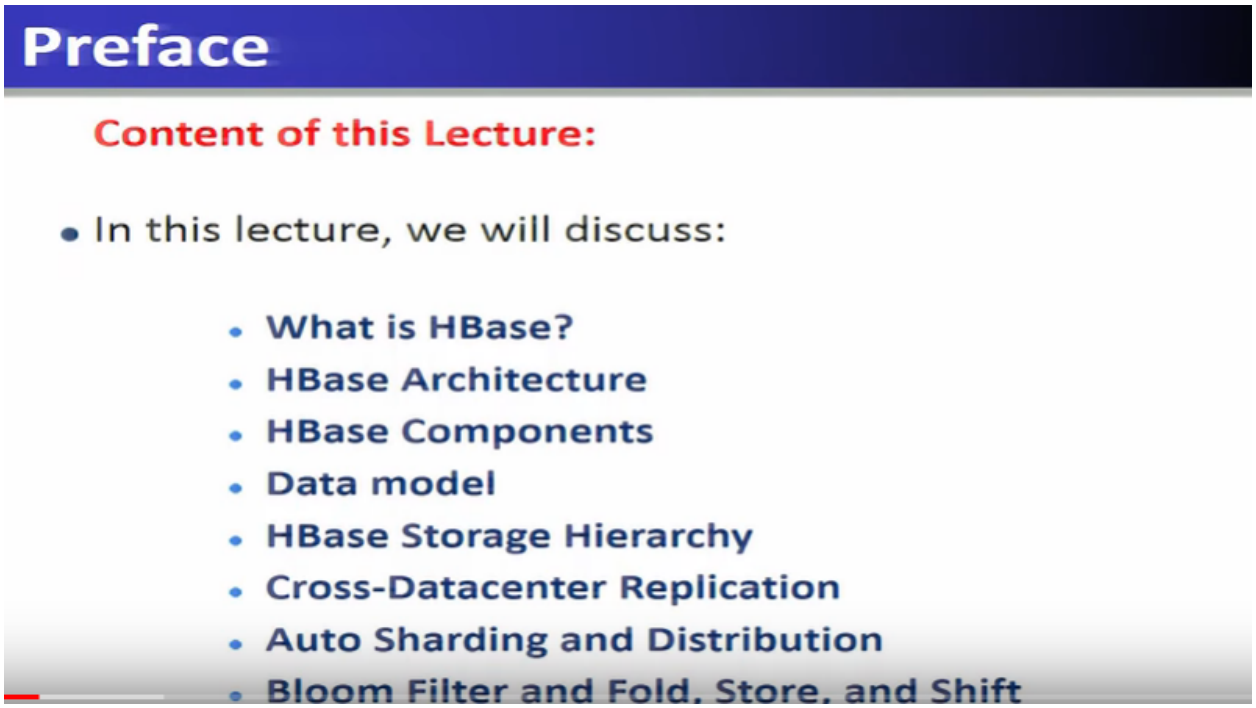
Lecture - 18
Design of HBase

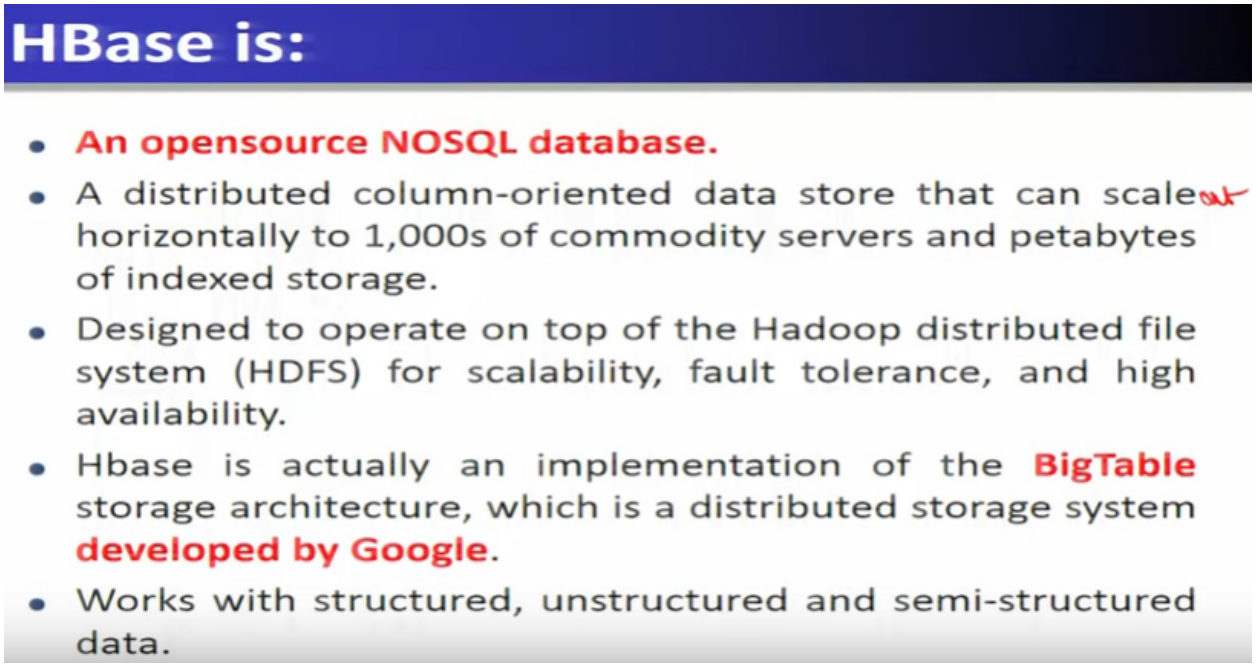
Design of HBase,

Refer Slide Time :(0: 17)



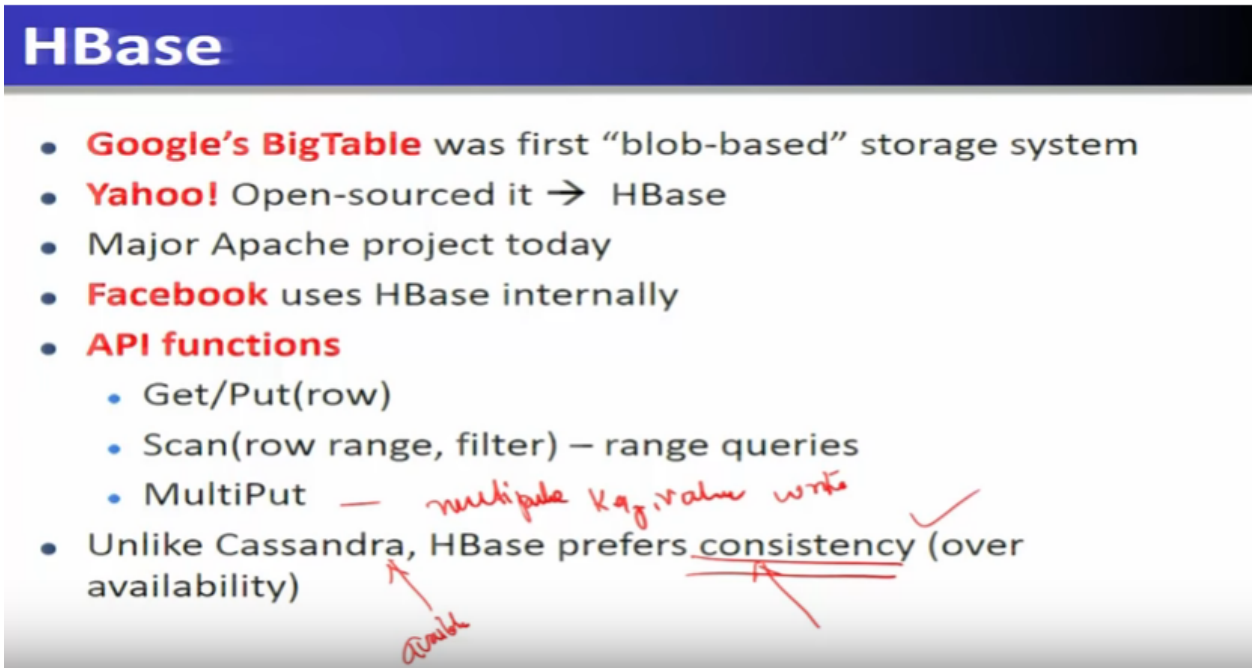
Preface content of this lecture; in this lecture we will discuss, what is HBase? HBase architecture, HBase components, data model, storage hierarchy, cross data center application, auto sharding and distribution, bloom filter, folding, fold store and shift.

Refer Slide Time :(0: 40)



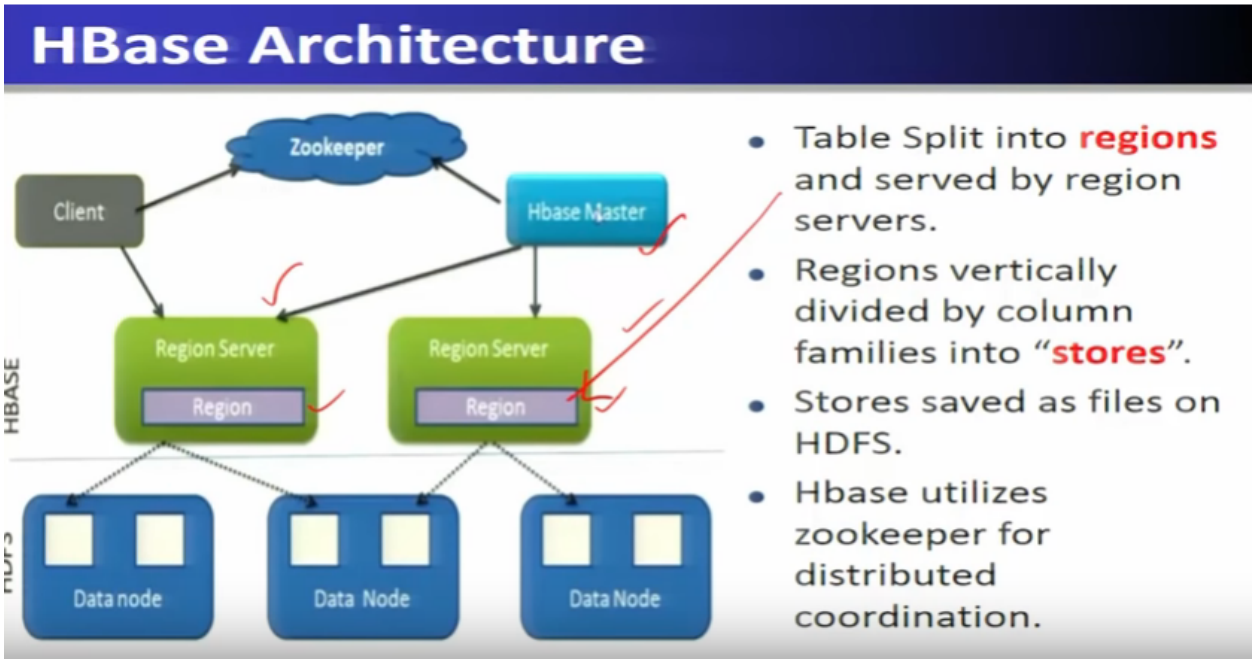
So, HBase is an open source, no sequel database, this is distributed column-oriented data store that can scale horizontally, to thousands of commodity, servers and pet bytes, of indexed storage, because this amount of commodity machines or the cluster, is required, to store the large amount of data, that is why? This particular, aspect is covered into the distributed column-oriented data store and also, it can scale out, horizontally to thousands of servers and pet bytes, of indexed storage. HBase is designed to operate on top of HDFS, for scalability, fault tolerance and high availability, HBase is actually, an implementation of Big Table, which is our storage architecture, given by Google and this is which is a distributed storage system, developed at Google. So, HBase works with a structured, unstructured and semi-structured data.

Refer Slide Time :(1: 59)



So, HBase is basically the is designed as the Google's, Big Table, was first "block-based" storage system and Yahoo! Open sourced this particular concept, a block based and which is now known, as HBase. So, HBase is a major Apache project today and Face book also, use HBase, internally and it has various, API functions, which provides for example, get and put by row, that is the key value pair and scan the dough range and filter all these operations, which are supported to perform the range queries, it is also having a multi put. So, that multiple key value, store can be stored, can be handled, at the same time, unlike Cassandra, HBase, prefers consistency over availability. Now, Cassandra prefers availability, whereas HBase prefers consistency over availability. So, now we are going to see that, where our consistency is a preferred, option for the application, HBase is used and wherever availability, is more important than Cassandra is used that is why? Both of them exist as, the no sequel solution.

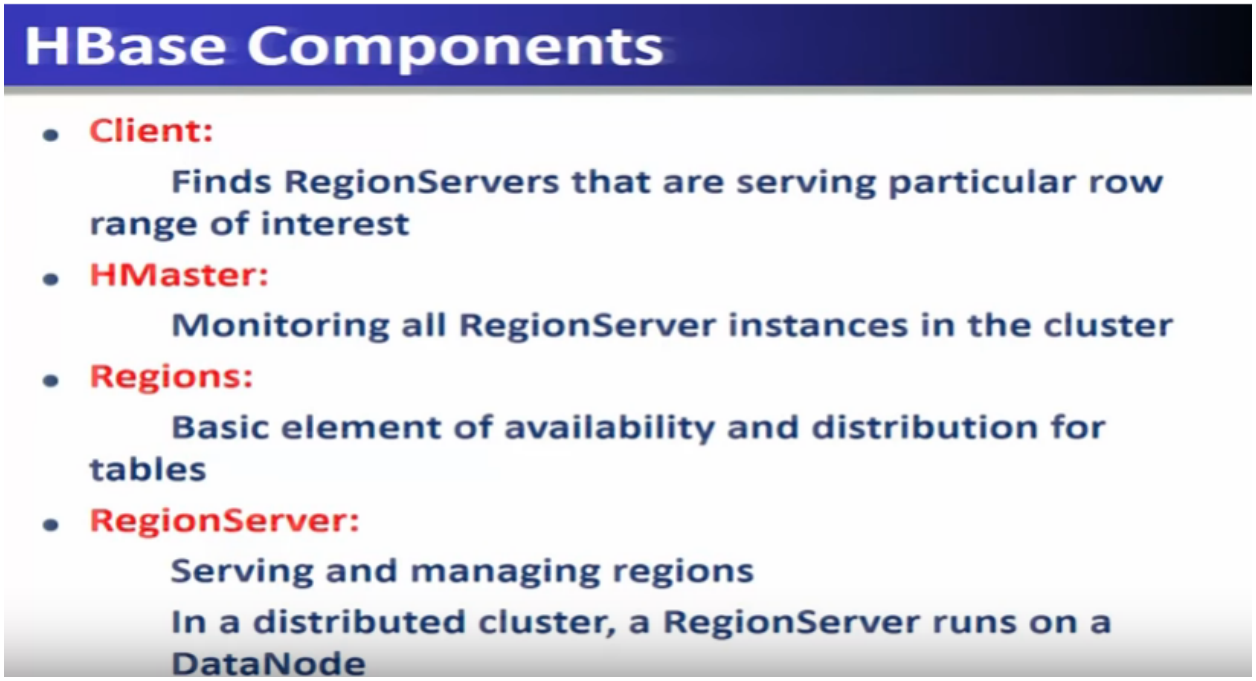
Refer Slide Time :(3:



51)

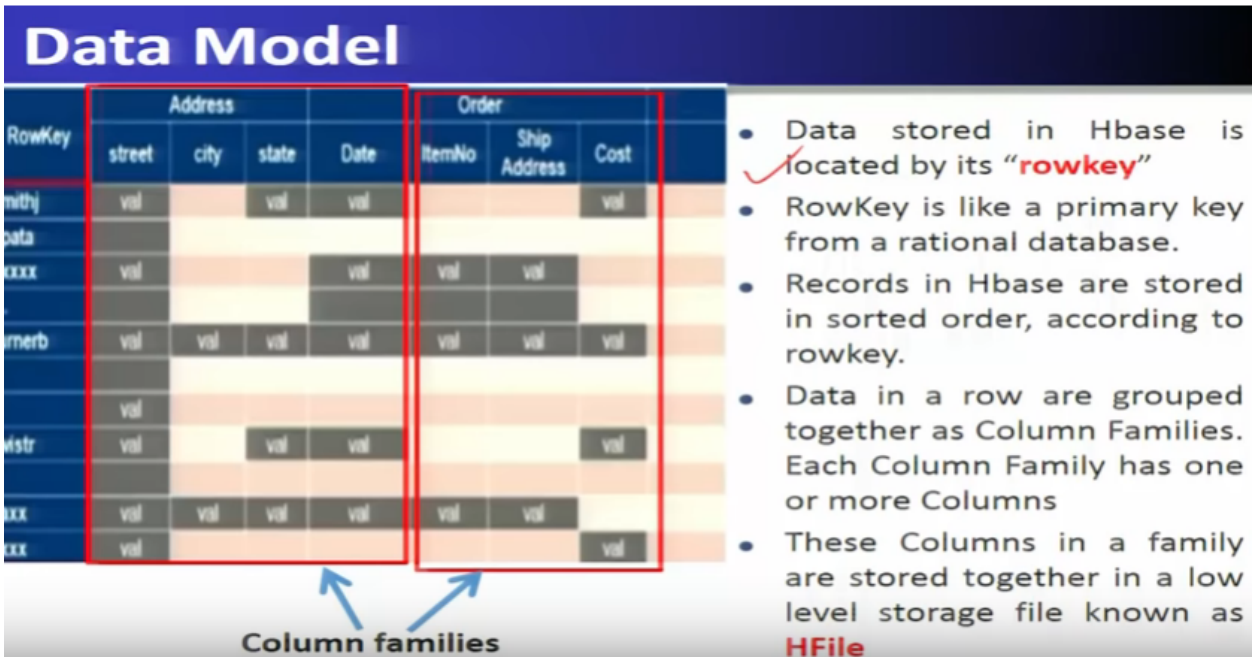
Let us see, some of the important aspects of HBase, architecture. So, HBase has the, the region servers and these region servers are basically handling the region's and there is one HBase master and this zookeeper, has to interact with the HBase master and all the other component and HBase, then as HBase also, deals with the data nodes. So, HBase master has to communicate with the region servers and zookeeper. So, we will see, in more detail about this. So, HBase architecture here the table, is split into the regions and served by the region servers. So, regions vertically, divided by the column, families, into the stores, that we will discuss later on. And stores save, as files on HDFS and as base utilizes, zookeeper for the distributed coordination service.

Refer Slide Time :(5: 11)



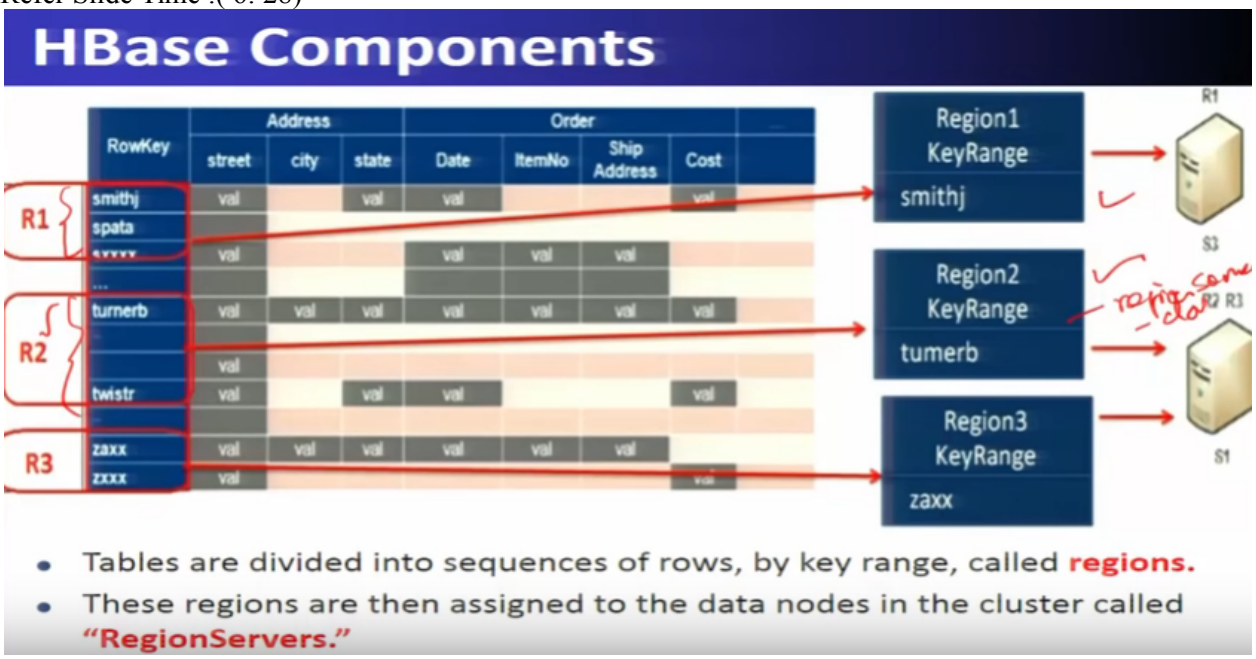
The components in more detail. So, client will find the region servers that are serving particularly the row range of interest. So, then H master, monitors all the region servers, instances in the, in the cluster system and regions our basic element, of availability and distribution for the tables, our region servers, serving and managing the regions and in a distributed cluster region, runs on the data node.

Refer Slide Time :(5: 42)



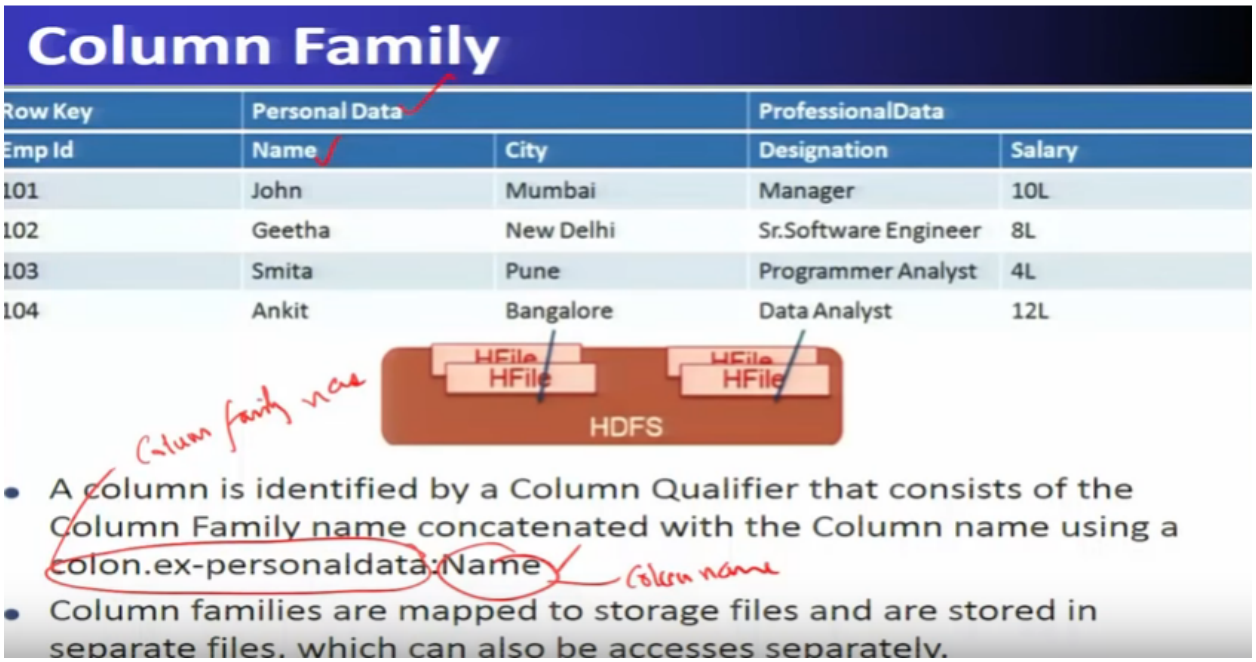
So, this is the, typical layout of the data model, which is also called as the, ‘Column Families’. So, data is stored in HBase, is located by its "rowkey". So, "rowkey" is the primary key, from the, from the notion of relational database management system. So, the records in HBase, are stored in the sorted order according to the rowkeys and the data in a row, are grouped together as the column families and each column family has, one or more columns. These columns in a family are stored together in a low level storage file, which is called as, ‘H File’.

Refer Slide Time :(6: 28)



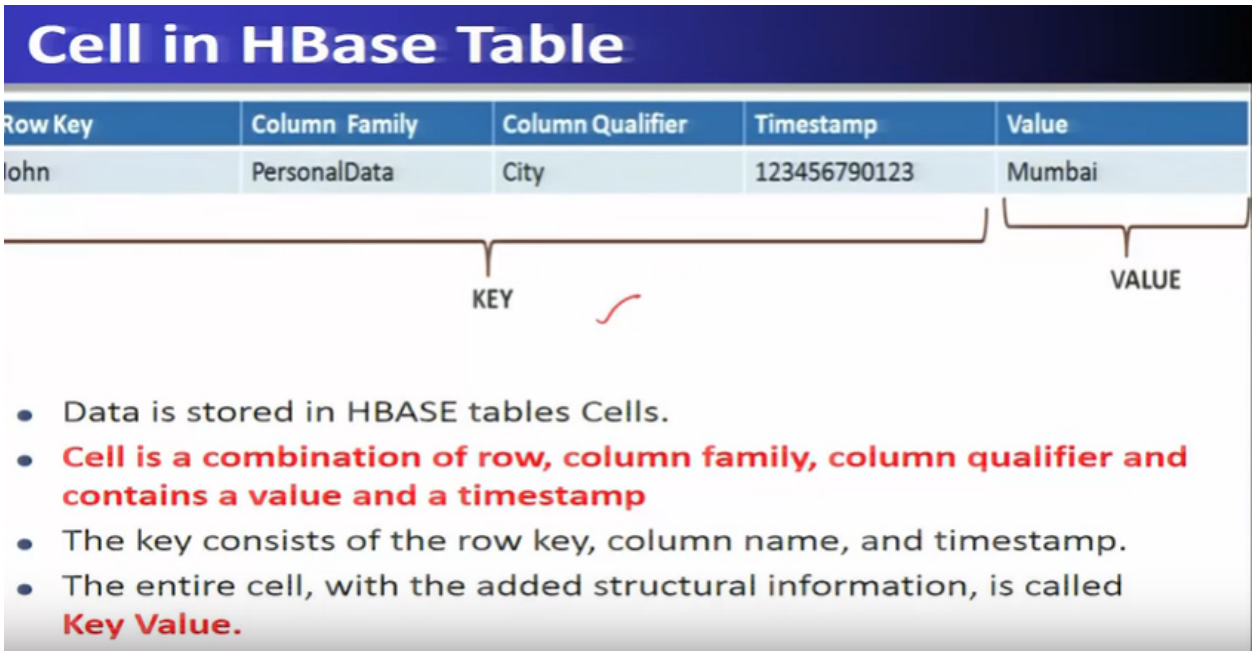
So, the tables are divided into the sequence of rows, by the key range called the, ‘Regions’. So, here we can see that, this particular key range, is basically nothing but, the row 1 and this will be, strode together into the region. So, these regions are then assigned, to the data node, in the cluster called the region servers. Now, that is shown over here. So, again for the example, let us say that, the range the sequence or the key range, the "rowkey” range r2, will store these set off the rows, sequence of the rows and this is will be stored in another, region server and the region servers, these regions are, managed by the data nodes, they are called, they are called the, ‘Region Servers’. Region servers as the data nodes.

Refer Slide Time :(7: 42)



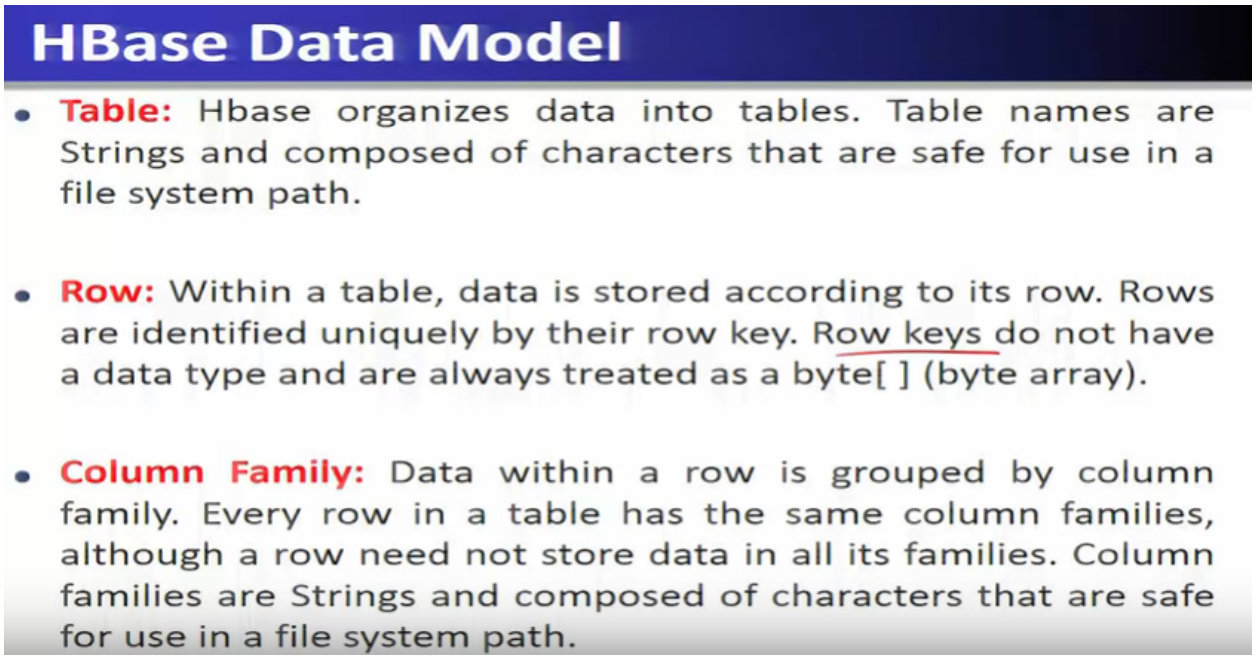
Now, column family, a column is identified by the column qualifier that consists of the column family, name concatenated with the column name using, the coolant dot X personal, personal data colon name. So, here we can see that, this particular column family, name column is identified by, by the columns of the column family name, concatenated with the column name, using the colon. So, this is the column name and the column family name is shown over here, this is the column family name. So, this is shown over here. So, column family name and the, the name will qualify, will identify, the particular column. So, column families are mapped to the storage, files and are stored in a separate files which can also, be accessed separately.

Refer Slide Time :(8: 53)



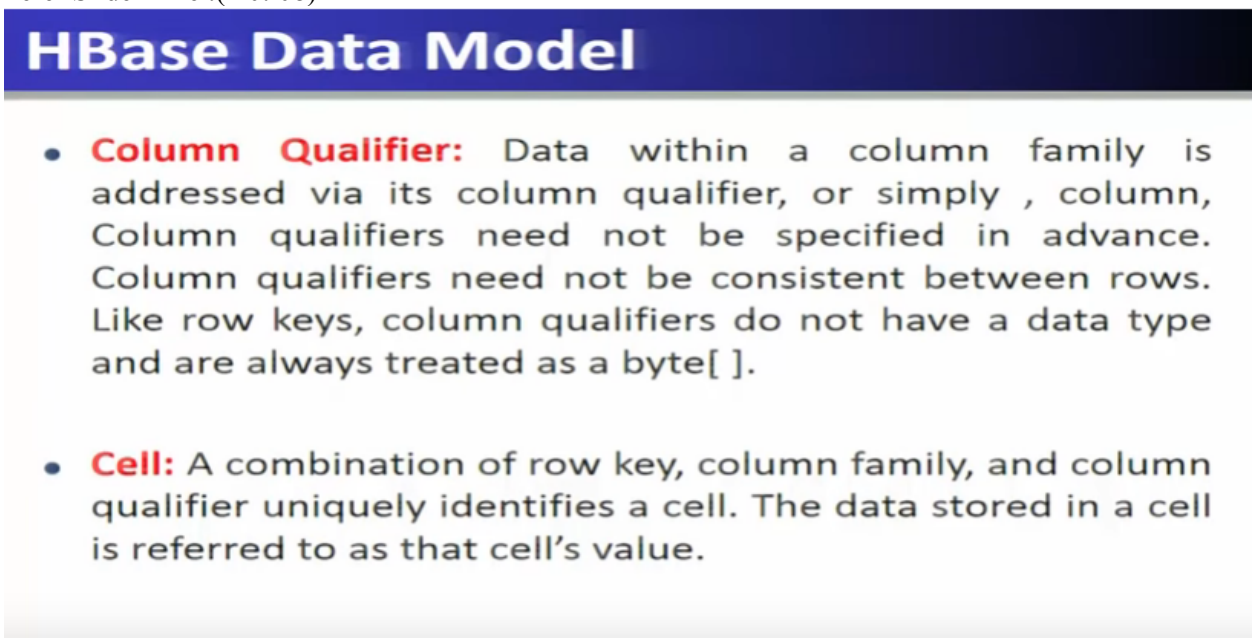
Now, cell in HBase a table data is stored in HBase table cells. And cell is a combination of row, column family and column qualifier and contains a value and a timestamp. So, the key consists of the row key, column name and a timestamp, that is shown here and the entire cell, with the added, structural information is called the, 'Key Value Pair'.

Refer Slide Time :(9: 21)



So, HBase data, model consists of a table, HBase our nicest data into the table and the table names are the string composed of the characters that are saved for use, in the file system path. And the rows within the table, the data is stored according to its row, rows are identified by the arrow key and row keys do not have the data type and are always treated as the byte. So, this aspect we have already covered. So, column family the data within the row, is grouped by the column family, every row, in the table has, the same column family, although the row need not store the data, in all its families, column families are the strings and composed of characters, that are safe for, use in the file system path.

Refer Slide Time :(10: 08)



Now, column qualifier the data, within the column family, is at rest via, the column qualifier and or simply, the column qualifier, need not be specified in the advance and column qualifier, need not be consistent between the rows, like "rowkeys" column qualifier do not have the data type and is always treated as a byte. Cell is a combination of row key, column family and column qualifier, uniquely identify the cell, the data I stored in the cell is referred to as the cell value.

Refer Slide Time :(10: 41)

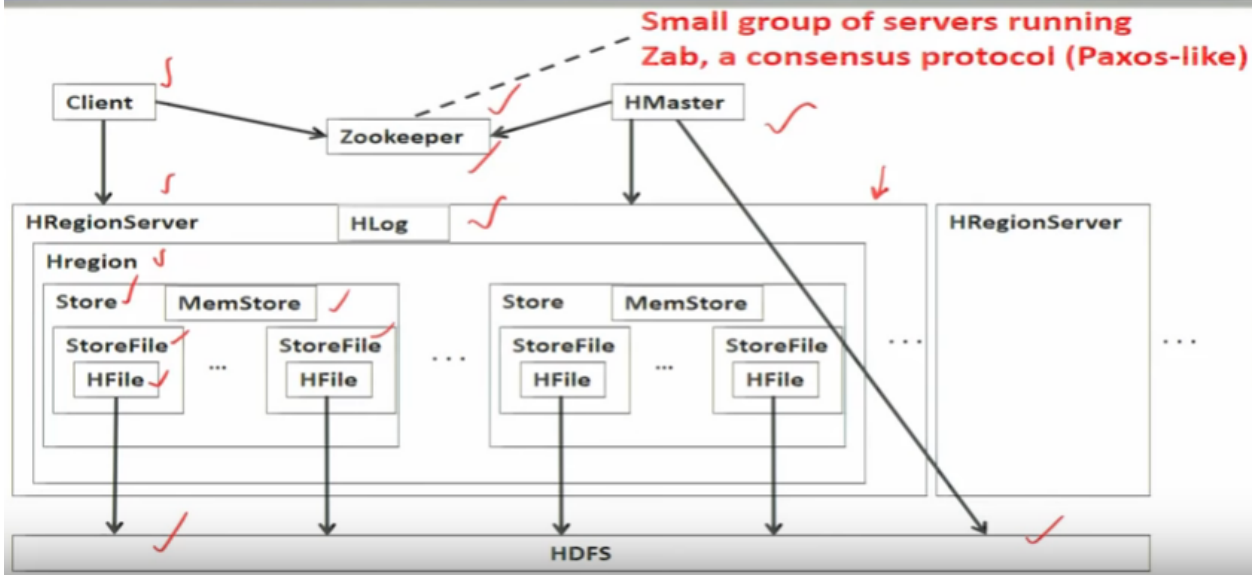
HBase Data Model

- **Timestamp:** Values within a cell are versioned. Versions are identified by their version number, which by default is the timestamp is used.
- If the timestamp is not specified for a read, the latest one is returned. The number of cell value versions retained by Hbase is configured for each column family. The default number of cell versions is three.

And timestamp the values, within the cell or version. Versions are identified by their virgin numbers, which by default is the timestamp, if the timestamp is not is specified, for the read, the latest one is returned, the number of cell values versions, retained by the HBase is configured for each family. The default number, of cell virgins is three.

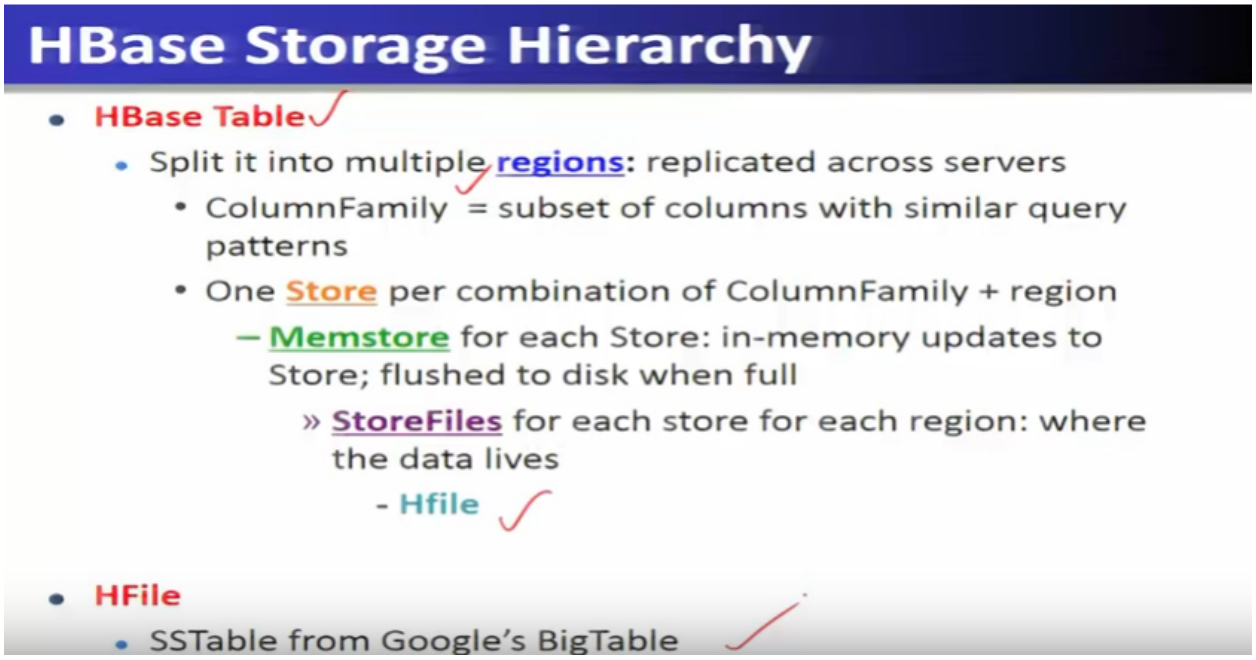
Refer Slide Time :(11: 03)

HBase Architecture



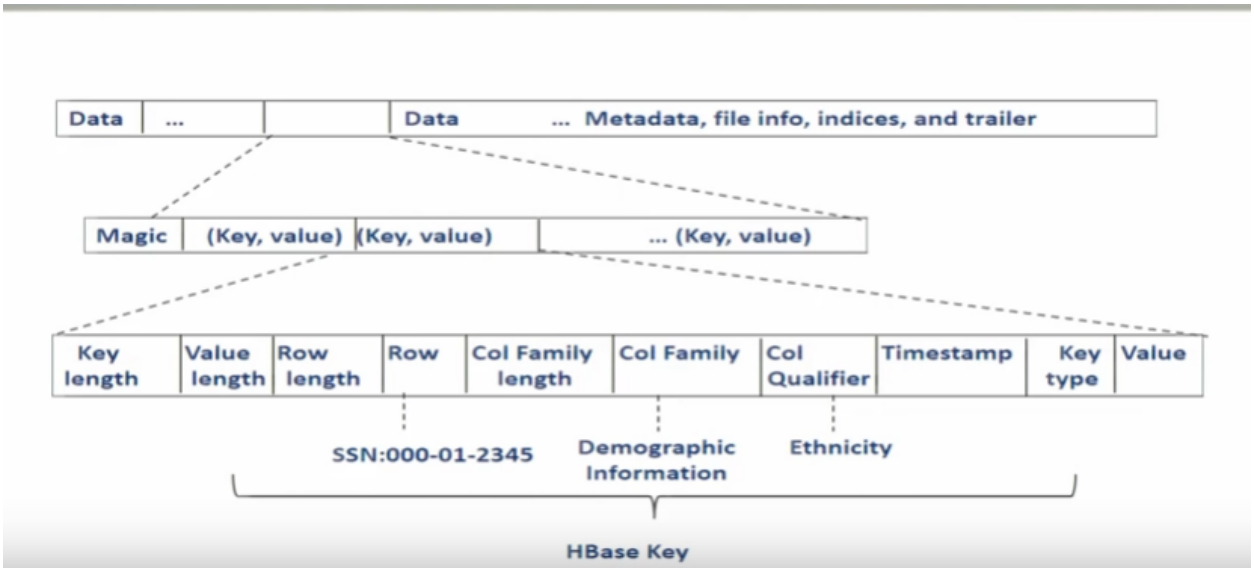
Let us see, in more detail once again the HBase architecture. So, HBase, has this one a client, client can access to the edge region servers and this edge region servers, are many each region servers, one such region server, is shown over here, which has H log and each region server, is further, is divided into different H regions, one such edge region is shown over here and edge regions will contain the store and also a mime store. So, within the store it will be having, the store files and store files will contain basic storage that is called, 'H File' and H file is stored in HDFS. Now as far as, there is one H master and H master communicates, with the zookeeper and with the edge region servers and HDFS, we will see, we have seen about, the H master and what is zookeeper? Is a small group of servers, which runs, consensus protocol like paxos and the zookeeper is, is the coordination service for HBase and assigns, the different nodes and servers, to this particular service if zookeeper is not there then HBase, will stop functioning.

Refer Slide Time :(12: 38)



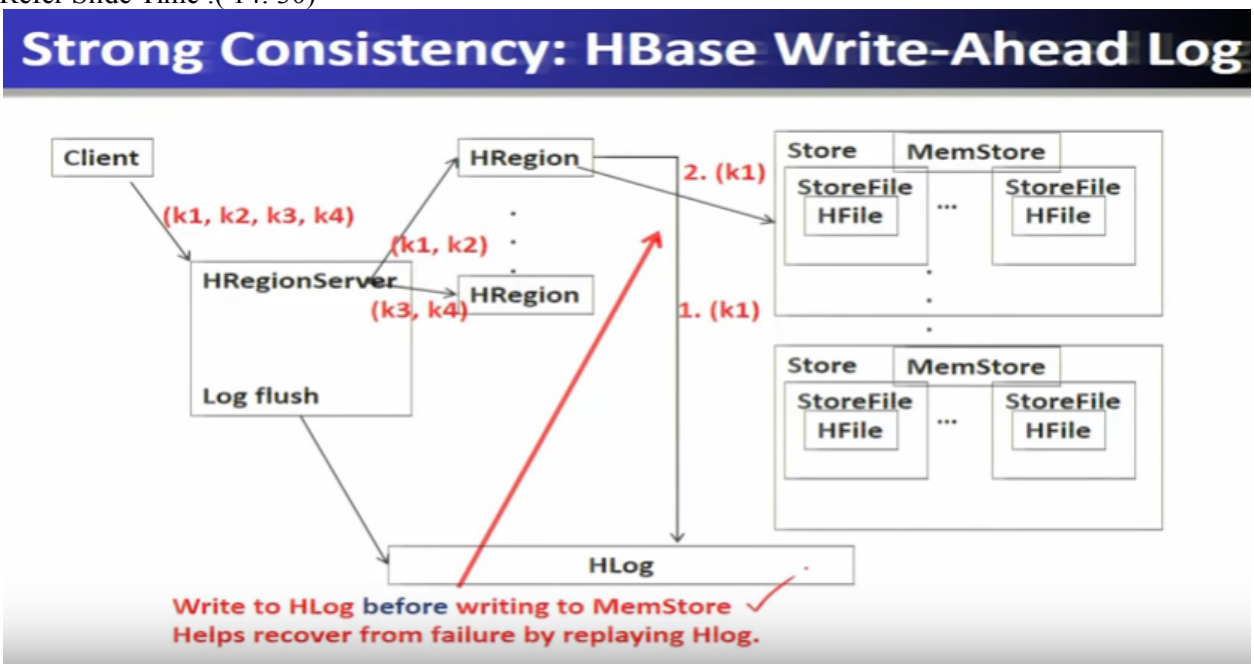
Now, let us see the HBase storage hierarchy. So, HBase has HBase table, which is split into the multiple regions, which is replicated across the servers. Now, then it will be having a column family, which is a subset of the columns, with similar query and one store, per combination of column family, plus a region, is there and also, it has, the mammy store, for each store, in memory updates to store and flush, to the disc when full, that like we have seen in the Cassandra. So, store files, for each store, for each region, where the data lives and within that, contains the basic and basic H file and as file will be stored in HDFS. So, each file is an SStables, from the Google's, Big Table.

Refer Slide Time :(13: 41)



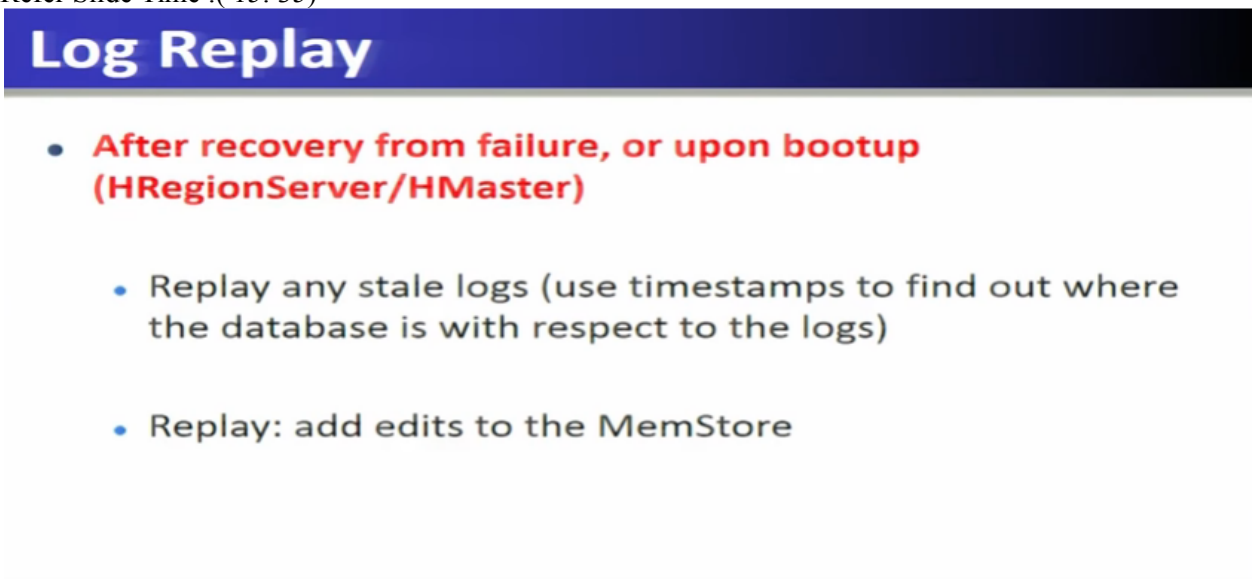
So, this is about the HBase H file. So, H file comprises, of data and Meta file, Metafile information indices and trailer, where the, the portion data, consists of the magic value key, value pairs. So, the key value pair, is having the key length, key value, key value length, row length, row and column family length, column family, column family qualifier, timestamp, key value and value. And these rows will have SSN numbers and column family will have, also the demographic information and column qualifier, will have ethnicity information and this becomes the HBS key.

Refer Slide Time :(14: 30)



Now, HBase prefers, the strong consistency or availability. So, HBase prefers, the right ahead log and whenever a client, comes with the, with the key K 1, K 2, K 3, K 4 gives to the H region server, then this let us say, K 1 and K 2, will be on a particular H region and K 3, K 4 will have another, H region and then this particular aspect, will be stored, in the store and these store will have the mammy store, store files and internally, they are stored in as file. So, right 2h log, before writing to mammy store is, there to ensure the, the fault tolerance, aspect and recovery from the failures. So, this helps recover from the failure, by replying H log.

Refer Slide Time :(15: 35)



So, log replay after the recovery from the failure or upon boot up, H region servers and H region master will do the replay. So replay any stale logs using the, timestamp to find out where the database is with respect to the log and replay will add it, to the mammy store.

Refer Slide Time :(15: 53)

Cross-Datacenter Replication

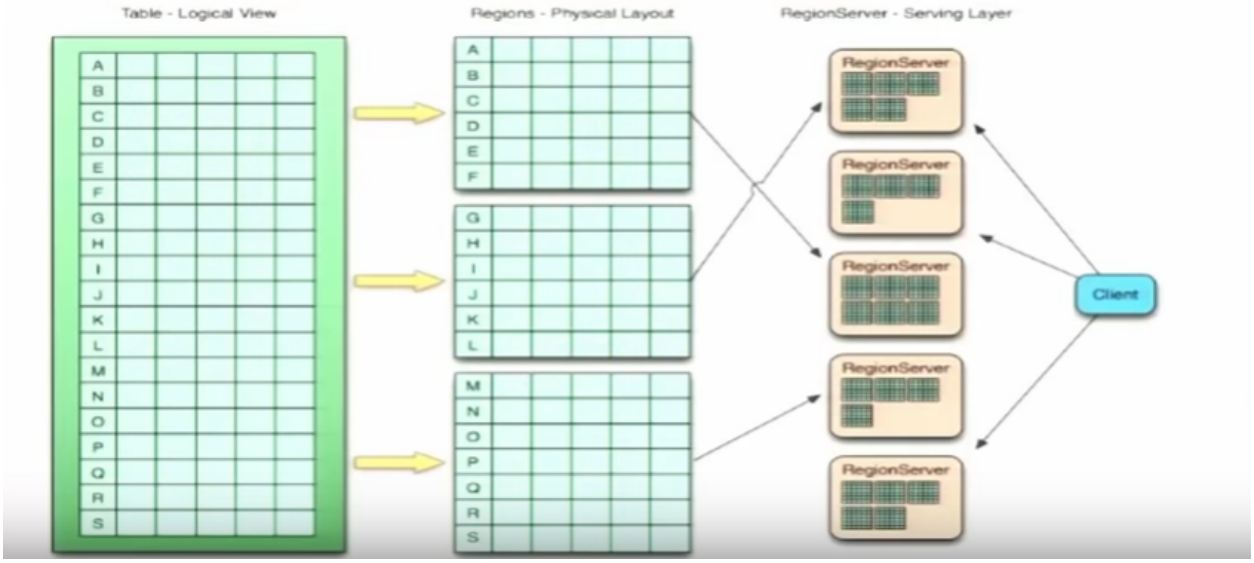
- Single "Master" cluster
- Other "Slave" clusters replicate the same tables
- Master cluster synchronously sends HLogs over to slave clusters
- Coordination among clusters is via Zookeeper
- Zookeeper can be used like a file system to store control information

1. /hbase/replication/state
2. /hbase/replication/peers/<peer cluster number>
3. /hbase/replication/rs/<hlog>

Now, cross data center replication now, there will be a single "Master" cluster others "Slave" cluster replicate the same table, master cluster synchronously sends H log over the slaves clusters, coordination among the cluster is done by a zookeeper, zookeeper can be used, like a file system to store the control information and also, this particular zookeeper, will use different paths, for invoking the state and peer cluster number and each log all this information is there.

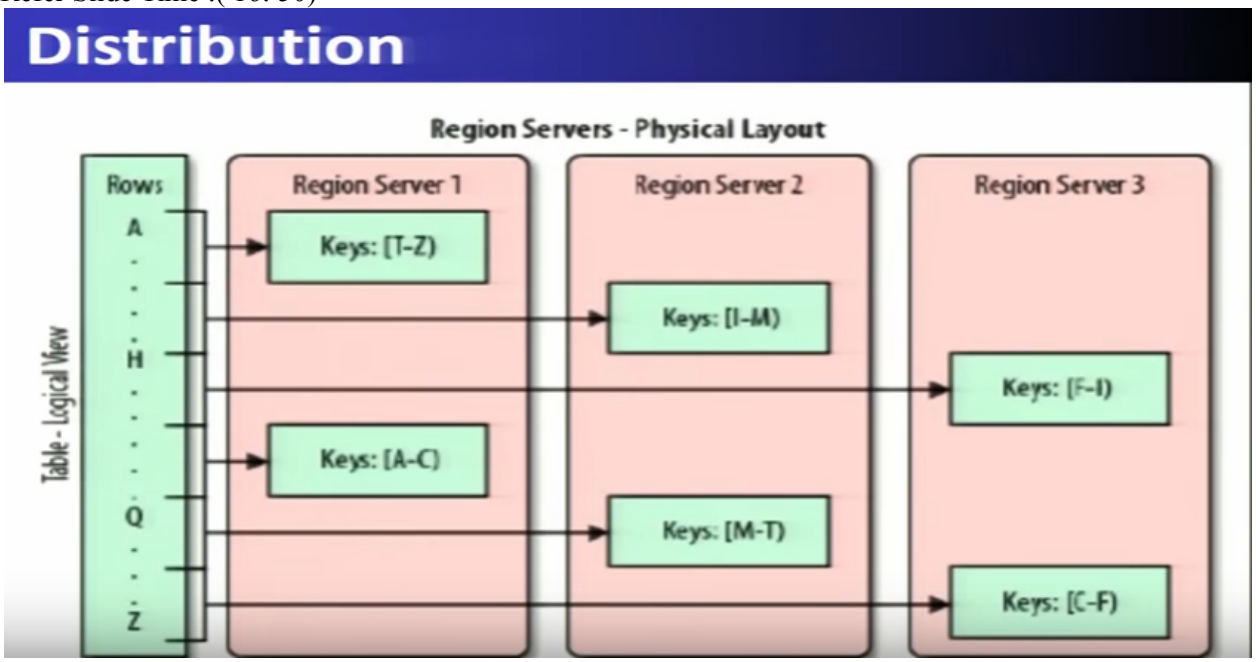
Refer Slide Time :(16: 29)

Auto Sharding



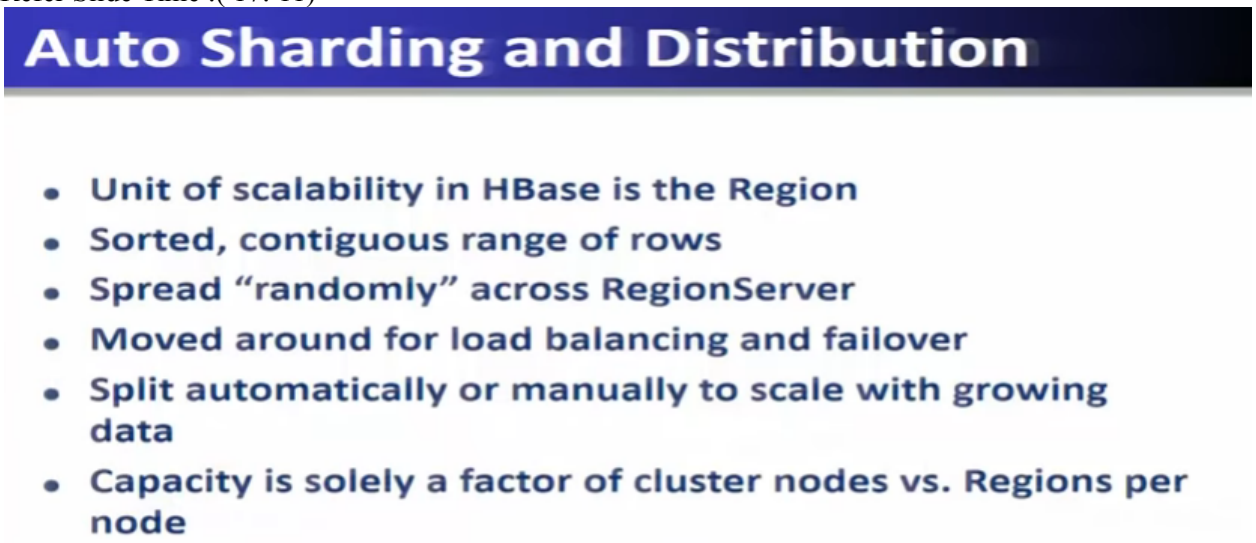
Now, let us see how, the auto sharding is done and Auto sharding means that tables, is divided into the row, range or a range keys and they are being stored in the region servers and we and these region servers are now solved, with the client.

Refer Slide Time :(16: 50)



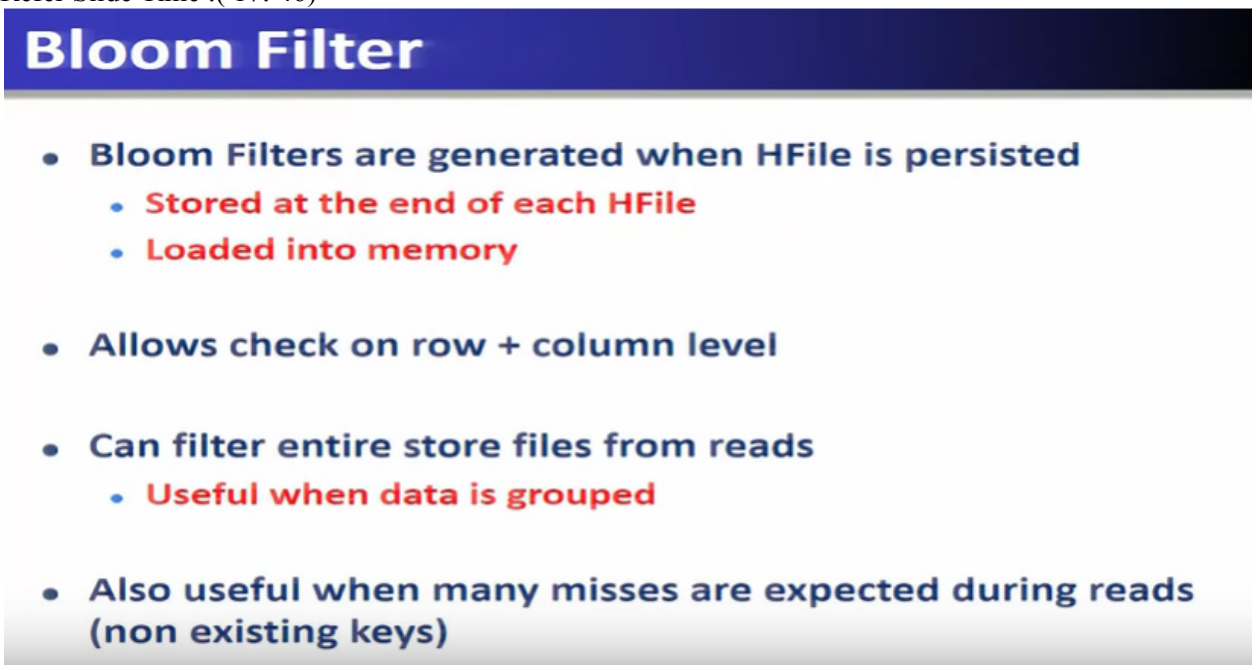
Now, similarly the, the table logical view, we can see that the rows, are now split into the, row keys, in the range keys and these range keys are given, charted into the region servers and we have shown here, that these rows from A to Z, are stored in three different region servers.

Refer Slide Time :(17: 11)



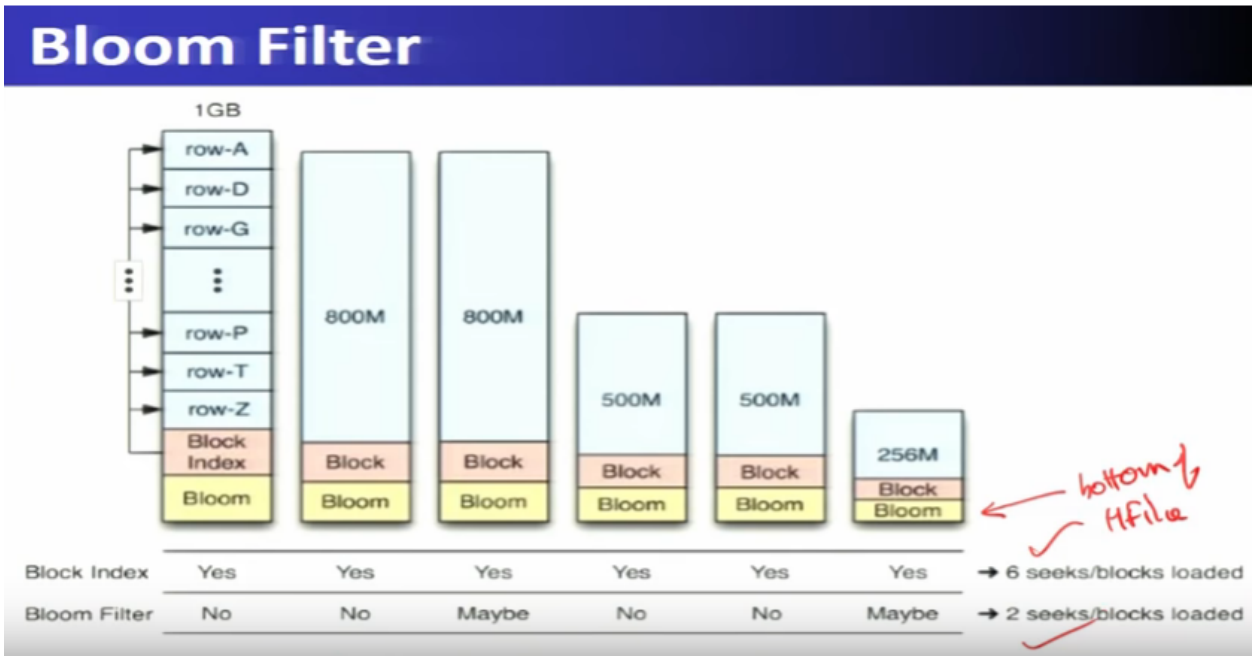
And this layout is, there and it is, automatically, done by that is called, ‘Auto Sharding’ and ‘Distribution’. So, unit of the scalability, in HBase is the region, that we have seen, which is managed by the region servers and they are Sorted and contiguous, arrange of rows, spread randomly across, the region servers, moved around, for the load balancing and failover, that we have already seen in the, previous slide. So, is split automatically or manually to scale with the growing data and capacity is only a factor of cluster nodes, versus the region per node.

Refer Slide Time :(17: 46)



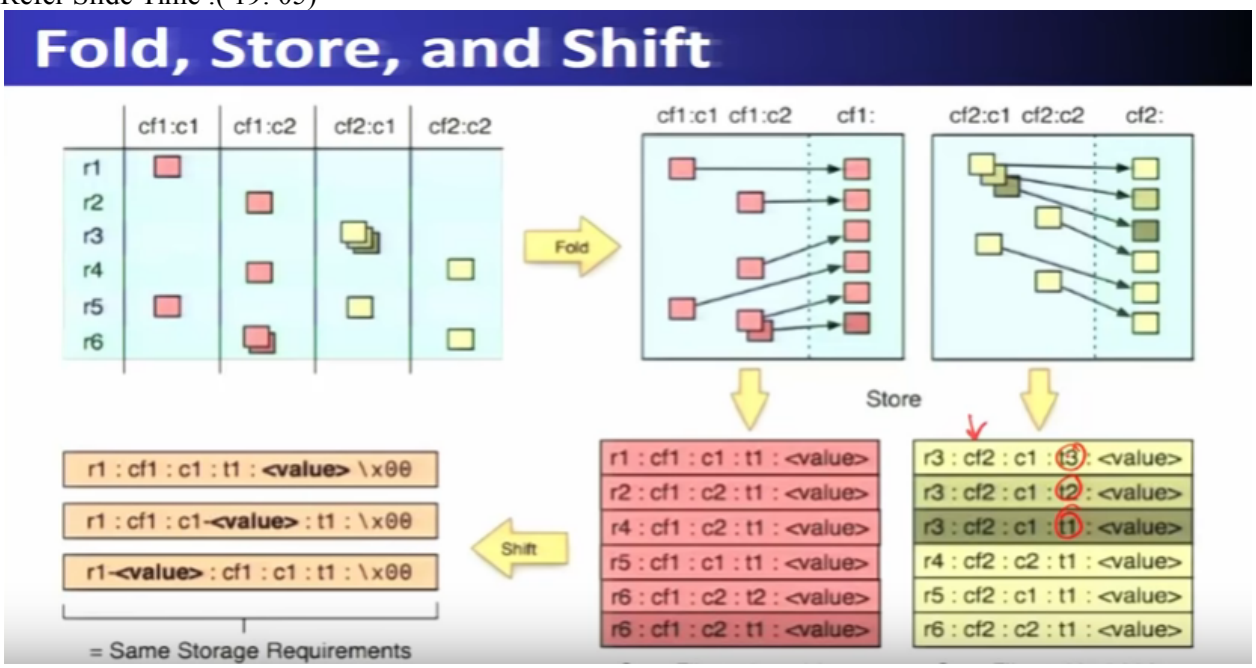
Now, there is a use of bloom filter here in HBase also, so bloom filters are generated, when HBase, when h-file is persisted, stored at the end of h-file, loaded into the memory. This bloom filter will allow, to check, on the rows and from the column level and they can filter the entire store files from the read, useful when the data, is grouped and also, useful when many misses, are expected during the reads.

Refer Slide Time :(18: 15)



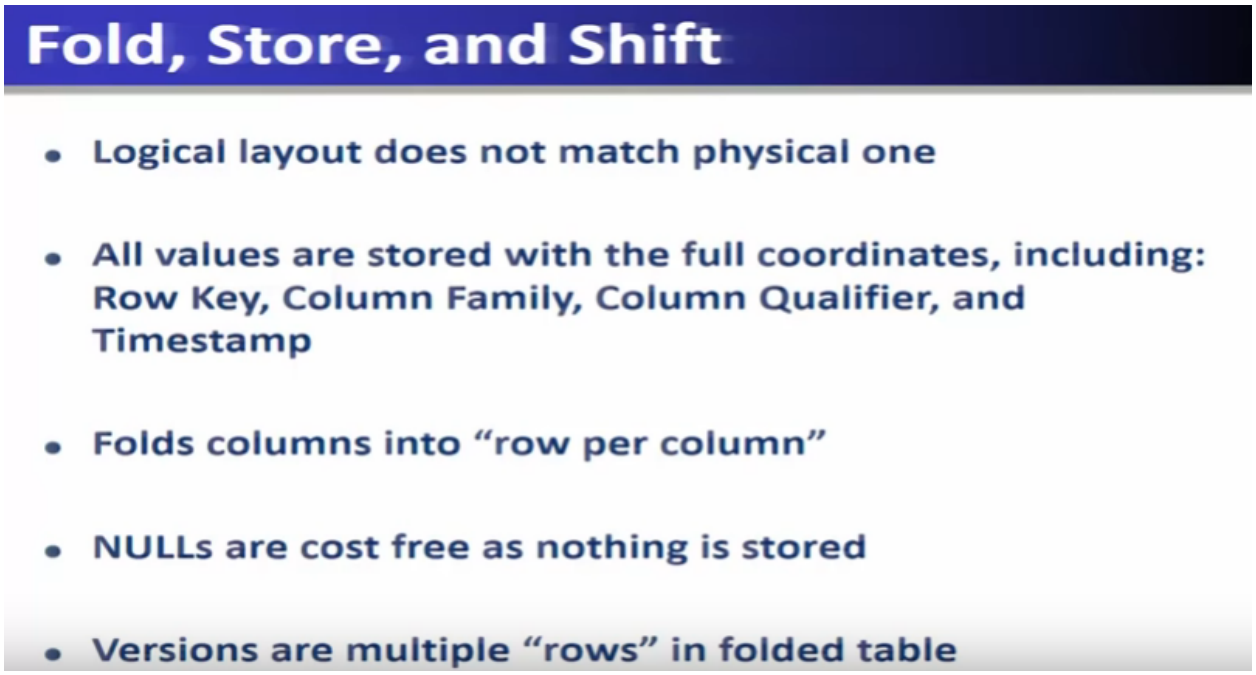
So, let us see, the positioning of bloom filter, into this h-file. So, at the bottom of h-file, this particular the bloom filter information is stored and this will, help, in identifying, which of these different blocks, are basically containing the block index or information. So, using bloom filter, you can access, at least two, at most two blocks, not the entire, set of block sequentially has to be read. So, hence the, access is reduced from six, six per block, to the two six per block, using the bloom filter and bloom filter, stone we is restored as a part of s five.

Refer Slide Time :(19: 05)



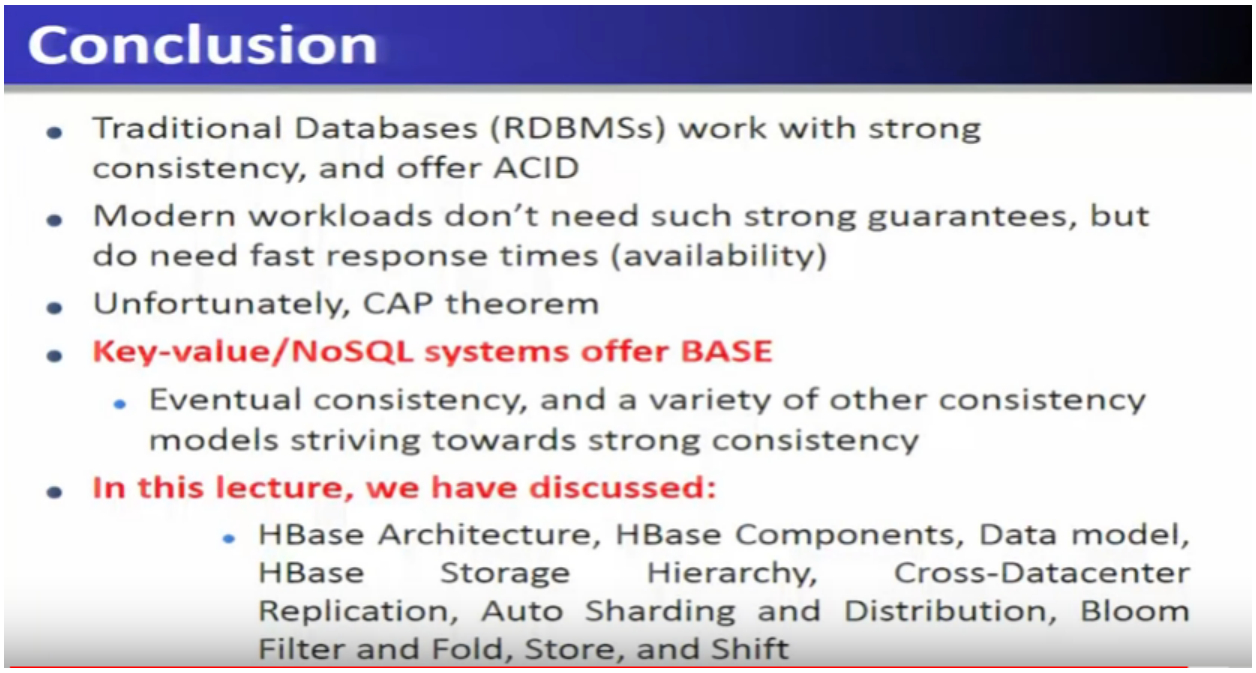
Now, was far as, fold store and shift is concerned, you can see here that, for a particular row, the same data is stored, multiple instances, because the time series data, is also stored, at different time stamp the data, is stored for a particular row key. So, that is why? Several data, on a particular row key is now appearing, they may be varying, at their timestamp T 1, T 2 and T 3 and they are a store, into the, into the five.

Refer Slide Time :(19: 44)



Logical lay out does not match the physical one and here all the values are stored with the full coordinates including the row key, column family and column qualifier and the timestamp. Folds column into the "row per column" and nulls are cost free as nothing is stored, versions are multiple rows, in a folded table.

Refer Slide Time :(20: 06)



Conclusion traditional databases RDBMS, works with strong consistency and offers, acid property and in the modern workload, work load doesn't need such strong guarantees, but do need fast response time that is the availability unfortunately, cap provides three out of two, that is here, given the partition tolerance, the HBase prefers the consistency, over the availability. So, key value pair or a new sequel system, offers the base property in this scenarios. So, Cassandra offers the eventual consistency and the other variety of consistency models, are striving towards, the strong consistency. So, in this lecture we have covered, about the HBase architecture components, data model, storage hierarchy, cross datacenter replication auto-starting, distribution and bloom filter and fold store and shift operation. Thank you.