Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

# PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE

**Erick Steve de la Cruz de la Cruz**
Asesorado por el Ing. Elder Sandoval Molina

Guatemala, noviembre de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

**ERICK STEVE DE LA CRUZ DE LA CRUZ**
ASESORADO POR EL ING. ELDER SANDOVAL MOLINA

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, NOVIEMBRE DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA


## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| VOCAL I | Ing. Alfredo Enrique Beber Aceituno |
| VOCAL II | Ing. Pedro Antonio Aguilar Polanco |
| VOCAL III | Inga. Elvia Miriam Ruballos Samayoa |
| VOCAL IV | Br. Juan Carlos Molina Jiménez |
| VOCAL V | Br. Mario Maldonado Muralles |
| SECRETARIO | Ing. Hugo Humberto Rivera Pérez |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| EXAMINADOR | Ing. Oscar Alejandro Paz Campos |
| EXAMINADOR | Ing. Pedro Pablo Hernández Ramírez |
| EXAMINADOR | Ing. Edgar Josué González Constanza |
| SECRETARIA | Inga. Marcia Ivónne Véliz Vargas |

## *HONORABLE TRIBUNAL EXAMINADOR*

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

**PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE**

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 29 de febrero de 2012.

**Erick Steve la Cruz de la Cruz**

Guatemala, 18 de Junio de 2012

Ing. Carlos Azurdia

Escuela de Ingeniería en Ciencias y Sistemas

Facultad de Ingeniería

Universidad de San Carlos de Guatemala

Respetable Ingeniero:

Por medio de la presente hago de su conocimiento que he revisado a detalle y apruebo el trabajo de graduación realizado por la estudiante **Erick Steve de la Cruz de la Cruz** quien se identifica con el carné número 200412394, y cuyo título es **"PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE".**

Sin otro particular, me suscribo atentamente,

Ing. Eider Sandoval Molina
Cel. 52058569

Ing. Elder Sandoval
Colegiado No. 12074

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de Julio de 2012

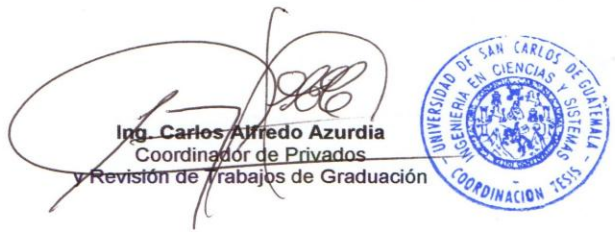
Ingeniero
**Marlon Antonio Pérez Turk**
**Director de la Escuela de Ingeniería**
**En Ciencias y Sistemas**

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **ERICK STEVE DE LA CRUZ DE LA CRUZ** carné **2004-12394,** titulado: **"PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,




**Ing. Carlos Alfredo Azurdia**
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación titulado* ***"PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE",*** *presentado por el estudiante ERICK STEVE DE LA CRUZ DE LA CRUZ, aprueba el presente trabajo y solicita la autorización del mismo.*

***"ID Y ENSEÑAD A TODOS"***

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

*Ing. Marlon Antonio Pérez Turk*
*Director, Escuela de Ingeniería en Ciencias y Sistemas*

*Guatemala, 14 de noviembre 2012*

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.590.2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **PARADIGMA NOSQL: BASES DE DATOS COLUMNARES PARA AMBIENTE DATA WAREHOUSE**, presentado por el estudiante universitario: **Erick Steve de la Cruz de la Cruz,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Alfredo Enrique Beber Aceituno
Decano a.i.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, 16 de noviembre de 2012

/gdech

## ACTO QUE DEDICO A:

| | |
|---|---|
| **Dios** | Por ser el centro de mi vida y guiarme en cada paso que he dado. |
| **Mis padres** | Julio de la Cruz y Aracely de la Cruz. Por darme la vida, su amor y su apoyo incondicional. |
| **Mis hermanos** | Kevin de la Cruz y Cindy de la Cruz. Por apoyarnos mutuamente y seguir unidos. |
| **Mi sobrina** | Sharon de la Cruz. Por alegrar nuestras vidas. |
| **Mi hija** | Emily Lucia de la Cruz. Por ser esa nueva luz en mi vida que me empuja a salir adelante. |
| **Mis amigos** | A Juan de la Roca, Marco Villavicencio y demás compañeros por todas las experiencias compartidas que fortalecieron nuestra amistad, sé que puedo contar con ustedes. |

# AGRADECIMIENTOS A:


| | |
|---|---|
| **La Universidad de San Carlos de Guatemala** | Por ser una importante influencia en mi carrera, entre otras cosas. |
| **Facultad de Ingeniería** | Por ser una importante influencia en mi carrera, entre otras cosas. |
| **Mis amigos de la Facultad de Ingeniería** | Hesler Solares, David Santisteban, Julio Rivera, Jorge Say, Nelson Larín, Luis Pérez, etcétera. |
| **Elder Sandoval** | Por su asesoría en este trabajo, sobre todo por su apoyo incondicional, muchas gracias por la ayuda que me ha brindado, con lo que le estaré eternamente agradecido. |

# ÍNDICE GENERAL

# ÍNDICE DE ILUSTRACIONES

## FIGURAS

# GLOSARIO

**Array** Es una estructura de datos utilizada comúnmente para el almacenamiento de información en memoria, estas contienen una serie de elementos del mismo tipo.

**Base de Datos** Es un conjunto de datos de un mismo contexto, los cuales están almacenados sistemáticamente para su posterior uso.

**Bloqueos** Es la solución al problema de las operaciones concurrentes en una base de datos relacional.

**Business intelligence** Es el conjunto de estrategias o herramientas enfocadas en la administración de información para el análisis y toma de decisiones en una organización.

***Byte*** Unidad básica de almacenamiento de datos, la cual es una secuencia de *bits* contiguos.

**Clúster** Es llamado así el conjunto de computadoras, configuradas de tal forma que uniendo sus recursos se comportan como si fueran una sola.

| | |
|---|---|
| **JOIN (uniones)** | Sentencia SQL que permite combinar registros de una o más tablas. |
| **Entidad Relación** | Herramienta para el modelado de datos que permite representar las entidades de un sistema de información, sus relaciones y propiedades. |
| **NULL (SQL)** | Palabra clave utilizada en lenguaje de consulta SQL, para indicar que un valor no existe dentro de una base de datos. |
| **Llave primaria** | Es un conjunto de uno o más atributos de una tabla, con las cuales es posible identificar de forma única un registro. |
| **OLAP** | Es una solución cuyo objetivo es agilizar las consultas de grandes datos, por medio de estructuras multidimensionales. |
| ***Open-source*** | Es el término con el que se le conoce al software que es distribuido y desarrollado de forma libre, esto quiere decir, que los usuarios no están sujetos a realizar un pago por el uso del software. |
| **RAID** | Se refiere a un conjunto redundante de discos en los que se distribuyen y replican datos. |

| | |
|---|---|
| **RDBMS** | Es el sistema de gestión de base de datos relacional, basado en las reglas publicadas por Edgar Codd. |
| **Sistema distribuido** | Un conjunto de computadoras separadas de forma física, pero conectadas entre sí, de forma que se comunican y comparten datos entre ellas. |
| **SQL** | Lenguaje de consulta estructurado para acceso a bases de datos relacionales, en donde es permitido especificar diferentes operaciones sobre los datos. |
| **Staging area** | Es el área donde los datos en un ambiente de Data Warehouse son analizados y validados para su posterior paso a las tablas de hecho. |
| **Tablas** | En Base de Datos se refiere al tipo de modelo de dato, donde son almacenados distintos tipos de datos los cuales tienen un contexto común. |
| **Tabla de hechos** | Es una tabla central en un esquema dimensional, el cual contiene los valores de las medidas de negocio. |

# RESUMEN

En la actualidad es utilizado un Data Warehouse para la toma de decisiones con base al uso de indicadores dentro de las organizaciones. La información es muy importante para estas organizaciones para el análisis, pero, constantemente esta información se incrementa de forma exponencial a tal punto que se está llegando a los límites de la capacidad de los repositorios donde se almacenan, por lo que, su acceso se vuelve mucho más complicado y se tienen problemas de performance.

En un modelo relacional los datos son presentados en un conjunto de tablas de las cuales existen relaciones entre sí. Con lo que realizar consultas de la información involucra uniones de conjuntos, los cuales equivalen a un costo en cuanto a recursos.

La información es representada de forma horizontal, por lo que esto se basa en el concepto de registro, los cuales contienen información como: número de cliente, nombre, dirección, teléfono, etcétera. Así trabajan los ambientes de Data Warehouse, con lo que en este paradigma la información de los registros es cargada a la memoria accediendo todos hasta obtener el resultado deseado, con esto se tiene una gran ineficiencia ya que al final se leen todos los campos de un registro, cuando algunos de ellos son irrelevantes para solucionar una pregunta de negocio. Y estos tiempos de respuesta terminan siendo insatisfactorios para el usuario.

NoSQL es un paradigma el cual se contrapone a SQL, esto quiere decir, no utilizar relaciones o bases de datos relacionales (NO RDBMS). En general este término abarca todas las bases de datos que no siguen los principios establecidos por un RDBMS, con esto se dice que es un conjunto de productos o conceptos acerca de manipulación de datos y su almacenamiento.

Las bases de datos columnares vienen en contraste con el formato orientado a filas (horizontal) en RDBMS. Las bases de datos columnares almacenan la información de forma vertical, esto quiere decir, que los datos serán almacenados de forma más efectiva, evitando consumir espacio y no se almacenan nulos.

La unidad de los datos se encuentra comúnmente como pares de Llave/Valor, donde cada unidad es identificada por el identificador primario. Las ventajas que ofrecen este tipo de bases de datos son el rendimiento y reducción de utilización de espacio en disco, con lo que es muy factible utilizarlas en ambientes Data Warehouse.

# OBJETIVOS

## General

Proponer un modelo de datos columnar y una arquitectura para la implementación de una base de datos NoSQL, teniendo como base la tecnología Hadoop y así brindar una guía de desarrollo.

## Específicos

1. Mostrar en detalle los conceptos y funcionamiento de la arquitectura Hadoop para una base de datos columnar (HBase) buscando la mejor configuración para un mejor performance y escalabilidad.

2. Implementar el modelo de datos no relacional propuesto para la base de datos HBase, indicando los comandos necesarios y el script utilizado para su creación.

3. Proveer un guía para el desarrollo e implementación de la tecnología NoSQL, específicamente bases de datos columnares.

4. Establecer las mejores prácticas y criterios de Tunning para las bases de datos columnares, aplicándolas en la propuesta realizada anteriormente.

5. Utilización de Hive para la implementación del Data Warehouse.

# INTRODUCCIÓN

El auge que ha tenido el Internet, así como, cada vez es mucho más fácil para las personas conectarse por medio de cualquier dispositivo a la red, ha generado que los usuarios generen mucha más información y esta sea muy importante para el análisis y las organizaciones tomen las mejores decisiones.

Esto ha ocasionado que las bases de datos relacionales se vean afectadas en su rendimiento, con lo que los desarrolladores deben optimizar consultas, programación y ejecución de procesos, dado que estos pueden llevar de entre minutos a horas en la recopilación de los datos.

Por lo que una solución a este tipo de problemas se propone un cambio de paradigma, lo que significa ya no analizar los datos de forma relacional, esto puede crear problemas de resistencia al cambio, pero las soluciones que ofrecen pueden mejorar en el rendimiento de los procesos.

A esto se presenta el paradigma NoSQL, lo cual se contrapone a las distintas bases de datos RDBMS que existen en la actualidad, en donde no existen operaciones como unión de tablas, las cuales generan un gran costo en su ejecución, por lo que se vuelve muy importante el modelo Map/Reduce como paradigma de programación y así ejecutar procesos en forma paralela, utilizando una arquitectura y sistema de archivos distribuido.

# 1. MARCO CONCEPTUAL

## 1.1. Contexto de ambientes de Data Warehouse en la actualidad

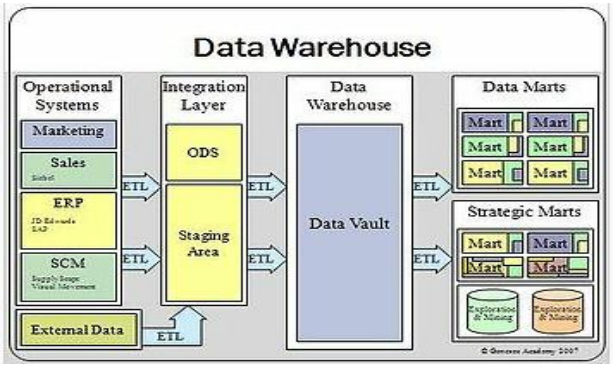
Para adentrarse al ambiente de bases de datos basados en columnas es importante comprender, qué conceptos son los aplicados actualmente, así como, los más aceptados en Data Warehouse, conocer sus ventajas y debilidades para aplicar el nuevo paradigma en los ambientes más apropiados.

### 1.1.1. ¿Qué es Data Warehouse?

Data Warehouse es un almacén de datos sobre el cual se extrae la información más importante de una empresa u organización. Este tipo de información es muy útil principalmente para la toma de decisiones, para contar con una eficiencia de datos y con un mejor análisis y visión de resultados.

En un ambiente de Data Warehouse se extrae la información necesaria de los sistemas operacionales para evitar cualquier tipo de bloqueo o saturación de sistemas, esto se aloja en lo que se llama Staging área, donde dicha información es transformada según las necesidades de la empresa, para finalmente trasladarla a un Data Warehouse y mantener la integridad y calidad de los datos, ver figura 1.

Figura 1. **Ambiente Data Warehouse**




Fuente: Almacén de datos.
http://upload.wikimedia.org/wikipedia/commons/4/46/Data_warehouse_overview.JPG.
Consulta: 24 de mayo de 2012.

Dichos Data Warehouse pueden dividirse en los llamados Data Marts, los cuales son almacenes de datos pero para áreas de negocios más específicas y disponer de información y generar reportes de inteligencia de negocios, los cuales serán de vital importancia para la toma de decisiones de la empresa.

### 1.1.2. Problemas y deficiencias en un ambiente de Data Warehouse

Uno de los principales problemas detectados en los diferentes ambientes de Data Warehouse es el aumento masivo de los datos, esto es a consecuencia del auge que ha tenido en la actualidad el Internet, con las redes sociales, blogs, etcétera. Así como, los diferentes dispositivos con los cuales es posible acceder a la nube.

La recopilación y procesamiento de datos es mayor, con lo que se detecta una deficiencia para manejar estos datos en los modelos convencionales de base de datos entidad relación. Muchos desarrolladores se encuentran con problemas de tiempo, en cuanto a la ejecución de sus procesos, los cuales deben ser programados en ventanas de tiempo que no afecten el rendimiento al momento de ser consumido por los usuarios. Siguiendo en esta misma línea, el aumento de datos ha generado que el espacio físico de disco sea consumido mucho más rápido con lo que afectará el rendimiento para los distintos análisis.

Las bases de datos basadas en un modelo entidad relación permiten mantener la integridad y calidad de los datos y ha sido utilizado por muchos años siendo aceptado en los ambientes de Data Warehouse, pero con este incremento masivo de datos es necesario buscar soluciones alternativas para resolver este problema.

## 1.2. Tecnología NoSQL

La tecnología NoSQL no es un concepto nuevo, la aparición de las redes sociales han incrementado la información generada por los usuarios, por lo que se ha optado esta tecnología como una segunda opción para el análisis de la información.

### 1.2.1. Historia

Con el paso de los años Google ha ido creciendo de forma masiva, con lo cual su infraestructura debe cumplir de forma eficaz y eficiente las solicitudes de los millones de usuarios, entre los cuales utilizan aplicaciones como, Google Maps, GMail, GFinance, etcétera. Lo cual indicaba que Google necesitaba un sistema el cual fuera capaz de obtener tiempos de respuesta óptimos.

Con estos problemas Google opto por crear una infraestructura escalable, la cual fuera capaz de procesar de forma paralela cantidades masivas de datos. Google creó un sistema de archivos distribuido, una base de datos orientada en columnas, un sistema distribuido y un algoritmo Map/Reduce el cual se basará en todos los ambientes mencionados. Con esta infraestructura Google logro el éxito, lo cual finalmente generó el lanzamiento de las investigaciones de Google, lo cual interesó por mucho a los desarrolladores *open-source*.

Con una guía de la infraestructura de Google muchos desarrolladores la imitaron, con lo que se dio el lanzamiento de la Infraestructura Hadoop y las bases de datos orientadas en columnas, las cuales utilizan los principios básicos utilizados por Google.

Figura 2. **Aplicaciones Google**


Fuente: Aplicaciones Google. http://blog.catedratelefonica.deusto.es/wp-content/uploads/2011/11/Google_Apps.jpg. Consulta: 24 de mayo de 2012.

### 1.2.2. Contexto

NoSQL es una combinación de dos palabras, No y SQL. Lo cual significa que esta es una tecnología que se contrapone al tradicional SQL. Esta puede ser llamada de igual forma como NoRDBMS o no Relacional.

Sin importar el término utilizado, NoSQL reúne a todas las bases de datos que no siguen los principios y conceptos de un RDBMS. Lo cual significa que NoSQL es un conjunto de tecnologías que representa diferentes conceptos y principios para el almacenamiento y manipulación masiva de datos. Por el hecho de ser un nuevo paradigma, genera ciertos cuestionamientos en cuanto a su funcionamiento y eficiencia para este tipo de ambientes.

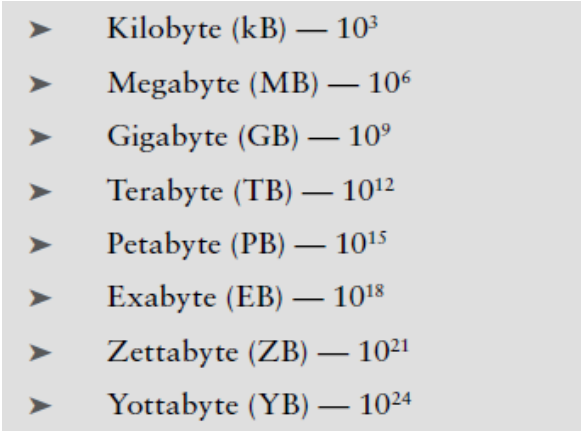
Cuando se habla de un incremento de contenido, es sinónimo de problemas en cuanto a la forma en cómo es manipulada, analizada y almacenada la información. Conforme mejora la tecnología son nuevas fuentes de datos las que aparecen en escena y por ello, es muy importante para las organizaciones analizarlas.

## 1.3. Descripción del término Big Data

El término Big Data se refiere a la cantidad masiva de datos, la cual depende del punto de vista de cada organización. En la actualidad una fuente de datos con unos cuantos *terabytes* es considerada como Big Data.

A continuación se muestran algunas medidas más comunes de *bytes*.

Figura 3. **Medidas en bytes**




Fuente: SHANSHANK, Tiwari. Profesional NoSQL. p. 7-9.

Hace unos años atrás un *terabyte* de información personal era considerado demasiado espacio en disco. Pero en la actualidad ya existen discos duros los cuales almacenan esta unidad de medida.

Por lo que es posible estimar que en un par de años más, esta tendencia ira incrementándose debido a las distintas fuentes de datos que aparecen día a día, se pueden mencionar las redes sociales, blogs, *tweets*, documentos electrónicos, etcétera, estas aumentan de forma exponencial el espacio.

Entre los desafíos que conlleva manejar cantidades masivas de datos se definen los siguientes:

- Eficiencia de almacenamiento y acceso a los datos
- Manipulación de los datos masivos
- Mantenimiento diario de las estructuras de estos datos

Por lo que el almacenamiento y manejo de cantidades masivas de datos no se ajusta para los actuales modelos, NoSQL es una de las soluciones adoptadas para el análisis de Big Data.

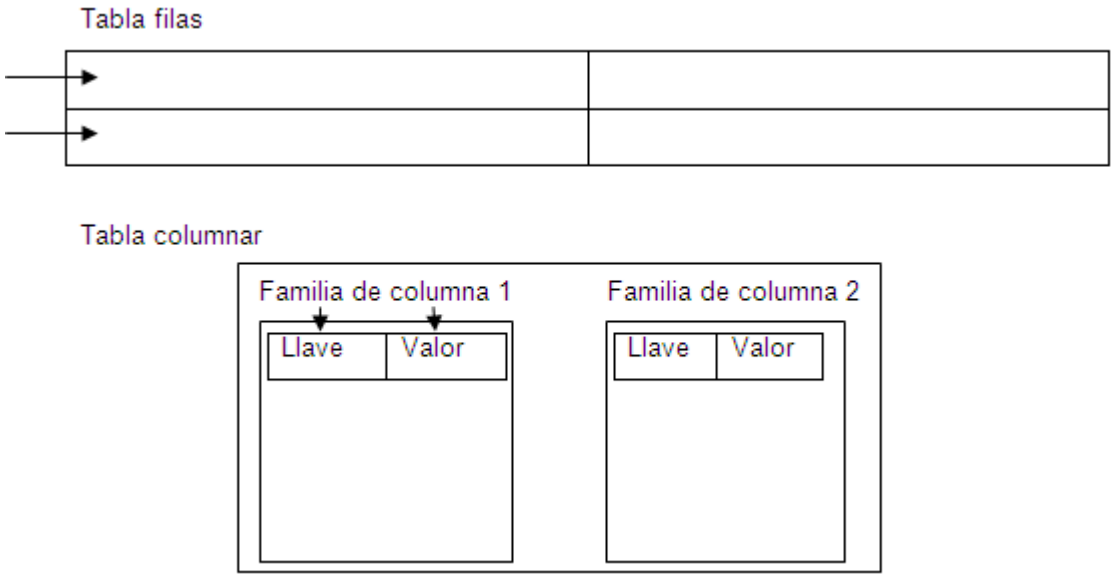
## 1.4. Introducción a NoSQL

NoSQL es un conjunto de tecnologías las cuales se contraponen a los modelos convencionales (RDBMS), las cuales buscan solucionar los problemas de eficiencia, escalabilidad y mantenimiento de datos masivos. Entre los cuales se mencionarán los modelos NoRDBMS más comunes y más utilizados:

### 1.4.1. Bases de datos columnares

Este modelo de base de datos permite que la información sea almacenada de forma efectiva, evitando los valores nulos, simplemente no insertando nada al momento que un valor no exista. Las unidades de datos ya no son insertadas como filas convirtiéndolas a pares de llaves y valores dando la definición de cada una de las columnas.

Figura 4. **Tabla basada en filas y columnas**




Fuente: elaboración propia.

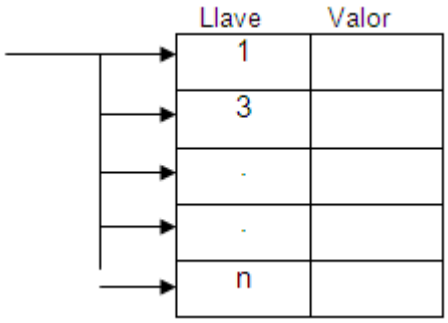
Este modelo de columnas es utilizado con más frecuencia en ambientes Data Warehouse para los análisis OLAP. Entre las bases de datos más conocidas están Hbase, Cassandra, Hypertable, etcétera.

### 1.4.2. Bases de datos Llave/Valor

Este tipo de base de datos como su nombre lo indica son parejas de valores identificadas por una llave, los mapas Hash o un array es una de las estructuras básicas en las cuales se almacena un par de valores con estas características. Este tipo de estructuras ofrecen muchísima eficiencia para el acceso a datos.

Sobre estos existen varios tipos de este modelo, entre los cuales se pueden mencionar: los que utilizan la memoria RAM para el acceso a datos, así como, son capaces de almacenar la información en disco, otro tipo de almacenamiento es el uso de Cache, con lo que se toma una fotografía de los datos para ser usados en las aplicaciones y así evitar los accesos de lectura y escritura en el disco.

Figura 5. **Base de datos Llave/Valor**



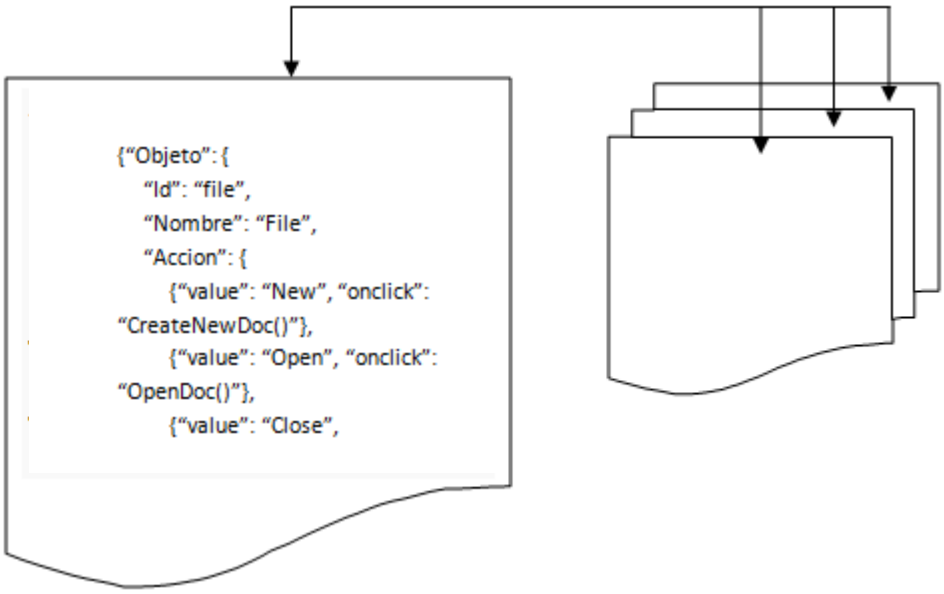

Fuente: elaboración propia.

Entre las bases de datos *open-source* más populares se pueden mencionar DynamoDB, Riak, Redis, Berkley DB, Voldemort, etcétera.

### 1.4.3. Bases de datos basadas en documentos

Este tipo de base de datos no es ningún sistema administrador de documentos como se puede llegar a pensar, sino se refiere principalmente a conjuntos estructurados de datos en documentos, los cuales pueden ser adjuntados como un conjunto de documentos sobre un área específica, además de ser indexados no sólo con base a un identificador primario sino también por sus atributos.

Figura 6. **Bases de datos basadas en documentos**



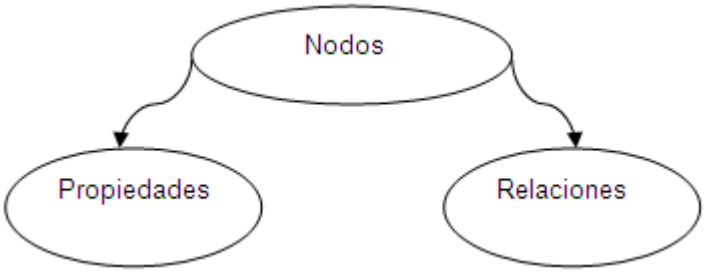

Fuente: elaboración propia.

Entre las bases de datos *open-source* más populares se pueden mencionar MongoDB, CouchDB, RavenDB, RaptorDB, etcétera.

### 1.4.4. Bases de datos basadas en grafos

Este tipo de bases de datos se basa principalmente en el uso de nodos y relaciones de un grafo, para representar cualquier tipo de dato manteniendo una estructura dominante.

Figura 7. **Base de datos basada en grafos**




Fuente: elaboración propia.

Entre las bases de datos *open-source* que se pueden mencionar son Neo4j, Infinite Graph, Sones, HyperGraphDB, Trinity, etcétera.

## 1.5. Bases de datos columnares para un ambiente Data Warehouse

Según observaciones indicadas anteriormente, es posible concluir que una base de datos RDBMS es muy poco eficiente en cuanto a tiempos de respuesta al momento que la cantidad de datos manejada por la misma, aumenta de forma masiva, lo cual genera problemas al desarrollar un ambiente Data Warehouse utilizando este modelo.

Estas bases de datos brindan una solución al problema del incremento masivo de datos, debido a la facilidad de acceso y resumen de datos que es posible lograr debido a su estructura de columnas. Esto es muy útil en los ambientes Data Warehouse y así generar de forma más eficiente reportes de inteligencia de negocios o reportes OLAP para el análisis de datos.

La principal comparación entre las bases de datos basadas en filas y columnas es comúnmente sobre la eficiencia de acceso al disco para la carga de datos. Se pueden mencionar los beneficios que provee esta tecnología:

- Las bases de datos columnares son más eficientes para el cálculo de valores, tales como, sumas, conteos, restas, promedios, etcétera.

- Estas bases de datos son más eficientes para la inserción de datos, esto debido a que un dato de columna puede ser escrito y reemplazado sin afectar o buscar otros valores de las columnas.

- Las bases de datos columnares encajan perfectamente para reportes de inteligencia de negocios por su rapidez de acceso a todos los datos para ser analizados.

## 1.6. Introducción a operaciones y consultas en bases de datos columnares

Un aspecto muy importante para la población de datos son las operaciones de creación, lectura, actualización y eliminación de datos. Por lo que es importante conocer cómo estas operaciones se aplican en el mundo de NoSQL, en esta tecnología las operaciones de creación y lectura son las más importantes, tanto así, en algunas ocasiones son las únicas utilizadas.

### 1.6.1. Creación de registros

Al crear un nuevo registro un usuario, significa que debe existir una forma de identificar que el registro ya existe, si no este debe ser actualizado y no creado nuevamente.

En las bases de datos relacionales los registros son almacenados según una llave primaria la cual identifica a cada registro como único. Cuando es necesario realizar una inserción se realiza una búsqueda por medio de la llave primaria para determinar si el valor existe o no.

Las bases de datos columnares no están definidas por el concepto de relaciones, ya que las tecnologías NoSQL evitan las uniones entre los datos. Estas bases de datos almacenan los datos de forma desnormalizada casi replicando una tabla de hechos en el Data Warehouse.

Con lo que la dimensión tiempo es muy utilizada para almacenar los datos, esto significa que la base de datos almacena diferentes versiones del valor asociado en las familias de columnas, por lo que el valor mostrado al momento de ser seleccionado se presenta el más reciente, las demás versiones de igual forma pueden ser consultadas, esto hace que la creación de registros sea muy eficiente.

### 1.6.2. Acceso a los datos

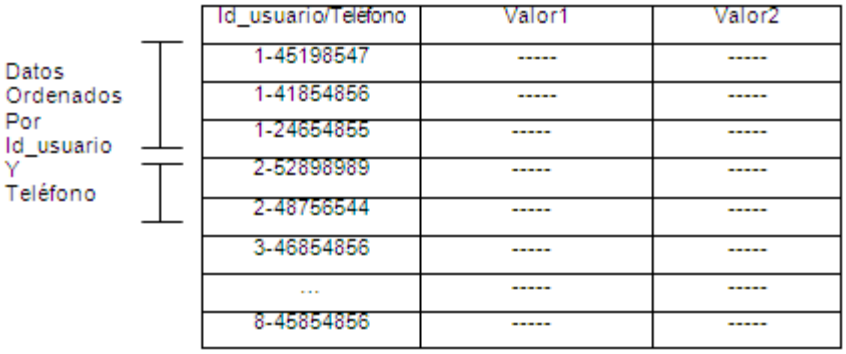
La forma más fácil y eficiente de ejecutar consultas en las bases de datos columnares, es por medio de un indicador que direccione al dato que se desea consultar. Estos valores son ordenados y almacenados para que su acceso sea óptimo. Por lo que, siempre es necesario indicar una Llave/Valor el cual identifique cada uno de los registros.

Esto significa que los datos serán accedidos por medio de un valor. Es de mucha utilidad relacionar el valor de la llave con los datos contenidos en la tabla. Como una buena práctica es muy común combinar las columnas de la tabla para crear valores de llave, esto dependerá de qué datos sean los más consultados.

A manera de ejemplo, si en un sistema se desean buscar todos los teléfonos que posee un usuario, sería una buena práctica combinar los valores de Id_usuario y número como llave, por lo que se tendría ordenado de forma física la llave en los archivos y será mucho más óptima su búsqueda.

En la figura 8 es posible observar la forma en que los datos serán almacenados y ordenados, a manera que estos se encuentren juntos y así será mucho más rápido el acceso a datos en el sistema de archivos. Normalmente el ordenamiento de los datos es realizado con base en los *bits* de cada carácter y no de forma alfanumérica.

Figura 8. **Llave para acceso a datos columnares**




Fuente: elaboración propia.

Este mecanismo es muy importante al momento de buscar registros en una cantidad masiva de datos, estas bases de datos además soportan índices secundarios, los cuales pueden aportar más soluciones para el acceso a los datos.

### 1.6.3. Actualizando y eliminando datos

Las bases de datos relacionales están basadas en los principios de atomicidad, consistencia, aislamiento y durabilidad, las cuales consolidan la integridad de los datos, para aplicar las actualizaciones sobre los datos.

Las bases de datos columnares al contrario no dan mayor importancia a las transacciones de actualización y en algunos casos las ignora por completo. Para iniciarse debe tener claro que los conceptos de bloqueo no existen en las bases de datos columnares, ya que los bloqueos en los sistemas distribuidos son muy complejos y genera deficiencia en tiempos de respuesta.

Algunas bases de datos columnares soportan muy limitadamente el bloqueo de lectura-escritura a nivel de valores. Esto significa que los valores serán bloqueados cuando cualquier columna dé ese valor, si está siendo modificada, actualizada o creada.

La atomicidad en las bases de datos columnares soportan los más mínimos principios, comparado con las tradicionales. Estas son comúnmente establecidas en sistemas distribuidos con lo que es muy complejo contar con una perfecta integridad y consistencia de datos, ya que algún fallo o error en un servidor puede afectar por completo los datos reales en el clúster.

La consistencia eventual es un término comúnmente aplicado en la programación paralela y sistemas distribuidos, este concepto significa que dado un determinado tiempo donde no se han realizado actualizaciones, se espera que estas actualizaciones eventualmente se propaguen en las replicas de datos para que sean consistentes. Por otro lado al contar con continuas actualizaciones, una actualización eventualmente será aplicada en una réplica o la réplica estará fuera de servicio.

Finalmente se ha dado una breve introducción de las operaciones de creación, lectura y actualización de los datos en donde los conceptos varían por cada una de las tecnologías. Así como, las operaciones de creación y selección son las más importantes y más limitadas las de actualización.

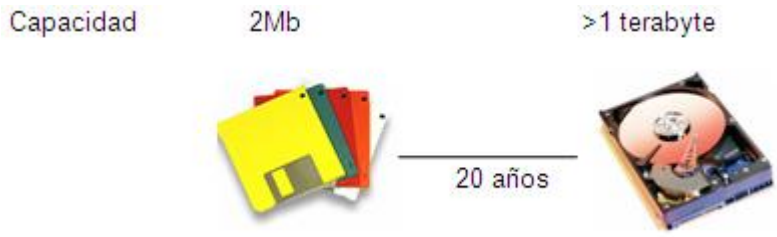
# 2. DISEÑO DE MODELO DE DATOS Y ARQUITECTURA DE BASE DE DATOS COLUMNAR

## 2.1. Almacenamiento y análisis de datos

Las necesidades en cuanto a la capacidad de almacenamiento ha ido variando conforme los años transcurren, el problema es que esto ha ido aumentando de forma masiva.

A inicios de 1990 se manejaban discos con una capacidad no mayor de 5 *megabytes*, con lo cual la lectura completa de los datos en los discos podía tomar no más de 5 minutos, conforme el paso de los años y las novedades de la tecnología aparecieron en escena los discos con capacidades mayores a 1 *terabyte* y el tiempo para consultar la totalidad de los datos oscila entre las 2 y 3 horas.

Figura 9. **Discos y su capacidad de almacenamiento**

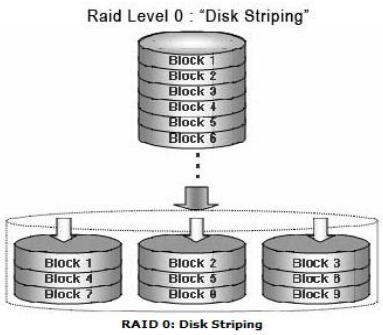



Fuente: elaboración propia.

Esto es demasiado tiempo para la lectura de los datos, la escritura es incluso más lenta, por lo que una solución para la reducción del tiempo de lectura es generar búsquedas de forma paralela en n discos, con lo que es posible leer los datos en unos cuantos minutos. Esto significa compartir la información a través de los discos, lo cual hará que los usuarios del sistema sean capaces de consultar la información compartida, para realizar análisis de datos en un tiempo más corto de manera que no exista interferencia entre ellos.

El hecho de trabajar en forma paralela con los discos trae consigo el problema de fallos de hardware, esto dado que al momento de contar con muchas piezas de hardware estas llegarán a fallar con el tiempo, esto significa que las probabilidades de fallos aumentan. Una solución a esto es aplicar la replicación de datos, esto significa la realización de copias redundantes de los datos de tal forma que estos estén disponibles para el sistema en caso de cualquier tipo de falla. Esta es la funcionalidad de RAID, sin embargo, el sistema de archivos Hadoop tiene otra solución cercana a los RAID.

Figura 10. **RAID 0**




Fuente: Raid nivel 0. http://www.ddmsa.com/prod/diskarray_raid.html. Consulta: 3 de junio de 2012.

Un segundo problema es la combinación de los datos extraídos de todas las fuentes. Los sistemas distribuidos permiten combinar la información de las diferentes fuentes pero realizar este tipo de operaciones es algo muy complejo. Con esto la programación Map/Reduce y la arquitectura Hadoop permite abstraer la complejidad de lecturas y escrituras, implementando un sistema de lectura y análisis compartido entre servidores.

Esta es una solución a los problemas de almacenamiento masivo y compartido, por lo que utilizar esta tecnología es una opción disponible en la actualidad.

## 2.2. Atributos de calidad

Los atributos de calidad son un aspecto importante para las aplicaciones en una organización, las cuales brindarán el mejor desempeño de las mismas. Y garantizan la calidad de los datos, entre las que se mencionan.

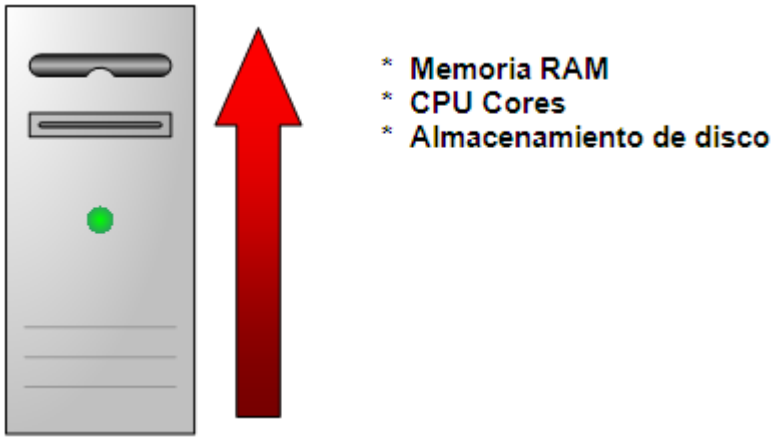
### 2.2.1. Escalabilidad

Un aspecto muy importante y a tomar en cuenta al implementar cualquier tipo de arquitectura es la escalabilidad. Esta es la habilidad de un sistema de incrementar sus recursos. El objetivo principal de esta propiedad es proporcionar por un largo tiempo los mejores recursos para cumplir con las demandas de los usuarios, normalmente esto es aplicado con la implementación de un clúster con muchos servidores no tan potentes para que funcionen como uno solo.

Cuando se habla de escalabilidad se mencionan dos tipos:

- Vertical: este tipo de escalabilidad se basa en preparar poderosas computadoras, instalándoles muchos CPU, aumentar su memoria y almacenamiento, para ir aumentando los recursos de los servidores y así contar con súper servidores.

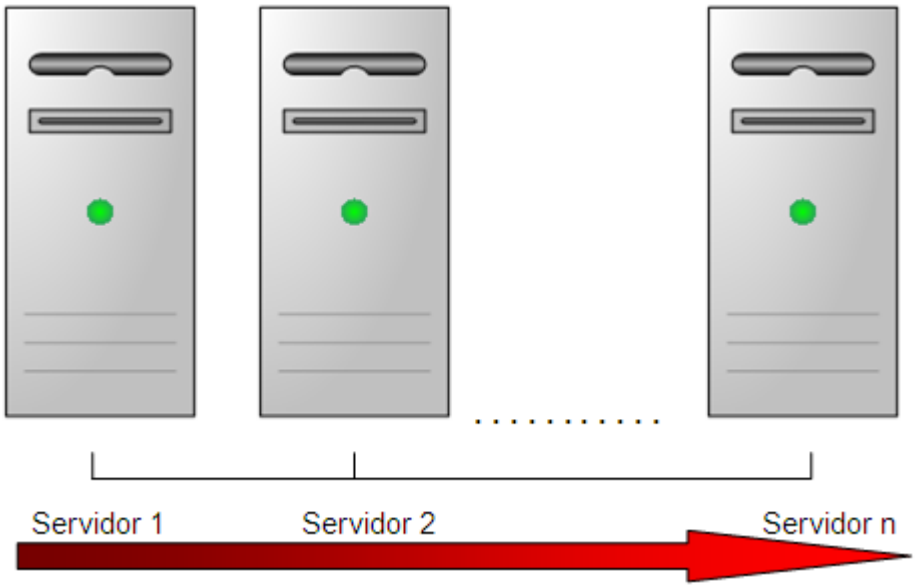
Figura 11. **Escalamiento vertical**




Fuente: elaboración propia, con base al programa Microsoft Visio 2007.

- Horizontal: la escalabilidad horizontal tiene que ver con un sistema de clúster, esto significa que se aumentan los recursos con base en los distintos nodos, los cuales van agregándose, dando como resultado un funcionamiento en conjunto para la mejora de recursos.

Figura 12. **Escalamiento horizontal**




Fuente: elaboración propia, con base al programa Microsoft Visio 2007.

Con el aumento masivo de los datos y la necesidad de manipular procesos paralelos, se ha adoptado principalmente la escalabilidad horizontal en las infraestructuras. Esta escalabilidad horizontal ha sido aplicada en compañías como Google, Amazon, Facebook, eBay y Yahoo teniendo algunas de estas estructuras de entre mil a más de cien mil servidores.

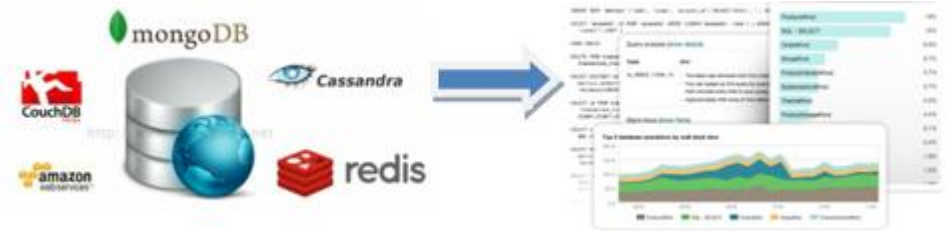
Dado que procesar demasiada información a través de los clúster horizontales es muy complejo, se han desarrollado aplicaciones utilizando los modelos de Map/Reduce, las cuales proveen los conceptos y métodos necesarios para trabajar con procesos a gran escala en un clúster horizontal de servidores.

### 2.2.2. Rendimiento

El rendimiento es un aspecto muy importante para las organizaciones, esto implica la ejecución de los procesos importantes de forma rápida y correcta, dicho atributo puede verse muy afectado con el aumento masivo de datos, lo cual hace que tome demasiado tiempo procesar los datos y así los usuarios se verán afectados para generar sus análisis.

Esto trae el desafío de optimizar las aplicaciones y procesos, los cuales deben ser solucionados según sus requerimientos y contextos para proveer una solución general a todos los escenarios. En los modelos RDBMS el rendimiento se ha visto directamente afectado por el aumento de datos, lo cual hace que se busquen nuevas soluciones de diseño y estructura de modelos de datos.

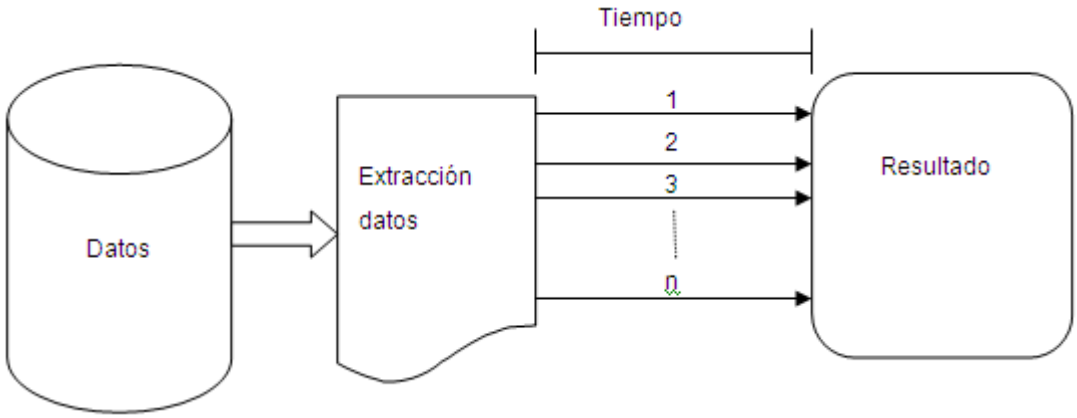
Figura 13. **Bases de datos NoSQL**




Fuente: SQL/NoSQL Performance. http://newrelic.com/features/sql-nosql-performance.html. Consulta: 3 de junio de 2012.

El modelo Map/Reduce ayuda a evitar estados de latencia, la latencia es precisamente la medida de tiempo que le toma a un proceso ejecutar la tarea más pequeña de un proceso o la ejecución completa del mismo. Mientras más rápido sea un programa tomará menos tiempo para que el usuario obtenga un resultado.

Figura 14. **Ejecución paralela de procesos**




Fuente: elaboración propia.

Reducir esta latencia implica utilizar los algoritmos correctos para obtener los resultados esperados, una de las soluciones más utilizadas es la ejecución paralela de procesos. Entre las ventajas que ofrece este tipo de algoritmos es que al momento de ejecutar subtareas paralelas independientes se reduce el tiempo de ejecución al momento de existir un fallo. Lo cual permite ejecutar el subproceso donde se detectó el fallo y así evitar la ejecución completa del proceso, con lo cual se reduce la latencia.

Con esto se concluye qué dados algoritmos proporcionan los modelos Map/Reduce, estos deben ser los adecuados para aplicarlos de manera más efectiva para que ayuden a mejorar la latencia de los procesos.

### 2.2.3. Integridad de los datos

La integridad de los datos es un aspecto muy importante para las bases de datos, a tal punto que es vital debido a que los datos deben persistir de forma correcta con el paso del tiempo y de las distintas operaciones sobre estas.

En una base de datos se espera que los datos no sean perdidos ni estén corruptos durante el almacenamiento, mantenimiento o procesamiento. Este riesgo aumenta mucho más según el volumen de datos que se maneje y si estos datos se encuentran de forma distribuida este problema se vuelve mucho más complejo.

Una forma muy común de detectar si los datos están corruptos es realizando cálculos de suma para comprobación de los datos, cuando esta ingresa por primera vez al sistema. Esto significa que por cualquier cambio realizado en los datos debe calcularse una suma de comprobación, la cual debe cuadrar con un cálculo de datos anterior a los cambios.

Esta técnica no corregirá los errores en los datos, únicamente es una forma de detección de estos.

Con esto se concluye que para el usuario es importante contar con los datos completos, debido a que esto puede afectar directamente y de manera muy crítica los reportes para análisis. Y así reportar datos erróneos los cuales pueden traer graves consecuencias para las organizaciones.

## 2.3. Introducción a Hadoop

Hadoop se enfoca principalmente en la ejecución de procesos distribuidos y remotos. Además de ser un software libre siempre es tomado como una opción de prueba para las organizaciones.

### 2.3.1. Breve historia

La implementación de un sistema web de búsquedas es una tarea muy compleja debido al software y arquitectura requerida para contar con una indexación óptima de los sitios web. Así como, se requiere de un amplio equipo de desarrollo donde los costos aumentan de forma directa.

Según una estimación de costos un sistema el cual tenga una capacidad de soportar 1 billón de páginas indexadas podría costar más de medio millón de dólares en hardware con un costo de mantenimiento de más de $30 000 dólares mensuales, lo cual requiere una fuerte inversión y un riesgo muy alto para cualquier organización.

Aproximadamente, en el 2003 se publicaron los documentos que describían la arquitectura de Google. En donde se detallaron las ventajas que ofrece su sistema de archivos el cual soluciona las necesidades de almacenamiento de archivos, de gran tamaño. Un año más tarde Google redactó un el documento donde introdujeron el concepto Map/Reduce.

En el 2006 aparecieron en escena los proyectos *open-source*, los cuales utilizaron los conceptos e implementaron una arquitectura la cual imitara a Google y así gozar de sus beneficios. En el 2008 emergió el proyecto Hadoop con Apache, dando así la confirmación de su éxito en la comunidad.

En el paso de las distintas pruebas y experiencias de los usuarios este fue siendo adoptado por organizaciones tales como Yahoo, Facebook y Amazon. Dejando huella en el mundo, rompiendo un record mundial convirtiéndose en el sistema más rápido para ordenar 1 *terabyte* de datos.

Figura 15. **Logo que identifica tecnología Hadoop**


Fuente: Ecosistema Hadoop. http://www.christian-ariza.net/wp-content/uploads/2010/10/hadoop+elephant_rgb.png. Consulta: 3 de junio de 2012.

### 2.3.2. ¿Qué es Hadoop?

Para los usuarios son muy importantes los datos, ya que de estos se basan para la toma de decisiones. Dicho esto se debe asegurar la disponibilidad, integridad, el rendimiento y la flexibilidad de los datos. Todo esto lo proporciona la tecnología Hadoop, para la solución de estos problemas los cuales se presentan normalmente con el manejo masivo de datos.

Hadoop es un ecosistema de productos alojados en Apache Software Foundation, el proyecto está relacionado con infraestructuras distribuidas para el procesamiento de datos a gran escala. Conforme ha ido creciendo esta tecnología, más proyectos se han desarrollado, ya sean de Apache o de terceros para ir complementando los servicios de Hadoop y así ampliando la tecnología.

Entre los distintos proyectos *open-source* Hadoop es posible mencionar algunos de ellos:

- Common: son todos los componentes e interfaces para el sistema de archivos distribuidos (Java RCP, codificación, etcétera).

- Avro: es un sistema de codificación de objetos como medio de almacenamiento de los datos.

- Map/Reduce: es el modelo distribuido de proceso de datos el cual ejecuta en clúster de máquinas.

- HDFS: es un sistema de archivos distribuidos el cual se ejecuta en un clúster de máquinas.

- Pig: es un lenguaje de flujo de datos y para ambiente de ejecución para grandes fuentes de datos. Este proyecto se ejecuta en clúster con HDFS y Map/Reduce.

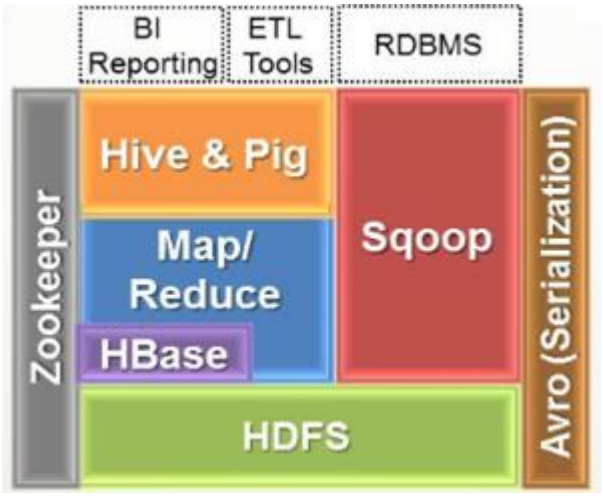
- Hive: es utilizado para Data Warehouse distribuidos. Este maneja los datos almacenados en HDFS y provee un lenguaje basado en SQL para la selección de datos.

- HBase: es una base de datos distribuida basada en columnas. Utiliza HDFS para el almacenamiento, así como, soporta Map/Reduce.

- ZooKeeper: es un servicio distribuido, el cual provee locks para utilizarlos en aplicaciones distribuidas.

- Sqoop: es una herramienta diseñada para transferir de forma eficiente los datos de estructuras de datos entre Apache Hadoop y bases de datos relacionales.

Figura 16. **Ecosistema Hadoop**




Fuente: Proyecto apache. http://simranjindal.files.wordpress.com/2011/10/bigdata_4.png. Consulta: 3 de junio de 2012.

Los proyectos derivados dependen y se ejecutan sobre HDFS y Map/reduce. Hadoop es un sistema basado en archivos y no una base de datos, generalmente todos los demás proyectos agregados a Hadoop se ejecutan y dependen del HDFS, el cual es completamente distribuido y tolera fallos, con esto se afirma que este es diseñado específicamente para implementarlo en clúster, por lo que es posible aumentar su escalabilidad de forma horizontal.

Map/Reduce es un modelo para desarrollo de software, el cual permite implementar aplicaciones para ejecutarse en paralelo, de forma distribuida. Otro aspecto importante es que esta tecnología trabaja con datos no estructurados, esto significa que los datos que almacena no deben tener aplicados ninguno de los conceptos de un RDBMS. Esta tecnología es considerada NoSQL dadas estas propiedades. Siendo una tecnología NoSQL no es posible acceder a los datos con cualquier motor de SQL convencional. Entre las aplicaciones que existen para el acceso a datos se tienen a las ya conocidas: Java, c++, c, Python, etcétera. Y las desarrolladas exclusivamente para Hadoop: Hive, Pig, etcétera.

Según el éxito de organizaciones tales como Google, Facebook, Amazon y las experiencias de otros usuarios, esta tecnología tiene un potencial enorme, el cual en la actualidad aún no ha sido completamente explotado por la comunidad, esto quiere decir, que no se puede pensar en reemplazar los sistemas de bases de datos tradicionales, más bien se espera que se experimente de forma conjunta para evolucionar junto a Hadoop.

## 2.4. Sistema de archivos distribuidos Hadoop (HDFS)

El aumento de datos ha ido creando problemas para los desarrolladores quienes tienen el desafío de mejora de rendimiento para el procesamiento de datos. Una de las soluciones recomendadas es implementar particiones de datos utilizando distintos servidores, dichos sistemas de archivos los cuales son administrados a través de una red de servidores llamados sistemas de archivos distribuidos.

En este tipo de sistemas se debe estar preparado para los problemas en la red, siendo el más común el fallo de los nodos, haciendo que se tengan pédidas de datos.

Hadoop contiene implementado un sistema de archivos llamado HDFS (Hadoop Distributed Filesystem) el cual ha sido principalmente diseñado para el almacenamiento de grandes archivos, los cuales son alojados en clúster. Esto quiere decir, que este sistema de archivos es capaz de administrar *gigabytes*, *terabytes* y hasta *petabytes*. Además, que es un sistema con escalamiento horizontal no es importante obtener servidores costosos con demasiados recursos, contando con servidores estándares, los cuales implementen un clúster y así aumentar los recursos de forma horizontal.

### 2.4.1. Bloques de datos

El tamaño de bloque de un disco es la unidad mínima de datos la cual se puede leer o escribir, normalmente los bloques de los sistemas de archivos tienen el tamaño de algunos *kilobytes*. HDFS también maneja el concepto de bloques con la única diferencia que estos, son mucho más grandes, una razón del porque del tamaño de bloque, es que se minimiza el costo para la búsqueda de datos. Con esto el tiempo de transferencia de datos es mayor pero se reduce el tiempo de búsqueda del inicio del bloque.

Una de las ventajas que ofrecen los sistemas distribuidos es que los bloques no deben estar necesariamente en un mismo disco, con esto es posible tener archivos de un gran tamaño distribuidos en el clúster para acceder a los datos de forma más rápida y sin el temor de tener poca capacidad de almacenamiento en los discos.

### 2.4.2. Name Nodes y Data Nodes

En un HDFS clúster existen dos tipos de nodos: Name Node (nodo maestro) y varios Data Nodes. Los Name Nodes manejan y administran los sistemas de archivos, estos contienen la definición de cómo se encuentra estructurado el sistema de archivos, de igual forma sabe en qué Data Node se encuentran alojados los bloques de los archivos. Los Data Nodes son los encargados de almacenar y proveer los bloques de datos cuando el cliente o el Name Node los necesitan, así como, indican al Name Node de forma periódica qué bloques están almacenando.

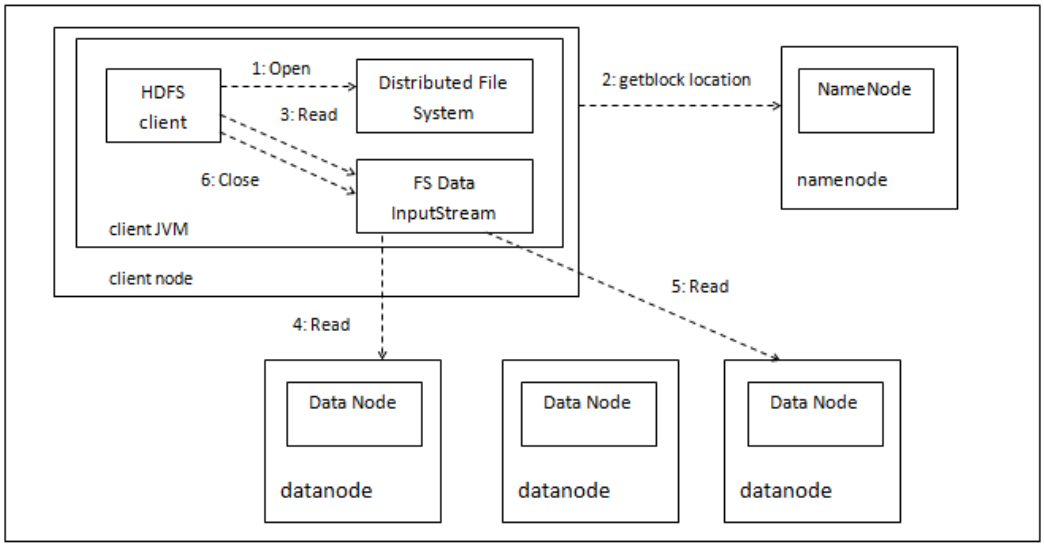
Comprendiendo cómo está distribuido y cómo es administrado el HDFS, los Name Node son muy importantes e indispensables, esto significa que si se pierde por completo o existe un fallo en el Name Node todos los bloques y archivos almacenados en los Data Nodes se perderán por completo ya que no habrá forma de reconstruir la definición de su estructura por lo que ya no podrán ser visualizados.

Una manera de evitar la pérdida de datos es realizando un respaldo de los archivos de estructuras del Name Node. Otra forma de solucionar este problema es colocando un segundo servidor Name Node, el cual se recomienda que cuente con los mismos recursos, con lo que su funcionamiento sería obtener una copia de la estructura de datos del Name Node principal para que en caso de fallo este se ponga en funcionamiento y así sea transparente para los usuarios.

### 2.4.3. Flujo de datos

Dado que se ha definido el bloque de archivos y como son estos administrados y estructurados, es importante aprender qué pasos sigue el flujo de datos de lectura y escritura.

Figura 17. **Flujo de lectura de datos**

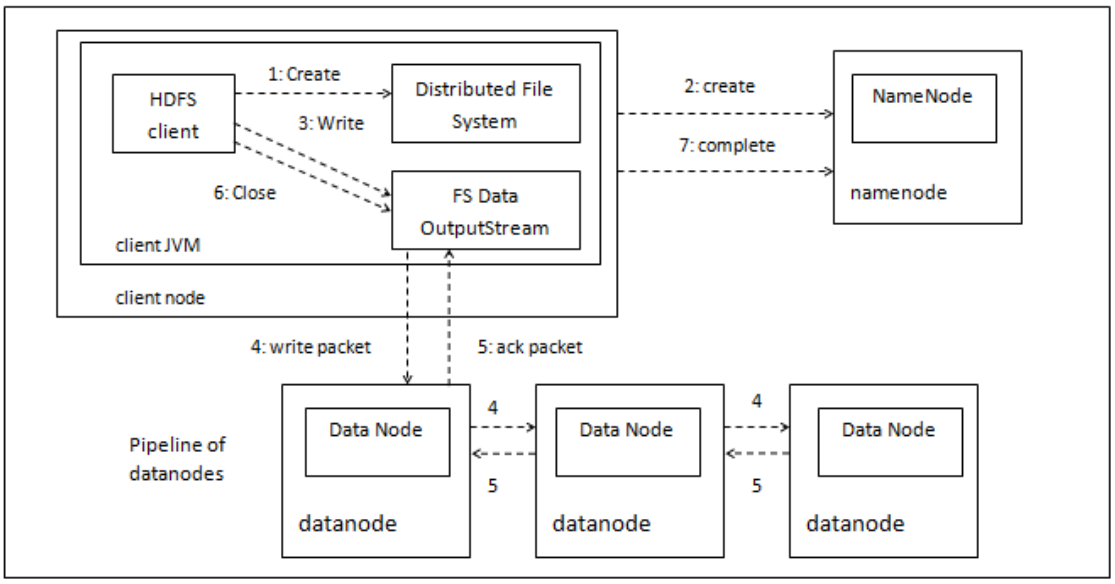



Fuente: WHITE, Tom. Hadoop: the definite guide. p 62-65.

El flujo de datos para una lectura, inicia con la petición de un cliente para leer un bloque de datos, al hacer la petición el cliente llama al Name Node para que le especifique la ruta de acceso para obtener el bloque de datos. Ya teniendo la ruta el cliente hace la petición a los distintos Data Nodes que contienen el bloque de datos solicitado, además, que verifica la distancia más cercana de los distintos nodos para obtener la información. Cuando el cliente ya obtiene su requerimiento todas las conexión son cerradas, esto ocurre de forma transparente para que el cliente únicamente lea los datos obtenidos.

Al momento de existir algún error en el Data Node donde se encuentra el bloque de datos solicitados, se busca en el siguiente Data Node más cercano que contenga el bloque que se necesita, luego el cliente notifica al Name Node, que Data Node ha fallado o que tenga información corrupta para tomarlo en cuenta en la próxima búsqueda.

Figura 18. **Flujo de escritura de datos**




Fuente: WHITE, Tom. Hadoop: the definite guide. p 65-68.

El flujo de escritura de datos inicia en el momento que el cliente desea almacenar datos, como primer punto se conecta con el Name Node el cual verifica si el cliente tiene permisos de escritura de archivos, siendo esto valido verifica que el dato no esté ya almacenado, de cumplirse estas condiciones el Name Node creará el registro de los nuevos datos.

Los datos son empaquetados para solicitar a cada Data Node el almacenamiento del bloque y tener replicas del bloque de datos. En caso exista alguna falla en un Data Node este quedará en lista de espera y se continuará realizando la réplica en los demás Data Node en la lista. Estos son los principales conceptos básicos a cubrir para tener una noción de cómo funciona y qué es posible aplicar en las tecnologías Hadoop y así implementarlas en los ambientes más adecuados para el manejo masivo de datos.

## 2.5. Propuesta para modelo de datos columnar

Para comprender el modelo de datos en las bases de datos columnares, se presentará un modelo conceptual utilizando los conceptos entidad/relación y los conceptos de datos columnares.

### 2.5.1. Introducción a HBase

Dado que las bases de datos relacionales se ven afectadas en su rendimiento según aumenta la cantidad de datos a almacenar, para este tipo de problemas traen soluciones como la replicación y particionamiento, las cuales son difíciles de instalar o administrar, por otro lado se tienen uniones, vistas, llaves foráneas, etcétera. Las cuales vienen generando un costo elevado al momento de inserción y consulta de datos.

HBase al ser una base de datos distribuida orientada en columnas está diseñada específicamente para dar solución a estos problemas escalando de forma horizontal, lo cual ayuda en el rendimiento del almacenamiento masivo de datos. Esta base de datos se ejecuta sobre el HDFS lo cual brindará las ventajas de este sistema de archivos.

### 2.5.2. Mapa conceptual

En HBase así como, cualquier otros DBMS, los datos están almacenados en tablas, cada una de las tablas está compuesta por filas y columnas, en la intersección de estas se encuentran las celdas, las cuales están versionadas. En cada una de las tablas se tienen las llamadas llaves de fila, las cuales están formadas por arrays de *bytes*, esto quiere decir, que cualquier tipo de valor o combinación de estos podrá ser permitido.

Todos los valores de la tabla están ordenados por su llave de fila para acceder a los datos de forma más rápida. Las columnas están agrupadas en lo que se llama familias de columnas, todas las columnas que conformá una familia de columna deben estar relacionadas de alguna manera. Esto debido a que cada una de ellas estará almacenada junta en el HDFS, con esto el sistema de archivos podrá acceder a los datos de forma más óptima.

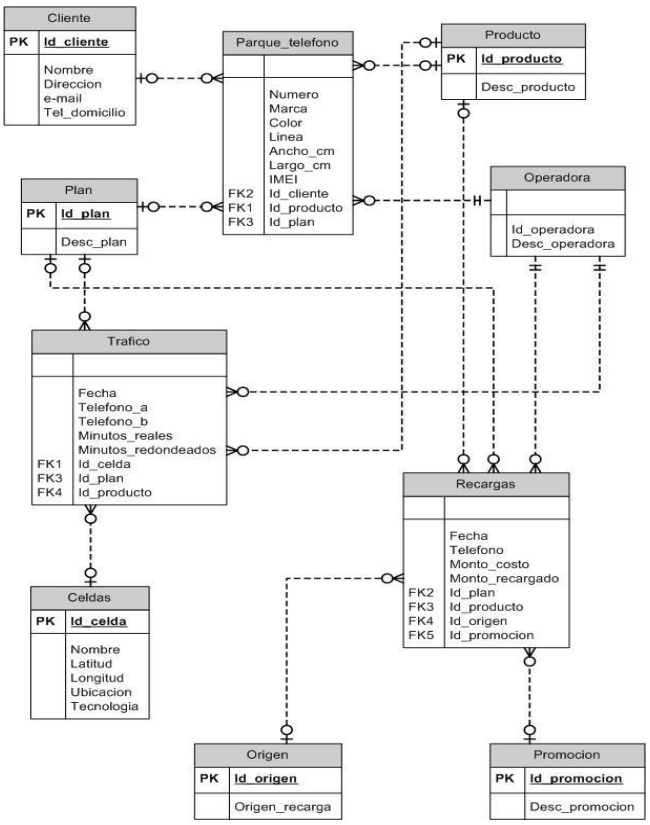
### 2.5.3. Modelo de datos propuesto (RDBMS)

Uno de los negocios que crece de forma exponencial es la telefonía móvil, cada día las personas adquieren muchos teléfonos móviles ya sea por necesidad o por adquirir la tecnología de punta, de igual forma los usuarios realizan o reciben muchas llamadas generando una cantidad considerable de tráfico, al realizar una llamada se consume tiempo de aire, para esto es necesario realizar recargas de tiempo de aire en donde se indica la cantidad a recargar.

Para estas organizaciones es importante analizar el comportamiento del mercado y se requieren reportes de la cantidad de teléfonos dados de alta por día, la cantidad de tráfico por operadoras, la cantidad de ingresos generados por las recargas, etcétera. Todo este tipo de información para los analistas les es de ayuda para la toma de decisiones, esta información debe estar disponible lo más pronto posible para los usuarios.

El principal problema para este tipo de análisis es que la cantidad de datos va aumentando demasiado, lo cual trae como consecuencia un procesamiento de datos demasiado lento, acá es donde las bases de datos relacionales comienzan a verse afectadas por el rendimiento, haciendo que se apliquen técnicas más complejas para mejorar el rendimiento, pero con el paso del tiempo y conforme sigan los datos aumentando cada vez será más complicado dar una solución práctica a este problema. El diseño de un modelo de base de datos relacional y es posible observarlo en la figura 19.

Figura 19. **Propuesta de modelo conceptual de estrella**




Fuente: elaboración propia.

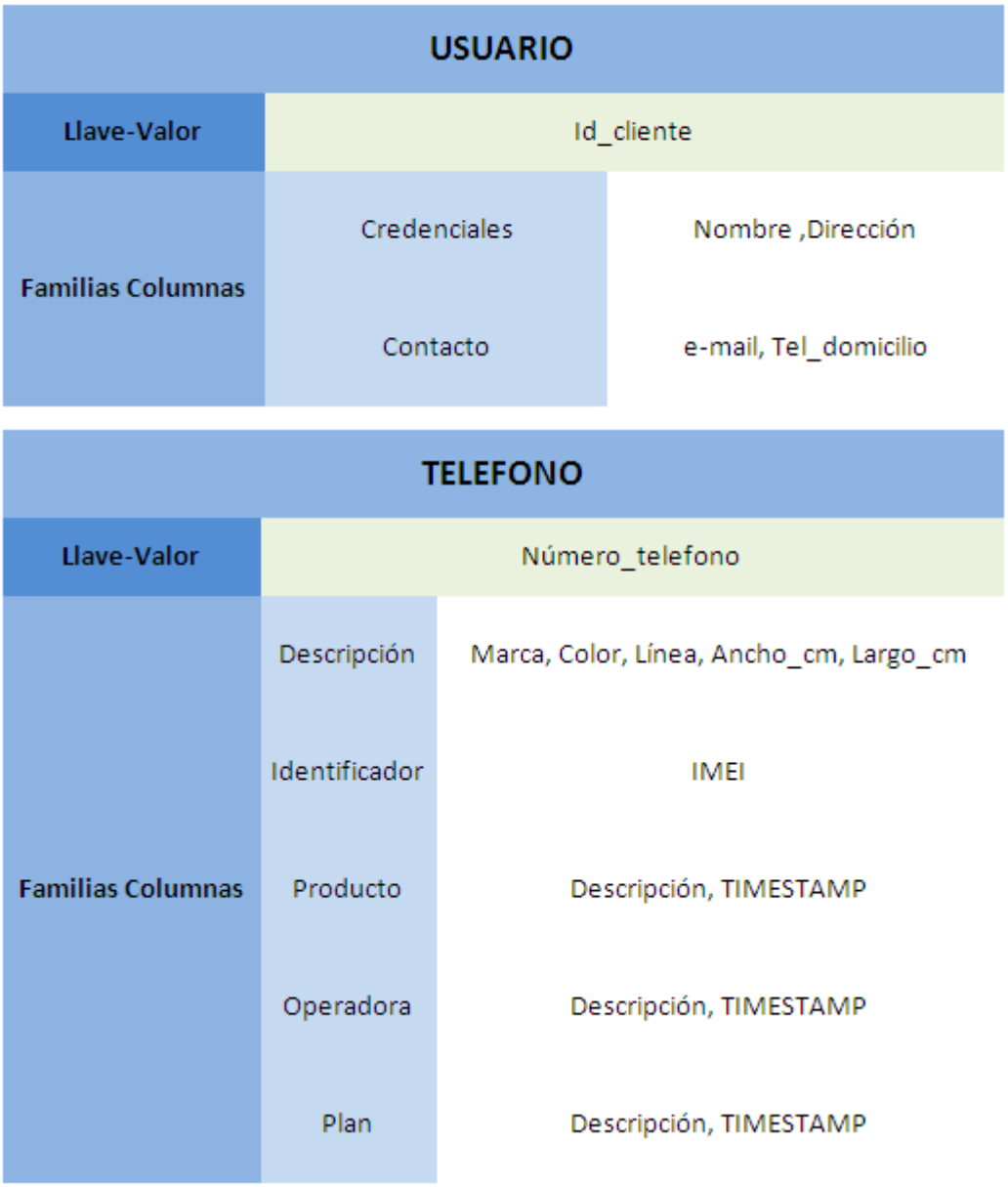
### 2.5.4. Modelo de datos propuesto (No RDBMS)

En las bases de datos columnares no existen las uniones, por lo que una solución a estos es el uso de tablas look-up las cuales indicarán una relación entre el usuario y los teléfonos, por lo que se creará la tabla usuario-teléfono, además se observa que la tabla teléfono ha absorbido las tablas plan, producto y operadora. Esto es debido a que estas tablas están relacionadas directamente con el teléfono por lo que se agregarán como familias de columnas de la tabla, a cada una de estas se ha agregado de forma explícita el manejo de versiones con el tipo de datos TIMESTAMP ya que el valor de esta puede cambiar con el tiempo.

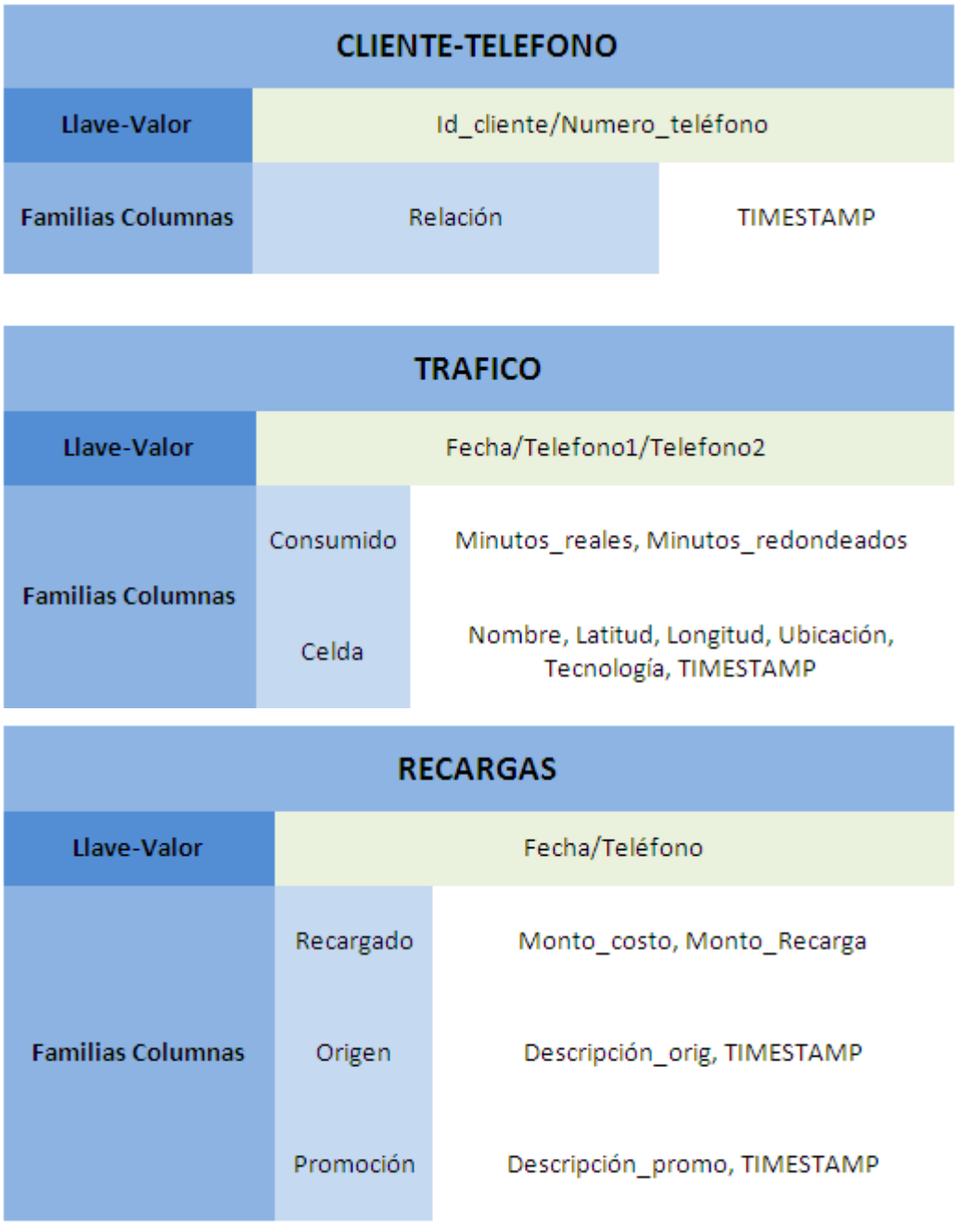
En las tablas, recargas y tráfico se ha tomado como llave valor la concatenación de la fecha y teléfonos, para acceder de forma eficiente los datos, estas tablas absorben la de catálogos, aplicando el mismo concepto en las tablas anteriores.

El diseño de la base de datos columnar, para implementarla en la base de datos HBase, se detalla en la figura 20.

Figura 20. **Modelo de base de datos columnar**



Continuación de la figura 20.




Fuente: elaboración propia.

## 2.6. Propuesta de arquitectura para la base de datos columnar

Para construir la arquitectura sobre la cual se instalarán y crearán tablas y familias de columnas, las cuales el usuario accederá e insertará datos.

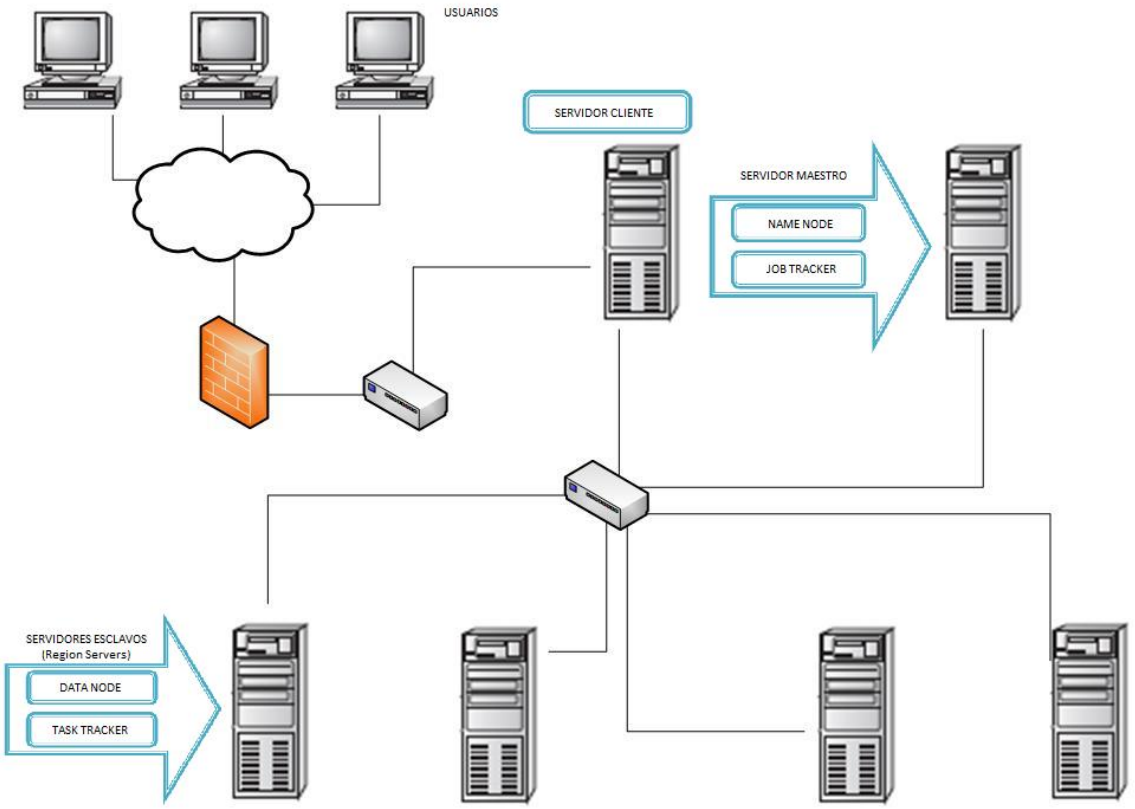
Como primer punto es necesario contar con un servidor cliente, el cual tiene configuradas las herramientas de Hadoop para proveer los datos de los usuarios a ser almacenados, así como, obtener y consultar los datos de la base de datos. Desde este servidor los usuarios se conectarán para realizar las distintas operaciones mencionadas.

De igual forma existirá un servidor el cual se llama Maestro, el cual será el encargado de coordinar y supervisar el almacenamiento de datos distribuido y los procesos paralelos, este servidor almacenará las bitácoras de inserción y configurado el Job Tracker para los procesos.

Los servidores esclavos los cuales pueden ser cualquier número de servidores, todo dependerá de cómo crezcan los datos, en estos es donde se almacenan los datos de forma distribuida y reportan al servidor maestro como se encuentran estructurados, de igual forma tienen configurado un Daemon el cual comunicará al Job Tracker del servidor maestro para la ejecución de los procesos.

En la figura 21 se muestra el diagrama de la topología de red de cómo estarán configurados un servidor cliente, un servidor maestro y cuatro servidores esclavos.

Figura 21. **Topología de arquitectura de red propuesta**




Fuente: elaboración propia.

## 2.7. Modelo Map/Reduce para implementar arquitectura

Map/Reduce es un modelo de desarrollo para el procesamiento de datos, que permite trabajar de forma paralela, escalando cada proceso en el clúster de servidores. El principal beneficio que ofrece este modelo es la eficiencia para analizar más de un *terabyte* de datos.

La forma en que este modelo funciona es dividiendo las tareas en dos fases: Map (Mapeo) y Reduce (Reducción). Cada una de las fases está compuesta por pares de llaves/valor las cuales son una entrada y salida de datos. La fase de mapeo tiene como entrada los datos a analizar, estos pueden estar almacenados en un texto, en una tabla, etcétera. La fase de reducción consiste en buscar el dato solicitado luego de que la información se encuentra agrupada o mapeada de una forma que sea más eficiente y eficaz su búsqueda, la entrada de la fase de reducción son los pares de Llave/Valor de la salida de la fase de mapeo y la salida de la fase de reducción será de igual forma pares de Llave/Valor, pero con el requerimiento solicitado.

La tecnología Hadoop proporciona ventaja de ejecutar cada una de las tareas de forma paralela, haciendo que cada servidor en el clúster ejecute el mapeo y reducción de forma separada para luego unir cada uno de los resultados y así procesar los datos de forma rápida, además se encarga de optimizar el balanceo de carga por cada uno de los servidores y así no exista diferencia en tiempos de ejecución dados los recursos de cada servidor.

Para comprender de forma más detallada este modelo se tomará como ejemplo con los datos del tráfico de teléfonos. Para este caso se tiene el tráfico de forma diaria ordenado en carpetas por su fecha, ver figura 22.

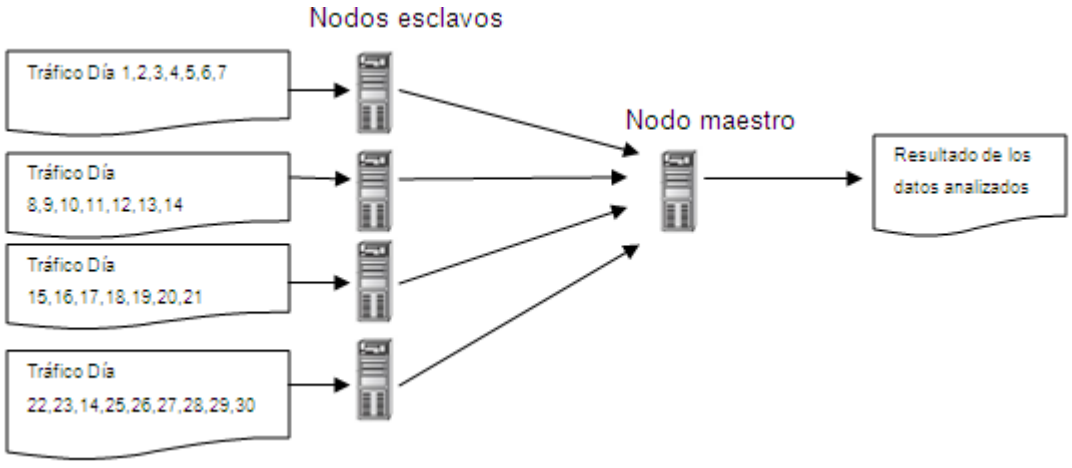
La cantidad de datos contenida en cada una de las carpetas es extremadamente grande, dado que la cantidad de llamadas realizadas por cada teléfono son millones por cada día, ahora sí se trata de analizar los datos en el mes o en el año, será una tarea extremadamente complicada, para esto ayuda el modelo Map/Reduce y la tecnología Hadoop.

Figura 22. **Datos del tráfico diario almacenado**


Fuente: elaboración propia.

Al utilizar Hadoop los datos serán ejecutados de forma paralela y almacenados en cada nodo del clúster, si se reparte la carga de trabajo por día en cada nodo, ver figura 23.

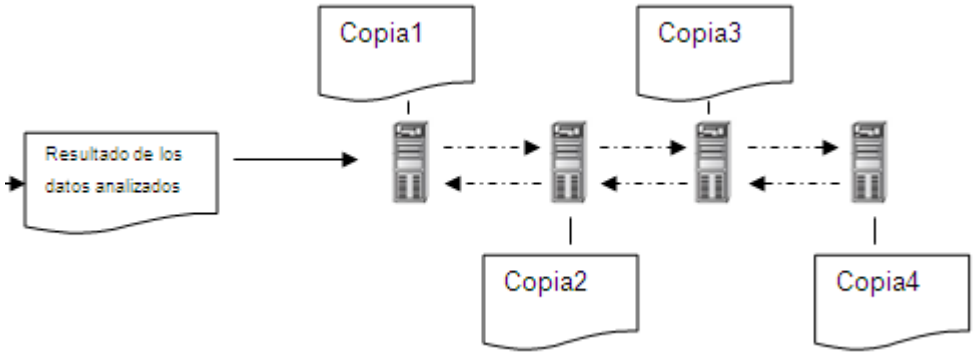
Figura 23. **Balanceo de carga en los nodos**




Fuente: elaboración propia.

Cada nodo esclavo procesará los días asignados de tráfico, el resultado final será reportado al nodo maestro, el cual contendrá el resultado final del conjunto de datos analizados. Si el resultado desea almacenarse el nodo maestro creará copias en cada uno de los nodos y replicar los datos para la prevención de pérdida de datos.

Figura 24. **Replicación de almacenamiento de datos**




Fuente: elaboración propia.

Los datos de un día de tráfico serán almacenados en un archivo de texto, en donde se tienen los minutos que han gastado los distintos números de teléfono, para este caso se analizará la forma en cómo operará un nodo dentro del clúster, ver figura 25.

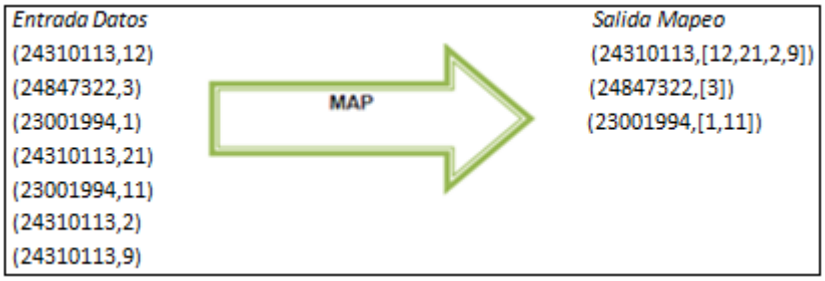
Figura 25. **Datos de un día de tráfico**

| Fecha | Teléfono1 | Teléfono2 | Minutos | Celda |
|---|---|---|---|---|
| 02-may-2012 | 24310113 | 43747894 | 12 | A00X32 |
| 02-may-2012 | 24847322 | 41198712 | 3 | A011B2 |
| 02-may-2012 | 23001994 | 44310901 | 1 | A0022A |
| 02-may-2012 | 24310113 | 42312098 | 21 | A090AA |
| 02-may-2012 | 23001994 | 24442233 | 11 | AAB44A |
| 02-may-2012 | 24310113 | 22123444 | 2 | BB3A42 |
| 02-may-2012 | 24310113 | 44231212 | 9 | AR43AA |
| ..... | ..... | ..... | ... | ..... |
| ..... | ..... | ..... | ... | ..... |

Fuente: elaboración propia.

Para este caso se desea obtener el valor de la mayor cantidad de minutos realizada por el teléfono que realiza llamada saliente, dado el requerimiento se pueden identificar dos valores que se necesitan utilizar, para este caso el teléfono 1 y minutos. Para la primera fase de mapeo se tomarán estos datos como parejas de Llave/Valor, ver figura 26.

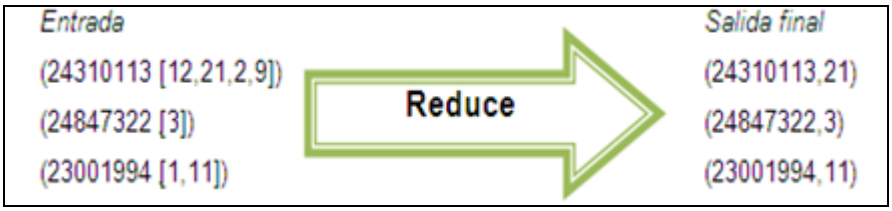
Figura 26. **Entrada y salida fase de mapeo**




Fuente: elaboración propia.

La salida de la fase de mapeo son los datos agrupados para un fácil acceso, la segunda fase será la reducción donde a continuación se verá el resultado con los datos de ejemplo.

Figura 27. **Entrada y salida de la fase de reducción**

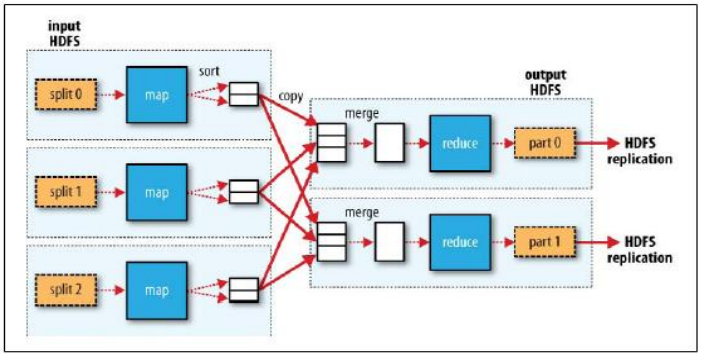



Fuente: elaboración propia.

Finalmente se obtiene el resultado del requerimiento solicitado, este es un pequeño ejemplo de cómo funciona Map/Reduce para tener una idea de su funcionamiento. Con las ventajas de ejecución en paralelo que ofrece Hadoop pueden ejecutarse los días del mes de mayo en distintos servidores y obtener el resultado de forma más rápida, en la fase de mapeo cada uno de los archivos puede incluso reducirse aún más, en lo que se llama Splits en donde cada archivo es divido en partes más pequeñas para su ejecución y balancear la carga de procesamiento en los distintos servidores.

De igual forma para la fase de reducción, luego de contar con la salida del mapeo es posible dividir el trabajo en los distintos servidores de los que se dispongan, copiando cada uno de los resultados y realizando el trabajo en paralelo, ver figura 28.

Figura 28. **Mapeo/Reducción de forma distribuida**



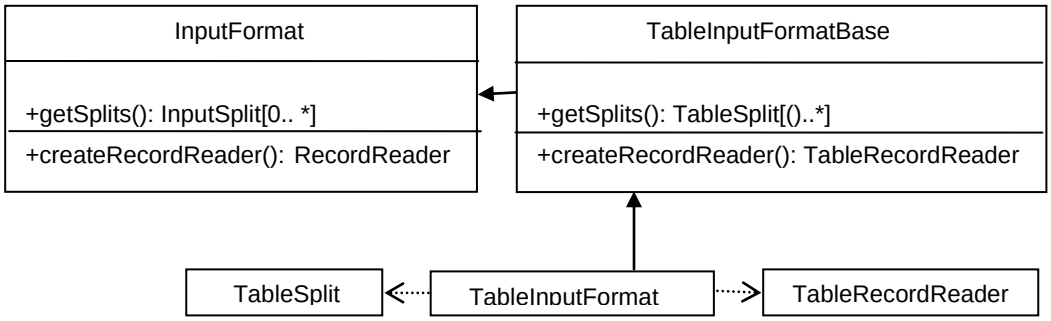

Fuente: WHITE, Tom. Hadoop: the definite guide. p 28-30.

### 2.7.1. Map/Reduce en HBase

En HBase existen implementadas clases para cada una de las fases del modelo, las cuales son formato de datos de entrada, mapeo, reducción y salida de datos.

La clase entrada de datos (InputFormat) es la encargada de obtener los datos de entrada, con esta se tienen las opciones de dividir los datos para seleccionar sólo la más importante, así como, definir cuál de las parejas es la llave y el valor. Esta clase ofrece el acceso a las distintas tablas que se encuentren creadas en HBase y un iterador para recorrerlas. Ver la definición de la clase en la figura 29.

Figura 29. **Jerarquía de clase InputFormat**

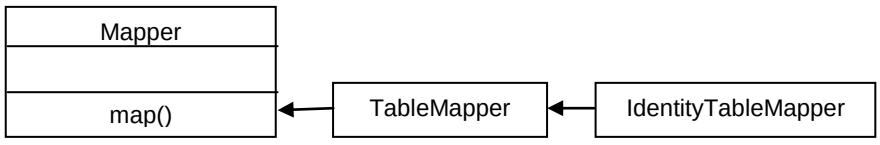



Fuente: LARSE, George. HBase: The definitive guide. p 290-291.

La clase mapeo (Mapper) implementa la segunda fase del modelo, la cual se encarga de leer cada uno de los pares de Llave/Valor para mapearlo según sea conveniente para los procesos posteriores.

La implementación de TableMapper es IdentityTableMapper la cual será de mucha ayuda para agregar la funcionalidad propia en los procesos.

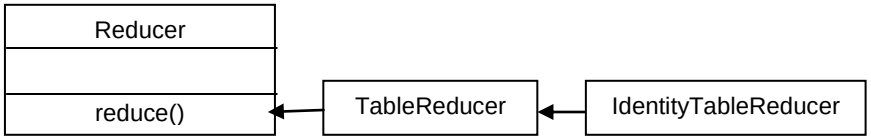
Figura 30. **Jerarquía de clase Mapper**




Fuente: LARSE, George. HBase: The definitive guide. p 291-292.

La clase reducción (Reducer) es similar a la clase Mapper con la única diferencia que la entrada de datos será la salida de la clase Mapper, teniendo los datos ya mapeados y ordenados para reducir y encontrar el requerimiento solicitado de forma más óptima.

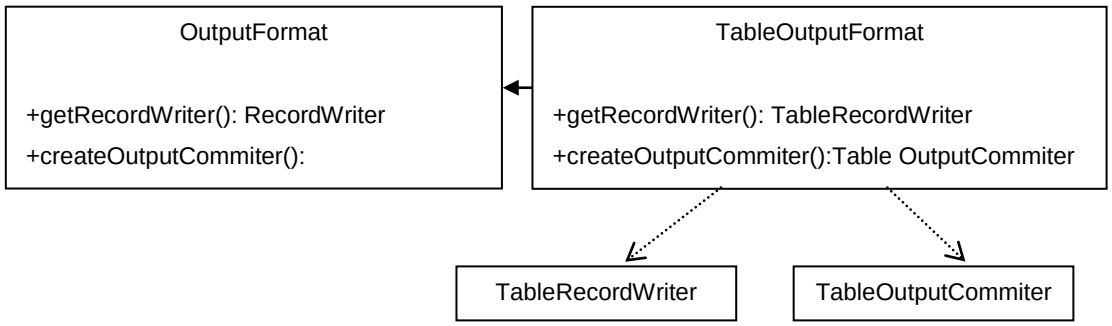
Figura 31. **Jerarquía de clase Reducer**




Fuente: LARSE, George. HBase: The definitive guide. p 292.

La clase salida de datos (OutputFormat) es la utilizada para el almacenamiento en alguna tabla o archivo, donde TableRecordWriter es la utilizada para almacenar la salida en una tabla.

Figura 32. **Jerarquía de clase OutputFormat**




Fuente: LARSE, George. HBase: The definitive guide. p 292-293.

# 3. GUÍA DE DESARROLLO DE IMPLEMENTACIÓN DE MODELO DE DATOS Y ARQUITECTURA PROPUESTA PARA AMBIENTE DATA WAREHOUSE

## 3.1. Requerimientos para la implementación de la arquitectura

Un clúster de Hadoop dadas las características que ofrece, tales como un sistema de archivos distribuido y la ejecución paralela de procesos entre los servidores, no es necesario realizar una inversión tan grande en cuanto a los servidores que conformarán el clúster. No es muy importante obtener recursos elevados en los servidores, como la cantidad de Cores y cantidad de procesadores, memoria RAM y cantidad de discos en cada uno de ellos.

Por otra parte no es rentable implementar un clúster Hadoop con recursos mínimos o de más baja calidad en los servidores, el riesgo con este tipo de equipo es que puede fallar de forma continua dada su calidad, lo cual puede que no sea mayor problema, pero el reemplazo de componentes en cuanto a costos, a largo plazo si serán afectados. Y por el hecho que se tengan fallas más a menudo, esto ocasionará que la información no esté siempre disponible y sean necesarios demasiados mantenimientos.

Servidores con demasiados recursos, esto afectará directamente la inversión del equipo, haciendo que se adquieran menos servidores con lo que se desaprovecharán las ventajas que ofrece Hadoop y en caso de fallos será mucho más costoso reemplazar el equipo defectuoso.

Por lo que adquirir equipo con un costo intermedio será la mejor opción, ya que se puede configurar mayor número de servidores dentro del clúster, con recursos relativamente intermedios en comparación con la tecnología de punta y contar de entre 4 a n nodos y así, aprovechar la eficiencia que ofrece Hadoop con el almacenamiento y procesamiento distribuido.

Los requerimientos recomendados para un servidor son los siguientes:

- 2 discos SATA con almacenamiento mínimo de 1 *terabyte*
- 2 Cpu's Dual Core de 2,5 GHz
- De 4 a 16 *gigabytes* de memoria RAM

Teniendo 5 servidores con estas características es posible implementar un clúster Hadoop en el cual se podrá procesar mucha información de forma eficiente, si los datos siguen aumentando o se necesita mejorar tiempos de respuesta o reducir carga en los servidores es posible agregar un nuevo nodo para mejorar el rendimiento.

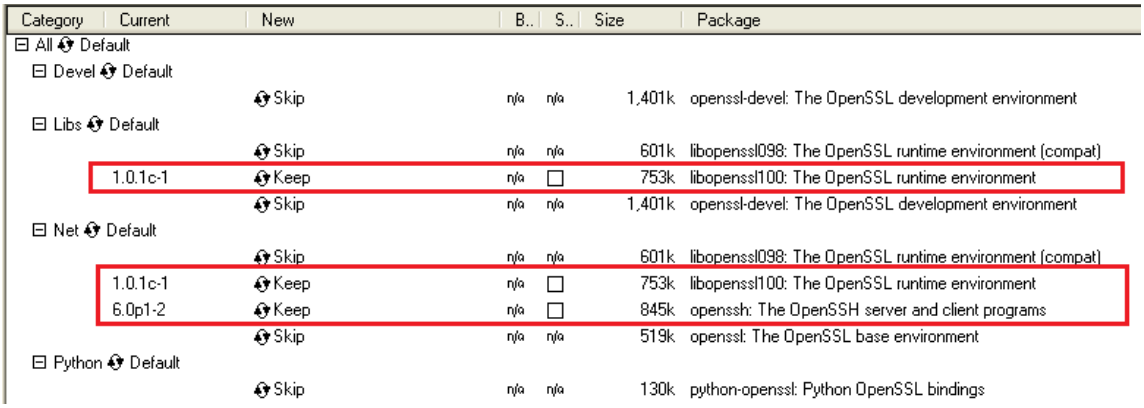
## 3.2. Configuración de arquitectura propuesta Hadoop

Como ejemplo de la configuración de un clúster Hadoop, se implementará una instalación Pseudodistribuida, lo cual significa que se ejecutarán los nodos en un único host, de manera de prueba y así tener el concepto de los archivos necesarios a configurar con Hadoop. Este modo de instalación solamente debe ser utilizado para pruebas y nunca para un ambiente productivo.

Para esta prueba se utilizará el Sistema Operativo Windows, por lo que a continuación se detallarán los prerrequisitos y las distintas configuraciones.

Dado que Hadoop ha sido desarrollado para sistemas UNIX es necesario instalar Cigwin el cual es un conjunto de paquetes UNIX para el Sistema Operativo Windows. Será necesario descargarlo e instalarlo. En la figura 33 se indican los paquetes OpenSSL y OpenSSH que deben ser seleccionados e instalados.

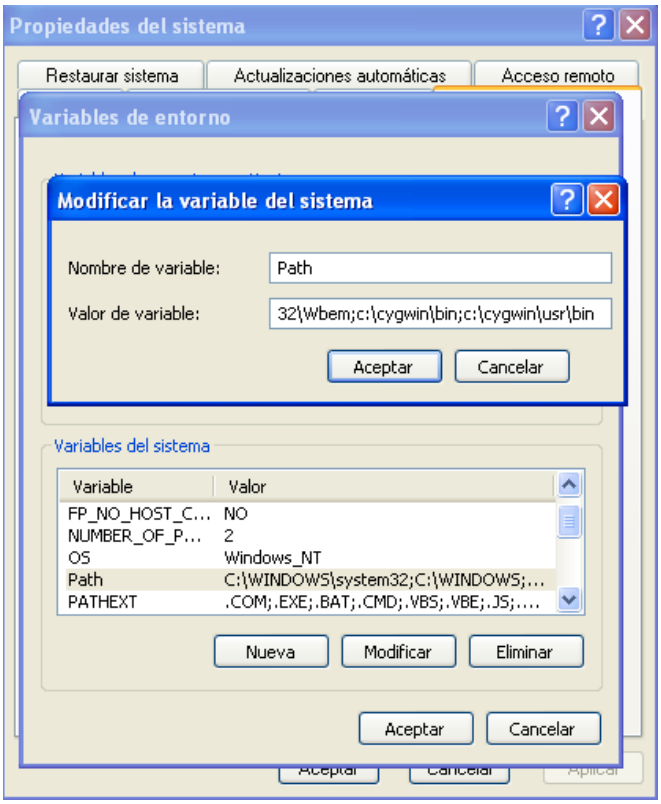
Figura 33. **Paquetes UNIX necesarios para instalación**




Fuente: elaboración propia.

Como segundo paso será necesario configurar las variables de entorno, para este caso en la variable Path se agrega la siguiente cadena C:\cygwin\bin;c:\cygwin\usr\bin.

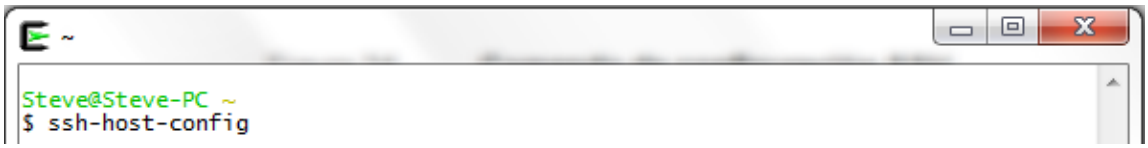
Figura 34. **Configuración de la variable de entorno PATH**




Fuente: elaboración propia.

Hadoop se basa con SSH para la comunicación y el lanzamiento de procesos remotos entre los nodos, por lo que es importante configurar los privilegios necesarios, esto lo proveerá Cygwin. Por lo que se debe abrir una consola Cygwin e ingresar el comando ssh-host-config.

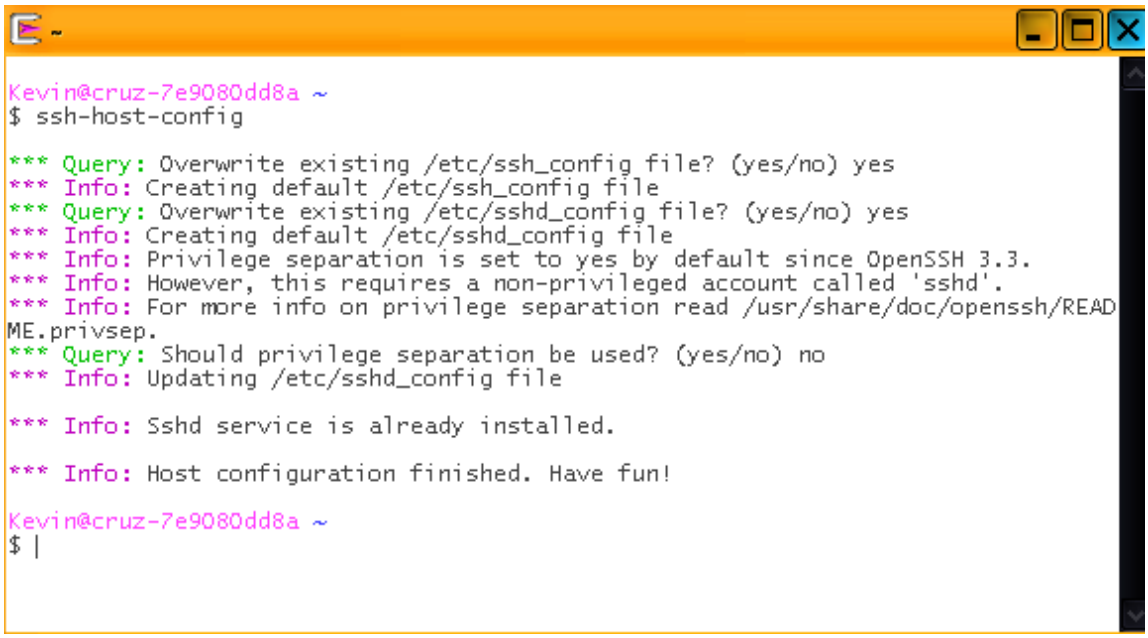
Figura 35. **Comando de configuración SSH**




Fuente: elaboración propia.

En la siguiente pantalla de configuración se consulta si se desea utilizar separación de privilegios, debe especificarse que no. Luego se requiere indicar si debe instalarse como un servicio, se define que sí. Luego se requiere ingresar el nombre de la variable de entorno de Cygwin, escribir la palabra ntsec.

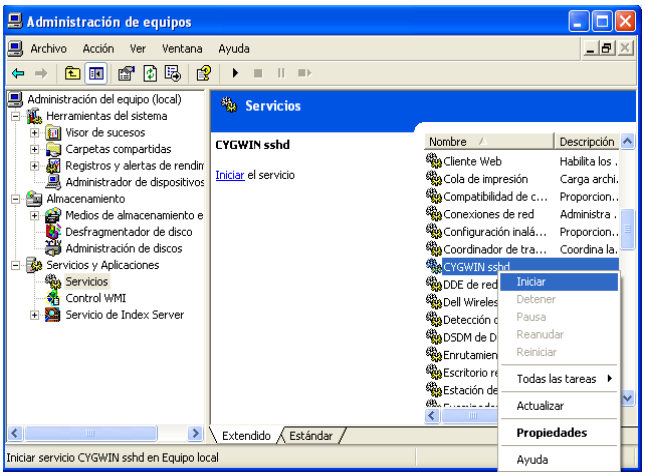
Figura 36. **Pantalla de configuración SSH**




Fuente: elaboración propia

Luego de finalizar satisfactoriamente la configuración, será necesario arrancar el servicio, por lo que debe dirigirse a la configuración del equipo y se selecciona Servicios, se debe buscar el servicio Cygwin y seleccionar Iniciar, será necesario esperar a que el servicio haya iniciado verificando que en la descripción muestre la palabra Iniciado.

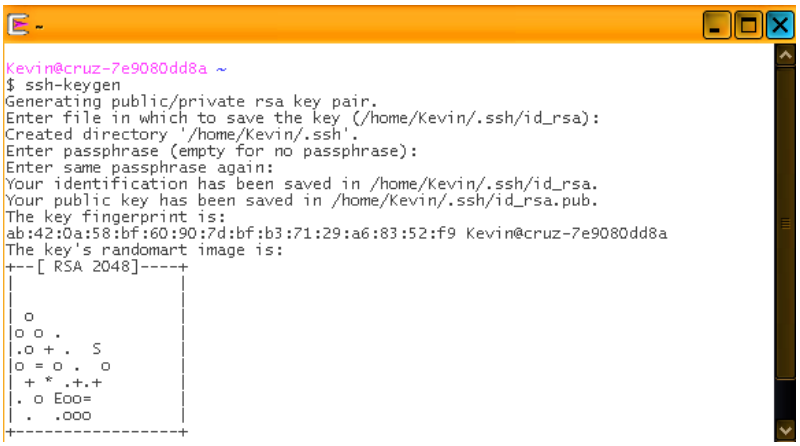
Figura 37. **Iniciando servicio SSH**




Fuente: elaboración propia.

Finalizada la configuración se procede a la generación de llaves de autorización entre los nodos. Para esto es necesario teclear el comando ssh-keygen en la consola de Cygwin, ver figura 38.

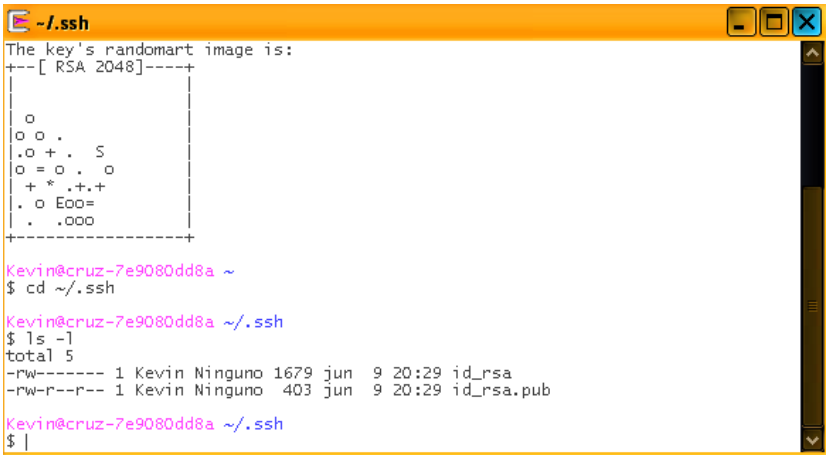
Figura 38. **Generación de llaves de autorización SSH**




Fuente: elaboración propia.

Con este comando se generan las llaves necesarias, para corroborarlo accesar a la carpeta con el comando cd ~/.ssh, luego listar los archivos con ls –l, con lo que la carpeta deberá contener los archivos id_rsa.pub y id_rsa.

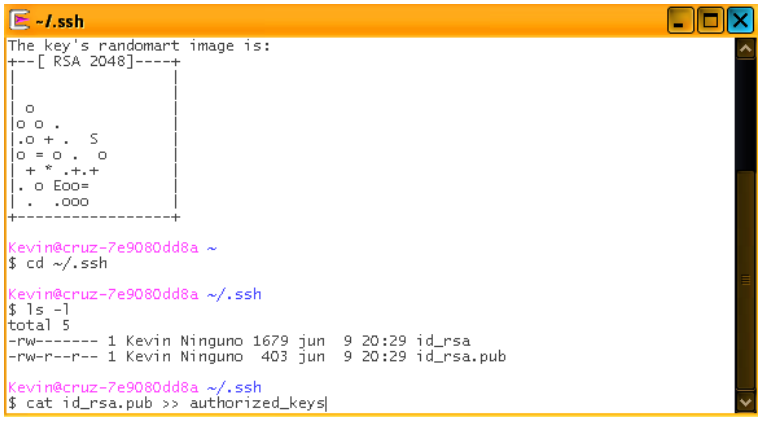
Figura 39. **Archivos de llaves de autorización SSH**




Fuente: elaboración propia.

Registrar el servidor, ejecutando: cat id_rsa.pub >> authorized_keys.

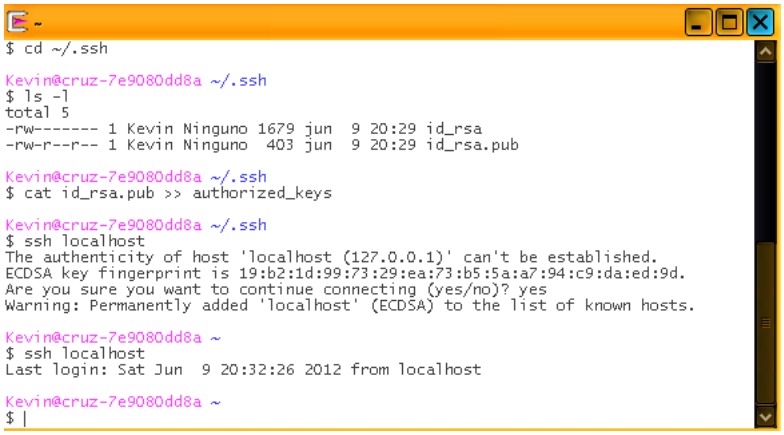
Figura 40. **Registro de llaves de autorización SSH**




Fuente: elaboración propia.

Finalmente probar la instalación, ejecutando el comando ssh localhost.

Figura 41. **Prueba de configuración SSH**




Fuente: elaboración propia.

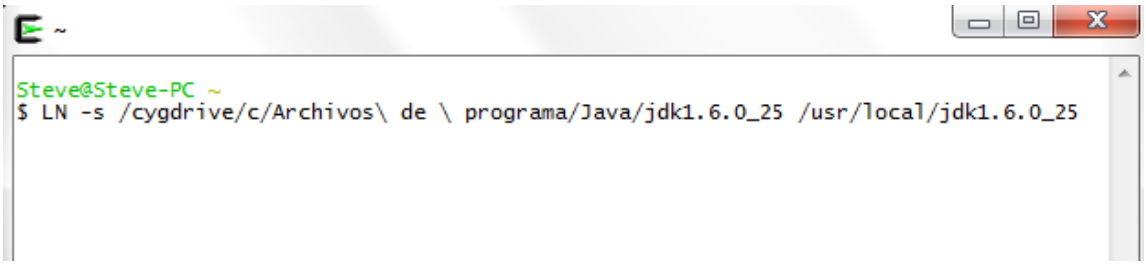
A continuación se procede a la instalación de Hadoop, para esto es necesario descargar desde el sitio oficial el archivo Hadoop-1.x.x.tar.gz el cual debe copiarse en la carpeta /home de la instalación de Cygwin. Descomprimir el archivo tar –xzf hadoop-1.0.3.tar.gz, con esto finaliza la instalación.

Figura 42. **Instalación de Hadoop**


Fuente: elaboración propia.

El otro importante requisito es la plataforma Java, descargar desde la página oficial el JDK 1.6. Finalizada la instalación es importante crear un *link* en la carpeta /usr/local al directorio Java para contar con el ambiente en Cygwin. Para esto ejecutar el comando LN -s /cygdrive/c/Program\ Files/Java/jdk1.6.x /usr/local/jdk1.6.x.

Figura 43. **Creación de *link* en carpeta root al directorio Java**




Fuente: elaboración propia.

Luego de haber instalado Hadoop iniciar con la configuración de la misma, para esto es necesario configurar los archivos core-site.xml, hdfs-site.xml, mapred-site.xml. Dentro de la carpeta /Hadoop-1.x.x/conf.

Estos archivos deben configurarse en formato XML, por lo que deben tomarse en cuenta qué valores son los necesarios en cada una de las propiedades del archivo de configuración.

Cada configuración debe especificarse dentro de la palabra clave configuration, donde cada propiedad se coloca dentro de la palabra properties, finalmente debe especificarse el nombre y valor de la configuración, utilizando la palabra name y value, ver figura 44.

Figura 44. **Configuración de archivos XML Hadoop**

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost/</value>
</property>
</configuration>
```

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:8021</value>
</property>
</configuration>
```

Fuente: elaboración propia.

De igual forma es posible configurar los nodos dentro del clúster en los archivos masters y slaves en la carpeta /Hadoop-1.x.x/conf y así indicar qué función cumplirá cada uno de los servidores.

Luego será necesario configurar la variable JAVA_HOME con el directorio de Java para que Hadoop utilice este ambiente, ver figura 45.

Figura 45. **Configuración de variable JAVA_HOME en Hadoop**

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.  Required.  /usr/lib/j2sdk1.5-sun
export JAVA_HOME=/usr/local/jdk1.6.x
```

Fuente: elaboración propia.

Finalmente es necesario dar formato al Namenode, con el siguiente comando bin/hadoop namenode –format.

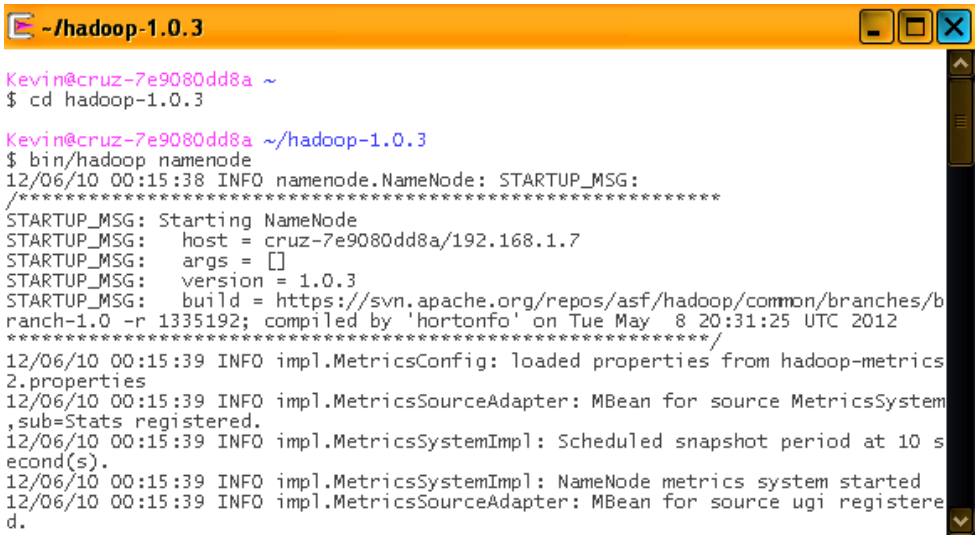
Figura 46. **Creación y formato HDFS**

```
~/hadoop-1.0.3
Kevin@cruz-7e9080dd8a ~
$ bin/hadoop namenode -format
-bash: bin/hadoop: No such file or directory

Kevin@cruz-7e9080dd8a ~
$ cd hadoop-1.0.3

Kevin@cruz-7e9080dd8a ~/hadoop-1.0.3
$ bin/hadoop namenode -format
12/06/10 00:13:49 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = cruz-7e9080dd8a/192.168.1.7
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 1.0.3
STARTUP_MSG:   build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.0 -r 1335192; compiled by 'hortonfo' on Tue May  8 20:31:25 UTC 2012
************************************************************/
12/06/10 00:13:50 INFO util.GSet: VM type       = 32-bit
12/06/10 00:13:50 INFO util.GSet: 2% max memory = 19.33375 MB
12/06/10 00:13:50 INFO util.GSet: capacity      = 2^22 = 4194304 entries
12/06/10 00:13:50 INFO util.GSet: recommended=4194304, actual=4194304
12/06/10 00:13:51 INFO namenode.FSNamesystem: fsOwner=Kevin
12/06/10 00:13:51 INFO namenode.FSNamesystem: supergroup=supergroup
12/06/10 00:13:51 INFO namenode.FSNamesystem: isPermissionEnabled=true
12/06/10 00:13:51 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
12/06/10 00:13:51 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessK
eyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
12/06/10 00:13:51 INFO namenode.NameNode: Caching file names occuring more than
10 times
12/06/10 00:13:52 INFO common.Storage: Image file of size 111 saved in 0 seconds
.
12/06/10 00:13:52 INFO common.Storage: Storage directory \tmp\hadoop-Kevin\dfs\n
ame has been successfully formatted.
12/06/10 00:13:52 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at cruz-7e9080dd8a/192.168.1.7
************************************************************/

Kevin@cruz-7e9080dd8a ~/hadoop-1.0.3
$
```

Fuente: elaboración propia.

Finalmente proceder a ejecutar los Daemons para iniciar el clúster. Iniciar el Namenode con el comando bin/hadoop namenode, este será el encargado de coordinar el almacenamiento distribuido dentro de los demás nodos.

Figura 47. **Ejecución de NameNode**

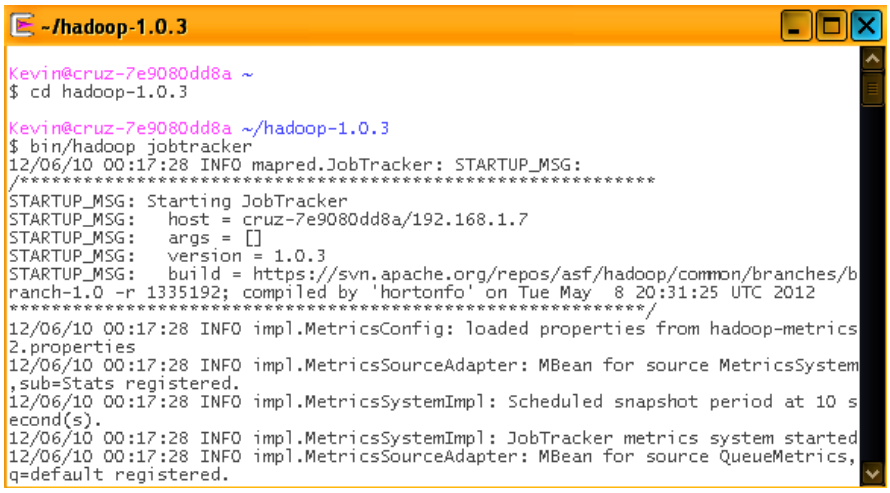



Fuente: elaboración propia.

Como segundo paso iniciar el JobTracker con el comando /bin/hadoop jobtracker. Este Daemon será el encargado de coordinar los procesos distribuidos dentro de los nodos.

Esto simulará la sincronización de tareas a realizar en cada uno de los Name Nodes para la ejecución paralela de los procesos.

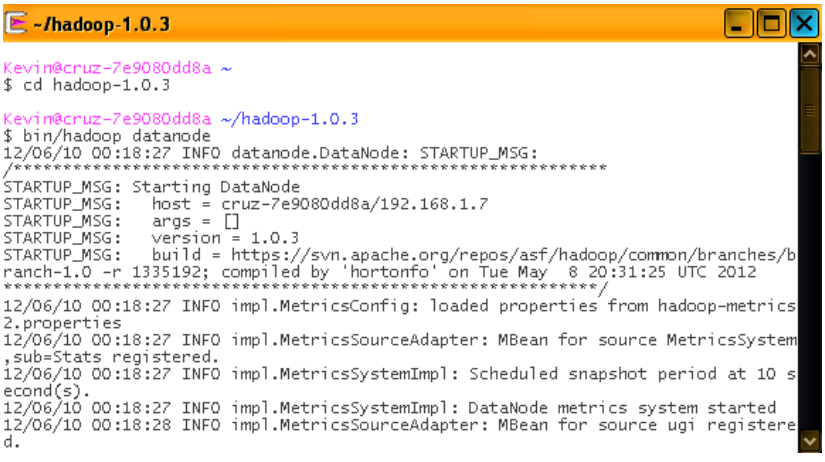
Figura 48. **Ejecución JobTracker dentro del NameNode**




Fuente: elaboración propia.

Iniciar los Datanodes, /bin/hadoop datanode. Estos almacenarán los datos de forma distribuida. Ejecutar según la cantidad de nodos en el clúster.

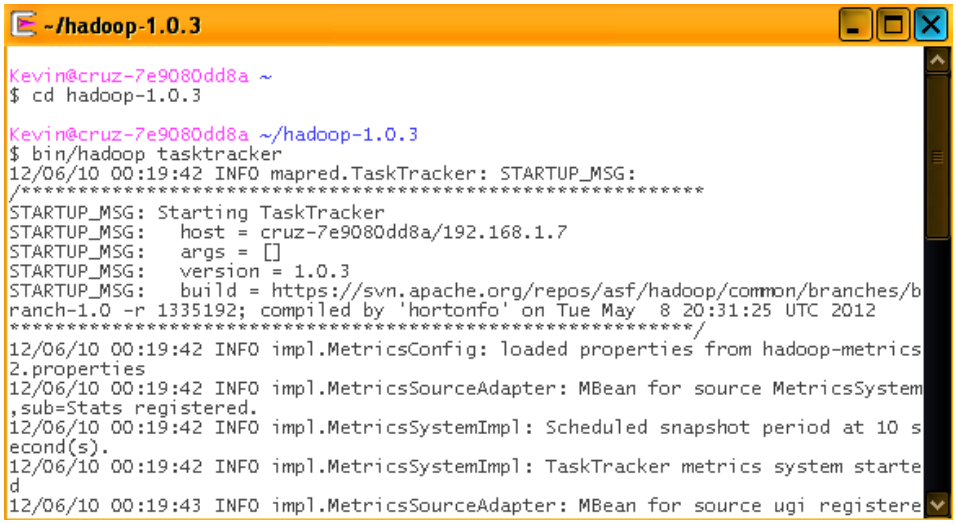
Figura 49. **Ejecución de DataNodes**




Fuente: elaboración propia.

Finalmente deberán ejecutarse los Daemons en cada Datanode del TaskTracker. /bin/hadoop tasktracker. Estos serán quienes ejecutarán los procesos paralelos de forma coordinada, reportando los resultados al JobTracker en el Namenode.

Figura 50. **Ejecución de TaskTracker en cada DataNode**




Fuente: elaboración propia.

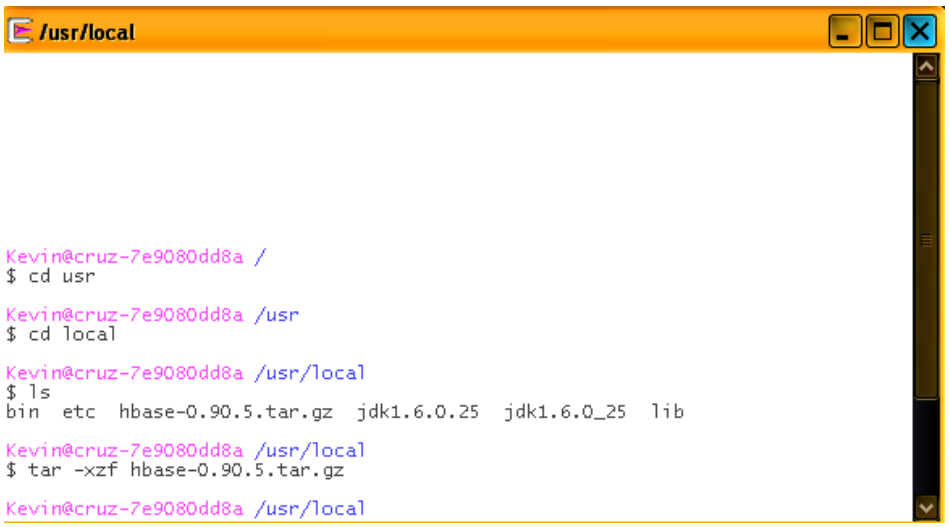
Con esto finaliza la configuración e inicialización de un clúster Hadoop funcional, estos son los archivos más importantes a tener en cuenta al implementarla en un sistema completamente distribuido.

### 3.3. Configuración e instalación de HBase

Para configurar la base de datos HBase los requerimientos serán los mismos, Java y Cygwin. En esta parte el enfoque estará en la configuración de HBase.

Para iniciar la instalación se necesita descargar la versión de Hbase-0.90.x en el sitio oficial de HBase. Habiendo finalizado la descarga se debe ir a la carpeta /usr/local y descomprimir el archivo.

Figura 51. **Instalación de HBase**




Fuente: elaboración propia.

Como siguiente paso será necesario dar los permisos a las carpetas /etc/password, /etc/group, /var, ver figura 52.

Figura 52. **Conceder permisos a carpetas utilizadas en HBase**


Fuente: elaboración propia.

El siguiente paso es corroborar la existencia de las siguientes líneas All : localhost 127.0.0.1/32 [::1] / 128 [::ffff:127.0.0.1] / 128 : allow y ademas se encuentre por encima de la línea All : PARANOID : deny. Realizar esta verificación en el archivo /etc/*host.allow*.

Figura 53. **Configuración archivo *host.allow***

```
1  #
2  # hosts.allow   This file describes the names of the hosts which are
3  #               allowed to use the local INET services, as decided
4  #               by the '/usr/sbin/tcpd' server.
5  #
6  #       CYGWIN note: if you use a software firewall (such
7  #       as ZoneAlarm or the "Windows Firewall" in Windows
8  #       XP), you must also open a 'hole' at the proper
9  #       port for the services you enable below.
10 #
11 ALL : localhost 127.0.0.1/32 [::1]/128 [::ffff:127.0.0.1]/128 : allow
12 ALL : PARANOID : deny
13 sshd: ALL : allow
```

Fuente: elaboración propia.

De igual forma que en Hadoop, es necesario configurar la variable JAVA_HOME en el archivo /conf/hbase_env.sh, la variable HBASE_IDENT_STRING debe tener el valor de $HOSTAME.

Figura 54. **Variable JAVA_HOME, HBASE_IDENT_STRING**

```
#A string representing this instance of hbase. $USER by default.
export HBASE_IDENT_STRING=$HOSTNAME

# The java implementation to use.  Required.  /usr/lib/j2sdk1.5-sun
export JAVA HOME=/usr/local/jdk1.6.x
```

Fuente: elaboración propia.

Es importante contar con la instalación del JDK 6, dado que al igual para la instalación de Hadoop, esta herramienta es basada principalmente en Java, haciendo que sea una herramienta transportable para ser ejecutada en cualquier Sistema Operativo.

Luego configurar el archivo /conf/*hbase-site.xml*. De igual forma como han sido configurados los archivos XML para Hadoop, siguiendo los mismos conceptos mencionados anteriormente, ver figura 55.

Figura 55. **Configuración archivo *hbase-site.xml***

```
<property>
  <name>hbase.rootdir</name>
  <value>file:///C:/cygwin/tmp/hbase/data</value>
  <description>The directory shared by region servers.
  Should be fully-qualified to include the filesystem to use.
  E.g: hdfs://NAMENODE_SERVER:PORT/HBASE_ROOTDIR
  </description>
</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>C:/cygwin/tmp/hbase/tmp</value>
  <description>Temporary directory on the local filesystem.</description>
</property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>127.0.0.1</value>
  <description>zookeper.</description>
</property>
```

Fuente: elaboración propia.

Es importante asegurarse que las carpetas contenidas en las propiedades hbase.rootdir y hbase.tmp.dir existan y tengan los permisos 777.

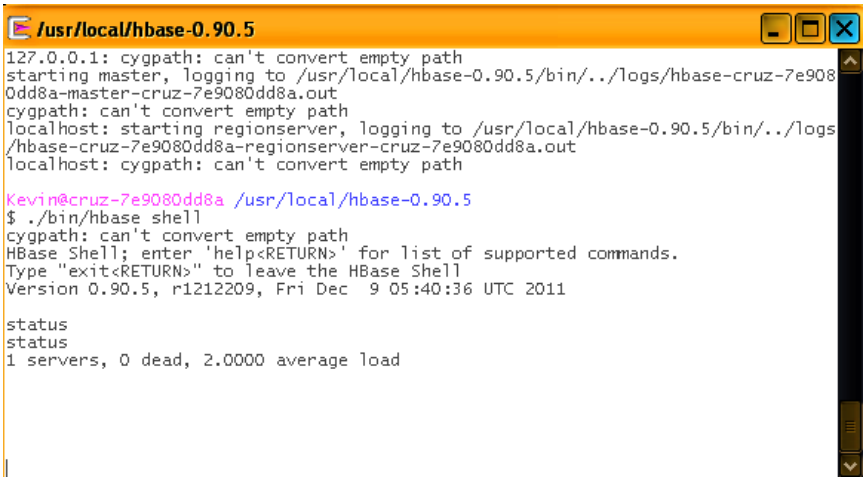
Finalmente se ha concluido con la configuración de HBase, con lo que se procede a arrancar la base de datos. Esto con el comando ./bin/start-hbase.sh.

Figura 56. **Arrancar base de datos HBase**


Fuente: elaboración propia.

Ya que ha iniciado el servicio, iniciar la consola de HBase para ingresar a la base de datos, con el comando ./bin/hbase Shell.

Figura 57. **Arrancar consola HBase**




Fuente: elaboración propia.

Con esto se tendrá una instancia de la base de datos HBase funcional como ambiente de pruebas y aprendizaje de esta tecnología.

## 3.4. Creación de Script de estructuras de columnas

A continuación se procederá con la creación del script de la base de datos propuesta, se aplicarán los comandos necesarios y la nomenclatura para la creación de tablas. Se definirán los comandos equivalentes a las operaciones DDL (Data Definition Language) más comúnmente utilizadas.

La sentencia para la creación de tablas inicia con la palabra clave create, seguido se define el nombre que tendrá la tabla. De manera opcional es posible definir de forma explícita cada una de las familias de columnas que conformarán la tabla, estas deberán ser definidas seguidas del nombre de la tabla.

create 'TABLA', ['FamiliaCol1', 'FamiliaCol2' ,… , 'FamiliaColn']

A continuación se presenta el script para la creación de tablas en la base de datos HBase.

Figura 58. **Script de creación de tablas en HBase**

```
1  --Creacion tabla usuario, familias de columna Credenciales, Contacto
2  create 'USUARIO','credenciales','contacto'
3
4  --Creacion tabla telefono, familias de columna Descripcion, Identificador, Producto, Operadora, Plan
5  create 'TELEFONO','descripcion','identificador','producto','operadora','plan'
6
7  --Creacion tabla cliente-telefono, familias de columna Relacion
8  create 'CLITEL','relacion'
9
10 --Creacion tabla trafico, familias de columna Consumido, Celda
11 create 'TRAFICO','consummido''celda'
12
13 --Creacion tabla recargas, familias de columna recargado, origen, promocion
14 create 'RECARGAS','recargado','origen','promocion'
```

Fuente: elaboración propia.

En el script se han definido las familias de columnas que formarán parte de cada una de las tablas, en el modelo de base de datos columnar se han definido también las columnas de cada una de estas, estas columnas se definen en la inserción de datos y no en la creación de tablas, esto se verá en el tema siguiente.

Finalmente se pueden observar las tablas creadas en la base de datos utilizando el comando list dentro de la consola de HBase.

Figura 59. **Comando list para visualizar tablas en HBase**


Fuente: elaboración propia.

Otra operación equivalente de DDL, es la palabra reservada alter la cual es la utilizada para modificar la estructura de las tablas. Para agregar una nueva columna a una tabla la sintaxis será la siguiente:

```
alter 'TABLA', {Name => 'FamiliaCol4' , Versions => N}
```

Para eliminar una familia de columnas:

```
alter 'TABLA', {Name => 'FamiliaCol4' , METHOD => 'delete'}
```

Si se desea eliminar una tabla, es necesario deshabilitarla, por lo que debe ejecutarse el comando disable.

```
disable 'TABLA'
```

Luego ya es posible proceder a la eliminación de la tabla con el comando drop.

```
drop 'TABLA'
```

Además para habilitar nuevamente una tabla con el comando enable.

```
enable 'TABLA'
```

Finalmente, si se desea comprobar la existencia de una tabla, será necesario ejecutar el comando exists.

```
exists 'TABLA'
```

## 3.5. Operaciones CRUD y consultas de datos

A continuación se definirá la forma en cómo los datos son creados, actualizados, eliminados y consultados en HBase, lo cual es equivalente a las operaciones DML (Data Manipulation Language).

Como primer punto se especificarán las operaciones de inserción, las cuales utilizan la palabra clave put. Con esta función se realizan dos operaciones: la inserción de los datos y la definición de las columnas en cada familia de columna, esto como se mencionó en el tema anterior es debido a que las columnas son creadas según los datos insertados en las tablas, ya que no se insertan valores nulos, únicamente si el dato existe es insertado, de lo contrario no ocupará espacio en el sistema de archivos.

En esta operación se debe indicar la tabla, la Llave/Valor de la celda, la familia de columna seguido por ":" luego se define la columna y finalmente se define el valor de la celda.

put 'TABLA', 'Llave/Valor', 'FamiliaCol:Columna', 'Valor '

A base de ejemplo se tiene una muestra de inserción de los datos, insertando datos de los usuarios y los teléfonos que cada uno de ellos utiliza, para comprender de mejor manera cómo funcionan estas operaciones en HBase, ver figura 60.

Figura 60. **Script inserción de datos en HBase**

```
put 'USUARIO', 'user1', 'credenciales:nombre', 'Juan'
put 'USUARIO', 'user1', 'credenciales:direccion', 'zona1'
put 'USUARIO', 'user1', 'contacto:email', 'abc@gmail.com'
put 'USUARIO', 'user2', 'credenciales:nombre', 'Pedro'
put 'USUARIO', 'user2', 'credenciales:direccion', 'Zona2'
put 'USUARIO', 'user2', 'contacto:tel_domicilio', '24324545'
...
put 'TELEFONO', '45491726', 'descripcion:color', 'Negro'
put 'TELEFONO', '45491726', 'identificador:imei', 'AB1212CD'
put 'TELEFONO', '45491726', 'operadora:descripcion', 'Claro'
put 'TELEFONO', '47383778', 'descripcion:color', 'Azul'
put 'TELEFONO', '47383778', 'descripcion:marca', 'Motorola'
put 'TELEFONO', '47383778', 'identificador:imei', 'BCC2324DD'
put 'TELEFONO', '47383778', 'producto:descripcion', 'Prepago'
put 'TELEFONO', '47383778', 'operadora:descripcion', 'Telefonica'
...
put 'CLITEL', '47383778user1', 'relacion:time', '11062012'
put 'CLITEL', '45491726user2', 'relacion:time', '11062012'
```

Fuente: elaboración propia.

La siguiente operación es get, la cual es utilizada, para la consulta de los datos en la base de datos, con esta se puede consultar con base en la tabla y a la Llave/Valor de la celda, de manera opcional es posible indicar la familia de columnas, columna o versión del dato.

get ‘TABLA’, ‘Llave/Valor’, [{COLUMN => ‘FamiliaCol[:Col]’, VERSION => versiones}]

Los datos almacenados, así como, las versiones son almacenados según el TIMESTAMP definido en cada tabla, ver figura 61.

Figura 61. **Resultado de la instrucción get en HBase**

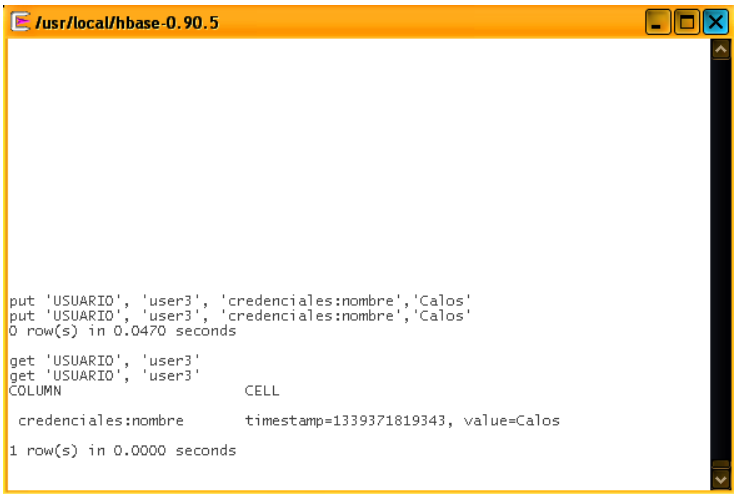

Fuente: elaboración propia.

Los datos han sido insertados en la tabla Usuario y es posible observar que los valores nulos no son insertados, si el dato no existe simplemente no existe.

La siguiente operación es la actualización, en este caso como se ha mencionado, un comando como tal no existe, es más la actualización no se realiza en HBase debido a los problemas que puede ocasionar en el rendimiento. Por lo que HBase maneja distintas versiones de datos lo cual consiste en almacenar distintas versiones de una celda.

A continuación se realiza la inserción de un usuario en la tabla Usuario, con Llave/Valor igual a user3 y valor de celda igual a 'Calos'.

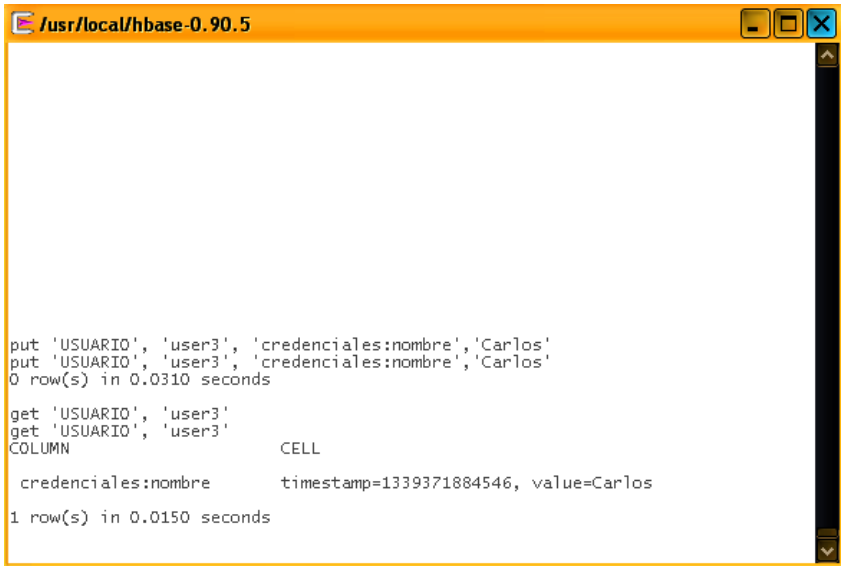
Figura 62. **Inserción con celda incorrecta en tabla Usuario**




Fuente: elaboración propia.

Ya que el valor de la celda es incorrecto es necesario modificarlo, para esto se realizará una nueva inserción, colocando el mismo valor de la Llave/Valor y colocando la celda de forma correcta.

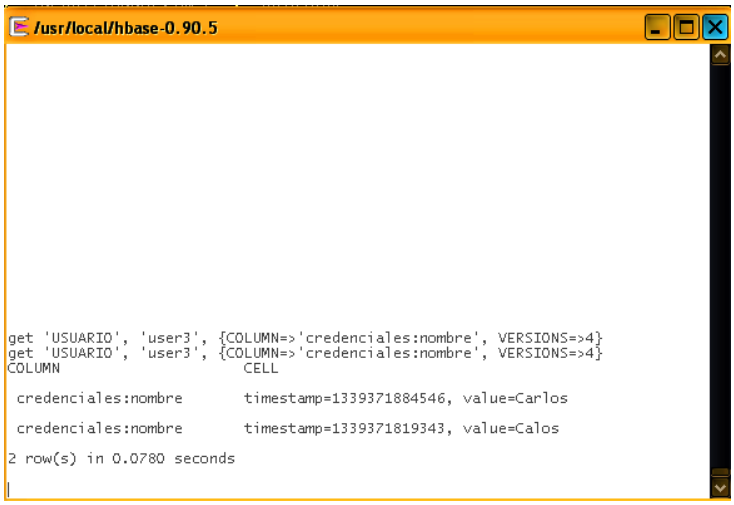
Figura 63. **Modificación de datos por versionamiento**




Fuente: elaboración propia.

Se inserta el valor ‘Carlos’ para el usuario user3, si se realiza un get sobre el usuario ‘user3’ y se observa que el valor ha sido modificado. Esto es debido a que las modificaciones o actualizaciones no existen en HBase, pues lo que hace esta base de datos es colocar versiones a las celdas.

Figura 64. **Operación get retornando versiones**



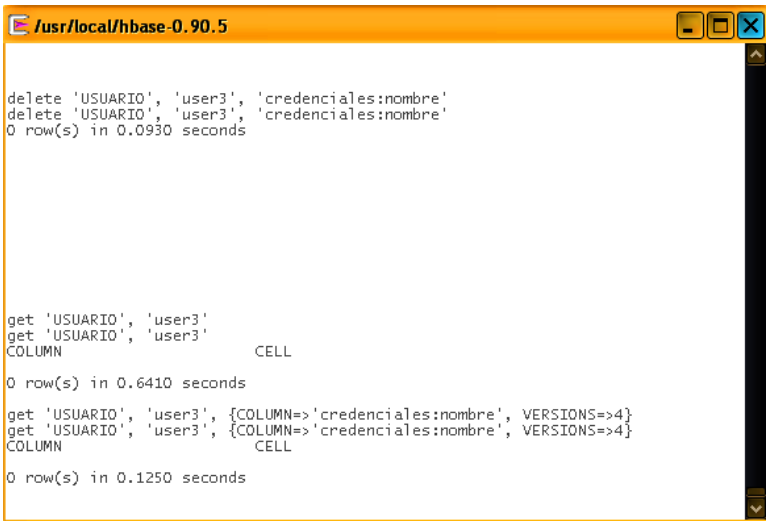

Fuente: elaboración propia.

Ahora al realizar un get al usuario ‘user3’ además de indicar las versiones que se desean observar para esa celda con la palabra clave Versions. Con esto es posible obtener los dos valores que han sido insertados para ese usuario. HBase únicamente retornará la última versión de la celda al momento de no especificar el número de versiones, por lo que, esta base de datos maneja las actualizaciones de una forma muy particular y no a como se acostumbra en las RDBMS.

Finalmente se tiene la operación de eliminación, la cual la palabra clave es delete, esta operación recibe el nombre de la tabla, la Llave/Valor y la columna para eliminar la celda. Esta operación de igual forma eliminará todas las versiones que existan sobre esta celda.

delete ‘TABLA’, ‘Llave/Valor’, ‘FamiliaCol[:columna]’

Figura 65. **Eliminación de datos en HBase**




Fuente: elaboración propia.

Con esto se definen las operaciones básicas en una base de datos para dar mantenimiento a la información.

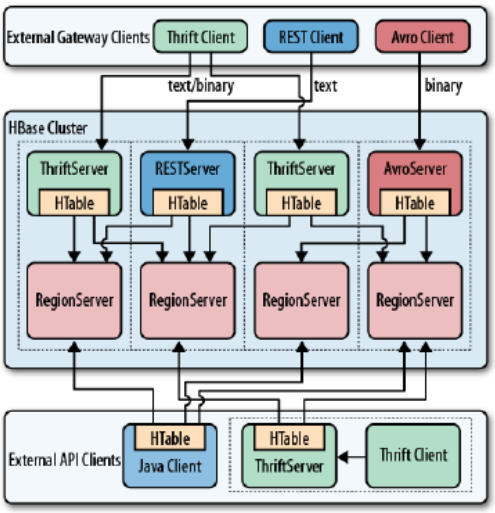
## 3.6. Interfaz Thrift

El siguiente aspecto a definir es la especificación de cómo o con qué herramientas un usuario interactúa con los datos contenidos en Hadoop. Acá es donde entra en escena Thrift, la cual es un conjunto de librerías y herramientas para los desarrolladores para crear servicios con los cuales los usuarios interactuarán con los datos contenidos en HBase. Esta interfaz es principalmente una puerta de enlace entre los usuarios y el servidor donde los procesos serán ejecutados, esta da la posibilidad de crear comunicaciones escalables y eficientes acorde al funcionamiento de Hadoop, permitiendo el llamado remoto de procedimientos.

Apache Thrift de igual forma que los anteriores proyectos analizados, está almacenado en el mundo de Apache. Esta interfaz da la posibilidad de crear interfaces en diferentes lenguajes de programación, actualmente soporta C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk y OCaml. Toda esta diversidad de lenguajes es posible desarrollar interfaces para el usuario según los gustos de los desarrolladores y Thrift será la puerta de enlace entre estas interfaces y HBase.

Thrift se encontrará alojado en un servidor cliente o incluso puede alojarse en cada cliente que accederá a los datos en el servidor, esto depende de cómo se desarrollen las interfaces y cómo estará configurada la arquitectura, en la figura 66, se especifica un diagrama general de cómo Thrift funciona con HBase.

Figura 66. **Diagrama general interfaz Thrift**




Fuente: LARSE, George. HBase: The definitive guide. p 241-244.

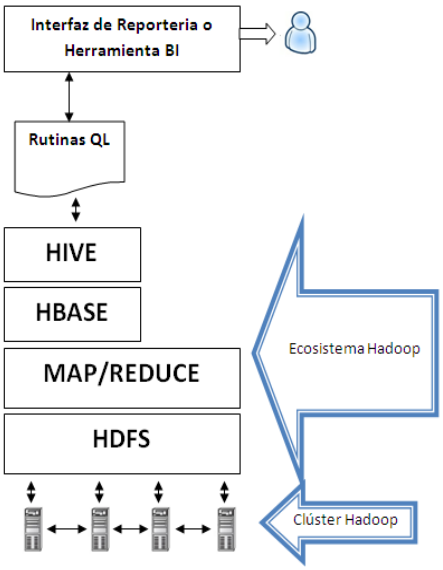
## 3.7. Hive herramienta para Data Warehouse

En el momento que se tiene la información almacenada y estructurada según el modelo de datos columnar, siempre es necesario para las organizaciones el análisis de las mismas para la toma de decisiones. Esta parte es siempre muy importante debido a que con esta información se diseña un camino a seguir el cual definirá el éxito de una organización.

Para esto se presenta el proyecto Hive, la cual es una infraestructura de Data Warehouse, que se aloja por encima de los demás proyectos Hadoop. Esta provee las herramientas necesarias para la Extracción, Transformación y Carga de datos (ETL), a manera de estructurar la información para un fácil y eficiente análisis de datos sobre la tecnología Hadoop.

Hadoop puede ser implementado para un ambiente de Data Warehouse, en donde la arquitectura está basada con un clúster Hadoop, los datos son almacenados de forma distribuida con el HDFS, con rutinas Map/Reduce y la base de datos HBase, sobre estas está implementado Hive la cual será la herramienta de extracción de datos más importantes para el análisis, los cuales pueden ser consumidos por cualquier otra herramienta para ambientes de reportes o de Business Intelligence por los usuarios finales.

Figura 67. **Hive herramienta de análisis de datos**




Fuente: elaboración propia.

Esta herramienta es muy popular debido a su parecido con el lenguaje SQL, al decir parecido quiere decir, que se utilizan algunas sentencias o palabras claves más utilizadas en los sistemas RDBMS, pero Hive provee las herramientas para que las consultas utilicen el modelo Map/Reduce y sean ejecutadas en subtareas en el clúster Hadoop. El lenguaje que provee Hive es popularmente llamado QL o HIVEQL.

Además, es importante dejar muy en claro que una de las metas de Hive no es alcanzar tiempos bajos de latencia en las distintas consultas realizadas por los usuarios, no como sucede en las distintas herramientas tales como Oracle, MySQL, etcétera.

La cantidad de datos a analizar son relativamente pequeñas. Más bien se trata de generar rutinas con las consultas deseadas utilizando QL y lanzandolas para que sean ejecutadas dentro del clúster, esto debido a que se espera analizar *terabytes* de datos lo cual puede tomar varios minitos e incluso horas, lo que en las anteriores herramientas puede tomar hasta días.

Por lo que lograr pequeños tiempos de latencia no es una prioridad, si no más bien aprovechar la escalabilidad, extensibilidad (Map/Reduce), tolerancia a fallos, que son las características principales del ecosistema Hadoop.

A continuación se describirán algunas de las sentencias más utilizadas, las cuales serán útiles para la implementación de un ambiente de Data Warehouse.

Hive permitirá crear estructuras de los datos que se tienen creados en HBase, para esto se utiliza la sentencia Create.

```
CREATE TABLE 'Nombre_tabla'
(nombre_columna1 tipo, nombre_columna2 tipo, …..)
Partitioned by nombre_columna1;
```

Esta sería la sintaxis para la creación de una tabla de forma general en la cual se especifica una estructura en la que los usuarios están más familiarizados para almacenar los datos que se deseen importar de HBase.

La sentencia Select es similar a las herramientas de SQL, su sintaxis sería.

```
FROM     Tabla
SELECT Columna1, Columna2, …, ColumnaN
```

Esta sería la forma más simple y primitiva de consultar datos en Hive, además es posible incorporar cualquier rutina o Script Map/Reduce, para obtener un resultado de forma más optimizada, estas son las ventajas que ofrece Hive sobre cualquier otra herramienta en el mercado. Para la importación de datos al Data Warehouse es posible realzarla de dos formas: utilizando la sentencia Insert Overwrite.

```
INSERT OVERWRITE TABLE Tabla_destino
SELECT Val1, Val2, … ValN
FROM FUENTE
```

Donde la palabra Overwrite es obligatoria e indica que los datos sobre la consulta serán sobre escritos en la tabla destino o partición de la tabla destino, la fuente será una tabla o un archivo, según sea necesario.

Hive provee las herramientas para el uso e implementación de Joins en las consultas, como se ha comentado en un tema anterior en las bases de datos no existen los Joins, lo cual indica que es necesario crear rutinas propias para realizar estas operaciones, normalmente utilizando rutinas Map/Reduce, Hive facilita esta tarea y ya implementa esta sentencia la cual está basada en rutinas Map/Reduce las cuales son ejecutadas a través de los Task Tracker dentro del clúster, lo cual brindará mejores tiempos de respuesta.

Las sentencias Join que provee Hive son limitadas en donde únicamente se cuenta con las opciones de Inner Join y Outer Join, el concepto es el mismo que en las demás bases de datos.

```
SELECT Col1, Col2, …, ColN
FROM Tabla1 JOIN Tabla2 ON (ID1 = ID2)
```

```
SELECT Col1, Col2, …, ColN
FROM Tabla1 [ LEFT | RIGHT | FULL ] JOIN Tabla2 ON (ID1 = ID2)
```

Hive no soporta la sentencia IN con lo que se utiliza comúnmente lo llamado Left Semi Join el cual es el equivalente a la subconsulta mencionada anteriormente, la sintaxis sería.

```
SELECT Col1, Col2, …, ColN
FROM Tabla1 LEFT SEMI JOIN Tabla2 ON (ID1 = ID2)
```

Lo cual significa que la tabla de la derecha de la consulta solamente aparecerá cuando la condición se cumpla.

Finalmente Hive provee los Map Join los cuales son utilizados normalmente cuando las tablas a consultar son pequeñas en datos, lo cual los cargará en memoria y realizará el mapeo de los datos, en este caso no habrá reducción, lo cual indica que las sentencias Outer Join no funcionarán con el Map Join.

```
SELECT /*+ MAPJOIN(Col1)*/ Col1, Col2, …, ColN
FROM Tabla1 JOIN Tabla2 ON (ID1 = ID2)
```

Con esta herramienta ya será posible crear y poblar el Data Warehouse, importando la información más importante de la base de datos para su análisis, así como, definir una estructura a estos. Así será posible crear rutinas con un lenguaje familiar al que se ha trabajado en el tiempo, el cual es un problema muy común, la resistencia al cambio.

Facilitando el análisis de datos a los usuarios y estos podrán visualizarlos en las distintas herramientas en el mercado o creadas por uno mismo, para aplicar reportes de Business Intelligence y así tomar las mejores decisiones para las organizaciones.

# 4. MEJORES PRÁCTICAS DE TUNNING PARA BASE DE DATOS COLUMNAR

## 4.1. Metas de Tunning

La tecnología Hadoop y el análisis de los datos se encuentra completamente reforzada por el modelo Map/Reduce el cual es de ayuda con el procesamiento de datos, pero además, son importantes algunas configuraciones con las cuales el rendimiento del clúster puede verse afectado de forma positiva, pero esto depende principalmente de la cantidad de datos que maneje la organización, así como, la cantidad de nodos en el clúster. En este tema se tratará de proporcionar las mejores prácticas, las cuales no significa que deben tomarse como base, sino más bien, como algunas recomendaciones importantes.

El primer aspecto a tomar en cuenta es el reducir latencia, la cual se define como el tiempo que le toma a un proceso finalizar una tarea específica, esta latencia normalmente se ve afectada por el algoritmo que se construya para dicho proceso, el cual debe ajustarse al modelo Map/Reduce para obtener mejores resultados. La ejecución de los procesos de forma paralela de igual manera será de mucha ayuda para reducir la latencia, más si los datos a procesar son excesivamente grandes.

El segundo aspecto a tomar en cuenta será la cantidad de datos la cual será manipulada en un proceso, mientras la cantidad de datos sea mayor, el tiempo de procesamiento se verá directamente afectado. En los sistemas comunes el tiempo que tome procesar cierta cantidad de datos dependerá de los recursos del servidor, tales como la memoria RAM, la cantidad y potencia de los CPUs, mientras más sean los datos a procesar mayor deben ser los recursos, pero estos aunque sean demasiado potentes tienen un límite.

Hadoop reduce estos problemas de forma considerable, dado a que escala de forma horizontal, el agregar un nodo al clúster se tendrá más espacio de almacenamiento y un nodo más que pueda ejecutar subtareas paralelas y así reducir los tiempos de latencia.

Esto quiere decir, que mientras la cantidad de datos aumente o si se desea reducir tiempos de latencia, deberá agregarse un nodo extra, esto debido a que un nodo puede procesar una cantidad x de *megabytes*, lo que significa que la escalabilidad se comporta de forma linear.

## 4.2. Tunning Garbage Collector

Dado que Hadoop es ejecutado sobre Java, es muy importante tomar en cuenta el Garbage Collector, el cual es el encargado de limpiar la memoria que es utilizada por las aplicaciones Java y así desechar todo lo que este ocupando espacio en memoria y se tenga disponible para el alojamiento de otros datos necesarios.

Por lo que este será el tuneo de más bajo nivel en el clúster, este debe ser principalmente tomado en cuenta para los nodos esclavos, los cuales son los encargados del almacenamiento de datos, por lo que el manejo de la memoria de cada uno de ellos debe ser lo más óptimo posible.

hbase.hregion.memstore.flush.size

Esta variable es importante debido a que mientras los objetos van siendo creados en memoria, estos se van acumulando, por lo que al llegar al tamaño mínimo esta será enviada al disco, por lo que es necesario configurar este valor. Al ir realizando estas operaciones se genera una fragmentación en memoria.

Otro aspecto a tomar en cuenta es lo que se llama Young Generation y Old Generation, lo cual significa que mientras los datos son alojados en memoria y escritos rápidamente en disco es a lo que se le llama Young Generation, mientras que al alojar datos en memoria y tome más tiempo la escritura en disco es a lo que se llama Old Generation, la cual es debida a que los datos procesándose son demasiado grandes.

Normalmente la Young Generation corresponde a un tamaño de 128 a 512 *megabytes*, por lo que es recomendado configurar la variable:

–Xmn128m

Esto significa que estos serán los espacios que quedarán en memoria, tendrán un valor mínimo de 128 *megabytes*. Es importante configurar este valor según, las necesidades ya que si es configurado demasiado pequeño, generará fragmentación en la memoria de forma más rápida, por otro lado si es demasiado grande creará demasiadas pausas en el proceso, lo cual afectará la latencia.

Los espacios que generan fragmentación deben ser mantenidos por el Garbage Collector, por lo que este al tratar de alojar información en memoria y cuando esta no se ajusta al espacio o espacios dejados por la fragmentación, deberá realizar una compactación de la memoria, lo cual generará el llamado de procesos propios del Garbage Collector para realizar la limpieza.

Por lo que es importante agregar a la configuración las siguientes variables:

–XX:+UseParNewGC -XX:+UseConcMarkSweepGC

La primera parte de la configuración significa que el Garbage Collector detendrá los procesos Java para limpiar la memoria fragmentada, si no es demasiada la fragmentación esto tomará muy poco tiempo. Pero al momento de que se presente la Old Generation tardará más tiempo la desfragmentación, lo cual puede generar errores de Time Out en los procesos del servidor, con la segunda configuración se le indica al Garbage Collector que no detenga el proceso Java, sino más bien lo siga ejecutando mientras desfragmenta la memoria, esto conllevará más carga para el procesador, pero es mejor esto a que se detengan por completo los procesos.

Por lo que es posible tunear la arquitectura a más bajo nivel, realizando las configuraciones básicas de la tecnología Java y una de las más importantes como lo es el Garbage Collector.

## 4.3. Particionamiento

El particionamiento en Hadoop tiene distintas variantes a comparación de como es conocido comúnmente en las bases de datos como Oracle, donde el particionamiento es específicamente para una tabla o índices, las cuales ayudan hasta cierto punto en el rendimiento, pero las cuales pueden llegar a ser demasiado complejas para los usuarios y requieren un mayor análisis y una variedad de estrategias para obtener los mejores resultados.

En Hadoop es posible identificar como primer particionamiento los procesos de Map/Reduce, un proceso a ejecutar será divido en subprocesos los cuales serán enviados a cada Task Tracker de los nodos, para que se ejecuten de forma paralela. Esto es realizado separando los datos a procesar por la Llave/Valor de cada registro, esto significa que el Job Tracker del nodo maestro agrupará los valores por la Llave/Valor y los asignará a cada nodo para su ejecución, esto puede afectar la latencia, debido a que la carga de procesamiento no se balancee de forma adecuada, con lo que es sumamente recomendado utilizar la interfaz llamada Partitioner la cual indicará la partición a la cual se está enviando el subproceso, para tener un control del procesamiento paralelo.

El segundo tipo de particionamiento tiene que ver con HBase, las tablas son particionadas automáticamente de forma horizontal, las cuales son llamadas Regiones quienes contienen los datos de la tabla, mientras más datos son almacenados las regiones irán incrementando y cuando sobrepasen el máximo configurado se generará una nueva región la cual contendrá el mismo tamaño configurado.

Con esto ya no se preocupa el usuario en realizar complicadas particiones en las tablas, más bien son manejadas por Hadoop de forma distribuida simplificando el mantenimiento de las mismas. Aún así, es importante tener en cuenta que al momento de eliminar datos en las tablas, algunas de las regiones tendrán almacenados menos cantidad de datos que otras, es recomendable realizar una unión de regiones para mantener la paridad.

Para esto HBase proporciona una herramienta para realizar estas uniones de regiones, la cual debe ser ejecutada en modo fuera de línea:

```
./bin/hbase org.apache.hadoop.hbase.util.Merge
```

Con esto las regiones distribuidas tendrán el mismo tamaño de datos, por lo que el intercambio entre nodos será igual de eficiente todo el tiempo.

Estos son los principales aspectos a tomar en cuenta para el particionamiento en la tecnología Hadoop, lo cual ayuda a reducir los tiempos de latencia y mantener la eficiencia en cada nodo del clúster.

## 4.4. Balanceo de carga

Así como es necesaria la unión de las regiones, de igual forma es importante mantener balanceada la cantidad de regiones entre los nodos. Esto lo realiza HBase ya que cuenta con un balanceador el cual es ejecutado cada cierto período de tiempo. Esto puede ser modificado de acuerdo con las distintas necesidades:

hbase.balancer.period
hbase.balancer.max.balancing

Con la primer variable se configura el período de tiempo que transcurrirá para que el balanceador inicie sus rutinas de balanceo, el segundo indica el tiempo máximo que pueden tardar estas rutinas, ya que según el tamaño de las regiones este tiempo puede variar, por lo que es importante tunear estas variables para no afectar el rendimiento del clúster debido a los movimientos de las regiones.

Esto puede crear problemas en ambientes productivos ya que las regiones o datos estarán en constantes movimientos, por lo que es posible desactivar el balanceo automático y realizarlo de forma implícita, haciendo uno mismo los llamados y ejecución de las rutinas de balanceo, para darle el mantenimiento a las regiones, en ventanas de tiempo de tal forma que sea completamente transparente para el usuario.

## 4.5. Tunning Map/Reduce

El objetivo de Map/Reduce es el procesamiento de grandes cantidades de datos, por lo que se pensaría que no es necesario realizar ningún tipo de tuneo en los procesos, lo cual es correcto debido a que precisamente Map/Reduce tiene como objetivo el analizar mucha información con una baja latencia.

Además se debe tomar en cuenta que en un clúster Hadoop cada Task Tracker realiza un subproceso con lo que estos mantienen una constante comunicación entre ellos, en donde se comparten o replican la información, por lo que es importante tomar en cuenta que puede existir una sobrecarga en las comunicaciones entre los nodos, dado que en efecto se procesarán grandes cantidades de datos.

### 4.5.1. Compresión

En cuanto a la compresión se hace referencia a las herramientas que existen en el mercado, las cuales codifican la información para reducir el espacio que ocupan en el disco, esto mejorará el rendimiento en la comunicación de los nodos ya que será menos cantidad de datos los que serán compartidos en el clúster.

Esto ayudará en cuanto a la capacidad de los discos duros que disponga el clúster y así utilizar de forma óptima los recursos.

Los aspectos a tomar en cuenta al momento de elegir la herramienta más óptima son los siguientes:

- La carga o tiempo que le puede llevar al CPU realizar la descompresión de los datos afectara el rendimiento.

- La capacidad de descomprimir en forma paralela, esto quiere decir, si es necesario realizar una descompresión antes de enviar los datos a cada nodo para su procesamiento, esto afectará el proceso paralelo ya que un nodo deberá encargarse de descomprimir los datos.

Con estos dos aspectos se puede decir que la herramienta más óptima que podrá mejorar el rendimiento en el clúster, deberá tener la capacidad de dividir los datos comprimidos y consumirlos en tiempo real por cada nodo, de forma que la entrada y salida de datos sean siempre datos comprimidos. Una herramienta que se recomienda dadas sus características es LZO la cual se basa en un algoritmo para comprimir datos en tiempo real.

En HBase es posible comprimir la información a nivel de columnas, con lo que es otra característica que ayudará a mejorar el rendimiento y reducir la latencia para el procesamiento de datos.

### 4.5.2. Tamaño del archivo de bloque

Hadoop maneja un tamaño de bloque estándar de 64 *megabytes*. Por lo que se debe tomar en cuenta la cantidad de subtareas necesarias para completar un proceso según la cantidad de datos que maneje el clúster.

Por ejemplo, si dentro del clúster se manejan cantidades de datos tales como 20 *gigabytes*, será necesario realizar un cálculo de las subtareas que se realizarán con la siguiente fórmula:

(20 * 1024) / 64 = 320

Esto significa que se realizarán 320 subtareas a procesar dentro del clúster. Con esto se concluye que acorde al número de nodos que contenga el clúster deberá tunearse el tamaño de bloque, para que las 320 subtareas sean distribuidas de forma uniforme el balanceo de carga para cada nodo y obtener el mejor rendimiento, de manera que todos los nodos sean aprovechados dentro del clúster. Por lo que mientras menos nodos existan en el clúster será necesario aumentar el tamaño de bloque, al contrario si se tienen muchos nodos deberá ser menor y así utilizar todos los nodos y balancear la carga de forma uniforme.

### 4.5.3. Copia paralela

En el modelo Map/Reduce al momento que los nodos finalizan el mapeo de datos, estos son copiados a cada reductor del clúster.

Con lo que al momento de que las salidas del mapeo sean demasiado grandes, será necesario trabajar de forma paralela, la desventaja que traerá esto, es que incrementará el uso del CPU, pero la principal ventaja será la baja latencia que se obtendrá con esta configuración.

La variable necesaria a configurar será: mapred.reduce.parallel.copies

Esta variable por defecto tiene un valor de 5, este debe modificarse según el tamaño del clúster, así como, la capacidad del CPU de los nodos.

## 4.6. Conclusiones de las prácticas de Tunning en Hadoop

En las distintas prácticas de tuneo se han especificado principalmente las configuraciones de los distintos nodos del clúster, donde la comunicación entre ellos es muy importante, así como, el análisis del funcionamiento de Map/Reduce, para optimizar el clúster.

No se ha enfocado el tuneo en el diseño ya que por defecto, la propuesta del modelo relacional y modelo columnar es una desnormalización en donde los conceptos son los mismos a aplicar como en cualquier otro modelo existente.

# 5. EL FUTURO DE DATA WAREHOUSE

## 5.1. Comparativo Data Warehouse basado en columnas con la actualidad

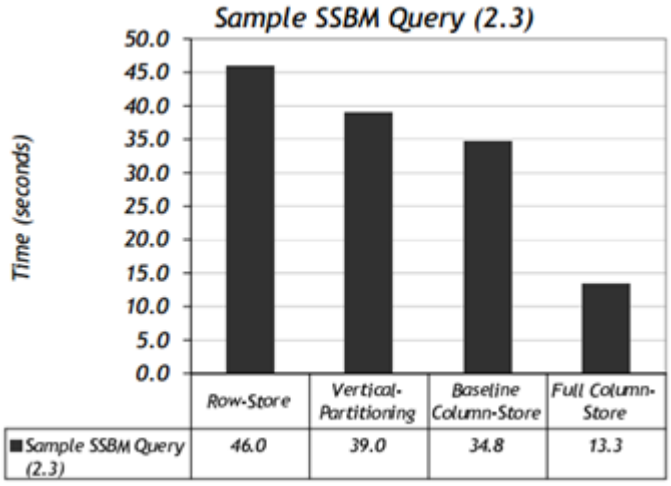
Si se analiza cada una de las soluciones que se tienen en la actualidad, en cuanto al manejo de datos, se detectan las siguientes. La primera y más utilizada es la basada en filas, en donde se tienen varias tablas con información la cual está relacionada una con la otra por medio de llaves. La segunda sería utilizando el particionamiento, en donde cada tabla es separada por medio de sus datos, esta solución es muy utilizada dado que es posible tener catálogos de datos para su análisis. Una tercera opción es improvisar un ambiente basado en columnas en donde se modificará la estructura de los datos y la forma de manipularla.

Como se puede observar en cuanto se buscan soluciones, más complejas se vuelven, dado que son aplicados otros conceptos y técnicas para manipular los datos. Por ejemplo, la primera solución es la más lenta dado que es un esquema relacional, donde existen problemas al aumentar los datos. En la segunda solución se tiene el problema del acceso a datos por particiones, lo cual genera que el costo de consultar los datos sea alto, ya que si se necesitan consultar datos de particiones distintas, el acceso a cada una de ellas afectará el rendimiento.

La tercera solución es por medio de cambiar la estructura de datos utilizando operaciones como Merges para organizar la información a manera de columnas, pero este tipo de solución es muy compleja y requiere de mucho análisis y programación.

Al realizar un comparativo tomando un caso de estudio realizado para comparar las distintas soluciones antes mencionadas, utilizando un esquema estrella convencional, ejecutando la misma consulta en cada una de las soluciones, se observan los resultados en la latencia generada por cada solución.

Figura 68. **Comparativa de bases de datos**



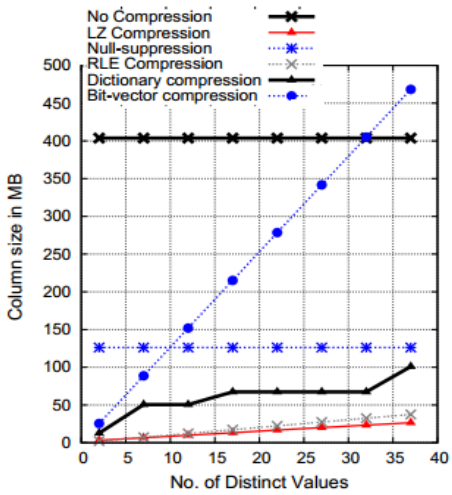

Fuente: Column-store Design.
http://www.dblab.ece.ntua.gr/~gtsat/column%20stores/abadiphd-1.pdf.
Consulta: 24 de julio de 2012.

Según se indica en el estudio, el rendimiento en una base de datos totalmente columnar es muy rentable, a comparación de las más conocidas soluciones aplicadas en los ambientes Data Warehouse, dado que es mucho más viable el evitar las uniones y omitir los valores nulos.

Ya que las bases de datos columnares mejoran el rendimiento en la generación de consultas, es importante remarcar y comprender porque es que el rendimiento mejora al momento de utilizar esta tecnología.

El primer aspecto a analizar será la compresión, como se ha comentado en los puntos anteriores, HBase permite comprimir los datos para así analizar los datos ya sea comprimiendo y descomprimiéndolos o de forma más directa analizarla de forma comprimida.

Figura 69. **Herramientas de compresión**

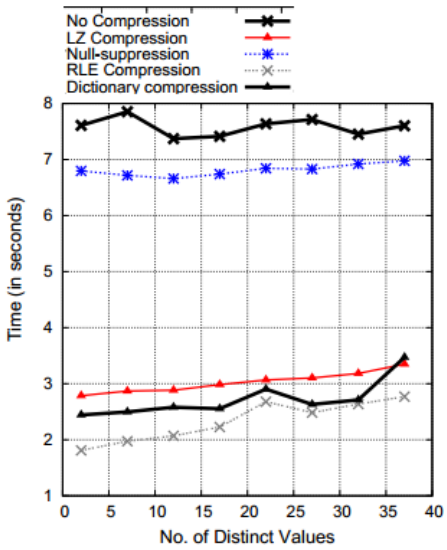



Fuente: Integrating Compretion and Execution.
http://www.dblab.ece.ntua.gr/~gtsat/column%20stores/abadiphd-1.pdf.
Consulta: 24 de julio de 2012.

Se ha comprimido una columna y se muestra la gráfica del tamaño de datos en cómo cada herramienta comprime la información, algunas herramientas disminuyen el tamaño de la información de forma considerable, pero no es la única característica que se debe tomar en cuenta, es importante la forma de descompresión de los datos en cada una de ellas.

Para formar una idea y comparativo de las distintas herramientas de compresión, se tomará en cuenta el ejemplo a continuación.

Figura 70. **Descompresión de datos según el uso de recursos**

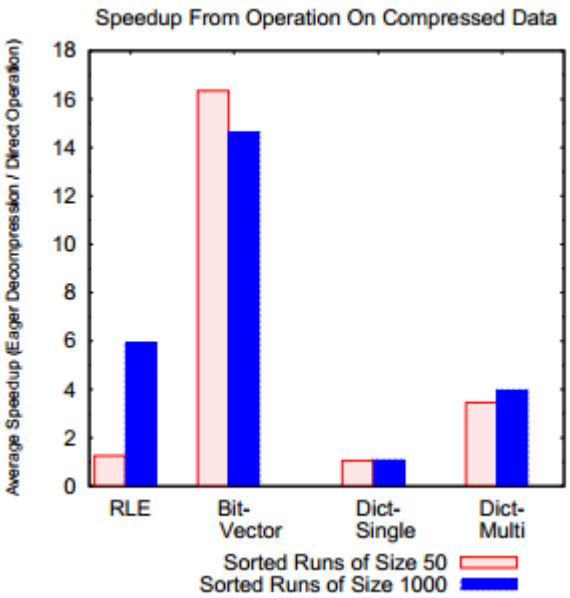



Fuente:Integrating Compretion and Execution.
http://www.dblab.ece.ntua.gr/~gtsat/column%20stores/abadiphd-1.pdf.
Consulta: 24 de julio de 2012.

El rendimiento puede verse afectado, principalmente el uso del CPU, dado que los algoritmos de descompresión en cada una de dichas herramientas varían, se ve el tiempo promedio que le tomará a un CPU realizar el trabajo.

Por lo que el objetivo de comprimir los datos y compartirla entre los servidores del clúster es mejorar el rendimiento. A continuación se realiza un comparativo de las herramientas de compresión, según su capacidad de analizar la información, estando comprimida.

Figura 71. **Rendimiento de la ejecución de datos comprimidos**




Fuente:Integrating Compretion and Execution.
http://www.dblab.ece.ntua.gr/~gtsat/column%20stores/abadiphd-1.pdf.
Consulta: 24 de julio de 2012.

En el cuadro comparativo el rendimiento de las distintas herramientas de compresión, la clave principal es elegir la compresión que se adecue más a las necesidades, para lograr mejores y eficientes análisis de información.

En el siguiente estudio a analizar se implementó un clúster con 1 nodo maestro y 5 nodos esclavos, de los cuales se tienen las siguientes especificaciones:

Nodo maestro

1 Intel Xeon  2,83 GHz (quad)

8 *gigabytes* memoria RAM

2 Discos duros SATA, 500 *gigabytes*

Nodos esclavos

Dell R720XD

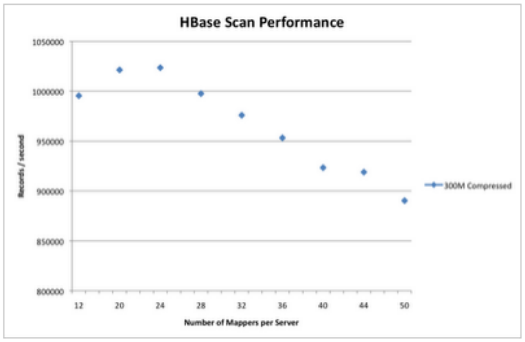
2 Intel Xeon 2,50 GHz (6-core)

64 *gigabytes* memoria RAM

12 Discos duros SATA, 1 *terabyte*

Los recursos de los nodos esclavos son los más completos, dado que serán los encargados de la carga de los datos. En la siguiente gráfica se observa el performance según la cantidad de mapeos en el clúster.

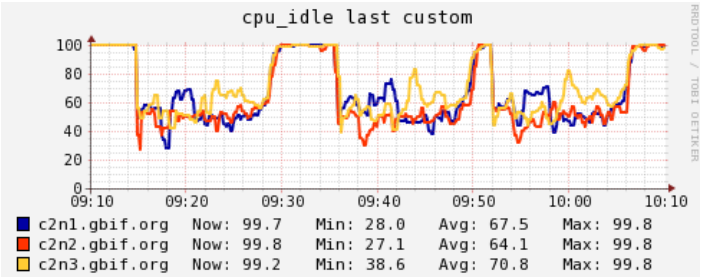
Figura 72. **Performance de clúster HBase**




Fuente:Hardware matters.
http://gbif.blogspot.dk/2012/06/faster-hbase-hardware-matters.html.
Consulta: 28 de julio de 2012.

Con lo que se puede observar que mientras sea mayor el número de mapeos repartidos en el clúster, se obtendrá mejor rendimiento para el análisis de los datos.

Figura 73. **Rendimiento del CPU en el clúster**




Fuente: Hardware matters.
http://gbif.blogspot.dk/2012/06/faster-hbase-hardware-matters.html.
Consulta: 28 de julio de 2012.

Al analizar la carga del CPU, es posible observar que el rendimiento es óptimo ya que la carga de trabajo es compartida por cada uno de los nodos.

Figura 74. **Comparación de lectura y escritura de datos HBase**

| Type | Threads | Mean Time (ms) | Time Standard Deviation (ms) | 90% percentile line (ms) | Throughput[3] (req / s) | Number of Requests |
|---|---|---|---|---|---|---|
| Get | 300 | 17.87 | 29.87 | 38.793 | 16788 | 2999374 |
| Put | 300 | 7.50 | 15.35 | 12.096 | 40000 | 2999978 |
| Get / Put | 300 | 14.62 | 15.83 | 33.976 | 20520 | 2999994 |
| Get | 600 | 55.68 | 66.92 | 161.571 | 10776 | 5999737 |
| Put | 600 | 9.96 | 17.08 | 15.812 | 60240 | 5999891 |
| Get / Put | 600 | 26.48 | 33.10 | 64.336 | 22659 | 5999783 |

Fuente: Performance testing.
http://hstack.org/hbase-performance-testing/.
Consulta: 29 de julio de 2012.

En la tabla comparativa se analizan las operaciones Get y Put según el número de peticiones, las operaciones de escrituras son mucho más rápidas que las de lectura, esto debido a que las escrituras tendrán la misma velocidad que la escritura al disco, dado que no se realiza ningún tipo de comparación en estas operaciones. Dada la desviación estándar en los resultados. Es normal que las lecturas aumenten la latencia, según el número de peticiones, pero se espera que al final se obtengan resultados lineales.

Con estos resultados se puede concluir que mientras más nodos y la carga en cada uno de los nodos sean de forma distribuida, se obtendrán mejores tiempos de latencia, esto de igual forma dependerá del hardware que se disponga y de la velocidad de conexión en la red. Pero estos problemas son específicamente en la arquitectura y no tanto en el diseño de los datos, con esto es posible observar que los ambientes Data Warehouse pueden ser aplicados para analizar los datos de forma óptima.

Con lo que la implementación de esta arquitectura será casi necesaria para las organizaciones que manejen grandes cantidades de datos, ya con esta tecnología se cumplirán los requerimientos de calidad necesarios para analizar la información de forma eficaz y eficiente. Dicho cambio será muy importante para en el futuro evolucionar en la forma en que operan las organizaciones.

## 5.2. Ventajas y desventajas de las bases de datos columnares

Las ventajas que brinda esta tecnología son las siguientes:

- La reducción de latencia en el procesamiento de datos masivo
- Contar con una arquitectura escalable de forma horizontal
- Reducción de costos en cuanto a equipo de cómputo
- Mejoras en el rendimiento de los datos
- Posibilidad de implementar arquitectura en la nube
- Tolerante a fallos

En cuanto a las desventajas es posible enumerar las siguientes:

- Resistencia al cambio en cuanto a los paradigmas actualmente establecidos.
- Existen limitaciones para realizar análisis de Business Intelligence utilizando herramientas *open-source*.

- Alta curva de aprendizaje para la implementación de la arquitectura Hadoop.

- Procesamiento batch en la tecnología Hadoop, no es una herramienta muy amigable para los usuarios.

### 5.3. ¿A quiénes va dirigida la tecnología?

Esta tecnología no ha emergido de forma reciente, sino más bien no era muy popular dado el éxito de las bases de datos relacionales, pero con el paso del tiempo y el auge del Internet los datos van en aumento. Lo cual como sucede con la tecnología, es importante evolucionar junto con ella y buscar alternativas a las soluciones comúnmente aceptadas.

Es posible diferenciar tres grupos de desarrolladores para la tecnología NoSQL:

- Personas que lo utilizan: es el grupo de personas que se encuentran experimentando con la tecnología NoSQL, quienes la utilizan y crean los distintos objetos para descubrir que ventajas proporciona sobre los demás paradigmas.

- Personas que lo niegan: es el grupo de personas que no aceptan la tecnología NoSQL, ya que ven en esto un futuro muy incierto.

- Personas que lo ignoran: es el grupo de personas que esperan a que dicha tecnología madure o simplemente no tienen idea que este tipo de tecnología existe.

Para todos los grupos antes mencionados se pretende dar a conocer esta tecnología, la cual es inmadura y aún no se ha desarrollado ni experimentado completamente, por lo que se desea introducir y así experimentar con cada una de las características que ofrecen.

## 5.4. ¿Quiénes sacan provecho a la tecnología NoSQL?

En la actualidad los ambientes de Data Warehouse residen principalmente en sistemas RDBMS, lo cual da ciertas características y ventajas para el análisis de datos, como por ejemplo, las distintas herramientas de Business Intelligence donde existe variedad en el mercado, se tiene desde herramientas *open-source* hasta herramientas pagadas las cuales ofrecen Oracle, SAP, etcétera.

Las bases de datos columnares aún no son muy conocidas en el entorno de bases de datos, hasta hace poco Oracle a iniciado a incursionar con esta tecnología, con su base de datos llamada Base de datos NoSQL, ya que ven cierto potencial, el cual puede ser explotado y así animar a sus clientes a que adopten esta tecnología.

Algo a tener muy claro es que si se desea implementar una verdadera base de datos columnar, esta ira siempre de la mano con la arquitectura Hadoop, ya que es muy difícil manejar cantidades masivas de datos y todo esto lo provee la arquitectura, con los conceptos que ha introducido Google. Por lo que las organizaciones deberán someterse a cambios en cuanto a la administración y forma de entender los datos, esto será muy complicado para las personas que se han establecido en un paradigma específico, por lo que es importante instruir y dejar muy claro los beneficios y costos que puedan llevar la implantación de dicha tecnología.

Pero sólo es de dar un vistazo y tomar como ejemplo a organizaciones tales como Google, Facebook y Amazon, las cuales son de las más fuertes y conocidas a nivel mundial y analizar el éxito que han tenido en el mundo. Como se ha mencionado Google publicó sus artículos de cómo ellos trabajan en la actualidad y con esto se inicio el desarrollo del ecosistema *open-source* Hadoop, por lo que es muy importante realizar las pruebas necesarias y aportar a la comunidad, para que la arquitectura mejore cada vez más y sea implementada para beneficios comunes.

Esta tecnología tiene mucho potencial, en donde la experimentación y comparación con los sistemas actuales será muy importante para las organizaciones y así tomar la decisión, de continuar en la línea actual o realizar cambios drásticos en la arquitectura implementada e iniciar la incursión con este nuevo paradigma.

Un paso muy importante al momento de implementar la tecnología NoSQL, es la elección de la base de datos columnar que más se acomode a las necesidades de las organizaciones, se ha mencionado principalmente la base de datos HBase, la cual brinda una infinidad de características y opciones de configuración, pero no es la única que existe en el mercado, por lo que se proporcionará un listado de las opciones comerciales y gratuitas para seleccionar la base de datos apropiada acorde al presupuesto y características.

Entre las herramientas comerciales se tiene un amplio listado, según las necesidades y presupuesto.

- Calpont infiniDB Enterprise Edition: es un software manejador de base de datos para manejar los datos de forma columnar y crear aplicaciones para análisis y versión Enterprise que provee facilidad de uso.

- Greenplum: es una base de datos para soportar Big Data y realizar análisis sobre estas para almacenar, administrar *terabytes* de datos, versión pagada para empresas.

- Infobright: es una base de datos columnar la cual está integrada con MySQL, pero su prioridad es el almacenamiento basado en columnas para mejorar el rendimiento y versión pagada para empresas.

- Oracle NoSQL Database: es la propuesta de Oracle que se contrapone a sus bases de datos relacionales, se integra al servidor ExaData.

- SAND CDBMS: es una base de datos columnar la cual ha sido creada especialmente para la optimización de aplicaciones de Business Intelligence.

- Sybase IQ: es una base de datos basada en columnas, utilizado comúnmente para Business Intelligence, Data Warehouse y Data Marts.

- Amazon SimpleDB: almacén de datos no relacional, utilizada para almacenar datos y consultarlos vía servicios Web.

Entre las gratuitas (*open-source*) es posible implementar las siguientes

- Calpont InfiniDB Community Edition: es un software de base de datos para manejar los datos de forma columnar y crear aplicaciones para análisis y versión de código libre y desarrollo de la comunidad.

- Infobright Community Edition: es una base de datos columnar la cual está integrada con MySQL, pero su prioridad es el almacenamiento basado en columnas para mejorar el rendimiento y versión libre para desarrollo de la comunidad.

- Greenplum Community Edition: es una base de datos para soportar Big Data y realizar análisis sobre estas para almacenar, administrar *terabytes* de datos y versión libre para el desarrollo de la comunidad.

- C-Store: base de datos columnar que optimiza las lecturas de datos mucho mejor que las escrituras, se encuentra bajo una licencia permisiva de uso libre.

  Apache Cassandra: base de datos columnar, basada en bigtable. Iniciada y desarrollada por Facebook.

- Hypertable: base de datos columnar, de la cual se obtiene un excelente rendimiento dado que ha sido implementado en C++.

- Acumulo: base de datos columnar, basada en bigtable. Con las mejoras de un control de acceso en las celdas y una mejora en la compresión de datos.

# CONCLUSIONES

1. La arquitectura que se expone está enfocada en reducir la latencia en el procesamiento de datos, por lo que la utilización de varios nodos dentro de un clúster mejorará notablemente el rendimiento, debido a la ejecución paralela de tareas, en donde la carga de trabajo es distribuido entre los nodos.

2. En las bases de datos columnares, la visión de cómo los datos son almacenados y estructurados, es completamente diferente a los modelos relacionales actuales, además las distintas operaciones de inserción, modificación y eliminación de datos, donde cada una de estas tienen algunas variaciones respecto a las operaciones de una base de datos relacional.

3. Las necesidades y problemas que tenga una organización, tales como la cantidad de datos analizada por estas, será el parámetro principal para optar por la implementación de la tecnología NoSQL, dado que no se desea que esta sustituya los modelos relacionales, sino más bien sea una solución alternativa, de la cual se puede analizar *terabytes* de datos y así tomar las mejores decisiones dentro de la organización.

4. Las bases de datos relacionales se ven afectadas en cuanto al rendimiento, debido a la estructuración para el almacenamiento y por esta razón los datos masivos crean problemas para las organizaciones, con esto las tecnologías NoSQL brindan una ventaja al almacenar los datos sin una estructura definida, pero, aumenta la complejidad en cuanto al acceso a los mismos.

5. El costo/beneficio se verá reflejado según los datos de una organización aumenten, dado que conforme la organización sea exitosa, serán más datos los necesarios para analizar, por lo que para las organizaciones que inician será una gran ventaja la implementación inicial de la tecnología NoSQL para así acoplarse de forma más rápida a la línea en la que se dirige el almacenamiento y estructuras de datos.

6. Para las organizaciones que ya cuentan con un sistema relacional para el almacenamiento de datos, será importante y muy difícilmente reemplazable, por lo que Hadoop deberá ser utilizado para el análisis de datos y el desarrollo de aplicaciones Business Intelligence. Utilizando como puente de conexión entre relacional y no relacional los distintos proyectos desarrollados para cumplir este objetivo.

7. Para las organizaciones que se inclinan a la implementación de aplicaciones *open-source*, normalmente enfocan la inversión en el mantenimiento y mejora de los Data Centers, con esto se tendrá la ventaja de obtener lo más reciente en equipo, pero existirá el inconveniente de implementación de software, donde será más complicado y puede llevar mucho tiempo la instalación y configuración de las aplicaciones.

8. Cuando las organizaciones adquieren aplicaciones comerciales, comúnmente existen facilidades de instalación y así como, algunas herramientas que le dan valor agregado al software y se reduce el tiempo de desarrollo, por lo que se podrán realizar los análisis rápidamente. Pero existirá el problema que la inversión aumentará, dado que estas aplicaciones por ser lo último en tecnología, su precio tiende a ser muy elevado.

# RECOMENDACIONES

1. Experimentar con la tecnología Hadoop, antes de implementar en un ambiente productivo, dado que es un paradigma y forma de pensar completamente nuevo y así, observar de primera mano las distintas soluciones que ofrece.

2. Evaluar las necesidades de la organización para la implantación de la tecnología, qué cantidad de datos es analizada y qué problemas existen actualmente en el procesamiento de datos.

3. Analizar las distintas tecnologías *open-source* y de pago, relacionadas con la tecnología Hadoop, ya que cada una de estas proveen diferentes ventajas, las cuales se puede explotar según las necesidades.

4. Utilizar la arquitectura propuesta en este estudio, dado su fácil mantenimiento y explorar la tecnología Hadoop, para luego según las necesidades incrementar los servicios y así optimizar el rendimiento.

5. Para las pequeñas y medianas empresas es ventajoso iniciar el almacenamiento y mantenimiento de datos de forma no estructurada para así evitar el problema de latencia y rendimiento. Principalmente si las redes sociales son una base importante para el análisis de datos.

6. Para las organizaciones que ya cuentan con un sistema relacional para almacenar los datos, utilizar o explorar las tecnologías NoSQL para el análisis de datos, como aplicaciones de Business Intelligence, esto dependerá del tipo de software que se elija. Para el caso de open-source utilizar Sqoop el cual será el puente de conexión entre las dos tecnologías, la otra opción comercial se recomienda Sybase IQ que está basada en columnas y además soporta el modelo relacional, esto proporcionará las distintas herramientas para facilitar los análisis de datos.

7. Para las pequeñas y medianas empresas la implementación de software *open-source* es más común, dado que la inversión normalmente para estas organizaciones no se enfoca en el software. Aún así es importante una correcta implementación de esta tecnología, por lo que deberá enfocarse en la reducción de la curva de aprendizaje para así balancear el rendimiento y el valor agregado entre una aplicación *open-source* y una comercial.

8. Para las grandes empresas la adquisición de software comercial es muy importante, dado que comúnmente los análisis de datos son solicitados de forma inmediata, por lo que el tiempo de aprendizaje debe ser mínimo. El software comercial provee las herramientas necesarias para generar análisis de datos de forma automática y simple, así es como, garantizan que el rendimiento en las aplicaciones no se verá afectado con el aumento de datos.

# BIBLIOGRAFÍA


1. ABADI, Daniel. *Query Execution in Column-Oriented database systems* [en línea]. Massachusetts, USA. [ref. febrero de 2008] Massachusetts Institute of Technology. Disponible en web: <http://www.dblab.ece.ntua.gr/~gtsat/column%20stores/abadiphd-1.pdf>.

2. ARIZA, Christian. *Ecosistema Hadoop* [en línea]. [ref. octubre de 2010]. Disponible en Web: < http://www.christian-ariza.net/ >.

3. DDM. *RAID Solutions* [en línea]. [ref. agosto de 2009]. Disponible en web: <http://www.ddmsa.com/prod/diskarray_raid.html>.

4. INMON, W.H. *Building the data warehouse.* 3a ed. USA: John Wiley & Sons, 2002. 540 p.

5. KIMBALL, R.; ROSS, M. *The Data Warehouse toolkit: the complete guide to dimensional Modeling.* 3a ed. USA: John Wiley & Sons, 2000. 640 p.

6. LARSE, George. *HBase: the Definitive Guide.* USA: O'REILLY, 2011. 522 p. ISBN 978-1-449-39610-7.

7. REFLECTS, Z. *Important resources for learning Google Apps* [en línea]. [ref. mayo de 2011]. Disponible en Web: <http://drzreflects.blogspot.com/2012/05/6-important-resources-for-learning.html>.

8. SHANSHANK, Tiwari. *Professional NoSQL.* USA: John Wiley & Sons, 2011. 361 p. ISBN 978-0-470-94224-6.

9. WHITE, Tom. *Hadoop: The Definite Guide.* 2a ed. USA: O'REILLY, 2010. 600 p. ISBN 978-1-449-38973-4.

10. WIKIPEDIA. *Data Warehouse* [en línea]. [ref. septiembre de 2010]. Disponible en Web: <http://en.wikipedia.org/wik/Data_warehouse>.