1. Hadoop is, essentially, HDFS (Hadoop Distributed File System) and MapReduce. ... Technically speaking, your question should be on the difference between HBase and HDFS. HBase is a non-relational database that can run on top of Hadoop and provides you random data access/querying capabilities.
2. HBase Tables – Logical collection of rows stored in individual partitions known as Regions.

HBase Row – Instance of data in a table.

RowKey -Every entry in an HBase table is identified and indexed by a RowKey.

Columns - For every RowKey an unlimited number of attributes can be stored.

Column Family – Data in rows is grouped together as column families and all columns are stored together in a low level storage file known as HFile.

1. The most important consideration when looking at HBase is that, while it is a great solution to many problems, it is not a silver bullet. HBase is not optimized for classic transactional applications or even relational analytics. It is also not a complete substitute for HDFS when doing large batch MapReduce.
2. HBase has two run modes: Standalone HBase and Distributed. Out of the box, HBase runs in standalone mode. Whatever your mode, you will need to configure HBase by editing files in the HBase conf directory. At a minimum, you must edit conf/hbase-env.sh to tell HBase which **java** to use. In this file you set HBase environment variables such as the heapsize and other options for the JVM, the preferred location for log files, etc. Set JAVA\_HOME to point at the root of your **java** install.
3. HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.
4. The documents stored in the database can have varying sets of fields, with different types for each field.  Of course, when using the database for real problems, the data does have a fairly consistent structure.
5. Basically you should try to stick with one and add a second if you have radically different access patterns. . HBase is sparse so it won't take more space and you can still get just one "family" with a column Prefix filter on scans if you need to.
6. We can use one connection per application because creating it is heavy and it is thread safe. Make sure however that you close table and admin that you get from the connection.
7. Memstore: Memstore is an in-memory storage, hence the Memstore utilizes the in-memory storage of each data node to store the logs. Rows are written to theMemStore. The data in the Memstore is ordered. When certain thresholds are met, Memstore data gets flushed into HFile. Every time Memstore flush happens one HFile created for each Column Family

HFile: HFile are the actual storage files i.e. physical representation of data in HFile, specifically created to serve one purpose: store HBase’s data fast and efficiently. Clients do not read HFile directly but go through region servers to get to the data.

1. HBase will try to combine HFiles to reduce the maximum number of disk seeks needed for a read. This process is called compaction.
2. Logically attributes are Unordered.

They are of 2 types. They are Physical models and Logical model

1. Row-keys are used to identify a row uniquely in Hbase. If you want two rows to have identical keys, then you are missing something.
2. When reading data from HBase using Get or Scan operations, you can use custom filters to return a subset of results to the client. While this does not reduce server-side IO, it does reduce network bandwidth and reduces the amount of data the client needs to process. Filters are generally used using the Java API, but can be used from HBase Shell for testing and debugging purposes.
3. The four primary data model operations are Get, Put, Scan, and Delete.
4. HBase provides a TableInputFormat, to which you provided a table scan, that splits the rows resulting from the table scan into the regions in which those rows reside. The map process is passed an ImmutableBytesWritable that contains the row key for a row and a Result that contains the columns for that row.
5. Region Server runs on an HDFS data node and has the following components:  
   WAL  
   • Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't  
   yet been persisted to permanent storage; it is used for recovery in the case of failure.  
   BlockCache  
   • It is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when  
   full.  
   MemStore  
   • It is the write cache. It stores new data which has not yet been written to disk. It is sorted before  
   writing to disk. There is one MemStore per column family per region.  
   Hfiles  
   • They store the rows as sorted KeyValues on disk.