HBase System

HBase is an open source, distributed database, developed by Apache Software foundation.

Initially, it was Google Big Table, afterwards it was re-named as HBase and is primarily written in Java.

HBase can store massive amounts of data from terabytes to petabytes.

It is column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is like Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

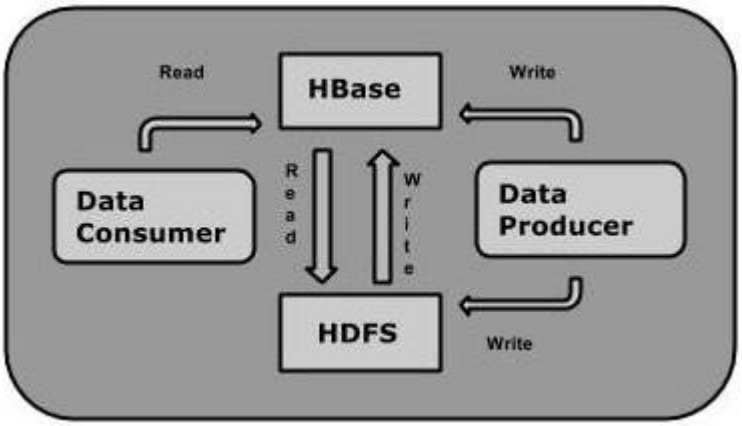
It is a part of the Hadoop ecosystem that provides random read/write access to data in the Hadoop File System.

Hadoop supports sequential access and batch processing. Hence to read or search a single record ,entire data need to be scanned because Hadoop is reads data sequentially.

Drawbacks of Hadoop

1. It allows batch processing
2. Supports sequential read, random access is not supported

To overcome these draw backs HBase is used on top of Hadoop.



Differences between HDFs and HBase

HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

Storage Mechanism in HBase

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp.

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.
-

Row wise storage and columnwise storage

Row stores are good for transaction processing. Column stores are good for analytical query models. Row stores can write data very quickly, whereas a column store is good at aggregating large volumes of data for a subset of columns.

One of the benefits of a columnar database is its crazy fast query speeds. In some cases, queries that took minutes or hours are completed in seconds. This makes columnar databases a good choice in a query-heavy environment. But you must make sure that the queries you run are really suited to a columnar database.

Data Storage

Let's think about a basic database, like a stockbroker's transaction records. In a row store, each client would have a record with their basic information – name, address, phone number, etc. – in a single table. It's likely that each record would have a unique identifier. In our case, it would probably be an account_number.

There is another table that stored stock transactions. Again, each transaction is uniquely identified by something like a transaction_id. Each transaction is associated to one account_number, but each account_number is associated with multiple transactions. This provides us with a one-to-many relationship and is a classic example of a transactional database.

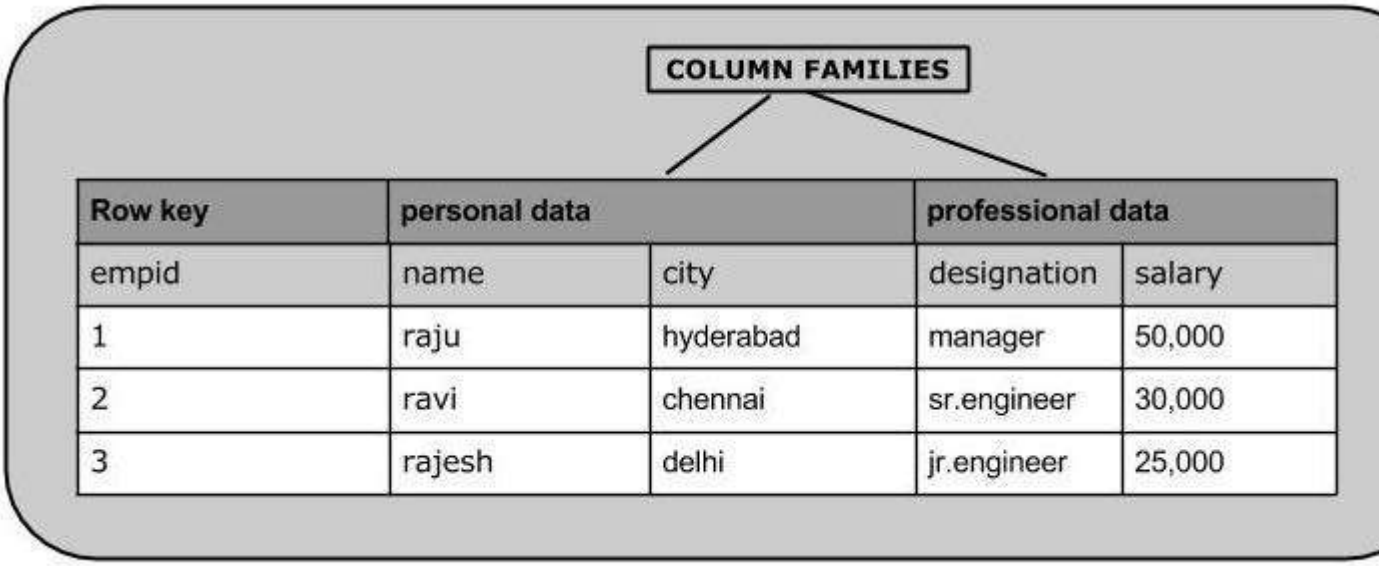
We store all these tables on a disk and, when we run a query, the system might access lots of data before it determines what information is relevant to the specific query. If we want to know the account_number, first_name, last_name, stock, and purchase_price for a given time period, the system needs to access all of the information for the two tables, including fields that may not be relevant to the query. It then performs a join to relate the two tables' data, and then it can return the information. This can be inefficient at scale, and this is just one example of a query that would probably run faster on a columnar database.

With a columnar database, each field from each table is stored in its own file or set of files. In our example database, all account_number data is stored in one file, all transaction_id data is stored in another file, and so on. This provides some efficiencies when running queries against wide tables, since it is unlikely that a query needs to return all of the fields in a single table. In the query example above, we'd only need to access the files that contained data from the requested fields. You can ignore all other fields that exist in

the table. This ability to minimize i/o is one of the key reasons columnar databases can perform much faster.

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

The following image shows column families in a column-oriented database:



HBase and RDBMS

HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.

It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.

Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

Applications of HBase

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.