Requirement of Database:

* Random fast access of data (indexes, partitioning, buckets)
* Data is stored in a structured manner.
* Low latency
* ACID compliance

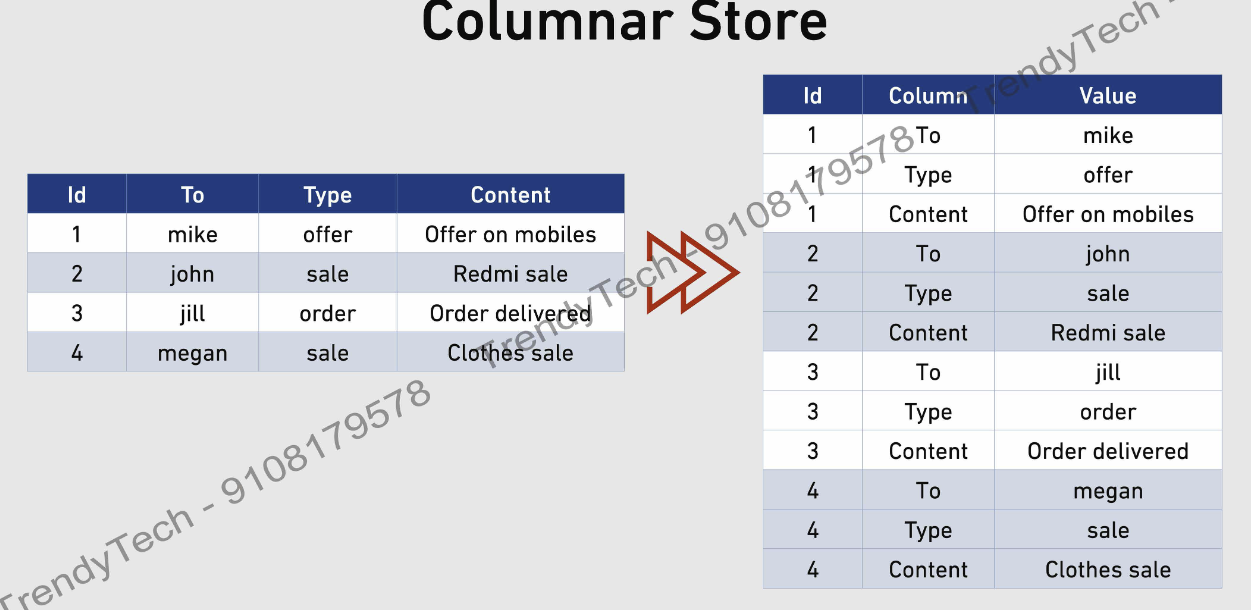
Isolation is achieved by locks

Quick search: Google published these papers :

* GFS (google file system) => HDFS
* Mapreduce => mapreduct
* Bigtable => hbase (implementation of bigtable is Hbase)

Hbase:

* Distributed system ( stored data in HDFS)
* Scalable (Directly proportional to number of nodes in cluster)
* Fault tolerant
* Low latency – Real time access using row-based indices called row keys.



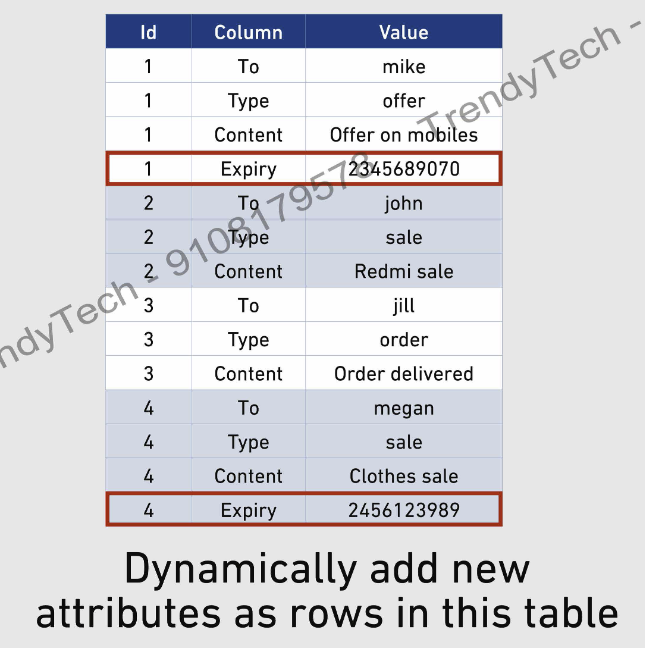
RDBMS

HBase

|  |  |
| --- | --- |
| **HBase** | **Traditional Databases** |
| Create id, key, value | Stores data in rows & columns |
|  |  |
| Adding new columns is easy | Adding row is easy |
|  |  |
| Will manage space properly | If there are empty fields in rows still they will contain full row space |
|  |  |
| Only CURD allowed | Join, group by also allowed |
|  |  |
|  |  |

Table

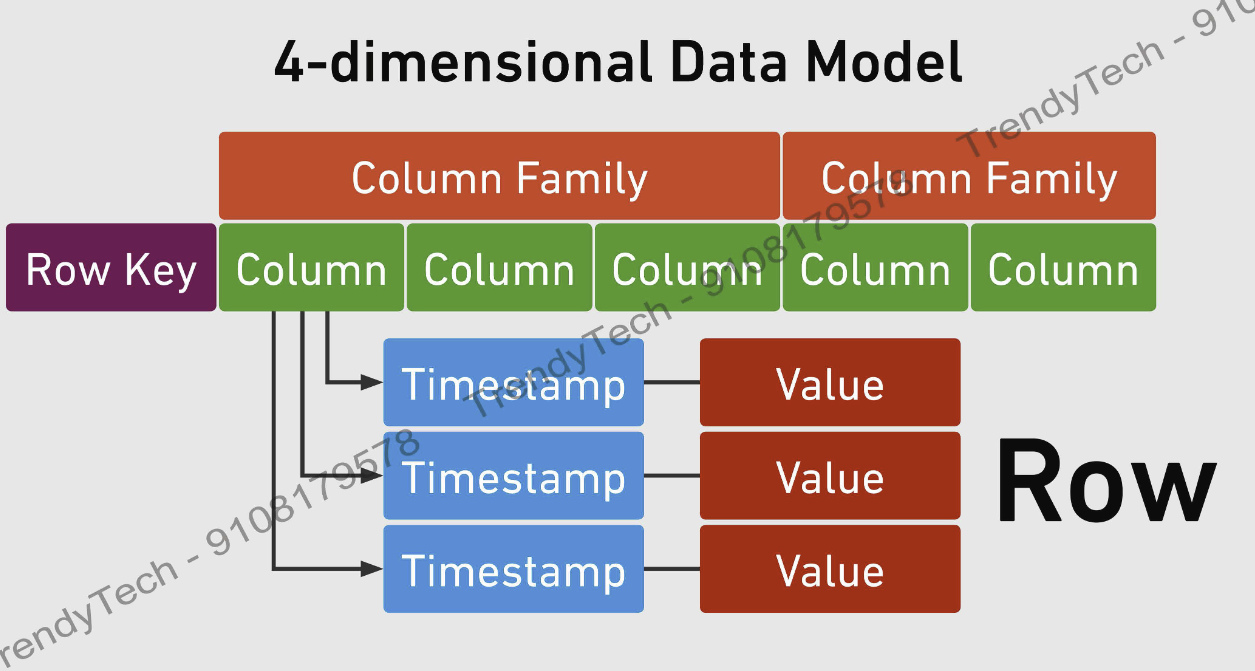
Description automatically generated

****

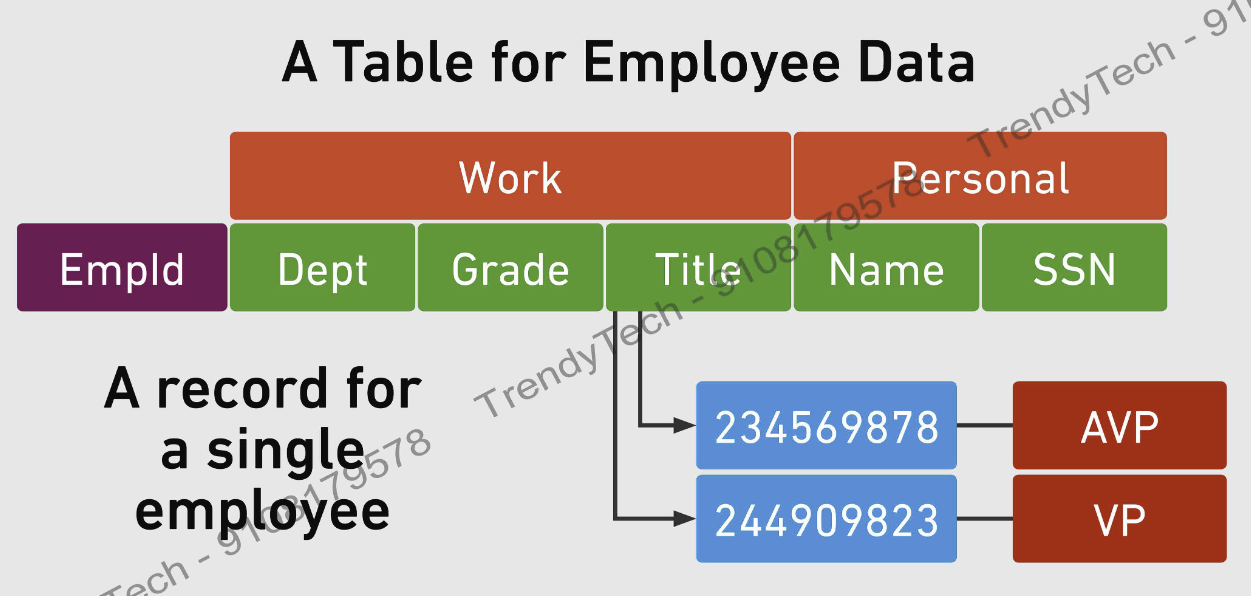
**HBase has 4d data model**

**4-D HBase Model**

* Uniquely identify a row.
* Stores in ascending order

****

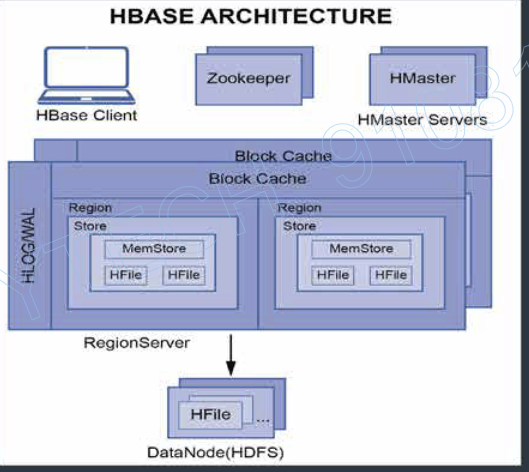
* Each column family is stored in separate file.

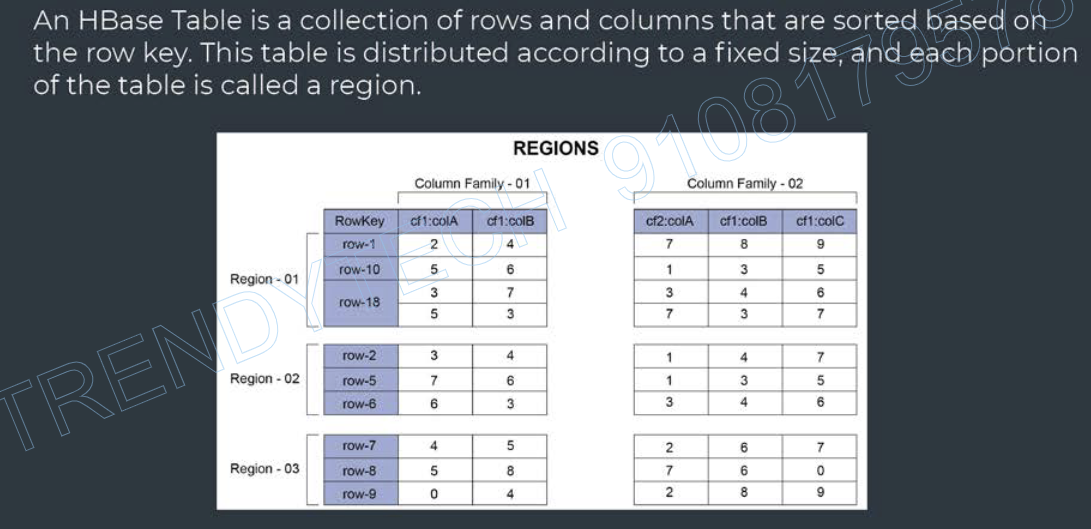


Timestamps

* Here this employee was an AVP and then VP.
* As per timestamps HBase stores different versions. By default, latest version is stored.
* In HBase all the data stored in byte arrays, there is no concept like string, int, float etc.
* New column can be added on the fly

**HBASE Architecture**

****



**Hbase Table**: A hbase table is stored based row key. This table is distributed to a fixed size. Each partition of the table called a **region.**

**Memstore:** Whenever you write a new record. Every insert is appended in memory in the memstore. When size of memstore grows to a threshold size then the content of the memstore will be flushed to the disk and a new file will be created called as Hfile. This Hfile is stored in HDFS. **For each region there is a memstore for every column family.**

**Region server:** Each node should run one region server. A client directly with region server to perform read and write operations.

Each region server contains:

* WAL (Write ahead log) or Hlog
* Block cache

**WAL (hlog):** wal keeps a trace of data store in memstore. In case server fails with the help of WAL we can retrieve the lost data. (bez memstore stores data in in-memory storage.So, in case of any server failure we will loose the data.) region server stores the WAL in HDFS.

**Block cache :** Whenever we read the data, data is cached in-memory. So, for the next read, it might get data directly from cache.

**HMaster**

* Hamster is the master server and region servers are the slave.
* Hbase cluster may have one or more master node, but only one Hmaster is active at the given time. The active node is responsible for:
  + Hmaster performs DDL operations, Such as creating or deleting a table.
  + Hmaster assign regions to region servers on start up.
* **Load balancing**: Whenever there is high load on any region server, Hmaster unloads the busy server and reassigns this load to less occupied servers.
* **Recovery:** Hmaster also handles region server failures by reassigning the regions or load of the failed regions to another regions.

**Zookeeper**

* It is a distributed open-source coordinating service for distributed applications.
* Zookeeper maintains the live server state. Every server sends the heartbeat to zookeeper.
* It stores the location of meta table.

Diagram

Description automatically generated

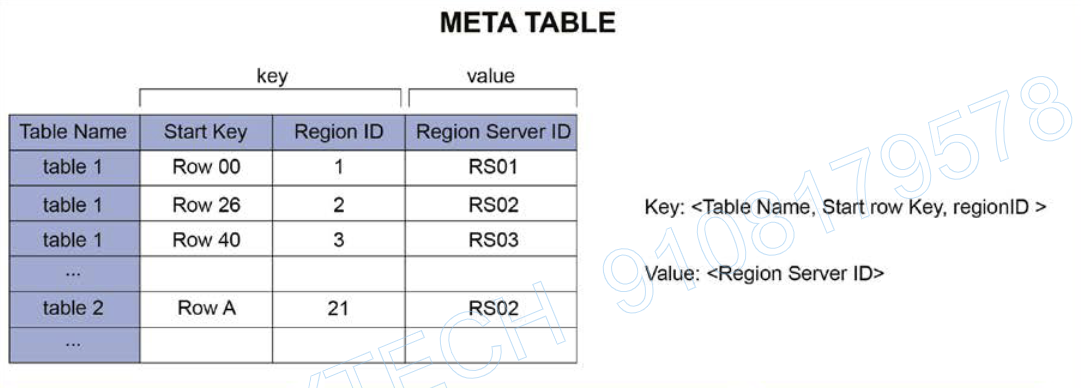
**HBASE stores 2 types of files in Data node of HDFS :**

1. **WAL (write ahead log)**
2. **H file**
   * It stores hbase table data in key-value pairs.
   * Hfiles are immutable.
   * Hfile are large in size and depend on the depend on the memstore flush size before compaction.
   * Hfile further stores data in set of blocks. This helps in only reading the block which is required not the complete file. Default block size is 64kb.

**Data block index :** In hfile it is use to locate the data block of interest. It contains the key range stored in data block.

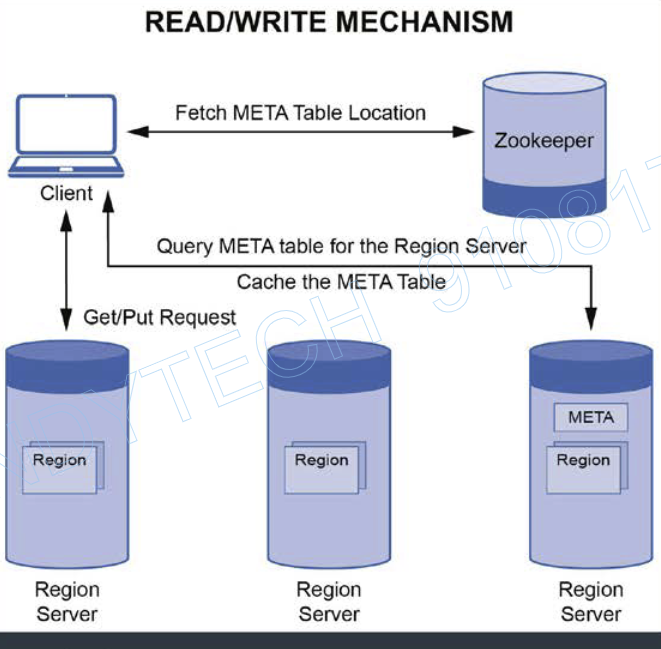
**Meta Table**

* A meta table is a data structure that stores location of regions along with region servers.
* If helps user to identify region server and its corresponding regions, where specific key-value is stored.
* Meta table location is stored in zookeeper, and it is stored in one of the region servers.



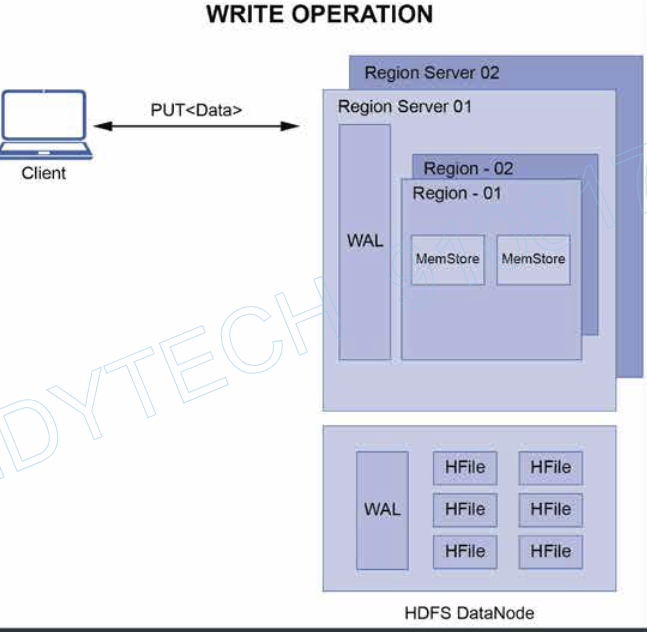
**HBase READ/Write Operations**

1. Client contacts the zookeeper to fetch the location of Meta table (if the client does not have latest version of Meta table)
2. Client queries the Meta table to find the location of region servers.
3. Client cashes the meta table location and identifies region server locations for future interactions.



**Hbase Write Operations**

1. Data first need to written to the WAL.
2. Once data is written to the WAL, it is placed in the memstore of the region.
3. When memstore reaches it’s threshold, it’s content are flushed to HDFS (data node) to form a new Hfile.
4. Finally an acknowledgement is sent to the client.



**HBASE Read operations**

1. Server first check the block chache of the region server that stores recently accessed data.
2. In case data is not available in the block cache, it checks do we have required data in memstore.
3. Then it will check with meta table if where are required data is stored. And will reach to the region server’s hfiles. Then it scans the ‘data block index’ of the hfile and reads required blocks.

**Compaction**

When memstore flushed data in HDFS, it creates multiple small Hfile. Which are bad for read performance. So, to resolve this we do compaction which means “combining small files into large file”

There are 2 types of Compaction are there:

1. Minor : Smaller hfiles rewrites into large file.
2. Major : All hfile in the store are picked and rewritten into large hfile.

**Hbase Data Deletion**

When we delete values in hbase. They don’t get immediately deleted instead they are marked with tombstone marker. (As hfiles are immutable. So, value can’t be deleted in hfiles). When major compaction happens then only these values get deleted.

**In Case of Server failure :**

1. If hamster server get fails, then zookeeper gives the master responsibility to one of the inactive hamster cluster.
2. If a region server fails, zookeeper notifies hamster about it and it reassign task to some other region server.