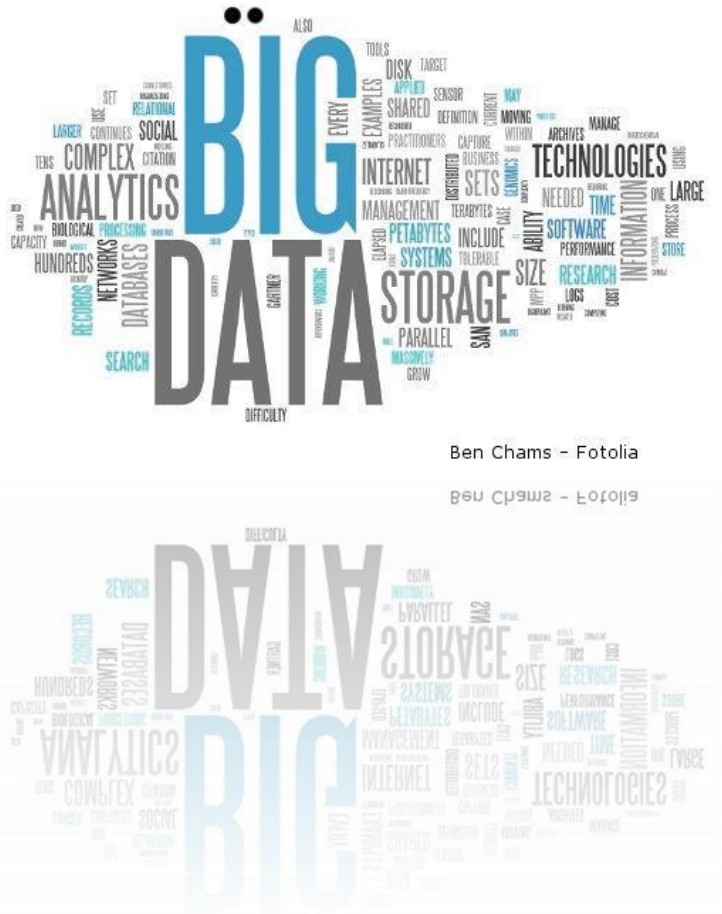
Programa de formación **MLDS**

UNIVERSIDAD NACIONAL DE COLOMBIA



Módulo BIG DATA HBase

Por
Ing. Elizabeth León Guzmán, Ph.D.

Agenda

- Preliminares
- Modelo de Datos
- Operaciones Principales
- Arquitectura
- Características
- Aplicaciones en la Industria

Preliminares

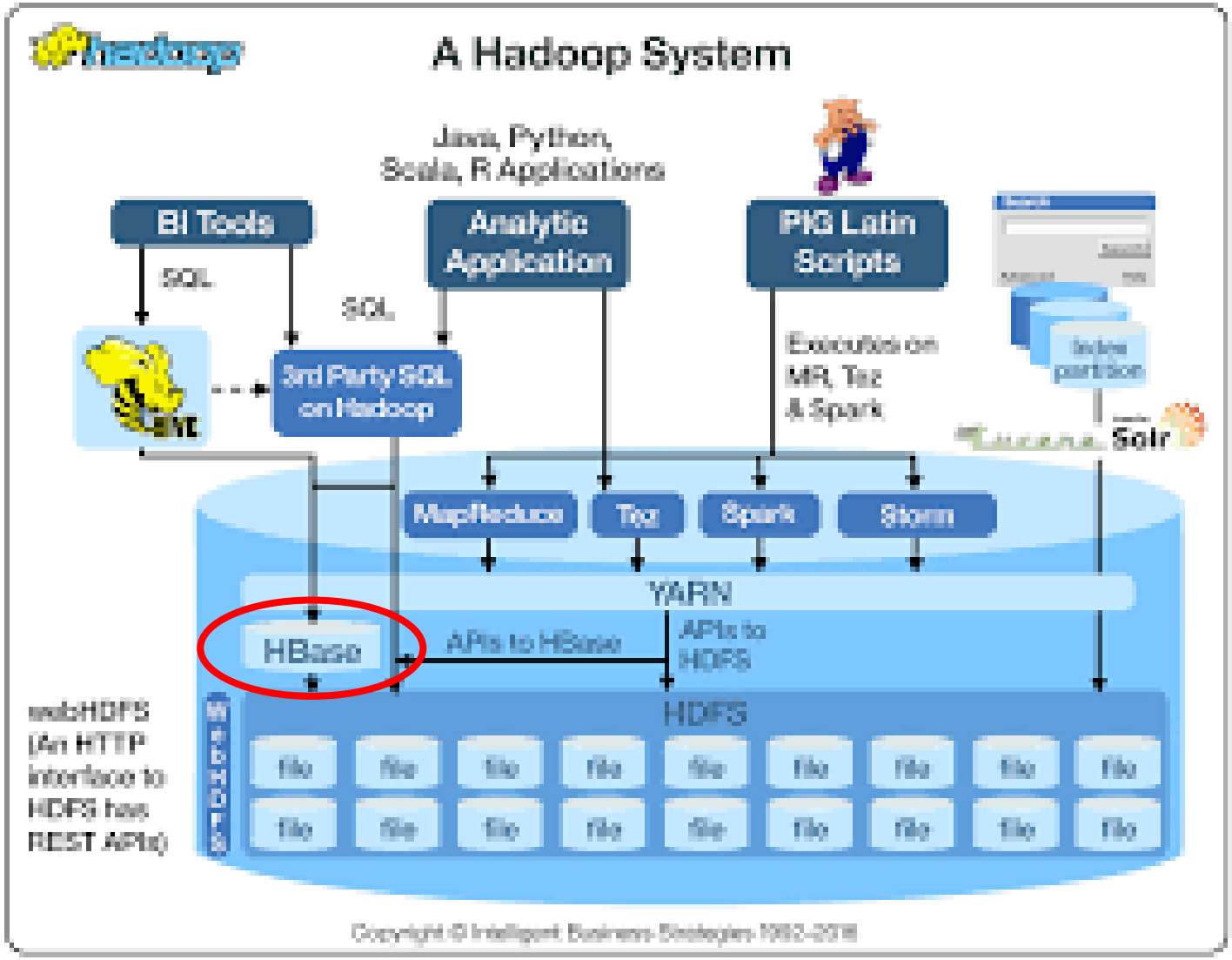
- Generalmente los datos se almacena en sistemas de gestión de bases de datos relacionales (RDBMS) tales como MySQL, Oracle, Postgres, etc.

- Con la llegada del Big Data, las compañías han optado por el beneficio de almacenar sus datos con opciones tales como Hadoop.

- Hadoop permite el almacenamiento de datos en formatos arbitrarios, semi-estructurados o sin estructura alguna.

- **HBase**, Mongo, Cassandra y Dynamo son bases de datos que permiten acceso aleatorio a datos.

- Es un “Framework” de software :
  - Colección de herramienta
  - Soporta aplicaciones distribuidas para “Big Data” (miles de nodos y petabytes de datos)
  - Open Source (Apache License)
  - Sistema de archivos HDFS



Tomado de IBM . https://www.ibmbigdatahub.com/blog/what-hadoop

HBase

- Está construida sobre el sistema de archivos de Hadoop (HDFS) por lo cual se integra fácilmente con este.

- El HDFS provee:

  - Alta tolerancia a fallos.

  - Ejecuta en hardware de bajo costo.

  - Permite trabajar con archivos y conjuntos de datos de gran tamaño.

HBase

- Apache HBase es una base de datos distribuida, escalable y almacén para Big Data.
- Su modelo de datos es similar al de Google BigTable.
- Es una base de datos no relacional de tipo column-wide (o key-value).
- Aplicable para acceder a BigData mediante lectura/escritura en tiempo real o de manera aleatoria.
- Proporciona almacenamiento eficiente para grandes tablas con billones de filas X billones de columnas.

Modelo de Datos

- HBase es un mapa ordenado, multidimensional, persistente, distribuido y disperso.
- Dicho mapa es accedido mediante una tupla *row-key*, *column-key* y *timestamp.*
- Una manera simple de describir HBase es mediante tablas que están compuestas de filas y columnas.
- Dichas tablas marcan el comienzo y el fin de la similitud con los RDBMS.

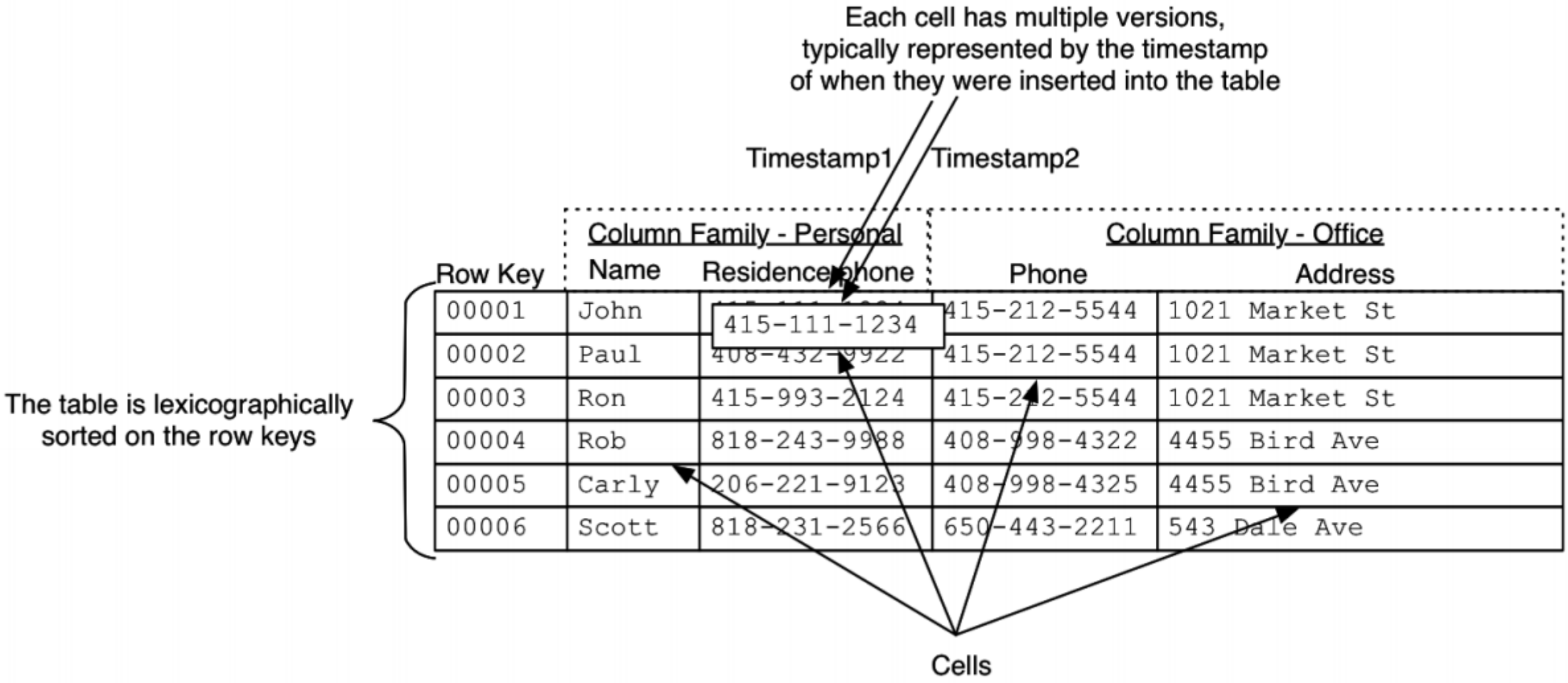
Modelo de Datos - Conceptos Clave

- **Tabla (Table)**:
  - HBase organiza los datos en tablas.
  - Los nombres de la tablas son cadenas de caracteres.
- **Fila (Row)**:
  - La tabla contiene múltiples filas.
  - Cada fila es identificada por un arreglo de bytes *row-key (byte[]).*
- **Familia de Columna (Column Family - CF):**
  - Los datos en una fila están agrupados por *CF.*
  - Todas las filas tienen los mismos *CF*.
  - Cada CF es identificada por una cadena de caracteres.

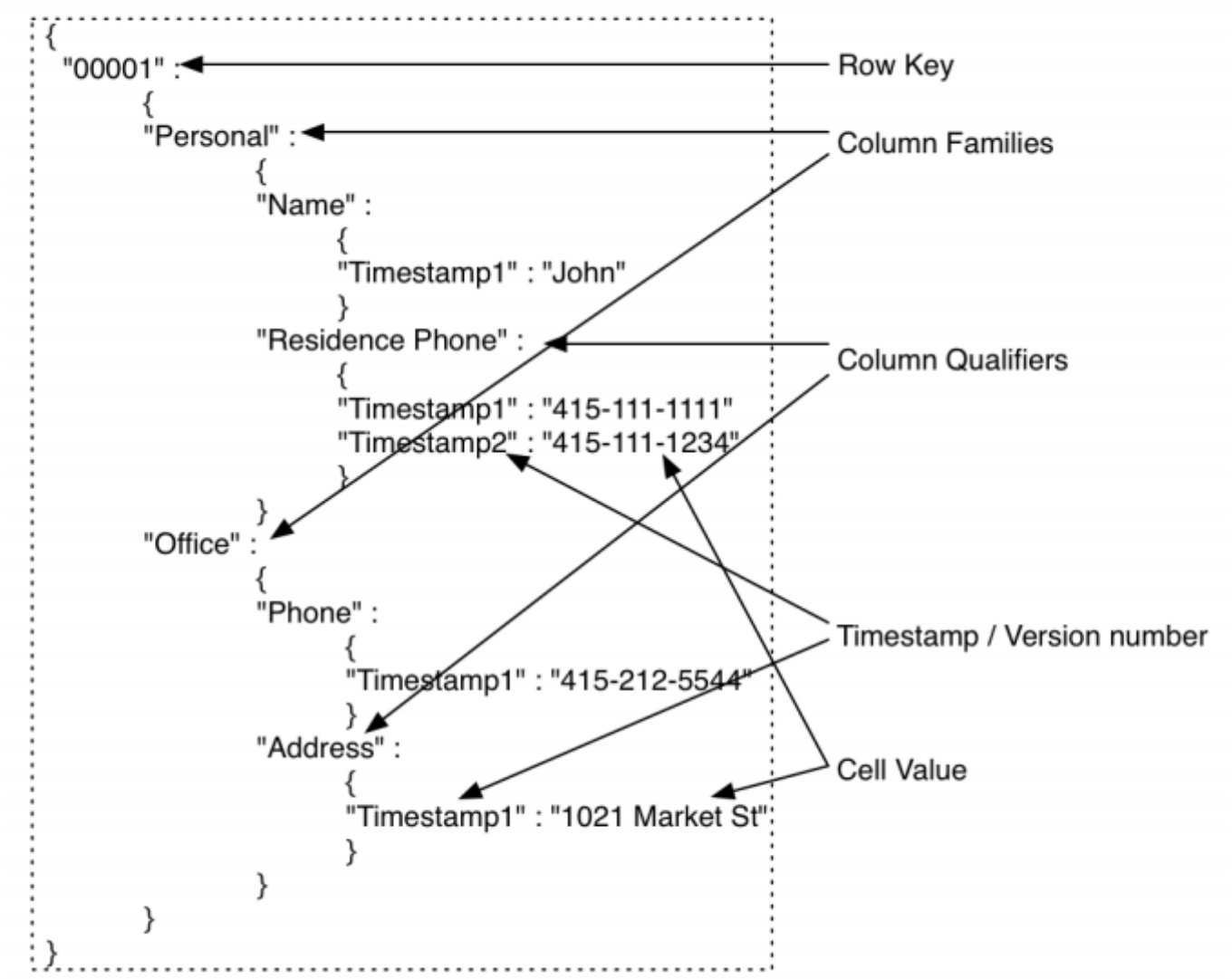
Modelo de Datos - Conceptos Clave

- **Calificador de Columna o Columna (Column Qualifier - CQ):**
  - Los datos en una *CF* son accedidos mediante su *CQ.*
  - Cada CQ está identificado con un arreglo de bytes.
- **Celda (Cell):**
  - La tupla (*row, CF, CQ*) identifica únicamente una celda.
  - Los datos son almacenados en una celda como un arreglo de bytes.
- **Versión (Timestamp):**
  - Es un valor numérico que indica la versión de los datos en una celda.
  - Una celda contiene varias versiones de los datos.

Modelo de Datos - Ejemplo



Modelo de Datos – Mapa Multidimensional



Modelo de Datos – Clave-Valor

00001 → { Personal : { Name : { Timestamp1 : John }, Residence Phone : { Timestamp1 : 415-111-1234 } },
{ Office : { Phone : { Timestamp1 : 415-212-5544 }, Address : { Timestamp1 : 1021 Market St } } }

00001 , Personal → { Name : { Timestamp1 : John }, Residence Phone : { Timestamp1 : 415-111-1234 } }

00001, Personal:Residence Phone → { {Timestamp1 : 415-111-1111 } , { Timestamp2 : 415-111-1234))

00001, Personal:Residence Phone , Timestamp1 → { 415-111-1111)

00001, Personal:Residence Phone , Timestamp2 → { 415-111-1234)

Modelo de Datos

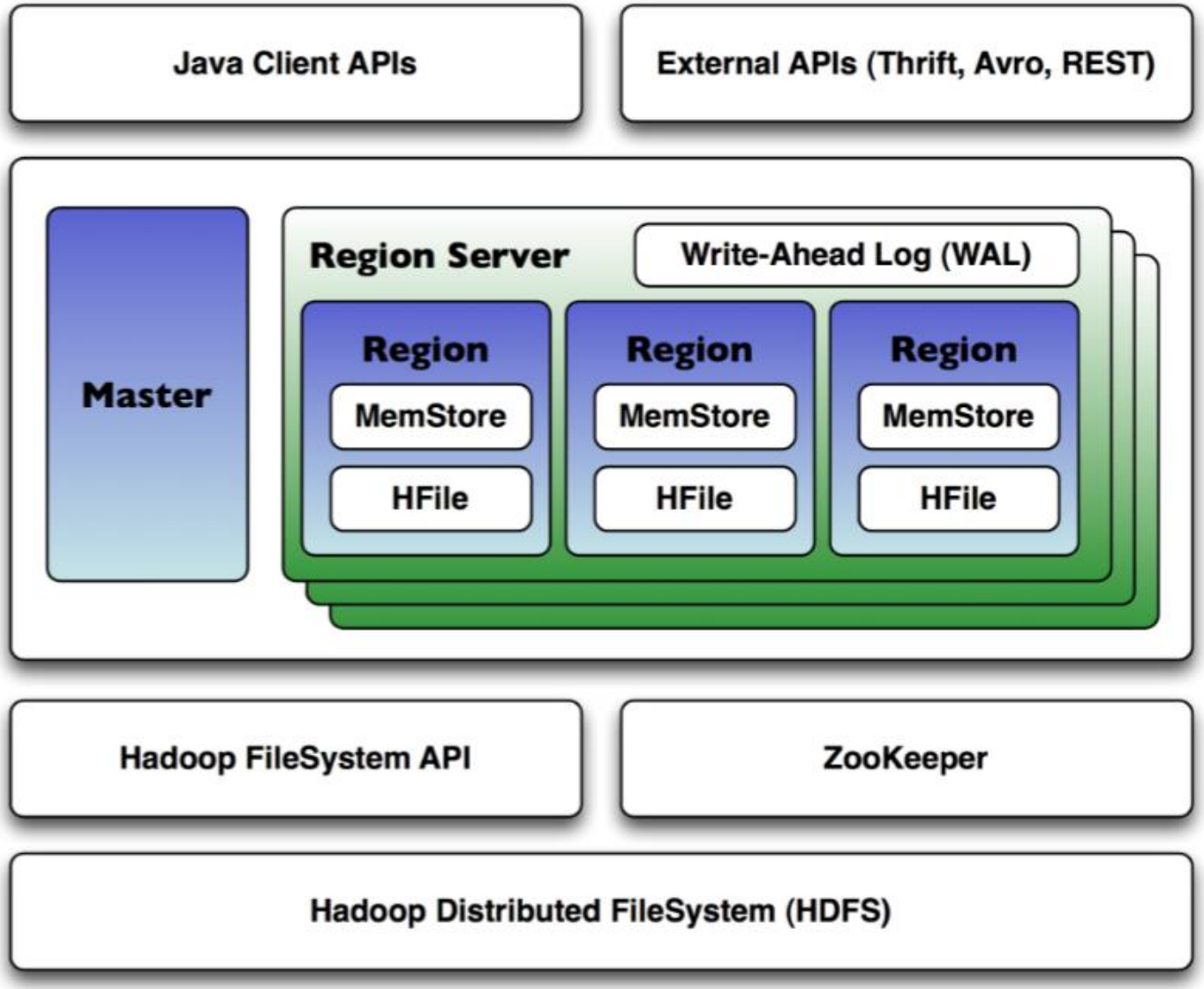
HBase	RDBMS
Libre de esquema. No tiene el concepto de esquema fijo de columnas.	Gobernado por su esquema el cual describe la estructura completa de las tablas.
Construido para tablas de gran tamaño. Es horizontalmente escalable.	Construido para tablas de menor tamaño. Difícil de escalar.
No maneja transacciones.	Son transaccionales.
Utiliza datos sin normalizar.	Utiliza datos normalizados.
Es bueno para datos semiestructurados y estructurados.	Es bueno para datos estructurados.

Operaciones Principales

Las cuatro operaciones principales sobre el modelo de datos son: *Get*, *Put*, *Scan* y *Delete*.

- **Get**:
  - Retorna atributos para una fila específica. Ver Table.Get
- **Put**:
  - Agrega nuevas filas a la tabla (si la clave es nueva) o actualiza las filas existentes (si la clave existe). Ver Table.Put o Table.batch
- **Scan**:
  - Permite iterar sobre múltiples filas para múltiples atributos.
- **Delete**:
  - Elimina filas de un tabla. Ver Table.delete
  - Las filas no son eliminadas al vuelo, son marcadas y eliminadas cuando el sistemas realiza una compactación mayor.

Arquitectura



Tomada de https://dzone.com/articles/handling-big-data-hbase-part-3

Arquitectura

- **Java Client APIs:**
  - API nativa para interactuar con HBase.
- **External APIs:**
  - Permiten interactuar con HBase mediante lenguajes diferentes de Java como C/C++ o diferentes protocolos tales como REST.
- **Master:**
  - Asigna regiones a los *Region Servers.*
  - Se encarga del balanceo de carga entre los *Region Servers*.
  - Administra las modificaciones al esquema y otras operaciones sobre los metadatos, tales como creación de tablas y familias de columnas.
- **Region Server:**
  - Es el encargado de servir y administrar las regiones.
  - Maneja las solicitudes de lectura/escritura para todas las regiones que administra.

Arquitectura

- **Write-Ahead Log (WAL):**
  - Almacena todos los cambios a datos en la HBase en archivos.
  - Si el *Region Server* falla antes que el MemStore es vaciado (*flushed*) el WAL permite que los cambios puedan ser realizados al restaurar el *Region Server.*
  - Si la escritura al WAL falla, la operación entera de modificación falla.
- **Region:**
  - Es el elemento básico de disponibilidad y distribución para las tablas. Está compuesta de un *Store* por *Column Family*.
  - La jerarquía de objetos es la siguiente:

```
Table                (HBase table)
    Region           (Regions for the table)
        Store        (Store per ColumnFamily for each Region for the table)
            MemStore     (MemStore for each Store for each Region for the table)
            StoreFile    (StoreFiles for each Store for each Region for the table)
                Block    (Blocks within a StoreFile within a Store for each Region for
the table)
```

Arquitectura

- **Store:**
  - Almacena un *MemStore* y cero o más *Store Files* (*HFiles*).
  - Un *Store* corresponde a un *Column Family* en para una tabla en una *Region* dada.
- **MemStore:**
  - Almacena en memoria las modificaciones hechas al *Store*.
  - La modificaciones son del tipo Celda/Clave-Valor.
- **Store File (HFile):**
  - Es donde los datos viven en el HFDS.
- **Apache ZooKeeper:**
  - Servicio centralizado para manejo de información de configuración, nombrado, sincronización distribuida, etc.

Aplicaciones en la Industria

- Facebook:
  - Soporta la plataforma de mensajería.
- Twitter:
  - Ejecuta HBase en su cluster basado en Hadoop.
  - Lo utiliza como un backup para las tablas MySQL del backend de la plataforma twitter.
  - Soporta la búsqueda de usuarios.
- Yahoo!:
  - Utiliza HBase para almacenar documentos y detectar duplicaciones de los mismos en el tráfico en tiempo real.
- Adobe:
  - Ejecuta HBase en clusters de 5 a 14 nodos. Aplica tareas Map-Reduce sobre datos estructurados.

Referencias

- Sitio Web HBase, http://hbase.apache.org/
- HBase Wiki, http://wiki.apache.org/hadoop/Hbase
- HBase Reference Guide http://hbase.apache.org/book/book.html
- Google BigTable Paper, http://research.google.com/archive/bigtable.html
- Hadoop web site, http://hadoop.apache.org/
- Introduction to HBase Schema Design, Amandeep Khurana