**Universidad Central de Venezuela**
**Facultad de Ciencias**
**Escuela de Computación**
**Centro de Investigación en Sistemas de Información**

# Desarrollo de una Arquitectura Big Data para Registros Mercantiles

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por el Bachiller

**Pedro Alfonso Paiva Muñoz**

para optar al título de

**Licenciado en Computación**

**Tutor: Dr. Pedro N. Bonillo R.**

**Octubre, 2016**

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

## ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por el Bachiller Pedro Alfonso Paiva Muñoz C.I: 21.291.683, con el título: **"Desarrollo de una Arquitectura Big Data para Registros Mercantiles"**, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 20 de Octubre de 2016 a las 5:00 pm para que el autor lo defendiera en forma pública, en la Sala 1 de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió satisfactoriamente a las preguntas formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de 20.

En fe de lo cual se levanta la presente Acta, en Caracas a los veinte (20) días del mes de Octubre del año dos mil dieciséis (2016), dejándose constancia de que actuó como Coordinador del Jurado el Doctor Pedro Bonillo.

Bonillo Ramos Pedro

**Profesor Pedro Bonillo**

Tutor

Edgar Bernal

**Profesor Edgar Bernal**

Jurado

**Profesora Rut Martínez**

Jurado

# AGRADECIMIENTOS

A Dios y La Virgen.

A mis familiares, por su constancia, dedicación y sacrificio, por apoyarme en todo momento y por inculcarme que se puede alcanzar cualquier meta si uno se lo propone.

A mi tutor, Prof. Pedro Bonillo, por haberme dado su mejor disposición en cada una de las consultas.  Gracias Prof. Pedro, por su paciencia y dedicación.

A cada uno de los profesores de la Escuela de Computación por compartir sus conocimientos y experiencias.

A todas aquellas personas no nombradas, pero que de una u otra manera prestaron su excelente colaboración.

A todos, muchas gracias.

# RESUMEN

Debido al gran volumen de datos que comienzan a manejar muchas organizaciones en Venezuela, dichas organizaciones requieren de soluciones tecnológicas distintas al modelo relacional, el cual no está diseñado para manejar tantos datos de forma óptima, por lo que estas empresas comienzan requerir de soluciones Big Data. Entre estas organizaciones encontramos el Servicio Autónomo de Registros y Notarías adscrito al Ministerio Publico, el cual maneja una cantidad considerable de datos transaccionales diariamente en sus oficinas encargadas del Registro Mercantil a nivel nacional .Este Trabajo Especial de Grado tiene como objetivo desarrollar una arquitectura de Big Data para almacenar, visualizar y consultar grandes volúmenes de datos sobre un Sistema de Registro Mercantil, en el que actualmente el modelo relacional no permite realizar adecuadamente operaciones de consulta sobre más de cien millones de registros, en menos de cinco segundos y con una alta variabilidad en las columnas que conforman las tablas del sistema. En este trabajo se utilizó la metodología *Attribute Driven Design* para diseñar la arquitectura y luego se implementó cada uno de los componentes de la misma con herramientas de software libre, entre las cuales podemos mencionar *Apache Cassandra* como base de datos transaccional, *Apache Hadoop* como estacionamiento, *Apache Hive* como motor de consultas, *Apache Solr* como herramienta de inteligencia de negocios y *HUE* como herramienta de administración y visualización de reportes y datos.

Para lograr éste objetivo fue necesario analizar, estudiar y comparar distintas herramientas de Big Data preferiblemente *open source,* con el fin de identificar los componentes que conformaron el diseño de la arquitectura, los cuales fueron probados y desarrollados sobre un Sistema de Registro Mercantil, para luego obtener las conclusiones y las recomendaciones respectivas.

**Palabras claves:** Registros Mercantiles, Big Data, Arquitectura de Software.

# ABSTRACT

Due to the large volume of data that manage many organizations in Venezuela, they require different technological solutions to the relational model, which is not designed to manage a high volume of data optimally, for that reason these companies require Big Data solutions. Among these organizations are SAREN, which handle a large amount of transactional data daily. The main goal of this work is analyze and implement a Big Data architecture for storing, querying and visualize a big amount of data in a commercial register system, in which relational model is not possible to make consult operations properly due to the large amount of data that handle. In this work, the Attribute Driven Design was implemented for architectural design and then their components were implemented. Between the software tools that were used in the architecture, we found: Apache Cassandra as the transactional data base, Apache Hadoop as the staging area, Apache Hive as the query engine, Apache Solr as the BI tool, HUE as the Data Governance Manager and visualization tool.

To achieve this goal, it was necessary to analyze study and compare open source tools that would help us to implement the necessary architecture to manage a significant amount of data to process, store and visualize.

**Keywords**: Commercial Register System, Big Data, Software Architecture.

# INDICE

# INDICE DE TABLAS

## INDICE DE FIGURAS

# INTRODUCCIÓN

En los últimos años se ha observado un aumento exponencial en la generación de datos por las empresas, esto quizás por el auge del internet como herramienta de globalización a la hora de expandirse, lo cual trae como resultado una cantidad masiva de datos generados, ya sea por las transacciones de la empresa o por los datos que generan los usuarios en las redes sociales. (Wall, 2014)

Todos éstos datos históricos pueden ser utilizados por las empresas u organizaciones para tomar mejores decisiones en su respectivo modelo de negocio, bien sea para saber cuándo enviar más productos a los almacenes, cuándo producir más de un producto, saber el momento ideal para lanzar un descuento o incluso aumentar las ventas ubicando los productos en sitios estratégicos (como el caso de Walmart) (The Register, 2006)

Estas empresas u organizaciones que utilizan sus datos como una fuente de información valiosa para el negocio vienen a conocerse como organizaciones *"Data Driven"*, las cuales han tenido muchos casos de éxito y nos enseñan que los datos son "el nuevo petróleo" para indicar su importancia dentro de las mismas. (Morgan, 2015)

El problema surge en situaciones en las que se manejan grandes volúmenes de datos. El modelo relacional propuesto por Edgar Frank Codd es difícil de escalar, por lo que no es aplicable en este tipo de situaciones.

La finalidad de este Trabajo Especial de Grado es desarrollar una arquitectura para manejo de grandes volúmenes de datos, basada en componentes de software libre.

Para cumplir con el objetivo del proyecto, se procedió a analizar cada una de las herramientas utilizadas en una arquitectura Big Data mediante el uso de patrones de diseño Big Data y se encontraron que aquellas que siendo de

Software Libre lograron cumplir las expectativas y los objetivos de la organización, utilizando como alcance la base de datos relacional desarrollada bajo Oracle en los Registros Mercantiles del Servicio Autónomo de Registros y Notarías (SAREN) de la República Bolivariana de Venezuela.

Este documento se encuentra estructurado de la siguiente manera:

**Capítulo 1**: describe a la empresa que representa el caso de estudio de este TEG. Se relata la reseña histórica y se definen la misión, visión y objetivos de la empresa.

**Capítulo 2**: en él se describe el problema de investigación, manifestaciones y evidencias necesarias para establecer el objetivo general, los objetivos específicos, la justificación, el alcance y las limitaciones.

**Capítulo 3**: establece las bases conceptuales que soportan la investigación. Se describen las tecnologías principales involucradas en Big Data, breve cronología, definición de arquitectura de Software y como diseñar una arquitectura de software utilizando una metodología.

**Capítulo 4**: define el marco metodológico, describe la Metodología de *Atribute-Driven Design*, indica el tipo de investigación, la población y muestra, así como también las técnicas e instrumentos de recolección de datos.

**Capítulo 5**: corresponde al marco aplicativo, en el cual se presentan los resultados obtenidos al aplicar la metodología seleccionada, mediante la cual se desarrolló y probó la arquitectura.

Para finalizar, se presentan las conclusiones y recomendaciones, seguido por la Bibliografía y Anexos.

# Capítulo 1: Identificación de la Empresa

En este capítulo se describirá la empresa SAREN (Servicio Autónomo de Registros y Notarías), ya que es el caso de estudio en este trabajo.

## 1.1 Reseña Histórica

El notariado en Venezuela, o escribano como institución independiente, propiamente dicha existió durante la Colonia y a comienzos de la República, rigiéndose por la legislación hispana.

La Recopilación de Indias y en algunas otras Real Cédula. Los oficios de escribanos se obtenían en los primeros tiempos por concesiones de la Corona a personas que en América habían desempeñado ciertos cargos de utilidad, posteriormente una Real Cédula ordenó que sólo se obtendrían por compra o sesión de su propietario.

La ley 24 título 20, libro VIII de la Recopilación de Indias fija los pormenores de semejante operación vacante. Un puesto de escribano se la otorgaba al mejor postor en venta pública, el Capitán General o Gobernador daba la institucionalidad al adquiriente, pero el expediente debía pasar a España para su calificación y expedición del título respectivo ya definido.

A partir de 1761 existía en Caracas el oficio de anotador de hipotecas, quien trasladó un libro especial, nota de todos los actos que constarán en los archivos de los escribanos, en los cuales se establecieron los gravámenes. Esto permitió examinar con exactitud el estado de los muebles e inmuebles, ambos susceptibles de hipotecas. En 1825 el Congreso de la República de la Gran Colombia dicta una Ley Orgánica del Poder Judicial el 11 de mayo.

En 1826 el Congreso de Colombia incorpora a la Hacienda Nacional, este oficio de anotador de hipotecas, con el fin de aumentar las rentas nacionales al establecer impuestos a los particulares con motivos de sus contratos y actos civiles. En 1826 se prohibió a los escribanos so pena de la pérdida del oficio, otorgar escritura alguna sin que se acreditase el Derecho de Registro establecido y se ordenó insertar en las escrituras las boletas en que constase

el pago del impuesto respectivo. Después de 1830 en que se separó Venezuela de la Gran Colombia, se mantuvieron las instituciones de las escribanías y de anotación de hipotecas y registros.

En 1836 se crea el primer Código de Procedimiento Judicial de Venezuela, se ordenaba que con excepción del otorgamiento de poderes y de registros, los escribanos y jueces donde no los había, continuarían otorgando los documentos hasta que se establecieran La Oficinas de Registros a los cuales pasarían las funciones de los escribanos. El 24 de mayo de 1836, se ordenó organizar en cada provincia una Oficina Principal de Registro.

El Gobierno Nacional creó el Ministerio de Justicia mediante Decreto No 40 contenido en la Gaceta Oficial No 23.418 del 30/12/1950, confiriéndole una serie de funciones de conformidad con la Ley Orgánica de la Administración Central, promulgada en Gaceta Oficial No 1932 Extraordinaria, de fecha 28/12/76, y donde su artículo 34 establecía "…corresponde al Ministerio de Justicia la planificación y la realización de las actividades del Ejecutivo Nacional en el sector de Justicia y de Defensa Social, que comprende las relaciones con el Poder Judicial, la Legislación y la Seguridad Jurídica, la Prevención y la Represión del Delito y las Relaciones con los Cultos establecidos en el país; y en particular las actividades siguientes:

- El Registro Público
- Las Notarías y los Registros Mercantiles
- El Archivo General de la Nación …"

Funcionó como Dirección General Sectorial de Registros y Notarías en el año de 1.994, y como Dirección General de Registros y Notarías a principios del año de 1.996.

Luego en ese mismo año mediante Decreto 3.148 publicado en Gaceta Oficial 36.615 de fecha 06/01/99 se crea de derecho la Superintendencia de Registros y Notarías, sin embargo, ella no ejerció las actividades administrativas correspondientes, manteniéndose como la Dirección General de Registros y de Notarías.

En fecha 13 de noviembre de 2001, mediante Decreto No 1.554, se promulga la Ley del Registro Público y del Notariado, la cual en su artículo 14 establece la creación de la Dirección Nacional de Registros y del Notariado.

Adicionalmente también se encuentra contemplada la Modernización conceptual de las instituciones registrales, según la Exposición de Motivos del Decreto No. 1554 con Fuerza de Ley de Registro Público y del Notariado. (SAREN, 2016)

## 1.2 Misión

Garantizar la seguridad jurídica de las actuaciones de los usuarios mediante la publicidad registral y fe pública, en el marco de la legalidad, de procesos expeditos y oportunos; ejerciendo el control de las operaciones a nivel nacional.

## 1.3 Visión

Ser un órgano que coadyuve a garantizar la seguridad jurídica de los actos protocolizados y autenticados de los usuarios, mediante un sistema integral de registros y notarías confiable, eficiente, auto-sustentable y transparente.

## 1.4 Objetivos de la empresa

- Internalizar por medio del plan comunicacional una cultura organizacional alineada con la nueva filosofía de gestión.

- Proveer a los ciudadanos venezolanos un Sistema de Registros y Notarías de fácil acceso a la información por medio de herramientas efectivas.

- Garantizar que los actos y solicitudes de los ciudadanos alcancen la máxima seguridad, eficiencia y rapidez, con la finalidad de obtener procesos oportunos y expeditos en los servicios solicitados.

- Proveer una plataforma tecnológica y una conectividad que garantice, por medio de la adquisición, captación y capacitación de recursos tecnológicos y humanos plenamente comprometidos, eficiente y eficaz, una base de datos con información, local y nacional, única, objetiva y confiable, con la participación todos los actores del sistema.

- Procurar que el máximo de los trabajadores alcance la idoneidad y un desarrollo personal y profesional acorde con los perfiles del cargo que ocupan, y así obtener la excelencia en los servicios.

- Garantizar el funcionamiento y optimización del SAREN, con el fin de proveer control, seguridad y calidad de servicio, en un marco de permanencia y sostenibilidad en el tiempo.

- Centralizar y controlar los ingresos, así como los gastos operativos, con la finalidad de lograr la sustentabilidad del SAREN y reinvertir parte de ellos para mejorar la efectividad de los servicios en todas las áreas que lo requieran y en función del desarrollo organizacional de la institución.

## 1.5 Organigrama

La Figura 1 (ver Figura 1 Organigrama SAREN) muestra el organigrama de la empresa SAREN, en ella se destaca La Oficina de Tecnología de la Información, ya que representa el área en la cual se desarrolló este Trabajo Especial de Grado.

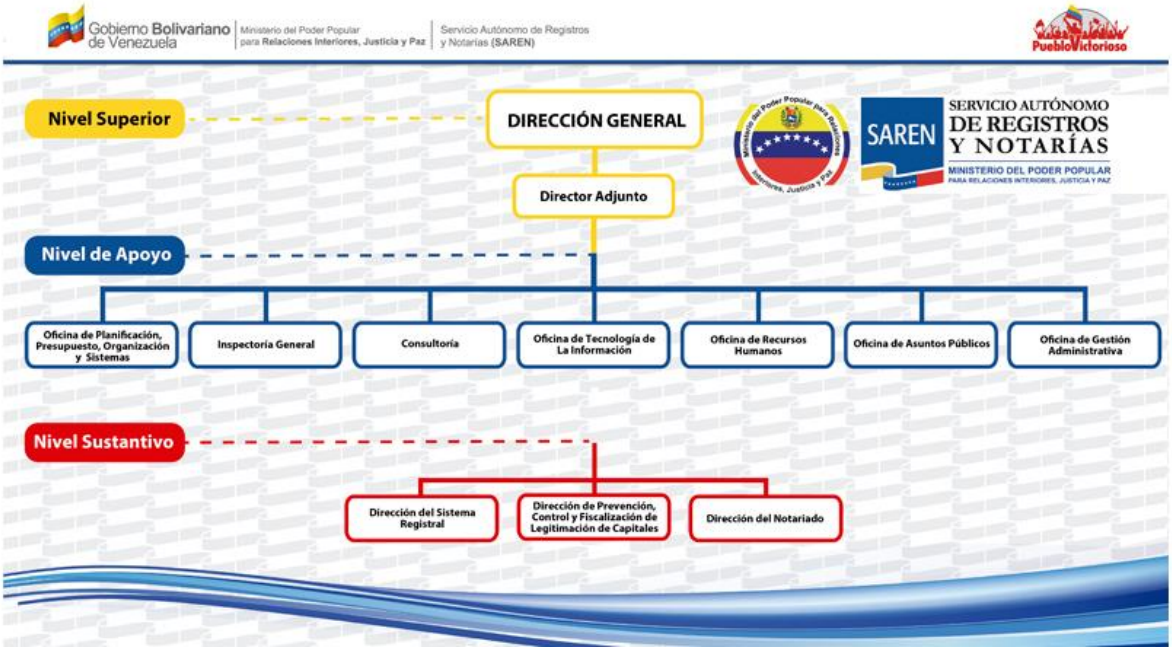



*Figura 1 Organigrama SAREN*

*Fuente: (SAREN, Organigrama, 2016)*

# Capítulo 2: Problema de Investigación

En este capítulo se describe la situación objeto de estudio, ubicándola en el contexto para entender su origen y de esta manera proceder a plantear, justificar y limitar el problema; también se define los objetivos del Trabajo Especial de Grado (TEG).

## 2.1 Contexto

En el mundo, la generación de datos ha incrementado exponencialmente en los últimos años con el auge del internet y la globalización. Estos datos provienen de diferentes fuentes y en diferentes formatos, y pueden estar estructurados o no estructurados, lo cual implica un reto en cuanto a su almacenamiento y procesamiento.

El presente trabajo está enfocado en diseñar una arquitectura *Big Data* para Registros Mercantiles por lo cual se explicará el contexto de la organización o institución que maneja dichos registros.

El ente encargado de realizar los Registros Mercantiles en Venezuela conocido como SAREN (Servicio Autónomo de Registros y Notarías), maneja una cantidad de registros enorme en sus bases de datos por oficina, las cuales son aproximadamente más de 400 a nivel nacional. Ellos son los encargados de almacenar todas las transacciones de los registros mercantiles, como serian el registro de firma personal, constitución de compañías anónimas, constitución de consorcios, entre otros.

## 2.2 Planteamiento del Problema

Como se mencionó anteriormente, SAREN maneja una gran cantidad de registros, proveniente de distintos formatos. El 100% de sus datos proviene de

transacciones, las cuales se almacenan dependiendo del tipo de transacción en una de sus 913 tablas en Oracle.

Los 10 servidores centralizados que disponen están alcanzando el máximo de su capacidad, y las consultas que se realizan en dichos servidores pueden llegar a tardar más de 7 minutos dependiendo de la complejidad de la consulta, lo cual indica que el modelo relacional ya no está en la capacidad de soportar sus necesidades del negocio. Además de que no disponen de un clúster centralizado actualizado, por lo que la actualización de sus datos se realiza una vez al mes en algunas oficinas y en alguna de sus tablas. Tampoco tienen un contrato de licenciamiento y de mantenimiento con la empresa Oracle.

El Problema radica en que el sistema actual de Base de Datos de los Registros Mercantiles del SAREN está a punto de colapsar y no permite obtener la información oportuna y necesaria para administrar de forma centralizada los datos, esto se traduce en pérdidas de dinero y de credibilidad en la institución.

En base a lo expuesto anteriormente surgieron las siguientes interrogantes:

- ¿Cuáles son las arquitecturas de Bases de Datos alternativas, que permitirían manejar adecuadamente la cantidad de registros que actualmente almacena el SAREN en sus Registros Mercantiles?
- ¿Cuáles son los componentes de estas arquitecturas de Bases de Datos alternativas?
- ¿Cuáles de estos componentes pueden estar en Software Libre?
- ¿Cuáles de estos componentes pueden incluirse en la propuesta de una nueva arquitectura Big Data para ser implementada en los Registros Mercantiles de SAREN?
- ¿Cuáles de estos componentes pueden tener un contrato de mantenimiento y una empresa que los respalde en Venezuela a fin de garantizar la continuidad de la arquitectura propuesta?

## 2.4 Objetivos

Como una solución al problema antes planteado, se tuvo previsto llevar a cabo el objetivo general y los objetivos específicos:

### 2.4.1 Objetivo General

El objetivo general del presente trabajo es Desarrollar una Arquitectura Big Data para Registros Mercantiles.

### 2.4.2 Objetivos Específicos

Los objetivos específicos del presente trabajo son los siguientes:

- Diseñar la arquitectura Big Data.
- Seleccionar las herramientas asociadas a cada componente de la arquitectura diseñada.
- Proponer e implementar la arquitectura con las herramientas seleccionadas.

## 2.5 Justificación

La justificación del presente trabajo se apoya principalmente en los siguientes puntos:

- Los servidores de SAREN alcanzaron su capacidad máxima de almacenamiento por lo que surge la necesidad de comprar más servidores con mayor capacidad o implementar una nueva arquitectura de almacenamiento de la información que se adapte a las necesidades de la empresa.
- SAREN no dispone de una base de datos centralizada que permita hacer reportes de todas las oficinas a nivel nacional.

- La licencia de Oracle que utiliza SAREN está llegando a su fin, por lo que se necesitó pensar en una solución de Software Libre, además de que el presupuesto de SAREN no abarca el pago a empresas que cobren en dólares.
- Las consultas realizadas en sus bases de datos consumen bastante tiempo en responder, debido a la enorme cantidad de registros que manejan a nivel nacional.
- Al disponer de tantos registros en sus bases de datos el sistema relacional comienza a tener bajas de rendimiento, como el caso de las consultas. Esto debido a que los sistemas de bases de datos relacionales no fueron diseñados para almacenar y procesar enormes cantidades de registros y el uso de *joins* consume muchos recursos del computador.
- La desnormalización de los datos ayuda a disminuir los tiempos de lectura y escritura, por lo que una solución utilizando una base de datos *NoSQL* junto con un paradigma *Map Reduce* es lo más indicado en este caso.
- En la ley de infogobierno de la República Bolivariana de Venezuela destaca en su artículo 34 que todo proyecto de tecnología de la Administración Pública debe hacerse utilizando Software Libre, para garantizar al Poder Público el control sobre las tecnologías de información empleadas y el acceso de las personas a los servicios empleados. SAREN al ser un órgano adscrito al Ministerio del Poder Público se rige por esta misma ley.
- El desarrollo de una arquitectura de Big Data en SAREN implica una solución innovadora a nivel nacional, debido a que la utilización de este tipo de soluciones de software no se ha implementado anteriormente en el país.

## 2.6 Alcance

En este trabajo de investigación se hizo el desarrollo de la nueva arquitectura.

El desarrollo de la arquitectura se realizó en base a los datos en Oracle del sistema actual, que abarca un total de 50 oficinas a nivel nacional encargadas de los registros mercantiles en SAREN.

La propuesta de la arquitectura se desarrolló en la sede principal de SAREN ubicada en Altamira, Municipio Chacao en Caracas, Venezuela.

## 2.7 Limitaciones

Dentro de las limitaciones se encuentran las siguientes:

- Permisología restringida para manejar el esquema de bases de datos Oracle del sistema actual de las oficinas del SAREN a nivel nacional.
- La poca información disponible para el diseño de una arquitectura de Big Data.
- La mayor parte de la información disponible se encuentra en el idioma inglés y el investigador tiene como lengua origen el castellano.
- La conexión a las distintas oficinas para la extracción de los esquemas de datos es compleja y en algunos casos los esquemas de datos son distintos. Adicionalmente la conexión se realiza por un enlace intermitente y de muy baja velocidad.

# Capítulo 3: Marco Conceptual

En éste capítulo se presentan los antecedentes y las bases teóricas que sustentan la investigación.

## 1.1 Registros Mercantiles

La Real Academia Española define los registros mercantiles como un registro que con carácter público sirve para la inscripción de actos y contratos de comercio, preceptuada legalmente en determinados casos. (Española, 2016)

El registro mercantil es una institución administrativa que tiene por objeto la publicidad oficial de los empresarios en él inscritos. La función principal del registro mercantil es ser un instrumento de publicidad. Los empresarios tienen obligación de publicar una serie de informaciones que se consideran esenciales de cara al tráfico jurídico y el registro mercantil permite la publicidad de dicha información, para mayor seguridad jurídica y económica. (Wikipedia, 2016)

También podríamos definir al Registro Mercantil como la Institución que se ha creado, como fuente de información para conocer el verdadero estado de las situaciones jurídicas en materia comercial. La finalidad que persigue el Registro Mercantil es hacer pública la vida mercantil del comerciante, tanto en su capacidad, condiciones para obligarse, responsabilidad en sus obligaciones, contratos, modificaciones y la solvencia en el respaldo de los actos de comercio.

El Registro Mercantil estará a cargo de un Registrador Mercantil en los lugares donde los haya, si no existe Registro Mercantil serán los Tribunales de Primera Instancia que conozcan de la materia Mercantil y ellos cumplirán con todas las atribuciones previstas y desarrolladas en el Código de Comercio. (Comercio, 2016)

El Servicio Autónomo de Registros y Notarías (SAREN) es un organismo dependiente del Estado Venezolano y adscrito al Ministerio del Poder Popular para Relaciones Interiores, Justicia y Paz, encargado de registrar y/o notariar documentos de ventas de bienes muebles e inmuebles, hierros y señales, compañías anónimas, firmas personales, registros de títulos universitarios, entre otras. Asimismo, haciendo de uso público y oficial estos documentos. Otra de sus funciones es dar Garantía de Seguridad Jurídica en el país. En este trabajo de investigación los Registros Mercantiles se corresponderán a los gestionados por el SAREN.

## 1.2 Big Data

A continuación, se presenta una breve introducción a *Big Data* su historia y tecnologías involucradas.

### 1.2.1 Introducción

El concepto de *Big Data* se refiere al almacenamiento y procesamiento de enormes cantidades de datos, tan desproporcionadamente grandes que no es posible tratarlos con las herramientas de base de datos convencionales. Sin embargo, el término *Big Data* no hace referencia a alguna cantidad en específico, debido a que lo que para una empresa determinada puede ser *Big Data*, puede no serlo para otra (Figura 2 Big Data en las empresas Fuente: Herramientas para Big Data: Entono Hadoop (Sanchez,2014)).

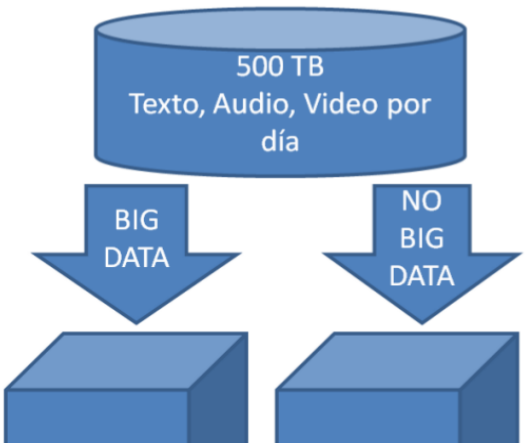



*Figura 2 Big Data en las empresas*
*Fuente: Herramientas para Big Data: Entono Hadoop (Sanchez,2014)*

Actualmente para considerar una solución Big Data, se toma en cuenta que los datos presenten cuatro características principalmente, los cuales son: Variedad, Velocidad, Volumen y Veracidad (también se hablan de otras V's como la visualización, sin embargo, las cuatro mencionadas anteriormente son las principales). La variedad indica que la data puede estar en diferentes formatos, puede ser estructurada o no estructurada y venir de distintas fuentes, la velocidad indica la rapidez con la que se generan los datos, el volumen indica la cantidad que se genera y la veracidad indica la certeza de los datos. (IBM, 2015)

Para poder comprender mejor el uso de *Big Data* primero debemos comprender la naturaleza de los problemas que lo han hecho necesario.

Actualmente vivimos en la era de la información, en la que el volumen total de datos almacenados de forma electrónica no se mide fácilmente. Sin embargo, en 2006 se estimó que el tamaño del universo digital ascendía a 0,18 *zettabytes* y en 2011 ya eran 1,8 *zettabytes.* Estos son unos ejemplos de la cantidad de datos que manejan algunos de los servicios más conocidos: (Magazine, 2013)

- Facebook almacena aproximadamente 10 mil millones de fotografías
- Ancestry.com almacena aproximadamente 2,5 *petabytes* de datos.
- New York Stock Exchange genera alrededor de 1 *terabyte* al día.
- El colisionador de hadrones de Suiza produce alrededor de 15 *petabytes* de datos al año.
- Los usuarios de YouTube suben 100 horas de video en 1 minuto

- Se producen 300.000 tweets en un minuto.
- 2.000.000 de consultas a Google en sólo un minuto.

Y como éstos existen muchos casos similares de diversas empresas u organizaciones.

Además del gran volumen de información que se genera diariamente, ésta existe en una gran variedad de datos que pueden ser representados de diversas maneras en todo el mundo, por ejemplo, de dispositivos móviles, audio, video, sistemas GPS, incontables sensores digitales en equipos industriales, automóviles, entre otros. Las aplicaciones que analizan éstos datos requieren que la velocidad de respuesta sea lo demasiado rápida para lograr obtener la información correcta en el momento preciso.

Los principales tipos de datos se resumen en 5 (Figura 3 Tipos de datos en Big Data)

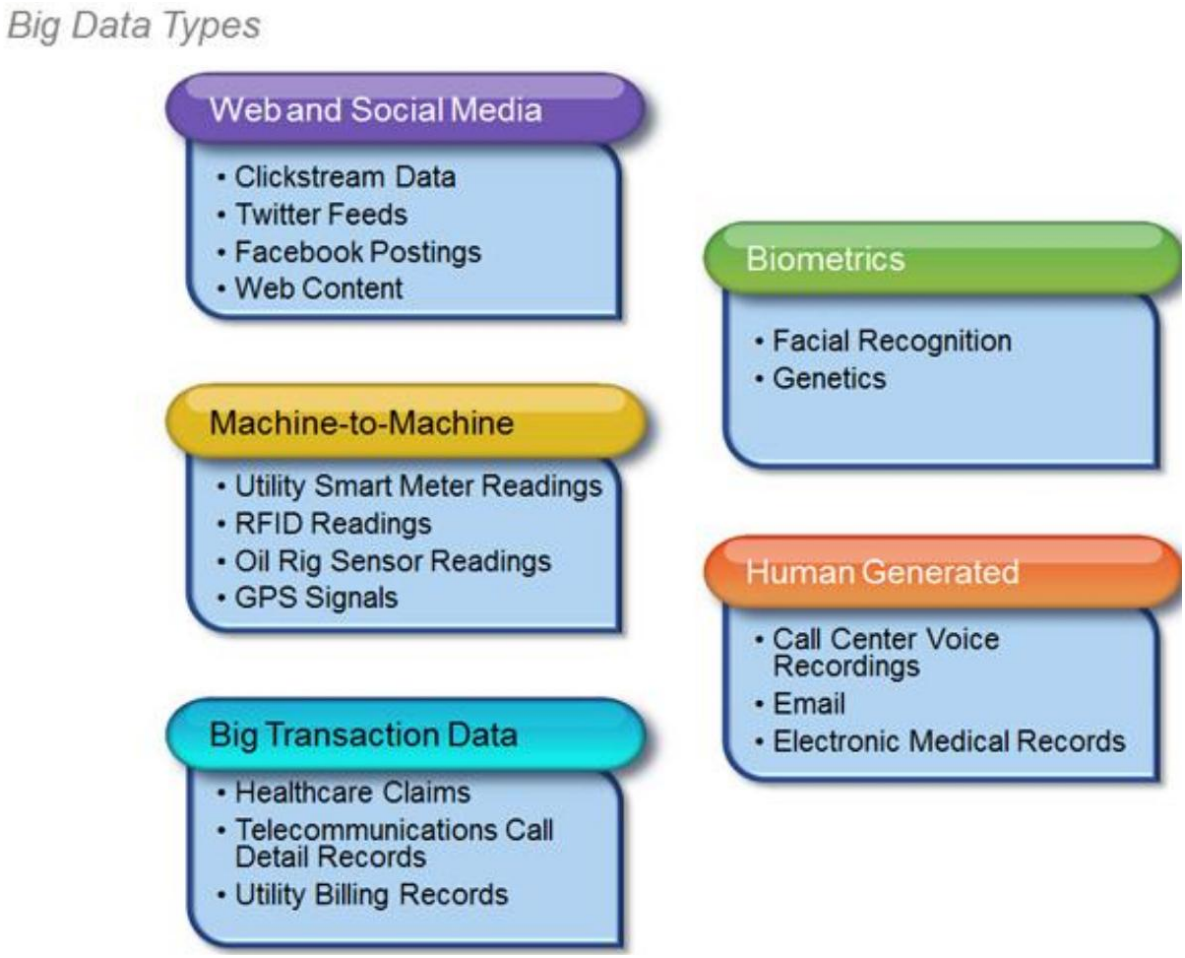



*Figura 3 Tipos de datos en Big Data*

*Fuente: (Soares, 2012)*

- **Web and Social Media:** Contenido web e información que es obtenida de las redes sociales como Instagram, Facebook, Snapchat, blogs.
- **Machine-to-Machine:** Sensores o medidores que capturan algún evento en particular (velocidad, temperatura, presión, variables meteorológicas o variables químicas) los cuales transmiten a través de redes alámbricas, inalámbricas o híbridas a otras aplicaciones que traducen estos eventos en información significativa.
- **Big Transaction Data:** Registros de facturación, en telecomunicaciones registros detallados de las llamadas, etc. Estos datos transaccionales están disponibles en formatos tanto semiestructurados como no estructurados.
- **Biometrics:** Información biométrica en la que se incluye huellas digitales, escaneo de retina, reconocimiento facial, genética, etc. En el área de seguridad e inteligencia, los datos biométricos han sido información importante para las agencias de investigación.
- **Human Generated:** Las personas generamos diversas cantidades de datos como la información que guarda un *call center* al establecer una llamada telefónica, notas de voz, correos electrónicos, documentos electrónicos, estudios médicos, etc.

### 1.2.2 Cronología de *Big Data*

Desde que surgieron las primeras formas de escritura hasta los centros de datos modernos, la raza humana no ha dejado de recopilar información. El crecimiento del sector tecnológico ha provocado el aumento desmesurado del volumen de datos, por lo que son necesarios sistemas de almacenamiento de datos más sofisticados. Se expondrá brevemente los hitos más importantes en el surgimiento del *Big Data*:
(winshuttle, 2016)

- El término *Big Data* es un concepto que hace referencia al almacenamiento de grandes volúmenes de datos y a los procesos necesarios para encontrar patrones repetitivos dentro de ésos datos.
- Comienzo de sobrecarga de información (1880): El Censo de los Estados Unidos del año 1880 tardó ocho años en tabularse, y se calcula que el censo de 1890 hubiera necesitado más de 10 años para procesarse con los métodos disponibles en la época. Si no se hubieran realizado avances en la metodología, la tabulación no habría finalizado antes de que tuviera que realizarse el censo de 1900.
- La máquina tabuladora de Hollerith (1881): La influencia de los datos del censo derivó en la invención de la máquina tabuladora de Hollerith (tarjetas perforadas), que fue capaz de domar esta ingente cantidad de información y permitir realizar el trabajo aproximadamente en un año. Hizo que Hollerith se convirtiera en emprendedor, y su empresa pasó a formar parte de lo que hoy en día conocemos como IBM.
- El *boom* del crecimiento demográfico (1932): La sobrecarga de información prosiguió con el aumento desmesurado de la población en los Estados Unidos, la emisión de los números de la seguridad social y el crecimiento general del conocimiento (y la investigación), aspectos que exigían un registro de la información más preciso y organizado.

Vea el resto de la cronología en ANEXO 10: Cronología Big Data

Para entender el funcionamiento de *Big Data* se expondrá cada una de las tecnologías involucradas:

### 1.2.3 Tecnologías involucradas

La implementación de una solución *Big Data* se descompone en diferentes tecnologías que trabajan conjuntamente para lograr el objetivo final de almacenar y procesar grandes volúmenes de datos. Entre las tecnologías involucradas se encuentran el framework del proyecto Apache Hadoop (Apache, Hadoop, 2016), el uso de base de datos NoSQL (NoSQL, 2016) entre

las que podemos mencionar Cassandra, Hbase, MongoDB, Neo4j, entre otras. Sin embargo, también se hace necesario mencionar el Teorema de Brewer o Teorema de CAP (Brewer, 2000), en el cual se enfocan las bases de datos NoSQL.

#### 1.2.3.1 Apache Hadoop

*Apache Hadoop* es una plataforma que permite el procesamiento de grandes volúmenes de datos a través de *clúster*, usando un modelo simple de programación. Proporciona un *framework*, escrito en *Java*, sobre el cual desarrollar aplicaciones distribuidas que requieren un uso intensivo de datos y de alta escalabilidad. Este proyecto es administrado por *Apache Software Foundation*.

Se presenta como una solución para los programadores sin experiencia en desarrollo de aplicaciones para entornos distribuidos, dado que oculta la implementación de detalles propios de estos sistemas: paralelización de tareas, administración de procesos, balanceo de carga y tolerancia a fallos. (Sánchez, 2014)

##### 1.2.3.1.1 Historia

Antes de empezar a hablar sobre el funcionamiento de *Hadoop*, vamos a destacar algunos detalles sobre su historia.
*Hadoop* no es un acrónimo, es un nombre inventado. Es el nombre que el hijo del creador, Doug Cutting, le dio a un peluche de un elefante amarillo. Corto, relati- vamente fácil de deletrear y pronunciar, sin significado y sin uso externo. Esto ha provocado que las herramientas relacionadas con *Hadoop* tiendan a tener nombres relacionados con elefantes u otras temáticas animales (*Pig, Mahout*, *Hive*...).
Sus orígenes se remontan a *Apache Nutch*, que era un motor de búsqueda web *open-source* dentro el proyecto *Lucene*. La construcción de este motor era

una meta ambiciosa y muy cara, estimaban sus creadores, pero el objetivo valía la pena. Fue lanzado en 2002, y se propagó de forma rápida y eficaz. Sin embargo, al poco tiempo, se dieron cuenta de que su arquitectura era incapaz de escalar a los billones de páginas que había en la Web por aquel entonces, y no sabían cómo solucionarlo. En 2003 *Google* publicó un artículo que describía la arquitectura del *Google's Distributed Filesystem*, lo cual proporcionaba a *Nutch* la solución al problema que se había encontrado. Esto también ayudaba a minimizar los tiempos gastados en tareas administrativas como la gestión de los nodos de almacenamiento. (Vance, 2009)

En 2004 se comenzó a escribir una implementación propia y open-source del sistema de ficheros utilizado por *Google*, el *Nutch Distributed FileSystem* (*NDFS*). En ese mismo año *Google* publicó otro artículo en el que se presentaba *MapReduce* al mundo. A principios de 2005 los desarrolladores de *Nutch* ya tenían su propia implementación de *MapReduce* funcionando para el *NDFS* (Figura 4 Procesamiento paralelo en MapReduce del artículo de Google).

En 2006 las implementaciones del *NDFS* y *MapReduce* de *Nutch* se quedan grandes para el proyecto al que pertenecen, por lo que se extraen para crear un subproyecto de *Lucene* con el nombre de *Hadoop*, ya que eran aplicables más allá de los ámbitos de la búsqueda en que se estaban utilizando. En ese mismo año Doug Cutting se unió a *Yahoo!*, lo que le proporcionó un equipo y recursos dedicados para desarrollar su proyecto.

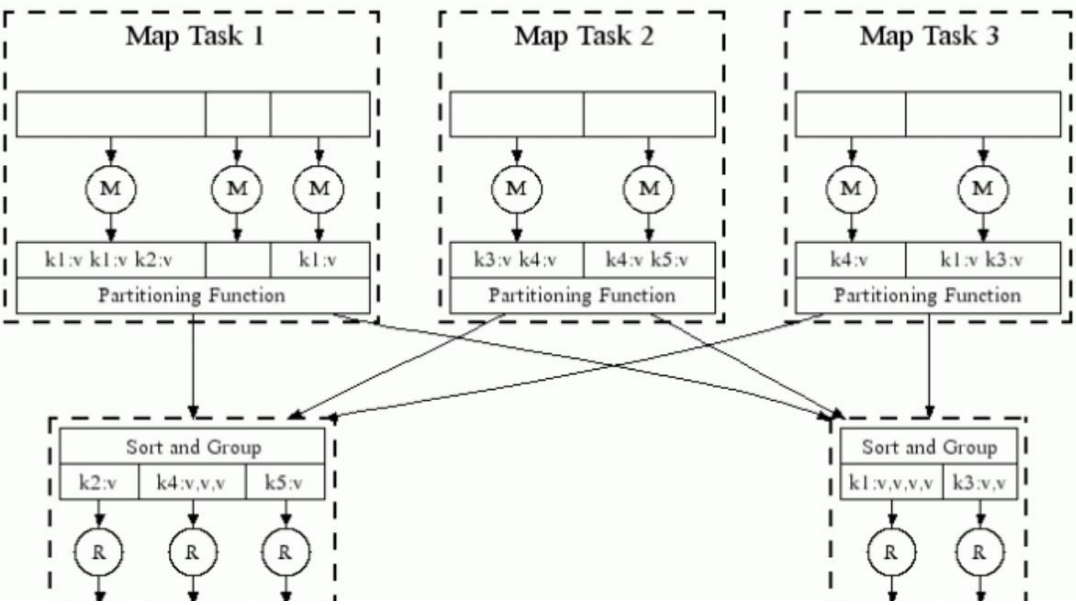



*Figura 4 Procesamiento paralelo en MapReduce del artículo de Google*

Fuente: (Dean & Ghemawat, 2003)

En 2008, *Yahoo!* anunció que su índice de búsquedas era generado por un *clúster Hadoop* de 10.000 *cores*. Por lo que, ese mismo año, *Hadoop* dejó de ser un sub- proyecto y se convirtió en un proyecto de *Apache* de máxima importancia. Por esta época, *Hadoop* ya era utilizado por otras compañías tales como *Last.fm*, *Facebook* y el *New York Times*.

En abril de 2008, *Hadoop* rompía un récord mundial convirtiéndose en el sistema más rápido en ordenar 1 *Terabyte* de datos. Ejecutándose en un *clúster* de 910 nodos 8 lo consiguió en 209 segundos. En noviembre de ese mismo año, *Google* anunció que su implementación de *MapReduce*, podía ordenar ese volumen de datos en tan solo 68 segundos. Y en mayo del año siguiente, un equipo de *Yahoo!* ya lo consiguió en 62 segundos.

Desde entonces *Hadoop* y su ecosistema es un referente en muchos aspectos y sus roles como plataforma de análisis y almacenamiento para grandes datos han sido reconocidos por la industria del sector. Hay distribuciones *Hadoop* tanto en las grandes empresas incluyendo *IBM*, *Microsoft* y *Oracle*, así como compañías especialistas en *Hadoop* tales como *Cloudera*, *Hortonworks* y *MapR*. (White, 2012)

#### 1.2.3.1.2 Funcionamiento

En ésta sección detallaremos el funcionamiento de *Hadoop* comenzando por el *HDFS*, que es la base de todo el sistema *Hadoop.*

##### 1.2.3.1.2.1 HDFS: Hadoop Distributed File System

Es un sistema de ficheros para almacenar grandes archivos de datos y permitir accesos continuos a éstos. Cuando hablamos de grandes archivos, nos

estamos refiriendo a archivos de cientos de *Megabytes*, *Gigabytes* o *Terabytes*. Hoy en día hay *clústers* de *Hadoop* que almacenan *Petabytes* en su sistema. *HDFS* está construido alrededor de la idea de que el patrón de procesamiento de datos más eficientes es el *write-once, read-many-times* (escribir una vez, leer varias veces). El funcionamiento que se suele hacer de un sistema así es que un conjunto de información es generado o copiado desde un origen y luego, sobre esos datos, se ejecutan varios procesos de análisis. Entonces este sistema da más importancia a la lectura de grandes bloques de datos consecutivos antes que a la realización de la primera escritura.

Ya que Hadoop no requiere grandes máquinas a nivel de componentes, está diseñado para ser tolerante a una alta probabilidad de fallos de la máquina. HDFS es capaz de continuar trabajando sin interrupciones perceptibles por el usuario en caso de que se produzca dicho fallo. (White, Hadoop: The definitive guide, 2012)

Para entender un poco mejor cómo funciona, debemos conocer algunos conceptos básicos de *HDFS:* (Shvachko, Kuang, Radia, & Chansler, 2006)

**Bloque:** Un Bloque es la mínima cantidad de datos que se puede leer o escribir. Por regla general los sistemas de ficheros tienen bloques de pocos kilobytes mientras que los bloques de disco son de 512 bytes, esto difiere del concepto que ha tomado HDFS en el cual toman 64MB de valor por defecto del bloque. Por lo tanto, la información que se almacena en el HDFS tiene como unidad mínima el bloque.

La razón de que esto sea así es para minimizar el coste de las búsquedas, ya que, manteniendo un tamaño de bloque grande, el tiempo de transferencia desde disco mejora significativamente. Esto no supone un problema sino un beneficio dado el enfoque del HDFS, almacenar ficheros muy grandes.

***Namenode* y *Datanode:*** El funcionamiento de *HDFS* sigue el patrón *master-workers*. En este caso el *master* es el *Namenode* y los *workers* son los *datanodes.*

El *namenode* es el encargado de gestionar el *namespace* del sistema de ficheros. Mantiene el árbol del sistema de ficheros, así como los metadatos de cada uno de los ficheros y directorios del árbol. También es el encargado de conocer la localización de los bloques que pertenecen a cada fichero dentro de los *datanodes*. Hay que tener en cuenta que esta información no es persistente debido a que se reconstruye desde los *datanodes* cada vez que el sistema inicia. De la misma forma en caso de pérdida de un *datanode*, éste se encarga de mantener la replicación en otro *datanode* y por lo tanto modifica esta información.

Por otro lado, tenemos los *datanodes*, éstos son los encargados de almacenar y obtener la información cuando se les solicita, ya sea por el cliente o por el *namenode*. Con tal de mantener un correcto funcionamiento del sistema, periódicamente se envía la información de los bloques que se están almacenando en ese *datanode* al *namenode*. (Figura 5 Arquitectura básica de Hadoop)

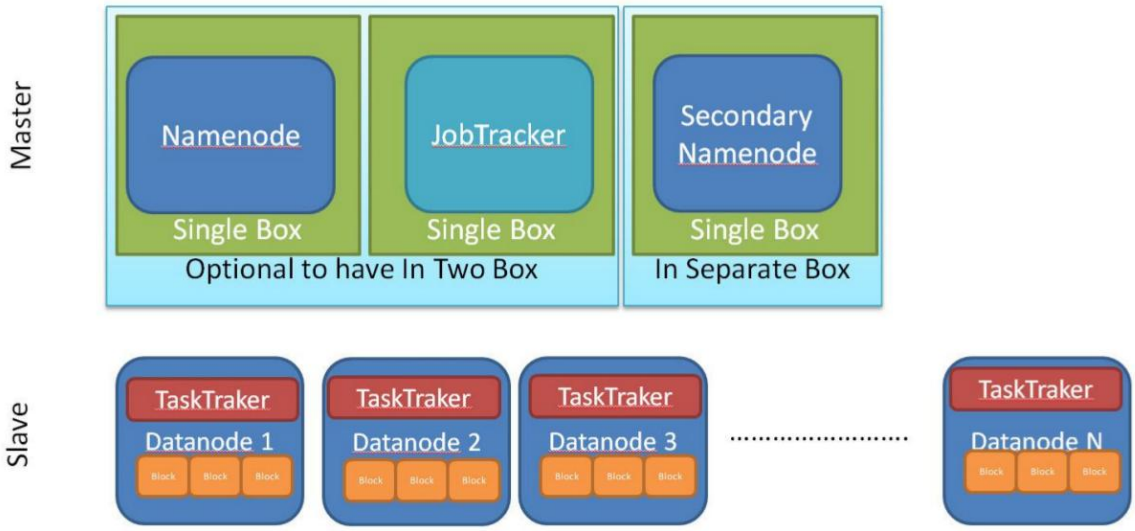



*Figura 5 Arquitectura básica de Hadoop*

Fuente: (Chauhan, 2012)

#### 1.2.3.1.2.2 MapReduce

*MapReduce* es un paradigma de programación que permite procesar y generar grandes cantidades de datos con un algoritmo paralelo y distribuido en un clúster. Fue desarrollado por *Google* para el indexado de páginas web. El objetivo era crear un *framework* adaptado a la computación paralela que permitiera procesar y generar grandes colecciones de datos sobre máquinas genéricas, sin la necesidad de utilizar supercomputadores o servidores dedicados, y que fuera fácilmente escalable.

En esencia, el modelo que propone *MapReduce* es bastante sencillo. El programador debe encargarse de implementar dos funciones, *map* y *reduce*, que serán aplicadas a todos los datos de entrada. Tareas como hacer particiones de los datos de entrada, despliegue de maestro y trabajadores, esquema de asignación de trabajos a los trabajadores, comunicación y sincronización entre procesos, tratamiento de caídas de procesos, quedan a

cargo del entorno de ejecución, liberando de esa manera al programador. (White, Hadoop: The definitive guide, 2012)

El funcionamiento de este paradigma está dividido en dos pasos (para abstracción del programador):

- **Map:** donde se realiza la ingestión y la transformación de los datos de entrada, en la cual los registros de entrada pueden ser procesados en paralelo. El nodo *master* obtiene la entrada, la divide en sub-problemas más pequeños y la distribuye a otros *workers*. Dicho *worker* procesa ese problema más pequeño y produce una lista de pares {clave, valor} y pasa la respuesta a su nodo *master*. Después de esto el *framework* de *MapReduce* junta los pares con la misma clave y los agrupa creando un grupo por cada clave generada.
- **Reduce:** fase de agregación o resumen, donde todos los registros asociados entre sí deben ser procesados juntos por una misma entidad. El nodo *master* coge cada uno de estos grupos y los envía a un nodo para ejecutar la tarea de *Reduce*, por lo que cada sub-tarea reduce trabaja sobre una clave. El reducer trata estos datos y devuelve un output resultado de ello.

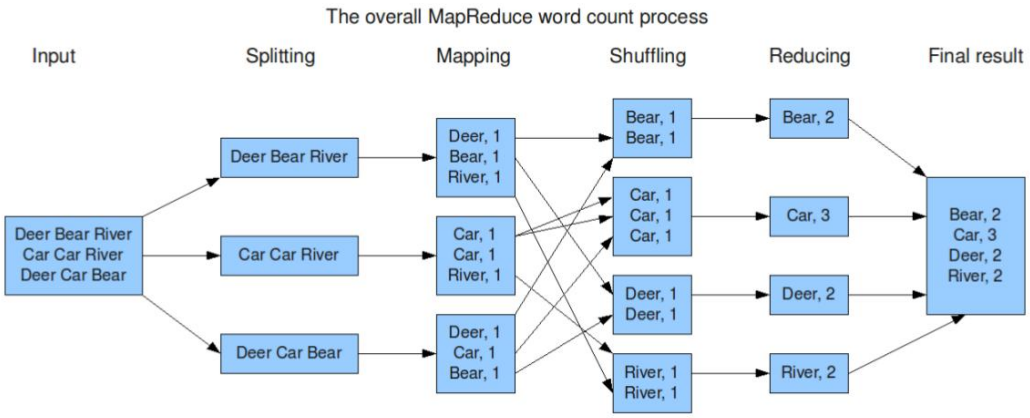



*Figura 6 Conteo de palabras con MapReduce*

Fuente: (Lowell, 2013)

#### 1.2.3.1.2.3 YARN (Yet Another Resource Manager)

Uno de los problemas fundamentales que presenta *Hadoop 1.0* es que solo admite un paradigma de programación: *MapReduce*. A pesar de que este modelo de programación es apropiado para el análisis de grandes conjuntos de datos, en ocasiones es necesario realizar otro tipo de análisis o sería más propicio utilizar otro tipo de software para analizar datos, pero aprovechándonos de la ventaja que proporciona un *clúster Hadoop*. Para intentar solventar este inconveniente surge un nuevo componente fundamental dentro de *Hadoop*: *YARN*.

*Apache Hadoop YARN* es un subproyecto de *Hadoop* en la *Apache Software Foundation* introducido en la versión *Hadoop 2.0* que separa la gestión de recursos de los componentes de procesamiento. *YARN* surge para corregir el inconveniente principal de *Hadoop* y permitir una gama más amplia de modelos de programación para analizar los datos almacenados en el *HDFS* más allá de *MapReduce*. La arquitectura de *Hadoop 2.0* basada en *YARN* provee una plataforma de procesamiento más general y no restringida a *MapReduce*.

En *Hadoop 2.0*, *YARN* toma las capacidades de gestión de los recursos que residían en *MapReduce* y las empaqueta para que puedan ser utilizados por los nuevos motores de procesado. Esto también simplifica la tarea de *MapReduce* en únicamente hacer lo que mejor sabe hacer, tratar datos. Con *YARN*, se permite ejecutar varias aplicaciones en *Hadoop*, todos compartiendo una gestión común de los recursos. *MapReduce* se convierte ahora en una librería *Hadoop* es decir una aplicación que reside en *Hadoop* y deja la gestión de recursos del *clúster* para el componente *YARN*. (Figura 7 Evolución de Hadoop 1.0 a Hadoop 2.0).

La aparición de *YARN* provoca el desarrollo de nuevas herramientas que cubren

múltiples necesidades que únicamente con *MapReduce* no se podían completar. (Figura 8 Nuevas aplicaciones YARN)

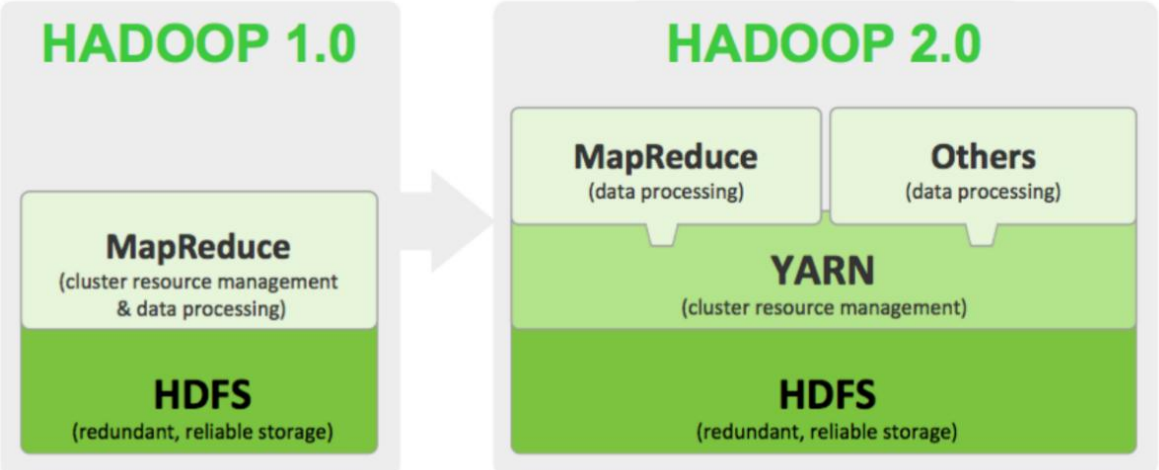



*Figura 7 Evolución de Hadoop 1.0 a Hadoop 2.0*

*Fuente (Murthy, 2013)*

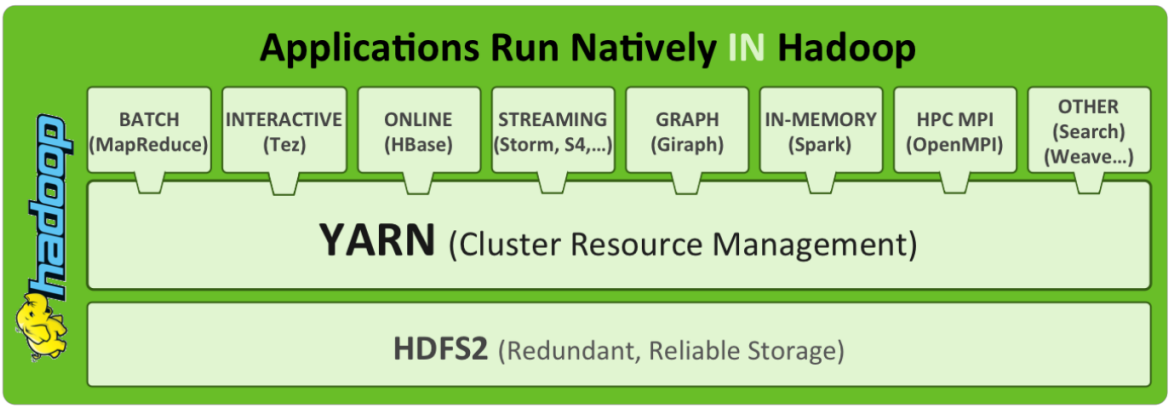



*Figura 8 Nuevas aplicaciones YARN*

*Fuente: (Gutierrez, 2014)*

#### 1.2.3.1.2.4 Arquitectura de *Hadoop 2.0*

La idea fundamental de *YARN* es la de separar las dos mayores responsabilidades del *JobTracker*: la gestión de los recursos y la planificación/monitorización de las tareas en dos servicios separados, para ello tendremos dos nuevos componentes en *YARN*: un *ResourceManager* global y un *ApplicationMaster* por aplicación (AM).

Surgen así el *ResourceManager* para el *master* (sustituyendo al *JobTracker*) y un *NodeManager* (sustituyendo al *TaskTracker*) por cada *slave*. Forman el nuevo, y genérico, sistema de gestión de aplicaciones de una manera distribuida. Además, surge un componente llamado *container* que representa los recursos disponibles en cada nodo del *clúster*.

El *ResourceManager* es la última autoridad que arbitra los recursos entre todas las aplicaciones en el sistema. El *ApplicationMaster* por aplicación es una entidad

específica que se encarga de negociar los recursos con el *ResourceManager* y trabajar con los *NodeManager* para ejecutar y supervisar las tareas que lo componen.

El *ResourceManager* está conectado a un planificador (*Scheduler*) que es responsable de la asignación de recursos a las diversas aplicaciones que se ejecutan con limitaciones conocidas de capacidades, las colas, etc. El *Scheduler* es un planificador puro en el sentido de que no realiza ningún monitoreo o seguimiento del estado de la aplicación. El *Scheduler* realiza su función de planificación en base a las necesidades de recursos de las aplicaciones fundamentándose en la noción abstracta de un *container* de recursos que incorpora elementos de recursos como la memoria, *CPU*, disco, red, etc.

El *NodeManager* se encuentra en cada uno de los equipos esclavos y es responsable del lanzamiento del *container* de las aplicaciones, el seguimiento del uso de recursos (CPU, memoria, disco de red) y de informar del mismo al *ResourceManager*.

El *ApplicationMaster* tiene la responsabilidad de negociar los *containers* de recursos necesarios desde el *Scheduler*, de realizar un seguimiento de su estado y de los progresos de ejecución. Desde la perspectiva del sistema, el propio *ApplicationMaster* se ejecuta como un *container* normal. (Figura 9 Arquitectura Hadoop 2.0 YARN)

A pesar de que los componentes evolucionen, una de las virtudes de *YARN*, es que mantiene el esquema de gestión de recursos utilizado con *MapReduce* lo

que genera que *Hadoop 2.0* esté totalmente integrado en este paradigma de programación.

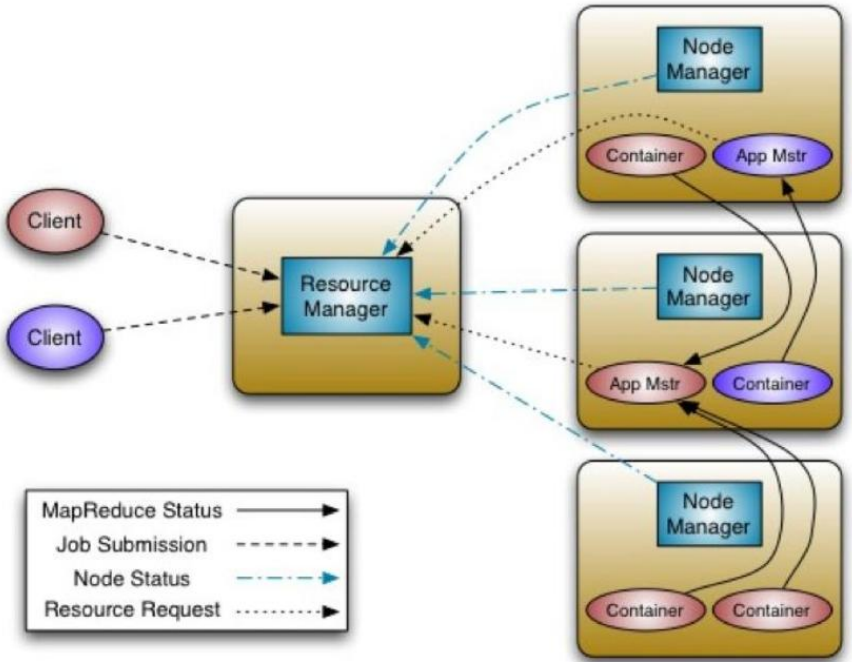



*Figura 9 Arquitectura Hadoop 2.0 YARN*

Fuente: (Apache, Hadoop Yarn, 2016)

#### 1.2.3.1.2.5 Ecosistema Hadoop

El proyecto *Hadoop* consta de una serie de subproyectos, que vienen a complementar su funcionalidad profundizando en aspectos como el tratamiento, flujo e importación de datos, la monitorización de trabajos, etc. Existen multitud de proyectos relacionados con *Hadoop* que completan distintas necesidades (Figura 10 Ecosistema Hadoop). La mayoría son dirigidos por *Apache,* aunque empresas privadas como *Cloudera* o *Hortonworks* trabajan desarrollando todo este tipo de plataformas.

A continuación, vamos a presentar los proyectos relacionados con *Hadoop* más importantes:

- **Ambari:** Una herramienta basada en web para el aprovisionamiento, administración y seguimiento de clústeres *Apache Hadoop*, que incluye soporte para *Hadoop HDFS*, *Hadoop MapReduce*, *Colmena*, *HCatalog*, *Hbase*, *ZooKeeper*, *Oozie*, *Pig y Sqoop*. *Ambari* también proporciona un panel de control para la visualización del estado del clúster, así como la capacidad de ver aplicaciones como *MapReduce*, *Pig* y *Colmena* con el objetivo de evaluar su rendimiento de una manera sencilla.
- **Avro:** se trata de un sistema de serialización de datos que provee numerosas estructuras de datos, un formato de datos binario compacto y rápido, un archivo contenedor para almacenar datos persistentes y una sencilla integración con lenguajes dinámicos.
- **Cassandra:** *Apache Cassandra* es una base de datos distribuida de segunda generación altamente escalable, que reúnele diseño totalmente distribuido de *Dynamo* y el modelo de datos basado en *ColumnFamily* de *Bigtable*. *Cassandra* se usa en *Facebook, Digg, Twitter, Mahalo, Ooyala, SimpleGeo, Rackspace*, y otras empresas que necesitan de una base de datos con alta escalabilidad, disponibilidad y tolerancia a fallos.
- **Chukwa:** es un sistema de recopilación de datos de código abierto para el seguimiento de grandes sistemas distribuidos. *Chukwa* hereda la escalabilidad y robustez de *Hadoop*. Además, incluye un conjunto de herramientas flexibles y potentes para la visualización, seguimiento y análisis de resultados para hacer el mejor uso de los datos recogidos.
- **HBase:** Una base de datos escalable, distribuida que soporta el almacenamiento de datos estructurados en tablas. Permite la realización de tablas a partir de ficheros de datos.
- **Hive:** facilita la consulta y gestión de grandes conjuntos de datos que residen en almacenamiento distribuidos. *Hive* proporciona un mecanismo para la ver la estructura de los datos utilizando un lenguaje similar a *SQL* llamado *HiveQL*.
- **Mahout:** se trata de un *software* libre centrado en la implementación de algoritmos de *machine learning* distribuidos.

- **Pig:** *Apache Pig* es una plataforma para el análisis de grandes conjuntos de datos que se caracteriza por un lenguaje de alto nivel para la creación de los programas de análisis de datos, junto con la infraestructura necesaria para la evaluación de estos programas. La propiedad más importante de los programas *Pig* es que su estructura es susceptible de una paralelización sustancial, lo que a su vez permite manejar grandes conjuntos de datos.
- **Spark:** proporciona un motor de cálculo rápido y general para datos *Hadoop*. *Spark* proporciona un modelo de programación sencillo y expresivo que soporta una amplia gama de aplicaciones, incluyendo *ETL*, *machine learning*, procesamiento de flujo, y computación gráfica.
- **ZooKeeper:** es un servicio centralizado construido para mantener la información de configuración, proporcionar sincronización distribuida y la prestación de servicios de grupo. Todos estos tipos de servicios se utilizan de una forma u otra por las aplicaciones distribuidas.

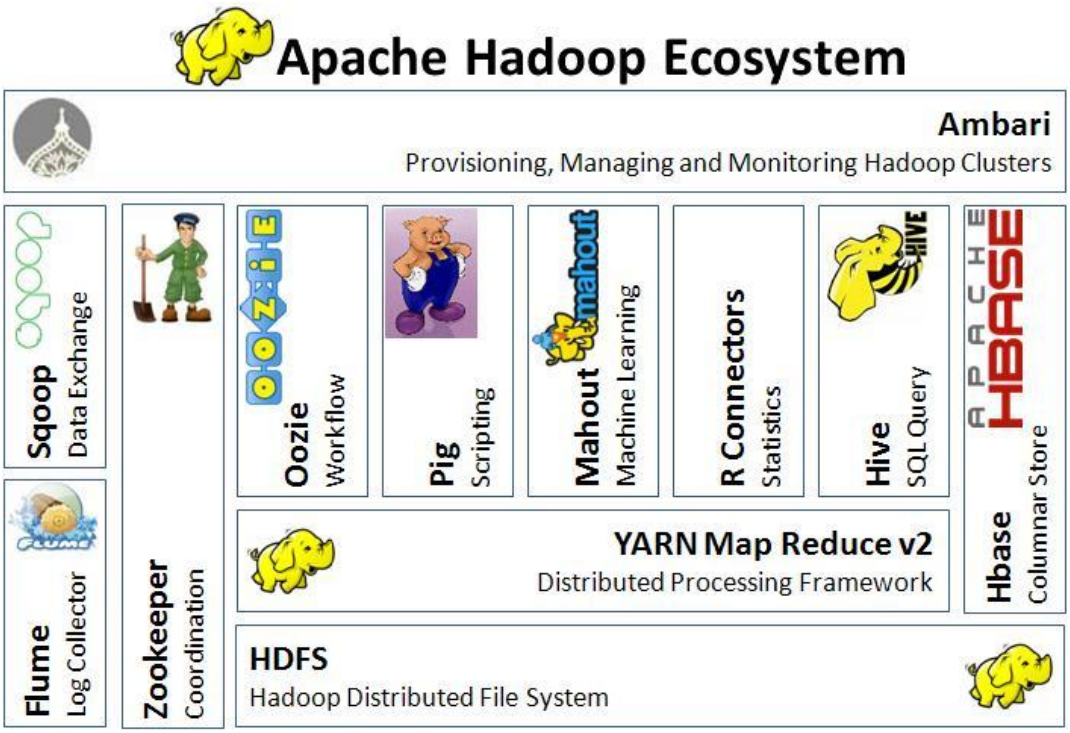



*Figura 10 Ecosistema Hadoop*

*Fuente: (Blog, 2016)*

### 1.2.3.2 NoSQL (*Not Only SQL*)

#### 1.2.3.2.1 Introducción

Los sistemas manejadores de bases de datos relacionales son la tecnología predominante para almacenar datos estructurados en la web y aplicaciones de software en los negocios. Desde la publicación científica de Codd "*A relational model of data for large shared data banks*" de 1970 estos almacenes de datos apoyándose en el cálculo relacional y proveyendo consultas comprensivas ad hoc facilitadas por SQL han sido ampliamente adoptados y son comúnmente la única alternativa para el almacenamiento de datos accesibles a múltiples clientes de manera consistente. Los RDMBS siguen una filosofía "una talla para todos" lo que quiere decir que el modelo relacional sirve para todos los casos, sin embargo, en los últimos años ésta filosofía ha sido cuestionada por la ciencia y compañías relacionadas con la web, lo que ha llevado al surgimiento de una gran variedad de alternativas en bases de datos. Las cuales no siguen el modelo relacional propuesto por Frank Codd ya que poseen un esquema flexible y cada una surge para solventar una clase de problema en específico. Este movimiento es lo que se conoce como *NoSQL* (*Not only SQL*).

El término NoSQL fue usado por primera vez en 1998 para una base de datos relacional que omitía el uso de SQL (Strozzi, 2010). El término fue utilizado de nuevo en el año 2009 para una conferencia realizada en San Francisco acerca de bases de datos que no seguían el modelo relacional, tales como *Last.fm* desarrollada por Jon Oskarsson quien fue el encargado de organizar el evento. (Evans, 2009)

#### 1.2.3.2.2 Críticas a las Bases de Datos *NoSQL*

Desde la perspectiva de los adeptos a los *RDBMS* podemos mencionar las siguientes críticas a las bases de datos *NoSQL*: (Antiñanco, 2013)

- **No hay un líder:** El mercado de *NoSQL* está muy fragmentado, lo cual es un problema para el *open-source* porque se requiere una gran cantidad de desarrolladores para tener *éxito*.
- **No hay estandarización:** Cada base de datos posee su propia interfaz y tipo de consultas, lo que ocasiona que la adaptación a una base de datos *NoSQL* tenga una inversión significativa para poder ser utilizada.
- **Se requiere una reestructuración de los modelos de desarrollo de aplicaciones:** Utilizar una base datos *NoSQL* típicamente implica usar un modelo de desarrollo de aplicaciones diferente a la tradicional arquitectura de 3 capas. Por lo tanto, una aplicación existente de 3 capas no puede ser simplemente convertida para bases de datos *NoSQL*, debe ser reescrita, sin mencionar que no es fácil reestructurar los sistemas para que no ejecuten consultas con *join* o no poder confiar en el modelo de consistencia *read-after-write*
- **Modelos de datos sin esquema podría ser una mala decisión de diseño:** Los modelos de datos sin esquema son flexibles desde el punto de vista del diseñador, pero son difíciles para consultar sus datos.

#### 1.2.3.2.3 Puntos a favor de las *Bases de Datos NoSQL*

Desde la visión de los adeptos a las bases de datos NoSQL podemos mencionar las siguientes razones para desarrollar y utilizar ésta tecnología:

- **Evitar la complejidad innecesaria:** Los RDBMS proveen un conjunto amplio de características y obligan el cumplimiento de las propiedades ACID, sin embargo, para algunas aplicaciones éste set podría ser excesivo y el cumplimiento estricto de las propiedades ACID innecesario.

- **Alto rendimiento:** Las bases de datos NoSQL proveen un rendimiento mayor a las relacionales, incluso hasta varios órdenes de magnitud.
- **Información no estructurada y hardware más económico:** La mayoría de las bases de datos NoSQL son diseñadas para poder escalar horizontalmente. También permiten el almacenamiento de datos no estructurados, provenientes de diversas fuentes como pueden ser las redes sociales.
- **La filosofía "One size fits all" estaba y sigue estando equivocada:** Existen muchos escenarios que no pueden ser abarcados con un enfoque de base de datos tradicional. Esto debido al continuo crecimiento de volúmenes de datos a ser almacenados y a la necesidad de procesar grandes volúmenes de datos en corto tiempo.

#### 1.2.3.2.4 Taxonomía *NoSQL*

De acuerdo a la manera en que las bases de datos *NoSQL* almacenan sus datos es posible clasificarlas de la siguiente manera:

- Almacenamiento clave-valor
- Bases de datos orientadas a columnas
- Base de datos documentales
- Base de datos orientada a grafos

#### 1.2.3.2.5 Propiedades *ACID vs BASE*

En esta sección se comparan las propiedades de los sistemas relacionales (ACID) con la de los sistemas no relacionales (BASE). Ademas, de una breve explicación del Teorema de CAP o Brewer, el cual es la base para el desarrollo de este tipo de tecnologías.

##### 1.2.3.2.5.1 Teorema de *CAP*

Durante el simposio de “Principios de computación distribuida” de ACM en el año 2000, Eric Brewer, un profesor de la universidad Berkeley de California y cofundador de Inktomi, a través de una presentación titulada “Hacia sistemas distribuidos robustos”, estableció la conjetura que los servicios web no pueden asegurar en forma conjunta las siguientes propiedades: Consistencia (C), Disponibilidad (A) y Tolerancia a particiones (P), esto es lo que se conoce como el teorema de CAP. Posteriormente en el año 2002, Seth Gilbert y Nancy Lynch de MIT publicaron una demostración formal de la conjetura de Brewer, convirtiéndola en un teorema. El teorema de CAP establece que, en un sistema distribuido con datos compartidos, se debe optar por favorecer dos de las tres características: Consistencia, Disponibilidad y Tolerancia a particiones. Bajo estas restricciones, Brewer indica que se debe utilizar como criterio de selección, los requerimientos que se consideren más críticos para el negocio, optando entre propiedades *ACID y BASE.* (Brewer, 2000).

En la siguiente tabla basada en la presentación de Brewer se muestran las alternativas, características y ejemplos (Tabla 1 Alternativas en Teorema de CAP)

*Tabla 1 Alternativas en Teorema de CAP*

*Fuente: (Antiñanco, 2013)*

| Alternativa | Características | Ejemplos |
|---|---|---|
| **CA:** Consistencia + Disponibilidad (sacrificando Tolerancia a particiones) | ❖ Protocolos de commit en 2 fases<br>❖ Protocolos de validación de caché | ❖ Bases de datos centralizadas<br>❖ Cluster de bases de datos<br>❖ LDAP<br>❖ Sistema de archivos xFS |
| **CP:** Consistencia + Tolerancia a particiones (sacrificando Disponibilidad) | ❖ Mecanismos de bloqueo pesimista<br>❖ Protocolos de Quorum mayoritario: Paxos | ❖ Bases de datos distribuídas<br>❖ Bloqueo distribuído |
| **AP:** Disponibilidad + Tolerancia a particiones (sacrificando Consistencia) | ❖ Protocolos de resolución de conflictos: Gossip<br>❖ Manejo de expiraciones y leasing<br>❖ Enfoque optimista | ❖ Sistema distribuido Coda<br>❖ Web cache<br>❖ DNS |

#### 1.2.3.2.5.2 Propiedades ACID en sistemas distribuidos

Las propiedades ACID presentes en las transacciones de las bases de datos relacionales simplifican el trabajo de los desarrolladores de aplicaciones al ofrecer garantías en cuanto a:

- **Atomicidad:** Todas las operaciones en la transacción se completarán o ninguna lo hará.
- **Consistencia:** El estado de la base de datos se mantiene válido tanto al inicio como al final de la transacción. Cualquier operación puede ver los cambios en la base de datos.
- **Aislamiento:** Las transacciones se ejecutan de manera que una no puede afectar a la otra.
- **Durabilidad:** Los cambios realizados durante una operación serán persistentes y no se podrá deshacer, aunque falle el sistema.

Los proveedores de bases de datos se percataron de la necesidad de particionar, por lo que introdujeron el protocolo de *commit* a 2 fases para seguir manteniendo las propiedades ACID en las instancias de las bases de datos. El cual consiste en:

- **Primera fase:** el coordinador de la transacción solicita a cada base de datos involucrada que indiquen si es posible que realicen *commit.* Si es posible se procede a continuar con la segunda fase.
- **Segunda fase:** El coordinador de la transacción solicita a cada base de datos que realice el *commit*.

Sin embargo, si alguna base de datos no puede realizar el commit, se le solicita a todas las involucradas que realicen un *roll-back* de ésa transacción. Sería análogo a que un avión no pudiese despegar hasta que todos los pasajeros estén en sus asientos, a pesar de que el vuelo haya sido programado para una hora en específico. En vista de esto podemos observar que se generan retrasos al realizar las transacciones, lo que afecta, de acuerdo al teorema de CAP, la disponibilidad del sistema. (Pritchett, 2008)

#### 1.2.3.2.5.3 Propiedades *BASE*

Si las propiedades *ACID* se enfocan en la consistencia, las propiedades *BASE* se enfocan en la disponibilidad. La palabra *BASE* se refiere a básicamente disponible (BA), estado flexible (S) y eventualmente consistente (E).

Las propiedades *BASE* son diametralmente opuestas a las propiedades *ACID*. Donde *ACID* es pesimista y fuerza la consistencia al finalizar cada operación, *BASE* es optimista y acepta que la consistencia de la base de datos estará en un estado flexible. Aunque esto suene imposible de lidiar, en la realidad es manejable y puede llevar a niveles de escalabilidad que no se pueden obtener con *ACID*.

La disponibilidad en las propiedades *BASE* es alcanzada a través de mecanismos de soporte de fallas parciales, que permite mantenerse operativos

y evitar una falla total del sistema. Así, por ejemplo, si la información de usuarios estuviera particionada a través de 5 servidores de bases de datos, un diseño utilizando *BASE* alentaría una estrategia tal que una falla en uno de los servidores impacte sólo en el 20% de los usuarios de ese *host.*

*Tabla 2 ACID vs BASE (Brewer, 2000)*

| **ACID** | **BASE** |
|---|---|
| Consistencia fuerte | Consistencia débil |
| Aislamiento | Disponibilidad primero |
| Centrado en “commit” | Mejor esfuerzo |
| Transacciones anidadas | Respuesta aproximada |
| ¿Disponibilidad? | Agresivo (optimista) |
| Conservativo (pesimista) | Más simple |
| Difícil evolución (ejemplo: el esquema) | Más rápido |
| | Fácil Evolución |

### 1.2.3.3 Motores de Búsqueda

Los motores de búsqueda son programas que permiten hacer búsquedas por palabras claves y retornan una lista de documentos en donde se encontraron las palabras claves. Funcionan mediante la creación de índices con los cuales luego se realizan las búsquedas y a diferencia de las bases de datos, permiten integrar búsquedas de diversas fuentes de datos, permiten la escalabilidad, ranking por relevancia y hacer búsquedas por facetas. (Wikipedia, Motores de busqueda, 2015)

Estas herramientas son necesarias para facilitar las búsquedas a los usuarios, ya que al hablar de grandes volúmenes de datos como es el caso de la web es muy probable que el usuario final encuentre información que no le sea de utilidad, por lo cual estas herramientas se manejan con análisis de contenido

como es el caso del algoritmo de *Google*, *Page Rank*, en el cual se hace un ranking de las páginas más importantes dependiendo de la búsqueda realizada.

Los principales motores de búsqueda en BigData son Apache Lucene y Apache SolR, los cuales se explicarán brevemente:

#### 1.2.3.3.1 Lucene

*Apache Lucene* es una API de código abierto para la recuperación de información, implementada originalmente en *Java* por Doug Cutting. Se trata de una tecnología adecuada para cualquier aplicación que requiera de búsquedas por texto completo, especialmente si son multiplataforma (Apache, 2016)

Puede indexar cualquier formato de texto, como MS Word, HTML, XML, PDF y *OpenDocument*, siempre y cuando la información textual pueda ser extraída, lo que quiere decir que no puede indexar imágenes.

Algunas características:

- Es escalable y tiene indexación de alto rendimiento, ya que puede procesar 150 GB / Hora en máquinas modernas y requiere solo 1 MB de memoria
- Potente, preciso y eficiente algoritmo de búsqueda. Búsqueda por ranking, mejores resultados devueltos primero. Diversos tipos de consultas como son: consultas por frases, por comodín, por proximidad, rangos y más.
- Multiplataforma. Esta implementado 100 por ciento en Java, aunque también está disponible en otros lenguajes de programación.
- Puede ordenar cualquier campo en cualquier documento
- Internamente todo se maneja como un documento y no necesariamente tiene que referirse a un archivo real en disco, también podría asemejarse a una fila de una base de datos relacional.

- Un documento es visto como una lista de campos, donde un campo tiene un nombre y un valor.
- La unidad de indexación en *Lucene* es un término, el cual puede ser a menudo una palabra. Los índices rastrean las frecuencias de los términos
- *Lucene* utiliza índices inversos que permiten localizar rápidamente los documentos asociados a la condición entrante de búsqueda.

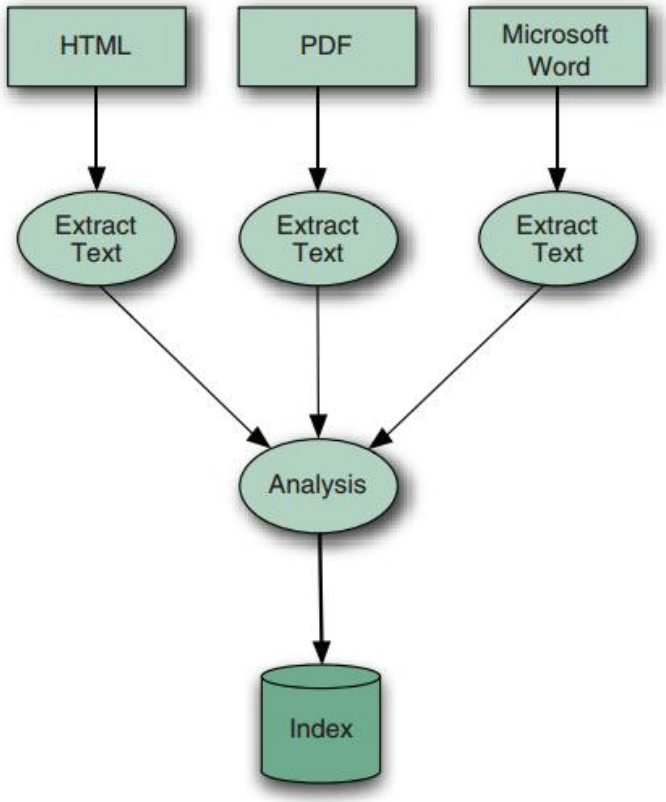



*Figura 11 Operaciones de indexación con Lucene*

*Fuente: (MacCandles, Hatcher, & Gospodnetic, 2010)*

Algunas ventajas:

- Poderosa sintaxis de búsqueda
- Rápido indexamiento
- Búsqueda rápida

- Ordenamiento por relevancia y por otros tipos de campos

Algunas desventajas:

- No hay contratos formales de apoyo
- No hay disponibilidad asegurada de formación u otros servicios profesionales para satisfacer las necesidades específicas del software o ayudar con la construcción de una aplicación.
- Ningún programa de pruebas de liberación formalizado, calendario de lanzamiento o garantía de compatibilidad de actualización.

#### 1.2.3.3.2 SolR

*Solr* es un motor de búsqueda de código abierto, basado en *Lucene*, que permite el resaltado de resultados y la búsqueda por facetas, además posee una interfaz para su administración. Se ejecuta sobre un contenedor de *servlets Java* como *Apache Tomcat* (Wikipedia, Apache Solr, 2016) . Solr es escalable, permitiendo búsquedas distribuidas y replicación de índices

La principal característica de *Solr* es su API estilo REST, ya que en vez de usar drivers para comunicarnos con *Solr*, podemos hacer peticiones HTTP y obtener resultados en XML o JSON.

Algunas de sus características son:

- Se comunica a través de HTTP enviando y recibiendo contenido en XML
- Esta optimizado para un alto volumen de tráfico web.
- Soporte de indexación distribuida (SolrCloud), replicación y la carga de consultas equilibradas.
- Búsqueda por facetas
- Análisis de texto (tokenización, normalización)
- Permite realizar búsquedas geoespaciales.

- Lee y escribe directamente al HDFS de Hadoop, además de soportar la replicación en el HDFS.
- Es posible construir índices escalables a través del paradigma Map Reduce

En su arquitectura Solr se divide en dos partes: (ver Figura 12 Arquitectura Solr)

- **Índice.** Sistema de ficheros que almacenan la información. Contiene la configuración de Solr y la definición de la estructura de datos.
- **Servidor.** Proporciona el acceso a los índices y las características adicionales. Admite *plugins* para añadir funcionalidades.

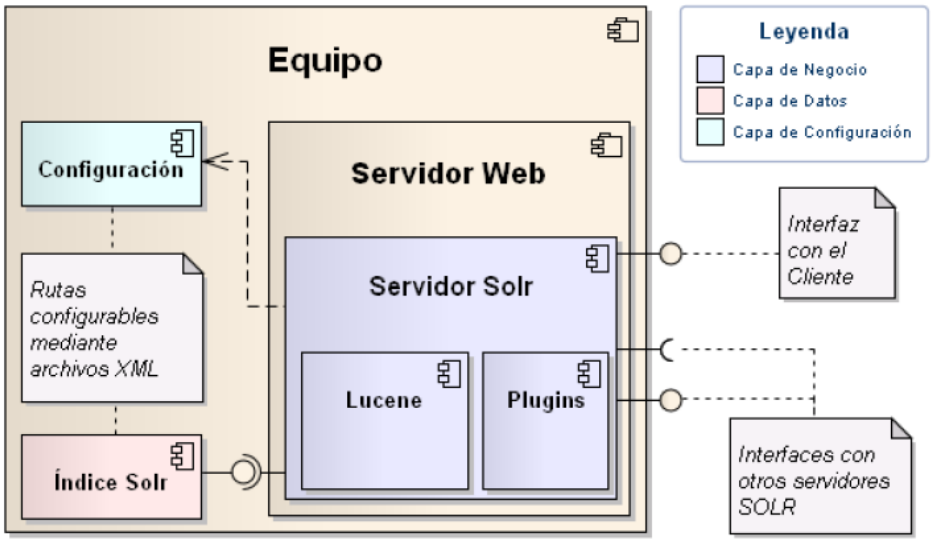



*Figura 12 Arquitectura Solr*

Fuente: (Marco, 2013)

### 1.2.3.4 Herramientas de visualización y análisis

Una vez que se tienen todos estos datos en la arquitectura, se utilizan herramientas de visualización y análisis que permiten sacarle el valor a los

datos, con los cuales es posible, mediante la aplicación de funciones estadísticas y de visualización, encontrar patrones, tendencias, hacer modelos predictivos, e incluso sistemas de recomendación.

La importancia de la visualización de los datos radica en que permite al analista de datos, bien sea un gerente o un científico de datos, facilitarles la comprensión de los datos, de modo que puedan realizar sus funciones con mayor facilidad y puedan a su vez transmitir la información que encontraron en los datos.

Algunas herramientas de visualización más importantes podemos mencionar Apache Hue (Hadoop User Experience) y Banano.

Entre las herramientas de análisis más conocidas en el mundo del Big Data se encuentran:

- *RStudio* que es más conocida en el ámbito científico por su manejo de lenguaje de análisis R, el cual es muy utilizado por científicos de datos.
- Podemos mencionar *Apache Mahout*, el cual me permite utilizar algoritmos en su librería para hacer máquinas de aprendizaje
- *Apache Spark* el cual se integra con *Hadoop* y posee un módulo llamado *Spark MILIB* en el cual también se utiliza para hacer *machine learning*, además de poseer módulos para hacer consultas en SQL y análisis de grafos, con la peculiaridad de que *Spark* utiliza la memoria RAM de la máquina, lo que lo hace una herramienta más rápida que utilizar solo *Hadoop*.

## 1.3 Ciencia de Datos

La ciencia de datos es un campo interdisciplinario de la ciencia que involucra los procesos y sistemas para extraer conocimiento de grandes volúmenes de datos en sus diferentes formas (estructurados o no estructurados) y formatos

(textos, imágenes, documentos, vídeos, entre otros), mediante el uso de la computación, la matemática y la estadística.

A pesar de que se puede aplicar ciencia de datos, sin la necesidad de tener una arquitectura *Big Data* y viceversa, en los últimos años hemos visto cómo es posible combinar ambas de manera armoniosa para solventar problemas, mejorar servicios o incluso prever eventos mediante el uso de modelos predictivos.

En la ciencia de datos se involucran 3 principales componentes de acuerdo a Drew Conway los cuales son (Figura 13 Diagrama de Venn Ciencia de datos):

- Habilidades *Hacking* o computacionales
- Conocimiento Estadístico y matemático
- Conocimientos sustantivos (área en la que se investiga).

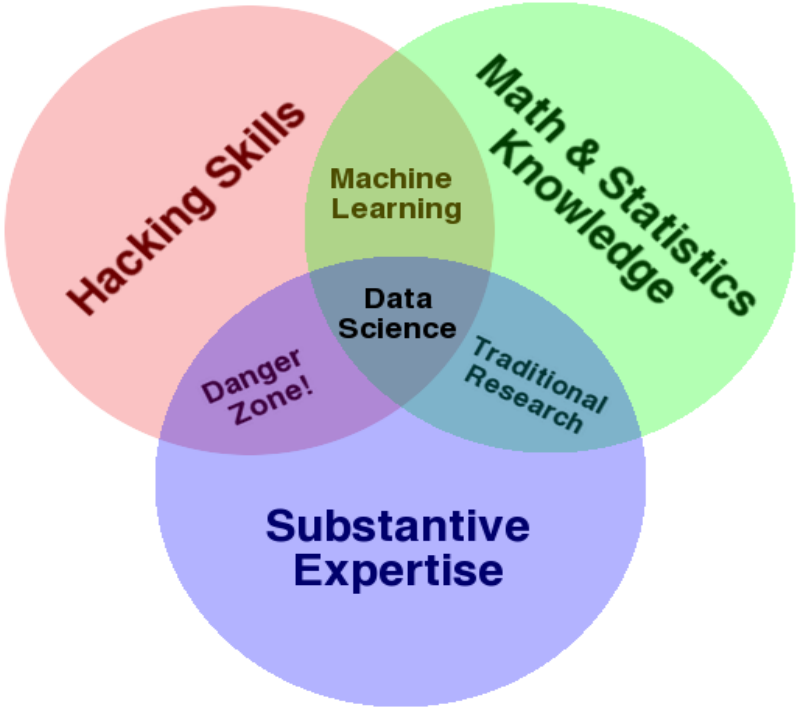



*Figura 13 Diagrama de Venn Ciencia de datos*

Fuente: (Conway, 2010)

Los científicos de datos son uno de los mayores beneficiados por el uso del *Big Data*, ya que de ésta manera disponen de mayor cantidad de datos a analizar y de las herramientas necesarias para sus respectivos análisis.

Entre las funciones que realizan los científicos de datos, podemos resumirlas en 3 principalmente**:** (Sánchez, 2014)

- **Captura de los datos:** Captura y almacenamiento de la información. Es el procedimiento manual de convertir "*raw data*" (información en bruto) en información con formato para que pueda ser analizada.
- **Análisis de los datos:** Obtención de valor a partir de la información. Para lograrlo es necesario realizar procesos de minería de datos como limpieza o transformaciones, luego aplicar funciones estadísticas, experimentar con modelos predictivos y mediante el cálculo de errores observar cuál es el que se adapta mejor al problema a resolver.
- **Visualización de los datos:** Visualización de los resultados obtenidos anteriormente. En muchos casos el científico de datos debe dirigirse con un lenguaje más accesible para explicar los resultados obtenidos, por lo que requiere de habilidad para expresar sus investigaciones de manera clara y entendible apoyándose en el uso de gráficos que puedan ayudarle a comunicar los resultados obtenidos.

Hoy en día es posible ver muchos casos de éxito en la aplicación de la ciencia de datos en conjunto con *Big Data,* bien sea para evitar epidemias, promocionar productos, evitar riesgos financieros, análisis y alertas de datos en tiempo real de pacientes, fraudes en tarjetas de crédito, entre otros.

## 1.4 Arquitectura de Software

Todos los sistemas tienen una arquitectura, es decir, una estructura de alto nivel que define todo el sistema. En la arquitectura de software se observa

como todas las piezas que componen el sistema encajan para aportar una solución al problema que se desea resolver el cliente.

Podemos definir la arquitectura de software de un programa o sistema computacional como la estructura de las estructuras del sistema, la cual comprende elementos de software, las propiedades externamente visibles de esos elementos y las relaciones entre ellos. (Bass, Clements, & Kazman, 2003)

### 1.4.1 Importancia de la Arquitectura de Software

Existen fundamentalmente 3 razones que dan importancia a las arquitecturas de software:

- **Comunicación entre los *stakeholders*:** La arquitectura de software representa un medio de abstracción del sistema, en la cual la mayoría de los *stakeholders* pueden usar como una base para el entendimiento mutuo, negociar, comunicarse y llegar a consensos.
- **Primeras decisiones en diseño de sistemas:** Las arquitecturas de software representan un primer ajuste en las decisiones de diseño (define restricciones en la implementación, dicta la estructura organizacional, ayuda en la evolución de prototipos, ayuda a una mayor estimación de los costos monetarios y en tiempo)
- **Abstracción de sistemas transferibles:** Proporcionan un modelo de como un sistema está estructurado y de cómo sus elementos interactúan entre sí. Este modelo puede ser transferido entre sistemas.

### 1.4.2 Componentes en una arquitectura de software

La arquitectura de software provee una abstracción de alto nivel del sistema a ser construido, por lo que debe cubrir los siguientes aspectos: (Hanmer, 2013)

- **Filosofía y objetivo del sistema:** La arquitectura explica los objetivos y describe el propósito del sistema, quien lo usa y que problema resuelve.
- **Suposición arquitectural y dependencias:** La arquitectura explica las suposiciones hechas del sistema con su medio ambiente. También explica cualquier dependencia con otro sistema.
- **Requerimientos significativos de la arquitectura**
- **Instrucciones de los subsistemas y componentes:** Explica como los componentes del sistema son desplegados en plataformas computacionales y como se deben combinar para su correcto funcionamiento.
- **Subsistemas críticos y capas:** La arquitectura describe las diferentes vistas y partes del sistema y como se relacionan. También explica en detalle sus subsistemas más críticos.
- **Interfaces criticas del sistema**
- **Escenarios claves que describen el comportamiento del sistema**

### 1.4.3 Estructuras de la arquitectura y vistas

Un neurólogo, un cardiólogo y un hematólogo tendrán diferentes vistas de las estructuras del cuerpo humano, cada uno de acuerdo a su respectiva especialidad, lo mismo ocurre en el mundo del software. Los sistemas modernos computacionales son tan complejos que es muy difícil entender todas sus estructuras como un todo, por lo que se diseñan y estudian modularmente.

Las estructuras arquitecturales pueden dividirse en 3 grupos principales, dependiendo de la naturaleza de elementos que muestren:

- **Estructuras modulares:** Aquí los elementos son módulos, que son unidades de representación, los cuales se le asignan responsabilidades

funcionales. Nos permiten responder preguntas tales como: ¿Cuál es la función principal asignada a cada módulo? ¿Qué elementos de software puede usar cada módulo? ¿Qué módulos están relacionados con otros módulos por especialización o generalización?

- **Estructuras de componentes y conectores:** Aquí los elementos son componentes de ejecución y sus conectores. Estas estructuras nos permiten responder preguntas tales como: ¿Cuáles son los mayores componentes de ejecución y cómo interactúan? ¿Qué partes del sistema están replicadas? ¿Cómo la data progresa a través del sistema? ¿Cuáles son los mayores almacenes de datos compartidos?
- **Estructuras de asignación:** Las estructuras de asignación muestran cuales son las relaciones entre los elementos de software y los elementos en uno o más ambientes externos en los que el software es creado y ejecutado. Nos permite responder preguntas como ¿En qué procesador se ejecuta cada elemento de software? ¿En qué archivos se almacena cada elemento durante su desarrollo, prueba y creación del sistema? ¿Cuáles son los elementos de software asignados a cada equipo de desarrollo?

Estas tres estructuras corresponden a tres tipos de decisiones que envuelven el diseño de la arquitectura, como son:

- ¿Cómo se estructura el sistema como un conjunto de unidades de código? (módulos)
- ¿Cómo se estructura el sistema como un conjunto de elementos que tienen un comportamiento de ejecución (componentes) y sus interacciones (conectores)?
- ¿Cómo el sistema se relaciona con estructuras que no son de software en su medio ambiente? (CPU, archivos, equipos de desarrollo, entre otros).

### 1.4.3.1 Estructuras de software

Algunas de las estructuras de software más comunes se muestran en la siguiente figura (ver Figura 14 Estructuras de software)

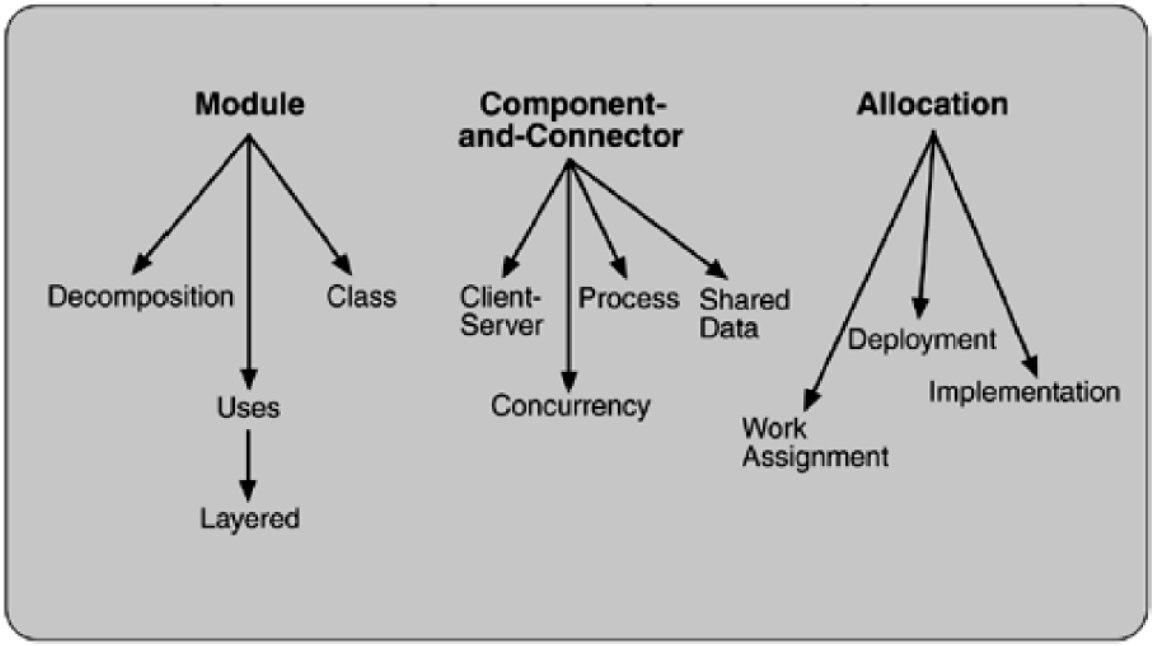



*Figura 14 Estructuras de software*

Fuente: (Bass, Clements, & Kazman, 2003)

#### 1.4.3.1.1 Módulos

Las estructuras modulares incluyen lo siguiente:

- **Descomposición.** Las unidades son módulos relacionados entre sí por la relación “es submodulo de”, mostrando como módulos grandes son descompuestos en unos más pequeños recursivamente, hasta que son los suficientemente pequeños para ser fácilmente comprendidos.
- **Usos.** Las unidades de estas estructuras también son módulos, o procedimientos o recursos en las interfaces de los mismos. Las

unidades son relacionadas por una relación de “uso”. Una unidad utiliza otra si la primera necesita la presencia de la segunda. Estas estructuras son utilizadas para diseñar sistemas que puedan ampliarse fácilmente para añadir funcionalidades o de la que subconjuntos funcionales útiles se puedan extraer fácilmente.

- **Capas.** Cuando las relaciones de usos en esta estructura son controladas de forma particular, un sistema de capas emerge, en el cual una capa es un conjunto de funcionalidades relacionadas.
- **Clases, o generalización.** Las unidades de módulo de estas estructuras, son llamadas clases. La relación es “hereda de” o “es una instancia de”. Esta vista permite el razonamiento de colecciones con el mismo comportamiento o capacidad y parametrizar diferencias que son capturadas por subclases.

#### 1.4.3.1.2 Componentes y conectores

Estas estructuras incluyen lo siguiente:

- **Procesos.** Las unidades aquí son procesos o hilos que están conectados entre sí por la comunicación, sincronización, y/o exclusión de operaciones. Las estructuras de procesos ayudan a diseñar en los sistemas el rendimiento de ejecución y disponibilidad.
- **Concurrencia.** Estas estructuras ayudan a los arquitectos a determinar oportunidades de paralelismo y la locación donde la contención de recursos pueda ocurrir. Las unidades son componentes y los conectores son “hilos lógicos”
- **Datos compartidos o repositorios.** Esta estructura incluye componentes y conectores que crean, almacenan y acceden a data persistente. Muestra como la data es producida y consumida por elementos de software en tiempo de ejecución y puede ser usado para asegurar un buen rendimiento e integridad de la data.

- **Cliente-Servidor.** Los componentes son clientes-servidores y lo conectores son los protocolos que comparten para el pase de mensajes. Si el sistema está construido como un grupo de clientes y servidores que cooperan entre sí, esta es una buena estructura de componentes y servidores a ser utilizada.

#### 1.4.3.1.3 Asignación

Las estructuras de asignación incluyen lo siguiente:

- **Despliegue.** Las estructuras de despliegue muestran cómo se asigna el software a los elementos de hardware de procesamiento y de comunicación. Esta vista permite al ingeniero razonar acerca del rendimiento, integridad de la data, disponibilidad y seguridad. Es de un interés particular en sistemas distribuidos o paralelos.
- **Implementación.** Muestran como las estructuras de software (usualmente módulos) son asignados a las estructuras de archivos en el desarrollo del sistema.
- **Asignación de trabajo.** Estas estructuras asignan las responsabilidades a los equipos de trabajo apropiados para implementar e integrar los módulos.

### 1.4.3.2 Relaciones entre estructuras

- Elementos de una estructura se relacionan con elementos de otras estructuras.
- A menudo la estructura dominante es la descomposición de módulos, ya que genera la estructura del proyecto.
- Las estructuras proporcionan una poderosa separación de problemas para la creación de la arquitectura.
- Las estructuras son la base de la documentación de la arquitectura.

### 1.4.3.3 Escogencia de la estructura

En 1995, Philippe Krutchen público un paper, describiendo el concepto de arquitecturas que comprende estructuras separadas y aconsejando concentrarse en cuatro. Las cuales son:

- **Lógica.** Asigna el sistema en clases y componentes, se enfoca en las partes del sistema que proveen una funcionalidad y que los usuarios verán cuando interactúen con el sistema.
- **Procesos.** Explica como las partes del sistema trabajan en conjunto y como se mantienen en sincronización. También explica como el sistema asigna las unidades computacionales, como son los procesos e hilos.
- **Desarrollo.** Explica cómo se gestionará el software durante el desarrollo
- **Física.** Explica como el software que implementa el sistema es asignado en las plataformas computacionales.

### 1.4.4 Métodos y procesos de desarrollo de software

El desarrollo de software puede ser hacerse de muchas maneras. Estas maneras distintas son llamadas métodos o procesos, las cuales pueden ser:

- **Método cascada:** En los métodos cascada, las diferentes fases de las actividades de desarrollo del sistema se siguen de forma secuencial.
- **Proceso unificado:** Es un proceso popular en el que varias actividades, tales como generación de requerimientos, desarrollo y pruebas se superponen. En lugar de estar asociados a determinados productos y las tareas que los crean, las fases en el proceso unificado siguen la vida del producto, desde el inicio hasta la elaboración de su construcción y, finalmente, la transición.
- **Métodos agiles:** Los métodos agiles son consecuencia del manifiesto ágil (Manifesto, 2001) el cual declara (entre otras cosas) que hay más

valor en el software trabajado que en la documentación generada por métodos como la cascada o el proceso unificado.

## 1.5 Diseño de una Arquitectura de Software

A continuación, se explicará brevemente los pasos para realizar un diseño de arquitectura de software, los cuales son:

- Arquitectura en el ciclo de vida
- Diseñando la arquitectura
- Formando la estructura del equipo y su relación con la arquitectura
- Creación de un esqueleto del sistema

### 1.5.1 Arquitectura en el ciclo de vida

Cualquier organización que adopte una arquitectura como fundamento para sus procesos de desarrollo de software, necesita entender su lugar en el ciclo vital.

Existen muchos modelos de ciclo vital, pero uno que pone la arquitectura como tema central es el modelo de Entrega Evolutiva o *Evolutionary Delivery Life Cycle* (ver Figura 15 Ciclo de Vida de entrega evolutiva). El objetivo de este modelo es involucrar al usuario y clientes en el desarrollo, obteniendo retroalimentación de ellos e iterar entre varios lanzamientos hasta generar un lanzamiento final del producto.

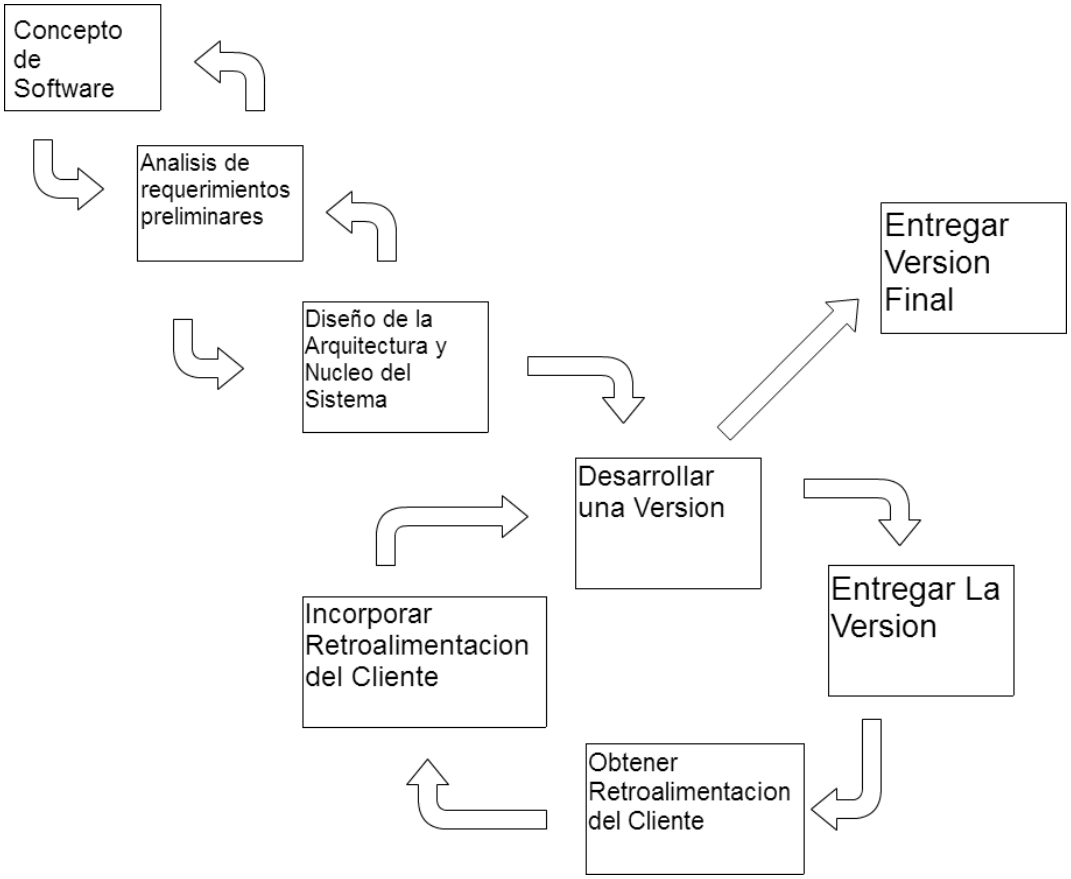



*Figura 15 Ciclo de Vida de entrega evolutiva*

Fuente: (Bass, Clements, & Kazman, 2003)

Para saber cuándo comenzar a diseñar la arquitectura es necesario conocer los requerimientos del sistema de antemano. Es necesario identificar los principales objetivos del negocio, luego volverlos en escenarios de calidad o casos de usos. Al hacer esto se escogen aquellos que tendrán un mayor impacto en la arquitectura.

### 1.5.2 Diseñando la arquitectura

En esta sección se explicará un método de diseño de arquitectura para satisfacer los requerimientos funcionales y de calidad. Este método es conocido como *Atribute-Driven Design* (ADD).

ADD es una aproximación para definir una arquitectura de softwate que basa su proceso de descomposición en los atributos de calidad que el software debe satisfacer. Este proceso de descomposición es recursivo y es un método *top-down*, donde en cada etapa se escogen tácticas y patrones de arquitectura para satisfacer un conjunto de escenarios de calidad.

La salida que genera ADD es la primera de varios niveles de vistas de descomposición de una arquitectura. No todos los detalles de las vistas provienen de aplicar el modelo ADD; el sistema es descrito como un conjunto de contenedores de funcionalidades e interacciones entre ellos.

Esta es la primera articulación de la arquitectura durante el proceso de diseño y, por tanto, es necesariamente grano grueso. Sin embargo, es fundamental para el logro de las cualidades deseadas, y proporciona un marco para lograr la funcionalidad.

La diferencia entre una arquitectura que resulta de aplicar ADD y una lista para su implementación está en las decisiones de diseño más detalladas que se deben hacer.

#### 1.5.2.1 Entrada del ADD

La entrada de ADD es un conjunto de requerimientos. ADD asume requisitos funcionales (típicamente expresados como casos de uso) y restricciones como entrada, así como otros métodos de diseño. Éste método necesita que los requerimientos de calidad sean expresados como un conjunto de escenarios de calidad de especificaciones del sistema, los cuales deben ser definidos con el detalle necesario para la aplicación.

Como escenarios de calidad para una puerta de garaje podríamos incluir los siguientes:

- Los dispositivos y controles para abrir y cerrar la puerta son diferentes por la variedad de productos en la línea de productos.
- El procesador utilizado en diferentes productos diferirá. La arquitectura del producto para cada procesador en específico debe ser directamente derivado de la arquitectura de la línea de productos.
- Si un obstáculo (persona u objeto) es detectado por la puerta del garaje durante su cierre, debe detenerse y reabrirse en 0.1 segundos.
- La puerta del garaje debe ser accesible para recibir diagnósticos y administración de un sistema de información del hogar, usando un protocolo de diagnóstico para productos.

#### 1.5.2.2 Comenzando el ADD

ADD depende de la identificación de los *drivers* de la arquitectura, los cuales como se mencionó anteriormente son los escenarios de calidad que tengan mayor impacto en la arquitectura. Durante el diseño, el determinar que *drivers* de arquitectura son clave puede cambiar, ya sea como resultado de un mejor entendimiento de los requerimientos o por cambio en los mismos. Aun así, el proceso puede comenzar cuando los requerimientos de los *drivers* arquitecturales son conocidos con cierta garantía.

#### 1.5.2.3 Pasos del ADD

A continuación, se mencionan brevemente los pasos para diseñar una arquitectura utilizando el método ADD:

- Escoger el modulo a descomponer: El modulo para empezar usualmente es el sistema completo. Todas las entradas requeridas para este módulo deben estar disponibles (restricciones, requerimientos funcionales, requerimientos de calidad).
- Refinar el módulo de acuerdo a los siguientes pasos:

- Escoger los drivers de la arquitectura del conjunto de escenarios de calidad y requerimientos funcionales. Este paso determina que es lo más importante para la descomposición.
- Escoger un patrón arquitectural que satisfaga los drivers de la arquitectura. Crea (o selecciona) el patrón basado en las tácticas que pueden ser usadas para lograr los *drivers* arquitecturales. Identificar los módulos secundarios necesarios para poner en práctica las tácticas.
- Instanciar los módulos y asignar la funcionalidad de los casos de usos representándolos usando múltiples puntos de vista.
- Definir las interfaces de los módulos secundarios. La descomposición proporciona módulos y restricciones en los tipos de interacción de los módulos. Documenta esta información en la interfaz de documentación para cada módulo.
- Verificar y refinar los casos de uso y los escenarios de calidad y convertirlos en restricciones para los módulos secundarios. En este paso se verifica que nada importante fue olvidado y prepara los módulos secundarios para su futura descomposición o implementación.

- Repetir los pasos anteriores para cada módulo que necesite ser descompuesto.

### 1.5.3 Formando la estructura del equipo y su relación con la arquitectura

Una vez que los primeros niveles de la descomposición de los módulos de la arquitectura son bastante estables, estos módulos pueden ser asignados a los equipos de desarrollo.

Una vez que se ha acordado con una arquitectura para el sistema en construcción, los equipos de trabajo son asignados a cada módulo principal. Cada equipo entonces crea sus propias prácticas de trabajo internas. Además, los procedimientos de calidad y de prueba se establecen para cada equipo y

cada equipo debe establecer sus enlaces de comunicación y coordinación con otros grupos.

Reconocer los módulos como mini dominios inmediatamente sugiere que el modo más efectivo de utilizar al personal es el de asignar miembros a los equipos de acuerdo a su experticia. Por ejemplo, un experto en base de datos vera el problema desde su punto de vista (almacenamiento de la data, persistencia, replicación), en cambio un experto en telecomunicaciones vera el sistema en términos de telecomunicaciones.

### 1.5.4 Creación de un esqueleto del sistema

Una vez que la arquitectura está suficientemente diseñada y los equipos de trabajo están posicionados para comenzar su construcción, el esqueleto del sistema puede ser construido. La idea en esta etapa es proporcionar una capacidad subyacente para implementar la funcionalidad de un sistema en un orden ventajoso para el proyecto.

Las practicas clásicas de la ingeniería del software recomiendan el método *stub* (Wikipedia, Stub Method, 2016) para que así secciones del sistema puedan ser agregados de forma separada y ser probadas independientemente. Sin embargo, ¿qué secciones de código deben aplicársele el método stub? Mediante el uso de la arquitectura como guía, se hace clara una secuencia para su implementación.

En primer lugar, implementar el software que se ocupa de la ejecución y la interacción de los componentes arquitectónicos. Luego se pueden escoger cuales componentes proveedores de funcionalidad deben ser agregados al sistema, ya sea basado en disminuir el riesgo direccionando las áreas más problemáticas primero o basado en los tipos de personal disponible, o puede estar basado en sacar algo útil al mercado lo más pronto posible.

Una vez que se han elegido los elementos que proporcionan el siguiente incremento de la funcionalidad, se puede emplear las estructuras de usos que digan cual software adicional debe estar funcionando correctamente en el sistema para apoyar esa funcionalidad.

Este proceso continúa creciendo y creciendo en los incrementos de funcionalidades del sistema, hasta que todo está en su lugar. En ningún momento la integración y las pruebas se hacen abrumadoras ya que en cada incremento se hace fácil encontrar cualquier falla.

# Capítulo 4: Marco Metodológico

Este capítulo describe la manera en que se realizó el estudio como respuesta al problema planteado previamente.

## 4.1 Bases metodológicas de la investigación

En esta sección se describirá el procedimiento utilizado para lograr el objetivo de la investigación.

La metodología hace referencia al conjunto de procedimientos basados en principios lógicos, utilizados para alcanzar una gama de objetivos que rigen en una investigación científica. (Eyssautier de la Mora, 2006).

A continuación, se hablará acerca del tipo de investigación con el cual se elaboró este trabajo.

### 4.1.1 Tipo de investigación

Tomando en cuenta el problema planteado, este Trabajo Especial de Grado se define del tipo Proyecto Factible, que según UPEL (2003):

> "Consiste en la elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El proyecto debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades" (p. 16).

El proyecto factible conforma un proceso de planificación en el cual la investigación es una etapa que le proporciona información para sustentar la propuesta.

Tomando en cuenta las diversas concepciones, el proyecto factible se desarrolla a través de las siguientes etapas (Moya, 2002):

- El diagnóstico de las necesidades, el cual puede basarse en una investigación de campo o en una investigación documental. Las necesidades del presente trabajo fueron obtenidas al interactuar y dialogar con los *stakeholders* de SAREN, además de realizar investigación bibliográfica de casos de uso similares.
- Planteamiento y fundamentación teórica de la propuesta. El cual se encuentra en el marco teórico del presente trabajo.
- El procedimiento metodológico. En este trabajo se aplicó la metodología *Atribute Driven Design* (ADD).
- Las actividades y recursos necesarios para su ejecución. En el presente trabajo, por cada objetivo específico se realizaron una serie de actividades como son: (a) La investigación bibliográfica y de casos de uso de las diferentes propuestas de arquitectura para grandes volúmenes de datos, en las cuales se encontraron patrones para solucionar problemas comunes; (b) Para el diseño de la arquitectura se procedió a aplicar la metodología *Atribute Driven Design* (ADD); (c) Para la selección de las herramientas asociadas a cada componente de la arquitectura diseñada se utilizaron matrices de evaluación en base a la metodología *Desmet* (Kitchenham, 1996); (d) Para implementar la arquitectura seleccionada, fue necesario el diseño de los mecanismos de Extracción, Transformación y Carga (ETL), además se desplego y probo la arquitectura en ambientes de desarrollo y calidad en la oficina principal de SAREN.
- El análisis de viabilidad o factibilidad del proyecto. Se contó con el apoyo de los encargados de SAREN y de los asesores de Phd 2014

Consultores CA, quienes, a través del juicio de experto, confirmaron la viabilidad y factibilidad del proyecto. (ver ANEXO 12: Juicio de Experto)

- En el caso de que se tenga que desarrollar el proyecto factible, es necesario indicar la ejecución de la propuesta y la evaluación tanto del proceso como de sus resultados. Para la ejecución del presente trabajo se contó con un ambiente de desarrollo y un ambiente de calidad, los cuales fueron desplegados en la oficina de SAREN ubicada en Altamira. En estos ambientes se instalaron y configuraron todos los componentes de la arquitectura de acuerdo a las herramientas seleccionadas a través de las matrices de evaluación en base a la metodología *Desmet.* Se desarrollaron los mecanismos de Extracción, Transformación y Carga (ETL). Se diseñó el modelo de datos *NoSQL* y se realizaron las pruebas respectivas.

### 4.1.2 Población y Muestra

La población de este Trabajo Especial de Grado está representada por todos los Registros Mercantiles de Venezuela, que están conformadas por 48 oficinas a nivel nacional.

Tomando en cuenta que la investigación pretende desarrollar una Arquitectura Big Data, la muestra establecida fueron datos entregados por el cliente para hacer las pruebas, esta data fue solicitada para aplicarla en ambiente de desarrollo y no tiene implicaciones de confidencialidad.

### 4.1.3 Técnicas e Instrumentos de Recolección de Datos

Las técnicas de recolección de datos son las distintas formas o maneras de obtener la información (Arias F. G., 1999).

Las técnicas utilizadas para la recolección de datos fueron:

- **Revisión Bibliográfica:** en ella se acude especialmente a revistas científicas, informes y monografías, medios de comunicación que reflejan con más dinamismo que los libros los adelantos que se producen. (Sabino C. , 1992).

  La técnica consiste en recopilar y revisar todos aquellos documentos que permiten confrontar el aspecto teórico con la situación real o práctica dentro del diseño e implementación de arquitecturas Big Data en el mercado moderno.

  La revisión de los estudios previos nos permitirá (Pedraz A. , 2004):

  - Ahondar en la explicación de las razones por las que hemos elegido dicho tema de investigación
  - Conocer el estado actual del tema: qué se sabe, qué aspectos quedan por investigar
  - Identificar el marco de referencia, las definiciones conceptuales y operativas de las variables estudiadas
  - Descubrir los métodos para la recolección y análisis de los datos utilizados
  - Contar con elementos para la discusión, donde se compararán los resultados que obtengamos con los de los estudios previos
- **Fuentes Infográficas:** consiste en recopilar información a través de fuentes en línea tales como webinars, foros, páginas web.

## 4.2 Metodología de Desarrollo

Se utilizó para el desarrollo de la Arquitectura Big Data para Registros Mercantiles la metodología *Atribute-Driven Design (ADD),* explicada previamente en el Marco Teórico.

A continuación, se presenta la ejecución de los pasos de la metodología ADD en el siguiente capítulo.

# Capítulo 5: Marco Aplicativo

En este capítulo se describe el diseño y desarrollo de la Arquitectura Big Data para Registros Mercantiles, así como también las pruebas realizadas, las cuales fueron analizadas para elaborar las conclusiones correspondientes.

A continuación, se explicarán los entregables del Trabajo Especial de Grado que se obtuvieron al aplicar la Metodología de *Atribute Driven Design* (ADD).

## 5.1 Entrada del ADD

En la entrada de este método para diseño de arquitecturas de software, se tienen 3 tipos principalmente, los cuales son:

- Requerimientos funcionales del Sistema
- Restricciones de diseño
- Requerimientos de calidad

Las entradas deben ser obtenidas de los stakeholders del sistema, por lo que se involucró a las personas que estarían afectadas con la implementación del sistema. En nuestro caso, los directores de tecnología de SAREN, técnicos del sistema y usuarios.

Se obtuvieron las siguientes entradas, mediante interacción con dichos *stakeholders*:

### 5.1.1 Requerimientos funcionales

- El Sistema debe permitir al usuario almacenar las transacciones diarias hechas de los Registros Mercantiles.
- El Sistema debe integrarse con el sistema actual de SAREN, hasta que el sistema actual se haya migrado por completo.

- El Sistema debe ser capaz de facilitar la visualización de los datos para la generación de reportes.
- El Sistema debe permitir realizar búsquedas rápidas al usuario.
- El Sistema debe permitir integrar los datos de todas las oficinas encargadas del Registro Mercantil a nivel nacional.
- El Sistema debe mantener su rendimiento aun cuando se maneje un gran volumen de datos.

### 5.1.2 Restricciones de diseño

- El sistema debe cargar datos de la Base de datos Oracle 10g a un Clúster NoSQL.
- Se debe utilizar Apache Hadoop como estacionamiento para operaciones de MapReduce, debido a que manejará un gran volumen de datos.
- El Sistema debe correr en un ambiente GNU/Linux.
- Los datos deben ser replicados en cada nodo del Clúster NoSQL.
- Las consultas del sistema, deben poder ser accedidas mediante una interfaz web.

### 5.1.3 Requerimientos de calidad

Entre los requerimientos principales de calidad, de acuerdo a la norma ISO 9126 en su primera parte, la Arquitectura de Big Data propuesta para los Registros Mercantiles, debe cumplir con características como funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad, calidad en uso. De estas características, los *stakeholders* seleccionaron las siguientes:

*Tabla 3 Seguridad*

| | |
|---|---|
| Estímulo | Un usuario desea acceder a los datos del sistema |
| Fuente de Estímulo | Algún usuario intentando acceder o alguna otra persona ajena a la |

| | |
|---|---|
| | empresa. |
| Ambiente | El usuario intenta acceder a la plataforma del sistema mediante una autenticación. |
| Artefacto | Clúster NoSQL. |
| Respuesta | Se solicita al usuario que ingrese una contraseña, la cual previamente fue creada siguiendo normas de seguridad |
| Medida de la Respuesta | La contraseña es alfanumérica y con más de 7 dígitos. |

*Tabla 4 Escalabilidad*

| | |
|---|---|
| Estimulo | El sistema procesa y almacena cada vez más datos |
| Fuente de Estimulo | Datos provenientes de las transacciones diarias de las más de 400 oficinas a nivel nacional. |
| Ambiente | Los datos son enviados al clúster NoSQL mediante procesos de ETL.<br>Los cuales se cargan desde Oracle 10g a un Clúster en Cassandra. |
| Artefacto | Clúster NoSQL Cassandra. |
| Respuesta | El sistema debe replicar los datos, en diferentes nodos, de modo que los servidores no lleguen a su máxima capacidad.<br>La escritura debe ser rápida.<br>Se debe poder almacenar gran cantidad de datos sin perder significativamente el rendimiento del |

| | |
|---|---|
| | sistema. |
| Medida de la Respuesta | Las consultas de los datos, no deben tardar más de 2 minutos si revisa más de 50 millones de registros, dependiendo de la cantidad de datos.<br>El sistema debe aceptar la escalabilidad horizontal. |

*Tabla 5 Capacidad de pruebas*

| | |
|---|---|
| Estimulo | El Sistema debe ser probado antes de pasar a producción. |
| Fuente de Estimulo | Stakeholders deben cerciorar que el Sistema cumple sus requerimientos de negocio. |
| Ambiente | Ambiente de desarrollo, el cual fue asignado en una máquina virtual por SAREN. |
| Artefacto | Ambiente GNU/Linux |
| Respuesta | En el sistema se cargarán los datos del modelo relacional.<br>Para la carga, el modelo debe estar desnormalizado. |
| Medida de la Respuesta | Se debe verificar que la cantidad de registros que se almacenan en el Clúster de Cassandra, concuerda con la cantidad de registros en las tablas del modelo relacional. |

*Tabla 6 Tolerancia a fallos*

| | |
|---|---|
| Estimulo | En el sistema falla alguno de sus componentes o Nodos del Clúster. |
| Fuente de Estimulo | Ocurre una falla al hacer una consulta a un dato de algún Nodo. Reinicio del Sistema, Apagón eléctrico, catástrofe ambiental. |
| Ambiente | Ocurre una falla al hacer una consulta a un dato de algún Nodo. Reinicio del Sistema, Apagón eléctrico, catástrofe ambiental. |
| Artefacto | Clúster NoSQL u otro componente de la arquitectura. |
| Respuesta | El sistema debe ser capaz de responder a las solicitudes de consulta mediante algún respaldo de los datos. |
| Medida de la Respuesta | El Sistema replica los datos, de manera que, si algún nodo de almacenamiento falla, otro pueda responder en su lugar. Por lo que el sistema debe responder a la consulta de datos siempre. |

*Tabla 7 Visualización de datos*

| | |
|---|---|
| Estimulo | El usuario desea visualizar datos históricos de las transacciones. |
| Fuente de Estimulo | Generación de reportes para tomar decisiones del negocio |
| Ambiente | Interfaz web |
| Artefacto | Componente de visualización o motor |

| | |
|---|---|
| | de búsqueda especializado en visualizar datos. |
| Respuesta | Se debe permitir al usuario visualizar los datos por medio de dashboards y gráficos. |
| Medida de la Respuesta | Acceder a los datos y generar gráficos de los datos existentes en el almacén de datos. |

*Tabla 8 Análisis de datos*

| | |
|---|---|
| Estimulo | Disponer de datos históricos para realizar consultas y generar reportes |
| Fuente de Estimulo | Los stakeholders desean generar reportes de sus transacciones. |
| Ambiente | Almacén de datos en HDFS. |
| Artefacto | Estacionamiento BI, el cual será Apache Hadoop por su capacidad de hacer operaciones MapReduce. |
| Respuesta | Almacenamiento histórico de la data, mediante la creación de cubos. |
| Medida de la Respuesta | Los datos se almacenarán en HDFS y luego se crean los cubos utilizando motores de consultas para analizar los datos y generar los reportes. |

## 5.2 Primera iteración de la metodología ADD

### 5.2.1 Paso 1: Confirmar que haya suficiente información de los requerimientos

En este primer paso de ADD, se revisa que los requerimientos proporcionados por los *stakeholders* sean suficientes*.* Los requerimientos mostrados en la sección anterior (5.1) son efectivamente suficientes para comenzar el diseño de la Arquitectura. Fueron obtenidos previamente en reuniones con los *stakeholders* y cualquier responsable de la empresa afectado por el diseño de la nueva arquitectura.

### 5.2.2 Paso 2: Escoger un elemento del sistema a descomponer

En este paso, se descompuso el sistema completo.

### 5.2.3 Paso 3: Identificar los drivers de la arquitectura

En este paso se analizaron los escenarios y su importancia en la arquitectura.

*Tabla 9 Drivers de la Arquitectura*

*Fuente: Elaboración Propia*

| # | Drivers de la arquitectura | Importancia | Dificultad |
|---|---|---|---|
| 1 | Velocidad | Alta | Mediana |
| 2 | Volumen | Alta | Mediana |
| 3 | Variedad | Alta | Mediana |
| 4 | Consulta de imágenes | Mediana | Alta |
| 5 | Tolerancia a fallos | Alta | Mediana |
| 6 | Generación de reportes | Alta | Alta |

- El primer driver de la arquitectura es la Velocidad, lo cual quiere decir que las consultas realizadas a un conjunto de datos no deben tardar más de 60 segundos en responder. El sistema debe poder realizar consultas a gran cantidad de datos tomando menos de 60 segundos.

- El segundo Driver a analizar es el Volumen, lo cual significa que el sistema debe ser escalable debido a la enorme cantidad de datos transaccionales provenientes de las distintas oficinas a nivel nacional.
- El tercer driver es la Variedad, debido a que el Sistema contiene muchos datos en sus tablas en NULL, es necesario que el esquema sea flexible y acepte una amplia variedad de datos.
- El cuarto driver es las consultas de imágenes, debido a que un importante porcentaje de los datos que maneja SAREN son de tipo blob, por lo cual es necesario poder almacenar y realizar consultas a esas imágenes, que por lo general son documentos o firmas.
- El quinto driver es la tolerancia a Fallos. Usualmente en Big Data se utiliza *commodity hardware* el cual tiene un alto riesgo a fallar, por lo cual deben existir mecanismos para evitar perder datos.
- El sexto driver es la generación de reportes, lo cual significa que en la empresa de SAREN es de gran importancia que sus plataformas permitan la generación de reportes y *dashboards,* mediante la utilización de herramientas de inteligencia de negocio para tomar acciones acertadas.

### 5.2.4 Paso 4: Escoger un patrón que satisfaga los drivers de la arquitectura

Este es el primer paso de diseño del método.

#### 5.2.4.1 Aspectos de diseño asociados a los drivers de la arquitectura

En el sistema se distinguen los siguientes aspectos de diseño por driver:

- **Velocidad:** Se generan muchos datos en un corto periodo de tiempo. Los datos se generan principalmente de fuentes de datos relacionales. Las consultas no deben consumir largos periodos de tiempo.
- **Volumen:** La cantidad de datos crece exponencialmente con el tiempo. El procesamiento y almacenamiento de los datos debe hacerse a gran escala. El desempeño del sistema no debe decaer significativamente.

- **Variabilidad:** Existen datos de diferentes formatos y pueden ser no estructurados.
- **Consulta de imágenes:** Las imágenes forman una parte importante del sistema, por lo que el sistema debe ser capaz de almacenar adecuadamente las imágenes que vienen en formato blob desde una base de datos relacional.
- **Tolerancia a fallos:** Los datos pueden perderse debido a fallas en el hardware, por lo que el sistema debe ser capaz de replicar los datos a otros nodos.
- **Generación de reportes:** Para la generación de reportes es necesario que se permita acceder directamente a los datos para realizar consultas y también el uso de herramientas de generación de *dashboards* o inteligencia de negocio que sean compatibles con la plataforma.

#### 5.2.4.2 Lista de patrones por cada aspecto de diseño

Los patrones mostrados a continuación son específicos de arquitecturas Big Data, los cuales son aplicados dependiendo del caso de uso y la problemática a solventar. (Buhler, Erl, & Khattak, 2015)

Se generan muchos datos en un corto periodo de tiempo:

- Alto volumen de almacenamiento binario
- Alto volumen de almacenamiento tabular

Los datos se generan principalmente de fuentes de datos relacionales.

- Fuente Relacional

La cantidad de datos crece exponencialmente con el tiempo:

- Alto volumen de almacenamiento binario
- Alto volumen de almacenamiento tabular
- Fuente Relacional
- Procesamiento Batch a gran escala

El procesamiento y almacenamiento de los datos debe hacerse a gran escala:

- Procesamiento Batch a gran escala
- Desnormalización del conjunto de datos

El desempeño del sistema no debe decaer significativamente:

- Fragmentacion de datos automático
- Replicación y reconstrucción de los datos automático
- Gobierno de conjunto de datos centralizada

Existen datos de diferentes formatos y pueden ser no estructurados:

- Alto volumen de almacenamiento binario
- Alto volumen de almacenamiento tabular

Los datos pueden perderse debido a fallas en el hardware:

- Replicación y reconstrucción de los datos automático
- Fragmentación de datos automático

Acceder directamente a los datos y herramientas de inteligencia de negocios compatibles con la plataforma:

- Acceso directo a los datos
- Acceso indirecto a los datos
- Procesamiento Batch a gran escala

#### 5.2.4.3 Selección de los patrones

Para hacer la tabla más sencilla se le asignó a cada driver un número:

1. Velocidad
2. Volúmen
3. Consulta a imágenes
4. Tolerancia a fallos
5. Variabilidad
6. Generación de reportes

Por cada driver de la arquitectura se tienen los siguientes patrones:

*Tabla 10 Patrones vs Drivers*

*Fuente: Elaboracion Propia*

| | Drivers | | | | | |
|---|---|---|---|---|---|---|
| Patrones | 1 | 2 | 3 | 4 | 5 | 6 |
| Alto volumen de almacenamiento binario | Si | Si | Si | No | Si | No |
| Alto volumen de almacenamiento tabular | Si | Si | Si | No | No | No |
| Fuente Relacional | Si | Si | No | No | No | No |
| Procesamiento Batch a gran escala | Si | Si | No | No | No | Si |
| Desnormalización del conjunto de datos | Si | Si | No | No | No | No |
| Fragmentación de datos automático | Si | Si | No | Si | No | No |
| Replicación y reconstrucción de los datos automático | No | Si | No | Si | No | No |
| Gobierno de conjunto de datos centralizada | No | Si | No | No | No | No |

| | | | | | | |
|---|---|---|---|---|---|---|
| Acceso directo a los datos | No | No | No | No | No | Si |
| Acceso indirecto a los datos | No | No | No | No | No | Si |

A continuación, se explicarán brevemente los patrones seleccionados los cuales no son mutuamente excluyente, por lo tanto, es posible aplicar varios patrones a la vez sin ningún inconveniente. En muchos casos los patrones hacen uso de los mismos elementos o mecanismos:

- **Alto volumen de almacenamiento binario:** Este patrón fue escogido debido a los datos de tipo blob o binarios almacenados en Oracle.
  - **Problema:** El almacenamiento de grandes cantidades de datos no estructurados en tecnologías de bases de datos tradicionales, no solo incurre en penalización del rendimiento, también sufre de problemas de escalabilidad cuando la cantidad de datos aumenta.
  - **Solución:** Los datos no estructurados se almacenan en base a una simple técnica de almacenamiento basado en clúster que implementa el accedo a las bases de datos a través de llaves o claves primarias.
  - **Aplicación:** Se utiliza una base de datos NoSQL que se encarga de almacenar los datos binarios con una clave de identificación única, de manera que cada unidad de datos pueda ser sustituida, eliminada, encontrada o recuperada de forma individual.
  - **Mecanismos:** Mecanismo de almacenamiento y motor de serialización.

- **Fuente Relacional:** Patrón utilizado debido a que los datos de SAREN se encuentran en bases de datos relacionales de licenciamiento privado, como es Oracle. Este patrón viene a satisfacer un nuevo driver de la arquitectura no contemplado anteriormente, que es el de fuentes de datos relacionales.

- **Problema:** Exportar grandes cantidades de datos de una fuente relacional para luego importarlas, no solo consume mucho tiempo, también es ineficiente.
- **Solución:** Para importar datos relacionales se hace una conexión directa desde la plataforma Big Data hasta la base de datos relacional.
- **Aplicación:** Se utiliza un motor de transferencia de datos (herramienta ETL) en el cual se emplean diferentes conectores para conectarse a diferentes bases de datos y luego ejecutar consultas SQL para obtener los datos que serán importados.
- **Mecanismos:** Motor de transferencia de datos, motor de procesamiento, mecanismo de almacenamiento, motor de flujo de trabajo, portal de productividad, manejador de recursos, motor de coordinación.

❖ **Alto volumen de almacenamiento tabular:** Los datos de SAREN pueden llegar a contener muchas columnas debido a la desnormalización de sus tablas, por lo que este patrón es útil para solucionar este problema.

- **Problema:** Las tecnologías de bases de datos tradicionales, no son compatibles con el almacenamiento de grupos relacionados de columnas como una sola columna y sufren de problemas de rendimiento cuando se almacenan filas con un número muy grande de columnas.
- **Solución:** Los datos se almacenan utilizando tecnología de almacenamiento en clúster, que admite un almacenamiento de tablas con capacidad para agrupar columnas relacionadas dentro de una columna padre.
- **Aplicación:** Mediante una tecnología de almacenamiento NoSQL con capacidad para grandes volúmenes de datos, que provee una abstracción de filas/columnas y permite almacenar múltiples pares clave-valor dentro de una columna, además de proveer de un lenguaje parecido al SQL o una interfaz basada en API para realizar operaciones CRUD.
- **Mecanismos:** Mecanismo de almacenamiento y motor de serialización.

- **Procesamiento Batch a gran escala:** Este patrón ayuda para que se puedan hacer operaciones en los datos a gran escala sin problemas de rendimiento, de manera que el sistema pueda tener un estacionamiento para luego hacer los reportes.
  - **Problema:** Procesar grandes cantidades de datos puede llegar a tener bajo rendimiento, además de que las técnicas tradicionales de procesamiento de datos son ineficientes para grandes volúmenes de datos debido a la latencia de transferencia de datos.
  - **Solución:** Los datos son consolidados en forma de un gran conjunto de datos y luego se procesan utilizando una técnica de procesamiento distribuido.
  - **Aplicación:** Los datos se procesan utilizando un sistema de procesamiento por lotes distribuidos, de tal manera que todo el conjunto de datos se procesa como parte del mismo ciclo de procesamiento de una manera distribuida.
  - **Mecanismos:** motor de procesamiento, motor de transferencia de datos, mecanismo de almacenamiento, manejador de recursos, motor de coordinación.

- **Desnormalización de los datos:**
  - **Problema:** El procesamiento distribuido de un conjunto de datos, donde el conjunto de atributos que constituyen un registro no están disponibles como partes del mismo registro físico, puede requerir un extenso enlazamiento de datos o implicar una lógica de procesamiento muy compleja.
  - **Solución:** El conjunto de datos se pre procesa para crear un conjunto de datos, donde cada registro consiste en su conjunto completo de atributos como un agregado.
  - **Aplicación:** El conjunto de datos es desnormalizado mediante un motor de procesamiento, aplicando lógica de consolidación para que cada atributo que se encontraba esparcido en varios registros forme parte de uno solo.

- **Mecanismos:** Motor de procesamiento, mecanismo de almacenamiento, manejador de recursos, motor de coordinación.

- **Fragmentación de datos automático:** Este patrón se utiliza para mejorar el rendimiento de acceso de los clientes al conjunto de datos, sin embargo, en caso de tener muchos datos fragmentados, el rendimiento puede decaer. Este patrón es usado en conjunto con el patrón de replicación y reconstrucción de datos automáticos para evitar perdida de datos.
  - **Problema:** Mientras la cantidad de datos y el número de clientes accediendo a los datos aumentan, la latencia de acceso de los datos va aumentando gradualmente, lo que afecta el tiempo en completar las consultas.
  - **Solución:** El conjunto de datos es dividido horizontalmente por lo que los subconjuntos de filas son almacenados en diferentes maquinas a través del clúster, de este modo se distribuye la carga garantizando un alto rendimiento.
  - **Aplicación:** Se utiliza una base de datos NoSQL que implementa fragmentación automática que dirige a los clientes a diferentes fragmentos en función de su respectivo criterio de consulta.
  - **Mecanismos:** Mecanismo de almacenamiento que soporte la fragmentación automática.

- **Replicación y reconstrucción de datos automático:** Patrón utilizado para cumplir con el driver de tolerancia a fallos.
  - **Problema:** Por lo general en Big Data se utiliza hardware básico, por lo que comparado con hardware de nivel empresarial tiene una mayor probabilidad de fallo y por consecuencia de perdida de datos.
  - **Solución:** Varias copias de los datos son guardados y cualquier perdida en los datos debido a fallas de hardware son reconstruidos automáticamente.
  - **Aplicación:** Se utiliza una tecnología de almacenamiento Big Data que implemente replicación de datos automático, de manera que un mismo

conjunto de datos se encuentra en varias máquinas del clúster, además de proporcionar reconstrucción de los datos.

- **Mecanismos:** Mecanismo de almacenamiento que soporte la replicación de los datos.

- **Gobierno de conjunto de datos centralizado:** Este patrón es necesario para proporcionarle al usuario final una interfaz centralizada para manejo de inteligencia de negocio.
  - **Problema:** El análisis de los datos utilizando tecnologías de Big Data, garantizando una continua gobernabilidad de los datos, desde su adquisición hasta su almacenamiento, puede ser una tarea desalentadora debido a la gran variedad y escenarios de uso no previstos de los datos.
  - **Solución:** La gobernabilidad de los datos está centralizado y se introduce un sistema que automatice las tareas de control de datos, incluyendo la gestión de ciclo de vida de los datos, la auditoria de acceso de datos e identificación del linaje de datos.
  - **Aplicación:** Se introduce un componente dentro de la plataforma Big Data que proporciona una interfaz centralizada para las políticas de creación y el seguimiento de auditoria.
  - **Mecanismos:** Manejador de gobierno de datos, motor de flujo de trabajo, motor de seguridad.

- **Acceso directo a los datos:** Este patrón es utilizado para realizar conexiones directas a los datos y hacer consultas complejas.
  - **Problema:** Analizar un gran volumen de datos utilizando herramientas avanzadas de análisis que se basan en primero exportar los datos y luego importarlos en otro mecanismo de almacenamiento compatible con la herramienta, no solo es ineficiente, sino también consume mucho tiempo.

- **Solución:** Una conexión directa es hecha entre la plataforma Big Data y la herramienta de análisis mediante algún conector o estandarización que permita el acceso de los datos.
- **Aplicación:** Un conector de dos vías es introducido entre la herramienta de análisis y la plataforma Big Data, el cual se encargará de traducir las llamadas de la herramienta a la plataforma.
- **Mecanismos:** Mecanismo de almacenamiento, motor de consulta, motor de procesamiento, manejador de recursos, motor de coordinación.

❖ **Acceso indirecto a los datos:** Este patrón es utilizado para realizar reportes en herramientas BI en las cuales los datos deben ser exportados y transformados para su correcto procesamiento.

- **Problema:** Los analistas de datos que utilizan herramientas tradicionales de Inteligencia de negocio quizás deban acceder a datos procesados en la plataforma Big Data. Sin embargo, el uso de tecnologías de almacenamiento no relacional hace de esta tarea algo difícil, debido a que las herramientas tradicionales de inteligencia de negocio soportan solo conexiones a almacenes de datos relacionales.
- **Solución:** Los datos ya procesados son exportados al almacén de datos relacional, desde donde puede ser accedido por las herramientas existentes de inteligencia de negocio sin la necesidad de hacer conexiones separadas.
- **Aplicación:** Los datos procesados son convertidos al esquema requerido, para luego ser exportados al almacén de datos relacional usando una conexión.
- **Mecanismos:** Motor de transferencia de datos, mecanismo de almacenamiento, portal de productividad, motor de flujo de trabajo, motor de consultas.

#### 5.2.4.4 Combinación de los patrones

Algunos patrones es posible combinarlos, debido a que poseen los mismos mecanismos o existen componentes que satisfacen varios de estos patrones a la vez.

Se encontró que los siguientes patrones pueden combinarse en un solo patrón, debido a la similitud de sus componentes o mecanismos y a que mediante las matrices de evaluación en base a la metodología *Desmet* se encontraron herramientas que cumplen con varios de estos patrones de diseño.

Para el primer conjunto de patrones combinados se encuentran: Replicación y reconstrucción de datos automáticos, fragmentación de datos automáticos, alto volumen de almacenamiento binario y alto volumen de almacenamiento tabular. Estos patrones pueden combinarse en un solo patrón que satisfaga todos los requerimientos, el cual llamaremos **"patrón combinado de almacenamiento".**

Para el segundo grupo de patrones combinados encontramos: Desnormalización de los datos y Fuente relacional. Es posible combinar estos dos patrones en uno solo, debido a que ambos necesitan de mecanismos similares y es posible realizar una desnormalización de los datos, los cuales vienen de fuentes relacionales. A este nuevo patrón le llamaremos **"patrón combinado de transferencia relacional".**

### 5.2.5 Paso 5: Instanciar los elementos de la arquitectura y asignar responsabilidades.

En la realización del paso anterior se observó cómo los patrones seleccionados fueron capaces de satisfacer uno o más drivers de la arquitectura, por lo cual en este paso se explicará qué solución y elementos se encargaron de satisfacer las necesidades de la arquitectura, de manera que se compararan los patrones que satisfacen los respectivos drivers y la solución a la que se llega.

La instanciación de los elementos se realizó de la siguiente manera, al comparar el patrón más el driver que satisface, dando como resultado una solución o componente de la arquitectura.

A continuación, se presenta la instanciación de los elementos de la arquitectura y sus respectivas responsabilidades:

- **Patrón combinado de almacenamiento**
  - Drivers relacionados: Volumen, Velocidad y Variabilidad.
  - Componente de la arquitectura: Base de dato *NoSQL.*
  - Responsabilidad: Componente encargado de almacenar todos los datos transaccionales de la empresa.
- **Patrón combinado de transferencia relacional**
  - Drivers relacionados: Velocidad y Volumen.
  - Componente de la arquitectura: Mecanismos de Extracción, Transformación y Carga.
  - Responsabilidad: Componente encargado de realizar la transferencia de datos a otros componentes de la arquitectura que lo requieran.
- **Procesamiento Batch a gran escala**
  - Drivers relacionados: Volumen, Velocidad y Generación de reportes (paso de pre-procesar los datos).
  - Componente de la arquitectura: Estacionamiento de grandes volúmenes de datos con sistema de archivos distribuidos con capacidad de realizar procesamiento en paralelo.
  - Responsabilidad: Componente encargado de almacenar grandes volúmenes de datos para realizar operaciones sobre ellos.
- **Acceso directo a los datos:**
  - Drivers relacionados: Volumen y Generación de reportes.
  - Componente de la arquitectura: Herramienta de software con lenguaje parecido a *SQL* y capacidad de conectarse directamente al componente de Estacionamiento.

- Responsabilidad: Componente encargado de realizar las operaciones sobre el conjunto de datos del estacionamiento para la creación de cubos y consultas complejas.

❖ **Acceso indirecto a los datos:**

- Drivers relacionados: Volumen y Generación de reportes.
- Componente de la arquitectura: Herramienta de inteligencia de negocios tradicional o con capacidad para grandes volúmenes de datos.
- Responsabilidad: Componente encargado de realizar los reportes y *dashboards* de los datos que previamente fueron procesados por las diferentes consultas del negocio.

❖ **Gobernabilidad centralizada de los datos:**

- Drivers relacionados: Generación de reportes y centralización del sistema.
- Componente de la arquitectura: Portal de Inteligencia de negocio o Interfaz web especializada en el manejo de las diferentes herramientas *Big Data*.
- Responsabilidad: Componente que se encarga de integrar y manejar la mayoría de las herramientas de la arquitectura mediante una interfaz usable y segura.

### 5.2.6 Paso 6: Definir las interfaces de los elementos instanciados.

En el paso anterior se instanciaron los elementos y se obtuvieron una serie de componentes que conforman la arquitectura. Por lo que en este paso se procedió a la selección de herramientas de software comerciales o libres que conforman los componentes, utilizando la metodología Desmet y matrices de evaluación de herramientas de software. También se indica la información que se transmite cada uno de los componentes entre ellos.

#### 5.2.6.1 Evaluación de herramientas de software

Para la comparación de herramientas se utilizó la metodología Desmet ya que es una metodología creada para evaluar herramientas de software. (Kitchenham, 1996)

Las herramientas que son objeto de estudio pertenecen al ámbito de los sistemas de archivos distribuidos, bases de datos *NoSql*, inteligencia de negocios y motores de búsqueda para *Big Data*.

Seguidamente se detallan todos los componentes que integran la metodología Desmet, como una herramienta fiable para la evaluación de aplicaciones de software:

Los criterios de evaluación considerados son:

- **Requerimientos funcionales**: Abarcan aspectos de rendimiento, características avanzadas, funcionalidades específicas, etc.

- **Requerimientos específicos**: Abarcan aspectos generales de cada herramienta y condiciones necesarias que deben cumplir las herramientas para encajar en la cultura de la organización.

Para la aplicación efectiva del método Desmet, se usó una matriz de evaluación que consta de los siguientes atributos:

- **Tipo:** Tipo de requerimiento.

- **Descripción del criterio evaluado:** El criterio a evaluar.

- **Condición:** Condición que debe cumplir el criterio (O-Obligatorio, D-Deseable, S-Suplementario e I-Informativo).

- **Peso:** El valor de importancia del criterio (0= sin valor o criterio no importante, 5= máximo valor o importancia).

- **Cumplimiento:** Si la herramienta cumple o no con el criterio.

- **Observaciones:** Información relevante sobre la herramienta en relación al criterio.

- **Estrategia:** El valor que posee el criterio en la herramienta. Si el criterio tiene como condición que No se cumple, este valor será de cero (0). Si el criterio se cumple, este valor puede ir de 1 a 6 según en qué porcentaje aproximado se cumple el criterio.

- **Calificación ponderada:** Es la calificación obtenida por la herramienta para el criterio dado. Se obtiene de la siguiente fórmula: Peso multiplicado por la Estrategia dividido entre 6.

A continuación, se presentarán los resultados de la evaluación de las herramientas. (Para ver las matrices de evaluación ir a ANEXO 1: Pentaho vs Palo; ANEXO 2: Hadoop vs Otros; ANEXO 3: Cassandra vs Hbase; ANEXO 4: Solr vs Elasticsearch y ANEXO 5: HUE vs Banano).

#### 5.2.6.1.1 Componente para mecanismos de Extracción, Transformación y Carga de datos

En base a los resultados obtenidos de las matrices de evaluación, la herramienta seleccionada es *Pentaho Data Integration.*

Con respecto a la plataforma que soportará las actividades correspondientes a la inteligencia de negocios, *Pentaho* comprende un conjunto de componentes robustos respaldados por una comunidad activa que garantiza un desempeño aceptable de los procesos necesarios para llevar acabo la inteligencia de negocios. Entre las características más relevantes que permiten a *Pentaho* destacarse del resto está la capacidad de realizar actividades de forma offline, soportar lenguaje *SQL* y permitir análisis en tiempo real. Por último, mencionar

que su versión más reciente, incorpora el concepto de integración con *Big Data* y posee conectores para tal fin.

#### 5.2.6.1.2 Componente de Estacionamiento con sistema de archivos distribuido

En base a los resultados obtenidos de las matrices de evaluación, la herramienta seleccionada es *Apache Hadoop*.

Hadoop es un framework de código abierto auspiciado por *Apache Foundation*, el cual desde su lanzamiento se ha transformado en un estándar en la industria, las razones que soportan este rápido auge estriban en las bondades de Hadoop para procesar grandes conjuntos de datos mediante equipos con bajas capacidades de computo agrupados en clúster o sistemas distribuidos.

Adicionalmente Hadoop se integra con un ecosistema de aplicaciones que le facilita cubrir de forma transversal todas las actividades inmersas en la consulta, análisis, obtención de datos, publicación de mecanismos de seguridad, estudio y predicción de comportamiento.

#### 5.2.6.1.3 Base de datos NoSQL transaccional

En base a los resultados obtenidos de las matrices de evaluación, la herramienta seleccionada es Apache Cassandra.

Apache Cassandra se posiciona en el mercado como la herramienta líder en el campo de bases de datos NoSQL de naturaleza transaccional, aunque el proyecto HBASE resulta una alternativa madura y bien posicionada en el mercado open source, la mayoría de los usuarios sigue inclinándose por el proyecto Apache Cassandra producto de alta disponibilidad, tolerancia a fallos, su extensa comunidad y sus APIS para desarrollo.

#### 5.2.6.1.4 Motor de búsqueda Big Data como herramienta de inteligencia de negocio

Para el componente de herramienta de inteligencia de negocio se pudo haber seleccionado cualquier herramienta tradicional de este tipo. Sin embargo, los motores de búsqueda actuales disponibles comercialmente, permiten la generación de gran variedad de reportes para inteligencia de negocio, además de que soportan la generación de reportes por facetas, es decir, filtrando las búsquedas sobre datos que fueron cargados previamente a la herramienta, por lo que todos los reportes son hechos directamente sobre la herramienta. También los motores de búsqueda están diseñados para ser escalables, lo que le da una mayor ventaja competitiva en comparación con las herramientas tradicionales de inteligencia de negocio.

En base a los resultados obtenidos de las matrices de evaluación, la herramienta seleccionada es *Apache Solr.* Esta herramienta fue comparada con Elasticsearch.

*Apache Solr* y *Elasticsearch* son los motores de búsqueda más populares en la actualidad. Ambas herramientas son muy similares y brindan funcionalidades parecidas. En el 95% de los casos será indiferente escoger uno o el otro, ambas herramientas poseen licencia de Apache y una comunidad amplia. Sin embargo, *Apache Solr* tiene una contribución de compañías importantes en el área de *Big Data*, tales como *Hortonworks*, *MapR* y *Cloudera*.

#### 5.2.6.1.5 Componente de Acceso directo a los datos

Como herramienta seleccionada para este componente se decidió por *Apache Hive*, debido a que esta herramienta es un requerimiento particular del cliente. Apache Hive permite realizar consultas parecidas a *SQL* y así crear cubos y tablas para obtener los indicadores del negocio. Además automatiza la creación del algoritmo *MapReduce* por lo que el usuario no tiene que programar el algoritmo directamente.

.

#### 5.2.6.1.6 Componente de manejo centralizado de datos y herramienta de visualización

En base a los resultados obtenidos de las matrices de evaluación, la herramienta seleccionada es *Hadoop User Experience* (HUE).

*Apache Hue* es una herramienta que permite la visualización de los datos que se encuentren en *Hadoop*, la cual se puede integrar con cualquiera de sus versiones. Posee diversas funcionalidades para generar reportes, analizar, procesar, consultar, entre otras. Además de que permite integrar varias herramientas *Big Data* como son: *Hive, Pig, Solr, HDFS, Impala*, entre otras.

### 5.2.6.2 Información transmitida entre las herramientas:

- **Pentaho Data Integration:** Este componente consume los datos que provienen de Oracle, los procesa y transforma para que puedan ser insertados en *Cassandra*, para luego insertarlos en el nodo semilla de *Cassandra*.
- **Apache Cassandra:** Este componente recibe la información ya transformada de la herramienta ETL de *Pentaho*, para luego transmitir los datos necesarios al componente de estacionamiento o Apache Hadoop mediante el uso de otra transformación ETL.
- **Apache Hadoop:** Recibe los datos de la transformación que lee de *Cassandra*. En este componente se almacenan la mayoría de los archivos que serán utilizados por el componente de consultas especializadas para crear tablas o cubos en el *HDFS*.
- **Apache Hive:** En esta etapa la información es consultada directamente al HDFS el cual realiza un trabajo de mapeo y reducción sobre los datos, traduciendo la consulta hecha en *HQL* a un trabajo *Map Reduce*.
- **Apache Solr:** Se hace una conexión a la base de datos por defecto utilizada por *Hive,* para luego enviar mediante una transformación en *Pentaho,* los datos al motor de búsqueda *Solr*, para posteriormente realizar reportes, gráficos, *dashboards*, entre otros.

- **HUE:** Este componente se conecta directamente con *Solr*, *HDFS* y *Hive*, por lo que sirve de interfaz para realizar las consultas en *Hive* y luego generar los *dashboards* o reportes en *Solr*.

### 5.2.7 Paso 7: Verificar y refinar los requerimientos para hacer restricciones en los elementos instanciados

En este paso se procede a verificar que los elementos instanciados cumplan con los diferentes requerimientos funcionales, requerimientos de calidad y las restricciones de diseño.

#### 5.2.7.1 Componentes versus los requerimientos de calidad, requerimientos funcionales y restricciones de diseño

- **Componente para mecanismos de extracción, transformación y carga**
  - Requerimientos Funcionales: Cumple con los requerimientos de permitir al usuario almacenar las transacciones diarias, debido a que es un mecanismo para transferir datos a su respectiva base de datos. También ayuda en la integración de las demás oficinas a nivel nacional, mediante el uso de transformaciones de mediación, que permitan transferir cada cierto tiempo los datos actualizados de las distintas oficinas del país.
  - Requerimientos de calidad: Capacidad de pruebas, Escalabilidad, Tolerancia a fallos. La herramienta fue probada en ambiente GNU/Linux, además de que se transfirieron millones de registros en una gran cantidad de tablas migradas y se utilizó un trabajo en la herramienta que indicaba las transformaciones fallidas.
  - Restricciones de diseño: Este componente es muy versátil, ya que permite realizar migraciones a bases de datos relacionales como a base de datos NoSQL. *Pentaho Data Integration* no tiene ningún problema de ser ejecutada en un ambiente GNU/Linux.
- **Clúster de base de datos NoSQL**
  - Requerimientos Funcionales: Permite realizar inserciones directas en la base de datos, por lo que en algún futuro será posible prescindir del

modelo relacional por completo, permitiendo a los usuarios almacenar sus transacciones diarias en esta base de datos NoSQL. Este componente también es utilizado para almacenar todos los datos de todas las oficinas a nivel nacional. Permite mantener el rendimiento aun cuando se maneje grandes volúmenes de datos, debido a que la base de datos NoSQL puede crecer horizontalmente, permitiendo así repartir la carga de trabajo entre sus nodos y mantener un rendimiento aceptable.

- Requerimientos de calidad: La herramienta fue probada en un ambiente de desarrollo y calidad GNU/Linux. Se insertaron cientos de millones de registros y mantuvo un comportamiento óptimo, debido a que este tipo de base de datos está diseñada para ser escalable. Con respecto a la tolerancia a fallos, la herramienta posee un factor de replicación lo cual ayuda a que los datos se encuentren respaldados en varios nodos del clúster. La base de datos está configurada con la seguridad activada, lo cual implica que para acceder a los datos le solicitara un nombre de usuario y una contraseña.
- Restricciones de diseño: Los datos se cargaron a un Clúster de la base de datos NoSQL Cassandra.

❖ **Componente de estacionamiento Hadoop**

- Requerimientos funcionales: Permite integrar todos los datos para su posterior análisis.
- Requerimientos de calidad: La herramienta posee tolerancia a fallos al igual que *Cassandra*, debido a que los datos pueden ser replicados a través de varios nodos para mantener un respaldo.
- Restricciones de diseño: Es utilizada como estacionamiento debido a que posee la capacidad de realizar operaciones *Map Reduce* sobre un gran volumen de datos.

❖ **Componente de Acceso directo a los datos**

- Requerimientos funcionales: Este componente sirve de ayuda previa para la creación de reportes, debido a que permite realizar consultas

complejas y con los resultados crear los reportes en el siguiente componente de la arquitectura.

- Requerimientos de calidad: La herramienta fue instalada y probada en ambiente de desarrollo. La seguridad que posee, es que se accede a ella por medio de otra herramienta de interfaz web como HUE la cual solicita usuario y contraseña.
- Restricciones de diseño: La herramienta se ejecuta sin ningún inconveniente en el ambiente GNU/Linux, como lo exige la restricción de diseño.

❖ **Componente de integración de herramientas y visualización de datos**

- Requerimientos funcionales: Mediante esta herramienta se generan los reportes de la empresa, debido a que integra las principales herramientas de análisis de datos (Solr y Hive).
- Requerimientos de calidad: La herramienta permite la integración de las demás herramientas. Si falla, solo es necesario volver a levantar el servidor sin ningún problema. Cuando el usuario desea acceder a las funcionalidades de la herramienta, le solicita una autenticación. Permite la creación de grupos de usuario con sus respectivos permisos (roles).
- Restricciones de diseño: La herramienta puede ejecutarse bajo ambiente GNU/Linux como un servidor. Las consultas y generación de reportes son accedidas mediante esta interfaz web.

❖ **Componente de Motor de búsqueda**

- Requerimientos funcionales: Permite realizar consultas y generar reportes facetados, es decir, por filtros. Mantiene un rendimiento estable al haber muchos datos debido a que está diseñada para ser escalable. Contiene diversas funciones que permiten y facilitan la generación de reportes, gráficos y *dashboards* para la inteligencia del negocio.
- Requerimientos de calidad: La herramienta fue probada en ambiente de desarrollo y es escalable horizontalmente.
- Restricciones de diseño: Se ejecuta bajo ambiente GNU/Linux, permite integración con interfaz web para visualización y generación de reportes.

Con esto concluye la ejecución de la metodología ADD. A continuación, se procede a mostrar el diagrama de componentes y el diagrama de casos de uso para el nivel cero y primer nivel.

- **Diagrama de componentes:** En el siguiente diagrama de componentes se puede observar la manera en la que se conectan cada uno de ellos. (Ver Figura 16 Diagrama de componentes)

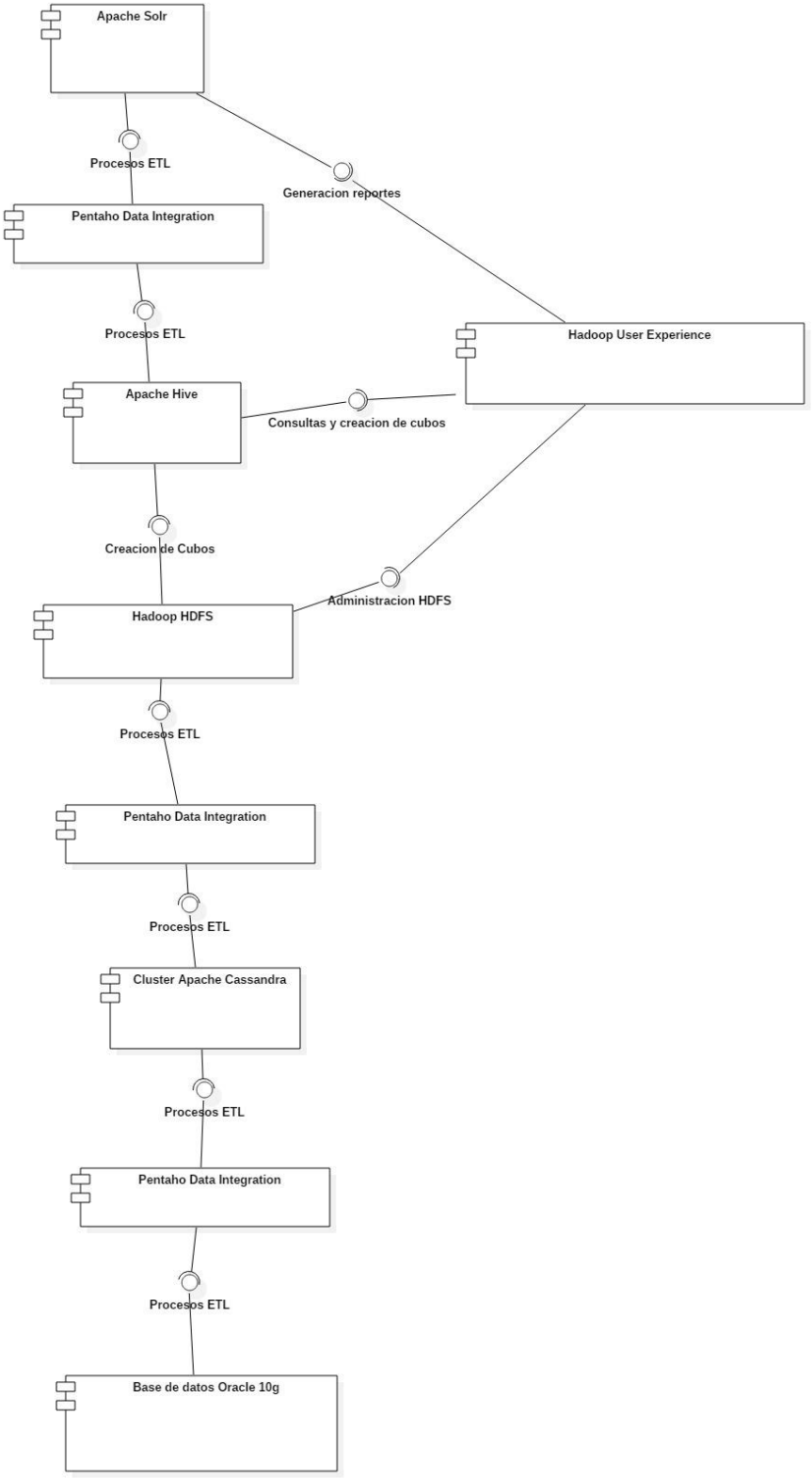



*Figura 16 Diagrama de componentes*

*Fuente Elaboración Propia*

- **Diagrama de casos de uso nivel 0:** En el diagrama de casos de uso se puede observar el nivel cero, en el cual se muestra de forma general los actores y el rol que desempeñan en el sistema. (Ver Figura 17 Caso de uso Nivel 0 y Tabla 11 Actores - Casos de Uso Nivel 0)

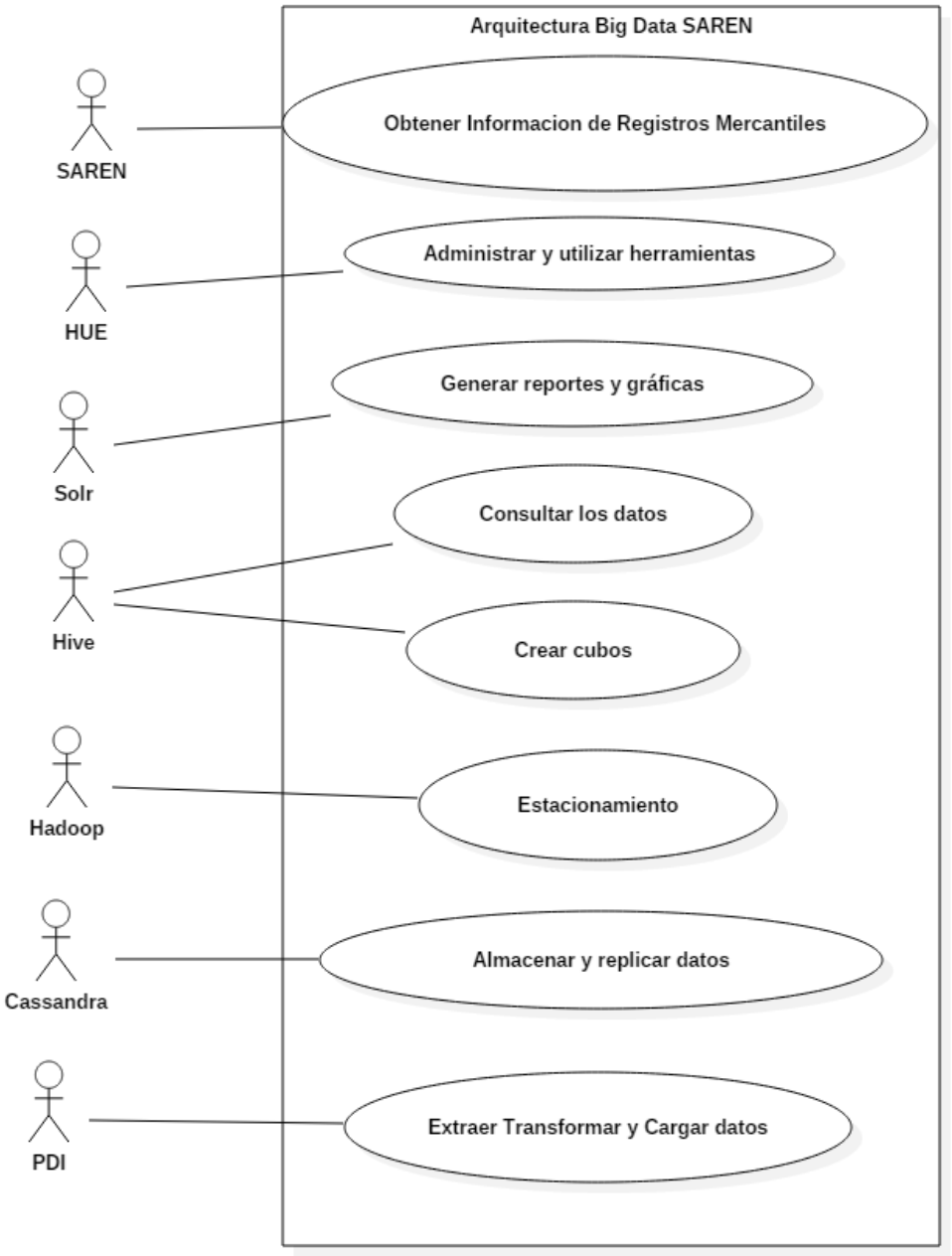



*Figura 17 Caso de uso Nivel 0; Fuente: Elaboración Propia*

*Tabla 11 Actores - Casos de Uso Nivel 0*

*Fuente: Elaboración Propia*

| Actor | Descripción |
|---|---|
| SAREN | Usuario final del sistema, el cual se encarga de utilizar la interfaz web de HUE con todas sus funcionalidades, como generar reportes y hacer consultas para obtener información. |
| HUE | Interfaz web encargada de administrar las herramientas, generar reportes y gráficos (Integra Solr, Hive y HDFS) |
| Solr | Es un motor de búsqueda, utilizado como herramienta de inteligencia de negocio para generar reportes por facetados |
| Hive | Es el encargado de realizar consultas complejas, parecidas a SQL y crear cubos para los indicadores del negocio. |
| Hadoop | Funciona como el estacionamiento de la arquitectura, sobre el cual se realizan la mayoría de las operaciones de análisis, limpieza, procesamiento, entre otros. |
| Cassandra | Almacena todos los datos transaccionales en un clúster de alta capacidad. |

| | |
|---|---|
| Pentaho Data Integration | Encargado de crear y ejecutar transformaciones o Jobs que permiten la transferencia de datos a los componentes requeridos. |

*Tabla 12 Funcionalidades Casos de uso nivel 0*

*Fuente: Elaboración Propia*

| | |
|---|---|
| Caso de uso | Obtener información de Registros Mercatiles |
| Actor | SAREN |
| Flujo Básico | El usuario de SAREN inicia sesión en la interfaz web de HUE. Luego puede realizar una serie de tareas como:<br>a) Consultas utilizando Hive<br>b) Generar reportes/dashboards con Solr<br>c) Administrar HDFS |
| Pre-Condición | Principalmente debe estar funcionando Hadoop, Solr, Hive y el servidor de HUE. El usuario fue creado por el administrador o es administrador. |
| | |
| Caso de uso | Extraer Transformar y cargar datos |
| Actor | Pentaho Data Integration |
| Flujo Básico | Se procede a realizar la transferencia de datos entre los componentes |
| Pre-Condición | Es necesario establecer la conexión con la fuente de datos a extraer transformar y con el componente a cargar los datos |
| | |
| Caso de uso | Almacenar y replicar datos |
| Actor | Cassandra |
| Flujo Básico | Los datos cargados de la fuente relacional son almacenados en el clúster, que, a su vez de acuerdo al factor de replicación, son replicados a través de los nodos del clúster. |
| Pre-Condición | Los datos debieron haber sido cargados previamente en el clúster, bien sea directamente por algún servicio o desde Oracle mediante |

| | |
|---|---|
| | alguna transformación. |
| | |
| Caso de uso | Estacionamiento |
| Actor | Hadoop |
| Flujo Básico | Los datos son almacenados en el HDFS de Hadoop, para su posterior procesamiento utilizando Map Reduce. |
| Pre-Condición | Los datos fueron cargados previamente desde el clúster de Cassandra utilizando una transformación de Pentaho. |
| | |
| Caso de uso | Crear cubos |
| Actor | Hive |
| Flujo Básico | Se crean tablas o cubos en la base de datos de Hive con los datos de Hadoop. |
| Pre-Condición | Las tablas son almacenadas en HDFS mediante una configuración (hive-site.xml) indicando que se utilizara como base de datos una ruta en el HDFS |
| | |
| Caso de uso | Consultar los datos |
| Actor | Hive |
| Flujo Básico | Los datos pueden ser consultados mediante un lenguaje de alto nivel parecido a SQL, el cual consulta directamente a los datos que se encuentran en el HDFS de Hadoop |
| Pre-Condición | Previamente los datos fueron cargados a Hadoop y Hive pre configurado para usar HDFS como lugar de almacenamiento físico. |
| | |
| Caso de uso | Generar reportes y graficas |
| Actor | Solr |
| Flujo Básico | Se generan reportes con las funcionalidades de Solr, como es el facetado.<br>Se crean dashboards o graficas geo referenciales si el usuario lo solicita. |
| Pre-Condición | Los datos fueron previamente cargados a Solr en formato JSON mediante una transformación de Pentaho, los cuales fueron extraídos |

| | |
|---|---|
| | de las tablas creadas por Hive. |
| | |
| Caso de uso | Administrar y utilizar herramientas |
| Actor | HUE |
| Flujo Básico | El usuario inicia sesión<br>Puede generar consultas con Hive<br>Crear tablas en Hive<br>Administrar el HDFS<br>Crear grupos de usuarios<br>Generar reportes y gráficos en Solr |
| Pre-Condición | HUE fue configurado previamente para conectarse con las distintas herramientas a utilizar en la arquitectura. |

Para ver el diagrama de casos de uso nivel 1 ver

ANEXO 9: Casos de uso Nivel 1. A continuación, se procede a explicar las actividades realizadas para desarrollar la Arquitectura Big Data, como son: diseño de ETL, diseño de modelo de datos no relacional y pruebas de carga, volumen y estrés.

## 5.3 Implementación de la Arquitectura

Se mostrará a continuación, la evidencia de la instalación de las herramientas, para una información más detallada de como instalar y configurar cada una de ellas, vea el ANEXO 11: Instalación y configuración de Herramientas.

En la figura 18 se muestra la versión utilizada de Java, la cual se instalo mediante repositorios apt-get install en todos los servidores de la arquitectura.


*Figura 18 Instalación Java*

En la figura 19 se evidencia como se levantaron los nodos del cluster de Cassandra, mediante la herramienta *nodetool*.

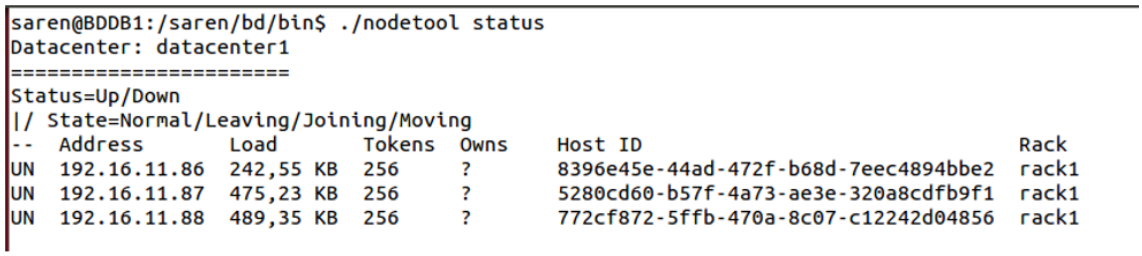



*Figura 19 Cluster Cassandra 3 nodos*

En la figura 20, puede observarse los procesos levantados por el Nodo Maestro, entre los cuales encontramos el NameNode, que es levantado solo por el Maestro, el SecondaryNamenode (puede levantarse en otra maquina para mayor seguridad) y su respectivo Datanode.


*Figura 20 Procesos Hadoop Nodo Maestro*

En la figura 21 se evidencia la consola de cliente de hive, mediante la cual se hacen las consultas HQL.

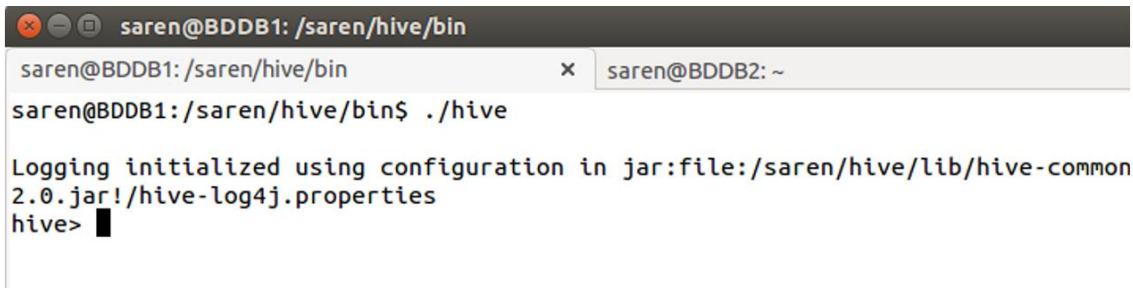



*Figura 21 Evidencia ejecución Hive*

En la figura 22 se evidencia el despliegue del servidor de solr, mediante el comando *./solr start*

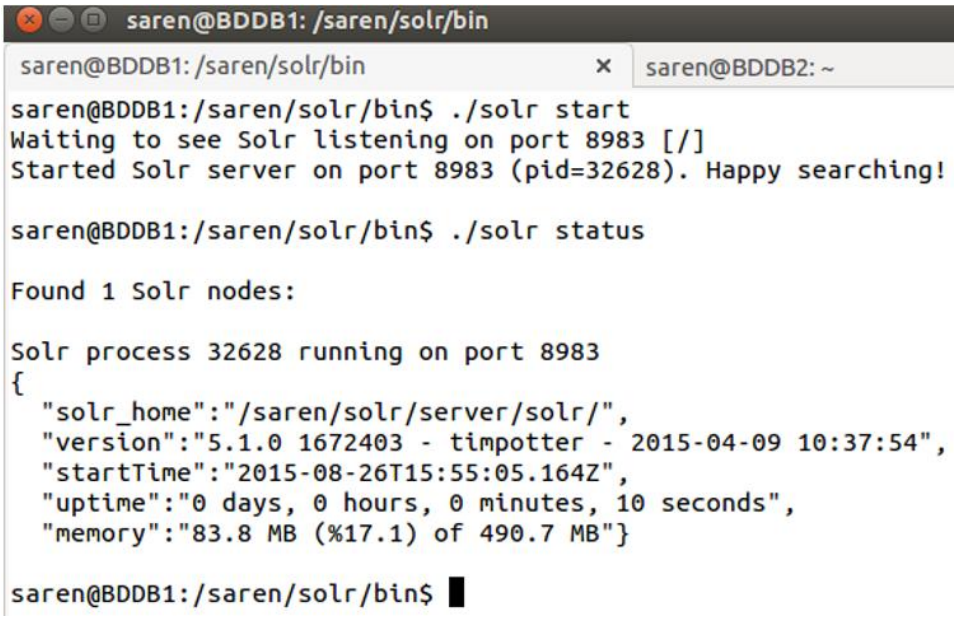



*Figura 22 Ejecución Solr*

Para implementar la Arquitectura fue necesario diseñar el modelo de datos clave valor en Cassandra, para lograr esto se analizó el modelo de datos de SAREN (bajo 3era forma normal en Oracle 10g con 913 tablas) de manera que se pudiese hacer un correcto modelo de datos no relacional, que cumpla con los requerimientos del negocio. (Ver ANEXO 6: Diseño NoSQL SAREN).

Luego fue necesario el diseño de los mecanismos de ETL y las pruebas para lograr una correcta extracción, limpieza, transformación e inserción de los datos en Cassandra y demás componentes de la arquitectura. (Ver ANEXO 7: Diseño de los mecanisoms ETL).

Una vez que se finalizó la construcción de un modelo no relacional y de las transformaciones necesarias para la migración, seguidamente fue necesario monitorear y observar el comportamiento de las herramientas en ejecución, principalmente de las transformaciones, las cuales manejaron gran volumen de datos y la base de datos Cassandra, por lo que se observó el comportamiento del sistema realizando pruebas de carga, volumen y estrés, utilizando

herramientas como hdparm e iostat y para Cassandra la herramienta nodetool. (Ver ANEXO 8: Documento Pruebas de carga volumen y estrés).

# Conclusiones y Recomendaciones

La creación de arquitecturas de software que permitan almacenar y procesar grandes volúmenes de datos para sacarles el mayor provecho, viene a ser una gran ventaja competitiva en la actualidad. En casi todas las áreas de negocio se está aplicando Big Data en conjunto con la ciencia de datos, para tomar decisiones más precisas o generar campañas de mercadeo más exitosas. En los próximos años esta tendencia será casi obligatoria para gran parte de las empresas, ya que con el aumento exponencial de datos en la web y con el auge del internet de las cosas, será aún más necesario procesar y almacenar todos esos grandes volúmenes de datos, utilizando técnicas y metodologías innovadoras y creativas.

Con respecto al primer objetivo específico (Diseñar la arquitectura), se puede concluir que en el diseño de una arquitectura fue importante tomar en cuenta los requerimientos del cliente, en este caso SAREN, ya que esto permitió identificar los casos de uso y los escenarios que influyeran más en la arquitectura. Sin embargo, en la arquitectura se tomó en cuenta que la fuente de datos principal, proviene de sus servidores de *Oracle*, esto mientras puedan migrarse por completo a *Cassandra*. Se pudo observar que los patrones de Big Data, mostrados en el marco aplicativo, para solucionar problemas específicos, los cuales se aplicaron exitosamente junto a la metodología ADD. Cumpliendo con el principio de Ingenieria de Software que impulsa el uso de una Arquitectura inicial con la finalidad de reutilizar componentes y patrones en el desarrollo de software. Todo esto, además, permite generalizar la arquitectura planteada a otros sistemas, como un proceso de migración a Big Data.

Para el segundo objetivo específico (Seleccionar las herramientas de la Arquitectura) se concluye, que la metodología Desmet permitió satisfacer los requerimientos del sistema para la organización, ya que con el sistema de requerimientos y puntajes se logra tomar una mejor decisión en la elección de herramientas, dependiendo del caso de estudio.

En el último objetivo específico se concluye, que para la implementación de la arquitectura fue necesario realizar pruebas de cada una de las herramientas que la conforman, de manera que cada una de ellas pudiese interoperar. En cada uno de los componentes de la arquitectura fue necesario realizar actividades de entonación para lograr satisfacer las necesidades del cliente. Ademas, debido a que el cliente no esta familiarizado con el paradigma Big Data, fue necesario emprender actividades de gestion del cambio.

Recomendaciones a nivel técnico:

- Debido al gran volumen que se maneja en la transferencia de datos utilizando la herramienta Pentaho Data Integration, es recomendable aumentar la memoria del Java Heap en el archivo spoon.sh, con aproximadamente 8GB de RAM.
- Si los datos transferidos son muy pesados, como fue el caso de la transferencia de imágenes, es recomendable que la transformación se configure como una transformación transaccional o por lotes, debido a que transferir datos muy pesados puede consumir rápidamente los recursos de la máquina virtual de Java, lo que ocasiona que el kernel del sistema finalice el proceso antes por consumir muchos recursos.
- Se recomienda que los servidores en los que se despliegue la arquitectura o herramientas, sea de alta capacidad para empresas grandes como SAREN, minimo 12 GB de RAM y 2 procesadores con sistema operativo GNU/Linux.
- Se recomienda probar las herramientas en ambiente de calidad para adaptarse y solucionar cualquier inconveniente que pueda surgir durante la implementación de arquitecturas Big Data.
- Se recomienda que la red en la cual se maneja el tráfico de datos sea una red de alta velocidad y en una red virtual separada (VLAN).

Recomendaciones a nivel General:

- No en todos los casos es viable la utilización de arquitecturas o plataformas Big Data, por lo que es recomendable hacer un estudio de planificación de capacidad en la organización antes de hacer cualquier propuesta de Big Data.
- En algunos casos en las organizaciones es estricto que sus sistemas sean consistentes y tengan alta disponibilidad, que según el teorema de CAP esto solo puede ser satisfecho por el modelo relacional, por lo que una solución Big Data, en estos casos no puede ser viable para las necesidades del negocio.
- El implementar una arquitectura Big Data en una organización no significa que se deba descartar la utilización de sistemas relacionales, en algunos casos será necesario trabajar con ambas tecnologías.
- En Venezuela existen muchas organizaciones que manejan gran cantidad de datos y que utilizan un modelo relacional que comienza a tener problemas de renovación de licenciamiento en moneda extranjera, lo cual implica que existe una gran posibilidad de reutilización de la arquitectura planteada a través del presente trabajo.
- Ademas, es recomendable que el personal encargado del área de tecnologías de información se instruya en la utilización y desarrollo de la arquitectura propuesta en este trabajo para poder satisfacer las necesidades del mercado y que las empresas puedan brindar un mejor servicio a sus clientes.

# Bibliografía


Antiñanco, M. J. (2013). Bases de Datos NoSQL: Escalabilidad y alta disponibilidad a través de patrones de diseño. 10-20.

Apache. (2016). *Hadoop*. Obtenido de Apache Hadoop Org: http://hadoop.apache.org/

Apache. (26 de Enero de 2016). *Hadoop Yarn*. Obtenido de Apache Org: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Apache. (2016). *Lucene*. Obtenido de http://lucene.apache.org/core/

Arias, F. G. (1999). *El Proyecto de Investigacion : Guia para su elaboracion.* Caracas: Episteme.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice.* Addison-Wesley.

Blog, T. B. (7 de Noviembre de 2016). *The Big Data Blog*. Obtenido de Hadoop Ecosystem Overview: http://thebigdatablog.weebly.com/blog/the-hadoop-ecosystem-overview

Brewer, E. (2000). *Toward Robust Distributed Systems*. Obtenido de http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

Buhler, P., Erl, T., & Khattak, W. (2015). *Big Data Fundamentals: Concepts, Drivers & Techniques.*

Chauhan, A. (2012). *Blog Microsoft*. Obtenido de https://blogs.msdn.microsoft.com/avkashchauhan/2012/02/27/primary-namenode-and-secondary-namenode-configuration-in-apache-hadoop/

Comercio, C. d. (2016). *codigo de derecho*. Obtenido de https://derechovenezolano.wordpress.com/2012/11/01/el-codigo-de-comercio-el-registro-mercantil-concepto-documentos-sujetos-a-registro-efectos/

Conway, D. (30 de Septiembre de 2010). *Drew Conway Venn Diagram*. Obtenido de Drew Conway: http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

Dean, J., & Ghemawat, S. (2003). *Research Google Inc*. Obtenido de http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0008.html

Española, R. A. (Marzo de 2016). *Diccionario de la lengua española*. Obtenido de http://dle.rae.es/?id=Vj40asb

Evans, E. (12 de 05 de 2009). *NOSQL 2009. May 2009. – Blog post of 2009-05-12*. Obtenido de http://blog.sym-link.com/2009/05/12/nosql_2009.html

Eyssautier de la Mora, M. (2006). *Metodología de la investigación: desarrollo de la inteligencia.* Cengage Learning Editores.

Gutierrez, D. (Junio de 2014). *Yarn is All the Rage at Hadoop Summit 2014*. Obtenido de Kdnuggets: http://www.kdnuggets.com/2014/06/yarn-all-rage-hadoop-summit.html

Hanmer, R. (2013). Pattern-Oriented Software Architecture for Dummies. En R. Hanmer. Wiley.

IBM. (2015). *IBM*. Obtenido de Infografia: http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Kitchenham, B. (1996). DESMET: A method for evaluating software engineering methods and tools.

Lowell, U. (2013). *Computer Science*. Obtenido de Umass Lowell: http://www.cs.uml.edu/~jlu1/doc/source/report/MapReduce.html

MacCandles, M., Hatcher, E., & Gospodnetic, O. (2010). *Lucene in Action.* Manning.

Magazine, L. V. (10 de noviembre de 2013). Big Data, el tesoro oculto del siglo XXI.

Manifesto, A. (2001). *Agil Manifesto*. Obtenido de http://agilemanifesto.org/

Marco. (8 de Mayo de 2013). *Slideshare*. Obtenido de Seminario Apache Solr: http://es.slideshare.net/paradigmatecnologico/seminario-apache-solr

Morgan, L. (5 de Abril de 2015). *Information Week*. Obtenido de Information Week Big Data: http://www.informationweek.com/big-data/big-data-analytics/6-ways-to-master-the-data-driven-enterprise/d/d-id/1320234

Moya, D. d. (2002). *El Proyecto Factible: una modalidad de investigación.* Caracas: Sapiens.

Murthy, A. (15 de Octubre de 2013). *Apache Hadoop*. Obtenido de Hortonworks: http://hortonworks.com/blog/apache-hadoop-2-is-ga/

NoSQL. (2016). *NoSQL Database* . Obtenido de NoSQL Database org: http://nosql-database.org/

Pedraz, A. (2004). *La revisión bibliográfica.*

Pritchett, D. (2008). *BASE: An Acid Alternative*. Obtenido de http://queue.acm.org/detail.cfm?id=1394128

Sabino, C. (1992). *EL PROCESO DE INVESTIGACIÓN.* Caracas.

Sánchez, F. M. (2014). Herramientas para Big Data: Entorno Hadoop. 59.

SAREN. (2016). *Organigrama*. Obtenido de Servicio Autonomo de Registros y Notarias: http://www.saren.gob.ve/?page_id=15

SAREN. (2016). *SAREN*. Obtenido de Funcionó como Dirección General Sectorial de Registros y Notarías en el año de 1.994, y como Dirección General de Registros y Notarias a principios del año de 1.996.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2006). The Hadoop Distributed File System.

Soares, S. (2012). *Dataversity*. Obtenido de http://www.dataversity.net/not-your-type-big-data-matchmaker-on-five-data-types-you-need-to-explore-today/

Strozzi, C. (2010). *NoSQL – A relational database management system*. Obtenido de http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page

*The Register*. (2006). Obtenido de http://www.theregister.co.uk/2006/08/15/beer_diapers/

Vance, A. (2009). *Hadoop, a Free Software Program, Finds Uses Beyond Search.* New York: New York Times.

Wall, M. (6 de Marzo de 2014). *BBC*. Obtenido de BBC Mundo: http://www.bbc.com/mundo/noticias/2014/03/140304_big_data_grandes_datos_rg

White, T. (2012). Hadoop: The definitive guide. 41-47.

White, T. (2012). *Hadoop: The Definitive Guide* (3rd ed.). O'Reilly Media.

Wikipedia. (2015). *Motores de busqueda*. Obtenido de https://es.wikipedia.org/wiki/Motor_de_b%C3%BAsqueda

Wikipedia. (2016). *Apache Solr*. Obtenido de https://es.wikipedia.org/wiki/Apache_Solr

Wikipedia. (2016). *Stub Method*. Obtenido de https://en.wikipedia.org/wiki/Method_stub

Wikipedia. (2016). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Registro_mercantil

winshuttle. (2016). *winshuttle.com*. Obtenido de http://www.winshuttle.es/big-data-historia-cronologica/

# ANEXO 1: Pentaho vs Palo

| Leyenda: | Pentaho |
|---|---|
| | Palo |

| TIPO | DESCRIPCION DEL CRITERIO EVALUADO | CONDICION | Peso | CUMPLIMIENTO | OBSERVACIONES | Estrategia | Calificación Ponderada |
|---|---|---|---|---|---|---|---|
| REQUERIMIENTOS GENERALES | Debe poseer licencia de código libre o abierto | O-Obligatorio | 5 | S-Si Cumple | Libre: Apache License 2.0 | 6 | 5,00 |
| | | | | S-Si Cumple | Distribuido en versiones bajo licencias: GPLv2, LGPL, y Propietario | 6 | 5,00 |
| | Documentación | O-Obligatorio | 5 | S-Si Cumple | | 5 | 4,17 |
| | | | | S-Si Cumple | | 6 | 5,00 |
| | Comunidad | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 5 | 4,17 |
| REQUERIMIENTOS ESPECIFICOS | Debe permitir modelo de almacenamiento ROLAP | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 6 | 5,00 |
| | Debe ser capaz de trabajar Offline | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | N-.No Cumple | | 0 | 0,00 |
| | Debe manejar lenguajes XMLA (XML for Analysis) | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 6 | 5,00 |
| | Permitir procedimiento almacenados | O-Obligatorio | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | Cube Rules, SVS Triggers | 6 | 4,00 |
| | Permitir funciones personalizadas | O-Obligatorio | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 6 | 4,00 |
| | Debe soportar lenguaje SQL | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | N-.No Cumple | | 0 | 0,00 |
| | Capacidad de visualización de datos avanzada | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 4 | 2,67 |
| | Análisis en tiempo real | O-Obligatorio | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 3 | 2,00 |
| | Funciones semi-aditivas | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 3 | 2,00 |
| | | | | | | Calificación Total | **4,43** |
| | | | | | | | 3,24 |

# ANEXO 2: Hadoop vs Otros

| Leyenda: | Hadoop | |
|---|---|---|
| | Otros | Incluye software propietario y/o proyectos incipientes en el área: Disco, Sphere, Elastic Phonix, HPCC, Zillabyte, Apache Flink |

| TIPO | DESCRIPCION | CONDICION | Peso | CUMPLIMIENTO | OBSERVACIONES | Estrategia | Calificación Ponderada |
|---|---|---|---|---|---|---|---|
| REQUERIMIENTOS GENERALES | Soporte de operaciones Map Reduce | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 5 | 4,17 |
| | Mecanismos de acceso, procesamiento y análisis de grandes volúmenes de datos | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 4 | 3,33 |
| | Soporte de operaciones en tiempo real | D-Deseable | 4 | S-Si Cumple | Apache Storm | 6 | 4,00 |
| | | | | S-Si Cumple | | 0 | 0,00 |
| REQUERIMIENTOS ESPECIFICOS | Debe poseer licencia de código libre o abierto | O-Obligatorio | 5 | S-Si Cumple | Libre: Apache License 2.0 | 6 | 5,00 |
| | | | | S-Si Cumple | | 0 | 0,00 |
| | Documentación | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 4 | 3,33 |
| | Comunidad | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | | S-Si Cumple | | 3 | 2,50 |
| | Proporciona algoritmos de aprendizaje automático escalables | D-Deseable | 4 | S-Si Cumple | Apache Mahout | 6 | 4,00 |
| | | | | S-Si Cumple | | 0 | 0,00 |
| | Gobernabilidad e integración de datos | O-Obligatorio | 5 | S-Si Cumple | Apache Falcon, Apache Flume, Apache Sqoop | 6 | 5,00 |
| | | | | S-Si Cumple | | 0 | 0,00 |
| | Seguridad | O-Obligatorio | 5 | S-Si Cumple | Apache Knox | 6 | 5,00 |
| | | | | S-Si Cumple | | 3 | 2,50 |
| | Mecanismo de replicación | O-Obligatorio | 5 | S-Si Cumple | Manejo de Clusters | 6 | 5,00 |
| | | | | S-Si Cumple | | 4 | 3,33 |
| | Tolerancia a fallos y recuperación de errores | O-Obligatorio | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 3 | 2,00 |
| | Manejo de scripts del lado del servidor | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | | S-Si Cumple | | 0 | 0,00 |
| | | | | | | Calificación | 4,67 |
| | | | | | | Total | 1,76 |

# ANEXO 3: Cassandra vs Hbase

**CUMPLIMIENTO DE ESPECIFICACIONES TECNICAS**

**MATRIZ DE EVALUACION PARA:** BD NoSQL Transaccional

**OFERENTE:** PHD 2014 CONSULTORES, C.A

| Leyenda: | Cassandra |
|---|---|
| | Hbase |

| TIPO | DESCRIPCION | CONDICION | Peso | CUMPLIMIENTO | OBSERVACIONES | Estrategia | Calificación Ponderada |
|---|---|---|---|---|---|---|---|
| REQ. FUNCIONALES | Debe permitir lectura intensiva de datos | O-Obligatorio | 5 | S-Si Cumple | Cassandra tiene un excelente rendimiento de lectura de una sola fila, | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Optimizado para realizar lecturas basados en rango | 6 | 5,00 |
| | Debe permitir alta disponibilidad | O-Obligatorio | 5 | S-Si Cumple | Basado en concepto de anillo | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 5 | 4,17 |
| | Debe poseer alto rendimiento | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 6 | 5,00 |
| | Debe poseer API's u otros métodos de acceso | O-Obligatorio | 5 | S-Si Cumple | CQL(Cassandra Query Language, SQL-LIKE, Apache Thirft, Datastack API | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Java API, RESTful HTTP API,Thrift | 6 | 5,00 |
| | Debe poseer métodos de particionado de data | O-Obligatorio | 5 | S-Si Cumple | Sharding | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Sharding | 6 | 5,00 |
| | Debe poseer métodos de replicación | O-Obligatorio | 5 | S-Si Cumple | nodos distribuidos geograficamente | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Metodos de redundancia basados en nodos simples | 4 | 3,33 |
| | Debe permitir Querying data | D-Deseable | 4 | S-Si Cumple | Incluye una variedad de herramientas para realizar consultas | 6 | 4,00 |
| | | | 4 | S-Si Cumple | | 6 | 4,00 |
| | Debe permitir transacciones | S-Suplementario | 4 | N-.No Cumple | No (Atomicidad y aislamiento soportados para operaciones simples) | 0 | 0,00 |
| | | | 4 | N-.No Cumple | no | 0 | 0,00 |
| | Debe permitir la concurrencia | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 6 | 5,00 |
| | Balanceo durante tareas intensivas de lectura y escritura | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | 4 | S-Si Cumple | | 0 | 0,00 |
| | CAP Theorem (Consistencia) | D-Deseable | 3 | S-Si Cumple | Eventual | 5 | 2,50 |
| | | | 3 | S-Si Cumple | Fuerte | 6 | 3,00 |
| | Debe poseer tipos de datos predefinidos | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | 4 | S-Si Cumple | | 6 | 4,00 |
| | Debe permitir índices secundarios | D-Deseable | 3 | S-Si Cumple | | 6 | 3,00 |
| | | | 3 | S-Si Cumple | | 6 | 3,00 |
| | Debe permitir ejecutar script del lado del servidor | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | 4 | S-Si Cumple | | 6 | 4,00 |
| | Debe permitir ejecutar funciones MapReduce | D-Deseable | 5 | S-Si Cumple | | 5 | 4,17 |
| | | | 5 | S-Si Cumple | | 6 | 5,00 |
| | Mecanismo de replicación síncrono | D-Deseable | 4 | S-Si Cumple | Write.QUORUM or Write.EACH-QUORUM | 6 | 4,00 |
| | | | 4 | N-.No Cumple | | 0 | 0,00 |
| | Mecanismo de replicación asíncrona | D-Deseable | 3 | S-Si Cumple | Replica's can span data centers | 6 | 3,00 |
| | | | 3 | S-Si Cumple | | 6 | 3,00 |
| | Debe permitir control de acceso de usuarios | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 4 | 3,33 |
| REQ. ESPECIFICOS | Debe poseer licencia de código libre o abierto | O-Obligatorio | 5 | S-Si Cumple | Libre: Apache License 2.0 | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Libre: Apache License 2.0 | 6 | 5,00 |
| | Documentación | O-Obligatorio | 5 | S-Si Cumple | | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 4 | 3,33 |
| | Comunidad | O-Obligatorio | 5 | S-Si Cumple | Amplia comunidad de desarrolladores, redes sociales, grupos de usuarios, foros, entre otros | 6 | 5,00 |
| | | | 5 | S-Si Cumple | | 3 | 2,50 |
| | Esquema de datos: schema-free | D-Deseable | 4 | S-Si Cumple | | 6 | 4,00 |
| | | | 4 | S-Si Cumple | | 6 | 4,00 |
| | Lenguajes de programación soportados | O-Obligatorio | 5 | S-Si Cumple | C#,Erlang,Go,Haskell,Java,JavaScript,Perl,PHP,Python, Ruby, Scala, C++ | 6 | 5,00 |
| | | | 5 | S-Si Cumple | C,C#,C++,Groovy,Java,PHP,Python,Scala | 4 | 3,33 |
| | Sistemas Operativos soportados | O-Obligatorio | 5 | S-Si Cumple | BSD,Linux,OSX,Windows | 6 | 5,00 |
| | | | 5 | S-Si Cumple | Linux,Unix,Windows | 4 | 3,33 |
| | | | | | | Calificación Total | **4,24** |
| | | | | | | | 3,31 |

## ANEXO 4: Solr vs Elasticsearch

| Leyenda | Solr |
|---|---|
| | Elasticsearch |

| Tipo | Descripcion | Condicion | Peso | Cumplimiento | Observaciones | Estrategia | Calificacion Ponderada |
|---|---|---|---|---|---|---|---|
| Req. Funcionales | Replicacion | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | **S-Si Cumple** | | 6 | 5,00 |
| | Facetado | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | **S-Si Cumple** | | 6 | 5,00 |
| | Escalabilidad | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | **S-Si Cumple** | | 6 | 5,00 |
| | Busqueda orientada a texto | | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | **S-Si Cumple** | ES provee queries complejos de análisis | 5 | 4,17 |
| | Busqueda geo-espacial | D-Deseable | 4 | **S-Si Cumple** | | 6 | 4,00 |
| | | | | **S-Si Cumple** | | 6 | 4,00 |
| | Visualizacion | S-Suplementario | 4 | **S-Si Cumple** | | 6 | 4,00 |
| | | | | **S-Si Cumple** | | 6 | 4,00 |
| Req. Especificos | Debe poseer licencia de código libre o abierto | O-Obligatorio | 5 | **S-Si Cumple** | Licencia de Apache | 6 | 5,00 |
| | | | | **S-Si Cumple** | Licencia de Apache | 6 | 5,00 |
| | Documentación | O-Obligatorio | 5 | **S-Si Cumple** | Solr se encuentra muy bien documentado. | 6 | 5,00 |
| | | | | **S-Si Cumple** | | 5 | 5,00 |
| | Comunidad | D-Deseable | 4 | **S-Si Cumple** | La comunidad es muy amplia y puede realizar cambios. Aporte de varias compañías | 6 | 4,00 |
| | | | | **S-Si Cumple** | La comunidad tiene acceso al código, pero solo la compañía puede hacer cambios | 5 | 3,33 |
| | Sistemas operativos soportados | O-Obligatorio | 5 | **S-Si Cumple** | GNU/Linux, Unix, Windows | 6 | 5,00 |
| | | | | **S-Si Cumple** | GNU/Linux, Unix, Windows | 6 | 5,00 |
| Total | | | | | | | 4,7 |
| | | | | | | | 4,55 |

## ANEXO 5: HUE vs Banano

| Leyenda | Hue |
|---|---|

| | Banano |
|---|---|

| Tipo | Descripcion | Condicion | Peso | Cumplimiento | Observaciones | Estrategia | Calificacion Ponderada |
|---|---|---|---|---|---|---|---|
| Req. Funcionales | Req. Funcionales | Integración con Ecosistema Hadoop | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | Se integra con Solr y Solr al HDFS | 4 | 3,3 |
| | | Acceso al HDFS | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | Mediante Solr | 4 | 3,3 |
| | | Generación de Dashboards y gráficos | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | | 6 | 5,00 |
| | | Exportación de reportes y gráficos en diferentes formatos | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | | 6 | 5,00 |
| | | Plataforma fácil de desplegar | D-Deseable | 4 | **S-Si Cumple** | Basado en Web | 6 | 4,00 |
| | | | | | **S-Si Cumple** | Debe integrarse con Solr | 4 | 2,7 |
| Req. Especificos | Req. Especificos | Comunidad | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | | 6 | 5,00 |
| | | Gratuito y de licencia libre | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | | 6 | 5,00 |
| | | Documentación | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **S-Si Cumple** | | 5 | 5,00 |
| | | Seguridad | O-Obligatorio | 5 | **S-Si Cumple** | | 6 | 5,00 |
| | | | | | **N-No Cumple** | No posee autenticación | 0 | 0 |
| Total | | | | | **4,9** | | | |
| | | | | | **3,8** | | | |

# ANEXO 6: Diseño NoSQL SAREN

## Preliminares

Para la adecuada comprensión de los términos referentes a paradigmas, tecnologías y herramientas tecnológicas presentes en el informe; es necesario revisar los siguientes conceptos.

***Cassandra:*** es un sistema gestor de bases de datos distribuidas de código abierto diseñado para manipular grandes cantidades de datos a través de varios servidores proporcionando un servicio de alta disponibilidad sin puntos únicos de fallo.

***Column:*** Es el nivel más bajo del modelo de datos de Cassandra. Está compuesta por los elementos: column name, value y timestamp.

***Column name:*** es el identificador con el que podremos acceder a ella para obtener o modificar el valor que contiene. Es único y no puede haber dos iguales en el mismo conjunto de columnas.

***Value:*** es el dato de una columna. Es el único elemento modificable por el usuario. Se puede definir la validación de éste valor estableciendo así el tipo de dato que contiene.

Hay varios tipos de datos que acepta la validación entre los cuales se encuentran la codificación UTF-8, el tipo Long o el tipo Byte. Por defecto se establece el tipo Byte.

***Timestamp:*** nos indica cuándo se modificó por última vez esa columna. Éste elemento no es modificable por el usuario ya que se genera automáticamente al cambiar el campo value. Éste campo es único para las columns.

***Row:*** Es el siguiente nivel en el modelo de datos. Es un conjunto de columns identificado por una key o clave para acceder a él y es única en el conjunto de rows en el que se encuentra.

***Column Family:*** El siguiente nivel en el modelo de datos de Cassandra es la Column Family. Ésta contiene una colección de rows que podrán ser accedidas por su row key y está identificada por un nombre, el cual será único.

***Keyspace:*** El keyspace es el nivel más alto del modelo de datos. En él se definen todas las Column.

Al definir un Keyspace se pueden configurar los siguientes parámetros:

***Partitioner:*** éste parámetro especifica cómo se almacenarán las rows mediante
su key, lo cual nos permitirá decidir cómo se distribuirán las rows en los nodos.

***Replication factor:*** éste parámetro establece el número de nodos que actuarán
como copias de un conjunto de datos, normalmente rows, es decir, cuántas veces estarán repetidos los datos en el clúster.

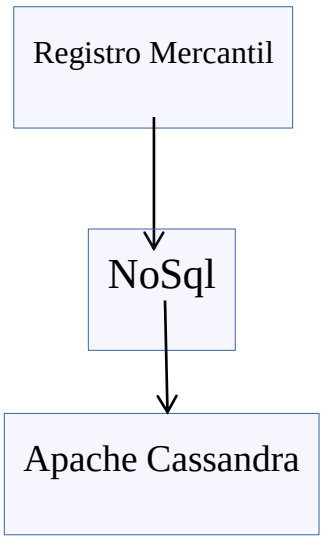
***Placement strategy:*** éste parámetro establece el modo en el que se replicarán los datos en el clúster. Tiene varias opciones para escoger.

**Dominios de Información Saren**

El Servicio Autónomo de Registros y Notarías para garantizar las actuaciones de los usuarios mediante la publicidad registral y fe pública, define dos grandes dominios de información:


Los cuales mediante sistemas informáticos facilitan el registro, modificación, auditoría y control de las actuaciones de los ciudadanos en materia mercantil e inmueble a nivel nacional.

Estos dominios actualmente se encuentran implementados bajo un enfoque relacional, el cual presenta restricciones ante el manejo de grandes volúmenes de datos.




En la figura anterior podemos apreciar que los dominios de información del modelo relacional; ahora están representados en un esquema no relacional, teniendo como principal habilitador tecnológico la aplicación Apache Cassandra.

Seguidamente se aprecia la relación entre la cadena de valor, el dominio de datos, el habilitador tecnológico y su correspondiente contenedor.

Para lograr representar los dominios de información antes descritos bajo un paradigma NoSQL, colocando toda la información en un ColumnFamily como se observó en la ilustración anterior, es necesario emprender un proceso de desnormalización del modelo relacional, para obtener una estructura de tripletas conformadas por claves, valores y marcas de tiempo.

El proceso de desnormalización no es una tarea simple ni de carácter mecánico, requiere de conceptos sólidos en ambos paradigmas (Relacional y NoSql), una clara comprensión del negocio en estudio y la aplicación de un conjunto de actividades que se describen a continuación:

Determinar las claves primarias y foráneas para las tablas en estudio.

Seleccionar y excluir las tablas que no serán objeto de análisis.

Interrogarse sobre cuáles son las consultas que el modelo debe responder.

Diseñar el nuevo column family.

Establecer criterios para el manejo de la auditoría.

Aplicar el proceso de desnormalización.

Construir claves y valores.

Probar el modelo e implementarlo.

Las actividades anteriormente señaladas constituyen un conjunto de observaciones a considerar para la aplicación del proceso de desnormalización de forma exitosa. Seguidamente se describe el proceso de desnormalización para los dominios Mercantil e Inmueble

| Tabla | Descripción | Clave Primaria | Clave Foránea |
|---|---|---|---|
| dadmbeneficiario | identificador del beneficiario | idbeneficiario | {tabla:doficina fk(idoficina)}, {tabla:dadmabonoordenpago fk(idabonoordenpago)}, {tabla:nadmtipobeneficiario fk(idtipobeneficiario)} |
| dadmbeneficiariocliente | registra los beneficiarios que son personas | idbeneficiario | {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioedescent | registra los beneficiarios que son entes públicos o privados | idbeneficiario | {tabla:dadmente fk(identedescentralizado)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioerecaudador | registra los beneficiarios que son entes recaudadores | idbeneficiario | {tabla:dadmenterecaudador fk(identerecaudador)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioproveedor | registra los beneficiarios que son proveedores | idbeneficiario | {tabla:dadmproveedor fk(idproveedor)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioresponcaja | registra los beneficiarios responsables caja | idbeneficiario | {tabla:dfcarespfondocajachica fk(idresponsable)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioresponfondo | registra los beneficiarios responsables de los fondos | idbeneficiario | {tabla:dadmresponsable fk(idresponsable)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiariotrabajador | registra los beneficiarios trabajadores | idbeneficiario | {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbienesinventario | registra los bienes asociados a las cuentas patrimoniales por inventario | idbienservicio | {tabla:dadmcuentasinventario fk(idcuentapatrimonial)}, {tabla:nadmcatalogosbienesserv fk(idbienservicio)} |
| dadmconversioncuentasecas | conversión de cuentas económicas. permite relacionar las cuentas económicas con las cuentas presupuestarias | idcdoreconomico,idcdorpresupuestario | {tabla:nadmclasificadorpresupuest fk(idcdorpresupuestario)}, {tabla:nadmclasificadoreconomico fk(idcdoreconomico)} |

| | | | |
|---|---|---|---|
| | correspondientes | | |
| dadmconversioncuentasples | conversión de cuentas patrimoniales. permite relacionar las cuentas patrimoniales con las cuentas presupuestarias correspondientes | idcuentapatrimonial,idcdorpresupuestario | {tabla:nadmclasificadorpresupuest fk(idcdorpresupuestario)}, {tabla:nadmcuentaspatrimoniales fk(idcuentapatrimonial)} |
| dadmcuentascontroladas | registra las cuentas contables controladas | idcuenta | {tabla:nadmcuentaspatrimoniales fk(idcuenta)} |
| dadmcuentasinventario | registra las cuentas que serán relacionadas a los bienes por inventario | idcuentapatrimonial | {tabla:nadmcuentaspatrimoniales fk(idcuentapatrimonial)} |
| dadmcuentastiporetencion | registra la relación entre las cuentas patrimoniales y presupuestarias con los tipos de retenciones | idcuentatiporetencion | {tabla:nadmclasificadorpresupuest fk(idcdorpresupuestario)}, {tabla:nadmcuentaspatrimoniales fk(idcuentapatrimonial)} |

## Proceso de desnormalización

El proceso de desnormalización inicia con la obtención las claves primarias y foráneas para las tablas en estudio.

### 1 Identificación de claves primarias y foráneas

A continuación, se presenta una muestra de la referida actividad para los esquemas mercantil.

| Tabla | Clave Primaria | Clave Foránea |
|---|---|---|
| dadmbeneficiario | idbeneficiario | {tabla:doficina fk(idoficina)}, {tabla:nadmtipobeneficiario fk(idtipobeneficiario)}, |

| | | |
|---|---|---|
| | | {tabla:dadmabonoordenpago fk(idabonoordenpago)} |
| dadmbeneficiariocliente | idbeneficiario | {tabla:dadmbeneficiario fk(idbeneficiario)}, {tabla:dpersonacomun fk(idcliente)} |
| dadmbeneficiarioedescent | idbeneficiario | {tabla:dadmente fk(identedescentralizado)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioerecaudador | idbeneficiario | {tabla:dadmenterecaudador fk(identerecaudador)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioproveedor | idbeneficiario | {tabla:dadmproveedor fk(idproveedor)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioresponcaja | idbeneficiario | {tabla:dfcarespfondocajachica fk(idresponsable)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiarioresponfondo | idbeneficiario | {tabla:dadmresponsable fk(idresponsable)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbeneficiariotrabajador | idbeneficiario | {tabla:dpersonacomun fk(idpersona)}, {tabla:dadmbeneficiario fk(idbeneficiario)} |
| dadmbienesinventario | idbienservicio | {tabla:dadmcuentasinventario fk(idcuentapatrimonial)}, {tabla:nadmcatalogosbienesserv fk(idbienservicio)} |
| dadmconversioncuentasecas | idcdoreconomico,idcdorpresupuestario | {tabla:nadmclasificadorpresupuest fk(idcdorpresupuestario)}, {tabla:nadmclasificadoreconomico fk(idcdoreconomico)} |
| dadmconversioncuentasples | idcuentapatrimonial,idcdorpresupuestario | {tabla:nadmclasificadorpresupuest fk(idcdorpresupuestario)}, {tabla:nadmcuentaspatrimoniales fk(idcuentapatrimonial)} |

## 2 Tablas que no serán objeto de estudio

Seguidamente se detallan las tablas para ambos dominios Mercantil e Inmueble que no serán objeto de estudio, posterior a los análisis realizados y la correspondiente revisión con el cliente.

| TABLAS | |
|---|---|
| DA | DADMDATOSBENEFICIARIOT |
| DADMESTRUCTURAPRET | DADMESTRUCTURAT |
| DADMIMPUTACIONT | DCOMDATOSOBLIGACIONOPT |
| DCOMDOCCOMPROMISOST | DFCAIMPUTACIONVALET |
| DFCATEMPLIBAUX | DPREEJECUCIONPRET |
| DPREEJEGPERIODOT | DPREOFICINASGASTOST |
| DPREPAGOANTICIPADOT | DPREPARTIDASUAT |
| DPREPRODMEST | DPREPROGPERIODOT |
| DPREPROGRAMACIONT | DRECTRAMITESPUBT |
| DRETRESUMENANUALRETISLRT | DTESLIBROAUXILIART |
| DTESORDENPAGOT | DTRAMITEACTOMULTIPLEM |
| NPRETEMPMESESMETAS | NPRETEMPPARTIDA |
| NRETCONFIGGEN | |

## 3 Diseño de Column Family

Seguidamente puede observarse, la estructura del Column Family denominado **SAREN** donde se almacena la información NoSQL en el motor de base de datos Cassandra.

| Clave | Column | Column | Valor | Column |
|---|---|---|---|---|
| Clave: Text | dominio: Text | tipoOperacion: Text | Valores: Map <text,text> | creadoPor: Date |
| | fechaCreacion: Date | ModificadoPor: Date | FechaUltimaModificacion: Date | TimeStamp |

El column family SAREN está conformado por ocho campos donde Clave representa el contenedor para el rowKey o clave primaria el cual puede ser simple o compuesta dependiendo de la naturaleza del registro.

El campo dominio indica la procedencia del trámite (Mercantil o Inmueble), el campo tipo de operación indica la naturaleza del acto, facilita el particionamiento de los datos en el clúster y eventualmente puede combinarse con la clave primaria para realizar búsquedas de mayor precisión.

El siguiente campo es *valor*, el cual almacena los valores del registro. De esto puede surgir una interrogante: si cada tipo de registro está asociado a una tabla del modelo relacional, entonces ¿cómo un campo del column family puede almacenar todos los campos de cada tabla? Esta precisamente es una de las ventajas de *cassandra*. Con el campo *valor* es posible tener tantos campos como sean necesarios por cada registro, producto que funciona como un map que tiene la siguiente estructura:

*valor = {'campo1':'valor1','campo2': valor2', ..., 'campoN': valorN'}*

Los otros campos son utilizados con fines de auditoría para cada registro.

**4 Construcción de los maps**

A continuación, se presenta un extracto del proceso de desnormalización para el dominio Mercantil.

| *TABLA* | *Clave Primaria* | *Dominio* | *Tipo de Operación* | *Map* |
|---|---|---|---|---|
| dtramitem | idtramitem | mercantil | solicitud_tramite | {idtramitem, fecha, numcons, idestadotramitem, descripcion_nestadotramitem, activo_nestadotramitem, idoficina, idcategoria_doficina, nombre_doficina, fechainicio_doficina, telefonos_doficina, numerofax_doficina, serialactivcn_doficina, direccion_doficina, idtipooficina_doficina, rif_doficina, iddocjuridico_doficina, idestadoactivacion_doficina, idmunicipio_doficina, certidumbre_doficina, idactom, abreviatura_nactom, descripcion_nactom, notariado_nactom, idclasifactom_nactom, descripcion_nclasifactom, activo_nclasifactom, idprotocolo_nactom, descripcion_nprotocolom, maxfolios_nprotocolom, activo_nprotocolom, activo_nactom, idtiposelloriginalm_nactom, descripcion_ntiposelloriginalm, nombrerpt_ntiposelloriginalm, activo_ntiposelloriginalm, abreviaturanaturaleza_nactom, asociaracompannia_nactom, tienerazonsocial_nactom, |

| | | | | |
|---|---|---|---|---|
| | | | | tienevariosotorgantes_nactom} |
| dtramitema | idtramitem | mercantil | otorgamiento_tramite | {idtramitem, fecha_dtramitem, numcons_dtramitem, idestadotramitem_dtramitem, idoficina_dtramitem, idactom_dtramitem, fechaotorg, fechainicio, fechafin, idabogadoredactor, idabogadorevisor, correo_dpersonalregistro, telefonos_dpersonalregistro, ocupacion_dpersonalregistro, otorgado, edicion} |
| Dtramitenotam | idnotaaportem, idtramitem | mercantil | registro_nota_aporte_tramite | {idnotaaportem, descripcion_nnotaaportem, nota_nnotaaportem, activo_nnotaaportem, idtramitem, fecha_dtramitem, numcons_dtramitem, idestadotramitem_dtramitem, idoficina_dtramitem, idactom_dtramitem} |
| dtramiteotorgadosm | idtramitem | mercantil | actualizar_tramite_otorgado | {iddenomin, denominacion_dsoldenomm, objeto_dsoldenomm, fecha_dsoldenomm, |

| | | | | |
|---|---|---|---|---|
| | | | | idtramitem_dsoldenomm, idestado_dsoldenomm, idpresentante_dsoldenomm, fechalimbusq_dsoldenomm, fechalimresnom_dsoldenomm, idtramitem, fecha_dtramitem, numcons_dtramitem, idestadotramitem_dtramitem, idoficina_dtramitem, idactom_dtramitem, idestado, descripcion_nestadosolicitudm, activo_nestadosolicitudm} |

**Conclusiones**

Al abordar un proyecto de BIG – DATA, uno de los factores críticos de éxito viene dado por el tratamiento que reciben los datos durante su ciclo de vida, especialmente en lo concerniente a su obtención, transformación, almacenamiento, organización lógica, veracidad, capacidad para responder a las consultas efectuadas por el negocio.

En el desarrollo del producto “Diseño Modelo de Base de Datos Big Data”, presenta un extenso análisis sobre los dominios de información Mercantil e Inmueble, donde se obtuvo una completa caracterización de la base de datos actual.

Para realizar el diseño de la Base de Datos Big Data sobresalen los siguientes elementos:

- El dominio Mercantil se diseñan íntegramente en un modelo no relacional, cuyo motor de datos es Apache Cassandra.

- Las tablas asociadas a los esquemas antes mencionados fueron estudiadas y desnormalizadas, generando un patrón constituido por clave, mapa de datos, tipo de operación, creado por, fecha de creación, modificado por, fecha última modificación y una marca de tiempo.

- La base de datos Big Data será poblada mediante un procedimiento de Carga Inicial basado en mecanismos de extracción, transformación y carga (ETL).

Por último, es importante destacar que el desarrollo de un modelo Big Data no radica únicamente en la incorporación de nuevas aplicaciones sobre arquitectura de software, es primordial comprender los conceptos que soportan este paradigma, al igual que las técnicas, principios, métodos y metodologías que se desprenden del referido enfoque.

# ANEXO 7: Diseño de los mecanisoms ETL

## Procesos de Extracción, Transformación y Carga

El mecanismo de ETL (Extracción, Transformación y Carga) facilita el movimiento de los datos entre diferentes sistemas aplicando métodos de Extracción, Limpieza, Organización, Transformación y Validación de información, de forma automatizada desde un Origen hacia otra base de datos.

La herramienta utilizada para construir los ETL es *Pentaho Data Integration (PDI)*. Esta herramienta cuenta con una interfaz gráfica que permite construir procesos de Extracción, Limpieza, Organización, Transformación y Validación de la Data.

Pentaho Data Integration no es un generador de código es un motor de transformaciones, las cuales son almacenadas en formato XML, donde se definen las acciones a realizar sobre los datos. Para construir las transformaciones, se utilizan pasos o componentes, que se enlazan entre sí mediante saltos que determinan el flujo acciones entre los diferentes componentes.

Para este proyecto se diseñaron un conjunto significativo de mecanismos ETL para la carga inicial de datos provenientes de 197 oficinas automatizadas.

## 1 Diseño de Mecanismos de ETL para la Carga Inicial

Seguidamente, se aborda en detalle los principios, elementos y características más destacadas de los mecanismos ETL, diseñados para la ejecución del proceso de Carga Inicial.

A continuación, se aprecia una representación de alto nivel del proceso de obtención de datos, desde las oficinas registrales y notariales hacia el esquema centralizado, mediante el uso de los mecanismos ETL.

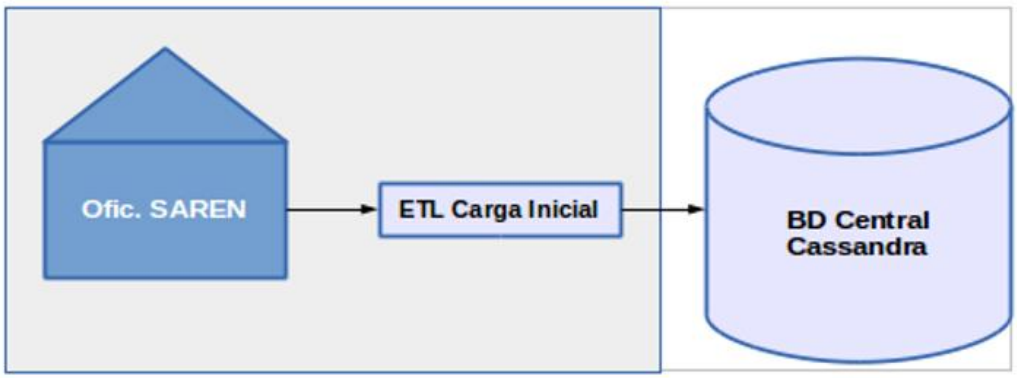



*Figura 23 Diseño ETL Saren*

## 2 Aspectos técnicos de los mecanismos de ETL de Carga Inicial.

Para el desarrollo de los ETL de carga inicial, es necesario tomar como punto de partida una tabla del esquema mercantil o público; procediendo de la siguiente forma:

- Se analizan las tablas revisando sus relaciones primarias y secundarias.
- Generar una consulta SQL capaz de extraer el contenido de la tabla principal junto con las tablas relacionadas.
- Se construye un campo MAP con la información relevante obtenida de la desnormalización.
- Se agregan los campos de auditoría.
- Se ejecuta un proceso de validación de datos y estructura.
- Posteriormente se inserta en la Base de Datos NoSql.

Observe el patrón de transformaciones usado por los mecanismos ETL, en los próximos párrafos comprenderá el comportamiento y detalle de cada uno de los pasos.

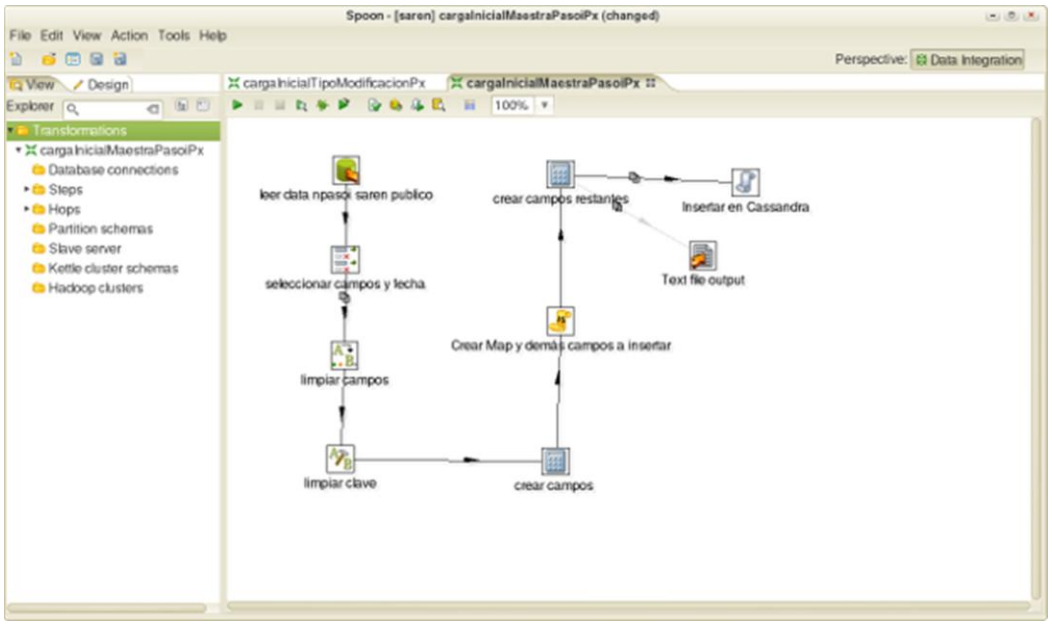



*Figura 24 Transformación Carga Inicial*

Cada transformación contiene Ocho (8) pasos, más Un (01) pasó de Control los cuales se detallan a continuación:

1. **Table Input:** Es el paso que inicia la transformación, donde se indica la consulta que extrae los datos que se requieren de la base de datos Oracle. En este paso se deben indicar:

    - **La Conexión a la Base de Datos**: para ello se utilizó una conexión compartida para todas las transformaciones. El nombre de la conexión es ***saren-publico***, donde se indica el *host* (Dirección IP del Servidor), *el nombre de la Base de Datos Oracle*, *el puerto a utilizar y* las *credenciales* de acceso (Usuario y Contraseña),

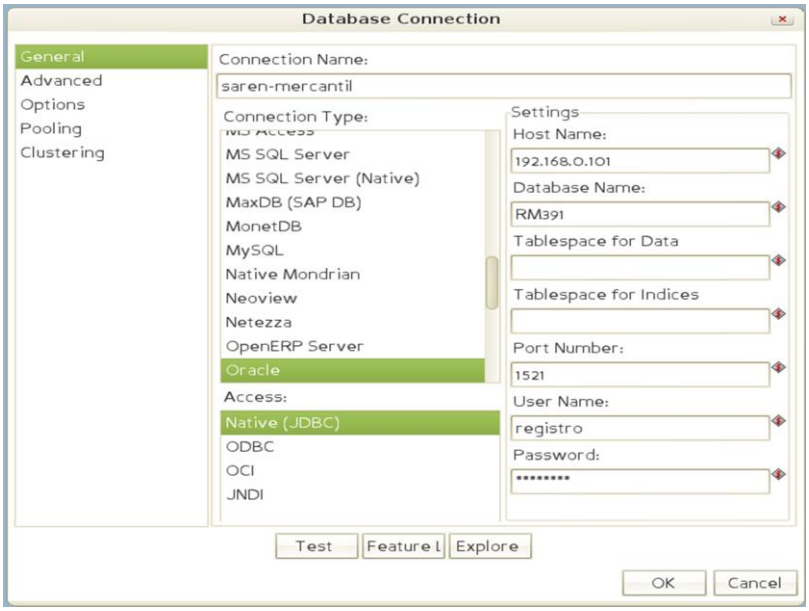



*Figura 25 Paso de conexión Base de datos Oracle*

    - **La Consulta SQL**: Es una consulta SQL de tipo *SELECT* para obtener los datos de la base de datos Oracle que se quieren cargar en la nueva base de datos. En esta consulta se añaden dos campos nuevos que son necesarios para insertar el registro en la nueva base de datos: ***"fecha_creacion"*** y ***"fecha_ultima_modificacion"*** que indican la fecha de creación y

fecha de última modificación del registro en la base de datos Big Data.

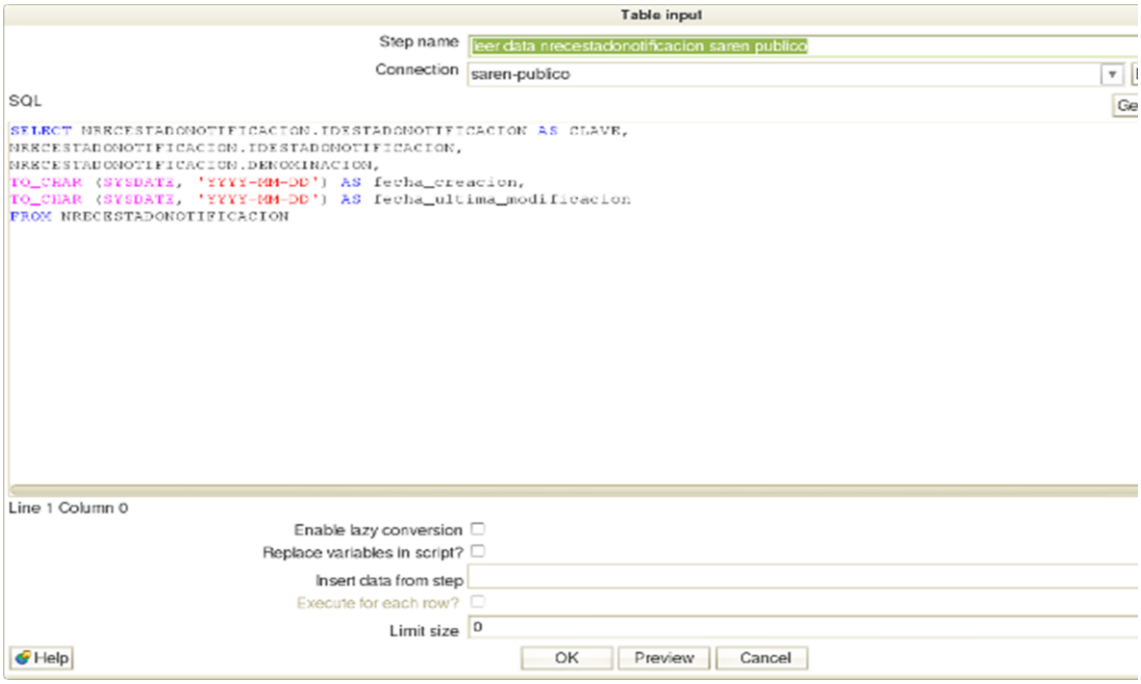



*Figura 26 Paso de consulta SQL sobre Oracle*

2. **Select Value:** En este paso se seleccionan los campos devueltos por el paso anterior (Table Input) y se indica además que el tipo de datos de los campos “clave”, *“fecha_creacion”* y *“fecha_ultima_modificacion”* es *String*. Lo anterior con el fin de evitar inconvenientes con la inserción en la base de datos Big Data.

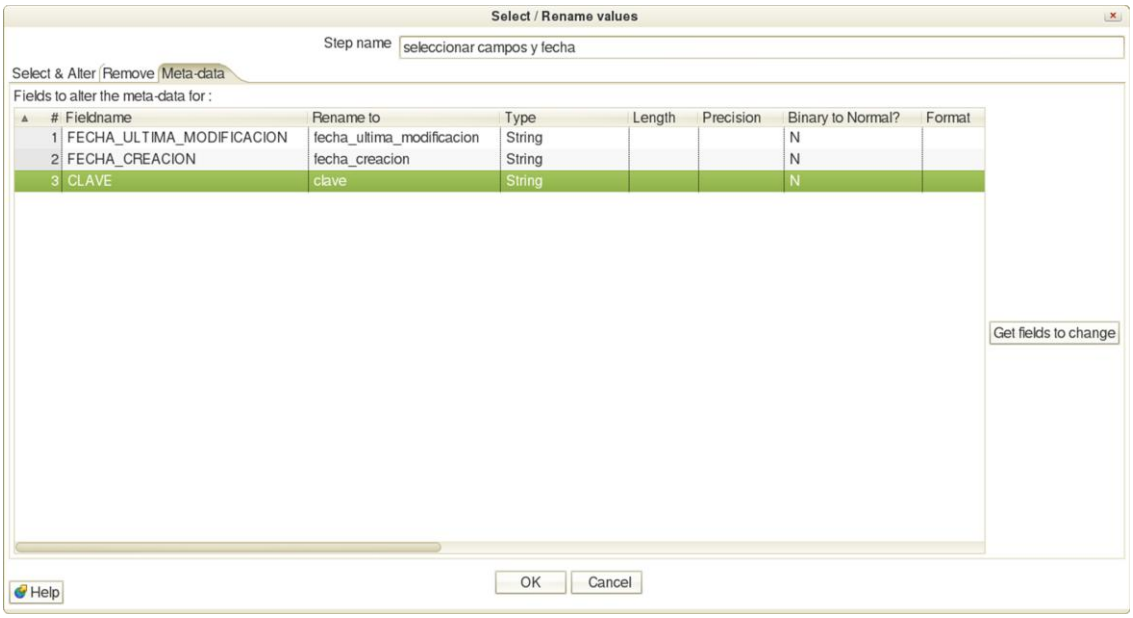



*Figura 27 Paso selección de valores*

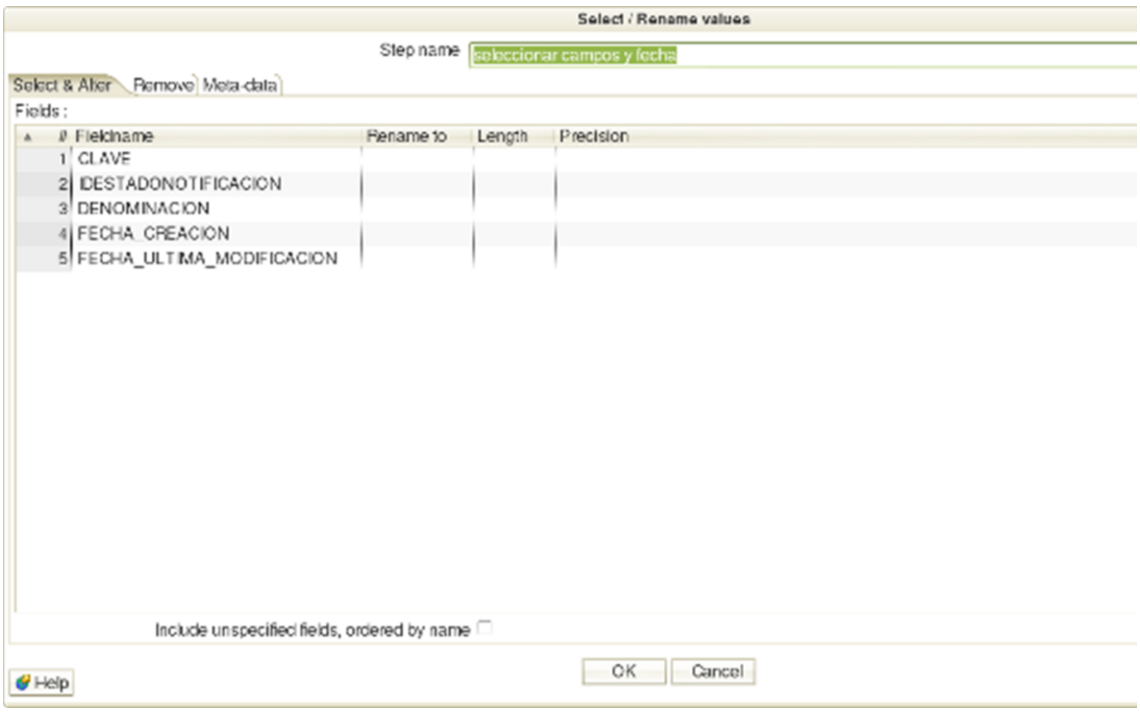



*Figura 28 Seleccionar los campos de la consulta*

3. **Replace in String:** Luego de buscar y seleccionar los campos requeridos para la creación del registro, se procede a limpiar los mismos, ya que existen datos con caracteres especiales que imposibilitan la inserción de la data en la base de datos Big Data.

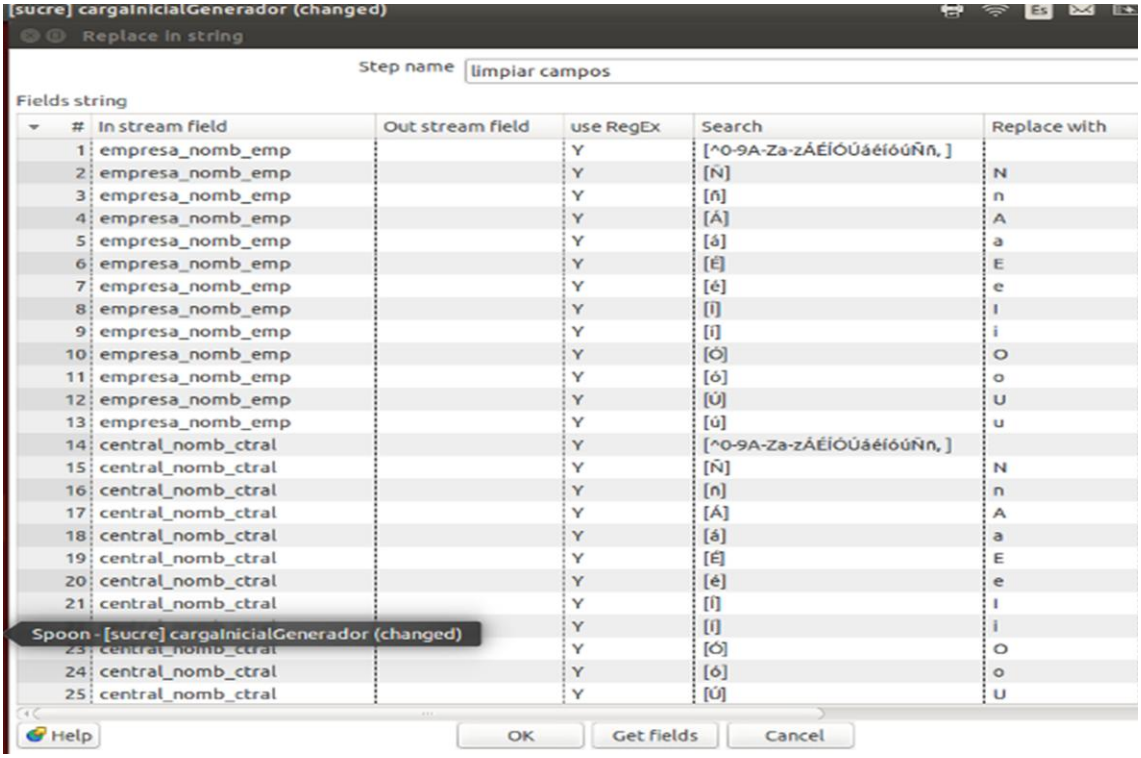



*Figura 29 Paso de limpieza de campos*

4. **Calculator:** Además de los campos *“fecha_creacion”* y *“fecha_ultima_modificacion”*, en el nuevo modelo se necesitan otros campos adicionales. En este paso se crean esos campos que son necesarios: *modificado_por, creado_por* y *tipo_registro*.

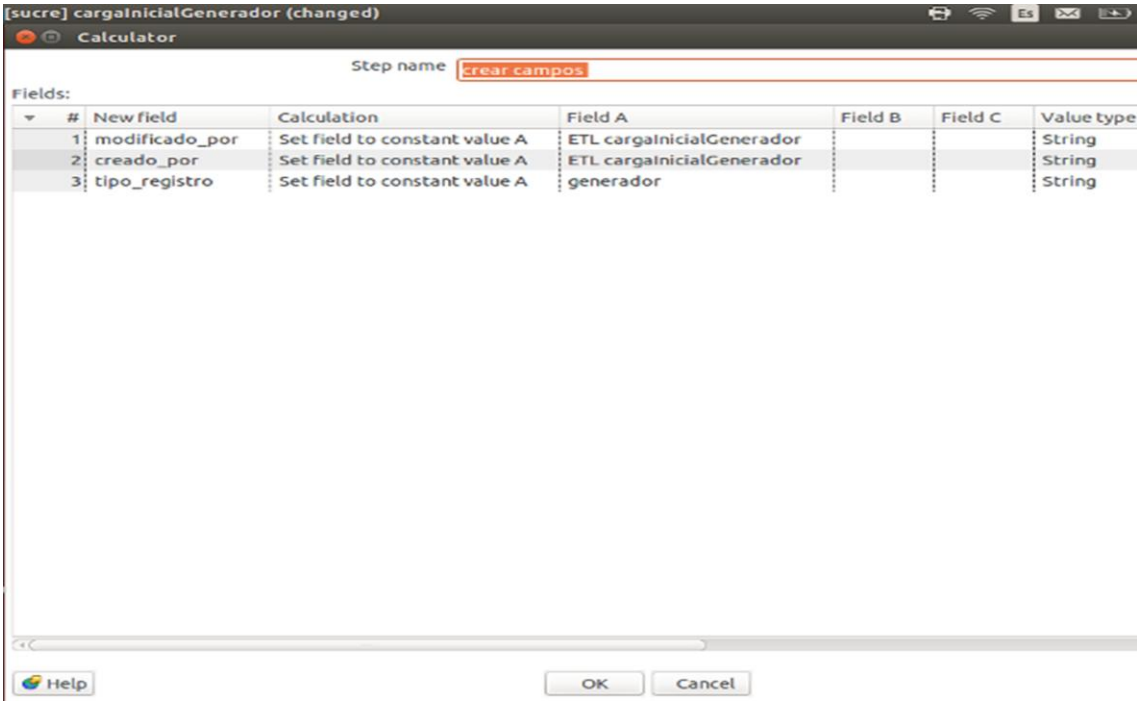



*Figura 30 Creación de campos nuevos*

**5 Modified Java Script Value**: Este es el paso más importante, realiza n conjunto de actividades adicionales mediante un script desarrollado en *Lenguaje de Programación Java* se obtienen cada uno de los campos de la tabla desde el modelo relacional (Oracle) y se transforman a un formato que pueda entender en nuevo modelo No Relacional (Big Data). Ahora, cada uno de los campos de la tabla se almacena en un solo campo de tipo *Map* con la siguiente estructura:

{'campo_1':'valor_campo_1', ..., 'campo_n':'valor_campo_n'}

También, en este paso se verifica cuáles de los campos tienen valor *null*. En este caso el campo no se almacena en el nuevo modelo, aprovechando así una de las grandes ventajas de la versión de la base de datos Big Data.

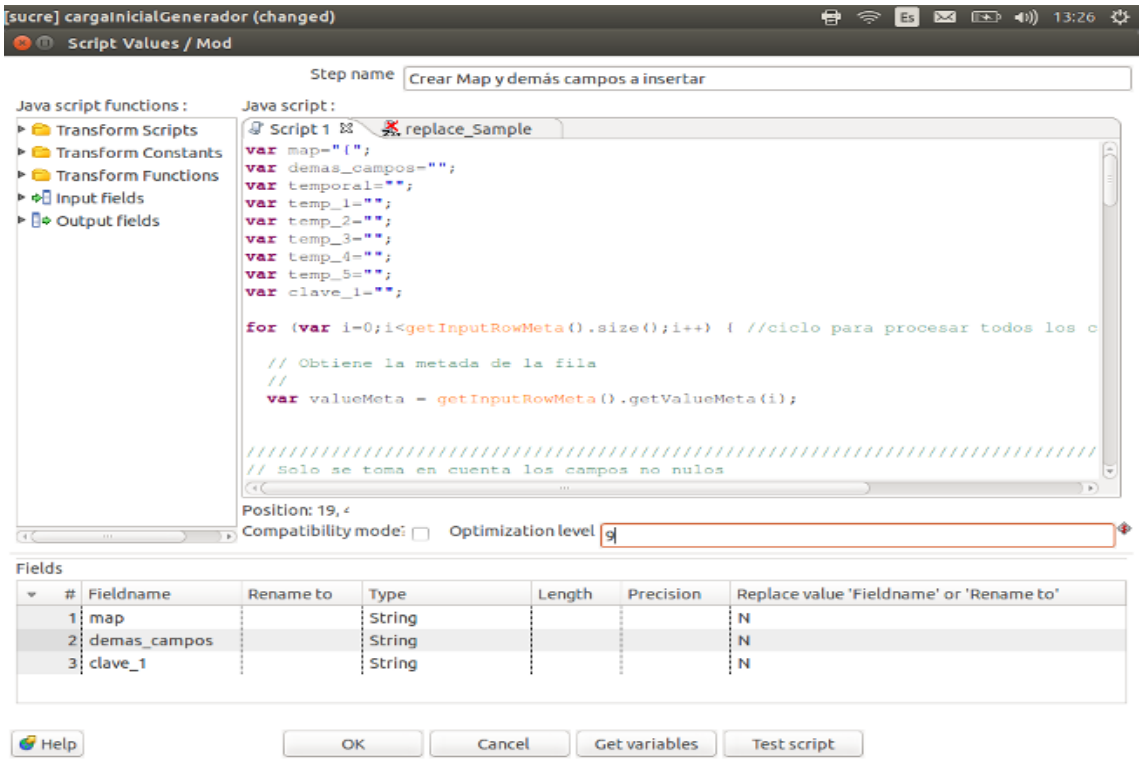



*Figura 31 Paso de Script de Java*

**6. Calculator:** Luego de crear el nuevo campo de tipo *Map* es necesario concatenarlo a la consulta de tipo *INSERT* que posteriormente se harán en base de datos Big Data para insertar el nuevo registro.

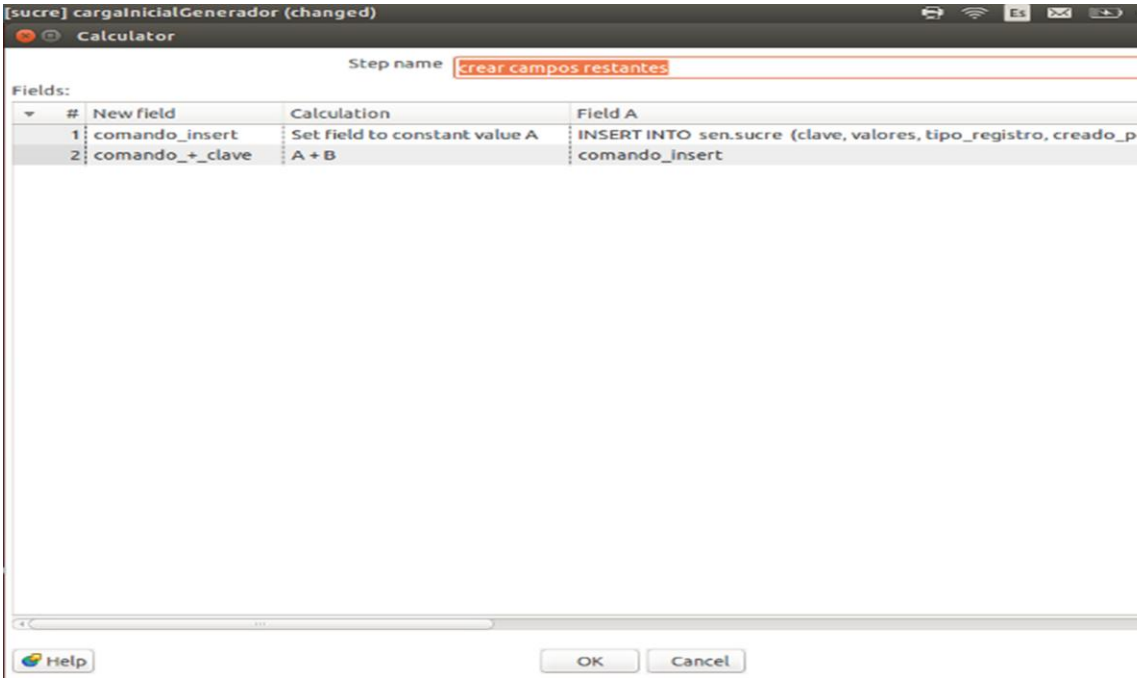



*Figura 32 Paso para crear los campos del map*

**7. File Output:** Finalmente, se obtienen cada uno de los *INSERT* generados en el paso anterior y se escriben en un archivo con un nombre con el siguiente formato:

Si el nombre de la tabla está compuesto por una palabra cargaInicial<*tiporegistro*>

*Si el nombre de la tabla está compuesto por dos o más palabras cargaInicial<tiporegistro>*

### 3.3 *Aspectos* técnicos de mecanismos de ETL para la carga de imágenes

Se utilizan los mismos pasos que en las transformaciones sin imágenes y se modifican de forma diferente los pasos: “Crear Map y demás campos a insertar” de la siguiente manera:

Crear Map y demás campos a insertar: Se agrega un var temp_0=""; para poder asignarle el campo que contenga el dato de la imagen y se procede a separarlo del map

```
var temp_4="";
var temp_5="";
var temp_6="";
var temp_0="";

var clave_1="";

for (var i=0;i<getInputRowMeta().size();i++) { //ciclo para procesa

  // Obtiene la metada de la fila
  //
  var valueMeta = getInputRowMeta().getValueMeta(i);


////////////////////////////////////////////////////////////////////////
// Solo se toma en cuenta los campos no nulos
//
// Los primeros 6 if, almacenan los campos tipo_registro, creado_po
// fecha_ultima_modificacion y oficina, en los campos temp_1, temp_
// respectivamente
// Luego se crea el campo map de tipo 'campo':''valor'
////////////////////////////////////////////////////////////////////////

if ((row[i]!=null)) {
    if (valueMeta.getName().equals("DCRL_CRL")){
        temp_0="'"+row[i]+"'"
    }else{
    if (valueMeta.getName().equals("tipo_registro")){
        temp_1="'"+row[i]+"'"
    }else{
```

*Figura 33 Paso Java script para inserción de imágenes*

En esta parte:
if ((row[i]!=null)) {
if (valueMeta.getName().equals("NOMBRE_CAMPO_IMAGEN")){
temp_0="'" +row[i]+"'" }

Luego se procede a modificar el campo “Insertar en Cassandra” de la siguiente manera:

a. Se crea una variable de tipo byte [] llamada “imagen”

```
Processor
String campo_host="";
String campo_keyspace="";
String campo_insert="";
byte[] imagen;
```

*Figura 34 Variable Byte buffer*

b. Se crean los campos necesarios en los parámetros de la función a insertar llamada guardarImagen()
c. A la variable “imagen” se le asigna el campo que contiene la imagen

```
public boolean processRow(StepMetaInterface smi, StepDataInterface sdi)th
{
    if (first) {
        campo_host= getParameter("HOST");
        campo_keyspace= getParameter("KEYSPACE");
        campo_insert=getParameter("INSERT");
        first=false;
    }

    Object[] r = getRow();

    if (r == null) {
        setOutputDone();
        return false;
    }

    String host=(get(Fields.In, campo_host).getString(r));
    String keyspace=(get(Fields.In, campo_keyspace).getString(r));
    String columnfamily=(get(Fields.In, "columnfamily").getString(r));
     imagen=(get(Fields.In, "DCRL_CRL").getBinary(r));
    String oficina=(get(Fields.In, "oficina").getString(r));
    String tipo_registro=(get(Fields.In, "tipo_registro").getString(r));
    String clave=(get(Fields.In, "clave").getString(r));
    String insert=(get(Fields.In, campo_insert).getString(r));
```

*Figura 35 Campos a insertar en Cassandra*

*Fuente: Elaboración propia*

d. Se agregan los parámetros con sus valores ya asignados en la función que inserta en cassandra “guardarImagen()”

```
public static ResultSet guardarImagen(String host,
                String keyspace, String columnfamily,String clave ,String oficina,String tipo_registro,
                byte[] imagen, String insert ) throws IOException, SQLException{

        hostName = host;
        keyspaceName = keyspace;

        //Se prepara la sentencia insert y se insertan todos los campos menos la imagen
        PreparedStatement preparedStatement = phconsultoresCassandra.obtenerSesion()
                        .prepare(insert);
        BoundStatement boundStatement = preparedStatement.bind();
        phconsultoresCassandra.obtenerSesion().execute(boundStatement);

        //Se prepara la imagen para hacerle el update

        ByteBuffer byteBuffer = ByteBuffer.wrap(imagen); //cassandra solo acepta Bytebuffer para imagenes
        Statement statement = QueryBuilder.update(keyspace, columnfamily)
                                                        .with(QueryBuilder.set("imagen", byteBuffer))
                                                        .where(QueryBuilder.eq("clave", clave))
                                                        .and(QueryBuilder.eq("tipo_registro", tipo_registro))
                                                        .and(QueryBuilder.eq("oficina",oficina ));


         return obtenerSesion().execute(statement);



}
```

*Figura 36 Código Función guardarImagen()*

*Fuente: Elaboración propia*

La función además de recibir el host y demás campos necesarios, recibe un parámetro llamado “insert” el cual es de tipo String y contiene los datos a insertar en cassandra, excepto la imagen o campo de tipo blob.

El parámetro imagen que recibe, es el campo que contiene la imagen y es de tipo byte[], se procede a convertir en ByteBuffer ya que es el tipo que maneja cassandra.

Se prepara el statement, que es un update y luego se procede a ejecutarse, retornando ese valor (ResultSet) el cual maneja el paso de insertar en cassandra del spoon.

# ANEXO 8: Documento Pruebas de carga volumen y estrés

*Tabla 13 Control de revisiones y cambios*

*Fuente: Elaboración propia*

| **Información del Documento** | |
|---|---|
| Nombre del Proyecto: | Proyecto Modernización y Migración de la Arquitectura de Base de datos de Oficinas Registrales y Notariales del SAREN. |
| Objeto del Contrato: | Suministrar Servicios Profesionales para la Modernización y Migración de la Arquitectura de Base de Datos de Oficinas Registrales y Notariales del SAREN. |
| Nombre del Producto: | Entonación y Pruebas de Carga, Volumen y Estrés. |
| Coordinador de Proyecto: | Ing. Jorge Goitia. |

**Resumen Ejecutivo**

El Servicio Autónomo de Registros y Notarías (SAREN), es un organismo adscrito al Ministerio del Poder Popular del Interior y Justicia, con la responsabilidad de garantizar la seguridad jurídica de los actos protocolizados y autenticados de los usuarios, mediante un sistema integral de registros y notarías confiable, eficiente, transparente e interconectado con otras dependencias del Estado venezolano.

Las competencias del Servicio Autónomo de Registros y Notarías en materia tecnológica, abarcan la planificación, equipamiento, supervisión y apoyo a todas las oficinas regionales con el fin de garantizar el acceso y disponibilidad de los servicios dispuestos al ciudadano en toda la geografía nacional.

A tales efectos, SAREN emprende esfuerzos en la adecuación y modernización de la plataforma tecnológica con el objetivo de manejar grandes volúmenes de datos de forma eficiente y confiable; regido sobre los principios de Big Data y arquitecturas orientadas a servicio, fortaleciendo las actividades de Minería de Datos, Inteligencia de Negocios y Monitoreo de Actividades.

En este orden de ideas, el siguiente documento describe el diseño, desarrollo e implementación de las pruebas de carga, volumen y estrés; un conjunto de mecanismos utilizados para evaluar el rendimiento, estabilidad de una

plataforma y los sistemas en ejecución, a los fines de determinar las acciones requeridas para garantizar tiempos de respuesta y desempeño óptimos, tanto del aplicativo como de la plataforma tecnológica.

### 2.1 Objetivo del Documento

Presentar los resultados, observaciones y recomendaciones, derivados de la ejecución del plan de pruebas (Carga – Volumen y Estrés), diseñado para el proyecto de modernización y migración de la Arquitectura de Base de Datos de Oficinas Registrales y Notariales del SAREN.

### 2.2 Alcance

Describir de forma estructurada los procesos inherentes al plan de pruebas (Carga, Volumen y Estrés).

Detallar las actividades, procedimientos, recursos y herramientas necesarias para la ejecución de las pruebas (Carga, Volumen y Estrés).

Evaluar el rendimiento de los componentes que conforman la plataforma tecnológica.

Presentar los resultados de forma tabular con sus observaciones y recomendaciones.

### Preliminares

Para la adecuada compresión de los términos, paradigmas, tecnologías, recursos, herramientas presentes en el desarrollo del plan de pruebas (Carga, Volumen y Estrés), es necesario revisar los siguientes conceptos.

**Pruebas de rendimiento:** se realizan para determinar velocidad, eficiencia y correctitud en las tareas ejecutadas por un sistema en condiciones particulares de trabajo. También se emplean para validar y verificar otros atributos de la calidad, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema.

**Pruebas de carga:** se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación.

**Pruebas de volumen:** Este tipo de prueba tiene por objetivo verificar si el sistema es estable durante un largo periodo de tiempo. Básicamente es como una prueba de carga con una duración superior, por ejemplo 24 horas. La idea es encontrar errores acumulativos, es decir, errores que pasan una prueba de esfuerzo por que producen un daño muy pequeño y que a la larga van a terminar deteriorando o colapsando el rendimiento del sistema.

**Pruebas de Estrés:** Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

**Pruebas de estabilidad**: Comprueban que no existe degradación del servicio por un uso prolongado del sistema.

**Pruebas unitarias:** Pruebas de rendimiento de los componentes desarrollados. Comprobaciones como:

tratamiento concurrente de las peticiones que recibe el módulo.

paralelización en las peticiones.

uso óptimo de los recursos (CPU, memoria, dispositivos de almacenamiento entre otros).

comprobación del número de peticiones concurrentes que recibe un componente hardware con diferentes configuraciones.

**Pruebas de Integración**: Pruebas de rendimiento para comprobar que los diferentes módulos o programas funcionan correctamente de manera integrada. Al igual que en pruebas funcionales se pueden utilizar técnicas de botton-up o top-down. Con estas pruebas también es posible detectar cuellos de botella en los componentes que se van integrando.

En otro orden de ideas, el marco metodológico requerido para la ejecución del plan de pruebas, se aplicará el círculo de Deming también conocido como PDCA (planificar-hacer-verificar-actuar).

El ciclo PDCA consta de 4 fases, las cuales se describen a continuación:

## PLAN (PLANIFICAR)

En ésta fase se establecen los objetivos y procesos necesarios para obtener el resultado esperado. Al basar las acciones en el resultado esperado, la exactitud y cumplimiento de las especificaciones a lograr se convierten también en un elemento a mejorar. Cuando sea posible conviene realizar pruebas a pequeña escala para probar los resultados.

Recopilar datos para profundizar en el conocimiento del proceso.

Detallar las especificaciones de los resultados esperados

Definir los procesos necesarios para conseguir estos objetivos, verificando las especificaciones.

## DO (HACER)

Implementar los nuevos procesos, llevar a cabo el plan. Recolectar datos para utilizar en las siguientes etapas.

Teniendo el plan bien definido, hay que poner una fecha a la cual se va a desarrollar lo planeado.

## CHECK (VERIFICAR)

Pasado un periodo previsto de antemano, volver a recopilar datos de control y analizarlos, comparándolos con los objetivos y especificaciones iniciales, para evaluar si se ha producido la mejora

Monitorizar la implementación y evaluar el plan de ejecución documentando las conclusiones.

## ACT (ACTUAR)

En base a las conclusiones del paso anterior elegir una opción:

Detección de errores parciales en el paso anterior, realizar un nuevo ciclo PDCA con nuevas mejoras.

Si no se han detectado errores relevantes, aplicar a gran escala las modificaciones de los procesos

Si se han detectado errores insalvables, abandonar las modificaciones de los procesos

Ofrecer una Retro-alimentación y/o mejora en la Planificación.

## Implementación del Ciclo PDCA

Seguidamente se amplía las diferentes fases del ciclo PDCA, requeridas para la aplicación del plan de pruebas.

**Fase 1: Planificar**

En esta sección se definieron las herramientas necesarias para llevar a cabo las pruebas de carga, volumen y estrés; destacando en el proceso de pruebas la necesidad de evaluar las métricas de comportamiento derivados del software Apache Cassandra y la plataforma tecnológica que la sustenta.

Para la selección de las herramientas que participaran el proceso de pruebas, se toma como referencia, las recomendaciones, buenas prácticas y sugerencias emitidas por la empresa Datastax, en sus diferentes artículos referentes al tema.

Para el análisis y rendimiento de hardware, fueron seleccionadas las siguientes herramientas:

**hdparm:** es una utilidad de línea de comandos de los sistemas operativos GNU/Linux y Windows para ver y ajustar los parámetros del hardware de los discos IDE y SATA1. La utilidad puede ajustar parámetros como el caché de disco, el modo de descanso, el control de energía, la gestión acústica y los ajustes DMA. Esta herramienta debe manipularse con extremo cuidado, un uso incorrecto puede acarrear perdida de datos o afectaciones sobre el disco.

**Iostat:** es una herramienta muy potente para monitorear el comportamiento de entrada y salida sobre los discos. El comando **iostat** genera informes que se pueden utilizar para cambiar la configuración del sistema para equilibrar mejor la carga de entrada / salida entre discos físicos.

**Análisis de Memoria:**

El almacenamiento dinámico en Java es un recurso limitado y en extremo valioso, incluso si la plataforma posee un número elevado de memoria RAM, debe emplearse de forma prudente. A continuación, se indica las herramientas a ser utilizadas para verificación y monitoreo de la memoria RAM y la JVM (Máquina Virtual de Java).

**Jmxterm:** es un cliente JMX interactivo basada línea de comandos. Está diseñado para permitir al usuario acceder a un servidor Java MBean de línea de comandos sin entorno gráfico. Resulta sencillo de utilizar en comparación con el jconsole que requiere la habilitación de un puerto.

**Pruebas Carga Volumen y Estrés Cluster Cassandra.**

La aplicación de las pruebas para carga, volumen y estrés, se ejecutarán completamente con el aplicativo cassandra-stress, el cual viene instalado en la distribución community de cassandra desde la versión 2.0.

Otros comandos utilitarios requeridos para las pruebas de carga volumen y estrés son:

nodetool cfstats: muestra información estadística para un column family en específico.

nodetool tpstats: provee estadísticas de uso de carácter general sobre el clúster

nodetool cfhistograms: proporciona un conjunto de estadísticas que pueden ser graficadas.

nodetool compactionstats: muestra información estadística sobre las operaciones de compactación ejecutadas sobre la base de datos.

**Determinación de Escenarios:**

Se contempló la conformación de dos escenarios para simular las condiciones de borde a la cual estará expuesta la plataforma tecnológica:

Bajo, nivel de transaccionalidad

Alto, nivel de transaccionalidad

**Instalación de las herramientas requeridas**

Para las pruebas de rendimiento de hardware fue necesario instalar las aplicaciones hdparm y iostat en los diferentes ambientes. Sobre este particular es importante aclarar, que en la actualidad SAREN dispone para este proyecto dos ambientes desarrollo y producción, no existe un ambiente de calidad. Por lo cual las pruebas se han realizado con las previsiones requeridas por el caso.

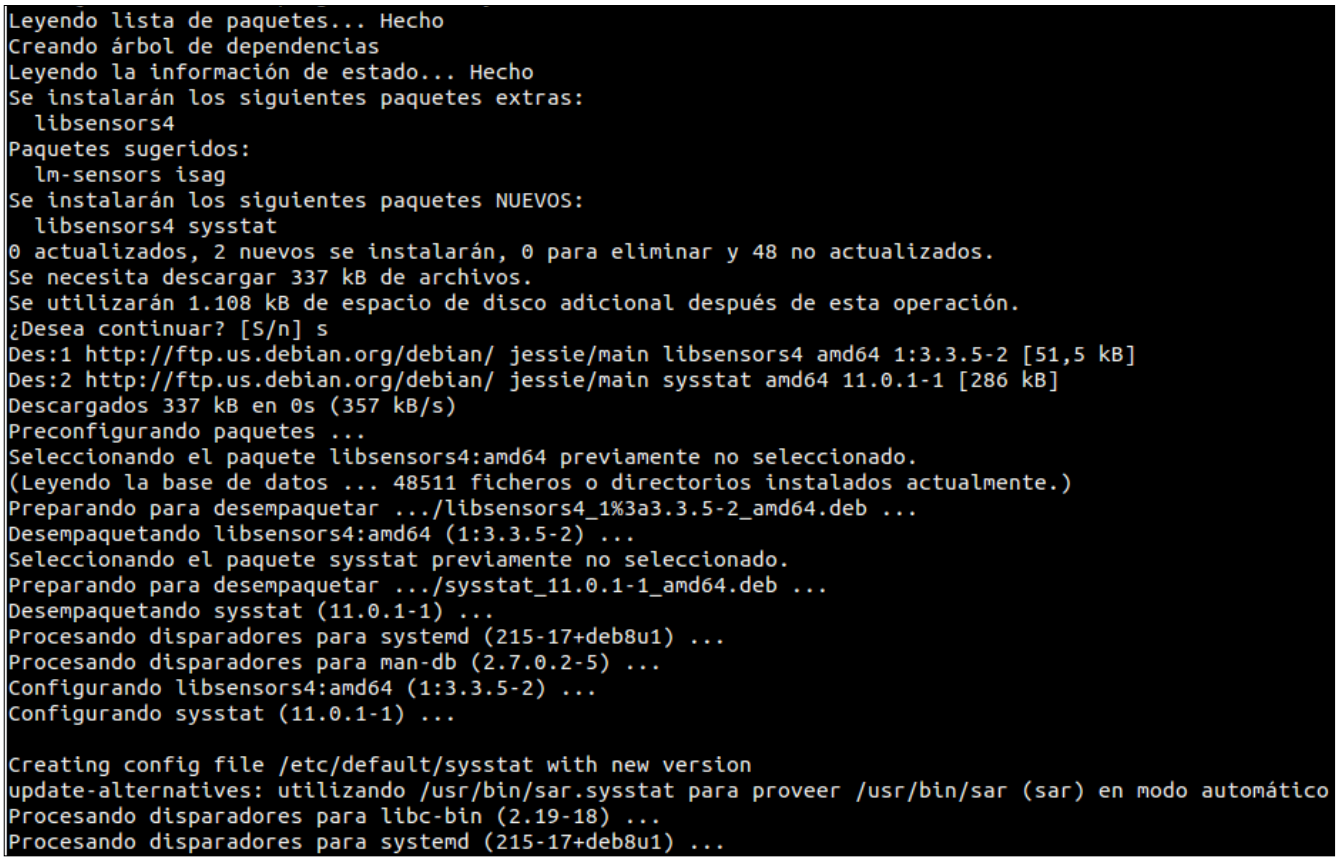



*Figura 37 Instalación de iostat*

*Fuente: Elaboración propia*

```
-bash: hdparm: no se encontró la orden
saren@BDDB1:~$ sudo  hdparm -t /dev/sda
sudo: hdparm: command not found
saren@BDDB1:~$ sudo apt-get install hdparm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  powermgmt-base
Paquetes sugeridos:
  apmd
Se instalarán los siguientes paquetes NUEVOS:
  hdparm powermgmt-base
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 48 no actualizados.
Se necesita descargar 115 kB de archivos.
Se utilizarán 349 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://ftp.us.debian.org/debian/ jessie/main hdparm amd64 9.43-2 [106 kB]
Des:2 http://ftp.us.debian.org/debian/ jessie/main powermgmt-base all 1.31+nmu1 [9.240 B]
Descargados 115 kB en 0s (159 kB/s)
Seleccionando el paquete hdparm previamente no seleccionado.
(Leyendo la base de datos ... 48603 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../hdparm_9.43-2_amd64.deb ...
Desempaquetando hdparm (9.43-2) ...
Seleccionando el paquete powermgmt-base previamente no seleccionado.
Preparando para desempaquetar .../powermgmt-base_1.31+nmu1_all.deb ...
Desempaquetando powermgmt-base (1.31+nmu1) ...
Procesando disparadores para systemd (215-17+deb8u1) ...
Procesando disparadores para man-db (2.7.0.2-5) ...
Configurando hdparm (9.43-2) ...
update-rc.d: warning: start and stop actions are no longer supported; falling back to defaults
Configurando powermgmt-base (1.31+nmu1) ...
Procesando disparadores para systemd (215-17+deb8u1) ...
```

*Figura 38 Instalación de hdparm*

**Jmxterm:** La instalación consiste en la ejecución de los siguientes pasos:

Descargar el archivo jmxterm-1.0-alpha-4-uber.jar

Posteriormente ejecutar de la siguiente forma java -jar jmxterm-<my version>-uber.jar.

Su ejecución y otros detalles serán abordados en la sección hacer del presente informe.

Fase 2: Hacer

En esta fase se inicia la ejecución de las pruebas programadas sobre el cluster de base de datos y el hardware que lo soporta.

**Pruebas de rendimiento sobre el hardware que conforma el cluster de base de datos:**

*Tabla 14 Prueba 1*

*Fuente: Elaboración propia*

| | |
|---|---|
| Titulo: | Prueba 1 |
| Descripción: | Rendimiento obtenido con una prueba de carga sostenida, migrando 12 millones de registro del clúster Oracle (instancia 72) al clúster Cassandra (direcciones 192.16.11.120/121/122). |
| Escenario: | Baja demanda |
| Herramienta | iostat |

Para la adecuada comprensión de la herramienta iostat, se presenta una tabla explicativa de cada uno de los parámetros que integran las estadísticas suministradas por el referido comando.

*Tabla 15 Comandos iostat*

*Fuente: Elaboración propia*

| ***Comandos*** | ***Descripción*** |
|---|---|

| | |
|---|---|
| r/s | Lecturas por segundo |
| w/s | Escrituras por segundo |
| Kr/s | Kbytes leídos por segundo |
| Kw/s | Kbytes escritos por segundo |
| wait | Número medio de transacciones que están en espera de servicio (longitud de cola) |
| actv | Número medio de transacciones que están siendo gestionadas de manera activa |
| svc_t | Tiempo medio de servicio (en milisegundos) |
| %w | Porcentaje de tiempo durante el cual la cola no está vacía |
| %b | Porcentaje de tiempo durante el cual el disco está ocupado |

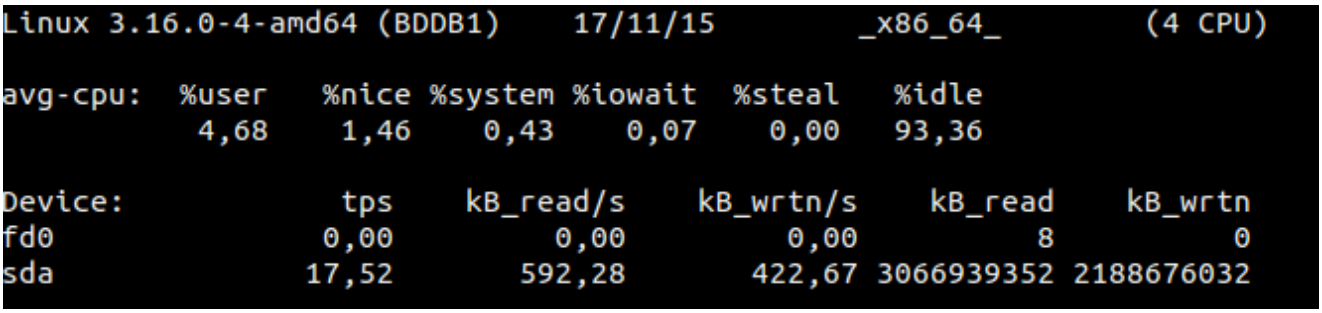



*Figura 39 Comando iostat*

*Fuente: Elaboración propia*

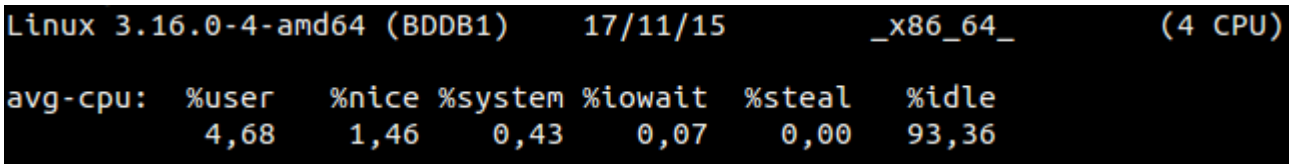
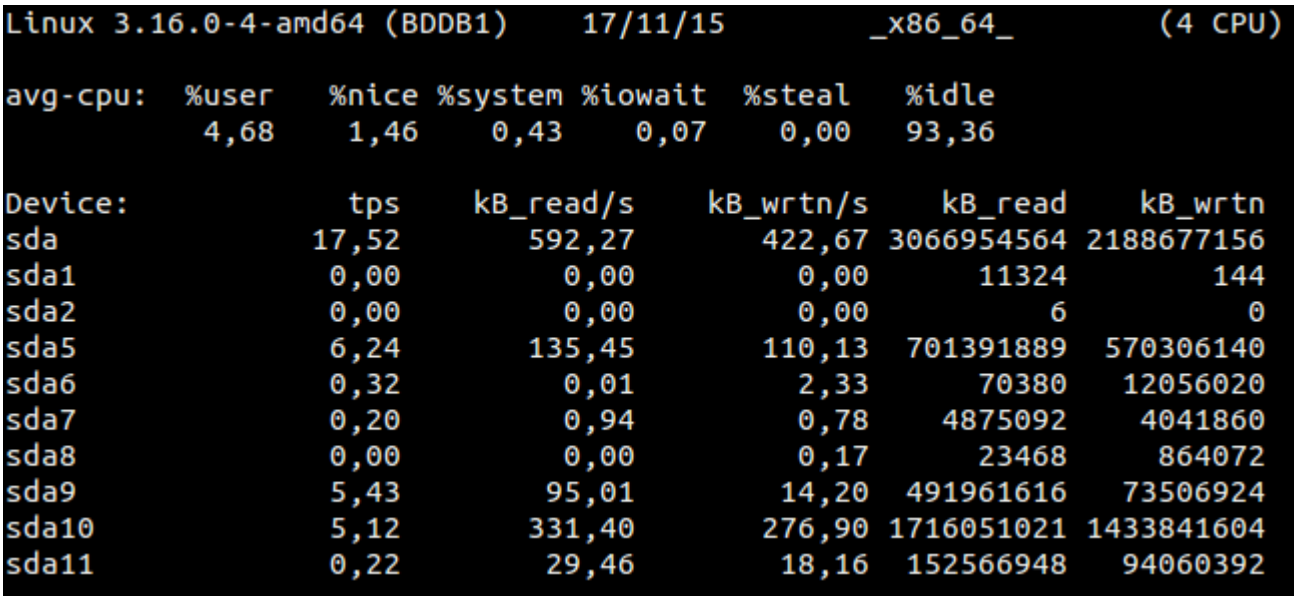



*Figura 40 Comando iostat -C*

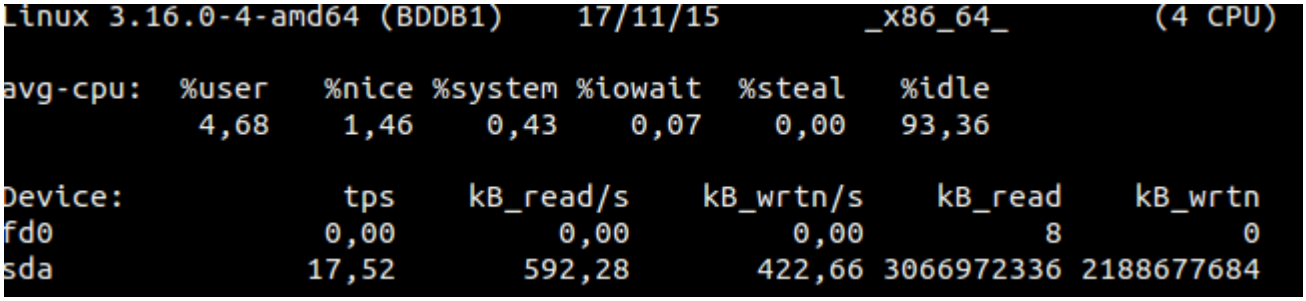
*Fuente: Elaboración propia*



```
Linux 3.16.0-4-amd64 (BDDB1)    17/11/15        _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           4,68    1,46    0,43    0,07    0,00   93,36

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              17,52       592,27       422,67 3066954564 2188677156
sda1              0,00         0,00         0,00      11324        144
sda2              0,00         0,00         0,00          6          0
sda5              6,24       135,45       110,13  701391889  570306140
sda6              0,32         0,01         2,33      70380   12056020
sda7              0,20         0,94         0,78    4875092    4041860
sda8              0,00         0,00         0,17      23468     864072
sda9              5,43        95,01        14,20  491961616   73506924
sda10             5,12       331,40       276,90 1716051021 1433841604
sda11             0,22        29,46        18,16  152566948   94060392
```

*Figura 41 Comando iostat -d*

*Fuente: Elaboración propia*



```
Linux 3.16.0-4-amd64 (BDDB1)    17/11/15        _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           4,68    1,46    0,43    0,07    0,00   93,36

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
fd0               0,00         0,00         0,00          8          0
sda              17,52       592,28       422,66 3066972336 2188677684
```

*Figura 42 Comando iostat -p sda*

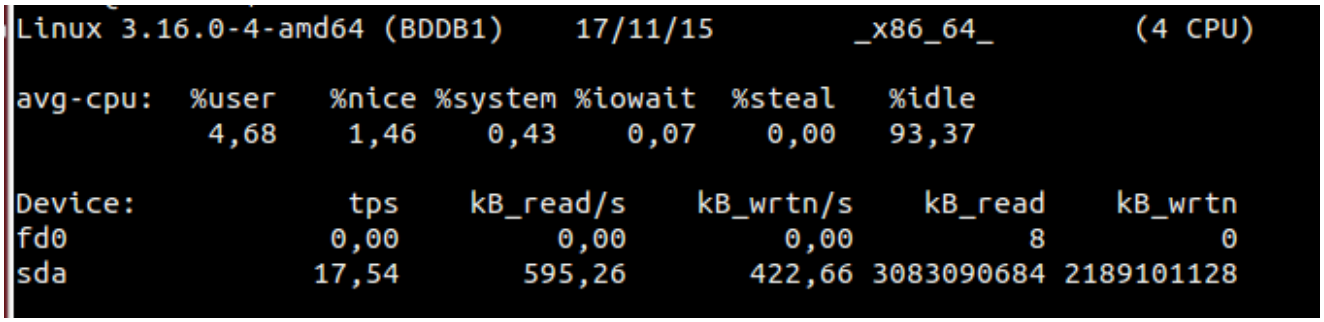



*Figura 43 Comando iostat -N*

*Fuente: Elaboración propia*

Análisis: Para la diferentes corridas de iostat sobre una prueba de carga continuada que involucra el movimiento de 8 millones de registros observamos un rendimiento bastante elevado del clúster; donde el uso del cpu (core de 4 cpu) en términos porcentuales es bastante bajo, las escrituras, lecturas por segundo y demás parámetros obtenidos por el comando denotan un buen rendimiento , destacando que el volumen de trabajo no genera mayores incidencias para el clúster en este escenario de baja demanda.

*Tabla 16 Prueba 2*

*Fuente: propia*

| | |
|---|---|
| Titulo: | Prueba 2 |
| Descripción: | Rendimiento obtenido con una prueba de carga sostenida, migrando 8 millones de registro del clúster Oracle (instancia 72) al clúster Cassandra (direcciones 192.16.11.120/121/122). |
| Escenario: | Baja demanda |
| Herramienta | hdparm |

Para la adecuada comprensión de la herramienta hdparm, se presenta una tabla explicativa de cada una de las opciones de configuración.

*Tabla 17 Comandos hdparm*

*Fuente: Elaboración propia*

| Comandos | Descripción |
|---|---|
| -t | Prueba de tiempos de lectura en el buffer |
| -T | Prueba de tiempos de lectura en la cache |
| -I | Devuelve información directamente desde la unidad (velocidad, tipo, modelo, serial, ...) |
| -d1 | Habilita el DMA (siempre y cuando no está ya activado) |
| for i in 1 2 3; do hdparm -tT /dev/sda; | done Realiza tres veces las pruebas de disco |

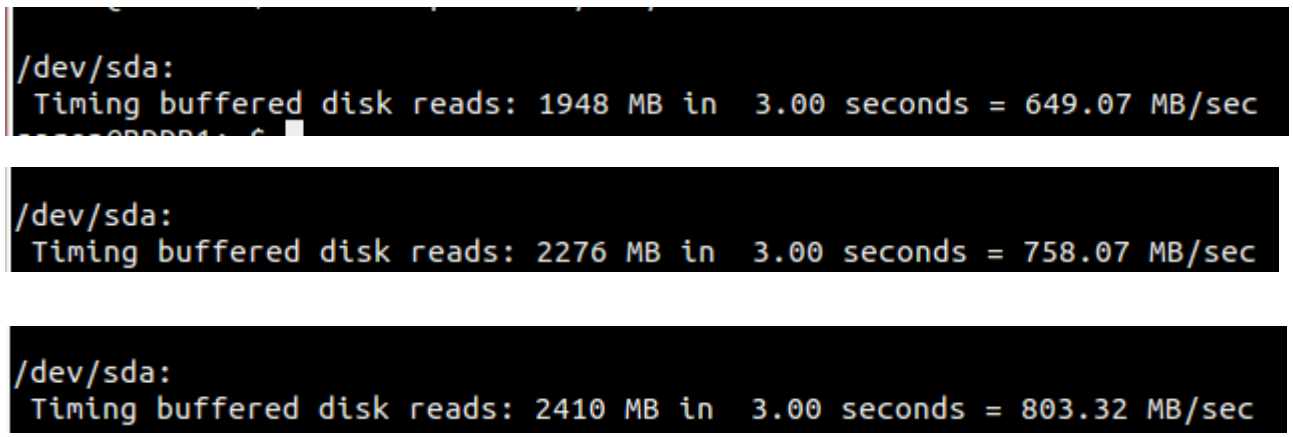
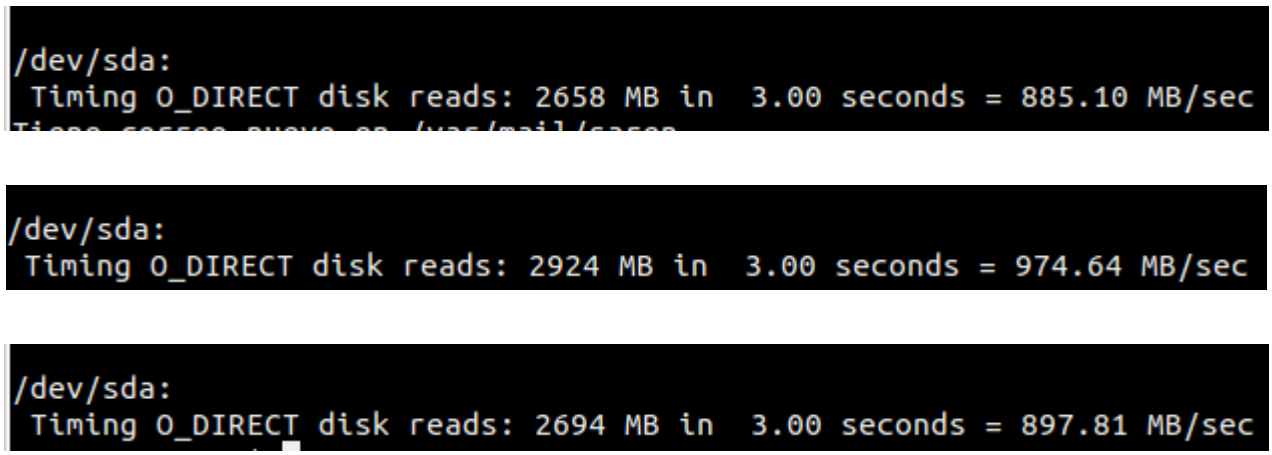



*Figura 44 Comando hdparm -t /dev/sda*

*Fuente: propia*

Análisis: Principalmente se destaca la velocidad de operación de los discos sometido al estrés generado durante la prueba de migración de datos, la cual genera un considerable número de operaciones de lectura y escritura, afectando la velocidad de operación de los discos.
No obstante, cuando se compara con la velocidad real de los discos determinados por (hdparm -t --direct /dev/sda), se observa un descenso en la

velocidad generado por la carga, sin embargo, la velocidad de operación se mantiene aceptable en comparación con la velocidad real.




Velocidad real de los discos en el escenario de baja demanda.

*Figura 45 Comando hdparm -t --direct /dev/sda*

*Fuente: propia*

| | |
|---|---|
| Titulo: | Prueba 3 |
| Descripción: | Rendimiento obtenido con una prueba de carga sostenida, migrando 60 millones de registro del clúster Oracle (instancia 72) al clúster Cassandra (direcciones 192.16.11.120/121/122). |
| Escenario: | Alta demanda |
| Herramienta | iostat |

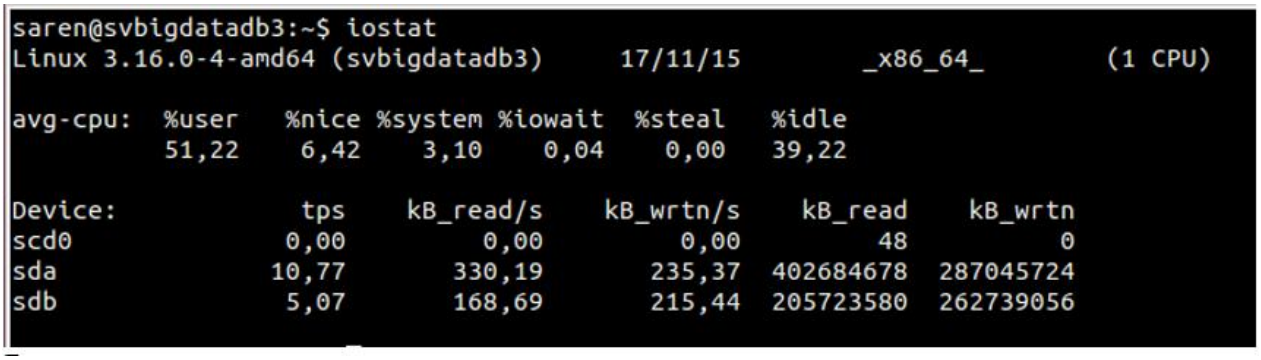



*Figura 46 Comando iostat*

*Fuente: propia*

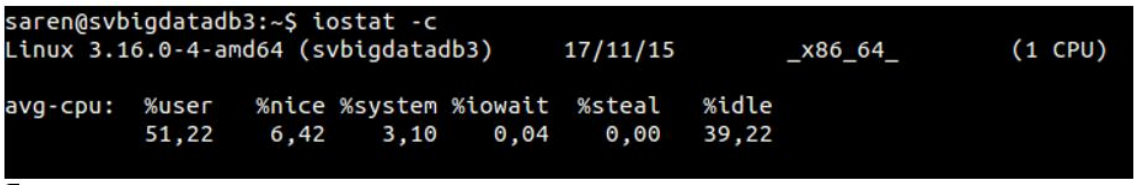



*Figura 47 Comando iostat -c*

*Fuente: propia*

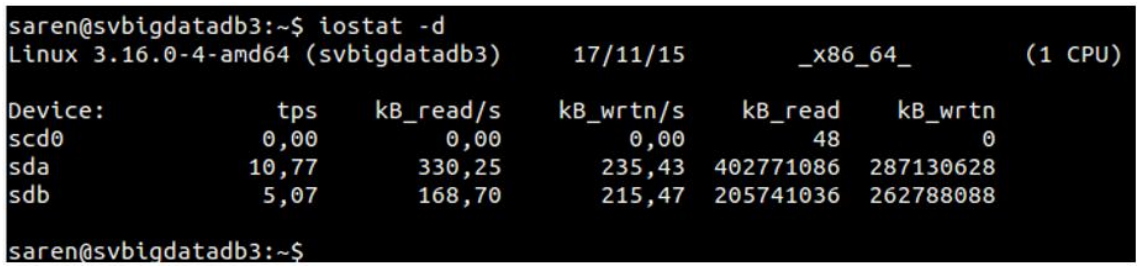



*Figura 48 Comando iostat -d*

*Fuente: propia*

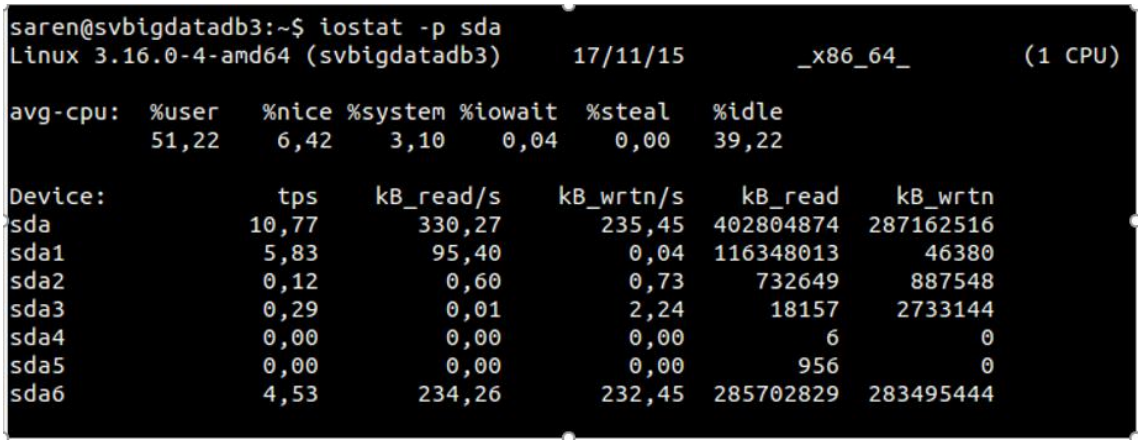



*Figura 49 Comando iostat -p sda*

*Fuente: propia*

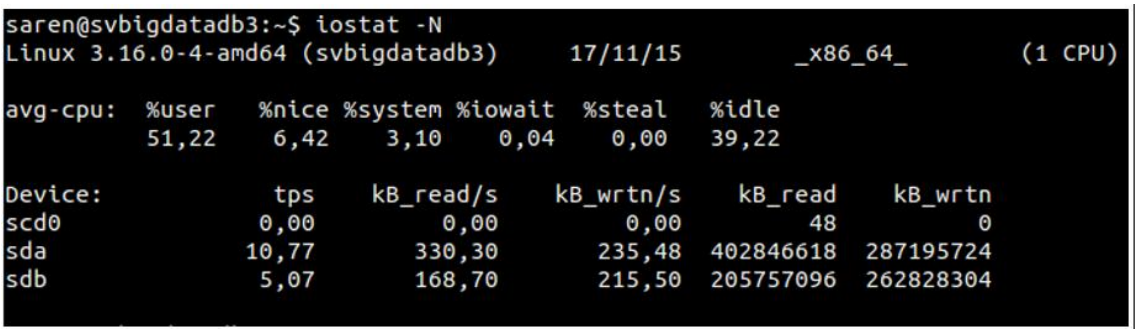



*Figura 50 Comando iostat -N*

*Fuente: propia*

Escenario de alta demanda, ambiente de producción clúster (192.16.11.120/122), sobre un proceso de migración que involucra una carga de 60 millones de registros con 2 ETL corriendo en simultaneo, observando principalmente el incremento del uso de CPU y el aumento de los Kb escritos y leídos por segundo en disco. No obstante, los valores como el nice, system y iowait se mantienen en rangos ideales.

*Tabla 18 Prueba 4*

*Fuente : propia*

| | |
|---|---|
| Titulo: | Prueba 4 |
| Descripción: | Rendimiento obtenido con una prueba de carga sostenida, migrando 60 millones de registro del clúster Oracle (instancia 72) al clúster Cassandra (direcciones 192.16.11.120/121/122). |
| Escenario: | Alta demanda |
| Herramienta | hdparm |

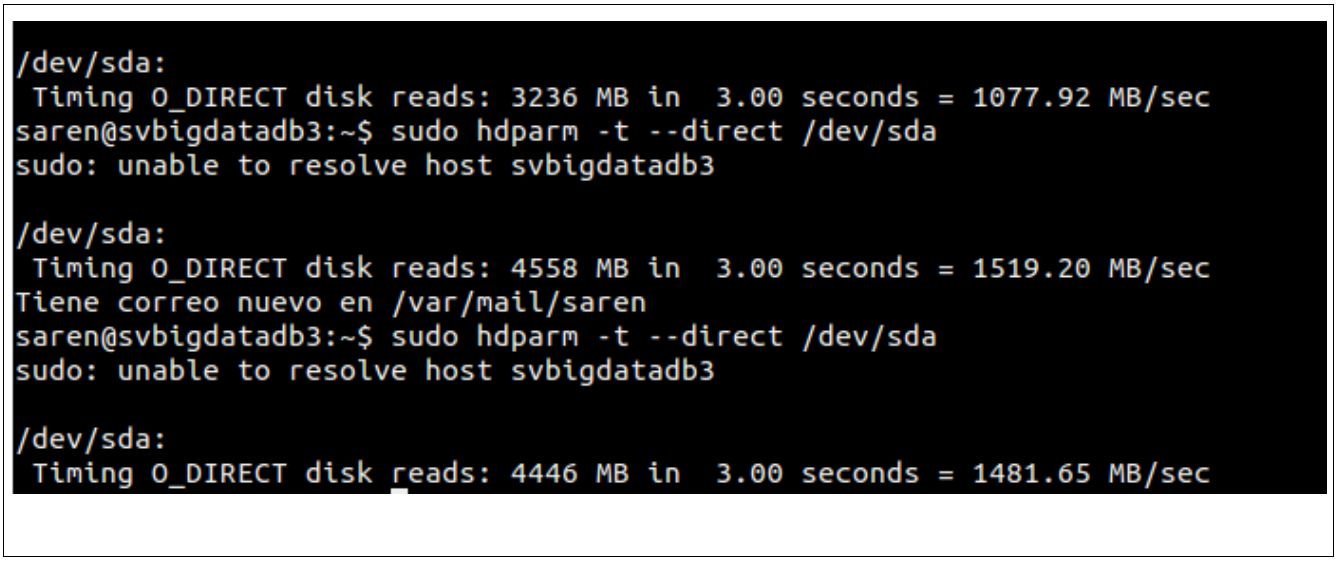



*Figura 51 Comando hdparm -t --direct /dev/sda*

En el escenario de alta demanda para el momento de ejecución de esta prueba el clúster de producción se encontraba ejecutando 2 transformaciones (ETL) que involucraban la migración de 60 millones de registros y más 150 transformaciones en espera para finalizar la carga inicial.
Sobre este escenario se observa una baja significativa en las velocidades de los discos, donde destaca una velocidad real cercana a un 1GB en virtud de las evidencias suministradas y la velocidad en operación sobre el escenario de alta demanda oscila entre **194 MB a 220 MB,** lo cual revela como el clúster está acercándose a su máxima capacidad instalada.

**Pruebas de rendimiento sobre Cassandra:**

La aplicación de las pruebas para carga, volumen y estrés, se ejecutarán completamente con el aplicativo cassandra-stress y los siguientes utilitarios.

- nodetool cfstats: muestra información estadística para un column family en específico.
- nodetool tpstats: provee estadísticas de uso de carácter general sobre el clúster
- nodetool cfhistograms: proporciona un conjunto de estadísticas que pueden ser graficadas.
- nodetool compactionstats: muestra información estadística sobre las operaciones de compactación ejecutadas sobre la base de datos.

Se inicia la sección con una explicación breve de la herramienta cassandra-stress y las estadísticas obtenidas ofrecidas por la herramienta.

**Cassandra-stress**: es una utilidad contenida en la distribución de cassandra, la cual facilita la ejecución de pruebas de carga, volumen y estrés en modo consola, ofreciendo una amplia información estadística sobre el comportamiento de la base de datos en general o en específico sobre un keyspace o estructura en particular.

A continuación, se aprecia un cuadro explicativo con las estadísticas presentadas por el comando.

*Tabla 19 Estadísticas Cassandra stress*

*Fuente: propia*

| ***Estadística*** | ***Descripción*** |
|---|---|
| total ops | Número total de operaciones ejecutadas durante la prueba. |
| op/s | Número de operaciones por segundo ejecutadas durante la prueba. |
| pk/s | Número de operaciones de partición ejecutadas durante la pruueba. |
| row/s | Número de operaciones ejecutadas por fila durante la prueba |
| mean | Promedio de latencia expresada en milisegundos durante la ejecución de la prueba. |
| med | Latencia media expresada en milisegundos. |
| .95 | 95% del tiempo de la latencia fue menor que el número que se muestra en la columna. |
| .99 | 99% del tiempo de la latencia fue menor que el número que se muestra en la columna. |
| .999 | 99.99% del tiempo de la latencia fue menor que el número que se muestra en la columna. |
| max | Máxima de latencia expresado en milisegundo durante la prueba |
| time | Tiempo total de la prueba. |

| | |
|---|---|
| stderr | Error estándar de la media. Es una medida de confianza en el número promedio de rendimiento; cuanto menor sea el número, más precisa será la medida de la prueba. |
| gc: # | Número de elementos en el garbage collections de java |
| max ms | Máximo del garbage collection expresado en milisegundos |
| sum ms | Total del garbage collection en milisegundos |
| sdv ms | Desviación estándar de la prueba en milisegundos |
| mb | Tamaño del en garbage collection MB |

Obtención de estadísticas de rendimiento sobre ambiente producción en escenario de alta demanda, con la herramienta nodetool cfhistograms:

./nodetool cfhistograms saren comun

./nodetool cfhistograms saren mercantil

Se calculan estadísticas para cada uno de los column families

```
saren@svbigdatadb3:/saren/bd/bin$ ./nodetool cfhistograms saren comun
saren/comun histograms
Percentile  SSTables     Write Latency      Read Latency    Partition Size        Cell Count
                              (micros)          (micros)           (bytes)
50%             0,00            924,00              0,00          43388628            545791
75%             0,00           1109,00              0,00          52066354            654949
95%             0,00           4768,00              0,00          74975550            943127
98%             0,00           8239,00              0,00          74975550            943127
99%             0,00           9887,00              0,00          74975550            943127
Min             0,00            150,00              0,00          20924301            263211
Max             0,00         315852,00              0,00       44285675122         464228842
```

*Figura 52 Estadísticas del columnfamily saren.comun en clúster producción*

```
saren@svbigdatadb3:/saren/bd/bin$ ./nodetool cfhistograms saren mercantil
saren/mercantil histograms
Percentile  SSTables     Write Latency      Read Latency    Partition Size        Cell Count
                              (micros)          (micros)           (bytes)
50%             0,00          1916,00              0,00            943127              9887
75%             0,00          2759,00              0,00           1955666             20501
95%             0,00         14237,00              0,00           4866323             51012
98%             0,00         17084,00              0,00           8409007             88148
99%             0,00         20501,00              0,00          12108970            126934
Min             0,00           180,00              0,00              2300                36
Max             0,00        654949,00              0,00         802187438           7007506
```

*Figura 53 Estadísticas del columnfamily saren.mercantil, cluster producción*

Análisis:
En primera instancia, es necesario destacar que los valores de Write Latency y Read Latency están expresados en micros segundos, el cual equivale a la millonésima parte de un segundo, $10^{-6}$s. Para expresarlo en milisegundo, puede emplearse la siguiente equivalencia **1 Microsegundo = 0.0010 Milisegundo.**

Observando las estadísticas suministradas referentes al column family saren.mercantil, se destaca primordialmente los tiempos mínimos y máximos en escritura (180,654949) microsegundos, convertidos en milisegundos Min: 0,18 miliseg y Max: 654,949 miliseg. Lo cual revela que en el peor escenario la escritura sobre el column familie mercantil no ha superado el medio segundo.
Para el otro column family, se evidencia sus tiempos mínimos y máximos a continuación:
Común: (Min: 0,15, Max: 315,852) en milisegundos en ambos casos claramente por debajo de un segundo.

Obtención de estadísticas de rendimiento sobre ambiente producción en escenario de alta demanda, con la herramienta cassandra-strees. Opciones disponibles para cassandra-stress:

*Tabla 20 Comandos Cassandra stress*

*Fuente: propia*

| ***Comando*** | ***Descripción*** |
|---|---|
| read | Múltiples lecturas concurrentes, el cluster debe leer primero antes de escribir. |
| write | Escrituras concurrentes |
| mixed | Intercalar comandos básicos con relación y distribución configurable |

| | |
|---|---|
| counter_write | Conteo de escrituras concurrentes ejecutadas |
| counter_read | Conteo de lecturas concurrentes ejecutadas. |
| user | Interleave user provided queries with configurable ratio and distribution. |
| help | Despliega las opciones de ayuda para el comando. |
| print | Inspeccionar la salida del comando |

```
Pruebas de lectura simulando un número de 200 mil usuarios.
cassandra-stress read n=200000 > salida.out
cassandra-stress read duration=3m > salida1.out
cassandra-stress read n=200000 no-warmup> salida2.out
```

Por la cantidad de información suministrada por la herramienta *cassandra-stress*, se mostrará la salida para la ejecución del comando *cassandra-stress read n=200000 > salida.out*

En esta prueba se están realizando 200 mil lecturas comenzando con 4 hilos concurrentes hasta llegar a 406 hilos, obteniendo los siguientes resultados al final de la prueba.

```
Warming up READ with 50000 iterations...
INFO  17:24:18 Using data-center name 'datacenter1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the correct datacenter name wit
 DCAwareRoundRobinPolicy constructor)
Connected to cluster: Test Cluster
Datatacenter: datacenter1; Host: localhost/127.0.0.1; Rack: rack1
INFO  17:24:18 New Cassandra host localhost/127.0.0.1:9042 added
Sleeping 2s...
Running with 4 threadCount
Running READ with 4 threads for 200000 iteration
total ops , adj row/s,    op/s,    pk/s,   row/s,    mean,     med,     .95,     .99,    .999,     max,   time,   stderr,  gc: #,  max ms,  sum ms,
dv ms,      mb
5530      ,      5530,    5530,    5530,    5530,    0,7,     0,6,     1,0,     1,5,     8,2,     9,6,    1,0,  0,00000,      0,       0,       0,
    0,       0
11180     ,      5626,    5626,    5626,    5626,    0,7,     0,6,     1,0,     1,5,     8,9,    11,9,    2,0,  0,00000,      0,       0,       0,
    0,       0
16883     ,      5816,    5679,    5679,    5679,    0,7,     0,6,     1,0,     1,4,    16,9,    24,4,    3,0,  0,00614,      1,      23,      23,
    0,     204
22714     ,      5799,    5799,    5799,    5799,    0,7,     0,6,     1,0,     1,5,     2,4,     4,0,    4,0,  0,01214,      0,       0,       0,
    0,       0
28607     ,      5861,    5861,    5861,    5861,    0,6,     0,6,     1,0,     1,4,     4,3,     5,9,    5,0,  0,01053,      0,       0,       0,
    0,       0
34543     ,      6067,    5905,    5905,    5905,    0,6,     0,6,     0,9,     1,4,     3,9,    27,5,    6,0,  0,00989,      1,      22,      22,
    0,     203
40647     ,      6069,    6069,    6069,    6069,    0,6,     0,6,     0,9,     1,4,     2,2,    11,4,    7,0,  0,01212,      0,       0,       0,
    0,       0
46589     ,      5906,    5906,    5906,    5906,    0,6,     0,6,     1,0,     1,5,     3,8,     9,0,    8,0,  0,01219,      0,       0,       0,
    0,       0
52597     ,      6108,    5973,    5973,    5973,    0,6,     0,6,     0,9,     1,3,     3,4,    23,5,    9,0,  0,01078,      1,      22,      22,
    0,     204
58540     ,      5897,    5897,    5897,    5897,    0,6,     0,6,     1,0,     1,5,     2,7,     5,7,   10,0,  0,01071,      0,       0,       0,
    0,       0
64632     ,      6047,    6047,    6047,    6047,    0,6,     0,6,     1,0,     1,6,     3,4,     7,8,   11,1,  0,00965,      0,       0,       0,
    0,       0
70153     ,      5480,    5480,    5480,    5480,    0,7,     0,6,     1,3,     2,8,     5,4,     6,8,   12,1,  0,00913,      0,       0,       0,
    0,       0
```

*Figura 54 Salida.out*

*Fuente: propia*

Los totales de la prueba se muestran a continuación:

Resultado final de la prueba expresada en microsegundos, observando bajos niveles de latencia en el clúster y un buen manejo del *garbage collector*.

```
Results:
op rate                   : 18190
partition rate            : 18190
row rate                  : 18190
latency mean              : 33,9
latency median            : 28,5
latency 95th percentile   : 75,9
latency 99th percentile   : 107,5
latency 99.9th percentile : 145,2
latency max               : 226,8
total gc count            : 6
total gc mb               : 1229
total gc time (s)         : 0
avg gc time(ms)           : 18
stdev gc time(ms)         : 2
Total operation time      : 00:00:10
Improvement over 406 threadCount: 1%
```

*Figura 55 Resultados finales de la prueba*

*Fuente: propia*

# ANEXO 9: Casos de uso Nivel 1


*Figura 56 Casos de uso nivel 1*

*Fuente: propia*

*Tabla 21 Funcionalidades casos de uso nivel 1*

*Fuente: propia*

| Caso de uso | Conectarse a Repositorio |
|---|---|
| Actor | PDI |
| Flujo Básico | Para ejecutar una transformación de carga de datos, es necesario que antes se haya conectado al repositorio de las transformaciones, las cuales están almacenadas en |

| | |
|---|---|
| | alguna base de datos, como puede ser postgreSQL |
| Pre-Condición | Las transformaciones fueron almacenadas en una base de datos relacional. |
| | |
| Caso de uso | Conectarse a los Nodos del Cluster |
| Actor | Cassandra |
| Flujo Básico | Cada nodo del cluster de Cassandra se comunica entre si utilizando el protocolo Gossip, en el cual un nodo transmite la información a los nodos cercanos. |
| Pre-Condición | Los nodos fueron configurados previamente y se asignó un nodo como el nodo semilla. |
| | |
| Caso de uso | Ejecucion MapReduce |
| Actor | Hadoop |
| Flujo Básico | Una vez que se solicita la ejecución de un trabajo MapReduce, el jobtracker se encarga de asignar cada trabajo de mapeo a los tasktracker mas cercanos a los datanodes, luego de que los tasktrackers finalicen su trabajo le notifican al jobtracker, el cual se encargara de asignar la tarea de reducción a un tasktracker en específico y guardar el resultado en la ruta esperada. |
| Pre-Condición | Se solicita la ejecución de un trabajo MapReduce y deben estar activos los procesos principales como el namenode, el datanode, el resource manager, el jobtracker, tasktracker. |
| | |
| Caso de uso | Almacenamiento en HDFS |
| Actor | Hadoop |
| Flujo Básico | Un proceso o usuario solicita almacenar en HDFS bien sea mediante un comando o mediante una configuración previa. |
| Pre-Condición | Debe estar ejecutándose los procesos principales de Hadoop (namenode, datanode, resource manager) |

| | |
|---|---|
| Caso de uso | Conectarse a HDFS |
| Actor | Hive |
| Flujo Básico | Para realizar las consultas en Hive, primero debe realizarse una conexión al HDFS la cual fue configurada previamente en el archivo hive-site.xml. |
| Pre-Condición | Preconfiguracion de hive-site.xml para almacenar en HDFS. |
| | |
| Caso de uso | Conectarse a las herramientas |
| Actor | HUE |
| Flujo Básico | Mediante HUE se hace una integración con las herramientas principales de generación como son Hadoop, Hive y Solr, las cuales fueron configuradas previamente. |
| Pre-Condición | Se configuro el archivo hue.ini para indicar las herramientas a conectarse y los puertos mediante el cual se harán las conexiones |
| | |
| Caso de uso | Generacion de reportes con Solr |
| Actor | SAREN |
| Flujo Básico | El usuario de SAREN al ingresar a la interfaz web, puede escoger la opción de generar reportes y dashboards en Solr para obtener información de los Registros Mercantiles |
| Pre-Condición | El usuario ingreso al sistema mediante autenticación y las herramientas están funcionando correctamente. |
| | |
| Caso de uso | Consultas con Hive |
| Actor | SAREN |
| Flujo Básico | Una vez que el usuario ingreso a la interfaz de HUE y escoge la opción de hacer consultas con Hive, puede realizar cualquier consulta en HQL mediante un panel de texto de consultas. |
| Pre-Condición | El usuario ingreso previamente al sistema y el Hiveserver2 está corriendo. |

| | |
|---|---|
| Caso de uso | Administrar HDFS |
| Actor | SAREN |
| Flujo Básico | El usuario puede observar y administrar el HDFS mediante una interfaz que permite hacer búsquedas de archivos. |
| Pre-Condición | Los procesos de Hadoop deben estar levantados. |

## ANEXO 10: Cronología Big Data

- El efecto en las bibliotecas (1940): Las bibliotecas, fuente original de la organización y el almacenamiento de datos, tuvieron que adaptar sus métodos de almacenamiento para responder al rápido aumento de la demanda de nuevas publicaciones e investigación.
- La explosión de la información (1941): Los académicos comenzaron a denominar a esta increíble expansión de la información como la "explosión de la información". Tras aparecer por primera vez en el periódico *Lawton Constitution* en el año 1941, el término se desarrolló en un artículo del *New Statesman* en marzo del año 1964, en el que se hacía referencia a la dificultad que suponía gestionar los volúmenes de información disponibles.
- El primer aviso del problema del almacenamiento y recuperación de datos (1944): La primera señal de aviso sobre el crecimiento del conocimiento como problema inminente a la hora de almacenar y recuperar los datos tuvo lugar en 1944, cuando Fremont Rider, bibliotecario de la Universidad Wesleyana, calculó que las bibliotecas de las universidades de EE.UU. duplicaban su tamaño cada dieciséis años. Rider calculó que, si la tasa de crecimiento se mantuviera, la biblioteca de Yale tendría en el año 2040 "aproximadamente 200.000.000 de volúmenes, que ocuparían 9656 km de estanterías... [por lo que se necesitaría] un personal de catalogado de más de seis mil personas"
- La teoría de la información de Shannon (1948): Claude Shannon publicó la Teoría matemática de la comunicación, en la que se estableció un marco de trabajo para determinar los requisitos de datos mínimos para transmitir la información a través de canales afectados por ruido (imperfectos). Fue un trabajo histórico que ha hecho posible gran parte de la infraestructura actual. Sin su teoría, el volumen de los datos sería mucho mayor que el actual. Utilizó como referencia *"Certain Factors Affecting Telegraph Speed"*, una obra de Nyquist cuyo título nos suena muy lejano, pero que nos permitió muestrear señales analógicas y

representarlas digitalmente, que no es sino la base del procesamiento de datos moderno.

- Memoria virtual (1956): El concepto de memoria virtual fue desarrollado por el físico alemán Fritz-Rudolf Güntsch, como una idea que trataba el almacenamiento finito como infinito. El almacenamiento, administrado mediante hardware integrado y software para ocultar los detalles al usuario, nos permitió procesar los datos sin las limitaciones de memoria de hardware que anteriormente provocaban la partición del problema (haciendo de la solución un reflejo de la arquitectura de hardware, una medida ilógica de base).
- El conocimiento científico se amplía (1961): El científico de la información Derek Price generalizó las conclusiones de Rider para incluir casi todos los tipos de conocimiento científico. La revolución científica, tal como él la llamó, era la responsable de la comunicación rápida de ideas nuevas como información científica. Este rápido crecimiento se materializaba en la duplicación cada 15 años de los registros nuevos creados.
- Los pioneros en el reconocimiento de voz: Los científicos han trabajado en el reconocimiento de voz casi desde que empezaron a fabricar ordenadores. En el año 1962, William C. Dersch de IBM desveló la máquina Shoebox en la Feria Mundial. Fue la primera máquina capaz de entender 16 palabras y diez dígitos en inglés hablado mediante el uso de los datos disponibles en ese momento, y era capaz de procesarlos correctamente. Sin embargo, hasta transformar esta innovación en el reconocimiento de voz en productos con una utilidad comercial real, aún quedaba mucho camino por delante. Este camino exigiría avances importantes en la potencia de procesamiento y la reducción del coste de la tecnología informática. La existencia de un volumen de datos mayor también ayudaría a entrenar los sistemas de reconocimiento de voz.
- En la búsqueda de una solución organizativa (1963): A principios de la década de 1960, Price observó que la enorme mayoría de investigación científica suponía un esfuerzo abrumador para los humanos. Los

resúmenes documentales, creados a finales de la década de 1800 como forma de gestionar los conocimientos, cada vez de mayor volumen, crecían también con la misma progresión (multiplicándose por un factor de diez cada cincuenta años), y ya habían alcanzado una magnitud preocupante. Habían dejado de ser una solución de almacenamiento o de organización de la información.

- Entran en escena los sistemas de computación centralizados (1966): La información no solo se encontraba en pleno auge en el sector científico, también lo estaba en el sector de los negocios. Debido a la influencia que tuvo la información en la década de 1960, la mayoría de organizaciones empezaron a diseñar, desarrollar e implementar sistemas informáticos que les permitían automatizar los sistemas de inventario.
- Base de datos relacional (1970): En el año 1970, Edgar F. Codd, un matemático formado en Oxford que trabajaba en IBM Research Lab, publicó un artículo en el que se explicaba la forma en la que podía acceder a la información almacenada en bases de datos de gran tamaño sin saber cómo estaba estructurada la información o dónde residía dentro de la base de datos. Hasta ese momento, para recuperar la información se necesitaban conocimientos informáticos relativamente sofisticados, e incluso para los servicios de especialistas, por lo que se convertía en una tarea ardua que exigía tiempo y recursos económicos. Hoy en día, la mayoría de transacciones de datos rutinarias (acceder a cuentas bancarias, utilizar tarjetas de crédito, comerciar con acciones, realizar reservas de viaje, realizar compras a través de Internet) utilizan estructuras basadas en la teoría de la base de datos relacional.
- El crecimiento de la comunicación bidireccional (1975): El Censo del Flujo de la Información, realizado por el Ministerio de Correos y Telecomunicaciones de Japón, comenzó a realizar un control del volumen de información que circulaba por el país en 1975. Utilizando como unidad de medición el número de palabras utilizadas a través de todos los medios de comunicación, el estudio pudo comprobar que el

suministro de información superaba considerablemente al volumen de información consumida, y que la demanda de comunicación unidireccional se había estancado. Ahora, la tendencia era el aumento de la demanda de comunicación bidireccional, más personalizada, que respondía a las necesidades de las personas.

- Sistemas de Planificación de necesidades de material (1976): A mediados de la década de 1970, los sistemas de Planificación de necesidades de material (MRP) se diseñaron como herramienta que ayudaba a las empresas de fabricación a organizar y planificar su información. A esas alturas, la popularidad de los PC en las empresas estaba en auge. Esta transformación marcó un cambio de tendencia hacia los procesos de negocio y las funcionalidades de contabilidad, y en este ámbito se fundaron empresas como *Oracle*, *JD Edwards y SAP*. Fue *Oracle* la que presentó y comercializó el Lenguaje de consulta estructurado o *Structure Query Language* (SQL) original.
- La ley de los datos de Parkinson (1980): A medida que aumentaba la velocidad con la que se creaba información, las opciones de almacenamiento y organización de datos eran cada vez menores. ¿En su charla *«Where Do We Go From Here?»*, I.A. Tjomsland afirmó que «aquellos que trabajan en dispositivos de almacenamiento descubrieron hace mucho tiempo que la primera ley de Parkinson puede parafrasearse para describir nuestro sector: “los datos se expanden para llenar el espacio disponible”. Desde mi punto de vista, las grandes cantidades de datos se guardan porque los usuarios no tienen forma de identificar los datos obsoletos; las penalizaciones derivadas de almacenar datos obsoletos tienen una importancia inferior a las que conlleva eliminar datos potencialmente útiles».
- El crecimiento de la información y el sector de la comunicación (1983): Los avances tecnológicos permitieron a todos los sectores beneficiarse de nuevas formas de organizar, almacenar y generar datos. Las empresas estaban empezando a usar los datos para tomar mejores decisiones de negocio. En el artículo *Tracking the Flow of Information*,

publicado en la revista *Science*, el autor Ithiel de Sola Pool analizó el crecimiento del volumen de información de 17 importantes medios de comunicación desde el año 1960 hasta 1977. Atribuye el enorme crecimiento de la información a la expansión del sector de las comunicaciones.

- Sistemas de Planificación de recursos de fabricación (1985): Tras el auge de los sistemas de MRP, se introdujo la Planificación de recursos de fabricación (MRP II) en la década de 1980, con un énfasis en la optimización de los procesos de fabricación mediante la sincronización de materiales con las necesidades de producción. MRP II incluía áreas tales como la gestión del área de producción y la distribución, la gestión de proyectos, las finanzas, los recursos humanos y la ingeniería. No fue hasta mucho después de adoptar esta tecnología cuando otros sectores (p. ej. agencias gubernamentales y organizaciones del sector servicios) comenzaron a tener en cuenta, y posteriormente adoptar, la tecnología ERP.
- La necesidad de datos precisos (1985): En el año 1985, Barry Devlin y Paul Murphy definieron una arquitectura para los informes y análisis de negocio en *IBM* (*Devlin & Murphy, IBM Systems Journal* 1988) que se convirtió en la base del almacenamiento de datos. En el centro neurálgico de dicha arquitectura, y en el almacenamiento de datos en general, se encuentra la necesidad de almacenamiento homogéneo y de alta calidad de datos históricamente completos y exactos.
- Desde las tablillas de barro hasta la memoria de semiconductores: ¿En su artículo *«Can users really absorb data at today's rates? Tomorrow's?»,* Hal Becker mencionaba que «la densidad de recodificación lograda por Gutenberg fue aproximadamente de 500 símbolos (caracteres) por pulgada cúbica; 500 veces la densidad de las tablillas de barro [sumerias del año 4000 antes de cristo]. En el año 2000, la memoria de acceso aleatorio de los semiconductores será capaz de almacenar 1,25×10^11 *bytes* por pulgada cúbica».

- La superficie de los nuevos sistemas de *software* (1988): A finales de la década de los 80 y principios de los 90, fuimos testigos del aumento de los sistemas de Planificación de recursos empresariales (ERP), ya que pasaron a ser más sofisticados y ofrecían la posibilidad de coordinarse e integrarse entre todos los departamentos de las empresas. Las bases tecnológicas de los sistemas de MRP, MRP II y ERP comenzaron a integrar áreas de empresas entre las que se incluían la producción, la distribución, la contabilidad, las finanzas, los recursos humanos, la gestión de proyectos, la gestión del inventario, el servicio y el mantenimiento, y el transporte, y ofrecer así accesibilidad, visibilidad y homogeneidad en la totalidad de la empresa.
- Inteligencia empresarial (1989): En 1989, Howard Dresner amplió el popular término genérico *«Business Intelligence (BI)»* o Inteligencia empresarial, inicialmente acuñado por Hans Peter Luhn en el año 1958. Dresner lo definió como los «conceptos y métodos que mejoran la toma de decisiones de negocio mediante el uso de sistemas de apoyo basados en datos reales». Poco tiempo después, y como respuesta a la necesidad de una mejor BI, pudimos ver el auge de empresas como *Business Objects, Actuate, Crystal Reports* y *MicroStrategy*, que ofrecían informes y análisis de los datos de las empresas.
- El primer informe de base de datos (1992): En 1992, *Crystal Reports* creó el primer informe de base de datos sencillo con *Windows*. Estos informes permitían a las empresas crear un informe sencillo a partir de diversos orígenes de datos con escasa programación de código. De esta forma, se redujo la presión existente sobre el panorama saturado de datos, y se permitió que las empresas emplearan la inteligencia empresarial de un modo asequible.
- Explosion de la *World Wide Web* (1995): En la década de 1990 se produjo un crecimiento tecnológico explosivo, y los datos de la Inteligencia empresarial comenzaron a apilarse en forma de documentos de *Microsoft Excel*.

- El espectacular crecimiento de la potencia informática e internet (1996): El aumento desmesurado del volumen de datos supuso otros problemas para los proveedores de sistemas ERP. La necesidad de tener que diseñar de nuevo los productos ERP, y que incluía romper los límites de titularidad y de personalización, obligó a los proveedores a adoptar de forma gradual un método de negocio colaborativo, en lugar de la intranet.
- Inteligencia empresarial 2.0 (1996): La influencia de la información trajo consigo un nuevo problema en la gestión de los datos, además de un aumento del coste que suponía publicarla y almacenarla. Como los datos resultaban más difíciles de mantener, para poder ofrecer más funcionalidades, el almacenamiento digital empezó a resultar más rentable que el papel para almacenar los datos, y comenzaron a emerger las plataformas de BI. *R.J.T. Morris y B.J. Truskowski* analizaron el almacenamiento de datos en su artículo *The Evolution of Storage Systems*, publicado en IBM *Systems Journal.*
- El problema del *Big Data* (1997): El término «Big Data» se empleó por primera vez en un artículo de los investigadores de la *NASA* Michael Cox y David Ellsworth. Ambos afirmaron que el ritmo de crecimiento de los datos empezaba a ser un problema para los sistemas informáticos actuales. Esto se denominó el «problema del *Big Data*».
- El futuro del almacenamiento de datos (1997): Michael Lesk publicó How much information is there in the world?. Su conclusión fue que «Puede que la cantidad de información ascienda a varios miles de petabytes, y la producción de cinta y disco alcanzará ese nivel en el año 2000. Esto significa que, en unos años, (a) podremos guardarlo todo, no será necesario eliminar información, y que (b) la mayoría de la información jamás será consultada por un ser humano».
- El problema de la inteligencia empresarial (1998): A finales de los 90, muchas empresas creían que sus sistemas de extracción de datos no funcionaban. Los trabajadores eran incapaces de encontrar respuestas y de acceder a los datos que necesitaban de las búsquedas. Los

departamentos informáticos eran responsables del 80 % del acceso a BI. Cada vez que los empleados necesitaban acceso, tenían que llamar al departamento informático, ya que acceder a la información no resultaba tan fácil.

- Internet de las cosas (1999): El término "Internet de las cosas" o IoT, por sus siglas en inglés, fue acuñado por el emprendedor británico Kevin Ashton, cofundador de*l Auto-ID Center* del *MIT*, durante una presentación que enlazaba la idea de identificación por radiofrecuencia (RFID) en la cadena de suministro con el mundo de Internet. "Si tuviéramos equipos que supieran todo lo que hay que saber acerca de las cosas, a partir de datos que, recopilados sin nuestra ayuda, seríamos capaces de monitorizar y contar todo, y reducir así considerablemente los costes, los desperdicios y las pérdidas."
- Se cuantifica la información (1999): Peter Lyman y Hal R. Varian de *UC Berkeley* publicaron el primer estudio que cuantificaba, en términos de almacenamiento informático, la cantidad total de información nueva y original creada en el mundo al año. El estudio, titulado *How Much Information?*, se completó en 1999, un año en el que el mundo produjo unos 1,5 exabytes de información.
- El análisis predictivo cambia el perfil del negocio (1999): *ComputerWeekly* cuenta con un artículo destacado en el que se explica la forma en la que elegir e instalar la solución ERP adecuada y utilizar pronósticos de análisis predictivo cambia el método de trabajo de todo tipo de organizaciones.
- Software como servicio (2001): Las siglas SaaS aparecen por primera vez en un artículo de la división de comercio electrónico de *Software & Information Industry* (SIIA).
- Las tres V (2001): Doug Laney, analista de *Gartner*, publicó un artículo titulado *3D Data Management: Controlling Data Volume, Velocity, and Variety*. A día de hoy, las tres V siguen siendo las dimensiones comúnmente aceptadas del *Big Data*.

- Sistemas ERP ampliados (2002): Durante la década de los 90, los proveedores de sistemas ERP añadieron más módulos y funciones como complementos de los módulos básicos, con lo que surgieron los sistemas ERP extendidos o ampliados. El número de opciones de *software* y *hardware* aumentó exponencialmente y, a principios de la década del 2000, comenzaron a surgir importantes empresas de *software*. *Oracle* y *SAP* fueron las principales empresas de *software* ERP que sobrevivieron a este auge.
- Servicios web y ERP (Junio 2002): Los principales proveedores de sistemas ERP, como *SAP*, *PeopleSoft*, *Oracle* y *JD Edwards*, comenzaron a centrarse de una forma agresiva en el uso de servicios web para enlazar sus propios conjuntos de aplicaciones, y en facilitar a los clientes la creación de aplicaciones nuevas a partir de datos de varias aplicaciones utilizando XML.
- El enfoque en la usabilidad del usuario final (Marzo 2005): Las empresas de *SaaS* entraron en escena para ofrecer una alternativa a *Oracle* y *SAP* más centrada en la usabilidad del usuario final. Una de las primeras fusiones de empresas fue la que dio origen a *Workday, Inc.*, fundada en marzo de 2005, como alternativa a *Oracle* y *SAP*, más económica y utilizable. El software de *Workday Inc.* es más intuitivo para el usuario final y funciona de la misma forma que la gente, «de forma colaborativa, sobre la marcha y en tiempo real».
- La gestión de la base de datos, el centro del universo (Septiembre 2005): Tim O'Reilly publicó *What is Web 2.0?*, donde afirma que «los datos son el próximo *Intel Inside*». En el artículo, O'Reilly afirma lo siguiente: «Como Hal Varian apuntó en una conversación personal el año pasado, "el *SQL* es el nuevo *HTML*". La gestión de la base de datos es una competencia básica de las empresas *Web 2.0*, tanto que a veces denominamos a estas aplicaciones "infoware" en lugar de simplemente software».
- Una solución de código abierto para la explosión del *Big Data* (2006): *Hadoop* se creó en el año 2006 a raíz de la necesidad de sistemas

nuevos para gestionar la explosión de datos de la web. De descarga gratuita, y libre para potenciarlo y mejorarlo, *Hadoop* es un método de código abierto para almacenar y procesar los datos que «permite el procesamiento en paralelo distribuido de enormes cantidades de datos en servidores estándar del sector, económicos, que almacenan y procesan los datos, y que pueden escalarse sin límite».

- El primer estudio que calcula y prevé la cantidad de crecimiento de la información (Marzo 2007): Los investigadores de *International Data Corporation* publicaron un artículo titulado *The Expanding Digital Universe: A Forecast of Worldwide Information Growth through 2010*, en el que se calcula y pronostica la cantidad de datos digitales que se crearán y reproducirán cada año. En el artículo se calcula que, solo en el año 2006, se crearon en todo el mundo 161 exabytes de datos, y prevé que, en los próximos cuatro años, la información creada aumentará hasta multiplicarse por seis (hasta los 988 exabytes). En otras palabras, predicen que la información se duplicará cada 18 meses durante los próximos cuatro años. Si consultamos los informes de los años 2010 y 2012, la cantidad de datos digitales creados cada año superó los pronósticos iniciales (1227 en 2010 y 2837 exabytes en 2012).
- La explosión de datos continúa (2008): Bret Swanson y George Gilder proyectaron que el tráfico IP estadounidense podría alcanzar el *zettabyte* en el año 2015, y que la Internet estadounidense del 2015 será, como mínimo, 50 veces más grande que lo era en el 2006.
- El aluvión de datos hace que el método científico quede obsoleto (Junio 2008): El término *Big Data* comenzó a utilizarse con cada vez más frecuencia en los círculos tecnológicos. La revista *Wired* publicó un artículo en el que se presentaba el impacto positivo y negativo del aluvión de datos reciente. En este artículo, *Wired* anunció que este era el «principio de la era del *petabyte*». A pesar de que era una buena hipótesis, la clasificación de "*petabyte"* era demasiado técnica para el público en general. Inevitablemente, un *petabyte*, que equivale a

1.000.000.000.000.000 *bytes* de datos, dará paso dentro de poco a *bytes* de datos todavía mayores: *exabytes, zettabytes* y *yottabytes*.

- *SAP* desvela su estrategia de *SaaS* (Noviembre 2008): SAP realizó un movimiento estratégico de cara al mercado de *SaaS* mediante el desarrollo de una estrategia de software como servicio destinada a las grandes empresas. Como parte del mismo, se contrató a John Wookey (antiguo empleado de *Oracle*) en noviembre de 2008 como nuevo jefe de aplicaciones de software a la carta para grandes empresas. Tras trasladar su propuesta a la junta en el mes de enero, desarrollaron un plan para lanzar las nuevas ofertas de productos de *SaaS* en series de aplicaciones de software concretas para cada función. Estas aplicaciones, disponibles por suscripción, se conectan con los sistemas *SAP Business Suite in situ* que *SAP* alojará en régimen de tenencia múltiple.
- Avances revolucionarios (Diciembre 2008): Un grupo de investigadores científicos en el ámbito de la informática publicó el artículo titulado *Big Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science, and Society*. En el artículo se afirma lo siguiente: «De la misma forma que los motores de búsqueda han cambiado la forma de acceder a la información, otras formas de informática de *Big Data* pueden transformar y transformarán las actividades de empresas, investigadores científicos, médicos y las operaciones de defensa e inteligencia de nuestra nación…. Probablemente, la informática de *Big Data* sea la mayor innovación informática de la última década. A día de hoy, tan solo hemos visto el potencial que tiene para recopilar, organizar y procesar los datos en todos los aspectos de nuestras vidas. Si el gobierno federal efectuara una modesta inversión, su desarrollo e implantación podrían acelerarse enormemente». Este apoyo hizo que el *Big Data* finalmente lograra la credibilidad intelectual que necesitaba.
- La inteligencia empresarial pasa a ser una prioridad (Enero 2009): En el año 2009, la Inteligencia empresarial pasó a ser una de las principales prioridades para los directores de tecnologías de la información.

- *Linked Data* (1 Febrero 2009): Tim Berners-Lee, director del *World Wide Web Consortium* (W3C) e inventor del *World Wide Web*, fue el primero en usar el término "*linked data*" (datos enlazados) durante una presentación sobre el tema en el congreso *TED* de 2009. *Linked Data* describe un método de publicación de datos estructurados, basado en protocolos web estándar, para que puedan ser interconectados, leídos automáticamente por ordenadores y enlazados desde otros conjuntos de datos externos.
- Análisis ERP (Mayo 2009): *Gartner* predijo que los datos empresariales crecerían un 650% durante los próximos cinco años. Estos datos representan el conglomerado de todos los datos operativos de ERP internos, además de los datos externos que tienen interconexión con las operaciones de la empresa; pensemos más allá de los datos de proveedores, hasta llegar a los datos económicos globales (estadísticas macroeconómicas o microeconómicas). Jon Reed, analista independiente, mentor de *SAP* y bloguero en *JonERP.com*, no ve demasiado claro que *Google* se lance a crear un conjunto de aplicaciones ERP o a comprar una empresa de sistemas ERP «sería una decisión extrema», afirma. Sin embargo, «si una empresa tipo *Google* fuera capaz de presentar una forma de recopilar toda esta información de forma conjunta en un entorno basado en la nube, para posteriormente conectarla de alguna forma a una plataforma estructurada (uniendo la información estructurada y la información no estructurada) estaríamos ante un hito muy importante».
- La aparición del ERP en la nube (Abril 2010): *Netsuite y Lawson Software*, entre otras empresas, fueron las primeras que adoptaron las tecnologías de nube para los sistemas ERP. Comenzaron ofreciendo a medianas empresas y organizaciones soluciones de sistemas ERP ligeros, flexibles y asequibles.
- Coordinación del ERP con los procesos de negocio (Julio 2010): No todas las organizaciones que han invertido en sistemas ERP han logrado el éxito de sus iniciativas. Son numerosos los casos de

implementaciones erróneas y, en algún caso, han sido un fracaso total. La implementación de ERP es un problema socio técnico que necesita una perspectiva diferente a la de las innovaciones informáticas, depende profundamente de una perspectiva equilibrada de toda la organización. Entre los principales factores de éxito estratégicos podemos encontrar la coordinación de los procesos de negocio y de los procesos ERP integrados, que se encuentran bajo la influencia de la cultura de la organización.

- La implantación de *SaaS* se duplica (Agosto 2010)
- Mayor adopción de la Inteligencia empresarial (BI) (Diciembre 2010): A finales del año 2010 aumentó la adopción de la Inteligencia empresarial (BI), ya que el 35% de las organizaciones comenzó a emplear BI dominante y el 67% de las mejores empresas de cada sector empezó a ofrecer BI autoservicio.
- Tendencias de la Inteligencia empresarial (Enero 2011): En 2011, las principales tendencias emergentes de Inteligencia empresarial fueron los servicios en la nube, la visualización de datos, el análisis predictivo y el *Big Data*.
- #*IBMBigData* (2011): En 2011, IBM introdujo la etiqueta de Twitter, #*IBMbigdata*, que tenía como objetivo desarrollar el sitio web temático del *Big Data* que crearon en 2008 con intención de integrarlo en sus acciones de mercadeo.
- El crecimiento real de los datos (Febrero 2011): En un artículo titulado *The World's Technological Capacity to Store, Communicate and Compute Information de Science Magazine*, se calculó que la capacidad mundial de almacenamiento de información creció a una tasa anual del 25% anual desde 1987 hasta 2007. En el mismo sentido, se afirmó que, en el año 1986, el 99,2% del almacenamiento de datos era analógico, pero en 2007 el 94% de dicho almacenamiento era digital. Esto supone un cambio radical en un periodo de tiempo de tan solo 20 años (en 2002, el almacenamiento digital superó al no digital por primera vez).

- Las grandes empresas amplían sus sistemas de almacenamiento de datos (Mayo 2011): Los científicos del *McKinsey Global Institute* publicaron el artículo titulado *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. En él, se estimó que «en 2009, casi todos los sectores de la economía estadounidense tendrán, de media, un mínimo de 200 *terabytes* de datos almacenados (un tamaño que supone el doble del almacén de datos de la cadena de tiendas *Wal-Mart* del año 1999) por cada empresa con más de 1000 empleados». En el mismo sentido, también afirman que los sectores de inversión y de valores estaban a la zaga en volumen de datos almacenados por organización. Los científicos calcularon que, solo en 2010, las grandes empresas guardaron 7,4 *exabytes* de datos originales, mientras que los consumidores almacenaron 6,8 *exabytes*.
- Capacidad de la información (2012): En 2012, en el artículo *Tracking the Flow of Information into the Home* del *International Journal of Communication*, se calculó que el suministro de información por parte de los medios de comunicación a los hogares estadounidenses había pasado de ser de unos 50.000 minutos al día en el año 1960, a cerca de 900.000 en 2005. Igualmente, se calculó que los Estados Unidos «se estaban aproximando a los mil minutos de contenido a través de medios de comunicación disponibles por cada minuto disponible para su consumo».
- Lanzamiento Mundial de IPv6 (Junio 2012): El 6 de junio de 2012, la *Internet Society* llevó a cabo el Lanzamiento Mundial de IPv6 con el fin de que los participantes (*Wikipedia, Google* o *Facebook*, entre otros) desplegaran IPV6 de forma permanente en sus productos y servicios. El *Internet Protocol version 6* (IPv6) es la versión más reciente del protocolo de Internet, que proporciona un sistema de identificación y localización para dispositivos dentro de una red y el enrutamiento a través de Internet. El IPv6 fue desarrollado por la *Internet Engineering Task Force* (IETF) para resolver el problema del agotamiento de

direcciones IPv4 (el formato de 32 bits del protocolo IPv4 soporta "únicamente" 4300 millones de direcciones IP).

- SAP HANA (Diciembre 2013): Las empresas empiezan a implementar nuevas tecnologías en memoria, como *SAP HANA*, para analizar y optimizar cantidades de datos masivas. Las empresas cada vez confían más en el uso de datos como activo de negocio para lograr ventajas sobre la competencia. El *Big Data* muestra el camino, ya que es indudablemente la principal nueva tecnología a entender y utilizar para mantenerse al día en un mercado que cambia tan rápido como el actual.
- Adopción de ERP en la nube (Febrero 2014): Mientras que el 47% de las organizaciones encuestadas por *Gartner* tienen pensado migrar sus sistemas ERP principales a la nube en un plazo de cinco años, el ERP en la nube presenta una tasa de aceleración superior a la prevista inicialmente, debido a las estrategias de ERP de dos niveles que amplían el sistema ERP existente de la empresa y permiten llegar a nuevos mercados y escalar a una velocidad más alta.
- El año del Internet de las cosas (Noviembre 2014): El *IoT* se ha convertido en una fuerza poderosa para la transformación de negocios, y su enorme impacto afectará en los próximos años a todos los sectores y todas las áreas de la sociedad. Existen enormes redes de objetos físicos dedicados (cosas) que incorporan tecnología para detectar o interactuar con su estado interno o medio externo. Según *Gartner*, había 3700 millones de "cosas" conectadas en uso en 2014 y esa cifra se elevará hasta los 4900 millones en 2015.
- Ciudades inteligentes (2015): Una ciudad inteligente (*smart city*) hace uso del análisis de información contextual en tiempo real para mejorar la calidad y el rendimiento de los servicios urbanos, reducir costes, optimizar recursos e interactuar de forma activa con los ciudadanos. Según estimaciones de *Gartner* habrá más de 1100 millones de dispositivos conectados y en uso en diversas ciudades en 2015, incluyendo sistemas de iluminado *LED* inteligentes, de monitorización de

salud, cerraduras inteligentes y numerosas redes de sensores para detección de movimiento, estudio de contaminación atmosférica, etc.

# ANEXO 11: Instalación y configuración de Herramientas

## Instalación y Configuración de un clúster de Apache Cassandra

### Preparación del Entorno:

1. Entrar con su usuario saren al servidor.
2. Cree la siguiente estructura de directorio:

- /saren/
  - bd/
  - hdfs/
  - instaladores/

```
$> mkdir /saren
$> mkdir /saren/bd
$> mkdir /saren/hdfs
$> mkdir /saren/instaladores
```

3. Mueva los siguientes archivos a la carpeta ***instaladores*** utilizando el comando ***scp*** desde su máquina local. Puede borrar luego esta carpeta.

- apache-cassandra-2.1.8-bin.tar.gz
- hadoop-2.6.0.tar.gz
- jdk-8u45-linux-x64.tar.gz

```
$> scp jdk-8u45-linux-x64.tar.gz saren@192.16.11.86:/saren/instaladores

$> scp hadoop-2.6.0.tar.gz saren@192.16.11.86:/saren/instaladores

$>  scp apache-cassandra-2.1.5-bin.tar.gz saren@192.16.11.86:/saren/instaladores
```

```
$> ls
apache-cassandra-2.1.8-bin.tar.gz  hadoop-2.6.0.tar.gz  jdk-8u45-linux-x64.tar.gz
```

## 1 Instalación de Java

1. Ingresar al servidor con el usuario ***saren***.

```
$> ssh saren@<ip-servidor>
```

2. Descomprima el archivo jdk-8u45-linux-x64.tar.gz ubicado en ***/saren/instaladores/jdk-8u45-linux-x64.tar.gz.*** Mueva el contenido de la carpeta descomprimida al directorio ***/usr/lib/jvm/jdk1.8.0_45***. Sí el directorio no existe, debe crearlo.

```
$> cd /saren/instaladores
$> sudo mkdir /usr/lib/jvm/
$> sudo mkdir /usr/lib/jvm/jdk1.8.0_45/
$> tar -xvf jdk-8u45-linux-x64.tar.gz
$> sudo mv jdk1.8.0_45/* /usr/lib/jvm/jdk1.8.0_45/
$> rm -rf jdk1.8.0_45
```

3. Ejecutar los siguientes comandos para ***java, javac y javaws***, respectivamente:

```
$> sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk1.8.0_45/bin/java" 1

$> sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/lib/jvm/jdk1.8.0_45/bin/javac" 1

$> sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/lib/jvm/jdk1.8.0_45/bin/javaws/" 1
```

4. Ahora, se le debe indicar al sistema que utilice ***jdk1.8.0_45*** como java por defecto.

```
$> sudo update-alternatives --config java
```

**Nota:** Cuando el sistema pregunte que opción desea, seleccione la que indique ***jdk1.8.0_45***

5. Verificar con el siguiente comando que el java se instaló correctamente:

```
$> java -version
```

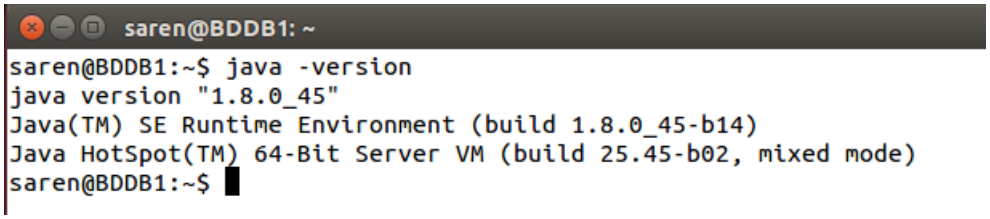



*Ilustración 1: Evidencia de la instalación de java en ambiente de desarrollo*

6. Para termina, incluir la variabla ***JAVA_HOME*** al ***path*** del sistema. Para esto, edite el archivo ***/home/saren/.bashrc***

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_45
export PATH=$PATH:$JAVA_HOME/bin
```

**Nota:** Luego de modificar el archivo **.bashrc**, este debe ser cargado nuevamente para que tome los cambios realizados, ejecute **source ~/.bashrc** para volver a cargar el archivo.

## 2 Instalación de Cassandra

Inicie sesión en el servidor con las credenciales**,** descomprima el archivo ***apache-cassandra-2.1.8-bin.tar.gz*** que se encuentra en el directorio

***/saren/instaladores/*** y mueva la carpeta descomprimida al directorio ***/saren/bd/***.

```
$> cp /saren/instaladores/apache-cassandra-2.1.8-bin.tar.gz /saren/bd/
$> cd /saren/bd/
$> tar xzvf apache-cassandra-2.1.8-bin.tar.gz
$> rm -f apache-cassandra-2.1.8-bin.tar.gz
$> mv apache-cassandra-2.1.8/* .
$> rm -rf apache-cassandra-2.1.8/
```

La estructura de directorio debe quedar de la siguiente manera:

```
/saren/
bd/
bin/
conf/
javadoc/
tools/
interface/
lib/
pylib/
LICENSE.txt
NOTICE.txt
CHANGES.txt
NEWS.txt
```

## 3 Configuración de Cassandra

Una vez que inicie sesión en el servidor de desarrollo (**BDDB1**)**,** ubíquese en el directorio ***/saren/bd/conf/*** y edite el archivo ***cassandra.yaml***.

```
$> cd /saren/bd/conf/

$> nano cassandra.yaml
```

Las propiedades a configurar en el archivo son las siguientes:

**- cluster_name:** Nombre del clúster.

**- num_tokens:** Define el número de tokens que se asignan al azar al nodo. Los tokens son unos identificadores (números) que cassandra utiliza para el particionado de la data y hacer búsquedas más rápidas.

**- data_file_directories:** Directorio de datos almacenados en cassandra.

**- commitlog_directory:** Directorio de almacenamiento de los *logs* de cassandra.

**- saved_caches_directory:** Directorio de la *caché* de cassandra.

**- seeds:** IP interna de cada nodo semilla.

**- listen_addres:** Dirección IP o Hostname local del nodo.

**- rpc_address:** IP o Hostname que los clientes deben utilizar para conectarse a este nodo.

**- endpoint_snitch:** Cassandra utiliza esta propiedad para localizar a los otros nodos y enrutar las solicitudes.

Las propiedades deben quedar configuradas como se detalla a continuación. Tome en cuenta que el nombre del servidor cambia según el nodo que se esté configurando. Solo la propiedad ***seed_provider*** toma el mismo valor para todos

los nodos, ya que esta señala el nodo semilla del clúster, y todos los nodos tienen referencia la mismo nodo semilla (BDDB1).

```
cluster_name: 'SAREN'
num_tokens: 256
data_file_directories: /cassandra/data
commitlog_directory: /var/cassandra/commitlog
saved_caches_directory: /var/cassandra/saved_caches
seed_provider:
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          - seeds: BDDB1
listen_address: BDDB<n>
rpc_address: BDDB<n>
endpoint_snitch: GossipingPropertyFileSnitch
```

Donde ***<n>*** corresponde al servidor que se este configurando: 1,2,3.

Crear los directorios de cassandra para almacenar ***data, commitlog y saved_caches***.

```
$> sudo mkdir -p /cassandra/data
$> sudo mkdir -p /var/cassandra/commitlog
$> sudo mkdir -p /var/cassandra/saved_caches
```

Cambie el propietario del directorio ***/var/lib/cassandra/***.

```
$> sudo chown -R saren:saren /var/cassandra/
$> sudo chown -R saren:saren /cassandra/
```

Editar el archivo ***/etc/hosts***

```
<ip-nodo1>    BDDB1
<ip-nodo2>    BDDB2
<ip-nodo3>    BDDB3
```

***Nota:*** *Este archivo tiene la misma configuración para los tres nodos del clúster.*

Para desplegar la base de datos ***Cassandra*** (después de configurar todos los nodos), se inicia cassandra con el comando:

```
$> cd /saren/bd/bin
$> ./cassandra BDDB-<n>
```

Donde ***<n>*** puede tomar uno de los siguientes valores, dependiendo del nodo que se está iniciando: ***1, 2, 3***. Se debe tomar en cuenta que el primer nodo a iniciar debe ser el ***semilla (BDDB1)***.

Para conectarse a cassandra desde su máquina local, debe tener configurado el archivo ***/etc/hosts*** de la siguiente manera:

```
<ip-nodo1>    BDDB1
<ip-nodo2>    BDDB2
<ip-nodo3>    BDDB3
```

Luego, se debe ejecutar el siguiente comando:

```
$> cd <directorio_instalacion_cassandra>/bin
$> ./cqlsh BDDB-<n>
```

Donde ***<n>*** puede tomar uno de los siguientes valores, dependiendo del nodo que este conectando: ***1, 2, 3***.

***Nota:*** *se presume que en su máquina local tiene una instalación simple de cassandra.*

Para verificar el estado del clúster, ejecute:

```
$> cd /saren/bd/bin/

$> ./nodetool status
```

```
saren@BDDB1:/saren/bd/bin$ ./nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address       Load       Tokens  Owns    Host ID                               Rac
UN  192.16.11.86  242,55 KB  256     ?       8396e45e-44ad-472f-b68d-7eec4894bbe2  rac
UN  192.16.11.87  475,23 KB  256     ?       5280cd60-b57f-4a73-ae3e-320a8cdfb9f1  rac
UN  192.16.11.88  489,35 KB  256     ?       772cf872-5ffb-470a-8c07-c12242d04856  rac
```

*Ilustración 2: Evidencia que verifica estado del clúster*

Para detener cassandra en cada nodo ejecute los siguientes comandos:

```
$> sudo bash

$> ps aux | grep cassandra

$> kill <pid>
```

Donde ***<pid>*** es el id del proceso que se encuentra ejecutando la instancia de ***cassandra.***

Adicionalmente, se puede añadir la variable ***CASSANDRA_HOME*** al ***path*** del sistema editando el archivo ***/home/saren/.bashrc***:

```
export CASSANDRA_HOME=/saren/bd

export PATH=$PATH:$CASSANDRA_HOME/bin
```

12. Luego de modificar el archivo ./bashrch debe volver a cargarlo para que se tomen los cambios realizados:

```
$> source ~/.basrch
```

# 4 Configuración de la Seguridad en Cassandra

### Autenticación y Autorización interna en Cassandra

La autenticación interna se basa en las cuentas y contraseñas de acceso controlados en Cassandra y funciona para los siguientes clientes cuando se proporciona un nombre de usuario y contraseña para iniciarlo:

Astyanax

Cassandra-cli

Cqlsh

Controladores DataStax: Producidos y certificados por DataStax para trabajar con Cassandra.

Hector

Pycassa

En otro orden de ideas, la autorización interna es utilizada para conceder o revocar permisos que acceden a los datos de Cassandra.

Cuando se instala Cassandra, ésta no posee ninguna configuración de seguridad, lo que implica que cualquier persona puede conectarse a la base de datos sin previa autorización.

A continuación, se muestran los pasos para configurar la autenticación y autorización interna en Cassandra

**Procedimiento**

1. Para iniciar el proceso es importante tomar en cuenta los siguientes aspecos:

Primero se debe configurar el nodo semilla y luego el resto de los nodos que se van a emplear en el cluster

Para detener o reiniciar el cluster, se deben detener los nodos *dejando de ultimo el nodo semilla*

Para arrancar nuevamente el cluster se debe iniciar con el nodo semilla y luego con el resto de los nodos

2. Conéctese a cada uno de los nodos vía ssh.

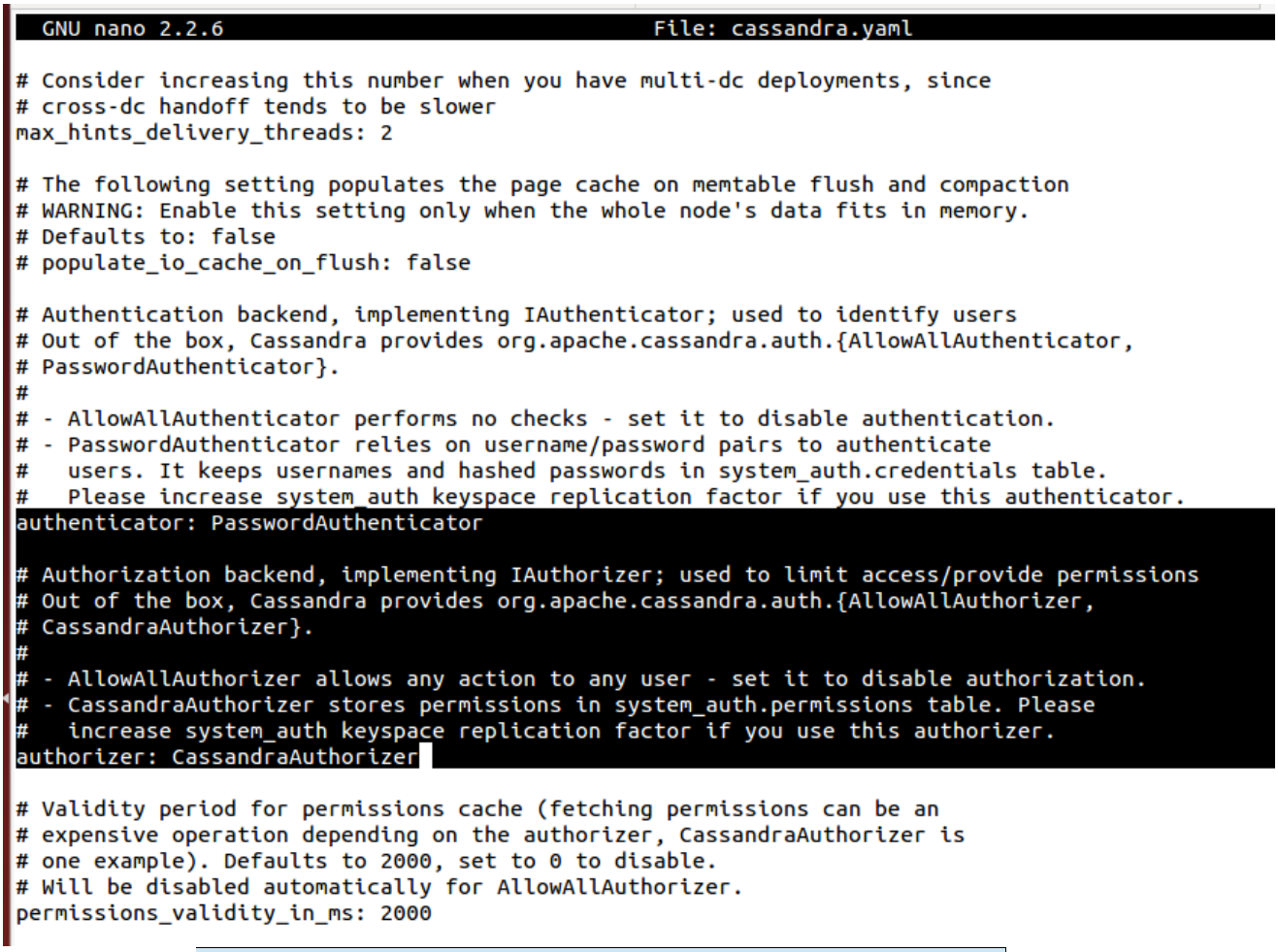
Edite el archivo *cassandra.yaml.* Modifique el valor de la opción *authenticator* y de la opción *authorizer.*

```
nano /saren/bd/conf/cassandra.yaml
```

Por defecto, la opción *authenticator* se establece en *AllowAllAuthenticator*. Se cambia el valor de esta opción como se muestra a continuación:

authenticator: PasswordAuthenticator

Por defecto, la opción authorizer se establece en *AllowAllAuthenticator*. Cambie el valor de esta opción como se muestra a continuación:




authorizer: CassandraAuthorizer

2. Luego de cambiar el archivo de configuración *cassandra.yaml* en cada nodo, debe reiniciar Cassandra, detener primero los nodos (para desarrollo: 192.16.11.88 y 192.16.11.87) y por último el nodo semilla (para desarrollo: 192.16.11.86).

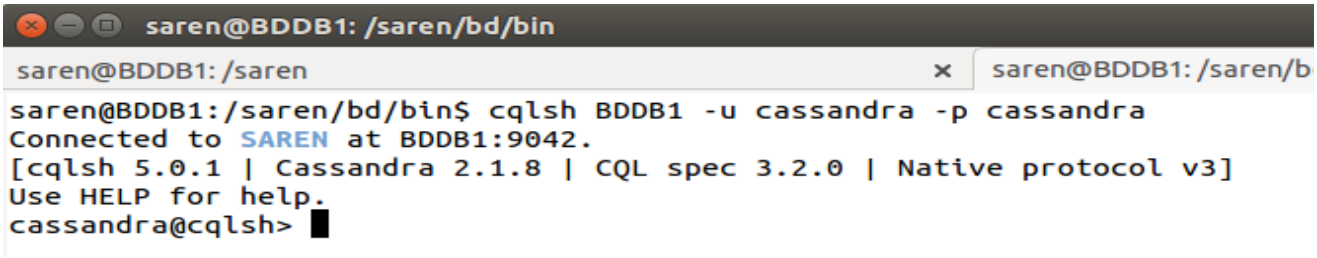
3. Se debe aumentar el factor de replicación para el keyspace *system_auth* a N (siendo N el número de nodos del clúster).

El valor predeterminado es 1, si el nodo con la réplica tiene problemas y deja de funcionar, no será capaz de iniciar sesión en el clúster porque el keyspace *system_auth* no se replicó y es allí donde se almacena la información sobre los accesos a la base de datos (usuarios, contraseñas, entre otros).

Conectados al nodo vía ssh y con permisos de root inicie el cliente cqlsh, luego se modifica el factor de replicación para el keyspace system_auth.

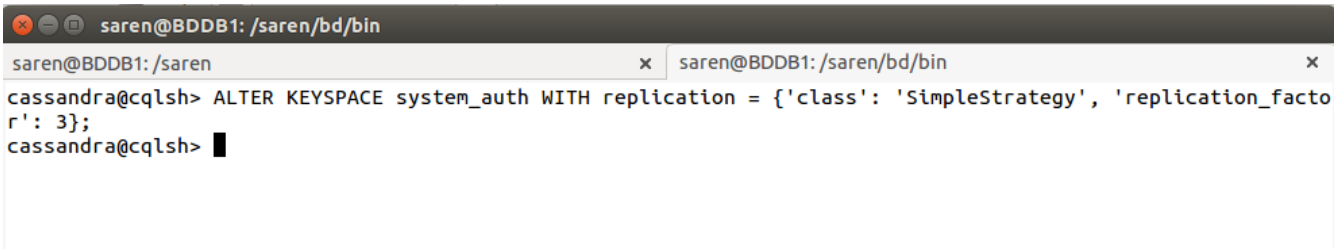
El valor predeterminado de superusuario, nombre y la contraseña, que se utiliza para iniciar el cliente cqlsh es ***cassandra***. Inicie cqlsh usando el nombre y la contraseña del superusuario.

```
CASSANDRA_HOME/bin/cqlsh <host> -u cassandra -p cassandra
```




Se modifica el factor de replicación del keyspace *system_auth*.

```
ALTER KEYSPACE system_auth WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3};
```




En cada nodo del clúster, comenzando con los nodos semilla, se ejecuta el comando nodetool *repair*. Se debe esperar hasta que la reparación se complete en un nodo, a continuación, puede pasar al siguiente nodo.

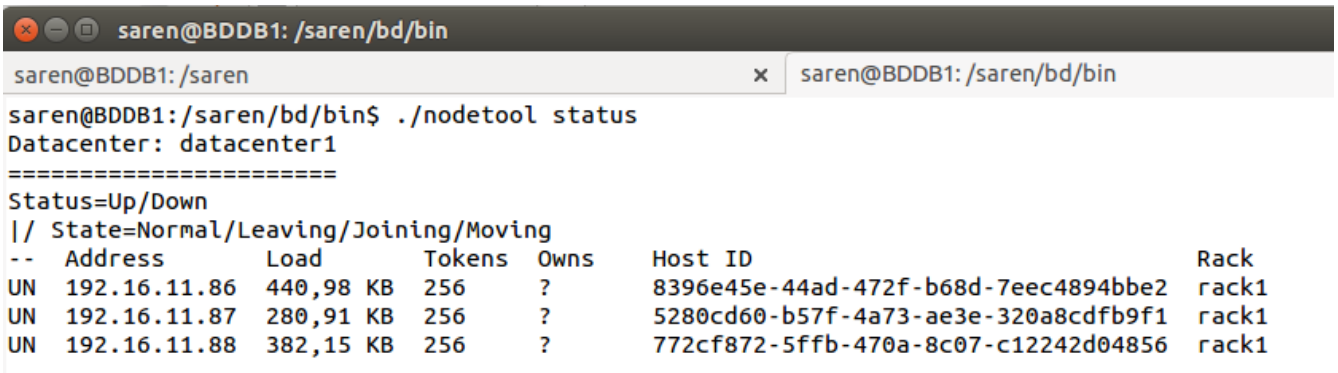
```
./nodetool  repair
```

4. **Reiniciar el cliente Cassandra.** Detenga los nodos y luego el nodo semilla. Para iniciarlos nuevamente, se inicia primero el nodo semilla luego los otros nodos.

5. **Verificar que todo el clúster se haya levantado de forma exitosa**. En la carpeta de instalación de Cassandra ejecute:

```
CASSANDRA_HOME/bin/nodetool status
```

Si todo el clúster inicio correctamente se debe mostrar lo siguiente:




6. **Inicie nuevamente el cliente cqlsh**. El valor predeterminado de superusuario, nombre y la contraseña, que utiliza para iniciar el cliente es *cassandra.* Inicie cqlsh usando el nombre y la contraseña del superusuario.

```
CASSANDRA_HOME/bin/cqlsh <host> -u cassandra -p cassandra
```

7. **Cree otro superusuario**. Se crea un usuario "*administrador*" el cual tiene asignados todos los permisos. Para crear el superusuario *administrador* se inicia cqlsh con el superusuario *cassandra* luego se ejecuta lo siguiente:

```
CREATE USER administrador WITH PASSWORD 'administrador2015#' SUPERUSER;
```

8. Asignar permisos al usuario *administrador.* Este usuario tiene todos los permisos sobre todos los keyspace de la base de datos. Conectados al cliente cqlsh ejecutamos lo siguiente:

```
GRANT ALL PERMISSIONS ON ALL KEYSPACES TO administrador;
```

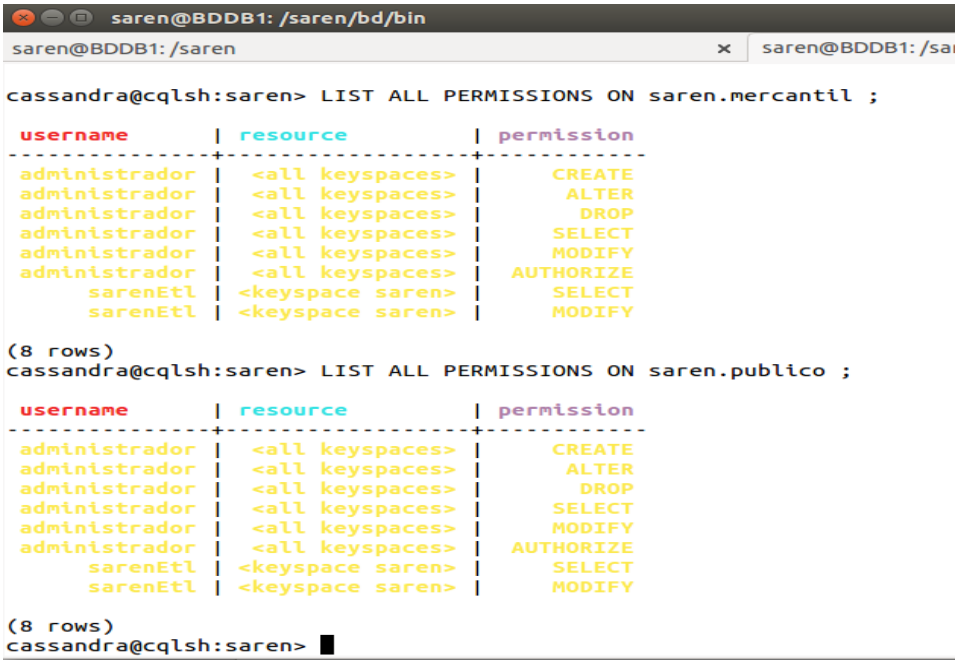
**NOTA: Para ejecutar los ETL que cargan y mantienen actualizada la base de datos Cassandra se requiere crear un usuario '*saren*ETL ' con contraseña '*sarenEtl2015*#', con permisos para leer/escribir.**

```
CREATE USER saren**ETL** WITH PASSWORD 'sarenEtl2015#' NOSUPERUSER ;

GRANT **MODIFY** PERMISSION ON KEYSPACE saren TO saren**ETL**;

GRANT **SELECT** PERMISSION ON KEYSPACE saren TO saren**ETL**;
```

9. Iniciar sesión con el usuario *administrador* y eliminar el usuario *cassandra:*

```
DROP USER IF EXISTS cassandra;
```

Liste los usuarios y sus permisos.




10. Ya puede utilizar las declaraciones CQL enumerados a continuación para configurar cuentas de usuario y conceder permisos para acceder a los objetos de la base de datos.

**Comandos CQL**

A continuación, se listan los comandos para la gestión de usuarios y permisos.

Crear usuarios:

```
CREATE USER user_name WITH PASSWORD 'password' (NOSUPERUSER | SUPERUSER)
```

Eliminar un usuario:

```
DROP USER IF EXISTS user_name
```

Modificar opciones de usuarios existentes:

```
ALTER USER user_name WITH PASSWORD 'password' ( NOSUPERUSER | SUPERUSER )
```

Nota: Los superusuarios pueden cambiar la contraseña de un usuario o el estado (nosuperuser, superuser). Los nosuperusuarios no pueden cambiar su estado a superusuario. Los usuarios normales sólo pueden cambiar su propia contraseña.

Listar usuarios existentes y su estado:

```
LIST USERS
```

Proporcionar acceso a los objetos de base de datos. Comando GRANT:

```
GRANT permission_name PERMISSION | ( GRANT ALL PERMISSIONS ) ON resource TO user_name
```

Los permisos que se pueden asignar son:

*ALL*

*ALTER*

*AUTHORIZE*

*CREATE*

*DROP*

*MODIFY*

*SELECT*

En la siguiente tabla se muestran los permisos necesarios para utilizar declaraciones CQL:

| Permiso | Declaración CQL |
|---|---|
| ALL | Todas las declaraciones |
| ALTER | ALTER KEYSPACE, ALTER TABLE, CREATE INDEX, DROP INDEX |

| Permiso | Declaración CQL |
|---|---|
| AUTHORIZE | GRANT, REVOKE |
| CREATE | CREATE KEYSPACE, CREATE TABLE |
| DROP | DROP KEYSPACE, DROP TABLE |
| MODIFY | INSERT, DELETE, UPDATE, TRUNCATE |
| SELECT | SELECT |

Los recursos sobre los que se asignan permisos son:

*ALL KEYSPACES*

*KEYSPACE keyspace_name*

*TABLE keyspace_name.table_name*

Listar permisos asignado a los usuarios:

```
LIST permission_name PERMISSION | ( LIST ALL PERMISSIONS )  ON resource OF user_name  NORECURSIVE
```

Revocar permisos a los usuarios:

```
REVOKE (permission_name PERMISSION ) | ( REVOKE ALL PERMISSIONS ) ON resource FROM user_name
```

## Instalación de Hadoop

Antes de comenzar la Instalación y configuración del clúster hadoop es importante entender los siguientes conceptos:

**DataNode:** Un DataNode almacena los datos en el sistema de archivos Hadoop (HDFS). Un sistema de archivos funcional tiene más de un DataNode, con los datos replicados a través de ellos.

**NameNode:** Es la pieza central de un sistema de archivos HDFS. Mantiene el directorio de todos los archivos en el HDFS y realiza un seguimiento donde todo el clúster mantiene los datos, es decir, metadata de los datos. No almacena los datos de estos archivos en sí.

**JobTracker:** Es el servicio dentro de Hadoop que gestiona los trabajo MapReduce hacia nodos específicos del clúster, idealmente los nodos que tienen los datos.

**TaskTracker:** Es un nodo del clúster que acepta tareas MapReduce y operaciones aleatorias de un JobTracker.

**Secondary Namenode:** El propósito del Secondary NameNode es tener un puesto de control en el HDFS. Es sólo un nodo ayudante para NameNode.

Una vez iniciada la sesión en el servidor, descomprima el archivo ***hadoop-2.6.0.tar.gz*** que se encuentra ubicado en el directorio ***/saren/instaladores/*** y mueva la carpeta descomprimida al directorio ***/saren/hdfs/***.

```
$> cp /saren/instaladores/ hadoop-2.6.0.tar.gz /saren/hdfs/

$> cd /saren/hdfs/

$> tar xzvf  hadoop-2.6.0.tar.gz

$> rm -f hadoop-2.6.0.tar.gz

$> mv hadoop-2.6.0/* .

$> rm -rf hadoop-2.6.0
```

2. La estructura de directorio debe quedar de la siguiente manera:

```
/saren/

hdfs/
```

```
bin/
etc/
include/
lib/
sbin/
share/
libexec/
LICENSE.txt
NOTICE.txt
README.txt
```

***Nota:*** *Para esta instalación ip-nodo-1 corresponde al servidor BDDB1, ip-nodo-2 BDDB2 y ip-nodo-3 BDDB3. Donde BDDB1 es el nodo maestro y BDDB2 y BDDB3 los nodos esclavos.*

## Configuración de Hadoop

A continuación se especifica la configuración del clúster de Apache Hadoop para SAREN, el cual cuenta con tres nodos, designando el servidor de <IP-nodo1> como nodo maestro-esclavo y los servidores de <IP-nodo2> y <IP-nodo3> como nodos esclavos.

Iniciar sesión en el servidor.

Editar el archivo /***etc/host*** en todos los nodos del clúster. Todos los nodos deben tener la misma configuración en el archivo /etc/hosts ya que esto permitirá conocer los demás nodos pertenecientes al clúster

```
<ip-nodo1>    BDDB1
<ip-nodo2>    BDDB2
<ip-nodo3>    BDDB3
```

***Nota:*** *Este archivo tiene la misma configuración para los tres nodos del clúster.*

Edite los siguientes archivos en el directorio ***/saren/hdfs/etc/hadoop***.

***hadoop-env.sh:*** Esta configuración debe aplicarse a todos los nodos Hadoop del clúster. En este archivo debemos especificar el directorio de instalación de Java, para ello modifique la propiedad JAVA_HOME.

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_45
```

***core-site.xml:*** El nodo maestro y los nodos esclavos deben utilizar el mismo valor para la propiedad ***fs.defaultFS.*** Todos los nodos esclavos deben apuntar al nodo maestro. Todos los servicios de Hadoop y clientes usan este archivo para localizar el NameNode, ya que este archivo contiene el nombre del sistema de archivos por defecto. El nodo NameNode sera el servidor BDDB1 con <ip-nodo1>. Además, se especifica la propiedad ***hadoop.tmp.dir,*** la cual se utiliza como la base para los directorios temporales a nivel local, y también en el HDFS. Agregue las siguientes propiedades dentro de la etiqueta <configuration>al archivo. Este archivo queda igual para todos los nodos.

```
<property>
        <name>fs.defaultFS</name>
        <value>hdfs:BDDB1:9000</value>
</property>

<property>
        <name>hadoop.tmp.dir</name>
        <value>file:/var/hadoop/data/hdfs/tmp</value>
</property>
```

***Nota:*** *Este archivo tiene la misma configuración para los tres nodos del clúster.*

Antes de configurar el archivo hdfs-site.xml debemos crear los directorios para los datos del datanode y el namenode.

```
$> sudo mkdir -p /var/hadoop/data/hdfs/namenode

$> sudo mkdir -p /var/hadoop/data/hdfs/datanode

$> sudo mkdir -p /var/hadoop/data/hdfs/tmp

$> sudo chown -R saren:saren /var/hadoop/*
```

***hdfs-site.xml:*** Esta configuración debe aplicarse al nodo maestro y a los nodos esclavos. En este archivo especificamos el directorio en el sistema de archivos local donde el NameNode almacenará su archivo de metadatos y DataNode almacenará los bloques de datos. Además, se indica el factor de replicación que tendrá la data del HDFS, esto significa que, por cada archivo almacenado en el HDFS, habrá una repetición redundante de ese archivo en algún otro nodo del clúster. Este archivo queda igual para todos los nodos.

Agregar al archivo las siguientes propiedades:

```
<property>
        <name>dfs.replication</name>
        <value>3</value>
</property>

<property>
        <name>dfs.name.dir</name>
        <value>file:/var/hadoop/data/hdfs/namenode</value>
</property>

<property>
        <name>dfs.data.dir</name>
        <value>file:/var/hadoop/data/hdfs/datanode</value>
</property>
```

***Nota:*** *Este archivo tiene la misma configuración para los tres nodos del clúster.*

***yarn-site.xml:*** Esta configuración debe aplicarse al nodo maestro y los nodos esclavos. Este archivo es necesario para que un nodo trabaje como un nodo hilado, porque posee la configuración adecuada que permite dividir las dos principales funcionalidades del JobTracker, la gestión de recursos (ResourceManager) y trabajo de planificación/monitoreo (NodeManager), en demonios separados (hilos). Los Nodos maestro y esclavo deben utilizar el mismo valor para las siguientes propiedades, y deben estar apuntando a nodo maestro solamente.

```
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>

<property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

<property>
        <name>yarn.resourcemanager.address</name>
        <value>BDDB1:8050</value>
</property>

<property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>BDDB1:8025</value>
</property>

<property>
        <name>yarn.resourcemanager.scheduler.address</name>
```

```
        <value>BDDB1:8035</value>
</property>
```

***Nota:*** *Este archivo tiene la misma configuración para los tres nodos del clúster.*

***mapred-site.xml:*** Esta configuración debe aplicarse solo al nodo maestro. En este archivo se especifica el host y el puerto donde se ejecuta el JobTracker y el framework para la ejecución de trabajos MapReduce. Este archivo no existe y se debe crear copiando el archivo ***mapred-site0.xml.template*** con el nombre de ***mapred-site.xml.***

```
<property>
        <name>mapreduce.jobtracker.address</name>
        <value>BDDB1:5431</value>
</property>

<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
```

Actualiza el archivo ***slaves*** que se encuentra en el directorio ***/saren/hdfs/etc/hadoop*** solamente para el nodo maestro. Coloque solo el nombre de los servidores que actúan como esclavos en el clúster. Este archivo es utilizado por los scripts de Hadoop para iniciar los servicios apropiados en los nodos maestro y esclavos.

```
BDDB1

BDDB2

BDDB3
```

Cree el archivo ***masters*** en el directorio ***/saren/hdfs/etc/hadoop*** solamente para el nodo maestro. Coloque solo nombre del servidor que actúa como maestro en el clúster. Este archivo es utilizado por los scripts de Hadoop para iniciar los servicios apropiados en los nodos maestro y esclavos.

```
BDDB1
```

4. Configuración SSH

Hadoop requiere acceso SSH para administrar sus nodos (máquinas remotas) y su máquina local. El próximo paso es generar una clave ssh sin contraseña de inicio de sesión entre el nodo maestro y los nodos esclavos. Ejecute los siguientes comandos sólo en el nodo maestro.

```
$> su – saren

$> ssh-keygen -t rsa -P ""

$> cat /home/saren/.ssh/id_rsa.pub >> /home/saren/.ssh/authorized_keys

$> chmod 600 authorized_keys

$> ssh-copy-id -i ~/.ssh/id_rsa.pub  BDDB2

$> ssh-copy-id -i ~/.ssh/id_rsa.pub  BDDB3

$> ssh BDDB1
```

```
$> ssh BDDB2

$> ssh BDDB3
```

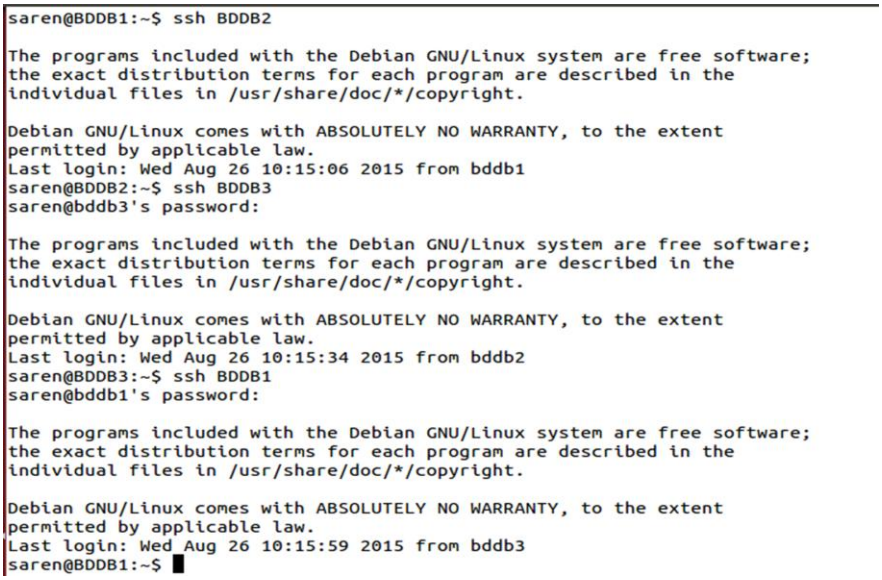



*Ilustración 3: Evidencia de conexión ssh*

5. Copiar las siguientes líneas en el archivo ***.bashrc*** ubicado en el directorio ***/home/saren*** Realizar este paso en el nodo maestro y cada nodo esclavo.

```
export HADOOP_HOME=/saren/hdfs

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

***Nota:*** *Luego de modificar el archivo* ***.bashrc****, éste debe ser cargado nuevamente para que tome los cambios realizados, ejecute* ***source ~/.bashrc*** *para volver a cargar el archivo.*

6. Formatear el NameNode

Antes de iniciar el clúster, se debe formatear el NameNode solo en el nodo maestro. Utilice el siguiente comando para formatear el nodo. Tenga en cuenta que cada vez que formatea el namenode se elimina la data almacenada en el HDFS.

```
$> cd /saren/hdfs/bin

$> ./hdfs namenode -format
```

7. Inicie el sistema de archivo desde el nodo maestro.

```
$> cd /saren/hdfs/sbin

$> ./start-dfs.sh
```

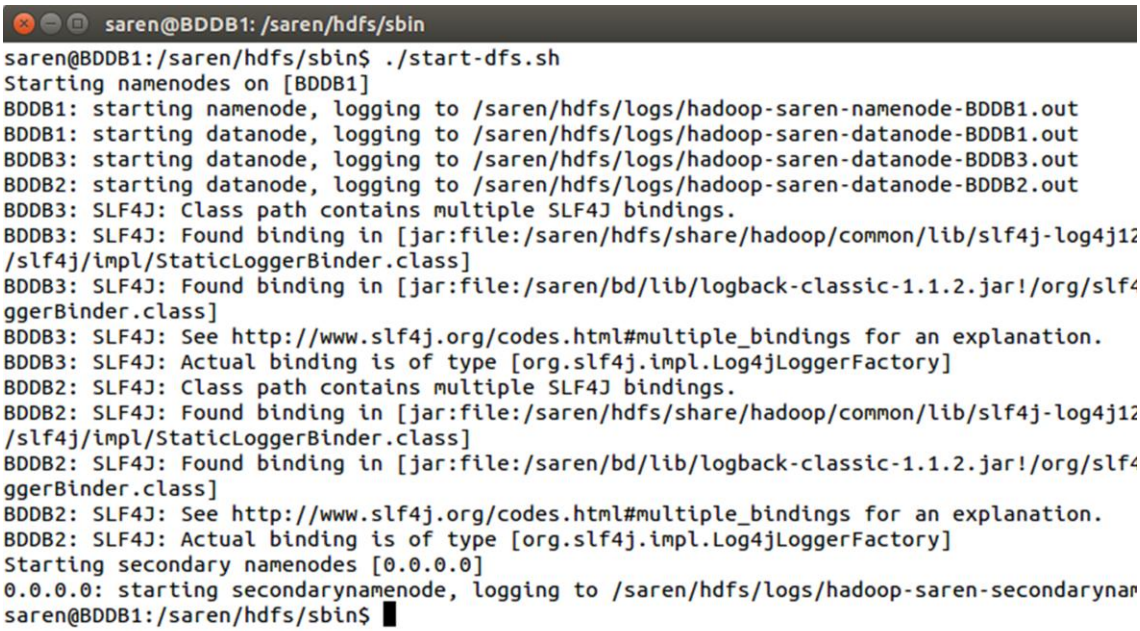



*Ilustración 4: Evidencia inicio del sistema de archivos*

Para validar que el DFS se inició con éxito, ejecute el siguiente comando en el nodo maestro y el nodo esclavo:

```
$> jps
```

La salida de este comando debe enumerar ***NameNode*** y SecondaryNameNode iniciado en el nodo maestro y ***DataNode*** en todos los nodos esclavos.

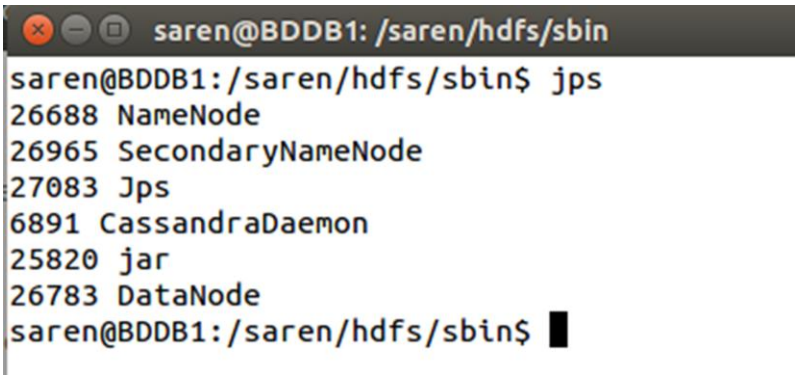



*Ilustración 5: Evidencia de salida del comando. Enumera NameNode y SecondaryNameNode*

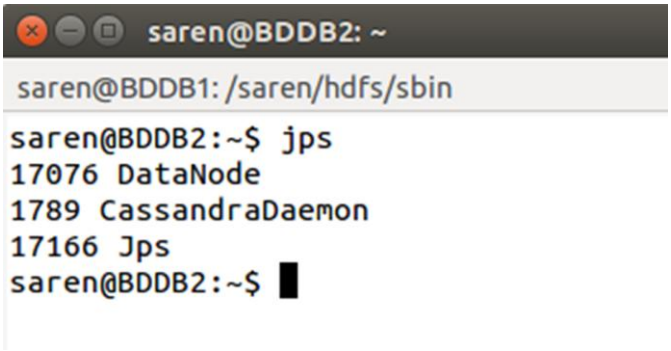



*Ilustración 6: Evidencia de salida del comando jps. Desde un nodo esclavo*

8. Inicie el Yarn MapReduce desde el nodo maestro.

```
$> cd /saren/hdfs/sbin

$> ./start-yarn.sh
```

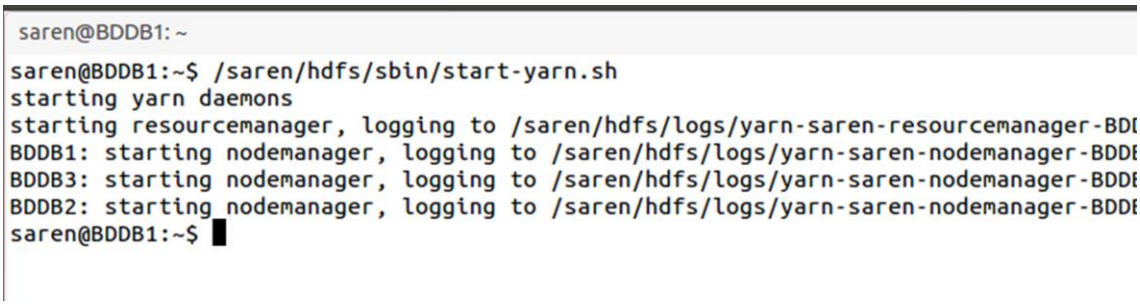



*Ilustración 7: Evidencia del inicio del Yarn*

Para validar que el Yarn Mapreduce se haya iniciado correctamente, ejecute el comando **jps** nuevamente en los nodos maestro y esclavos. La salida de este comando debe enumerar ResourceManager en el nodo maestro y NodeManager, en todos los nodos esclavos.

9. Valide el inicio exitoso del clúster a través de las Consolas Web:

Para ResourceManager: http://<ip-nodo-1>:8088

Para NameNode: http://<ip-nodo-1>:50070

***Nota**: Para esta instalación ip-nodo-1 corresponde al servidor BDDB1.*

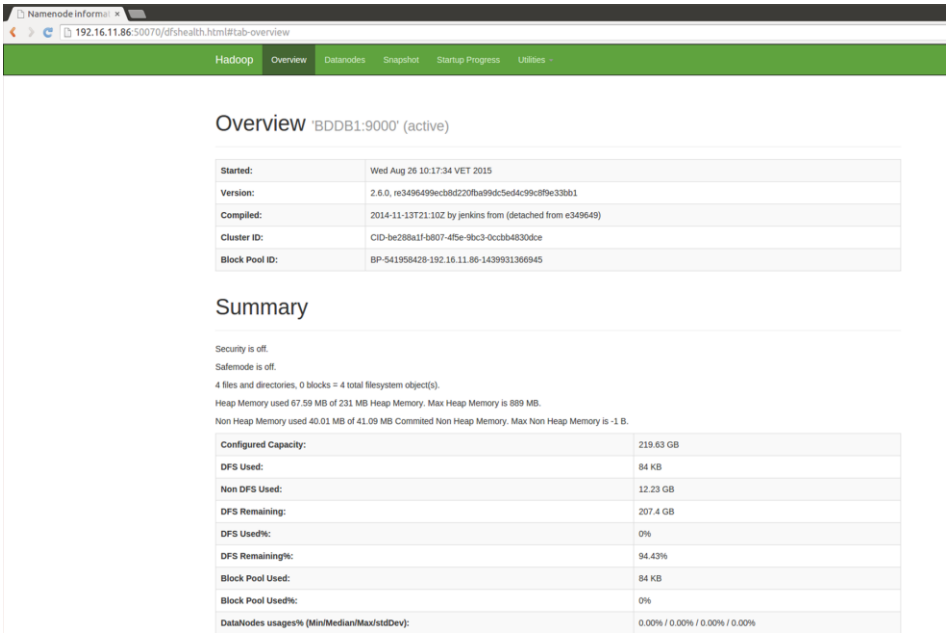



*Ilustración 8: Evidencia de inicio exitoso del clúster a través de Consola Web*

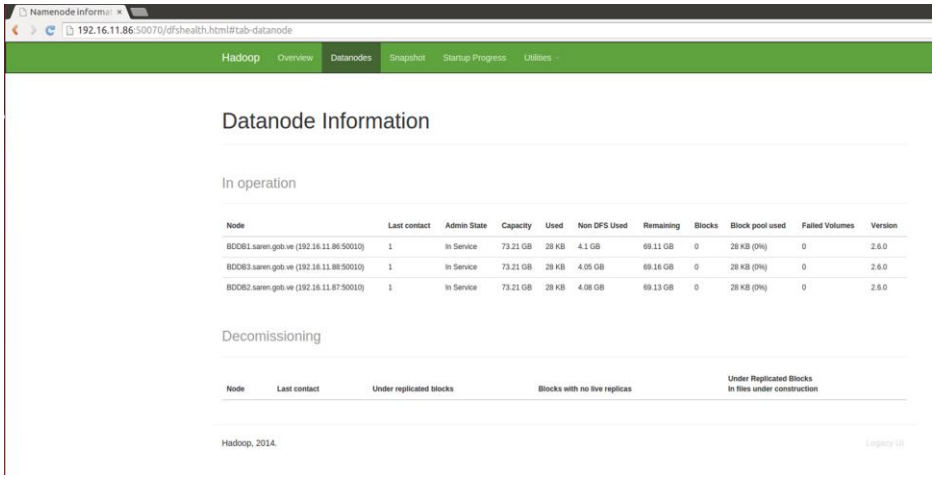



*Ilustración 9: Evidencia de inicio exitoso del clúster a través de Consolas Web. Vista de la información de los Datanode*

## Integración Cassandra – Hadoop

La instalación de Hadoop se realiza en la parte superior de una instalación de Cassandra, colocando un DataNode y un TaskTracker en cada uno de los nodos del clúster de Cassandra, el NameNode se ejecuta en su propio servidor independiente. Los nodos de Cassandra donde se ejecuta el DataNode y el TaskTracker se deben modificar como se describe a continuación:

1. Se debe indicar a Hadoop dónde encontrar todos los archivos jar que necesitará para ejecutar los MapReduce en Cassandra. Estos archivos (jar) se encuentran en la carpeta "lib" del directorio de instalación de Cassandra ***/saren/bd/lib***.

2. Todos los ajustes de configuración necesarios se encuentran en el archivo de entorno Hadoop ***/saren/hdfs/etc/hadoop/hadoop-env.sh***.

Para hacer lo anterior es necesario incorporar los directorios home y lib de Cassandra a la variable ***HADOOP_TASKTRACKER_OPTS***. Para esto edite el archivo: ***/saren/hdfs/etc/hadoop/hadoop-env.sh,*** *y coloque la siguiente propiedad.*

```
HADOOP_TASKTRACKER_OPTS="-classpath /saren/bd/*:/saren/bd/lib/* -Dhadoop.security.logger=ERROR,console -Dmapred.audit.logger=ERROR,console $HADOOP_TASKTRACKER_OPTS"
```

Agregue la variable de entorno HADOOP_CLASSPATH en el archivo: ***/saren/hdfs/etc/hadoop/hadoop-env.sh***.

```
export HADOOP_CLASSPATH="/saren/bd/*:/saren/bd/lib/*"
```

Repita esta modificación para los nodos ***esclavo***. Ahora cuando se proporcione un programa MapReduce para leer y escribir en Cassandra, Hadoop tendrá los componentes necesarios para ejecutarlo.

# Instalación de Apache Hive, Solr y HUE

A continuación, se muestra la instalación de las herramientas Hive, Solr y HUE, las cuales permiten la ejecución de consultas sobre la data almacenada en el clúster de ***Hadoop*** de una manera eficiente. Dichas instalaciones solo deben realizarse en el nodo Hadoop Maestro.

Preparación del Entorno:

Una vez copiado el paquete ***apache-hive-1.2.0-bin.tar.gz*** al directorio ***/saren/instaladores/***, se debe ingresar al servidor,

Luego, se crean los directorios ***/saren/hive/ , /saren/solr/ y /saren/hue/***

```
$> mkdir /saren/hive

$> mkdir /saren/hue

$> mkdir /saren/solr
```

## Instalación de Apache Hive

Descomprima el archivo ***apache-hive-1.2.0-bin.tar.gz,*** ubicado en el directorio ***/saren/instaladores/*,** dentro del directorio ***/saren/hive/***. La estructura de directorio debe quedar como se muestra a continuación:

```
/saren/
        hive/
            bin/
            conf/
            examples/
            hcatalog/
```

```
lib/
scripts/
LICENSE
NOTICE
README.txt
RELEASE_NOTES.txt
```

Edite el archivo ***/home/saren/.bashrc*** y agregue las siguientes lineas:

```
export HIVE_HOME=/saren/hive
export PATH=$PATH:$HIVE_HOME/bin
```

***Nota:*** *Luego de modificar el archivo* ***.bashrc****, éste debe ser cargado nuevamente para que tome los cambios realizados, ejecute* ***source ~/.bashrc*** *para volver a cargar el archivo.*

3. Hive ha actualizado la librería Jline2, pero existe una versión anterior de esta librería en el directorio HADOOP_HOME/share/hadoop/yarn/lib la cual debemos eliminar antes de iniciar Hive, para ello ejecute el siguiente comando:

```
rm -r /saren/hdfs/share/hadoop/yarn/lib/jline-0.9.94.jar
```

4. Ejecute ***hive*** con el siguiente comando y verifique la instalación:

```
$> hive

Logging initialized using configuration in jar:file:/saren/hive/lib/hive-common-1.2.0.jar!/hive-log4j.properties



hive>
```

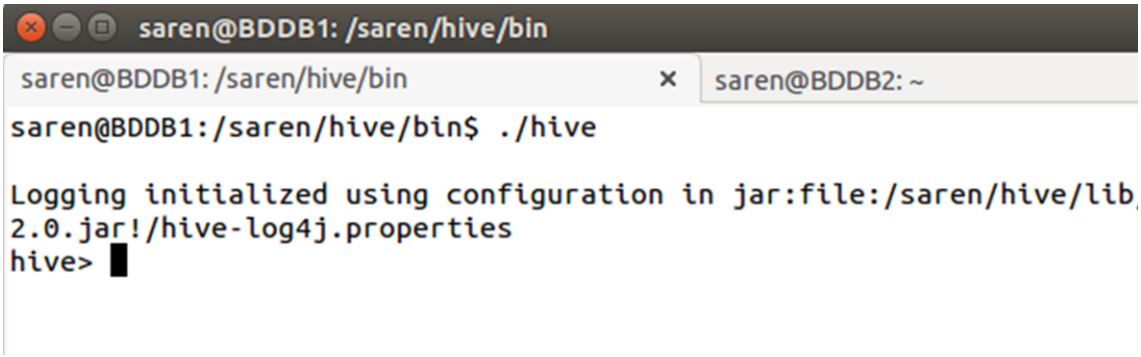



*Ilustración 10: Evidencia ejecución de Hive*

5. Para salir de ***hive*** ejecute:

```
hive> exit;
```

## Instalación de Solr

1. Descomprimir el archivo ***solr-5.1.0.zip,,*** ubicado en la ruta: ***/saren/instaladores/*****,** dentro de ***/saren/solr/***.

La estructura del directorio debe quedar como se muestra continuación:

```
/saren/
        solr/
            bin/
            contrib/
            dist/
            docs/
            example/
            licenses/
```

```
server/
        CAHNGE.txt
        LICENSE.txt
        README.txt
        NOTICE.txt
LUCENE_CHANGES.txt
```

2. Edite el archivo ***/home/saren/.bashrc*** y agregue las siguientes líneas:

```
export SOLR_HOME=/saren/solr
export PATH=$PATH:$SOLR_HOME/bin
```

***Nota:*** *Luego de modificar el archivo* ***.bashrc****, éste debe ser cargado nuevamente para que tome los cambios realizados, ejecute* ***source ~/.bashrc*** *para volver a cargar el archivo.*

3. Ejecute ***solr*** con el siguiente comando.

```
$> cd /saren/solr/bin
$> ./solr start
```

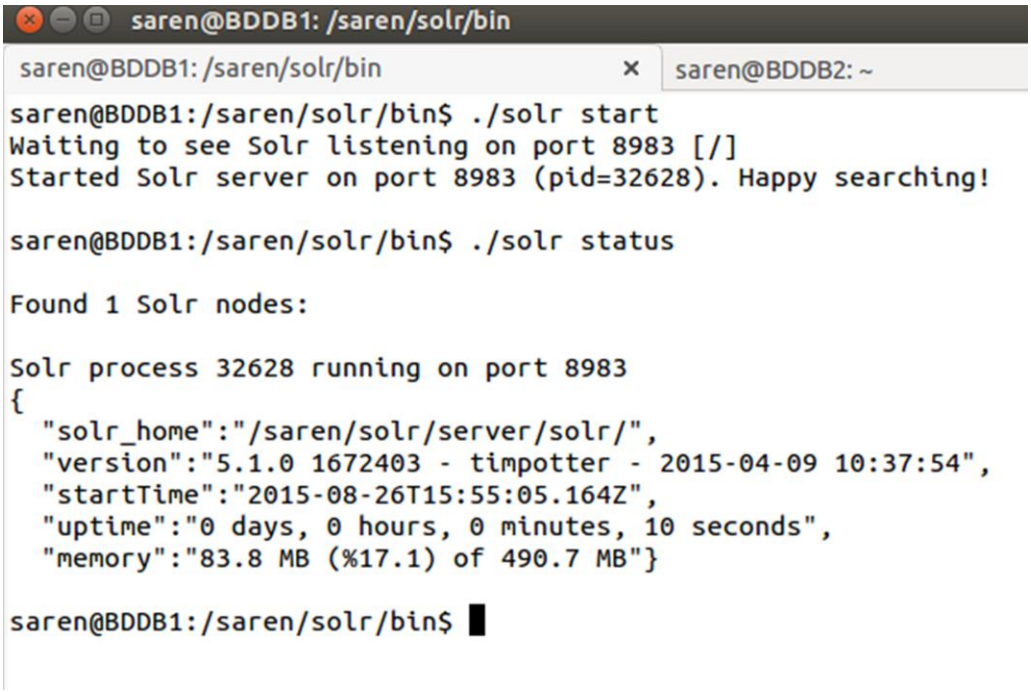
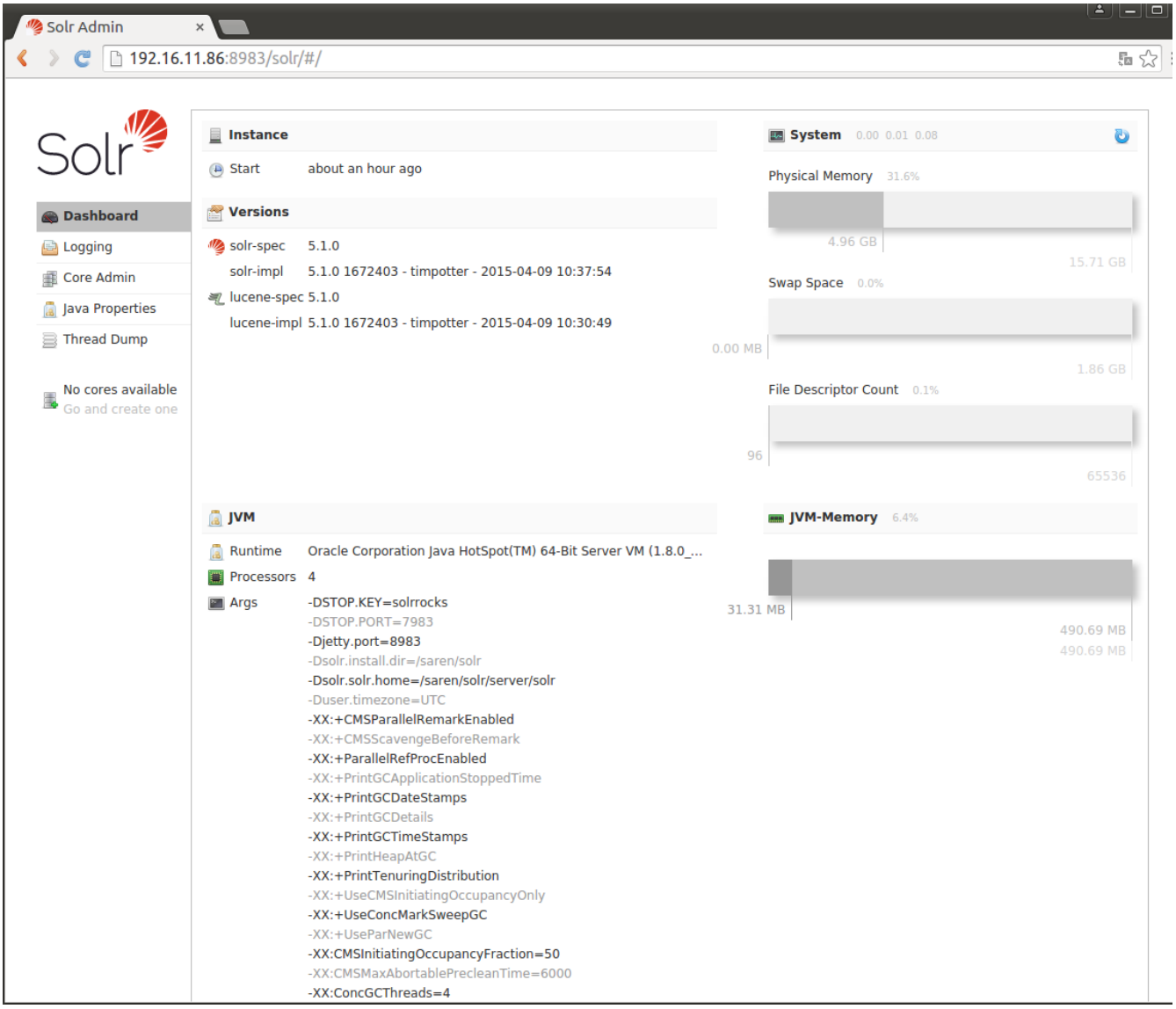



*Ilustración 12: Evidencia de instalación de Solr*

4. Verifique la instalación:

| **http://<ip-nodo-1>:8983** |
|---|




## Instalación de Hadoop User Experience (HUE)

Nota: La instalación debe realizarse en el mismo usuario donde esté instalado Hadoop. Este debe poseer permisos de administrador para poder realizar la instalación.

Si el usuario no posee los permisos de administrador, estos pueden ser otorgados desde el usuario root modificando el archivo sudoers con el comando sudo nano etc/sudoers, luego agregamos la línea "<usuario> ALL=(ALL:ALL) ALL" justo debajo de los privilegios de root y guardamos el

## Instalando requisitos previos

Ejecutar los siguientes comandos para instalar los paquetes de desarrollo

```
sudo apt-get install python2.7-dev make libkrb5-dev libxml2-dev
libxslt-dev libsqlite3-dev libssl-dev libldap2-dev python-pip
```

```
sudo apt-get install ant gcc g++ libmysqlclient-dev libssl-dev
libsasl2-dev libsasl2-modules-gssapi-mit libtidy-0.99-0 make
libldap2-dev maven python-dev python-setuptools libgmp3-dev
```

### Instalación de Maven

Utilizamos este comando para descargarnos la última versión de Maven Tarball

```
wget http://www-eu.apache.org/dist/maven/maven-
3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
```


Descomprimimos el archivo

```
tar xzvf apache-maven-3.3.9-bin.tar.gz
```

```
phd2014@phd2014:~/Descargas$ tar xzvf apache-maven-3.3.9-bin.tar.gz
apache-maven-3.3.9/boot/plexus-classworlds-2.5.2.jar
apache-maven-3.3.9/lib/maven-embedder-3.3.9.jar
apache-maven-3.3.9/lib/maven-settings-3.3.9.jar
apache-maven-3.3.9/lib/plexus-utils-3.0.22.jar
apache-maven-3.3.9/lib/maven-core-3.3.9.jar
apache-maven-3.3.9/lib/maven-model-3.3.9.jar
apache-maven-3.3.9/lib/commons-lang3-3.4.jar
apache-maven-3.3.9/lib/maven-settings-builder-3.3.9.jar
apache-maven-3.3.9/lib/maven-builder-support-3.3.9.jar
apache-maven-3.3.9/lib/plexus-interpolation-1.21.jar
apache-maven-3.3.9/lib/plexus-component-annotations-1.6.jar
apache-maven-3.3.9/lib/plexus-sec-dispatcher-1.3.jar
apache-maven-3.3.9/lib/plexus-cipher-1.7.jar
apache-maven-3.3.9/lib/maven-repository-metadata-3.3.9.jar
apache-maven-3.3.9/lib/maven-artifact-3.3.9.jar
apache-maven-3.3.9/lib/maven-plugin-api-3.3.9.jar
apache-maven-3.3.9/lib/org.eclipse.sisu.plexus-0.3.2.jar
apache-maven-3.3.9/lib/cdi-api-1.0.jar
apache-maven-3.3.9/lib/jsr250-api-1.0.jar
apache-maven-3.3.9/lib/javax.inject-1.jar
apache-maven-3.3.9/lib/org.eclipse.sisu.inject-0.3.2.jar
apache-maven-3.3.9/lib/maven-model-builder-3.3.9.jar
apache-maven-3.3.9/lib/guava-18.0.jar
apache-maven-3.3.9/lib/maven-aether-provider-3.3.9.jar
apache-maven-3.3.9/lib/aether-api-1.0.2.v20150114.jar
apache-maven-3.3.9/lib/aether-spi-1.0.2.v20150114.jar
apache-maven-3.3.9/lib/aether-util-1.0.2.v20150114.jar
```

Verificamos el valor de nuestra variable $JAVA_HOME (este valor variará según la versión de Java instalada)

```
echo $JAVA_HOME
```

```
phd2014@phd2014:~$ echo $JAVA_HOME
/usr/lib/jvm/java-1.7.0-openjdk-amd64
phd2014@phd2014:~$
```

Agregamos a PATH y verificamos instalación

```
export PATH=/rutahastaelarchivomaven/apache-maven-3.3.9/bin:$PATH
```

```
mvn -v
```

```
phd2014@phd2014:~$ export PATH=/home/phd2014/apache-maven-3.3.9/bin:$PATH
phd2014@phd2014:~$ mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T12:11:47-04:30)
Maven home: /home/phd2014/apache-maven-3.3.9
Java version: 1.7.0_03, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-7-openjdk-amd64/jre
Default locale: es_VE, platform encoding: UTF-8
OS name: "linux", version: "3.11.10.10-canaima+", arch: "amd64", family: "unix"
phd2014@phd2014:~$
```

## Instalando y configurando Hue.

Para descargar Hue utilizamos el siguiente comando

Wget https://dl.dropboxusercontent.com/u/730827/hue/releases/3.8.1/hue-3.8.1.tgz

```
phd2014@phd2014:~$ wget https://dl.dropboxusercontent.com/u/730827/hue/releases/3.8.1/hue-3.8.1.tgz
--2016-04-18 12:08:26--  https://dl.dropboxusercontent.com/u/730827/hue/releases/3.8.1/hue-3.8.1.tgz
Resolviendo dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 108.160.173.5
Conectando con dl.dropboxusercontent.com (dl.dropboxusercontent.com)[108.160.173.5]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 64637734 (62M) [application/x-gtar]
Grabando a: “hue-3.8.1.tgz.2”

100%[======================================================================>] 64.637.734  1,15M/s   en 55s

2016-04-18 12:09:23 (1,12 MB/s) - “hue-3.8.1.tgz.2” guardado [64637734/64637734]
```

Descomprimimos el archivo y renombramos la carpeta

tar xzvf hue-3.8.1.tgz

mv hue-3.8.1 hue

```
hue-3.8.1/apps/beeswax/src/beeswax/locale/pt/LC_MESSAGES/django.mo
hue-3.8.1/apps/beeswax/src/beeswax/locale/ja/
hue-3.8.1/apps/beeswax/src/beeswax/locale/ja/LC_MESSAGES/
hue-3.8.1/apps/beeswax/src/beeswax/locale/ja/LC_MESSAGES/django.po
hue-3.8.1/apps/beeswax/src/beeswax/locale/ja/LC_MESSAGES/django.mo
hue-3.8.1/apps/beeswax/src/beeswax/locale/en_US.pot
hue-3.8.1/apps/beeswax/src/beeswax/locale/ko/
hue-3.8.1/apps/beeswax/src/beeswax/locale/ko/LC_MESSAGES/
hue-3.8.1/apps/beeswax/src/beeswax/locale/ko/LC_MESSAGES/django.po
hue-3.8.1/apps/beeswax/src/beeswax/locale/ko/LC_MESSAGES/django.mo
hue-3.8.1/apps/beeswax/src/beeswax/data_export.py
hue-3.8.1/apps/beeswax/src/beeswax/server/
hue-3.8.1/apps/beeswax/src/beeswax/server/hive_server2_lib.py
hue-3.8.1/apps/beeswax/src/beeswax/server/dbms.py
hue-3.8.1/apps/beeswax/src/beeswax/server/__init__.py
hue-3.8.1/apps/beeswax/src/beeswax/admin.py
hue-3.8.1/apps/beeswax/src/beeswax/models.py
hue-3.8.1/apps/beeswax/src/beeswax/tests.py
hue-3.8.1/apps/beeswax/src/beeswax/__init__.py
hue-3.8.1/apps/beeswax/src/beeswax/conf.py
hue-3.8.1/apps/beeswax/src/beeswax/test_base.py
hue-3.8.1/apps/beeswax/src/beeswax/create_table_tests.py
hue-3.8.1/apps/beeswax/src/beeswax/common.py
hue-3.8.1/apps/beeswax/src/beeswax/settings.py
hue-3.8.1/apps/beeswax/src/beeswax/hive_site.py
hue-3.8.1/apps/beeswax/regenerate_thrift.sh
hue-3.8.1/Makefile.vars.priv
```

```
phd2014@phd2014:~$ mv hue-3.8.1 hue
phd2014@phd2014:~$
```

Para instalar Hue nos posicionamos en la carpeta Hue e instalamos

Cd Hue

make apps

```
phd2014@phd2014:~$ cd hue
phd2014@phd2014:~/hue$ make apps
```

Así debería finalizar la instalación

```
576 static files copied to '/home/phd2014/hue/build/static', 576 post-processed.
make[1]: se sale del directorio `/home/phd2014/hue/apps'
phd2014@phd2014:~/hue$
```

Después de terminar la instalación podemos iniciar Hue posicionándonos otra vez en la carpeta de Hue y ejecutando

```
./build/env/bin/hue runserver
```

```
phd2014@phd2014:~/hue$ ./build/env/bin/hue runserver
Validating models...

0 errors found
April 18, 2016 - 10:04:33
Django version 1.6.10, using settings 'desktop.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Y luego accedemos desde el navegador web a

```
http://127.0.0.1:8000/
```

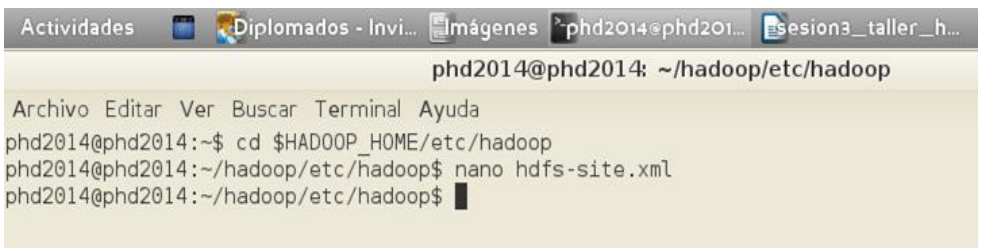
Hue solicitará que se cree un usuario para poder continuar. Es recomendado que este coincida con el nombre de usuario de GNU/linux donde está instalado Hue y Hadoop.


**Enlazar Hue y Hadoop**

Abrimos y modificamos el archivo hdfs-site.xml de hadoop y agregamos las siguientes líneas

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

```
cd  $HADOOP_HOME/etc/hadoop

nano hdfs-site.xml
```

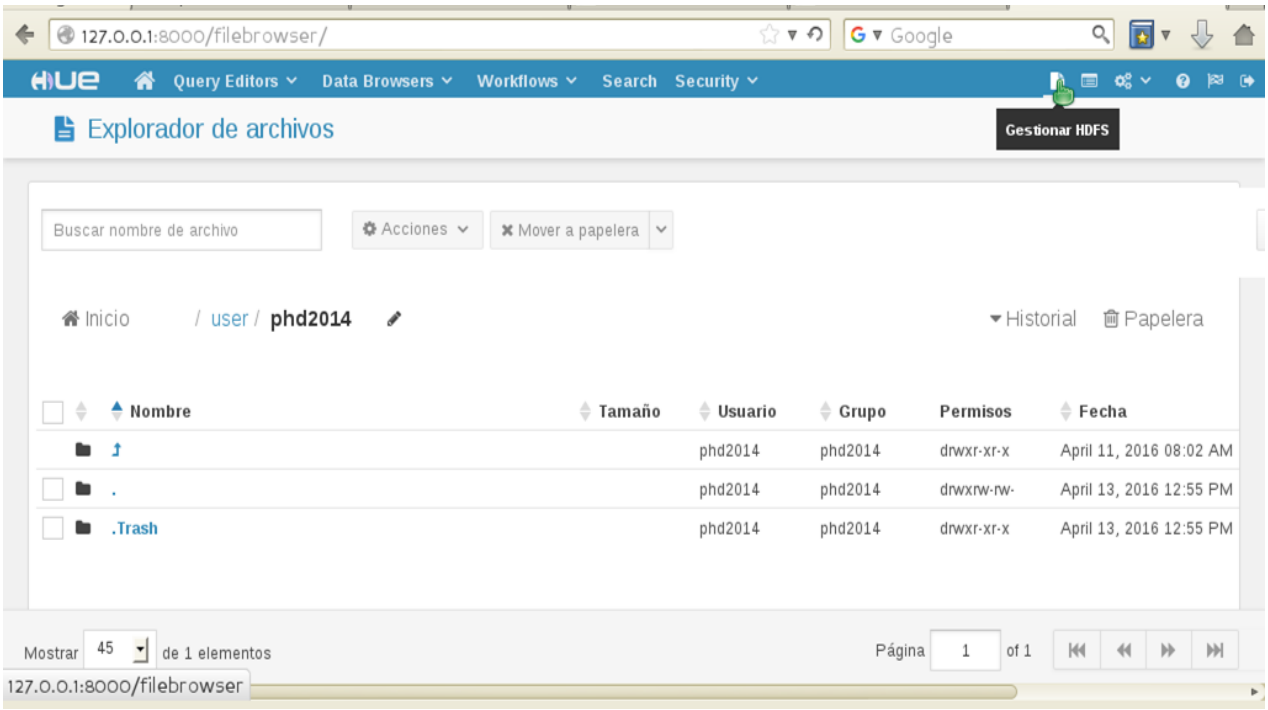



Abrimos y modificamos el archivo core.site.xml de hadoop y agregamos las siguientes líneas

```
<property>
  <name>hadoop.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

```
cd  $HADOOP_HOME/etc/hadoop
nano core-site.xml
```


Para verificar que Hue y Hadoop fueron enlazados exitosamente iniciar los demonios de hadoop, iniciar hue con el comando ./build/env/bin/hue runserver, acceder a http://127.0.0.1:8000/, y seleccionar la opción Gestionar HFDS que se encuentra en la parte superior derecha de la página principal de Hue. Ahí deberían visualizarse los archivos HDFS.



# ANEXO 12: Juicio de Experto

Caracas, Diciembre de 2015

Estimado (a) señor (a):

Motiva la presente el solicitar su valiosa colaboración en la revisión del instrumento anexo, el cual tiene como objeto obtener la validación del cuestionario que se aplicará para la fundamentación y desarrollo de la tesis de grado titulada “Desarrollo de una Arquitectura Big Data para Registros Mercantiles”.

Acudo a usted debido a sus conocimientos y experiencias en la materia, los cuales aportarían una útil y completa información para la culminación exitosa de este trabajo especial de grado.

Gracias por su valioso aporte y participación.

Atentamente,

Pedro Alfonso Paiva Muñoz

INSTRUCCIONES

A) Lea detenidamente las preguntas antes de responder.

B) Este instrumento de validación consta de una primera parte de identificación del experto, seguidamente otra en donde se identifica el título de la investigación, los objetivos, indicadores y alternativas de respuesta del cuestionario objeto de esta validación. Luego se encuentra una sección en la que se pide el juicio de experto con respecto al cuestionario, la cual está formada por siete preguntas, cuyas respuestas son: suficiente, medianamente suficiente e insuficiente, las dos primeras interrogantes, y si o no las restantes, seleccione la opción de su preferencia marcando una equis (x) en el espacio indicado para tal fin.

C) Seguido del juicio del experto se solicita una opinión sobre el instrumento diseñado.

D) Por último, se pide al experto que analizó el cuestionario una constancia de que realizó dicha tarea.

1. Identificación del Experto:

Nombre y Apellido:_____

Instituto donde Trabaja:_____

Título de Pregrado: _____

Título de Postgrado: _____Institución

donde lo obtuvo:_____

Año:_____Trabajos Publicados:

_____
_____

2. Título de la Investigación:

Desarrollo de una Arquitectura Big Data para Registros Mercantiles

2.1. Objetivos del Estudio:

2.2. Objetivo General:

Desarrollar una Arquitectura Big Data para Registros Mercantiles.

2.3. Objetivos Específicos:

- Diseñar la Arquitectura Big Data para registros Mercantiles.
- Seleccionar las herramientas necesarias para cada uno de los componentes de la arquitectura diseñada.
- Proponer e implementar la arquitectura con las herramientas seleccionadas.

3. Variable que se pretende medir:

Desarrollo de una Arquitectura Big Data para Registros Mercantiles.

3.1. Indicadores:

3.1.1. Seguridad

3.1.2. Capacidad de pruebas

3.1.3. Escalabilidad

3.1.4. Tolerancia a fallos

3.1.5. Análisis de datos

3.1.6. Visualización de datos

4. Alternativas de respuestas:

Sí

No

5. JUICIOS DEL EXPERTO:

5.1. En líneas generales, considera que los indicadores de la variable están inmersos en su contexto teórico de forma:

_____ Suficiente

_____ Medianamente suficiente

_____ Insuficiente

5.2. Considera que los reactivos del cuestionario miden los indicadores seleccionados para la variable de manera:

_____ Suficiente

_____ Medianamente Suficiente

_____ Insuficiente

5.3. Considera que existe pertinencia entre los objetivos de la investigación.

_____ Si

_____ No

Observaciones: _____

_____

5.4. Considera que existe pertinencia entre los indicadores y la variable de estudio.

_____ Si

_____ No

Observaciones: _____

_____

5.5. Considera que existe pertinencia entre los indicadores y los objetivos de la investigación.

_____ Si

_____ No

Observaciones: _____

_____

5.6. Considera que existe pertinencia entre los indicadores y las dimensiones de la investigación.

_____ Si

_____ No

Observaciones: _____

_____

5.7. Considera que los reactivos del cuestionario están redactados de manera adecuada.

_____ Si

_____ No

Observaciones: _____

_____

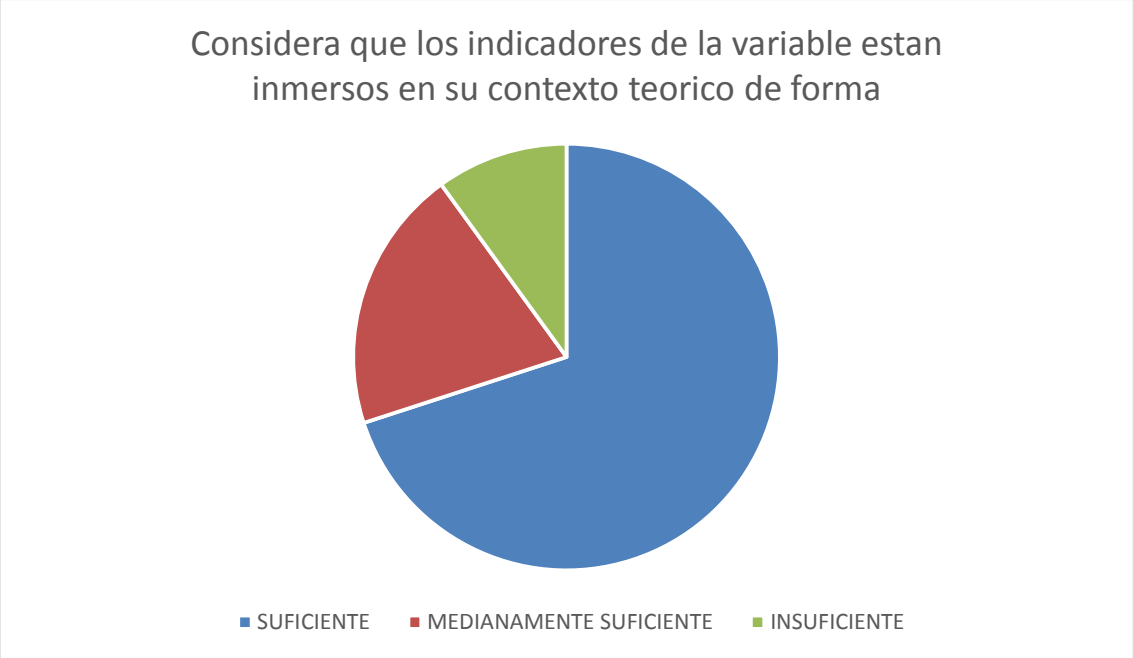
6. El instrumento diseñado es:

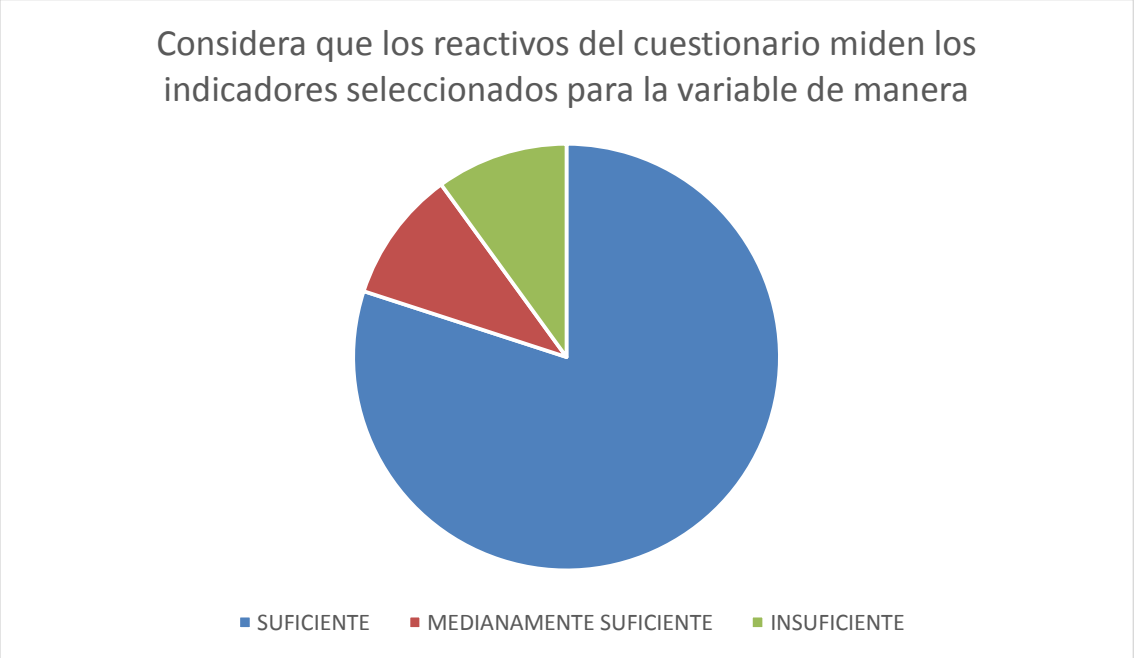
_____
_____
_____
_____
_____
_____
_____
_____

7. Constancia de Juicio de experto:

Yo, _____, titular de la cédula de identidad No. _____ certifico que realicé el juicio del experto al instrumento diseñado por el bachiller Pedro Paiva en el Trabajo Especial de Grado: **“DESARROLLO DE UNA ARQUITECTURA BIG DATA PARA REGISTROS MERCANTILES”**

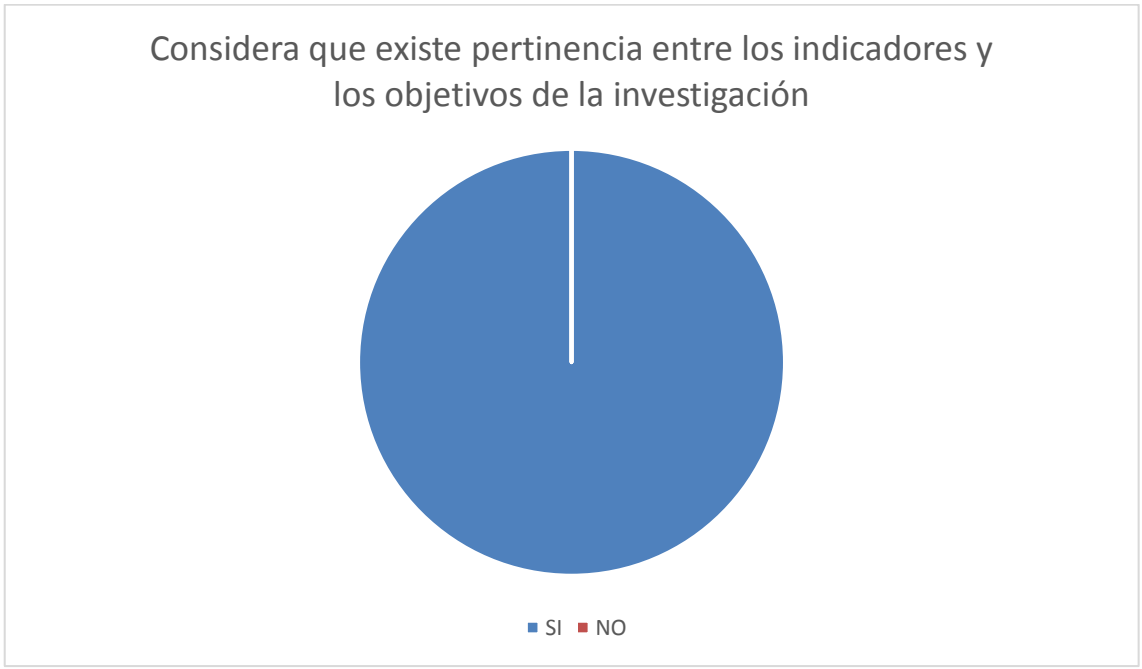
## Gráficos de los resultados



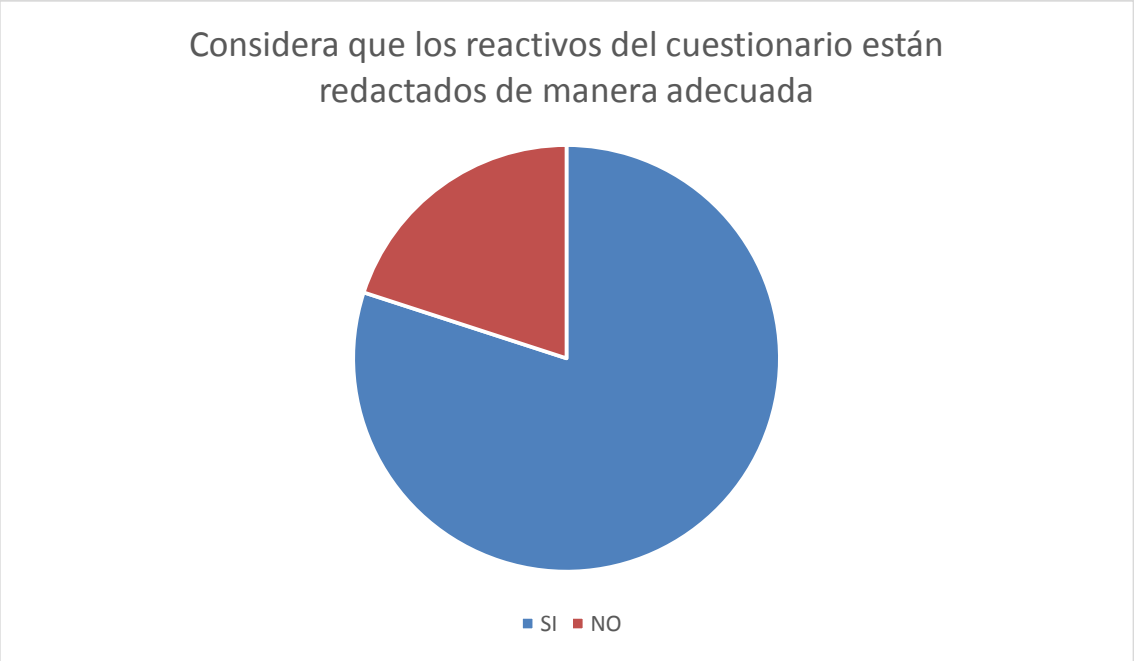




Considera que existe pertinencia entre los objetivos de la investigación
SI
NO


Considera que existe pertinencia entre los indicadores y la variable de estudio
SI
NO


Considera que existe pertinencia entre los indicadores y los objetivos de la investigación
SI
NO


Considera que existe pertinencia entre los indicadores y las dimensiones de la investigación
SI
NO


Considera que los reactivos del cuestionario están redactados de manera adecuada
SI
NO