**When should we use HBASE, list some of the scenarios for the same in real time.**

**Ans)**

HBase offers lot of good functionalities, but it is still not a **‘Fit for All’** solution. Following are some of the key areas to be considered before finalizing HBase for your application.

**Data volume:** The volume of data is the most common point to be considered. You should have peta bytes of data to be processed in a distributed environment. Otherwise, for a small amount of data, it will be stored and processed in a single node, keeping other nodes idle. So, it will be a misuse of technology framework.

**Application Types:** HBase is not suitable for transactional applications, large volume MapReduce jobs, relational analytics, etc. It is preferred when you have a variable schema with slightly different rows. It is also suitable when you are going for a key dependent access to your stored data.

**Hardware environment:** HBase runs on top of HDFS. And HDFS works efficiently with a large number of nodes (minimum 5). So, if you have good hardware support, then HBase can be a good selection.

**No requirement of relational features:** Your application should not have any requirement for RDBMS features like transaction, triggers, complex query, complex joins etc. If you can build your application without these features, then go for HBase.

**Quick access to data:**If you need a random and real time access to your data, then HBase is a suitable candidate. It is also a perfect fit for storing large tables with multi structured data. It gives ‘flashback’ support to queries, which makes it more suitable for fetching data in a particular instance of time.

Apart from the above points, HBase is also suitable when you need fault tolerant, fast and usable data management in a non-relational environment.

[Apache HBase](https://hbase.apache.org/) is one of the most popular non-relational databases built on top of Hadoop and HDFS (Hadoop Distributed File system). It is also known as the Hadoop database. As an Apache project, HBase is an open-source, versioned and distributed NoSQL DB written in the Java language. It is built on Google’s [Bigtable](http://research.google.com/archive/bigtable.html) concepts. Apache HBase is suitable for use cases where you need real time and random read/write access to huge volumes of data (Big data). As HBase runs on top of HDFS, the performance is also dependent on the hardware support. We need to provide sufficient number of nodes (minimum 5) to get a better performance.

**Real time scenarios:**

There are a lot of real-life implementations of HBase. Some of the important use cases are:

* Use of HBase by Mozilla: They generally store all crash data in HBase
* Use of HBase by Facebook: Facebook uses HBase storage to store real-time messages.

1. **What are the different modes in which Hbase can be run?**

**Ans)**

HBase has two run modes:  [“Standalone HBase”](http://hbase.apache.org/0.94/book/standalone_dist.html#standalone) and  [“Distributed”](http://hbase.apache.org/0.94/book/standalone_dist.html#distributed). Out of the box, HBase runs in standalone mode. To set up a distributed deploy, you will need to configure HBase by editing files in the HBase conf directory.

Whatever your mode, you will need to 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.

* **Standalone HBase**

This is the default mode. Standalone mode is what is described in the  [“Quick Start”](http://hbase.apache.org/0.94/book/quickstart.html) section. In standalone mode, HBase does not use HDFS -- it uses the local filesystem instead -- and it runs all HBase daemons and a local ZooKeeper all up in the same JVM. Zookeeper binds to a well known port so clients may talk to HBase.

* **Distributed**

Distributed mode can be subdivided into distributed but all daemons run on a single node -- a.k.a *pseudo-distributed*-- and *fully-distributed* where the daemons are spread across all nodes in the cluster.

Distributed modes require an instance of the *Hadoop Distributed File System* (HDFS). See the Hadoop [requirements and instructions](http://hadoop.apache.org/common/docs/r1.1.1/api/overview-summary.html#overview_description) for how to set up a HDFS. Before proceeding, ensure you have an appropriate, working HDFS.

Below we describe the different distributed setups. Starting, verification and exploration of your install, whether a *pseudo-distributed* or *fully-distributed*configuration is described in a section that follows, [“Running and Confirming Your Installation”](http://hbase.apache.org/0.94/book/standalone_dist.html#confirm). The same verification script applies to both deploy types.

* **Pseudo-distributed**

A pseudo-distributed mode is simply a distributed mode run on a single host. Use this configuration testing and prototyping on HBase. Do not use this configuration for production nor for evaluating HBase performance.

First, setup your HDFS in [pseudo-distributed mode](http://hadoop.apache.org/docs/r1.0.3/single_node_setup.html).

Next, configure HBase. Below is an example conf/hbase-site.xml. This is the file into which you add local customizations and overrides  and [“HDFS Client Configuration”](http://hbase.apache.org/0.94/book/standalone_dist.html#hdfs_client_conf). Note that the hbase.rootdir property points to the local HDFS instance.

**3)Need and working of zookeeper in Hbase?**

**Ans)**

ZooKeeper is a high-performance coordination service for distributed applications(like HBase). It exposes common services like naming, configuration management, synchronization, and group services, in a simple interface so you don't have to write them from scratch. You can use it off-the-shelf to implement consensus, group management, leader election, and presence protocols. And you can build on it for your own, specific needs.

HBase relies completely on Zookeeper. HBase provides you the option to use its built-in Zookeeper which will get started whenever you start HBAse. But it is not good if you are working on a production cluster. In such scenarios it's always good to have a dedicated Zookeeper cluster and integrate it with your HBase cluster.