

HCIA Big Data HBase — Distributed Database

Eng. Hagar Muhammed

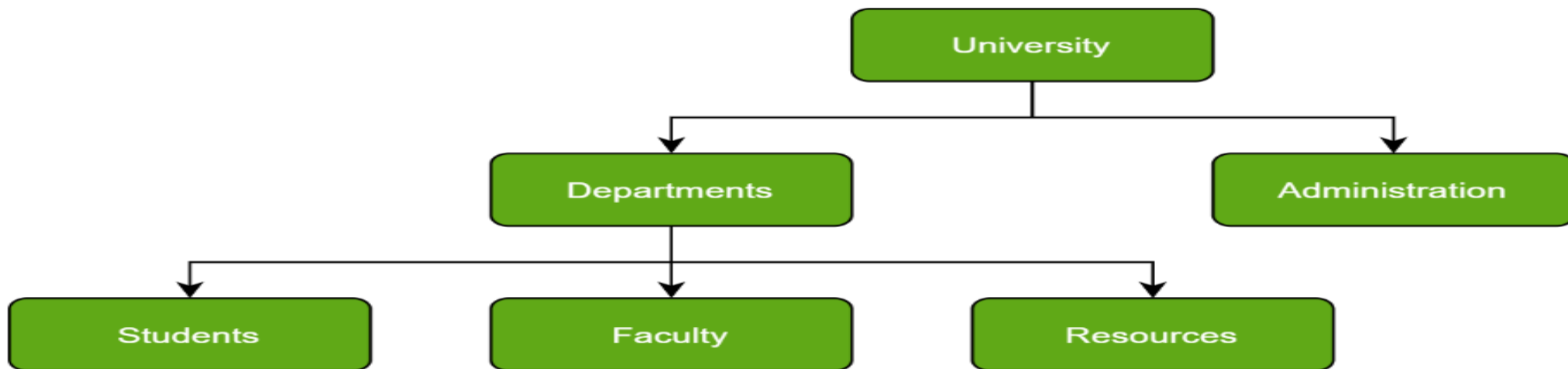
Lecture 5

Contents

- 1. Types of Databases**
2. HBase Overview
3. Storage Mechanism in HBase
4. Features of HBase
5. HBase Performance Tuning
6. Common Shell Commands of HBase

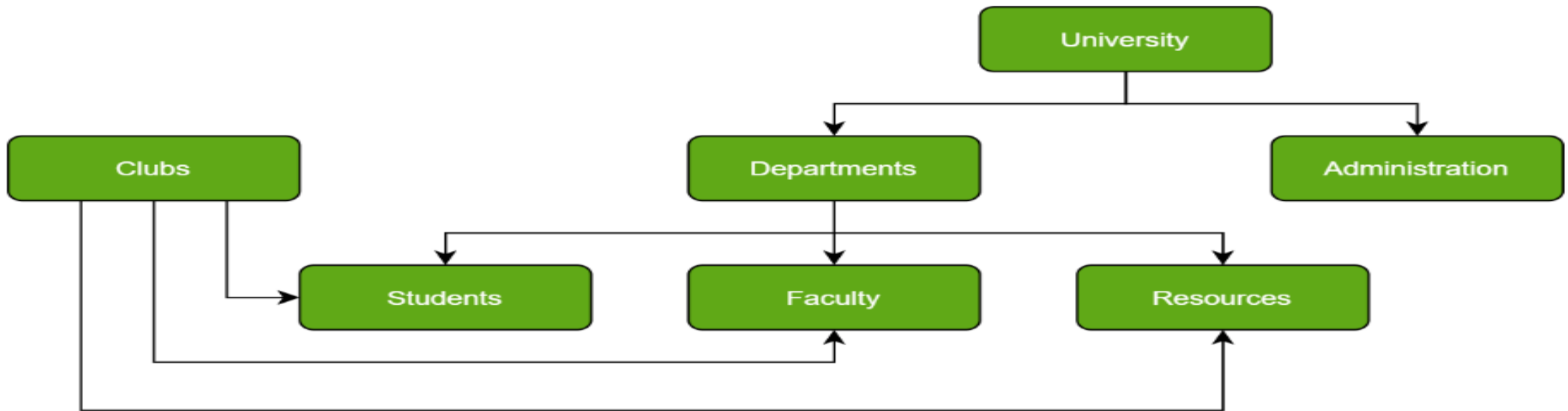
Hierarchical Databases

Just as in any hierarchy, this database follows the progression of data being categorized in ranks or levels, wherein data is categorized based on a common point of linkage. As a result, two entities of data will be lower in rank and the commonality would assume a higher rank. Refer to the diagram below:



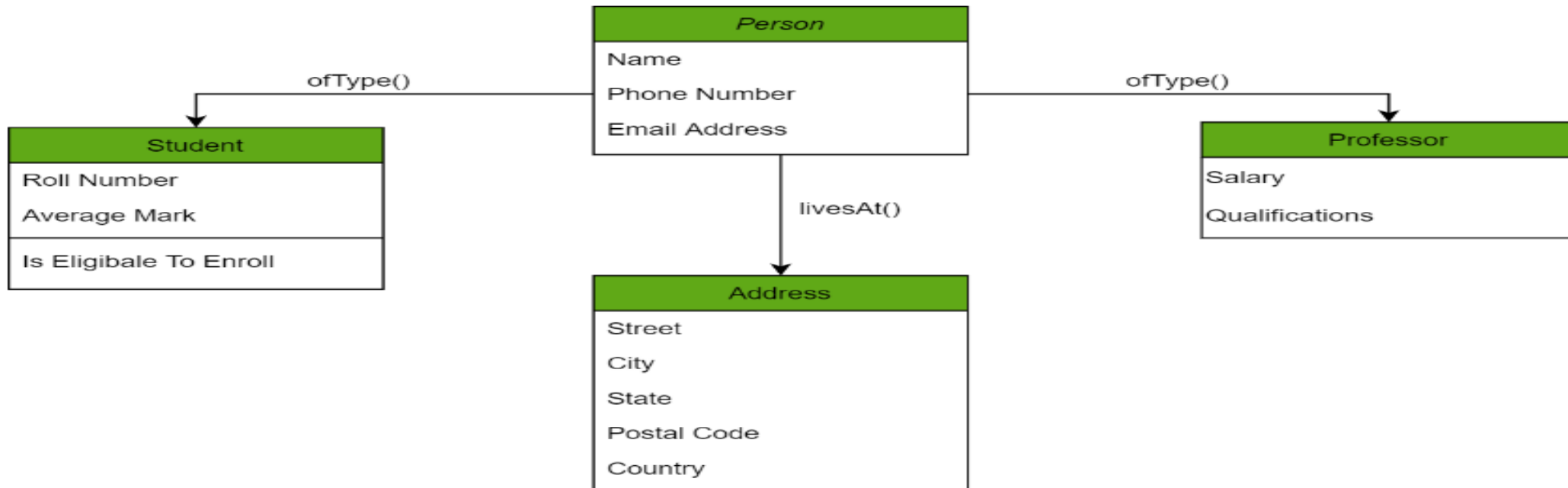
Network Databases :

A network database is a hierarchical database, but with a major tweak. The child records are given the freedom to associate with multiple parent records. As a result, a network or net of database files linked with multiple threads is observed. Notice how the Student, Faculty, and Resources elements each have two-parent records, which are Departments and Clubs.



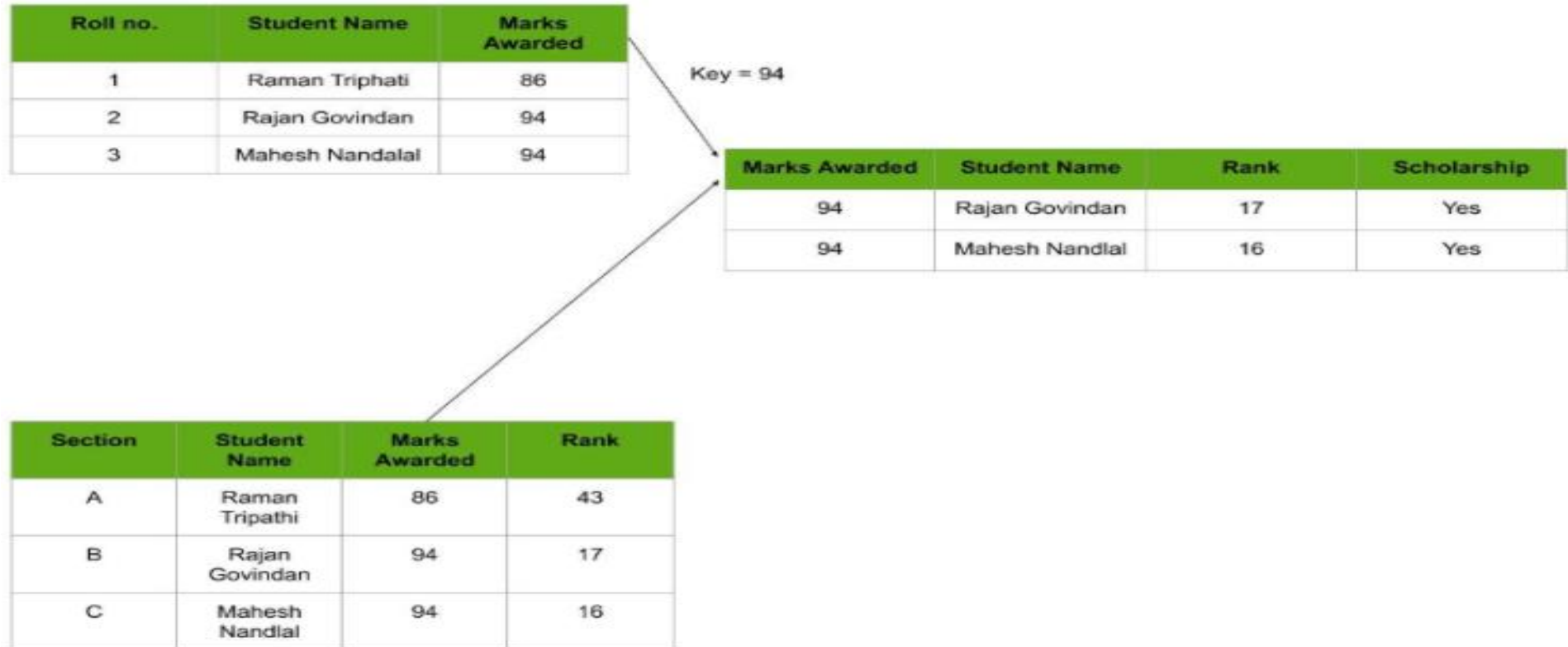
Object-Oriented Databases :

Those familiar with the Object-Oriented Programming Paradigm would be able to relate to this model of databases easily. Information stored in a database is capable of being represented as an object which response as an instance of the database model. Therefore, the object can be referenced and called without any difficulty. As a result, the workload on the database is substantially reduced.



Relational Database:

Considered the most mature of all databases, these databases lead in the production line along with their management systems. In this database, every piece of information has a relationship with every other piece of information. This is on account of every data value in the database having a unique identity in the form of a record.



Column Based Databases :

A **column-oriented DBMS** or **columnar DBMS** is a database management systems(DBMS) that stores data tables by column rather than by row. Benefits include more efficient access to data when only querying a subset of columns (by eliminating the need to read columns that are not relevant), and more options for data compression. However, they are typically less efficient for inserting new data.

Practical use of a column store versus a row store differs little in the relational Database world. Both columnar and row databases can use traditional database query languages like SQL to load data and perform queries

Contents

1. Types of Databases
- 2. HBase Overview**
3. Storage Mechanism in HBase
4. Features of HBase
5. HBase Performance Tuning
6. Common Shell Commands of HBase

HBase Overview

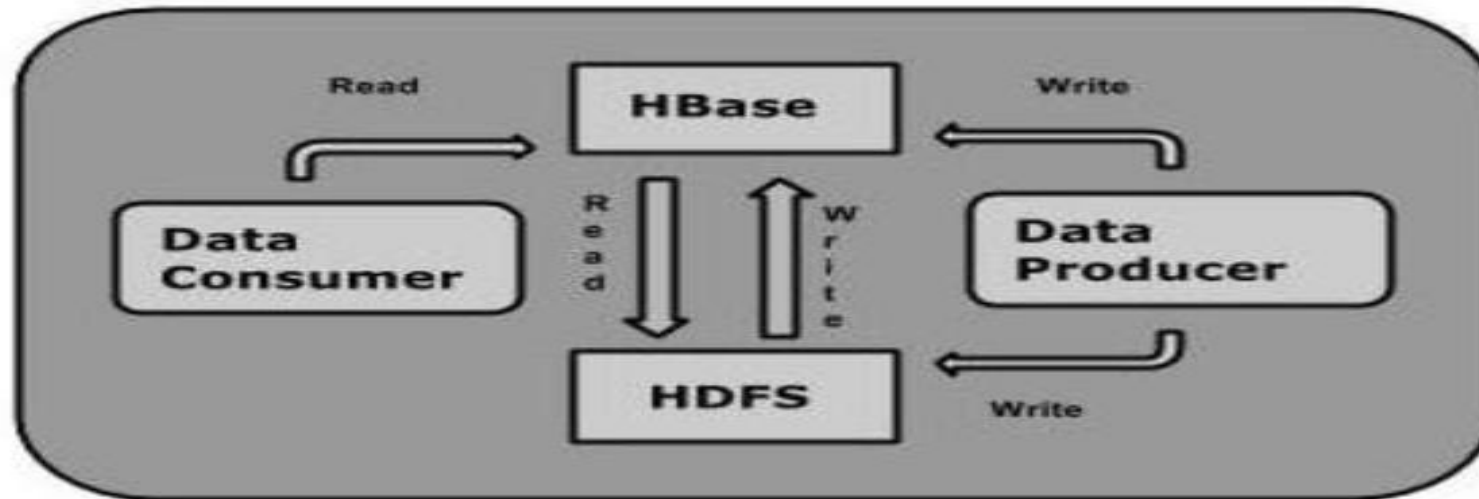
HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model 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 real-time read/write access to data in the Hadoop File System.

Hbase Overview

The user can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



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).

Contents

1. Types of Databases
2. HBase Overview
- 3. Storage Mechanism in HBase**
4. Features of HBase
5. HBase Performance Tuning
6. Common Shell Commands of HBase

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 on the disk. Each cell value of the table has a timestamp.

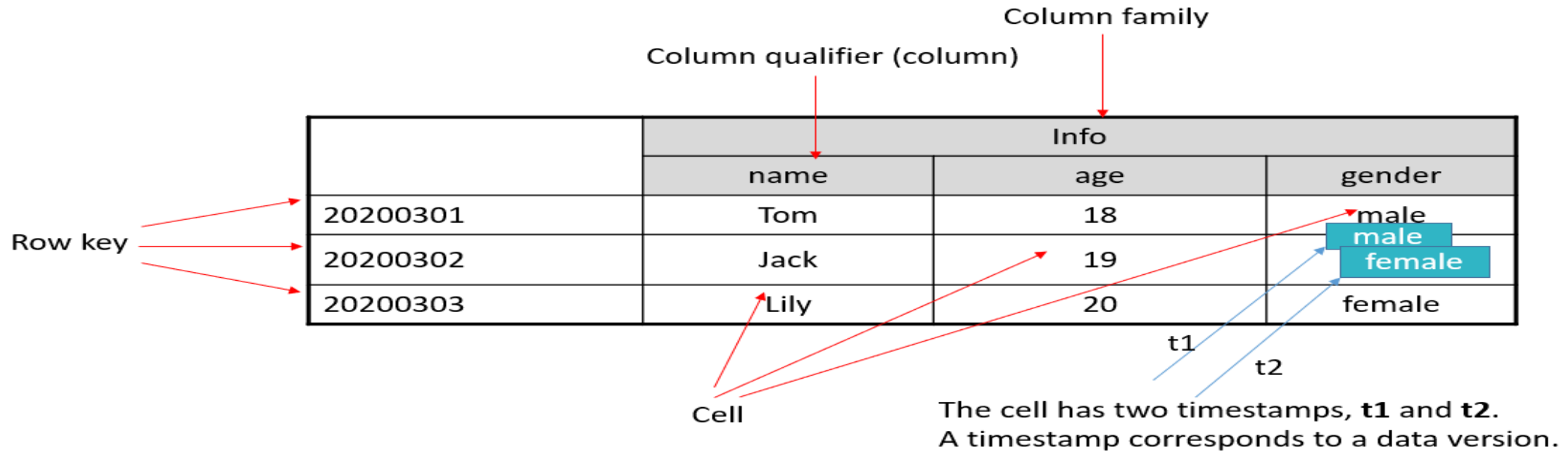
Storage Mechanism in HBase

Given below is an example schema of table in HBase.

Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												

- 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.

HBase Table Structure



Column Oriented and Row Oriented

Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

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.

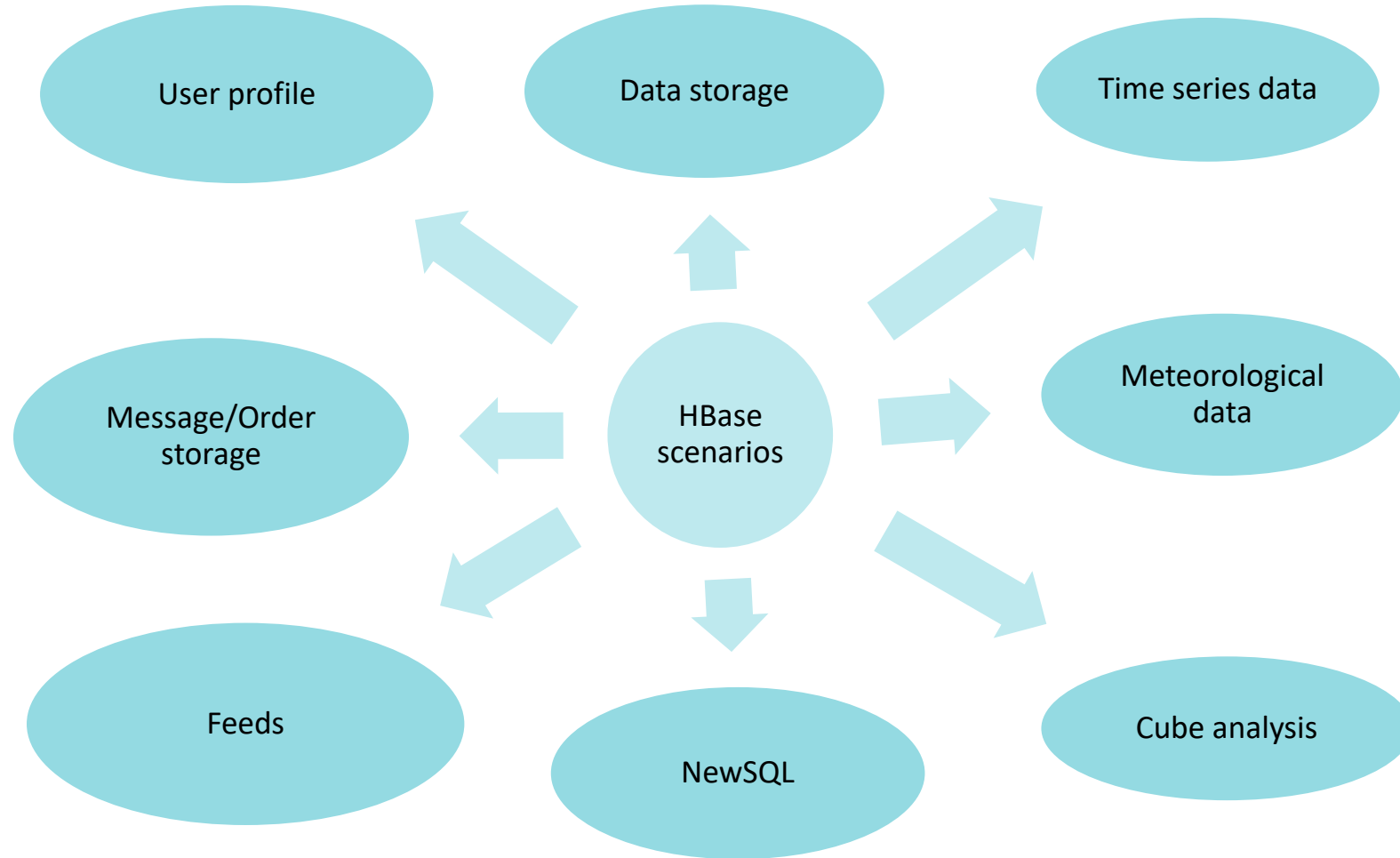
Contents

1. Types of Databases
2. HBase Overview
3. Storage Mechanism in HBase
- 4. Features of HBase**
5. HBase Performance Tuning
6. Common Shell Commands of HBase

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.

Use Cases of HBase



Where to use HBase

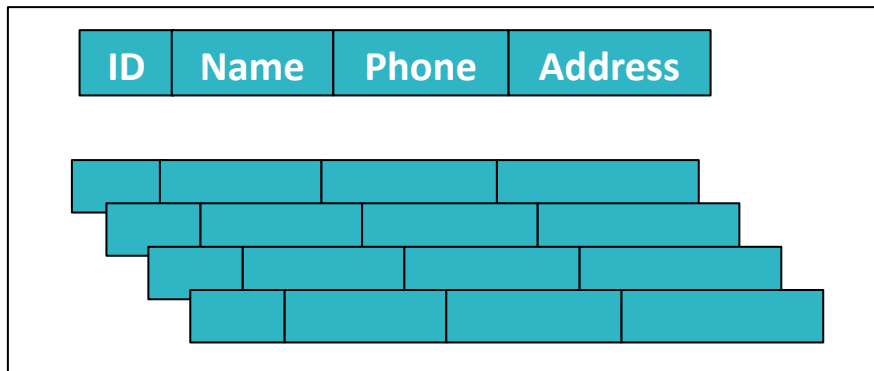
- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top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

HBase and RDBMS

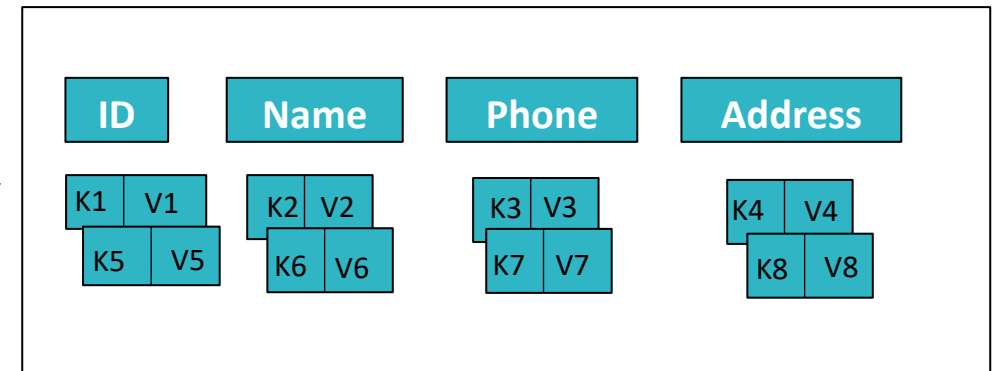
HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard 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.

- Row-oriented storage refers to data stored by rows in an underlying file system. Generally, a fixed amount of space is allocated to each row.
 - Advantages: Data can be added, modified, or read by row.
 - Disadvantage: Some unnecessary data is obtained when a column is queried.
- Column-oriented storage refers to data stored by columns in an underlying file system.
 - Advantage: Data can be read or calculated by column.
 - Disadvantage: When a row is read, multiple I/O operations may be required.

Row-oriented storage



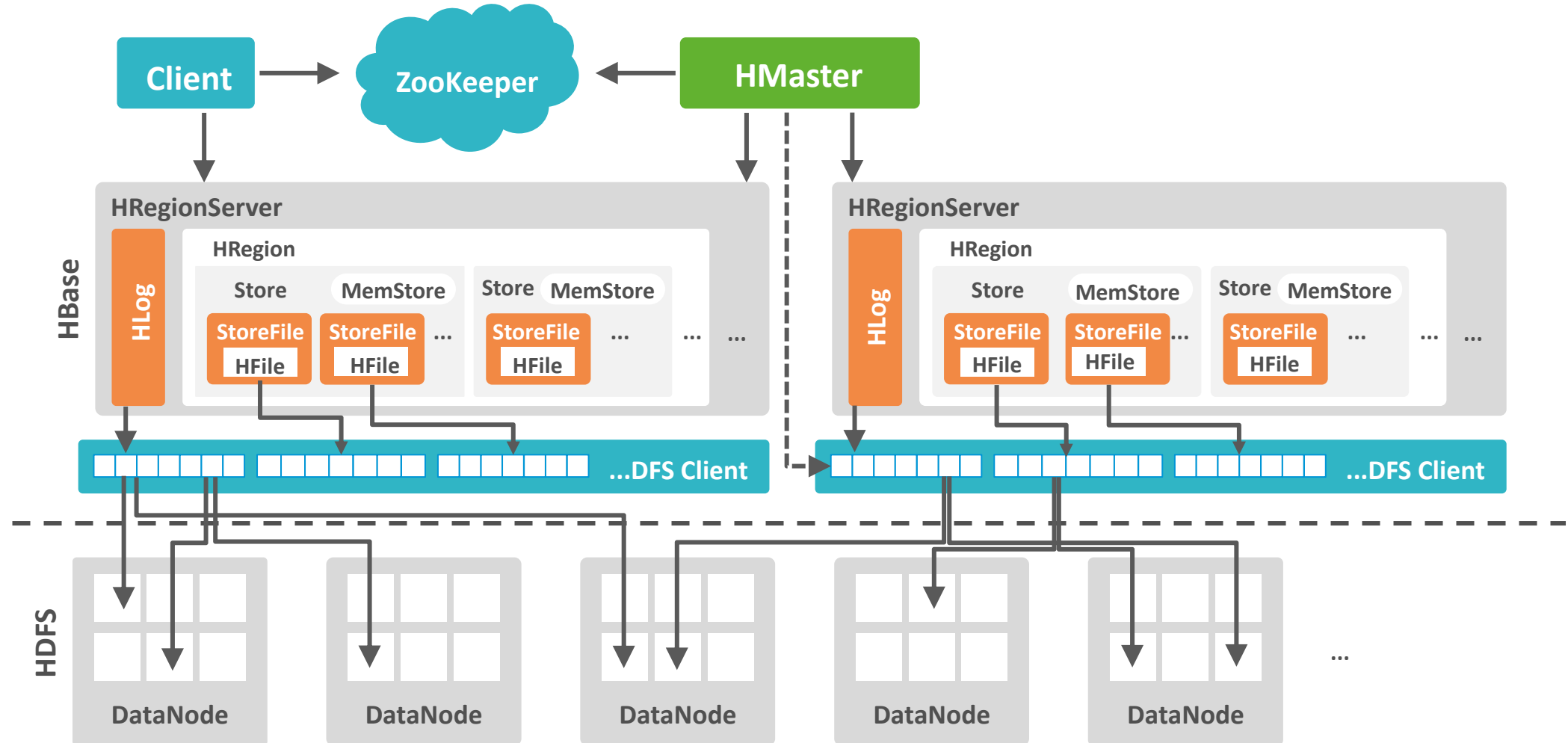
Column-oriented storage



Contents

1. Types of Databases
2. HBase Overview
3. Storage Mechanism in HBase
4. Features of HBase
- 5. HBase Architecture**
6. HBase Performance Tuning
7. Hbase shell commands

HBase Architecture



HMaster

Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.

- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

RegionServer

- Communicate with the client and handle data-related operations.
- Handle read and write requests for all the regions under it.
- Decide the size of the region

Data Read and Write Process

- When you write data, it is allocated to the corresponding HRegionServer for execution.
- Your data is first written to HLog, then to MemStore, and finally to disks to generate a StoreFile.
- The **commit()** invocation returns the data to the client only after the operation is written to HLog.
- When you read data, the HRegionServer first accesses the MemStore cache. If the MemStore cache cannot be found, the HRegionServer searches StoreFile on the disk.

Cache Refreshing

- The system periodically writes the content in the MemStore cache to the StoreFile file on the disk, clears the cache, and writes a tag in the HLog.
- A new StoreFile file is generated each time data is written. Therefore, each Store contains multiple StoreFile files.

Cache Refreshing

- Each HRegionServer has its own HLog file. Each time the HRegionServer is started, the HLog file is checked to determine whether a new write operation is performed after the latest cache refresh operation. If an update is detected, the HLog file is written to MemStore and then StoreFile to provide services for users.

Zookeeper

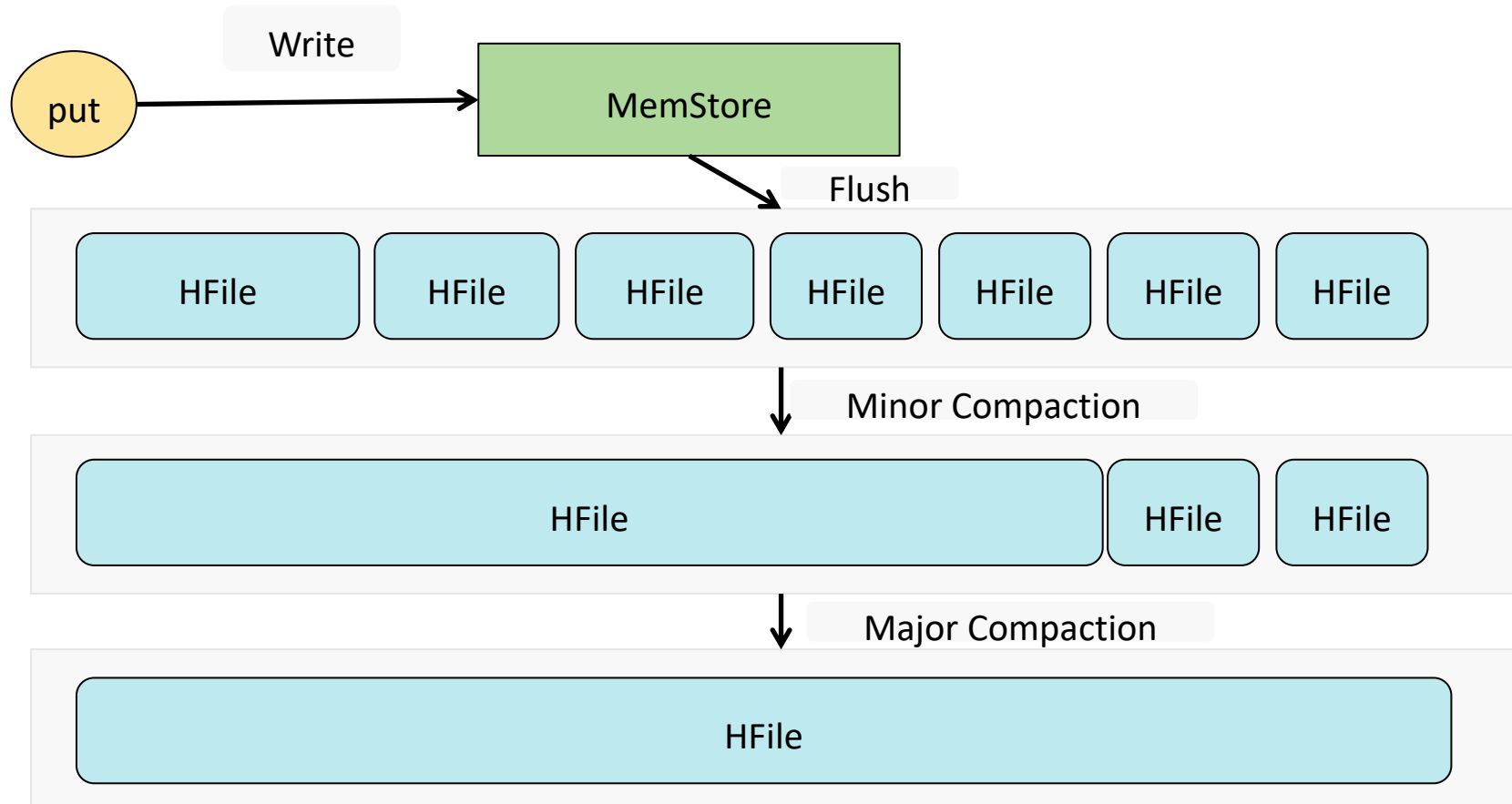
- Zookeeper is an open-source project .
- Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.

Contents

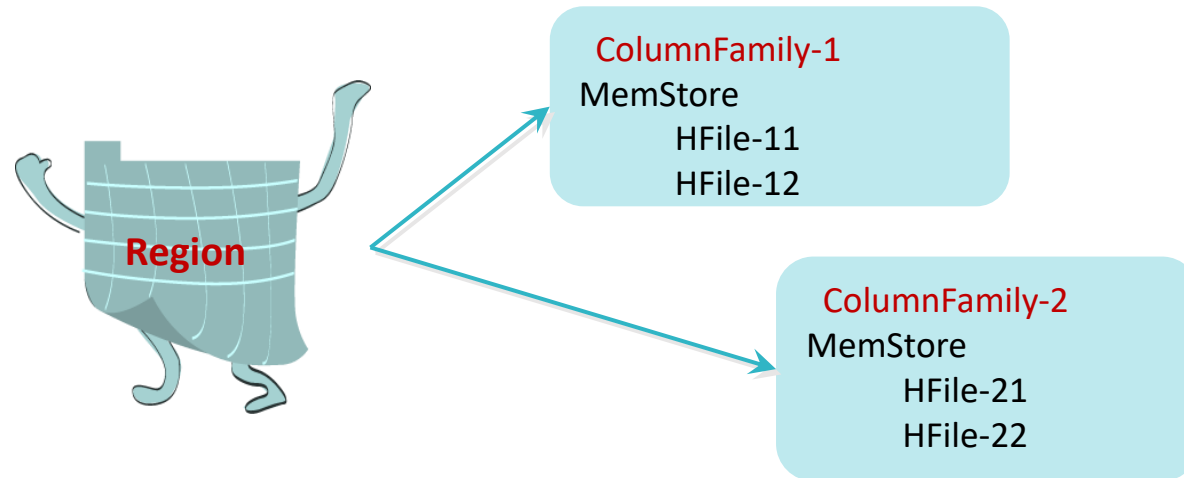
1. Types of Databases
2. HBase Overview
3. Storage Mechanism in HBase
4. Features of HBase
5. HBase Architecture
- 6. HBase Performance Tuning**
7. Hbase shell commands

- Read latency lengthens as the number of HFiles increases.
- Compaction reduces the number of small files (HFiles) in the same column family of the same region to improve read performance.
- Compaction is classified into minor compaction and major compaction.
 - Minor compaction: small-scale compaction. There are limits on the minimum and maximum number of files. Typically, small files in a continuous time range are merged.
 - Major compaction: all HFile files under the ColumnFamily of the region.
 - Minor compaction complies with a certain algorithm when selecting files.

Compaction



- In the OpenScanner process, two different scanners are created to read HFile and MemStore data.
 - The scanner corresponding to HFile is StoreFileScanner.
 - The scanner corresponding to MemStore is MemStoreScanner.



- BloomFilter optimizes random read scenarios, that is, the **Get** scenario. It can be used to quickly determine whether a user data record exists in a large dataset (most data in the dataset cannot be loaded to the memory).
- BloomFilter can misjudge whether a data record exists. However, the judgment result of "The subscriber data XXXX does not exist." is reliable.
- HBase BloomFilter data is stored in HFiles.

Row Key

- Row keys are stored in alphabetical order. Therefore, when designing row keys, you need to fully utilize the sorting feature to store frequently read data together and store recently accessed data together.
- For example, if data recently written to the HBase table is most likely to be accessed, you can use the timestamp as a part of the row key. Because the data is sorted in alphabetical order, you can use **Long.MAX_VALUE - timestamp** as the row key.

Contents

1. Types of Databases
2. HBase Overview
3. Storage Mechanism in HBase
4. Features of HBase
5. HBase Architecture
6. HBase Performance Tuning
- 7. Hbase shell commands**

- **create**: creates a table.
- **list**: lists all tables in HBase.
- **put**: adds data to a specified cell in a table, row, or column.
- **scan**: browses information about a table.
- **get**: obtains the value of a cell based on the table name, row, column, timestamp, time range, and version number.
- **enable/disable**: enables or disables a table.
- **drop**: deletes a table.

Summary

HBase is a distributed, scalable, and column-oriented database system that is part of the larger Apache Hadoop ecosystem. Designed for managing and storing massive amounts of data.

Key features of HBase include its horizontal scalability, allowing it to handle datasets by distributing data across multiple servers. HBase also offers impressive speed and performance, making it suitable for real-time or near-real-time applications. Its flexibility enables easy modifications to the schema and data, catering to evolving requirements.

Thank You
For Your Attention