.apache.org/) is required - a server for distributed coordination such as configuration, maintenance, and naming.**

**Use Cases**

**Hive** should be used for analytical querying of data collected over a period of time - for instance, to calculate trends or website logs. Hive should not be used for real-time querying since it could take a while before any results are returned.

**HBase** is perfect for real-time querying of Big Data. Facebook use it for [messaging](https://www.facebook.com/note.php?note_id=454991608919) and real-time analytics. They may even be using it to count Facebook likes.

**Summary**

**Hive** and **HBase** are two different Hadoop based technologies - Hive is an SQL-like engine that runs MapReduce jobs, and HBase is a NoSQL key/value database on Hadoop. But hey, why not use them both? Just like Google can be used for search and Facebook for social networking, Hive can be used for analytical queries while HBase for real-time querying. Data can even be read and written from Hive to HBase and back again.

**HBase FAQS**

**1) Explain what is Hbase?**

Hbase is a column-oriented database management system which runs on top of HDFS (Hadoop Distribute File System). Hbase is not a relational data store, and it does not support structured query language like SQL.

In Hbase, a master node regulates the cluster and region servers to store portions of the tables and operates the work on the data.

**2) Explain why to use Hbase?**

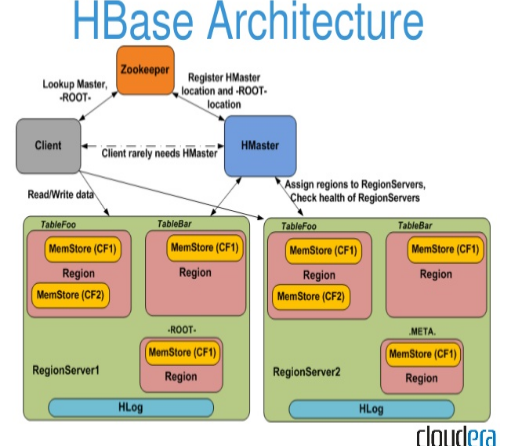
* High capacity storage system
* Distributed design to cater large tables
* Column-Oriented Stores
* Horizontally Scalable
* High performance & Availability
* Base goal of Hbase is millions of columns, thousands of versions and billions of rows
* Unlike HDFS (Hadoop Distribute File System), it supports random real time CRUD operations

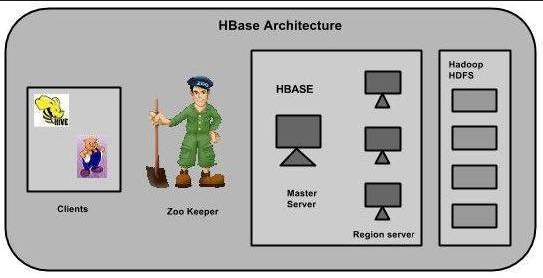
**3) Mention what are the key components of Hbase?**

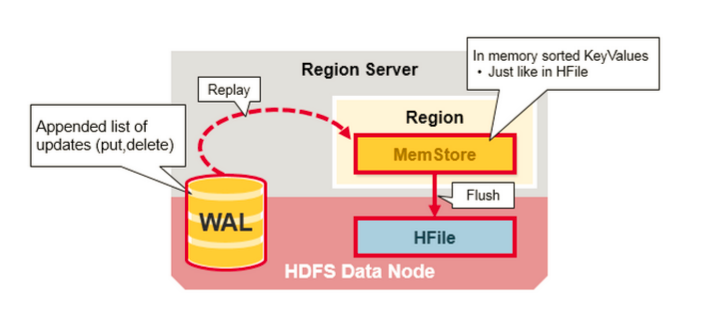
* **Zookeeper:** It does the co-ordination work between client and Hbase Maser
* **Hbase Master:** Hbase Master monitors the Region Server
* **RegionServer:** RegionServer monitors the Region
* **Region:** It contains in memory data store (MemStore) and Hfile.
* **Catalog Tables:** Catalog tables consist of ROOT and META

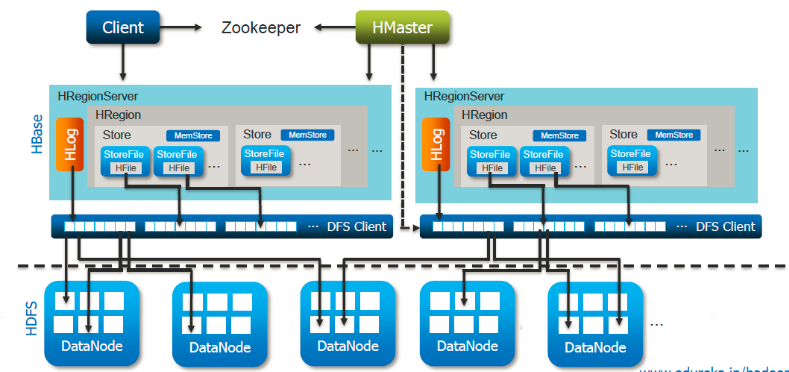
**4) Explain what does Hbase consists of?**

* Hbase consists of a set of tables
* And each table contains rows and columns like traditional database
* Each table must contain an element defined as a Primary Key(row key)
* Hbase column denotes an attribute of an object









**5) Mention how many operational commands in Hbase?**

Operational command in Hbases is about five types

* **Get** - to return cells for a specified row (**hbase> get ‘t1’, ‘r1’, ‘c1’**)
* **Put** - to add or update rows(**hbase> put ‘t1’, ‘r1’, ‘c1’, ‘value’, ts1**)
* **Delete** - to remove rows, columns or column versions from the table (**hbase> delete ‘t1’, ‘r1’, ‘c1’, ts1**)
* **Scan -** to retrieve a range of cells(**hbase> scan ‘t1’, {COLUMNS => [‘c1’, ‘c2’], LIMIT => 10, STARTROW => ‘xyz’}**  
   **hbase> scan ‘t1’, {COLUMNS => ‘c1’, TIMERANGE => [1303668804, 1303668904]}**
* **hbase> scan ‘t1’, {RAW => true, VERSIONS => 10}**)
* **Increment –** Increments a cell ‘value’ at specified table/row/column coordinates. **(hbase> incr ‘t1’, ‘r1’, ‘c1’ or hbase> incr ‘t1’, ‘r1’, ‘c1’, 1 or hbase> incr ‘t1’, ‘r1’, ‘c1’, 1)**
* [**https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/**](https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/)

**6) Explain what is WAL and Hlog in Hbase?**

WAL (Write Ahead Log) is similar to MySQL BIN log; it records all the changes occur in data. It is a standard sequence file by[Hadoop](http://www.guru99.com/bigdata-tutorials.html)and it stores HLogkey’s.  These keys consist of a sequential number as well as actual data and are used to replay not yet persisted data after a server crash. So, in crash of server failure WAL work as a life-line and retrieves the lost data’s.

**7) When you should use Hbase?**

* **Data size is huge:** When you have tons and millions of records to operate
* **Complete Redesign:** When you are moving RDBMS to Hbase, you consider it as a complete re-design then mere just changing the ports
* **SQL-Less commands:** You have several features like transactions; inner joins, typed columns, etc.
* **Infrastructure Investment:** You need to have enough cluster for Hbase to be really useful

**8) In Hbase what is column families?**

Column families comprise the basic unit of physical storage in Hbase to which features like compressions are applied.

**9) Explain what is the row key?**

Row key is defined by the application. As the combined key is pre-fixed by the rowkey, it enables the application to define the desired sort order. It also allows logical grouping of cells and make sure that all cells with the same rowkey are co-located on the same server.

**10) Explain deletion in Hbase? Mention what are the three types of tombstone markers in Hbase?**

When you delete the cell in Hbase, the data is not actually deleted but a tombstone marker is set, making the deleted cells invisible.  Hbase deleted are actually removed during compactions.

Three types of tombstone markers are there:

* **Version delete marker**: For deletion, it marks a single version of a column
* **Column delete marker**: For deletion, it marks all the versions of a column
* **Family delete marker**: For deletion, it marks of all column for a column family

**11) Explain how does Hbase actually delete a row?**

In Hbase, whatever you write will be stored from RAM to disk, these disk writes are immutable barring compaction. During deletion process in Hbase, major compaction process delete marker while minor compactions don’t. In normal deletes, it results in a delete tombstone marker- these delete data they represent are removed during compaction.

Also, if you delete data and add more data, but with an earlier timestamp than the tombstone timestamp, further **Gets** may be masked by the delete/tombstone marker and hence you will not receive the inserted value until after the major compaction.

**12) Explain what happens if you alter the block size of a column family on an already occupied database?**

When you alter the block size of the column family, the new data occupies the new block size while the old data remains within the old block size. During data compaction, old data will take the new block size.  New files as they are flushed, have a new block size whereas existing data will continue to be read correctly. All data should be transformed to the new block size, after the next major compaction.

**13) Mention the difference between Hbase and Relational Database?**

|  |  |
| --- | --- |
| **Hbase** | **Relational Database** |
| * It is schema-less * It is a column-oriented data store * It is used to store de-normalized data * It contains sparsely populated tables * Automated partitioning is done in Hbase | | * It is a schema based database * It is a row-oriented data store * It is used to store normalized data * It contains thin tables * There is no such provision or built-in support for partitioning |

<https://www.tutorialspoint.com/hbase/hbase_quick_guide.htm>

<https://learnhbase.wordpress.com/2013/03/02/hbase-shell-commands/>

<http://www.guru99.com/handling-tables-hbase.html>