**Deep dive - HBase**

### Introduction

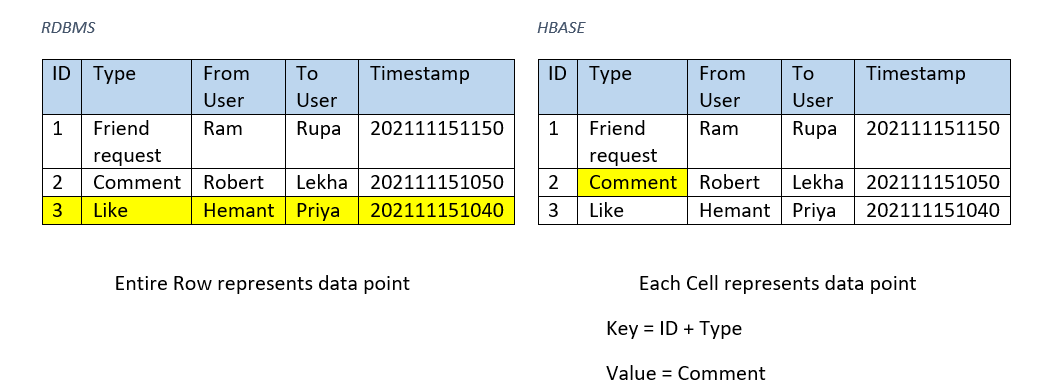
Pinterest, Goibibo and Facebook Messenger etc use HBase for data storage.

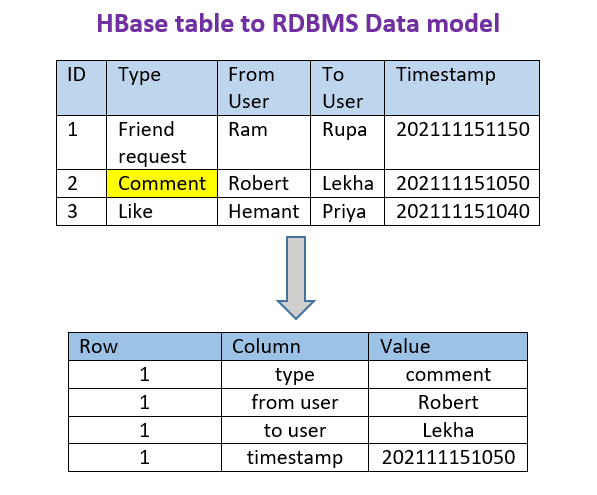
### Need for HBase

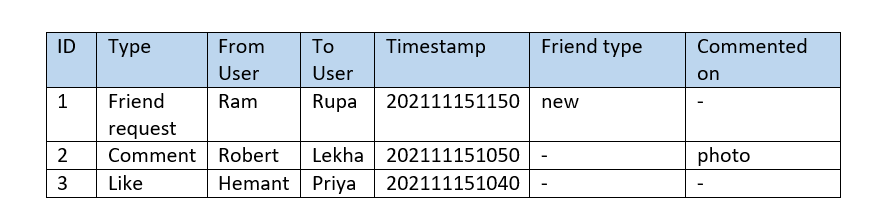
Drawbacks of HDFS:

* No Schema
* No Random Access
* High Latency
* No ACID support

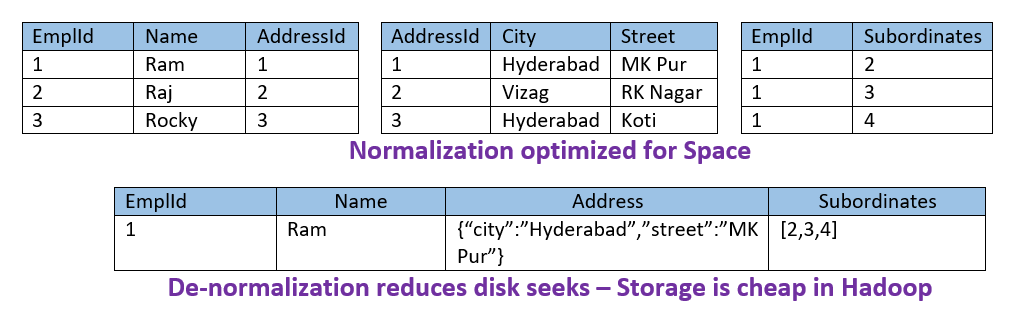
### Data Model

* Column Oriented



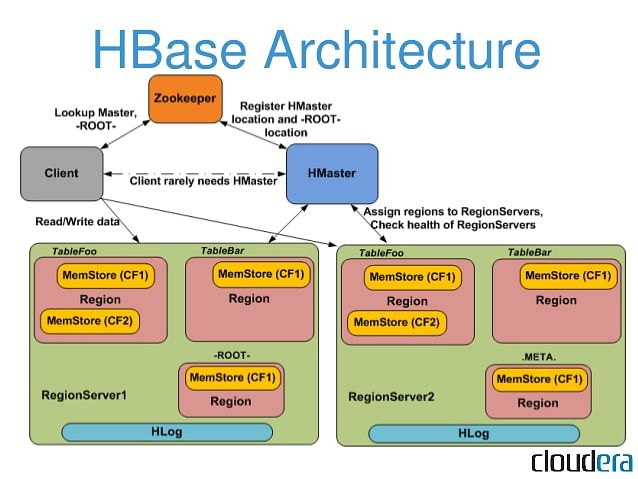


* Denormalized



* Only CRUD operations
* ACID at row level

### Architecture

****

1. HMaster

* Monitoring
* Failover controlling
* DDL operations and alters

1. Region Server

* Block cache
* MemStore
* Write Ahead Log
* HFile

1. Zookeeper

* Client communication
* Track Server failovers
* Config info

### Terminology

* HBase Tables
* HBase Row
* RowKey
* Columns
* Column Family

### CRUD operations and other Commands using CLI

docker exec -i -t h\_hbase-phoenix bash

start-hbase.sh

hbase shell

#### Data Definition Language

* create - Creates a table. create 'personal','personal\_data'
* list - Lists all the tables in HBase. list
* disable - Disables a table. disable 'personal'
* is\_disabled - Verifies whether a table is disabled. is\_disabled 'personal'
* enable - Enables a table. enable 'personal'
* is\_enabled - Verifies whether a table is enabled. is\_enabled 'personal'
* describe - Provides the description of a table. describe 'personal'
* alter - Alters a table. alter 'personal', {NAME=> 'column\_name'}
* exists - Verifies whether a table exists.
* drop - Drops a table from HBase. drop 'personal'
* disable\_all - Disables the tables matching the regex give in the command. disable\_all 'p.\*'
* drop\_all - Drops the tables matching the regex given in the command. drop\_all 'p.\*'

#### Data Manipulation Language

put - Puts a cell value at a specified column in a specified row in a particular table.

put 'personal','2','personal\_data:name','Ram'

put 'personal','2','personal\_data:city','Bengaluru'

* put 'personal','2','personal\_data:age' ,'25'
* get - Fetches the contents of row or a cell. get 'personal','2', {COLUMN => ['personal\_data:name']}
* delete - Deletes a cell value in a table. delete 'personal','2', 'personal\_data:name'
* deleteall - Deletes all the cells in a given row. deleteall 'personal',2

scan - Scans and returns the table data. scan 'personal'

* scan 'personal',{LIMIT=>0}
* count - Counts and returns the number of rows in a table. count 'personal'
* truncate - Disables, drops, and recreates a specified table. truncate 'personal'

### HBase Spark Connectors

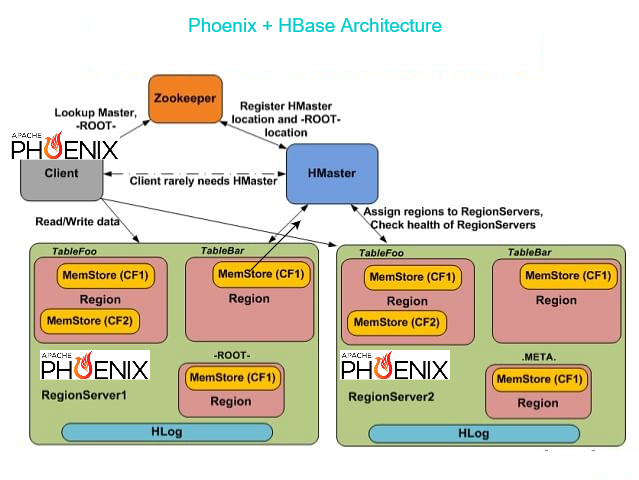
* Hive
* Phoenix
* HBase Spark connector
* SHC
* Happybase

Know Phoenix

What is Apache Phoenix?

* Query engine
* Metadata repository
* JDBC driver

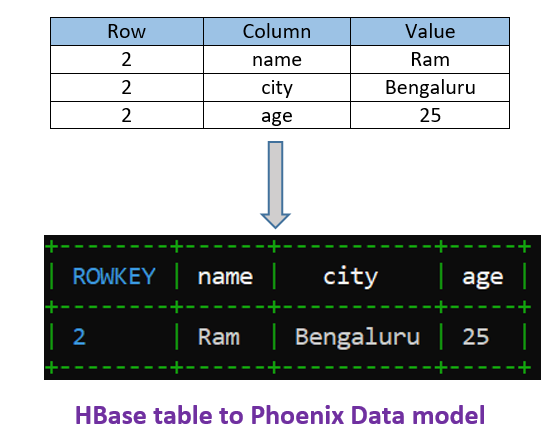
### Architecture



Performance

* SQL Queries to native HBase scans
* determines optimum start and stops
* orchestrates parallel execution
* computation to data by
  + pushing the predicates
  + aggregations queries through server-side hooks
  + secondary indexes
  + stats gathering
  + skip scan filter
  + optional salting

### Data Model - Example

****

#### Mapping Existing HBase Tables

* Phoenix supports read only access to existing HBase tables
  + Create a Phoenix table using CREATE TABLE
  + Or create a view using CREATE VIEW
  + Use appropriate quoting for mixed case HBase table and native column names

Queries

Enter Phoenix/HBase Container docker exec -i -t h\_hbase-phoenix bash

Check whether HBase daemons are up jps

* start-hbase.sh

HBase to Phoenix

CREATE view "personal"(

ROWKEY VARCHAR PRIMARY KEY,

"personal\_data"."name" VARCHAR,

"personal\_data"."city" VARCHAR,

"personal\_data"."age" VARCHAR

);

* Create sql file

vi us\_population.sql

CREATE TABLE IF NOT EXISTS us\_population (

state CHAR(2) NOT NULL,

city VARCHAR NOT NULL,

population BIGINT

CONSTRAINT my\_pk PRIMARY KEY (state, city));

* Create data file

vi us\_population.csv

NY,New York,8143197

CA,Los Angeles,3844829

IL,Chicago,2842518

TX,Houston,2016582

PA,Philadelphia,1463281

AZ,Phoenix,1461575

TX,San Antonio,1256509

CA,San Diego,1255540

TX,Dallas,1213825

CA,San Jose,912332

* Create table and insert data psql.py zookeeper:2181 us\_population.sql us\_population.csv
* Enter Phoenix shell and Query sqlline.py zookeeper:2181

SELECT state as "State",count(city) as "City Count",sum(population) as "Population Sum"

FROM us\_population

GROUP BY state

ORDER BY sum(population) DESC;

ctrl+D

* Check whether created table exists in Hbase hbase shell

list

* Please read this document for more details about Phoenix <https://phoenix.apache.org/presentations/OC-HUG-2014-10-4x3.pdf>