



**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**  
*La Universidad Católica de Loja*

**ÁREA TÉCNICA**

**TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y  
COMPUTACIÓN**

**Implementación de una base de datos NoSQL en ambientes distribuidos.**

**TRABAJO DE TITULACIÓN.**

**AUTOR:** Jiménez Esparza, Christian Fabián.

**DIRECTOR:** Morocho Yunga, Juan Carlos, Ing.

**LOJA - ECUADOR**

**2016**



*Esta versión digital, ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <http://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>*

Septiembre, 2016

## APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero.

Juan Carlos Morocho Yunga

**DOCENTE DE LA TITULACIÓN**

De mi consideración:

El presente trabajo de titulación: **Implementación de una base de datos NoSQL en ambientes distribuidos**, realizado por Jiménez Esparza Christian Fabián, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, Octubre de 2016

f) .....

## DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo Jiménez Esparza Christian Fabián declaro ser autor (a) del presente trabajo de titulación: **Implementación de una base de datos NoSQL en ambientes distribuidos**, de la Titulación Sistemas Informáticos y Computación, siendo Juan Carlos Morocho Yunga director (a) del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f.....  
Autor Jiménez Esparza Christian Fabián.  
Cédula 1900816164

## DEDICATORIA

El presente trabajo va dedicado principalmente a mis queridos padres Yolanda y Ramiro pilares fundamentales, por sus consejos, amor, apoyo incondicional y palabras de aliento en cada etapa de mi vida; a mis hermanos Arelis y Mateo por sus bromas y alegría; a mi apreciado abuelo el “papito” José por sus oraciones y su cariño infinito; y por último a mi segunda madre que estoy seguro que en lugar donde se encuentre me cuida y protege, a mi inolvidable y entrañable abuelita “mamita” María por toda esa bondad y amor brindado.

A todas ellas, infinitas Gracias porque fueron mi principal fuente de inspiración para poder cumplir con este logro tan importante en mi vida.

***Christian***

## **AGRADECIMIENTO**

A Dios por guiarme en cada paso de mi vida con su bendición y haberme dado salud para cumplir con este objetivo.

A todos los docentes del área de Sistemas que fueron parte de mi formación académica universitaria, por compartir todos sus conocimientos y experiencias que permitieron formarme como profesional; así como también a la Universidad Técnica Particular de Loja por brindarme la oportunidad de estudiar y cumplir con este logro muy importante en vida personal.

Finalmente, al Ing. Juan Carlos Morocho Yunga por tomarme en cuenta para desarrollar el presente proyecto, por su asesoramiento en el desarrollo del mismo; así como también por brindarme todas las facilidades para poder cumplir con el trabajo en el laboratorio de Tecnologías avanzadas de la Web.

## TABLA DE CONTENIDOS

CARATULA .....	i
APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN.....	ii
DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS.....	iii
DEDICATORIA .....	iv
AGRADECIMIENTO .....	v
TABLA DE CONTENIDOS.....	vi
ÍNDICE DE FIGURAS.....	ix
ÍNDICE DE TABLAS.....	xi
RESUMEN.....	1
ABSTRACT .....	2
INTRODUCCIÓN.....	3
CAPÍTULO I	
VISIONAMIENTO .....	5
1.1 Problemática .....	6
1.2 Alcance .....	6
1.3 Objetivos. ....	7
1.3.1 General.....	7
1.3.2 Específicos. ....	7
1.4 Introducción a las Bases de datos NoSQL .....	7
1.4.1 Teorema de CAP (Consistencia, Disponibilidad y Tolerancia).....	7
1.4.2 Tipos de Bases de datos NoSQL .....	9
1.5 Beneficios.....	12
1.6 Terminología Básica.....	12
CAPÍTULO II:	
ANÁLISIS DE SISTEMAS NOSQL EN ENTORNOS DISTRIBUIDOS .....	14
2.1 Estudios sobre comparativas entre bases de datos NoSQL. ....	15
2.2 Ámbito de la solución .....	20
2.2.1 Criterios No seleccionados.....	20
2.2.2 Criterios seleccionados para elección de Sistemas NoSQL candidatos. ....	21
2.2.3 Bases de datos NoSQL candidatas.....	23
2.2.4 Alternativas de Solución para selección de Sistemas NoSQL a implementar..	23

2.2.4.1 Análisis de criterios para elección de Base de datos. ....	24
CAPÍTULO III:	
DESARROLLO DE LA SOLUCIÓN.....	26
3.1 Ciclo de desarrollo de la implementación de las Bases de datos NoSQL en entornos distribuidos.....	27
3.1.1 Elección de Tipo de Base de datos NoSQL .....	28
3.1.2 Elección de Bases de datos.....	29
3.1.2.1 MongoDB.....	29
3.1.2.2 CouchDB.....	31
3.1.2.3 RethinkDB.....	31
3.1.3 Adaptar modelo relacional a NoSQL:.....	32
3.1.4 Definición de arquitectura .....	34
3.1.4.1 Selección de Software.....	34
3.1.4.2 Selección de Hardware .....	35
3.1.4.3 Comunicaciones.....	36
3.1.4.3.1 Red en anillo. ....	36
3.1.4.3.2 Red en árbol.....	37
3.1.4.3.3 Red en malla. ....	38
3.1.4.3.4 Red en bus.....	38
3.1.4.3.5 Red en estrella. ....	39
3.1.4.3.6 Elección y diseño de Topología a implementarse.....	40
3.1.5 Configuración e Implementación del Entorno Distribuido .....	43
3.1.5.1 Fragmentación en MongoDB.....	43
3.1.5.2 Fragmentación en RethinkDB.....	45
3.1.5.3 Replicación en CouchDB.....	47
3.1.6 Importación de datos a los Sistemas NoSQL elegidos:.....	50
3.2 Pruebas Piloto .....	53
3.2.1 Resultados previos .....	54



CAPÍTULO IV:	
REFINAMIENTO E IMPLEMENTACIÓN FINAL.....	57
4.1 Optimización en MongoDB.....	58
4.2 Optimización en RethinkDB.....	60
4.3 Optimización en CouchDB .....	61
4.4 Recopilación de Resultados. ....	62
4.5 Elección Final.....	63
TRABAJOS A FUTURO.....	66
CONCLUSIONES.....	67
RECOMENDACIONES.....	69
BIBLIOGRAFÍA.....	71
ANEXOS.....	73
ANEXO A. Comparativa General de Bases de datos NoSQL.....	74
ANEXO B. Instalación de mongodb en Mac OS X.....	78
ANEXO C. Instalación de Mongodb en Ubuntu .....	79
ANEXO D. Instalación de couchdb en Mac OS X.....	80
ANEXO E. Instalación couchdb en ubuntu .....	81
ANEXO F. Instalación de rethinkdb en Mac OS X.....	82
ANEXO G. Instalación rethinkdb en ubuntu.....	83
ANEXO H. Transformación de datos Relacional a No Relacional.....	84
ANEXO I. Configuración de fragmentación de datos en MongoDB. ....	89

## ÍNDICE DE FIGURAS

Figura 1. Teorema de CAP .....	9
Figura 2. Bases de datos Clave-Valor.....	10
Figura 3. Bases de Datos Documentales .....	11
Figura 4. Bases de datos de Columnas .....	11
Figura 5. Bases de datos de Grafos.....	12
Figura 6. Ciclo de Desarrollo del proyecto. ....	28
Figura 7. Documentos en MongoDB. ....	30
Figura 8. Colección en MongoDB .....	30
Figura 9. Comparación de almacenamiento de datos Relacional y NoSQL .....	33
Figura 10. Topología en anillo.....	37
Figura 11. Topología en árbol .....	37
Figura 12. Topología en malla. ....	38
Figura 13. Topología en bus. ....	39
Figura 14. Topología en estrella. ....	39
Figura 15. Topología a implementar en MongoDB.....	41
Figura 16. Topología a implementar en RethinkDB.....	42
Figura 17. Topología a implementar en CouchDB .....	43
Figura 18. Arquitectura Fragmentación en MongoDB .....	44
Figura 19. Creación de directorio en RethinkDB .....	46
Figura 20. Inicio Servidor Principal.....	46
Figura 21. Inicio RethinkDB en el segundo servidor.....	46
Figura 22. Inicio RethinkDB en el tercer servidor. ....	47
Figura 23. Verificación de conexión de nodos en RethinkDB .....	47
Figura 24. Inicio Servidor principal CouchDB.....	48
Figura 25. Inicio primer servidor secundario. ....	48
Figura 26. Inicio segundo servidor secundario .....	49
Figura 27. Replicación al Segundo servidor .....	49
Figura 28. Replicación al Tercer Servidor .....	49
Figura 29. Carga de Datos en MongoDB-Inicio.....	50
Figura 30. Carga de Datos en MongoDB-Finalización. ....	50
Figura 31. Distribución de datos entre los servidores de MongoDB .....	51
Figura 32. Importación de datos en RethinkDB.....	51
Figura 33. Distribución de datos entre los servidores de RethinkDB. ....	52
Figura 34. Importación de datos en CouchDB .....	53
Figura 35. Carga Total en CouchDB .....	53
Figura 36. Primera consulta en MongoDB .....	54
Figura 37. Segunda consulta en MongoDB.....	55
Figura 38. Primera consulta en RethinkDB .....	55
Figura 39. Segunda consulta en RethinkDB. ....	56
Figura 40. Optimización de importación de datos-MongoDB.....	59
Figura 41. Finalización de importación de datos optimizada-MongoDB. ....	59
Figura 42. Mejora de Tiempo en la Importación de datos en MongoDB.....	60

Figura 43. Optimización de carga de datos-RethinkDB.....	60
Figura 44. Mejora de Tiempo en la Importación de datos en RethinkDB.....	61
Figura 45. Configuración de mejora en el compresor de CouchDB.....	62
Figura 46. Mejora en el Tamaño de Base de datos en CouchDB.....	62
Figura 47. Comparación del Tamaño de la base de datos en cada Sistema NoSQL. ....	64
Figura 48. Comparación del Tiempo de importación en cada Base de datos.....	64

## ÍNDICE DE TABLAS

Tabla 1. Comparación de BD relacional con NoSQL.....	16
Tabla 2. Comparación de Bases de datos NoSQL 1 .....	17
Tabla 3. Comparación de Bases de datos NoSQL 2.....	18
Tabla 4. Comparación Bases de datos NoSQL 3.....	19
Tabla 5. Comparación Bases de datos NoSQL 4.....	20
Tabla 6. Comparación de términos en MongoDB.....	32
Tabla 7. Comparación de términos en CouchDB .....	33
Tabla 8. Comparación de términos en RethinkDB .....	33
Tabla 9. Tabla de resultados previos. ....	54
Tabla 10. Resultados de Recuperación de datos.....	56
Tabla 11. Optimización de Pruebas Piloto. ....	63

## RESUMEN

El presente Trabajo de Titulación describe los procedimientos de configuración para lograr la implementación de Bases de datos NoSQL en entornos distribuidos, que apoye la escalabilidad y soporte de grandes cantidades de información (big data), para ello se realiza la selección de tres sistemas que resultan de una definición de criterios basados en los requerimientos de la investigación y a la infraestructura disponible en el laboratorio de Tecnologías Avanzadas de la Web de la Universidad Técnica Particular de Loja; las características de mayor influencia para la elección son el soporte de las funcionalidades fragmentación y replicación. Finalmente se analiza los resultados con el propósito de recomendar la base de datos idónea para su uso en entornos de producción.

**PALABRAS CLAVES:** NoSQL, fragmentación, replicación, big data, escalabilidad.

## **ABSTRACT**

This project describes configuration procedures to achieve the implementation of NoSQL databases in distributed environments, supporting scalability and large amounts of information, for this are chosen three systems resulting from the definition of criteria based on the requirements of the research and infrastructure available in the laboratory of Advanced Technologies Web of the Technical University of Loja, the characteristics of greater influence for the election are the support of the fragmentation and replication features. Finally the results are analyzed to recommend suitable database for use in production environments.

**KEYWORDS:** NoSQL, fragmentation, replication, big data, scalability.

## INTRODUCCIÓN

En la actualidad la mayoría de Sistemas utilizan algún tipo de base de datos relacionales para su funcionamiento y gestión de la información, en los sistemas relacionales dichos datos se encuentran almacenados en tablas y la mayoría de ellas son manipuladas por medio del lenguaje SQL, gracias a este tipo de base de datos, los sistemas garantizan que su información este bien estructurada y mantengan la integridad y consistencia en los datos. Pero desde el surgimiento de la web, los datos comienzan a aumentar y con ello se generan grandes cantidades de información donde los sistemas relacionales, tienen muchas limitaciones al momento de procesar las consultas, incrementando los tiempos de respuesta.

Con la llegada de la web surgen nuevas herramientas de base de datos conocidas como no relacionales o NoSQL, estas poseen mucha más capacidad de almacenamiento y su principal ventaja frente a las relacionales es que ofrecen escalabilidad horizontal, lo que significa incluir varios servidores para que la información sea procesada eficientemente, otra característica a destacar de las bases de datos no relacionales es su versatilidad debido a que no necesitan tener una estructura definida para el almacenamiento de los datos, por lo que no se requiere tener creadas previamente tablas como ocurre en los sistemas relacionales, sino que crean automáticamente la estructura( tablas, colecciones, etc) para su almacenamiento lo que facilita su flexibilidad y adaptabilidad.

Las Bases de datos SQL normalmente están implementadas en un solo servidor, lo cual es una desventaja en términos de escalabilidad afectando al rendimiento de los mismos, ante esta necesidad nace esta iniciativa de desarrollar el presente trabajo de titulación que consiste en la Implementación de una base de datos no relacional en ambientes distribuidos.

La metodología que se aplica en el proyecto, adopta un enfoque iterativo e incremental. Para ello se establecen hitos y entregables que son revisadas y supervisadas por el director de trabajo de titulación.

El presente trabajo consta de 5 secciones para su desarrollo: En el primer capítulo se presenta el visionamiento del proyecto, esto con el fin de entender el problema y así plantear

la solución, después se indica una breve introducción sobre las bases de datos NoSQL para conocer sus inicios y propósitos, además de proponer los beneficios que trae consigo definir el alcance y los objetivos, por último se muestra algunos términos relevantes para una mejor comprensión del proyecto.

El segundo capítulo se enfoca en trabajos relacionados que sean de interés para el desarrollo del trabajo de titulación; se realiza un análisis de los diferentes productos de bases de datos NoSQL que permita la implementación de datos en un entorno distribuido.

El tercer capítulo corresponde al desarrollo de la solución, para ello, se propone un conjunto de pasos que demuestra todo el proceso realizado en la ejecución del trabajo, ahí se define la arquitectura, que está compuesta por las bases de datos elegidas, los servidores en donde se instala dichas bases de datos y la comunicaciones que se refiere a la conexión entre equipos para lograr la transmisión de los datos, por último, se presenta las pruebas obtenidas en el desarrollo para verificar si el funcionamiento obtenido cumple con el objetivo principal del proyecto.

El cuarto capítulo se refiere a la identificación de errores y dificultades generadas en el tercer capítulo y se procede a corregir dichos problemas, este proceso se conoce como refinamiento y permite realizar la implementación final de la solución del proyecto.

Finalmente la última sección corresponde al desarrollo de conclusiones y recomendaciones en base a los objetivos del trabajo, además se propone trabajos que pueden desarrollarse a futuro que sean un aporte al desarrollo de bases de datos NoSQL.



**CAPÍTULO I**  
**VISIONAMIENTO**

La descripción del problema comprende los aspectos más relevantes que permitan entender el Trabajo de Titulación a solucionar, para ello, se presenta una breve introducción a las bases de datos NoSQL y al teorema de CAP creado para explicar que no es posible garantizar a la vez las características de Consistencia, Disponibilidad y Tolerancia a la partición en sistemas distribuidos.

En el apartado 1.1 se indica la problemática, que consiste en conocer cómo se desarrolla actualmente la implementación de base de datos NoSQL, identificar las desventajas del mismo y plantear la respuesta que dé solución al problema; en el apartado 1.2 se plantea el alcance que define y delimita hasta donde abarca el desarrollo del presente trabajo; así mismo, en el apartado 1.3 se propone los objetivos que se deben lograr; en el apartado 1.4 se indica los inicios de los sistemas NoSQL, el significado del teorema de CAP en los sistemas distribuidos y los tipos de bases de datos NoSQL existentes; en el apartado 1.5 se indican los beneficios que son las ventajas y aspectos positivos de desarrollar el proyecto; y finalmente, el apartado 1.6 explica algunos términos que ayuden al lector a un mejor entendimiento del proyecto.

## **1.1 Problemática**

Las bases de datos denominadas NoSQL, están diseñadas para almacenamiento y recuperación de datos a gran escala y en formatos distintos a las bases de datos relacionales. Debido al incremento exponencial de la cantidad de información sea esta estructurada, semiestructurada o no estructurada, se requiere diseñar y probar una infraestructura que sea escalable y por lo tanto su potenciamiento sea lo más transparentemente posible. Generalmente los productos NoSQL, en su instalación estándar lo hacen simplemente sobre una máquina, pero lo que se pretende con el presente trabajo es definir el procedimiento para ejecutar una instalación en un ambiente distribuido, dicho procedimiento deberá ser ampliamente probado, para asegurar la disponibilidad y rendimiento de este tipo de bases de datos.

## **1.2 Alcance**

Analizar varias Bases de datos NoSQL y escoger dos que soporten ambientes distribuidos para el almacenamiento de información y realizar su implementación, apoyando la

escalabilidad y se pueda reutilizar con la ayuda de un proceso de instalación y configuración.

### **1.3 Objetivos.**

#### **1.3.1 General.**

Definir un procedimiento para instalación de una base de datos NoSQL, en un ambiente distribuido, que asegure escalabilidad.

#### **1.3.2 Específicos.**

- ✓ Analizar los productos (NoSQL) más destacados en base a los requerimientos del proyecto para realizar su implementación en entornos distribuidos.
- ✓ Definir criterios de evaluación en base a las características más relevantes de los productos seleccionados para recomendar uno en entornos de producción.
- ✓ Evaluar el rendimiento de los productos para seleccionar el o los más indicados de acuerdo a los criterios definidos para el proyecto.

### **1.4 Introducción a las Bases de datos NoSQL**

#### **1.4.1 Teorema de CAP (Consistencia, Disponibilidad y Tolerancia)**

Las bases de datos NoSQL aparecen como una solución a los problemas de escalabilidad que se presentan en los sistemas relacionales; desde el surgimiento de la web 2.0 aparecen las redes sociales, en dichas aplicaciones cada usuario puede generar su propio contenido y cargarla al sistema, provocando la generación de grandes cantidades de información, términos que se asocian al denominado BigData, el cual (Rodríguez, 2016) lo considera como: “La manipulación de grandes cantidades de información, lo que implica una alta demanda de hardware e infraestructura tecnológica”, por lo cual, gestionar dichos datos con un sistema relacional provocaría que las aplicaciones disminuyan su eficiencia y rendimiento. Las bases de datos NoSQL aparecen como una nueva alternativa de

almacenar y gestionar información y en la actualidad están acaparando la atención de muchas organizaciones como menciona (Zambrano, 2015) “Las nuevas tecnologías de bases de datos como NoSQL son un importante factor para conseguir que las empresas puedan anticiparse a las necesidades del negocio y construir estas nuevas capacidades en sus infraestructuras de datos” .

La principal premisa de las bases de datos NoSQL es la escalabilidad y la capacidad de aplicarse en varios servidores, según (Fernandez, 2014) al ser los sistemas NoSQL distribuidas es importante conocer el teorema de CAP propuesto por (Brewer, 2000) el cual afirma: “Un sistema de datos compartidos puede asegurar como mucho dos de estas tres propiedades: Consistencia, Disponibilidad y Tolerancia a particiones”.

Las características de este teorema se describen a continuación:

- **Consistencia:** Hace referencia a la capacidad de mantener la integridad de los datos en un sistema distribuido, en donde todos los nodos deben de tener la misma información y si existen actualizaciones, deben reflejarse dichos cambios automáticamente en todos los servidores.
- **Disponibilidad:** Garantiza que las solicitudes realizadas por los usuarios siempre obtengan respuesta por parte del sistema.
- **Tolerancia (Particiones):** Se relaciona con la característica anterior, ya que, si un servidor deja de funcionar, no debe de afectar al resto de nodos y cualquier petición del usuario debe de procesarse sin ningún inconveniente.



Figura 1. Teorema de CAP

Fuente: <http://www.rodenas.org/ferdyblog/2011/02/25/el-teorema-de-cap/>

Elaboración: Ferran Rodenas

La Figura 1 muestra las posibles combinaciones que pueden realizarse en un entorno distribuido, tomando en cuenta que solo dos características pueden relacionarse, a continuación se explica cada una de las combinaciones:

**CP:** Un entorno distribuido puede ser consistente y tolerante, manteniendo la integridad de los datos en todas las particiones que forman parte del sistema.

**AP:** Garantiza que el sistema siempre esté disponible para recibir y contestar las peticiones de usuarios y tolerante a fallos ante cualquier problema que afecte al entorno distribuido.

**CA:** Proporciona un sistema en donde la información sea coherente en todos los nodos y siempre estén disponibles, de tal manera que siempre procese las peticiones de usuarios.

#### 1.4.2 Tipos de Bases de datos NoSQL

El almacenamiento en las bases de datos NoSQL es distinto a los sistemas transaccionales, ya que no utilizan modelo entidad-relación y no presentan una estructura fija como ocurre en las bases de datos relacionales donde se crean tablas. Los formatos para el almacenamiento de datos en sistemas NoSQL son:

- **Clave- Valor:**

Es el formato más sencillo, formado por una colección compuesta por dos pares: clave y valor, lo que permite acceder a los datos de manera más rápida y eficiente. La Figura 2 es la representación gráfica del formato clave-valor.

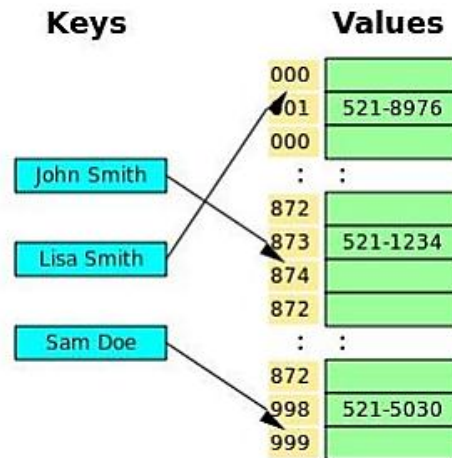


Figura 2. Bases de datos Clave-Valor

Fuente: <http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>

Elaboración: Acens

- **Documentales:**

Almacena la información en documentos que contienen colecciones de clave-valor, en cada documento se guardan datos sobre un objeto específico, esta información puede soportar los siguientes formatos XML<sup>1</sup>, JSON<sup>2</sup> o BSON<sup>3</sup>, los almacenes de documentos son el tipo de base de datos NoSQL más versátiles debido a que no necesitan tener una estructura definida para el almacenamiento de los datos. La Figura 3 muestra la representación gráfica de las Bases de datos NoSQL Documentales.

<sup>1</sup> Extensible Markup Language (XML): <https://www.w3.org/XML/>.

<sup>2</sup> Formato JSON: <http://www.w3schools.com/json/>.

<sup>3</sup> Formato BSON: <http://bsonspec.org/>.

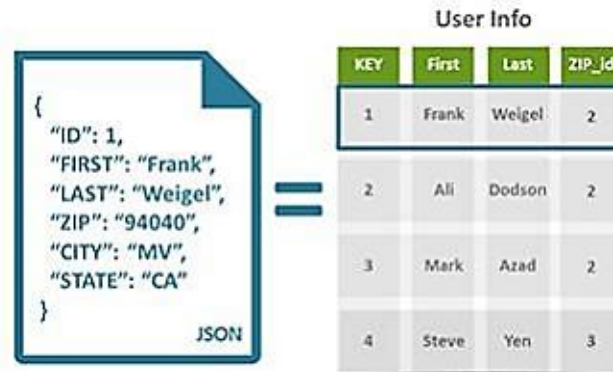


Figura 3. Bases de Datos Documentales

Fuente <http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>

Elaboración: Acens

- **Columnas:**

Presentan una estructura similar a la de las bases de datos relacionales, pero guardan su información en columnas, es decir, según (Garcete, 2014) “Todos los casos de un solo elemento de datos se almacenan de modo que se puede acceder como una unidad”. La Figura 4 muestra la representación gráfica de las Bases de datos de columnas.



Figura 4. Bases de datos de Columnas

Fuente: <http://jeuazarru.com/wp-content/uploads/2014/10/dbco.pdf>

Elaboración: Adrián Garcete.

- **Grafos:**

La información está constituida por nodos y aristas, el mejor representante para ese tipo de base de datos es el modelo RDF, el cual se encuentra formado por tripletas compuesto por sujeto, predicado y objeto, según (P. De la Torre, 2010) “el sujeto es el elemento sobre el que se realiza la descripción, el predicado es la propiedad descrita de dicho elemento y el objeto es el valor de esta propiedad”. La Figura 5 muestra la representación gráfica de las bases de datos de Grafos.

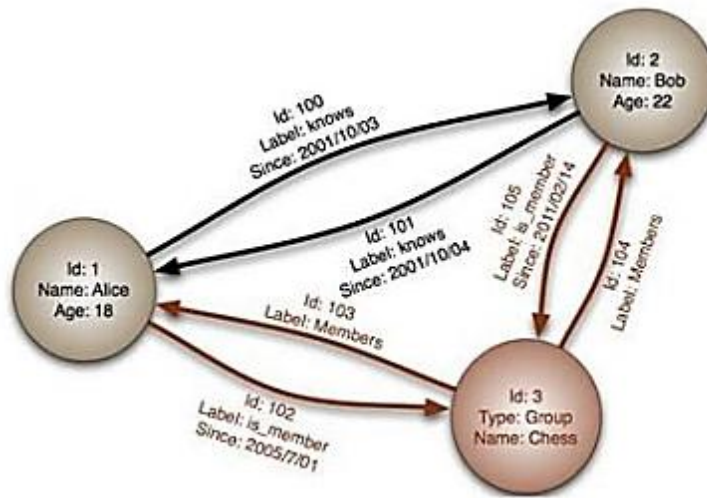


Figura 5. Bases de datos de Grafos

Fuente: <http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>

Elaboración: Acens

## 1.5 Beneficios

El almacenamiento de información en una base de datos NoSQL en ambientes distribuidos trae consigo una serie de beneficios:

- Permite mejorar la disponibilidad, de esta forma si un servidor no está disponible es posible acceder a los datos a partir de las copias almacenadas en otros servidores.
- Mejorar el rendimiento de los sistemas NoSQL y mayor capacidad de almacenamiento.
- Tiempos de respuesta mejorados en lo que respecta a la consulta de datos.
- Transparencia sencilla en la Implementación.

## 1.6 Terminología Básica

En este apartado se indican los conceptos básicos relacionados al Trabajo de Titulación a desarrollar, los cuáles se utilizan a lo largo de todo el documento para obtener una



mejor captación y comprensión de los temas planteados.

- **NoSQL:** Expresión asociada a las bases de datos para indicar que son sistemas No relacionales.
- **Document Store:** Modelo de base de datos que soporta las para el almacenamiento de la información como un documento, utilizando distintas estructuras como XML y JSON.
- **Escalabilidad:** Capacidad que tienen las Bases de datos de mantener su calidad y funcionamiento, a pesar del crecimiento de los datos.
- **Ambiente distribuido:** Sistema conformado por un conjunto de equipos hardware separados físicamente pero conectadas entre sí y que transfieren información de un lugar a otro.
- **Ambiente monolítico:** Sistema conformado por un único equipo hardware que es el encargado de recibir, procesar y transferir información.

El siguiente capítulo se involucra en el análisis de diferentes estudios realizados, en donde sobresalen las características más importantes de las bases de datos NoSQL, las mismas que permitan construir una comparativa general con los criterios considerados más relevantes para el desarrollo del Trabajo de Titulación y así elegir los sistemas ideales, los cuáles se utilizan a lo largo del presente trabajo.

**CAPÍTULO II:**  
**ANÁLISIS DE SISTEMAS NOSQL EN ENTORNOS DISTRIBUIDOS**

El presente capítulo tiene como objetivo indicar varios estudios realizados acerca de los diferentes sistemas de almacenamiento NoSQL con sus características más importantes, para luego realizar el análisis respectivo y determinar una comparativa general con los criterios más relevantes que ayuden con el desarrollo del Trabajo de Titulación, explicando las razones por las cuáles dichos criterios fueron seleccionados.

Partiendo de los criterios indicados en la comparativa general, se procede a seleccionar los posibles candidatos de las bases de datos que cumplan con todas las características propuestas. Finalmente será necesario realizar un último filtro con el objetivo de seleccionar de los posibles candidatos, cuáles Sistemas NoSQL cumplen con los criterios de soportar su implementación en entornos distribuidos con licencia libre y si es multiplataforma en cuanto a los sistemas operativos.

## **2.1 Estudios sobre comparativas entre bases de datos NoSQL.**

A continuación, indicaremos algunas comparativas realizadas en diferentes estudios en donde se resaltan las características más importantes de los sistemas NoSQL:

El primer estudio se trata de realizar una breve comparación entre los sistemas de bases de datos relacionales y no relacionales, para ello se considera utilizar como referencia la comparativa propuesta por (Salazar, 2014) en la Tabla 1, en donde menciona algunas características, como: el modelo de almacenamiento, el esquema que utilizan, el escalamiento que soportan, como es el rendimiento en la tolerancia a fallos y finalmente, algunos ejemplos de Sistemas de Base de datos.

Tabla 1. Comparación de BD relacional con NoSQL

	Relacional	NoSQL
Modelo de almacenamiento	Almacenamiento con filas y tablas	Varía dependiendo del tipo de base de datos NoSQL. Clave-valor, Documento
Esquema	Tipo estructura de datos, se fijan por adelantado.	Dinámico.
Escalado	Solo vertical, o sea con equipos hardware con características más poderosas.	Horizontalmente, para aumentar la capacidad se deben añadir más nodos al servidor, y a la base de datos se propaga automáticamente
Tolerancia a fallos	Bajo. Fallo en el nodo y generalmente hará fallar la consulta.	Alta. Configurados para que la pérdida de algunos nodos no interrumpa funcionamiento global.
Ejemplos	Oracle, Mysql, SqlServer	Mongodb, MarkLogic, Hbase

Fuente: <http://repositorio.utp.edu.co/dspace/bitstream/11059/5119/1/0057565S161.pdf>

Elaboración: José Salazar

El siguiente estudio presentado por (Moniruzzaman & Akhter, 2013) en la Tabla 2 resalta las bases de datos NoSQL más importantes para el análisis y soporte de grandes cantidades de información, en dicha comparación resaltan las principales características de cada sistema NoSQL, en base a los siguientes atributos: Diseño, Sistema y Distribución.

Tabla 2. Comparación de Bases de datos NoSQL 1

Atributos	Diseño		Distribución		Sistema	
Base de datos	Modelo	Métodos de Acceso	Escalabilidad Horizontal	Fragmentación de datos	Sistemas Operativos	Licencia
MongoDB	Documentos	API propietario o usando JSON	Si	Si	Windows, Ubuntu, Mac	Open Source
CouchDB	Documentos	RestFull, API JSON	Si	No	Ubuntu, Red Hat, Windows, Mac	Open Source
Neo4J	Grafos	Lenguaje de consulta Cypher y Restful	No	Si	Windows, Ubuntu, Mac	Open Source
Redis	Clave-Valor	API propietario o RESP	No	No	Ubuntu, Mac, Windows, NIX	Open Source
RethinkDB	Documentos	No	Si	Si	Ubuntu, Mac	Open Source

Fuente: <http://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf>

Elaboración: Syed Akhter

Tomando en cuenta la comparativa realizada por (Shankar, 2014) presentada en la tabla 3, se encuentra que los criterios considerados para realizar la comparación son: Desarrollador, Tipo de Almacenamiento, características y licencia. El objetivo de dicha comparación, es dar a conocer las diferentes opciones que existen en la actualidad en lo que respecta a bases de datos NoSQL con sus características básicas.

Tabla 3. Comparación de Bases de datos NoSQL 2

Base de datos	Propietario	Tipo de Almacenamiento	Características	Licencia
MongoDB	10gen	Documentos	<ul style="list-style-type: none"> <li>- Consistencia</li> <li>- Tolerancia a Particiones</li> <li>- Persistente</li> </ul>	Libre
CouchDB	Apache	Documentos	<ul style="list-style-type: none"> <li>- Alta Disponibilidad</li> <li>- Tolerancia a Partición</li> <li>- Persistencia</li> </ul>	Libre
Cassandra	Apache	Columnas	<ul style="list-style-type: none"> <li>- Alta Disponibilidad</li> <li>- Tolerancia a Partición</li> <li>- Persistencia</li> </ul>	Libre
RethinkDB	RethinkDB	Documentos	<ul style="list-style-type: none"> <li>- Consistencia</li> <li>- Alta Disponibilidad</li> <li>- Persistencia</li> </ul>	Libre
Riak	Basho Technologies	Clave-Valor	<ul style="list-style-type: none"> <li>- Alta Disponibilidad</li> <li>- Tolerancia a Partición</li> <li>- Persistencia</li> </ul>	Libre

Fuente: <http://www.infoivy.com/2014/05/nosql-databases-mongodb-simpledb-etc.html>

Elaboración: Shankar Sahai

DB-Engines, una organización encargada de reunir y presentar información sobre los sistemas de gestión de base de datos relaciones y NoSQL (DB-Engines, 2015), presenta estudios mensuales basados en la popularidad del sistema de base de datos, determinada por las búsquedas frecuentes en Google Trends, relevancia en las redes sociales, número de perfiles profesionales, donde se menciona dicho sistema y en el número de referencias en los buscadores Google y Bing. En la tabla 4 se muestra el estudio correspondiente al mes de Julio del 2015 con los siguientes criterios: Ranking, Replicación, Licencia, Carga Masiva, Sitio Web, Modelo de Base de datos y Sistemas Operativos.

Tabla 4. Comparación Bases de datos NoSQL 3

Base de datos	Ranking	Replicación	Licencia	Carga Masiva	Sitio Web	Modelo de base de datos	Sistemas Operativos
MarkLogic	1	Si	Comercial y libre (Por 6 meses)	Si	www.marklogic.com	Multi-modelo (Document store, RDF store, Search engine)	Linux OS X Solaris Windows
Virtuoso	2	Si	Comercial y Libre	Si	virtuoso.openlinksw.com	Multi-modelo (Native XML, Relational DBMS, RDF store)	FreeBSD HP-UX Linux OS X Solaris Windows
AllegroGraph	5	Si	Comercial	Si	www.franz.com/-agraph/-allegrograph	RDF Store	Linux Windows OS X
MongoDB	1	Si	Open Source	Si	www.mongodb.org	Document store	Linux OS X Solaris Windows

Fuente: <http://db-engines.com/en/ranking>  
 Elaborado: DB-Engines

El último estudio analizado se trata de los productos que soportan el modelo de base de datos documentales, en el trabajo realizado por (Vazquez & Sotolongo, 2013) describen a tres de sus representantes como son: MongoDB, CouchDB y Terrastore, dichos productos han sido seleccionados por ser bases de datos de código abierto y por su popularidad en lo que respecta a su modelo basado en documentos. Cada uno de ellos, están descritos identificando sus principales características. Para el desarrollo del estudio se considera los siguientes criterios: Leguajes de Programación, tipo de Replicación que soporta, recuperación ante fallos y protocolo de acceso y se presenta en la tabla 5.

Tabla 5. Comparación Bases de datos NoSQL 4

Nombre	Lenguaje	Tipo de Replicación que soporta	Recuperación ante fallos	Protocolo de Acceso
MongoDB	C C# Cold Fusion Erlang Haskell Java JavaScript Lisp PHP Python Ruby	Maestro-Eslavo	Si	TCP/IP
CouchDB	C y C# Java JavaScript Perl PHP Python Ruby Smalltalk	Maestro-Maestro	Si	HTTP
Terrastore	Java	Maestro-Maestro	Si	HTTP

Fuente: <http://publicaciones.uci.cu/index.php/SC/article/view/1382/720>

Elaborado: Yudisney Vazquez

## 2.2 Ámbito de la solución

Una vez analizados los estudios anteriores, se indica el ámbito de la solución; que tiene por objetivo la elección de las posibles bases de datos candidatas, para ello, en el apartado 2.2.1 se descartan los criterios que no fueron tomados en cuenta, después, en la sección 2.2.2 se muestra la selección y descripción de aquellos criterios que se consideran relevantes y que permitan cumplir con el objetivo principal del Trabajo de Titulación.

### 2.2.1 Criterios No seleccionados.

Las características de las bases de datos NoSQL que no fueron seleccionadas se debe, a que se consideran criterios básicos que debe de tener todo sistema NoSQL y que no son relevantes para el desarrollo de trabajo, a continuación se muestra los criterios eliminados:

- Carga Masiva
- Esquema



- Modelo de almacenamiento
- Métodos de acceso
- Escalabilidad horizontal
- Características Principales
- Desarrollador
- Recuperación ante fallos
- Protocolo de acceso.

### 2.2.2 Criterios seleccionados para elección de Sistemas NoSQL candidatos.

En el anexo A se muestra la comparativa general que representa los criterios seleccionados, la misma que permite realizar la elección de los sistemas considerados como posibles candidatos. A continuación se indican las características elegidas, estas describen su significado y la razón por la que fue tomado en cuenta para que sea elegido como objeto de estudio en nuestro trabajo.

- **Nombre:** Hace referencia al nombre del producto de base de datos.
- **Ranking:** Indica la posición en que se encuentra actualmente el producto en una categoría dada. Este criterio fue tomado en cuenta, debido a la importancia que incide la preferencia de los usuarios en la utilización de dicho producto. Las categorías consideradas para el ranking son los modelos: RDF Store, Document Store y Key Value Store.
- **Licencia:** Describe el acuerdo existente entre el licenciatarlo y los usuarios finales, para poder utilizar dicho producto bajo una serie de condiciones estipuladas dentro de sus cláusulas. La licencia es un aspecto importante para el desarrollo del presente trabajo, en base a los requerimientos del proyecto es preferible seleccionar los sistemas que sean gratuitas o libres.
- **Costo:** Indica el precio para adquirir las licencias de los sistemas de almacenamiento que tengan disponible una versión comercial.

- **Soporte:** Muestra el sitio oficial de la base de datos donde se puede tener acceso a su documentación, lo que ayuda como guía para el desarrollo del Trabajo de Titulación.
- **Modelo de Base de datos:** Indica el tipo de modelo de almacenamiento que soporta. En el desarrollo del proyecto, este aspecto es muy importante, ya que el modelo elegido debe ser el más flexible a la hora de transformar de base de datos relacional a NoSQL.
- **Sistemas Operativos:** Aspecto a considerar debido a que describe los Sistemas Operativos que soporta cada base de datos para su implementación y funcionamiento.
- **API:** (Application Programming Interface) Indica que la base de datos ofrece un conjunto de métodos o funciones que puede ser reutilizado para el desarrollo de aplicaciones.
- **Integración con Lenguajes de Programación:** Criterio que describe los distintos lenguajes de programación que soporta cada Base de datos para el desarrollo de aplicaciones.
- **Tipo de Replicación que soporta:** Indica uno de los aspectos más importantes a considerar dentro de esta comparativa ya que permite cumplir con el objetivo principal del Trabajo de Titulación, la replicación implica crear copias de datos en varios servidores a través de los diferentes mecanismos que soportan los sistemas de almacenamiento para realizar la replicación, estos pueden ser: Maestro-Esclavo y Maestro-Maestro.
- **Soporte de Fragmentación:** Junto con la replicación este criterio es de gran importancia debido a que permite el desarrollo del presente trabajo, la fragmentación se trata de una funcionalidad que las Bases de datos NoSQL ofrecen para repartir los datos en distintos servidores y así asegurar la escalabilidad y rendimiento en los sistemas de almacenamiento.

Una vez descritos dichos criterios, se procede a realizar la comparativa general como se muestra en el anexo A, con el objetivo de que sea el punto de partida y se convierta en nuestra base para el desarrollo del Trabajo de Titulación, ya que de la siguiente tabla se eligen las bases de datos para realizar su implementación y cumplir con el propósito general del proyecto.

### **2.2.3 Bases de datos NoSQL candidatas.**

Partiendo de la comparativa general, se seleccionan los posibles sistemas de bases de datos que sean útiles en la implementación en ambientes distribuidos, para ello se toma en cuenta aquellos sistemas que cumplan con todos los criterios propuestos que se muestran en la Tabla 6. La lista de base de datos candidatas es la siguiente:

- ✓ MarkLogic<sup>4</sup>
- ✓ Virtuoso<sup>5</sup>
- ✓ AllegroGraph<sup>6</sup>
- ✓ MongoDB<sup>7</sup>
- ✓ CouchDB<sup>8</sup>
- ✓ Redis<sup>9</sup>
- ✓ RethinkDB<sup>10</sup>
- ✓ Riak<sup>11</sup>
- ✓ Cassandra<sup>12</sup>

### **2.2.4 Alternativas de Solución para selección de Sistemas NoSQL a implementar.**

Las alternativas de solución se refieren a realizar el análisis a cada Base de datos seleccionada en la sección 2.2.3 con el fin de definir los criterios considerados fundamentales que permitan la elección de las bases de datos a implementarse.

---

<sup>4</sup> Sitio Web Oficial de MarkLogic: <http://www.marklogic.com/>.

<sup>5</sup> Sitio Web Oficial de Virtuoso: <http://virtuoso.openlinksw.com/>.

<sup>6</sup> Sitio Web Oficial de AllegroGraph: <http://franz.com/agraph/allegrograph/>.

<sup>7</sup> Sitio Web Oficial de MongoDB: <https://www.mongodb.com/>.

<sup>8</sup> Sitio Web Oficial de CouchDB: <http://couchdb.apache.org/>.

<sup>9</sup> Sitio Web Oficial de Redis: <http://redis.io/>.

<sup>10</sup> Sitio Web Oficial de RethinkDB: <https://www.rethinkdb.com/>.

<sup>11</sup> Sitio Web Oficial de Riak: <http://basho.com/products/#riak>.

<sup>12</sup> Sitio Web Oficial de Cassandra: <http://cassandra.apache.org/>.

#### **2.2.4.1 Análisis de criterios para elección de Base de datos.**

Los sistemas de almacenamiento NoSQL son seleccionados tomando en cuenta los criterios que se consideren más relevantes para el propósito del Trabajo de Titulación y aquellos que se acoplen a las condiciones e infraestructura con la que cuenta actualmente la Universidad Técnica Particular de Loja, para la implementación del ambiente distribuido. Los datos que se utilizan para su almacenamiento e implementación en un entorno distribuido se encuentran originalmente localizadas en una Base de datos relacional MySQL y se requiere de un proceso de conversión al formato NoSQL como se explica en el apartado 3.1.3.

Tomando en cuenta estas consideraciones, los criterios que influyen en la toma de decisión final son:

- **La licencia tiene que ser libre:** Porque actualmente la base de datos proporcionada contiene gran cantidad de datos, por lo tanto, implementarla en un entorno distribuido con sistemas relacionales demanda gastos para la adquisición de licencias y equipos Hardware, en cambio sí se aprovecha las licencias gratuitas proporcionadas por la mayoría de Bases de datos NoSQL se evita dichos costos, además, no se requiere tener equipos con grandes capacidades a nivel de Hardware.
- **Sistema Operativos más importantes:** Actualmente en el Laboratorio de Tecnologías Avanzadas de la Web cuentan con infraestructura hardware con los siguientes SO: Ubuntu y Mac OS. La base de datos seleccionada tiene que soportar algunos de estos sistemas operativos.
- **Modelo de Base de datos orientado a Documentos:** Tiene que ver con el modelo de almacenamiento que se adapte a la transformación de Base de datos relacionales a NoSQL. La base de datos elegida debe tener la capacidad de soportar el modelo en documentos por las siguientes razones: No requiere definir una estructura para el almacenamiento de datos (Esquema flexible) que está formado por una clave y un valor; Permiten cargar archivos de dos tipos de formato: .js que es el principal y su estructura es igual al formato JSON , y de tipo .csv cuyo esquema

está formado por filas y columnas; Las Bases de datos documentales poseen gran cantidad de información disponible para su implementación y (Vazquez & Sotolongo, 2013) afirma: “Cada documento simula una fila de una tabla de los sistemas relacionales”, lo que las convierte en el tipo que más se asemeja a las bases de datos relacionales.

- **Capacidad de replicación o fragmentación:** La característica más importante que debe tener la base de datos debe ser el soporte ya sea de la funcionalidad de replicación o la capacidad de fragmentación (sharding), ya que ambas permiten cumplir con el propósito del Trabajo de Titulación, por lo cual, en relación con los otros tres criterios, este será el más influyente para la toma de decisión final.

Una vez que se realiza el análisis para la selección de los Sistemas NoSQL a implementar en la sección 2.2.4.1 donde se muestra los criterios finales a considerar y revisando las bases de datos indicadas en la sección 2.2.3, se puede determinar que los sistemas de almacenamiento elegidas son: MongoDB, CouchDB y RethinkDB que son descritas con mayor profundidad en el siguiente capítulo.

**CAPÍTULO III:  
DESARROLLO DE LA SOLUCIÓN**

En el presente capítulo se indica el ciclo el desarrollo, donde constan los pasos que han sido considerados para llevar acabo el trabajo, aquí se indica el proceso de transformación de datos relacionales a NoSQL, la definición de la arquitectura, la misma que está compuesta por tres elementos esenciales: Hardware, Software y las Comunicaciones, después se presenta la instalación y configuración de la arquitectura indicada, luego la implementación del ambiente distribuido para finalmente concluir con las pruebas piloto que son experimentos que permiten verificar si el funcionamiento obtenido cumple con los requerimientos del Trabajo de Titulación.

### **3.1 Ciclo de desarrollo de la implementación de las Bases de datos NoSQL en entornos distribuidos.**

Es el conjunto de pasos o fases que es utilizado como guía para el desarrollo del proyecto y medir su progreso, dichos pasos se definen junto con el Director de Trabajo de Titulación y se adaptan al modelo de desarrollo evolutivo, para ello se toma como referencia un fragmento del libro (Sommerville, 2005) en donde se lo describe como una metodología “basada en las peticiones del cliente para producir un sistema que satisfaga sus necesidades” (pág. 61), en él se requiere que la entrega del sistema sea iterativa e incremental debido a los cambios que se pueden presentar durante el desarrollo. Los incrementos están representados por cada actividad que se realice y es entregada al Director del Trabajo de Titulación para su revisión, y la iteración se refiere a que el modelo se puede repetir varias veces en caso de que algún proceso no cumpla con las expectativas, hasta lograr su aprobación. En la figura 6 se presenta el esquema del ciclo de desarrollo a seguir, para su implementación en entornos distribuidos.



Figura 6. Ciclo de Desarrollo del proyecto.  
Elaboración: Propia.

### 3.1.1 Elección de Tipo de Base de datos NoSQL

El primer paso para el desarrollo del Trabajo de Titulación consiste en la elección del tipo de Base de datos NoSQL que como ya se conoce se clasifican en cuatro tipos: Documentales, Clave-Valor, Gráficas y basados en columnas.

No existe un tipo de base de datos ideal para elegirlo como el indicado para migrar de SQL a NoSQL, Para realizar la elección del tipo de base de datos, se toma como referencia el documento presentado por (Rocha, Vale, Cirilo, Barbosa, & Mourão, 2015) el mismo que trata sobre el desarrollo de un framework para la migración de datos de una base de datos relacional a NoSQL (NsqliLayer), en esta investigación, el sistema elegido es MongoDB de tipo Documental por su alta disponibilidad, por la creación automática de colecciones, que representa a cada tabla del sistema relacional, lo que permite mantener su semántica y la facilidad de repartir los datos en distintas máquinas. Además, otros aspectos considerados



para la elección se realizan de acuerdo a los requerimientos del Trabajo de Titulación, tomando en cuenta el soporte de grandes cantidades de información y la facilidad de escalar la base de datos entre distintos equipos, a continuación se indican otras características que ayudan a realizar la elección:

- No requieren definir una estructura para el almacenamiento de datos (Esquema flexible).
- Un mismo documento puede contener a otro, formando subdocumentos que pueden ser de cualquier tipo de dato.
- Su estructura de datos está formada por un campo y un valor, igual al formato JSON.
- Permiten cargar archivos de formato js o csv.
- Gran cantidad de información disponible para su implementación.
- (Vazquez & Sotolongo, 2013) afirma: “Cada documento simula una fila de una tabla de los sistemas relacionales”, lo que las convierte en el tipo que más se asemeja a las bases de datos relacionales.

### **3.1.2 Elección de Bases de datos**

Tomando como referencia la lista de Bases de datos candidatas presentada en el apartado 2.2.3 y considerando los criterios de selección que se muestran en el análisis planteado en el apartado 2.2.4.1, se obtiene como resultado que las bases de datos elegidas para el desarrollo del Trabajo de Titulación son los siguientes sistemas: MongoDB. CouchDB y RethinkDB, dichos sistemas cumplen con las características fundamentales para la configuración en un entorno distribuido.

#### **3.1.2.1 MongoDB.**

Es una base de datos cuyo modelo está orientado a documentos BSON (JSON Binario), su esquema es flexible por lo que el usuario no necesita definir una estructura antes de cargar los datos.

Los documentos en MongoDB están formados por pares: clave y valor, en la clave se coloca el nombre del campo y en el valor puede ir: los datos de distintos tipos o también se pueden anidar otros documentos. En cada documento existe el campo “\_id” que permite identificar de manera única a un documento. En la Figura 7 se muestra la representación gráfica de

los documentos en MongoDB.

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

← field: value  
← field: value  
← field: value  
← field: value

Figura 7. Documentos en MongoDB.

Fuente: <https://docs.mongodb.org/manual/core/crud-introduction/>

Elaboración: MongoDB

Las colecciones son un conjunto de documentos y permiten almacenar varios de ellos, si la comparamos con las bases de datos relacionales, las colecciones equivalen a las tablas que se crean. En la Figura 8 se muestra la representación gráfica de las colecciones en MongoDB.

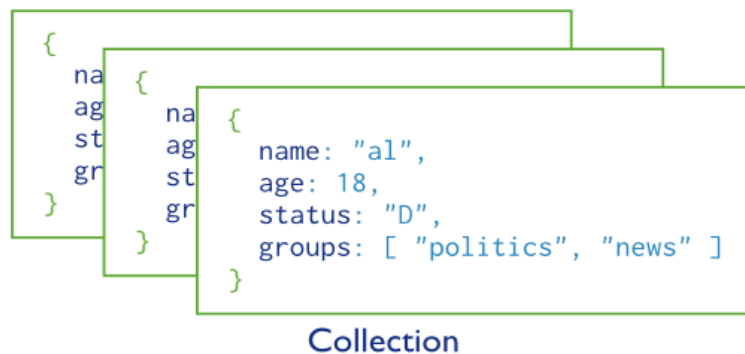


Figura 8. Colección en MongoDB

Fuente: <https://docs.mongodb.org/manual/core/crud-introduction/>

Elaboración: MongoDB

Entre las características más importantes destacamos su velocidad para realizar las operaciones CRUD (Crear, Leer, Actualizar y Borrar), la facilidad para su comprensión debido a su propio sistema de consulta, soporte de escalamiento horizontal, lo cual crean sistemas con eficiente rendimiento ya sea implementando las funcionalidades conocidas como fragmentación o replicación.

### **3.1.2.2 CouchDB.**

Apache CouchDB es una base de datos basada en documentos de licencia libre que soporta servicios REST<sup>13</sup> para tener acceso a su contenido, al igual que MongoDB no requiere definir una estructura para el almacenamiento de datos, además soporta archivos con formato JSON compuesto por pares: clave y valor. En cada documento siempre encontraremos estos campos: “\_id” y “\_rev” que se crean automáticamente, el primero nos ayuda a identificar unívocamente a un documento, mientras que el campo “\_rev” hace referencia a la versión de cada documento.

CouchDB soporta Vistas que sirven para la administración de los documentos, con las vistas podemos crear reportes de los mismos, también sirven para poder realizar las búsquedas de información en los documentos a través de lenguaje Javascript a través de funciones map/reduce, que permiten devolver pares: clave y valor.

El modelo de almacenamiento de CouchDB es como un árbol binario, que según (Hernández, 2009) “Es una estructura de datos ordenada que permite búsquedas, inserciones y eliminaciones en tiempo logarítmico”, lo que permite crear sistemas con gran velocidad, disponibilidad, rendimiento y escalabilidad a través de la replicación de datos en varios nodos.

### **3.1.2.3 RethinkDB.**

Es una base de datos de código libre que al igual que MongoDB y CouchDB está diseñada para el almacenamiento de datos documentales mediante el formato JSON. En lo que respecta a la administración de este sistema de almacenamiento, RethinkDB proporciona una interfaz gráfica web y también línea de comandos, los datos se guardan en tablas y no necesariamente debe de existir antes de realizar la carga, ya que se crean automáticamente, además cuenta con su propio lenguaje de consulta llamado Reql para la gestión y manipulación de información de documentos JSON.

Según su documentación oficial (RethinkDB, 2016) las características principales del lenguaje de consulta de rethinkdb son las siguientes:

---

<sup>13</sup> Concepto sobre servicios REST: <http://asiermarques.com/2013/conceptos-sobre-apis-rest/>.

- Las consultas se realizan por medio de un conjunto de sentencias que el sistema de almacenamiento ya reconoce.
- Cada consulta debe de ser ejecutada siguiendo una sintaxis o patrón, el punto es el principal separador de cada función. En las consultas realizadas por medio de la interfaz gráfica de la base de datos, se debe especificar en primer lugar el nombre de la base y en segundo lugar el nombre de la tabla, después se puede colocar todas las operaciones que se quiera realizar sobre los datos por medio de funciones ya predefinidas.
- Todas las consultas son compiladas y ejecutadas por medio de un servidor, en respuesta a las peticiones de los clientes.

### 3.1.3 Adaptar modelo relacional a NoSQL:

Proceso que se refiere a la conversión de los datos, del modelo Relacional al formato adecuado para el almacenamiento en los sistemas orientados a documentos.

Para comprender el proceso de conversión, es importante conocer la terminología utilizada en las bases de datos NoSQL de documentos, En las tablas 6, 7 y 8 se muestra la comparación de los términos empleados en los sistemas relacionales con cada una de las bases de datos documentales elegidas:

#### - En MongoDB

Tabla 6. Comparación de términos en MongoDB

Base de datos Relacional	MongoDB
Base de datos	Base de datos
Tabla	Colección
Filas	Documentos
Columnas	Campos(Clave-Valor)

Fuente: <https://www.mongodb.com/compare/mongodb-mysql>

Elaboración: MongoDB

- **En CouchDB**

Tabla 7. Comparación de términos en CouchDB

Base de datos Relacional	CouchDB
Base de datos	Base de datos
Fila	Documentos
Columnas	Campos (Clave: Valor)

Fuente: <http://couchdb.apache.org/>  
 Elaboración: CouchDB

- **En RethinkDB**

Tabla 8. Comparación de términos en RethinkDB

Base de datos Relacional	RethinkDB
Base de datos	Base de datos
Tabla	Tabla
Filas	Documentos
Columnas	Campos (Clave-Valor)

Fuente: <http://www.rethinkdb.com/docs/sql-to-reql/javascript/>  
 Elaboración: RethinkDB

Los datos que son almacenados, tienen que tener la estructura definida para un documento JSON, es decir, debe estar compuesta por pares: clave-valor, la Figura 9 muestra un ejemplo de cómo se almacenan los datos en relación con las bases de datos relacionales.



Figura 9. Comparación de almacenamiento de datos Relacional y NoSQL

Fuente: <http://www.solvetic.com/tutoriales/articulo/468-convertir-bases-de-datos-relacionales-y-sql-para-mongodb/>  
 Elaboración: Anónimo.

Los datos que se utilizan para la implementación en los sistemas NoSQL se encuentran actualmente en una base de datos relacional SQL, en ella se almacena información generada por la red social Twitter a través de los tweets más populares dentro de un determinado tiempo, dichos datos se encuentran en la tabla llamada “Tweets”, de ahí se obtendrá un total de 20 millones de registros y se realiza la conversión automática de sql a NoSQL con la aplicación PENTAHO<sup>14</sup> a través de su módulo data-integration<sup>15</sup> con el Sistema Operativo MAC OS X, el proceso de transformación se muestra en el Anexo H.

La transformación se almacena temporalmente en una instancia cualquiera de MongoDB, ya que PENTAHO solo soporta la conversión a esta base de datos, lo que ahora corresponde es realizar la exportación de los mismos para luego realizar la carga en el entorno distribuido configurado. Para esto se utiliza mongoexport<sup>16</sup>. Se decide que el formato de los datos a descargar sean de tipo JSON y CSV, debido a que son compatibles con cada base de datos, En el caso de MongoDB y RethinkDB se realiza la importación con el archivo en formato .js y en CouchDB con al archivo .csv.

### **3.1.4 Definición de arquitectura**

La arquitectura se refiere al diseño o a la topología del entorno distribuido, en donde interactúan a través de las comunicaciones los distintos nodos identificados por los servidores (hardware) en donde está instalado y configurado el sistema de almacenamiento NoSQL (Software).

Para definir la arquitectura se indica los requerimientos necesarios de cada uno de los elementos que conforman la topología del sistema distribuido, para su correcta instalación e implementación.

#### **3.1.4.1 Selección de Software.**

Apoyándonos en la definición de (Rodríguez, 2011) Sobre el software: “Conjunto de datos y programas que maneja el ordenador. Es la parte lógica o inmaterial de un sistema

---

<sup>14</sup> Sitio Oficial de la herramienta Pentaho: <http://www.pentaho.com/>.

<sup>15</sup> Enlace de descarga del módulo Data-Integration: <http://community.pentaho.com/projects/data-integration/>

<sup>16</sup> Comando mongoexport: <https://docs.mongodb.org/v3.0/reference/program/mongoexport/>

informático”, se propone los siguientes elementos que corresponden al software necesario para la implementación del sistema distribuido:

- Sistema Operativo (Ubuntu 12.04 y Mac OS X 10.9) de 64 bits.

#### **Requisitos para MongoDB**

- MongoDB 3.0.10
- Robomongo

#### **Requisitos para CouchDB**

- CouchDB 1.6.1
- Curl
- couchimport
- NodeJS

#### **Requisitos para RethinkDB**

- RethinkDB 2.2
- NodeJS

### ***3.1.4.2 Selección de Hardware***

El hardware es parte de una red informática, representado por todos los dispositivos o elementos físicos necesarios para el envío de los datos, entre los distintos nodos que conforman el sistema distribuido. En el Trabajo de Titulación los dispositivos hardware suficientes para la instalación son:

- **Servidores:** Los servidores son aquellas computadoras que son capaces de compartir sus recursos a otros servidores (Universidad de Azuay, 2002). En el Trabajo de Titulación se utiliza un servidor principal, encargado de realizar la replicación o fragmentación de datos a los servidores secundarios.
- **Cableado:** Permite la conexión del servidor principal con los servidores secundarios, el tipo de cable que es utilizado para la implementación del sistema distribuido es el UTP.
- **Tarjeta de Red:** Las tarjetas de red es un adaptador cuya función principal es la de permitir la comunicación con otros PCs (Sánchez, 2015). Los servidores para poder

comunicarse en el sistema distribuido deben de tener instalado su placa de red correspondiente con la interfaz Ethernet.

- Dos equipos con Sistema Operativo MAC con disco duro de 280 GB y Memoria RAM de 64GB.
- Equipo con Sistema Operativo Ubuntu con disco duro de 450 GB y Memoria RAM mínima de 4GB.
- Debido a la cantidad de datos y al tamaño del archivo a cargar, se ha considerado la implementación de disco externo My Book Thunderbolt<sup>17</sup> de 4TB para obtener más capacidad de almacenamiento.

### **3.1.4.3 Comunicaciones**

Las comunicaciones se refieren al diseño de la topología, la misma que se implementa en el sistema distribuido. Las topologías de red muestran la distribución física y la forma de cómo están conectados los equipos de cómputo del entorno distribuido (Sandoval, 2011).

En la siguiente sección se realiza una breve descripción de los diferentes tipos de topologías que existen, para luego seleccionar la que más convenga y se adapte a las necesidades del Trabajo de Titulación:

#### **3.1.4.3.1 Red en anillo.**

Topología de red que presenta forma de anillo y cada nodo tiene un receptor y un transmisor encargados del envío de señales a las otras estaciones. En este tipo de red (Buettrich & Escudero, 2007) afirma que “Todos los nodos se conectan entre sí, de manera que cada nodo se conecta directamente a otros dos dispositivos”. La desventaja principal de la red en anillo es que, si un nodo falla toda la red pierde comunicación. La Figura 10 muestra la representación gráfica de las redes tipo anillo.

---

<sup>17</sup> Más detalles sobre disco externo en: <http://www.wdc.com/sp/products/products.aspx?id=630>





Figura 10. Topología en anillo  
 Fuente: <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.  
 Elaboración: FITEC

### 3.1.4.3.2 Red en árbol.

Topología de red que presenta forma de árbol, donde todos los nodos están colocados jerárquicamente, el nodo de mayor jerarquía es el encargado de controlar la red y comparten el mismo canal de comunicación. Desde el punto de vista del diseño muestra un parecido a las redes de forma en estrella y cuenta con un cable principal (backbone) al que están conectados todos los nodos que forman parte de la red (Buettrich & Escudero, 2007). Uno de los problemas de las redes en árbol es que suelen ocasionar cuellos de botella. La Figura 11 muestra la representación gráfica de las redes en forma de árbol.

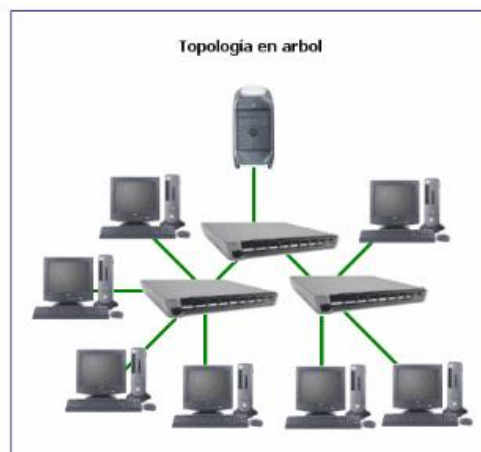


Figura 11. Topología en árbol  
 Fuente: <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.  
 Elaboración: FITEC

#### 3.1.4.3.3 Red en malla.

Topología de red llamada en malla debido a que los nodos pueden conectarse a uno o a varios nodos, lo que permite a los nodos enviar mensajes por varias rutas, (FITEC, 2011) afirma: “Si la red de malla está completamente conectada no puede existir absolutamente ninguna interrupción en las comunicaciones. Cada servidor tiene sus propias conexiones con todos los demás servidores”. Las redes en malla son bastante seguras y fiables ya que son libres de que se produzcan fallos y cuellos de botella. La Figura 12 muestra la representación gráfica de las redes en forma de malla.



Figura 12. Topología en malla.

Fuente: <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.

Elaboración: FITEC

#### 3.1.4.3.4 Red en bus.

Topología de red en la que todos los nodos comparten un mismo bus para poder comunicarse con el resto. (FITEC, 2011) afirma: “La topología de bus permite que todos los dispositivos de la red puedan ver todas las señales de todos los demás dispositivos, lo que puede ser ventajoso si desea que todos los dispositivos obtengan esta información”. Este tipo de redes son bastantes vulnerables de presentar problemas de tráfico y provocar colisiones debido a que comparten el mismo canal de comunicación. La Figura 13 muestra la representación gráfica de la topología en bus.

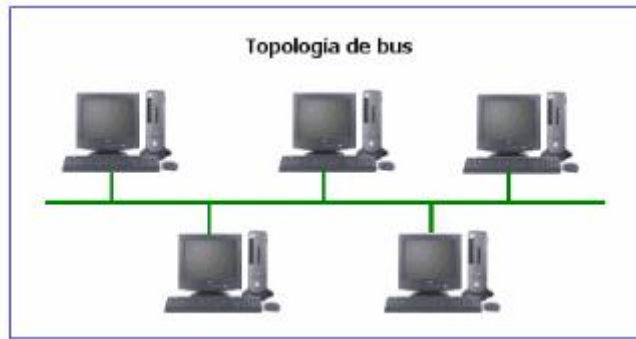


Figura 13. Topología en bus.  
Fuente: <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.  
Elaboración: FITEC

#### 3.1.4.3.5 Red en estrella.

Topología de red conocido como estrella, debido a que cada nodo está conectado a un nodo central, son bastantes flexibles en lo que respecta a la facilidad a la hora de incluir uno o varios nodos al sistema. Sobre este tipo de redes (FITEC, 2011) afirma: "Red en la cual las estaciones están conectadas directamente al servidor u ordenador y todas las comunicaciones se han de hacer necesariamente a través de él". Debido a que el nodo central es aquel que gestiona a toda la red, existe un mejor control sobre la información pero también es un punto débil ya que la seguridad del sistema puede verse afectada. La Figura 14 muestra la representación gráfica de las redes en forma de estrella.



Figura 14. Topología en estrella.  
Fuente: <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.  
Elaboración: FITEC

#### 3.1.4.3.6 Elección y diseño de Topología a implementarse

Tomando en cuenta los distintos tipos de topologías, se analiza las configuraciones tanto de las funcionalidades Fragmentación como Replicación en nuestras bases de datos y se hace un contraste con los elementos que conforman la arquitectura que tenemos disponibles, por tal razón se ha decidido aplicar la topología tipo Estrella.

Las redes de tipo estrella están compuestas por un nodo central que en nuestro caso es el servidor principal, el mismo que está encargado de realizar y distribuir la replicación y fragmentación de los datos a los demás servidores secundarios que conforman el entorno distribuido. Los servidores secundarios de esta topología están conectados directamente al servidor principal y son configurados para que reciban los datos.

Una de las ventajas de implementar la red tipo estrella frente a las otras topologías es la facilidad de adaptar o configurar un nuevo nodo a la red según lo afirma (Garrido, 2007), lo que permite cumplir con una de las metas principales del Trabajo de Titulación en lo que respecta a que el ambiente distribuido sea transparente, por medio de un proceso que resulte lo más sencillo posible, Otra razón por la cual se decide implementar esta topología es que en las tres bases de datos se requiere de un servidor principal encargado de distribuir los datos hacia los servidores secundarios.

En el caso de MongoDB, para la configuración de la fragmentación se requiere de un servidor principal que sea el encargado de atender las peticiones de los clientes y mediante su configuración, distribuya los datos lo más uniformemente posible entre las instancias de los servidores secundarios. Todo cambio que se realice en la base de datos del servidor principal, automáticamente se refleja en los servidores secundarios, cabe recalcar que todos los equipos están conectados gracias a la red formada en el laboratorio de Tecnologías Avanzadas de la Web. En la Figura 15 se presenta el esquema con todos los componentes que forman parte de la arquitectura con la base de datos MongoDB:

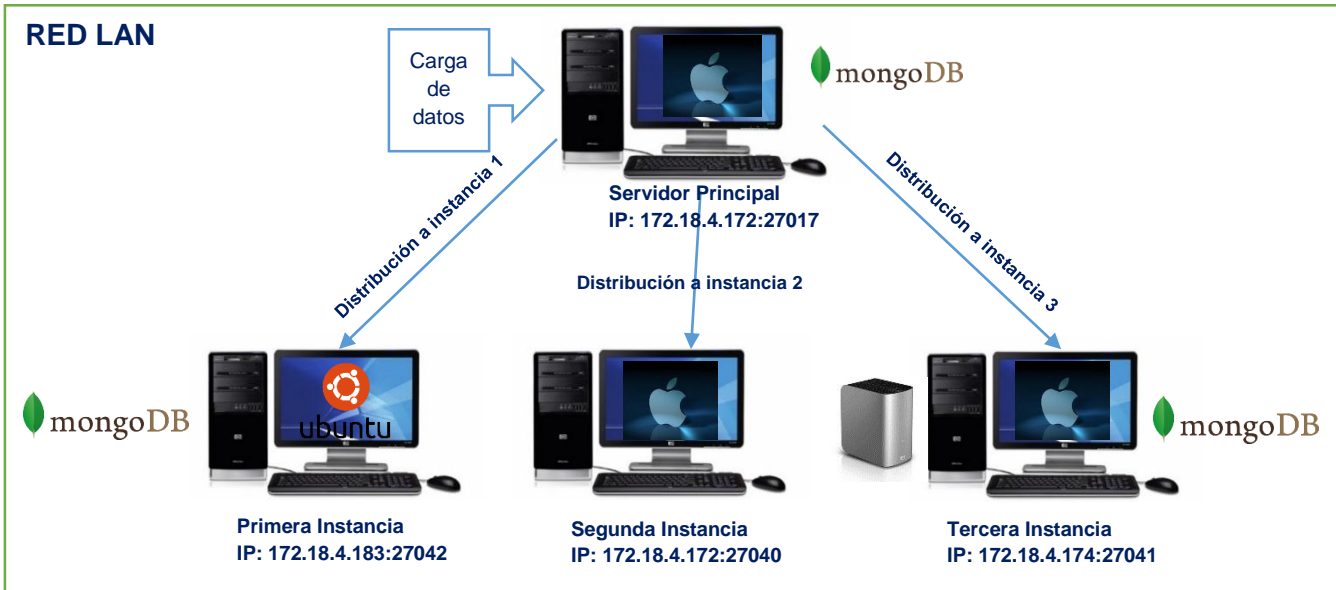


Figura 15. Topología a implementar en MongoDB  
Elaboración: Propia

En el caso de RethinkDB al igual que en MongoDB se aplica la fragmentación de datos, para ello se requiere de un servidor principal que sea el encargado de atender las peticiones de los clientes y mediante su configuración, distribuya los datos entre los servidores secundarios. Todo cambio que se realice en la base de datos del servidor principal, automáticamente se refleja en los servidores secundarios. En la Figura 16 se presenta el esquema con todos los componentes que se implementan en la base de datos RethinkDB:

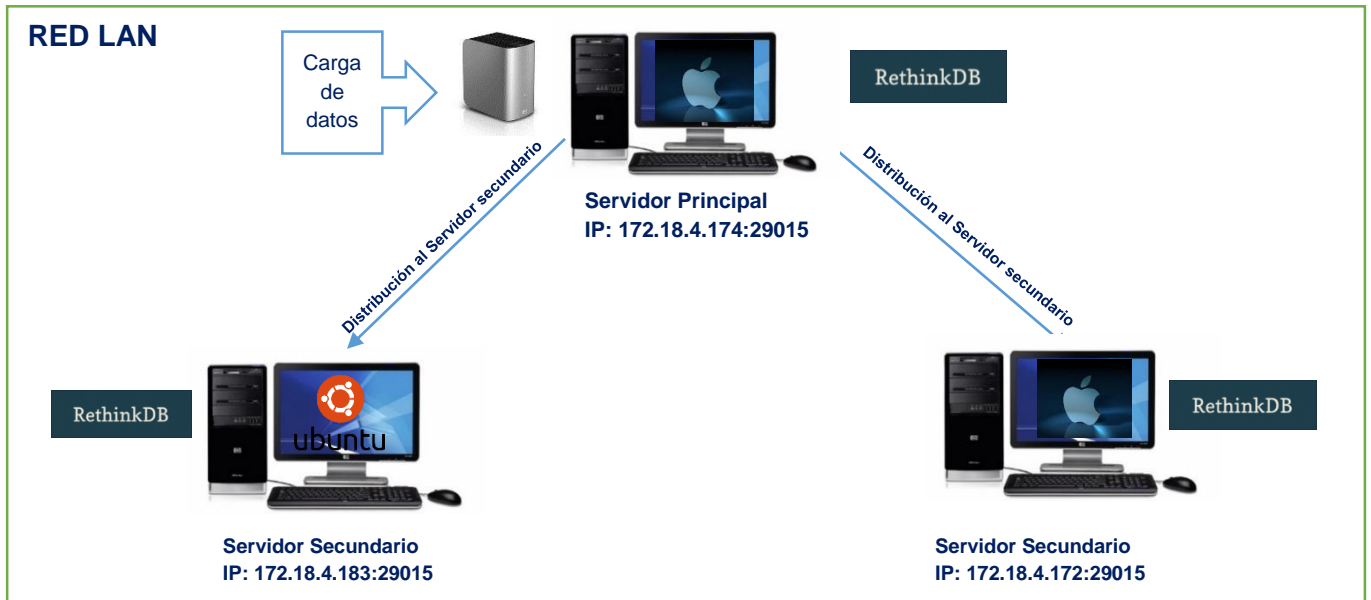


Figura 16. Topología a implementar en RethinkDB  
Elaboración: Propia

En CouchDB, la replicación es la capacidad de sincronizar y mantener copias de los datos en distintos servidores que formen parte de un entorno distribuido, conformado por un servidor principal denominado "MASTER" y de servidores secundarios. Según (Gutiérrez Pérez, 2014):

El servidor principal es el encargado de "publicar" los datos, es decir, es el servidor sobre el que los usuarios realizan las diferentes operaciones sobre los datos. Los servidores secundarios, están conectados por medio de una red al servidor principal remoto para mantener una copia de los datos publicados por el servidor principal. (p.201)

La topología con esta base de datos es similar a la indicada con MongoDB en la Figura 15, la diferencia radica en que con CouchDB no se reparten los datos en los servidores secundarios, si no, el servidor principal realiza una copia exacta de la base de datos en los servidores secundarios. En la Figura 17 se muestra el esquema que se aplica con CouchDB.

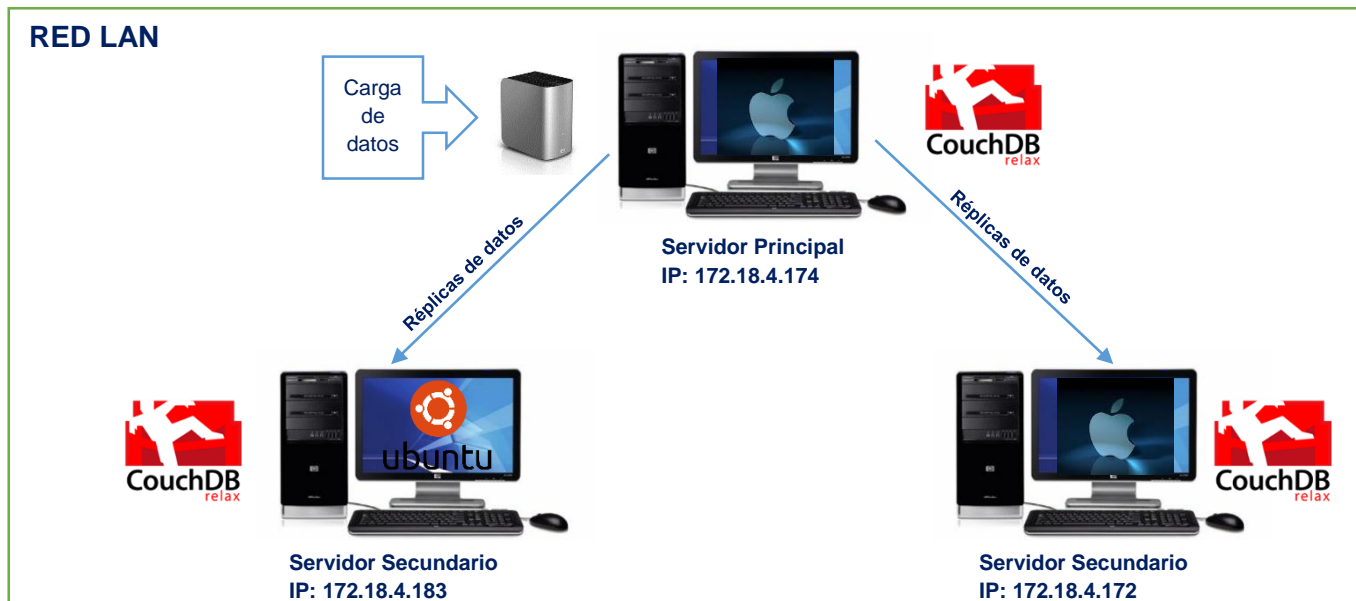


Figura 17. Topología a implementar en CouchDB  
Elaboración: Propia

### 3.1.5 Configuración e Implementación del Entorno Distribuido

Las secciones 3.1.5.1, 3.1.5.2 y 3.1.5.3 tienen una singular importancia, ya que es donde se muestran los procedimientos para la configuración del entorno distribuido, los mismos que consisten en implementar las funcionalidades de Fragmentación en las bases de datos MongoDB, RethinkDB y Replicación en CouchDB asegurando la escalabilidad entre los distintos servidores que conforman el sistema distribuido; lo que permite cumplir con el objetivo principal del proyecto.

#### 3.1.5.1 Fragmentación en MongoDB.

Es el método de almacenamiento donde los datos se distribuyen en un conjunto de equipos previamente configurados, su propósito se basa en apoyar la escalabilidad y el soporte de grandes cantidades de información, esta funcionalidad resulta la más indicada en sistemas donde los datos crecen constantemente y el rendimiento del servidor que lo soporta puede verse afectada, obligando a la adquisición o a mejorar los recursos de este, en cuanto a procesador, memoria RAM y espacio en Disco. La Fragmentación permite el escalamiento horizontal, con esto se logra tomar en cuenta principalmente el número de máquinas

dejando en un segundo plano, el Hardware de las mismas.

MongoDB realiza la distribución de los datos de acuerdo a un campo que sea elegido como clave (shardkey<sup>18</sup>), por lo tanto su elección influye tanto en el rendimiento como en la uniformidad de los datos en todos los equipos.

## - Arquitectura

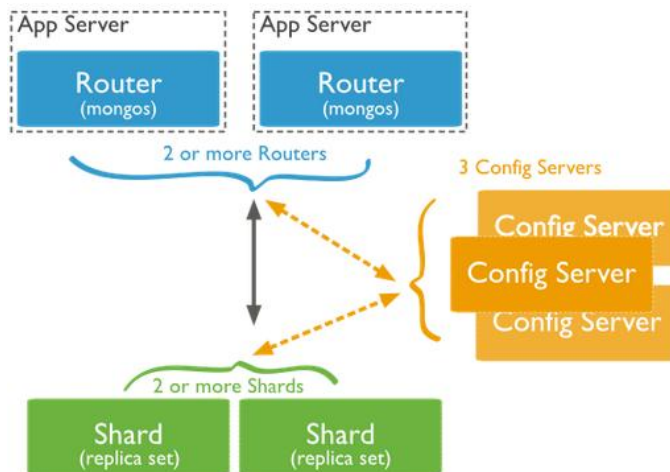


Figura 18. Arquitectura Fragmentación en MongoDB  
Fuente: <https://docs.mongodb.org/v3.0/core/sharded-cluster-architectures-production/>  
Elaboración: MongoDB.

Los componentes de un clúster en MongoDB son los siguientes:

- **Shard:** Componente en donde se almacena los datos de las colecciones que han sido fragmentadas.
- **Config Server:** Elemento que almacena los metadatos de la fragmentación, en él constan información sobre porcentajes de los datos que han sido divididos y en que servidor se encuentran.
- **Router:** Componente encargado de sincronizar las instancias config Server y es el encargado de distribuir los datos a los servidores configurados.

Para la gestión de grandes cantidades de información, MongoDB recomienda la

<sup>18</sup> ShardKey en MongoDB: <https://docs.mongodb.org/v3.0/core/sharding-shard-key/>



implementación de la arquitectura de un Clúster en Producción (Figura 18), la misma que está compuesta por los siguientes elementos:

- **Tres Config Server:** Cada servidor de configuración debe de estar en máquinas distintas, ya que si uno falla, el entorno distribuido deja de funcionar.
- **Dos o más Shards:** Para poder apreciar cómo se distribuyen los datos, mientras más shards configuremos, se obtiene más información fragmentada.
- **Uno o dos Routers:** Donde los clientes interactúan con la Base de datos, además estas instancias se encargan de equilibrar la carga en los servidores.

La configuración del entorno distribuido a través de la funcionalidad fragmentación y con el esquema que se muestra en la Figura 15 se la puede apreciar en el anexo I.

### ***3.1.5.2 Fragmentación en RethinkDB.***

En RethinkDB la fragmentación es posible gracias a la clave primaria de la tabla para realizar la división de la información en cada servidor, al igual de lo que ocurre en MongoDB con el campo clave, este sistema crea rangos para la distribución de datos, de acuerdo al número de fragmentos que el usuario requiera y posteriormente el sistema realiza un análisis en toda la tabla, esto con el fin de obtener una distribución en cada servidor, lo que permite que el escalamiento sea sencillo ya que toma los recursos que cada servidor pueda soportar.

La configuración se puede realizar de dos formas: mediante la interfaz web o por medio de línea de comandos ReQL, en ambos casos se pueden especificar los parámetros necesarios para los componentes que son los siguientes:

**Número de Shards (Fragmentos):** Hace referencia al número de fragmentos en que se divide los datos entre los servidores, este valor tiene que ser menor o igual al número de máquinas conectadas al servidor principal.

**Número de Réplicas:** Componente que describe el número de copias que realiza la base de datos por cada shard configurado.

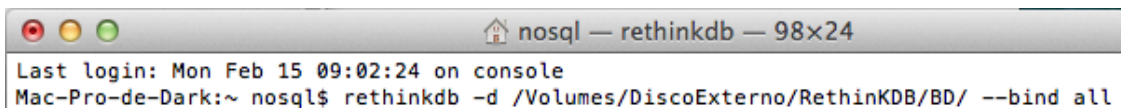
La configuración del entorno distribuido a través de la funcionalidad de fragmentación es de la siguiente forma y el esquema a implementar es la arquitectura como se muestra en la Figura 16.

- 1) Creación de directorio en el disco externo del servidor principal, donde se guarda la base de datos. La configuración se la puede apreciar en la Figura 19.

```
Mac-Pro-de-Dark:~ nosql$ rethinkdb create /Volumes/DiscoExterno/RethinkKDB/BD/
```

Figura 19. Creación de directorio en RethinkDB  
Elaboración: Propia

- 2) Inicio de la base de datos en el servidor principal. La configuración se la muestra en la Figura 20.



```
nosql — rethinkdb — 98x24
Last login: Mon Feb 15 09:02:24 on console
Mac-Pro-de-Dark:~ nosql$ rethinkdb -d /Volumes/DiscoExterno/RethinkKDB/BD/ --bind all
```

Figura 20. Inicio Servidor Principal  
Elaboración: Propia.

**Donde:**

**bind-all:** Parámetro que permite escuchar todas las direcciones de las interfaces de red.

**d:** Indica la dirección del directorio donde se almacena la base de datos.

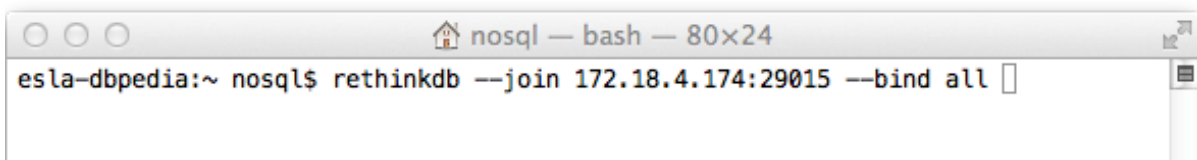
- 3) El inicio en los servidores secundarios se las puede apreciar en las Figuras 21 y 22 y se debe añadir el parámetro “join host:port”, lo que permite conectarse al servidor principal, para formar el clúster.

**Donde:**

**host:** Es la dirección IP correspondiente al servidor principal.

**port:** Puerto del servidor principal para formar el clúster, por defecto es 29015.

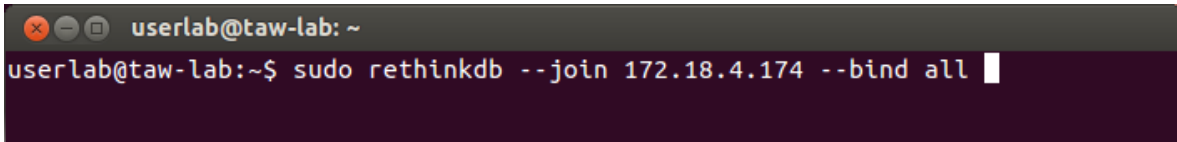
**Inicio de Base de datos en el Segundo Servidor.**



```
nosql — bash — 80x24
esla-dbpedia:~ nosql$ rethinkdb --join 172.18.4.174:29015 --bind all
```

Figura 21. Inicio RethinkDB en el segundo servidor.  
Elaboración: Propia.

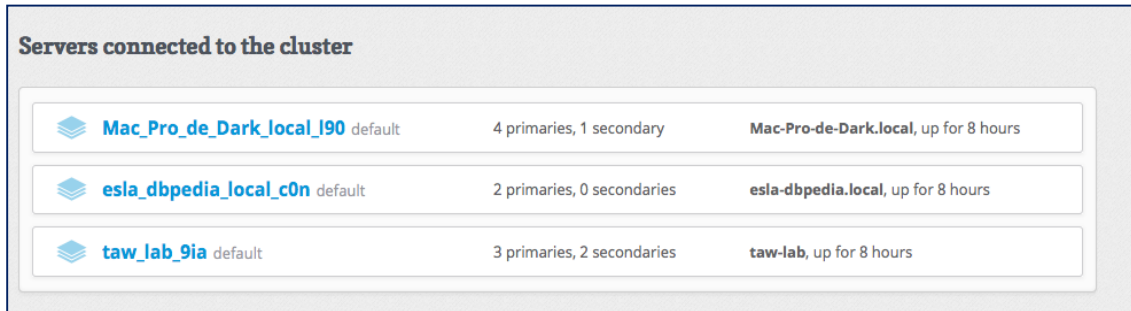
### Inicio de Base de datos en el Tercer Servidor.



```
userlab@taw-lab: ~  
userlab@taw-lab:~$ sudo rethinkdb --join 172.18.4.174 --bind all
```

Figura 22. Inicio RethinkDB en el tercer servidor.  
Elaboración: Propia.

Se verifica que los tres servidores estén conectados entre sí, para ello en la interfaz web en la pestaña “Severs” se puede apreciar la conexión, como lo indica la Figura 23.






Servers connected to the cluster		
 <b>Mac_Pro_de_Dark_local_l90</b> default	4 primaries, 1 secondary	Mac-Pro-de-Dark.local, up for 8 hours
 <b>esla_dbpedia_local_c0n</b> default	2 primaries, 0 secondaries	esla-dbpedia.local, up for 8 hours
 <b>taw_lab_9ia</b> default	3 primaries, 2 secondaries	taw-lab, up for 8 hours

Figura 23. Verificación de conexión de nodos en RethinkDB  
Elaboración: Propia

#### 3.1.5.3 Replicación en CouchDB.

A diferencia de la fragmentación, la replicación es el método que crea una copia exacta de toda la base de datos en cada máquina configurada al entorno distribuido, esto es posible gracias a la sincronización entre los nodos a través de una red de comunicación, donde los cambios realizados a dicha base de datos son registrados localmente; con este tipo de almacenamiento se garantiza que el sistema distribuido sea escalable, fiable y disponible.

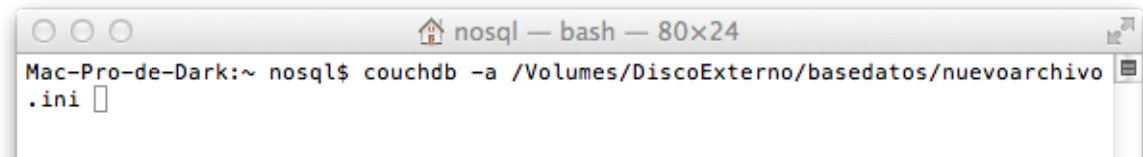
Debido a que la base de datos CouchDB no soporta la fragmentación, para poder cumplir con el objetivo principal del Trabajo de Titulación, se aplica la funcionalidad Replicación. CouchDB permite la replicación al estilo maestro-esclavo, donde el maestro es el denominado servidor principal y su función consiste en sincronizar y distribuir la base de datos replicada en varios nodos, en cambio los servidores esclavos son aquellos nodos que almacenan las copias de las bases de datos distribuidas por el nodo maestro, y podrán utilizar de manera independiente los datos, lo que garantiza el respaldo total de toda la información.

La configuración de la replicación en CouchDB es de la siguiente forma, y el esquema a implementar es la arquitectura que se muestra en la Figura 17.

- 1) Cambiar el nombre del host que viene por defecto, por las direcciones IP de los servidores en el archivo local.ini. En entornos MAC el archivo se encuentra ubicado en la siguiente dirección “/usr/local/etc/couchdb/local.ini”, en Ubuntu se localiza en “/etc/couchdb/local.ini”. El parámetro a cambiar es “*bind\_address*”.
- 2) Iniciar CouchDB en todos los servidores como se muestra en las Figuras 24, 25 y 26.

### Inicio de CouchDB en el Primer Servidor (Principal).

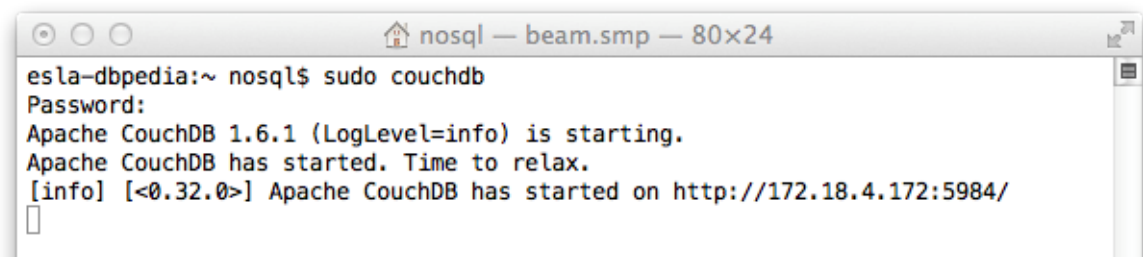
En este nodo la base de datos se localiza en la unidad de disco externo, para eso, se debe de copiar el archivo de inicio (local.ini) en alguna carpeta del disco externo y editar el parámetro “database\_dir” con la nueva dirección donde se quiera guardar la base de datos. Para ello en la línea de comandos se coloca el argumento “-a” indicando el inicio del servidor con la nueva configuración.



```
Mac-Pro-de-Dark:~ nosql$ couchdb -a /Volumes/DiscoExterno/basedatos/nuevoarchivo
.ini
```

Figura 24. Inicio Servidor principal CouchDB  
Elaboración: Propia

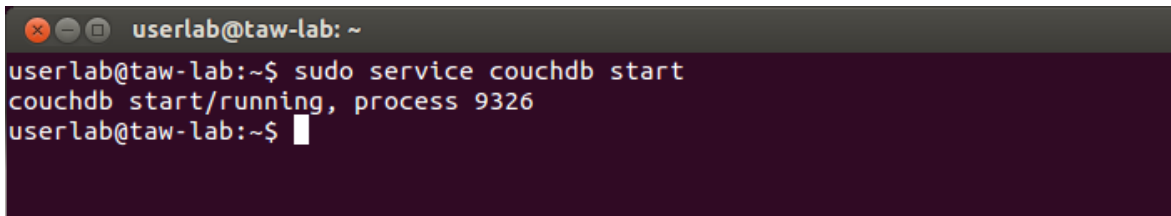
### Inicio de CouchDB en el Segundo Servidor.



```
esla-dbpedia:~ nosql$ sudo couchdb
Password:
Apache CouchDB 1.6.1 (LogLevel=info) is starting.
Apache CouchDB has started. Time to relax.
[info] [<0.32.0>] Apache CouchDB has started on http://172.18.4.172:5984/
```

Figura 25. Inicio primer servidor secundario.  
Elaboración: Propia

## Inicio de CouchDB en el Tercer Servidor.



```
userlab@taw-lab: ~  
userlab@taw-lab:~$ sudo service couchdb start  
couchdb start/running, process 9326  
userlab@taw-lab:~$
```

Figura 26. Inicio segundo servidor secundario  
Elaboración: Propia

- 3) Crear Base de datos a replicar en los servidores principales y secundarios con distinto o igual nombre.
- 4) Realizar la replicación de datos desde el servidor principal a los secundarios. Como se muestra en las Figuras 27 y 28.

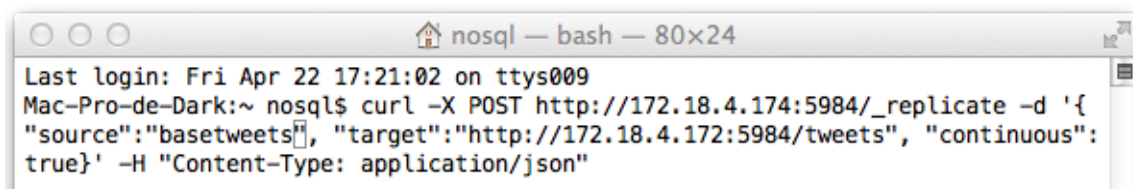
### Donde:

**Source:** Nombre de la base de datos a replicar.

**Target:** Indica el nombre de host y de base de datos remota

**H:** Parámetro donde se coloca el tipo de contenido de los datos.

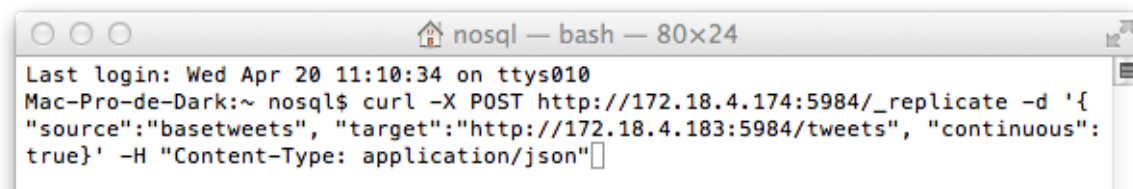
## Replicar datos hacia el Segundo Servidor (172.18.4.172).



```
Mac-Pro-de-Dark:~ nosql$ curl -X POST http://172.18.4.174:5984/_replicate -d '{  
"source":"basetweets", "target":"http://172.18.4.172:5984/tweets", "continuous":  
true}' -H "Content-Type: application/json"
```

Figura 27. Replicación al Segundo servidor  
Elaboración: Propia.

## Replicar datos hacia el Tercer Servidor (172.18.4.183).



```
Mac-Pro-de-Dark:~ nosql$ curl -X POST http://172.18.4.174:5984/_replicate -d '{  
"source":"basetweets", "target":"http://172.18.4.183:5984/tweets", "continuous":  
true}' -H "Content-Type: application/json"
```

Figura 28. Replicación al Tercer Servidor  
Elaboración: Propia.

### 3.1.6 Importación de datos a los Sistemas NoSQL elegidos:

En la sección 3.1.6 se muestra los comandos necesarios para la importación de los datos sobre la configuración del entorno distribuido realizada en las secciones 3.1.5.1, 3.1.5.2 y 3.1.5.3 en los sistemas MongoDB, RethinkDB y CouchDB.

- En mongoDB se realiza la carga de datos al servidor principal a través del comando mongoimport. La configuración se muestra en la Figura 29.

**Donde:**

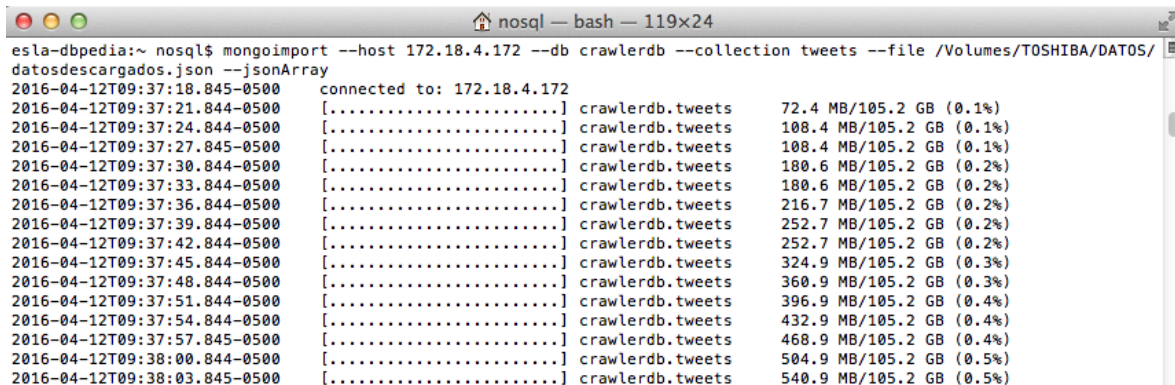
**host:** Indica la dirección IP del servidor donde se realiza la sincronización de los Config Servers.

**db:** Nombre de la base de datos.

**collection:** Nombre de la colección.

**file:** Archivo de la base de datos a importarse.

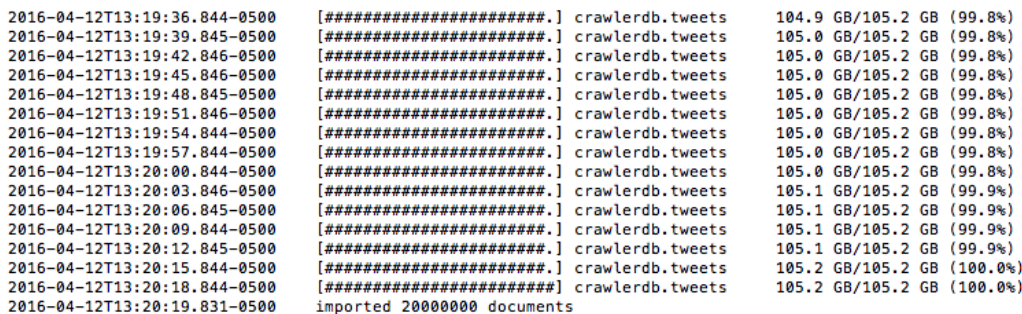
**jsonArray:** Parámetro que indica el formato del archivo .json de origen.



```
esla-dbpedia:~ nosql$ mongoimport --host 172.18.4.172 --db crawlerdb --collection tweets --file /Volumes/TOSHIBA/DATOS/datosdescargados.json --jsonArray
2016-04-12T09:37:18.845-0500 connected to: 172.18.4.172
2016-04-12T09:37:21.844-0500 [.....] crawlerdb.tweets 72.4 MB/105.2 GB (0.1%)
2016-04-12T09:37:24.844-0500 [.....] crawlerdb.tweets 108.4 MB/105.2 GB (0.1%)
2016-04-12T09:37:27.845-0500 [.....] crawlerdb.tweets 108.4 MB/105.2 GB (0.1%)
2016-04-12T09:37:30.844-0500 [.....] crawlerdb.tweets 180.6 MB/105.2 GB (0.2%)
2016-04-12T09:37:33.844-0500 [.....] crawlerdb.tweets 180.6 MB/105.2 GB (0.2%)
2016-04-12T09:37:36.844-0500 [.....] crawlerdb.tweets 216.7 MB/105.2 GB (0.2%)
2016-04-12T09:37:39.844-0500 [.....] crawlerdb.tweets 252.7 MB/105.2 GB (0.2%)
2016-04-12T09:37:42.844-0500 [.....] crawlerdb.tweets 252.7 MB/105.2 GB (0.2%)
2016-04-12T09:37:45.844-0500 [.....] crawlerdb.tweets 324.9 MB/105.2 GB (0.3%)
2016-04-12T09:37:48.844-0500 [.....] crawlerdb.tweets 360.9 MB/105.2 GB (0.3%)
2016-04-12T09:37:51.844-0500 [.....] crawlerdb.tweets 396.9 MB/105.2 GB (0.4%)
2016-04-12T09:37:54.844-0500 [.....] crawlerdb.tweets 432.9 MB/105.2 GB (0.4%)
2016-04-12T09:37:57.845-0500 [.....] crawlerdb.tweets 468.9 MB/105.2 GB (0.4%)
2016-04-12T09:38:00.844-0500 [.....] crawlerdb.tweets 504.9 MB/105.2 GB (0.5%)
2016-04-12T09:38:03.845-0500 [.....] crawlerdb.tweets 540.9 MB/105.2 GB (0.5%)
```

Figura 29. Carga de Datos en MongoDB-Inicio  
Elaboración: Propia.

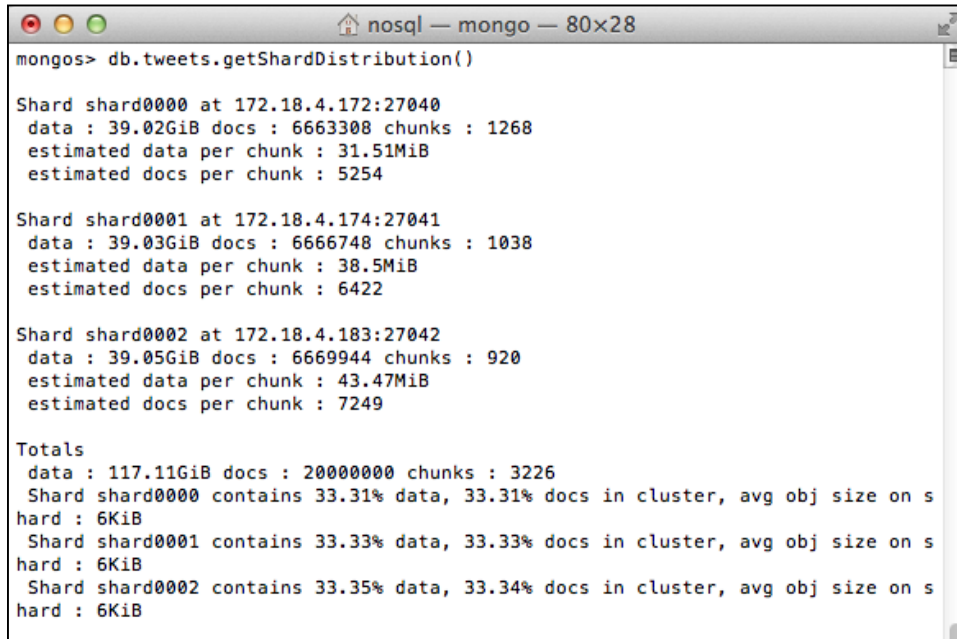
La Figura 30 muestra la finalización de la carga de datos en MongoDB.



```
2016-04-12T13:19:36.844-0500 [#####.] crawlerdb.tweets 104.9 GB/105.2 GB (99.8%)
2016-04-12T13:19:39.845-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:42.846-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:45.846-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:48.845-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:51.846-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:54.844-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:19:57.844-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:20:00.844-0500 [#####.] crawlerdb.tweets 105.0 GB/105.2 GB (99.8%)
2016-04-12T13:20:03.846-0500 [#####.] crawlerdb.tweets 105.1 GB/105.2 GB (99.9%)
2016-04-12T13:20:06.845-0500 [#####.] crawlerdb.tweets 105.1 GB/105.2 GB (99.9%)
2016-04-12T13:20:09.844-0500 [#####.] crawlerdb.tweets 105.1 GB/105.2 GB (99.9%)
2016-04-12T13:20:12.845-0500 [#####.] crawlerdb.tweets 105.1 GB/105.2 GB (99.9%)
2016-04-12T13:20:15.844-0500 [#####.] crawlerdb.tweets 105.2 GB/105.2 GB (100.0%)
2016-04-12T13:20:18.844-0500 [#####.] crawlerdb.tweets 105.2 GB/105.2 GB (100.0%)
2016-04-12T13:20:19.831-0500 imported 20000000 documents
```

Figura 30. Carga de Datos en MongoDB-Finalización.  
Elaboración: Propia.

Una vez culminada la importación de todos los datos, en MongoDB se puede mostrar la distribución de los datos en cada Servidor, como lo indica la Figura 31.



```
mongos> db.tweets.getShardDistribution()

Shard shard0000 at 172.18.4.172:27040
data : 39.02GiB docs : 6663308 chunks : 1268
estimated data per chunk : 31.51MiB
estimated docs per chunk : 5254

Shard shard0001 at 172.18.4.174:27041
data : 39.03GiB docs : 6666748 chunks : 1038
estimated data per chunk : 38.5MiB
estimated docs per chunk : 6422

Shard shard0002 at 172.18.4.183:27042
data : 39.05GiB docs : 6669944 chunks : 920
estimated data per chunk : 43.47MiB
estimated docs per chunk : 7249

Totals
data : 117.11GiB docs : 20000000 chunks : 3226
Shard shard0000 contains 33.31% data, 33.31% docs in cluster, avg obj size on s
hard : 6KiB
Shard shard0001 contains 33.33% data, 33.33% docs in cluster, avg obj size on s
hard : 6KiB
Shard shard0002 contains 33.35% data, 33.34% docs in cluster, avg obj size on s
hard : 6KiB
```

Figura 31. Distribución de datos entre los servidores de MongoDB  
Elaboración: Propia.

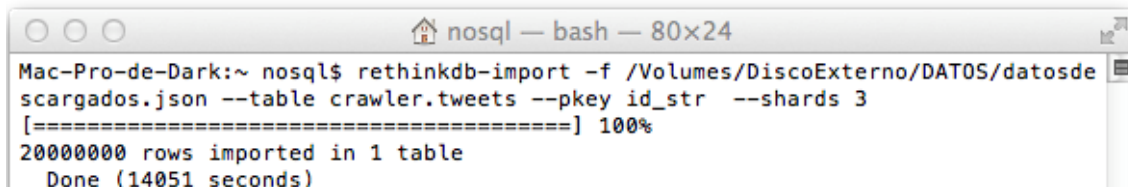
- En RethinkDB la carga de datos se muestra en la Figura 32 en el servidor principal, por medio del comando rethinkdb-import, donde se especifican los siguientes parámetros:

**f** : Dirección del archivo a cargar

**table** Se especifica el nombre de la base de datos y la tabla

**pkey**: Declaración de campo primario.

**shards**; Número de Nodos donde se distribuyen los datos.



```
Mac-Pro-de-Dark:~ nosql$ rethinkdb-import -f /Volumes/DiscoExterno/DATOS/datosde
scargados.json --table crawler.tweets --pkey id_str --shards 3
[=====] 100%
20000000 rows imported in 1 table
Done (14051 seconds)
```

Figura 32. Importación de datos en RethinkDB.  
Elaboración: Propia.

La distribución de los datos en cada shard, se la puede apreciar en la figura 33.

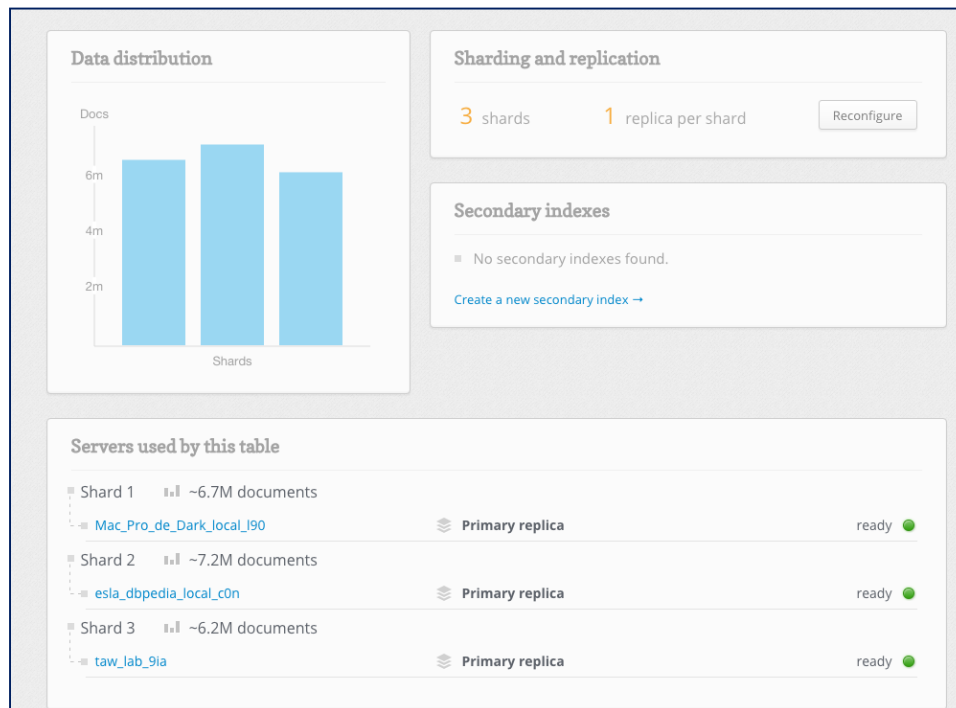


Figura 33. Distribución de datos entre los servidores de RethinkDB.  
Elaboración: Propia.

- En couchDB la carga de datos se realiza con la ayuda de la herramienta llamada “couchimport”, donde se especifican los siguientes parámetros:

**cat:** Directorio del archivo a importar

**db:** Nombre de la base de datos a importar.

**parallelism:** Parámetro que indica el número de operaciones concurrentes para la inserción de datos.

**delimiter:** Separador de columnas, en formato csv por defecto es ",".

**url:** Nombre del host a importar.

Por defecto, la importación de datos se realiza con una sola operación concurrente lo que provoca una carga demasiado lenta, por esta razón, se decidió utilizar como prueba inicial 80 operaciones concurrentes (parallelism). Las configuraciones se muestran en la Figura 34.



```
nosql — bash — 80x24
Last login: Fri Apr 22 17:21:02 on ttys009
Mac-Pro-de-Dark:~ nosql$ cat /Volumes/TOSHIBA/DATOS/datosTweets.csv | couchimport
--db basetweets --parallelism 80 --delimiter "," --url "http://172.18.4.174:5984"
```

Figura 34. Importación de datos en CouchDB  
Elaboración: Propia

La Figura 35 muestra la finalización de la carga de datos en CouchDB.

```
couchimport Written 500 (19993000) +448ms
couchimport Written 500 (19993500) +296ms
couchimport Written 500 (19994000) +255ms
couchimport Written 500 (19994500) +468ms
couchimport Written 500 (19995000) +184ms
couchimport Written 500 (19995500) +234ms
couchimport Written 500 (19996000) +38ms
couchimport Written 500 (19996500) +85ms
couchimport Written 500 (19997000) +115ms
couchimport Written 0 (19997000) +56ms
couchimport Written 500 (19997500) +4ms
couchimport Written 500 (19998000) +248ms
couchimport Written 500 (19998500) +1ms
couchimport Written 500 (19999000) +1ms
couchimport Written 500 (19999500) +1ms
couchimport Written 500 (20000000) +1ms
couchimport Import complete +72ms
```

Figura 35. Carga Total en CouchDB  
Elaboración: Propia.

### 3.2 Pruebas Piloto

Una vez realizada la implementación del entorno distribuido, para comprobar su funcionamiento, es necesario llevar a cabo pruebas con el fin de experimentar todas las configuraciones implementadas y encontrar aquellos errores que no permitan obtener un rendimiento óptimo en nuestro sistema y por ende, no cumplir con el objetivo principal del Trabajo de Titulación, por lo tanto las pruebas piloto son de gran importancia debido a que los resultados que arrojen las mismas, contribuirán hacia la mejora del sistema distribuido.

En estas pruebas iniciales se considera oportuno tomar en cuenta criterios de medición como son el tiempo que se demora en cargar los datos y la cantidad de disco duro que ocupa la base de datos implementada en los sistemas de almacenamiento.

### 3.2.1 Resultados previos

Las pruebas iniciales aplicando fragmentación en MongoDB y RethinkDB y replicación en CouchDB con una carga equivalente a 20 millones de datos, proporcionan los resultados que se muestran en la Tabla 9.

Tabla 9. Tabla de resultados previos.

Base de datos	Tiempo de Carga	Espacio ocupado en disco
MongoDB	223 min	137,2 GB
CouchDB	245 min	46.8 GB
RethinkDB	234 min	127,8 GB

Elaboración: Propia.

Aunque no es parte del presente trabajo, se ha considerado oportuno realizar la recuperación de información con el objetivo de comprobar el rendimiento y funcionalidad de cada base de datos. Para ello se realiza la consulta del primer millón de datos, cantidad que se considera suficiente para realizar la prueba por su gran contenido, la extracción de datos son de dos tipos: la primera consulta contiene todos los campos, mientras que la segunda se hace un filtrado con los campos más importantes, estos son: el id del tweet en String (id\_str) la fecha del tweet (tweet\_date), los hashtags del tweet (hashtag) y el contenido de la fila en formato json (json).

Para realizar las consultas en MongoDB, se utilizan las funciones disponibles por línea de comandos en la herramienta de administración Robomongo, la constancia de la ejecución de las consultas en MongoDB se la puede apreciar en las Figuras 36 y 37.

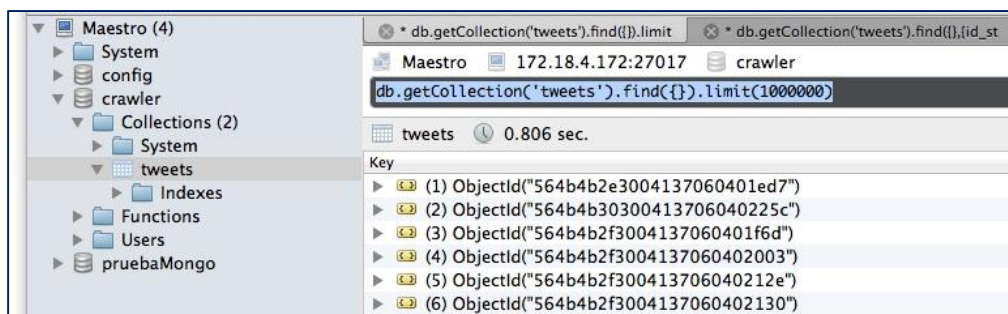


Figura 36. Primera consulta en MongoDB

Elaboración: Propia

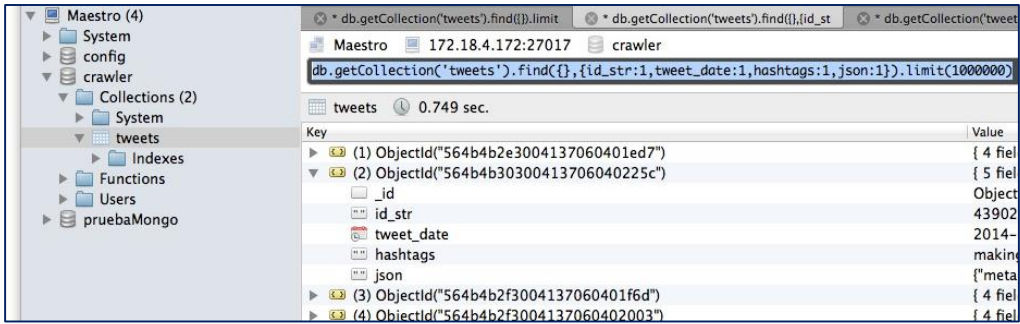


Figura 37. Segunda consulta en MongoDB.  
Elaboración: Propia.

Para realizar las consultas en RethinkDB, se utilizan las funciones disponibles por línea de comandos en la administración web del servidor principal, la constancia de la ejecución de las consultas se la puede apreciar en las Figuras 38 y 39.

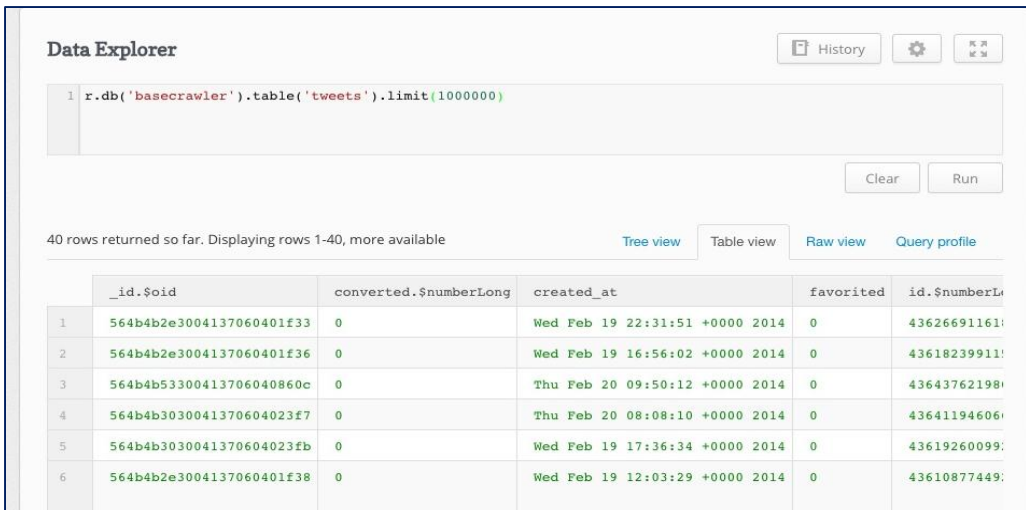


Figura 38. Primera consulta en RethinkDB  
Elaboración: Propia.

**Data Explorer** History Settings

```
1 r.db('basecrawler').table('tweets').pluck('id_str', 'tweet_date', 'hashtags', 'json').limit(1000000)
```

Clear Run

40 rows returned so far. Displaying rows 1-40, more available [Tree view](#) [Table view](#) [Raw view](#) [Query profile](#)

	id_str	json	tweet_date.\$date	hashtags
1	436266911618568192	{"metadata":{"result_type":"recent",	2014-02-19T22:31:51.000Z	yuri octave
2	436182399115100160	{"metadata":{"result_type":"recent",	2014-02-19T16:56:02.000Z	Centos Mobile Octav
3	436437621980815360	{"metadata":{"result_type":"recent",	2014-02-20T09:50:12.000Z	quiz presentation s
4	436411946066522112	{"metadata":{"result_type":"recent",	2014-02-20T08:08:10.000Z	undefined
5	436192600992014336	{"metadata":{"result_type":"recent",	2014-02-19T17:36:34.000Z	grants entrepreneur
6	436108774492086272	{"metadata":{"result_type":"recent",	2014-02-19T12:03:29.000Z	sleepingdogfx fx fu

Figura 39. Segunda consulta en RethinkDB.  
Elaboración: Propia.

En la tabla 10 se muestra el tiempo que corresponde a la duración en ejecutarse las consultas de los datos.

Tabla 10. Resultados de Recuperación de datos.

Base de datos	Tiempo de Consulta de 1 millón de datos	
	Con todos los campos	Filtrando campos
MongoDB	0.806 seg	0.749 seg.
RethinkDB	0.958 seg.	0.820 seg

Elaboración: Propia.

En la tabla 10 se aprecia que MongoDB supera ligeramente a RethinkDB en la consulta de datos pero ambas bases de datos presentan resultados que satisfacen totalmente la implementación de los sistemas NoSQL en un entorno distribuido.

El siguiente capítulo tiene por objetivo mejorar la configuración inicial en lo que respecta al tiempo de carga y al espacio en disco duro para concluir con la recomendación del producto de base de datos para su implementación.

**CAPÍTULO IV:  
REFINAMIENTO E IMPLEMENTACIÓN FINAL**

El capítulo 4 pretende mejorar los resultados que arrojan las pruebas piloto con cada una de las bases de datos y así obtener un sistema distribuido optimizado. Se considera que el entorno distribuido es mejorado tomando en cuenta principalmente los criterios del tiempo que se demora en cargar los datos en el sistema de almacenamiento, manteniendo su persistencia en todos los servidores que conforman el ambiente distribuido y el espacio que ocupa los datos almacenados en los discos duros en los servidores, para ello se investiga en cada documentación oficial aquellos parámetros que permitan la mejora. En las secciones 4.1 y 4.2 se muestran la optimización en MongoDB y RethinkDB aumentando procesos a la importación de datos, en cambio en la sección 4.3 se enseña la mejora en CouchDB en lo que respecta al espacio en disco duro. En la sección 4.4 se presenta la tabla con los datos optimizados en cada base de datos para finalmente realizar la elección final que se recomienda en la sección 4.5.

#### **4.1 Optimización en MongoDB**

La mejora en este sistema de almacenamiento se ejecuta sobre la misma arquitectura como lo indica la Figura 15 y se basa en el tiempo de carga, por medio del parámetro “numInsertionWorkers” en el comando mongoimport, lo que permite aumentar el número de procesos encargados de atender peticiones para realizar la carga, por defecto, viene configurado por el valor de 1 proceso, aumentando dicho parámetro se obtiene mayor consumo de memoria RAM lo que permite la aceleración de la carga. El valor que produjo la mejora en este sistema distribuido es de 150 procesos. El detalle de dicha configuración se la puede apreciar en la Figura 40.

```

^Cesla-dbpedia:~ nosql$ clear

esla-dbpedia:~ nosql$ mongoimport --host 172.18.4.172 --db crawler --collection tweets --file /Volumes/TOSHIBA
/DATOS/datosdescargados.json --jsonArray --numInsertionWorkers 150
2016-04-05T10:11:26.729-0500    connected to: 172.18.4.172
2016-04-05T10:11:29.728-0500    [.....] crawler.tweets      321.4 MB/105.2 GB (0.3%)
2016-04-05T10:11:32.728-0500    [.....] crawler.tweets      647.1 MB/105.2 GB (0.6%)
2016-04-05T10:11:35.729-0500    [.....] crawler.tweets      974.2 MB/105.2 GB (0.9%)
2016-04-05T10:11:38.729-0500    [.....] crawler.tweets      1.3 GB/105.2 GB (1.2%)
2016-04-05T10:11:41.729-0500    [.....] crawler.tweets      1.6 GB/105.2 GB (1.5%)
2016-04-05T10:11:44.729-0500    [.....] crawler.tweets      1.9 GB/105.2 GB (1.8%)
2016-04-05T10:11:47.729-0500    [.....] crawler.tweets      2.2 GB/105.2 GB (2.1%)
2016-04-05T10:11:50.729-0500    [.....] crawler.tweets      2.5 GB/105.2 GB (2.4%)
2016-04-05T10:11:53.728-0500    [.....] crawler.tweets      2.9 GB/105.2 GB (2.7%)
2016-04-05T10:11:56.729-0500    [.....] crawler.tweets      3.2 GB/105.2 GB (3.0%)
2016-04-05T10:11:59.729-0500    [.....] crawler.tweets      3.5 GB/105.2 GB (3.3%)
2016-04-05T10:12:02.729-0500    [.....] crawler.tweets      3.8 GB/105.2 GB (3.6%)
2016-04-05T10:12:05.729-0500    [.....] crawler.tweets      4.1 GB/105.2 GB (3.9%)
2016-04-05T10:12:08.729-0500    [#.....] crawler.tweets      4.5 GB/105.2 GB (4.3%)
2016-04-05T10:12:11.729-0500    [#.....] crawler.tweets      4.8 GB/105.2 GB (4.6%)
2016-04-05T10:12:14.728-0500    [#.....] crawler.tweets      5.1 GB/105.2 GB (4.9%)
2016-04-05T10:12:17.729-0500    [#.....] crawler.tweets      5.3 GB/105.2 GB (5.0%)
2016-04-05T10:12:20.729-0500    [#.....] crawler.tweets      5.3 GB/105.2 GB (5.0%)
2016-04-05T10:12:23.729-0500    [#.....] crawler.tweets      5.3 GB/105.2 GB (5.0%)

```

Figura 40. Optimización de importación de datos-MongoDB.  
Elaboración: Propia.

La Figura 41, muestra la finalización de la importación de datos con el tiempo optimizado, logrando su mejora.

```

2016-04-05T11:56:20.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:23.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:26.728-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:29.728-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:32.728-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:35.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:38.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:41.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:44.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:47.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:50.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:53.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:56.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:56:59.730-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:02.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:05.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:08.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:11.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:14.728-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:17.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:20.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:23.729-0500    [#####] crawler.tweets      105.2 GB/105.2 GB (100.0%)
2016-04-05T11:57:26.254-0500    imported 20000000 documents
esla-dbpedia:~ nosql$

```

Figura 41. Finalización de importación de datos optimizada-MongoDB.  
Elaboración: Propia.

En la Figura 42 se representa gráficamente la mejora que se obtiene en los tiempos de importación de datos, al emplear más procesos a la configuración.

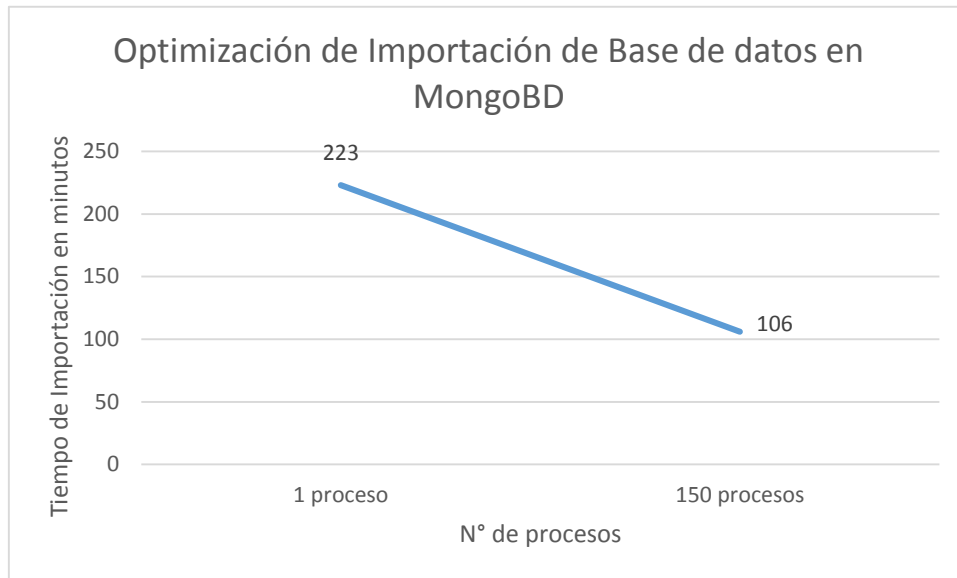


Figura 42. Mejora de Tiempo en la Importación de datos en MongoDB.  
Elaboración: Propia.

#### 4.2 Optimización en RethinkDB

La mejora en RethinkDB se ejecuta sobre la misma arquitectura como lo indica la Figura 16 y al igual que en MongoDB se basa en el tiempo de carga por medio del parámetro “clients” que se refiere al número de procesos que realiza la importación de los datos, por defecto viene con un valor de 8 procesos; aumentando la cantidad de este parámetro se logra el aumento de consumo de memoria RAM y por ende se obtiene mejor tiempo de carga. Dicha mejora se logra configurando el parámetro a 300 procesos. El detalle de dicha configuración se la muestra en la Figura 43.

```

Mac-Pro-de-Dark:~ nosql$ rethinkdb-import -f /Volumes/DiscoExterno/DATOS/datosdescargados.json
--table basecrawler.tweets --clients 300 --pkey id_str --shards 3
[=====] 100%
20000000 rows imported in 1 table
Done (6087 seconds)
Mac-Pro-de-Dark:~ nosql$

```

Figura 43. Optimización de carga de datos-RethinkDB  
Elaboración: Propia

En la Figura 44 se representa gráficamente la mejora que se obtiene en los tiempos de importación de datos, al emplear más procesos a la configuración.



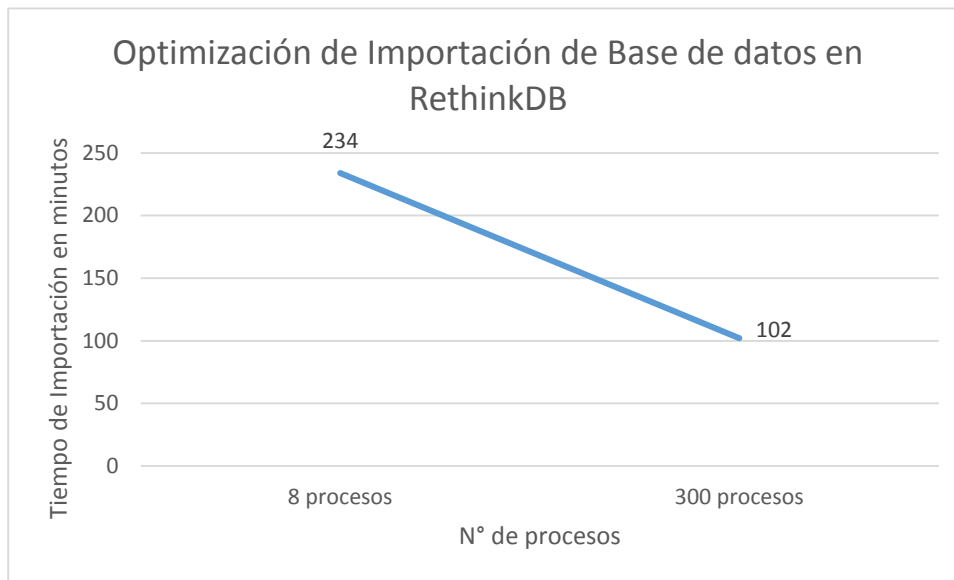


Figura 44. Mejora de Tiempo en la Importación de datos en RethinkDB.  
Elaboración: Propia.

### 4.3 Optimización en CouchDB

La mejora en CouchDB se ejecuta sobre la misma arquitectura como lo indica la Figura 17 y se la obtiene en almacenamiento en disco duro, a diferencia de los otros sistemas de bases de datos, ya que este producto de almacenamiento posee gran capacidad de compresión de los datos a través del parámetro *file\_compression*, por defecto viene configurado con la opción “snappy” el cuál es un eficiente y rápido compresor de los datos, pero con el fin de mejorar, se experimenta con otras opciones disponibles; una de ellas es “deflate\_N” donde N es un número que puede tener un rango desde el 1 al 9, siendo 1 el valor que permite rapidez en la carga de datos y nivel de compresión baja, en cambio el valor 9 ofrece lentitud en la carga de datos y nivel de compresión alta. Las pruebas se realizan con cualquier valor posible, obteniendo siempre mejores resultados a los alcanzados inicialmente, por lo tanto, la mejora se produce gracias a la opción “deflate\_N”, con el valor de compresión 1, logrando ahorrar aún más espacio en disco duro. El detalle de dicha configuración se la muestra en la Figura 45.

couchdb	attachment_stream_buffer_size	4294967296
	database_dir	/Volumes/DiscoExterno/basedatos
	delayed_commits	true
	file_compression	deflate_1

Figura 45. Configuración de mejora en el compresor de CouchDB.  
Elaboración: Propia.

En el gráfico 46 se representa gráficamente la mejora que se obtiene en el tamaño de la Base de datos, al cambiar el tipo de compresor.

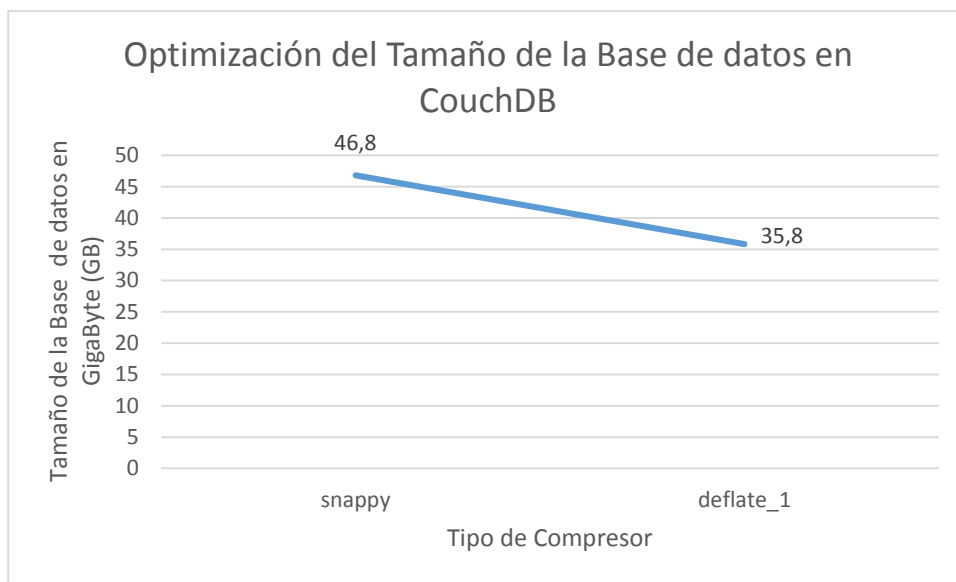


Figura 46. Mejora en el Tamaño de Base de datos en CouchDB.  
Elaboración: Propia.

#### 4.4 Recopilación de Resultados.

Una vez realizada la mejora en los criterios de tiempos de carga y espacio en disco duro de cada base de datos, se muestra en la Tabla 11 los nuevos resultados que se han obtenido, en comparación con las pruebas piloto realizadas, logrando optimizar el entorno distribuido.

Tabla 11. Optimización de Pruebas Piloto.

Base de datos	Resultados en pruebas piloto		Resultados optimizados	
	Tiempo de carga	Espacio ocupado en disco	Tiempo de carga	Espacio ocupado en disco
MongoDB	223 min	137,2 GB	106 min	137,2 GB
CouchDB	245 min	46,8 GB	245 min	35,8 GB
RethinkDB	234 min	127,8 GB	102 min	127,8 GB

Elaboración: Propia.

#### 4.5 Elección Final.

Una vez lograda la optimización en cada una de las bases de datos, ahora corresponde realizar la elección del sistema que se recomienda para su implementación. Para ello se realiza el análisis respectivo considerando los resultados optimizados como se muestra en la Tabla 11.

CouchDB ofrece un rendimiento menor en comparación a las otras bases de datos, la importación de datos no se logra optimizar aun aumentando recursos del servidor; Esta base de datos no cuenta con un comando propio para realizar la carga, por esta razón se procede a buscar herramientas extras para lograr su importación, además no ofrece fragmentación lo cual se considera otro punto en contra, pues esta funcionalidad ofrece más escalabilidad porque requiere de una sincronización de todos los nodos para la repartición de datos, la ventaja principal radica en su gran nivel de compresión para ahorrar considerablemente el espacio en disco duro, como se muestra los resultados en la gráfica 47.

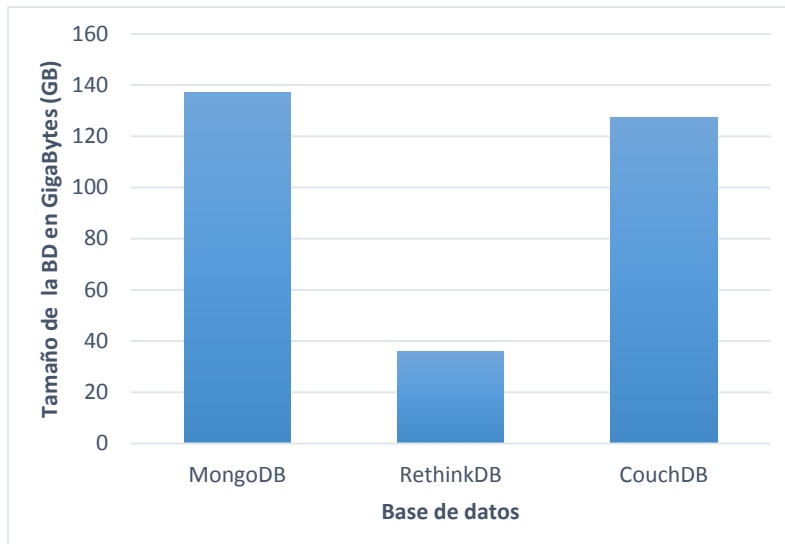


Figura 47. Comparación del Tamaño de la base de datos en cada Sistema NoSQL.  
Elaboración: Propia.

Si bien es cierto RethinkDB presenta mejores resultados en el tiempo de importación de datos como se muestra en la Figura 48, en el proceso de distribución ofrece un desequilibrio notorio con respecto a los datos que almacena el servidor principal y el resto de nodos, para ello se realiza el rebalanceo de datos por medio de la función *rebalance*<sup>19</sup>, con el objetivo de obtener un almacenamiento uniforme en cada servidor, este proceso requiere un determinado tiempo adicional lo que provoca la principal desventaja frente a MongoDB.

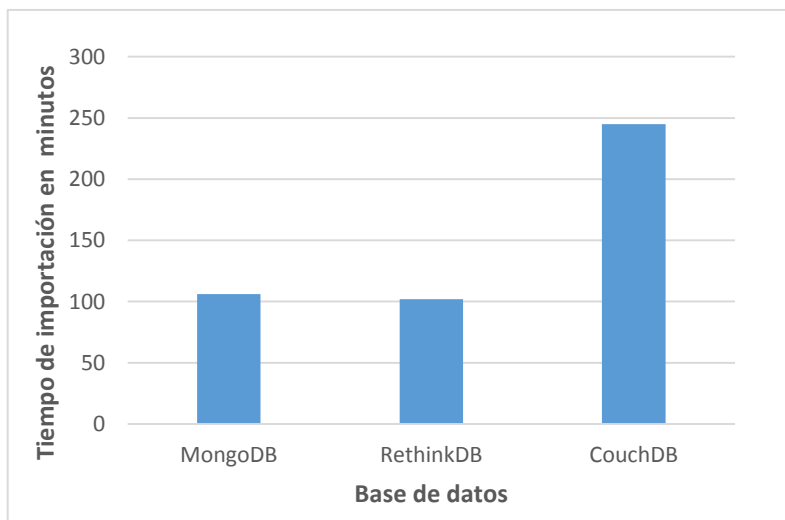


Figura 48. Comparación del Tiempo de importación en cada Base de datos.  
Elaboración: Propia.

<sup>19</sup> Información sobre comando rebalance disponible en: <https://www.rethinkdb.com/api/python/rebalance/>

Finalmente MongoDB es la base de datos que se recomienda en este Trabajo de Titulación para su implementación, pues satisface todos los requerimientos, considerando los resultados en la tabla 11 de optimización; MongoDB requiere más espacio en disco en comparación a las otras bases de datos pero en cambio ofrece optimización en la escalabilidad, si bien es cierto en el tiempo de carga se ve levemente superado por pocos minutos por RethinkDB, MongoDB ofrece una distribución uniforme sin necesitar de tiempo extra como ocurre en RethinkDB, lo que permite obtener su principal ventaja, además en lo que respecta a la transparencia al usuario, este requerimiento se logra mediante un proceso sencillo al gestionar sus nodos eficientemente, obteniendo la redistribución automática de los datos en cada servidor del clúster.

## TRABAJOS A FUTURO

Una vez culminado el presente trabajo de titulación lo que se propone como línea a seguir en el futuro es optimizar la recuperación de grandes cantidades de información para complementar lo desarrollado, para ello se puede elegir la base de datos que se ha utilizado en este trabajo, que tiene una cantidad de información suficiente para poder ejecutar las pruebas necesarias de recuperación.

Cómo recomendación, se puede optar por utilizar los comandos disponibles en cada base para la extracción de datos o también sería importante analizar dichas pruebas utilizando los distintos lenguajes de programación que soporta cada base, de ahí viene la importancia de siempre tener como referencia principal la documentación oficial de cada sistema para encontrar las soluciones al problema propuesto.

Otra opción como trabajo a futuro es el desarrollo de una aplicación donde se utilice como repositorio principal a una de las Bases de datos documentales aplicadas en el presente trabajo para la gestión de la información, con el objetivo principal de conocer cómo se crea el modelado de datos y se relaciona la información en un sistema NoSQL, si es por colecciones o por documentos; para ello se puede utilizar los distintos complementos que existen en los entornos de desarrollo como Java, JavaScript, php, entre otros, una opción puede ser mongoose<sup>20</sup> que es una librería para trabajar MongoDB en NodeJS y permite la relación de datos.

---

<sup>20</sup> Sitio oficial de la librería mongoose: <http://mongoosejs.com/>.

## CONCLUSIONES

Culminando con el Trabajo de Titulación referente a la implementación de las Bases de datos NoSQL en entornos distribuidos se obtiene las siguientes conclusiones:

- Las bases de datos candidatas fueron elegidas en base a la búsqueda de criterios que satisfaga los requerimientos del Trabajo de Titulación, enfocándose principalmente en la capacidad de implementar el sistema de almacenamiento en entornos distribuidos a través de la replicación o distribución de datos.
- La elección de la base de datos recomendada para su implementación, se realizó en base a dos criterios fundamentales: tiempo de carga y tamaño de la base de datos en disco duro, complementándose a las características que soportan la replicación y distribución de datos, y a la transparencia al usuario en la reconfiguración de nodos del clúster, lo que permite garantizar la escalabilidad de bases de datos NoSQL.
- La versión actual de la base de datos CouchDB no soporta la funcionalidad fragmentación y tomando en cuenta el tiempo de carga de los resultados obtenidos, se determina que este sistema se recomienda cuando se cuenten con equipos que no dispongan de grandes recursos (RAM, Disco Duro, etc), y en aspectos de seguridad, ya que mediante la replicación se almacenan los datos en su totalidad en cada servidor configurado, además por su nivel de compresión permite el ahorro de disco duro.
- Las bases de datos RethinkDB y MongoDB son muy parecidas en cuanto a la funcionalidad que ofrecen a los usuarios y ambas pueden optimizarse contando con grandes recursos a nivel de hardware (RAM, disco duro, etc), en este aspecto, MongoDB obtuvo mejores resultados por la distribución uniforme de los datos en todos los nodos; el tamaño de la base de datos en ambos sistemas aproximadamente es de 3 veces más de lo que ocupa CouchDB, finalmente en la configuración del entorno distribuido los dos sistemas necesitan de un proceso sencillo, más en RethinkDB, por esta razón en MongoDB se requiere más investigación de su documentación.
- Todas las Bases de datos NoSQL apoyan la filosofía de escalamiento horizontal, dando más prioridad a la cantidad de nodos que constituyen un entorno distribuido que a los recursos de cada servidor, es por eso que inicialmente, las bases de datos utilizadas en

el Trabajo de Titulación por defecto, están configuradas para el uso mínimo de recursos de los equipos, pero cada una cuenta con parámetros configurables que permite obtener mejores resultados, optimizando los recursos disponibles.

- Mientras más nodos se agreguen al entorno distribuido más escalabilidad y ahorro de recursos se obtiene, pero el rendimiento en la carga y en la distribución o replicación de los datos decrece, ya que la totalidad de los datos sin la configuración del sistema distribuido demora entre 40 y 45 minutos su carga, por otro lado, con el entorno distribuido tomando como referencia los datos de la tabla 11 se requiere de 151 minutos promedio.
- Para realizar la comunicación entre las distintas bases de datos NoSQL aplicadas en el Trabajo de Titulación no se requiere que todas se ejecuten sobre un mismo Sistema Operativo ya que, la conexión se realiza a través de los protocolos de red HTTP Y TCP.
- El contenido del presente trabajo de titulación es muy importante ya que pretende mostrar especialmente a los desarrolladores de software una nueva forma de almacenar y gestionar los datos en sus aplicaciones, más ahora en una era donde las redes sociales predominan en la web y requieren de millones de peticiones cada instante debido a su gran cantidad de usuarios que las usan, lo que genera un incremento considerable de información, y gracias a las bases de datos NoSQL mantienen su rendimiento y eficiencia debido a la facilidad con que se puede implementarla en un entorno distribuido sin la necesidad de invertir en supercomputadores y la mayoría de estas son de código libre.



## RECOMENDACIONES

Una vez finalizado con las conclusiones del Trabajo de Titulación, se propone las siguientes recomendaciones, con el objetivo de obtener mejores resultados en su implementación:

- Antes de realizar la configuración de los parámetros que producen la optimización del entorno distribuido en MongoDB (numInsertionsWorkers) y RethinkDB (clients), se recomienda revisar las especificaciones con los que cuenta cada servidor, ya que dichos parámetros consumen recursos del sistema (RAM, procesador, etc) conforme al valor que se asigne.
- Para la sincronización exitosa de los Config Servers se necesita que la hora entre todos los servidores sea la misma, o que exista una diferencia de hasta 10 segundos entre uno y otro(s) servidor. Para ello puede hacer uso del comando NTP (Network Time Protocol) y realizar la sincronización automática, o en su defecto cambiar la hora manualmente.
- Se recomienda que los servidores pertenezcan a la misma red, para facilitar la sincronización entre uno y otro nodo, así como también desactivar cortafuegos (firewall) con el objetivo de que no existan problemas de comunicación, caso contrario puede gestionar las conexiones por medio del comando iptables y habilitar las ip de cada servidor con su respectivo puerto.
- Si utiliza un disco externo para el almacenamiento y gestión de los datos, se recomienda que los formatos de estos sean compatibles con el Sistema Operativo que usa el servidor, para evitar problemas de pérdida de información en el caso de producirse cortes de luz inesperados ya que su recuperación no se realiza automáticamente, provocando problemas en el inicio de la base de datos.
- Utilizar siempre las versiones disponibles de 64 bits en cada base de datos cuando se desee implementarla en entornos de producción, debido a que las versiones de 32 bits cuentan con limitaciones que no permiten cumplir con los objetivos.

- Si los datos de origen están almacenados en una base de datos relacional y quiere migrar a NoSQL se recomienda para elegir el tipo ideal, tomar en cuenta lo que desea hacer con los datos, en el Trabajo de Titulación el objetivo principal era la escalabilidad en varios servidores y el formato elegido (Documentos) era el más idóneo para cumplir con las metas, ya que la mayoría de sistemas de este tipo permiten el almacenamiento en varios nodos.

## BIBLIOGRAFÍA

Brewer, E. (2000). Towards Robust Distributed Systems. Recuperado 5 de octubre de 2015, a partir de <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>.

Buettrich, S., & Escudero, A. (2007). Topología e Infraestructura Básica de Redes Inalmámbricas. Recuperado 10 de agosto de 2015, a partir de [http://www.itrainonline.org/itrainonline/mmtk/wireless\\_es/files/04\\_es\\_topologia-e-infraestructura\\_guia\\_v02.pdf](http://www.itrainonline.org/itrainonline/mmtk/wireless_es/files/04_es_topologia-e-infraestructura_guia_v02.pdf).

DB-Engines. (2015). DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. Recuperado 1 de agosto de 2015, a partir de <http://db-engines.com/en/>.

Fernandez, R. (2014). NoSQL: Clasificación de las bases de datos según el teorema CAP. Recuperado 5 de octubre de 2015, a partir de <http://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap>.

FITEC. (2011). Topologías de Red. Recuperado 10 de agosto de 2015, a partir de <http://csudp.wikispaces.com/file/view/Topologias+de+Red.pdf>.

Garcete, A. (2014). Base de Datos Orientado a Columnas. Recuperado 6 de octubre de 2015, a partir de <http://jeuazarru.com/wp-content/uploads/2014/10/dbco.pdf>.

Garrido, A. (2007). *Estudio y Análisis de la evolución de la topología de internet mediante las tablas BGP*. Universidad Pontificia Comillas, Madrid. Recuperado a partir de <http://www.iit.upcomillas.es/pfc/resumenes/46811d3ec4c65.pdf>.

Gutiérrez Pérez, E. (2014). Aplicación de persistencia políglota usando tecnología NoSQL. Recuperado a partir de <https://addi.ehu.es/bitstream/10810/12678/1/AplicacionPersistenciaPoliglotaPFC.pdf>.

Hernández, R. (2009). CouchDB: Introducción. Recuperado 19 de octubre de 2015, a partir de <http://rafinguer.blogspot.com/2009/12/couchdb-introduccion.html>.

Moniruzzaman, A., & Akhter, S. (2013). NoSQL Database: New Era of Databases for Big data Analytics -Classification, Characteristics and Comparison. Recuperado 16 de diciembre de 2015, a partir de <http://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf>.

RethinkDB. (2016). Introduction to ReQL - RethinkDB. Recuperado 9 de febrero de 2016, a partir de <https://www.rethinkdb.com/docs/introduction-to-reql/>.

Rocha, L., Vale, F., Cirilo, E., Barbosa, D., & Mourão, F. (2015). A Framework for Migrating Relational Datasets to NoSQL. Recuperado 10 de abril de 2016, a partir de <http://www.sciencedirect.com/science/article/pii/S1877050915011758>.

Rodríguez, D. (2016). Análisis de Grafos en paralelo mediante Graphx. Universidad Técnica Particular de Loja, Loja. Recuperado a partir de <http://dspace.utpl.edu.ec/bitstream/123456789/14555/1/RODRIGUEZ%20BAUTISTA%20DANIEL%20IGNACIO.pdf>

Rodríguez, L. (2011). Software: Sistemas Operativos y Aplicaciones. Recuperado 9 de agosto de

2015, a partir de <https://www.uclm.es/profesorado/licesio/Docencia/IB/IBTema3a.pdf>.

Sánchez, Á. (2015). ¿Qué es una tarjeta de red? Recuperado 11 de agosto de 2015, a partir de <http://computadoras.about.com/od/redes/fl/que-Es-Una-Tarjeta-De-Red.htm>.

Sandoval, E. (2011). Topologías de Red. Recuperado 10 de agosto de 2015, a partir de [http://www.uaeh.edu.mx/docencia/P\\_Presentaciones/huejutla/sistemas/redes/topologias.pdf](http://www.uaeh.edu.mx/docencia/P_Presentaciones/huejutla/sistemas/redes/topologias.pdf).

Shankar, S. (2014). Big Data - News, Views and Reviews: NoSQL databases characteristics comparison. Recuperado 15 de febrero de 2016, a partir de <http://www.infoivy.com/2014/05/nosql-databases-mongodb-simpledb-etc.html>.

Sommerville, I. (2005). *Ingeniería del Software*. (7ma ed.). Madrid: Pearson Educación, S.A. Recuperado a partir de <http://zeus.inf.ucv.cl/~bcrawford/Modelado%20UML/Ingenieria%20del%20Software%207ma.%200Ed.%20-%20Ian%20Sommerville.pdf>.

Universidad de Azuay. (2002). Comunicaciones y Redes de Computadoras. Recuperado 10 de agosto de 2015, a partir de [http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/apuntes\\_1/](http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/apuntes_1/).

Vazquez, Y., & Sotolongo, A. (2013). Mirada a bases de datos NoSQL de código abierto orientadas a documentos. Recuperado 2 de agosto de 2015, a partir de <http://publicaciones.uci.cu/index.php/SC/article/view/1382/720>.

Zambrano, M. (2015). Levantamiento, definición e implementación de la capa arquitectónica de sistemas de información del Banco de Loja, utilizando la descripción del modelado arquitectónico ADM/TOGAF. Universidad Técnica Particular de Loja, Loja. Recuperado a partir de [http://dspace.utpl.edu.ec/bitstream/123456789/12869/1/Zambrano\\_Mora\\_Mario\\_Alexander.pdf](http://dspace.utpl.edu.ec/bitstream/123456789/12869/1/Zambrano_Mora_Mario_Alexander.pdf).

## **ANEXOS**

## ANEXO A. Comparativa General de Bases de datos NoSQL.

Nombre	Ranking	Licencia	Costo	Soporte	Modelo de Base de datos	Sistemas Operativos	Integración con Lenguajes de Programación	API	Tipo de Replicación que soporta	Soporte de Fragmentación
MarkLogic	1°- RDF Store 5°- Document stores 1° Native XML DBMS	Comercial y Libre (hasta 6 meses)	\$18K Enterprise	www.marklogic.com	Multi-modelo (Document store, Native XML DBMS, RDF store )	Linux, OS X, Solaris, Windows	C++	Si	Maestro-Esclavo	Si
Virtuoso	2°- RDF Store	Comercial y Libre	desde 999\$ hasta 72000\$	virtuoso.openlinksw.com	Multi-modelo (Native XML DBMS, RDF store)	AIX FreeBSD HP-UX Linux OS X Solaris Windows	Net C C# C++ Java JavaScript Perl PHP Python Ruby Visual Basic	Si	Maestro-Esclavo y Maestro-Maestro	No
AllegroGraph	5°- RDF Store	Comercial	\$99 Student Edition	www.franz.com/agraph/-allegrograph	RDF Store	Linux Windows OS X	Java Python C# Perl Ruby Lisp	Si	Maestro-Esclavo	No

RethinkDB	9°- Document Store	Open Source		<a href="http://www.rethinkdb.com">www.rethinkdb.com</a>	Document Store	Linux OS X	C C# C++ Erlang Java JavaScript (Node.js) Perl PHP Python Ruby	No	Maestro- Esclavo	Si
Riak	4°- Key-value store	Open Source		<a href="http://basho.com/products/riak-overview">basho.com/products/riak-overview</a>	Clave-Valor	Linux OS x	C C# C++ Erlang Haskell Java JavaScript Lisp Perl PHP Python Ruby	Si	Maestro- Esclavo	Si
Neo4j	1° Graph Store	Open Source		<a href="http://neo4j.com">neo4j.com</a>	Graph Store	Linux OS X Windows	.Net Clojure Go Groovy Java JavaScript Perl PHP Python Ruby Scala	Si	Maestro- Esclavo	No
MongoDB	1°- Document Store	Open Source		<a href="http://www.mongodb.org">www.mongodb.org</a>	Document store	Linux OS X Solaris Windows	C C# Erlang Haskell Java JavaScript	Si	Maestro- Esclavo	Si

							Lisp Lua Perl PHP PL/SQL Python Ruby			
CouchDB	2°- Document Store	Open Source		<a href="http://couchdb.apache.org">couchdb.apache.org</a>	Document Store	Android BSD Linux OS X Solaris Windows	C C# Erlang Java JavaScript Lisp Perl PHP PL/SQL Python Ruby	Si	Maestro-Maestro	No
Terrastore	22°- Document Store	Open Source		<a href="http://code.google.com/p/terrastore">code.google.com/p/terrastore</a>	Document Store	Windows	Java	Si	Maestro-Maestro	No
Redis	1°- Key-value stores	Open Source		<a href="http://redis.io">redis.io</a>	Key-value stores	BSD Linux OS X Windows	C C# C++ Erlang Java JavaScript (Node.js) Lisp Lua MatLab OCaml Perl PHP Prolog Pure Data Python Ruby	Si	Maestro-Esclavo	Si



Cassandra	1°- Wide column store	Open Source		cassandra.apache.org	Wide column store	Linux OS X Windows	C# C++ Clojure Erlang Go Haskell Java JavaScript Perl PHP Python Ruby	Si	Maestro- Esclavo	Si
-----------	-----------------------	-------------	--	----------------------	-------------------	--------------------------	--	----	---------------------	----

## ANEXO B. Instalación de mongodb en Mac OS X

1. Descarga de la versión que se requiera a través del siguiente comando:

```
curl -O https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.0.10.tgz
```

2. Descomprime y extrae los archivos descargados

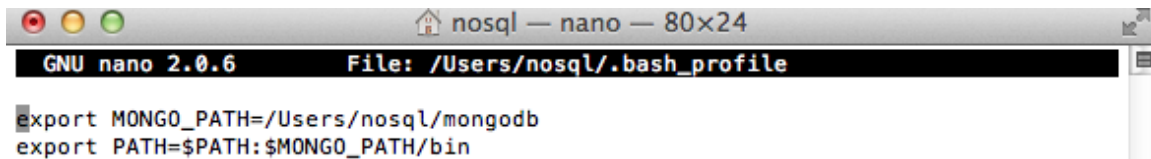
```
tar -zxvf mongodb-osx-x86_64-3.0.10.tgz
```

3. Crea un directorio y copia los archivos anteriormente extraídos.

```
mkdir -p mongodb  
"cp -R -n mongodb-osx-x86_64-3.0.10/ mongodb"
```

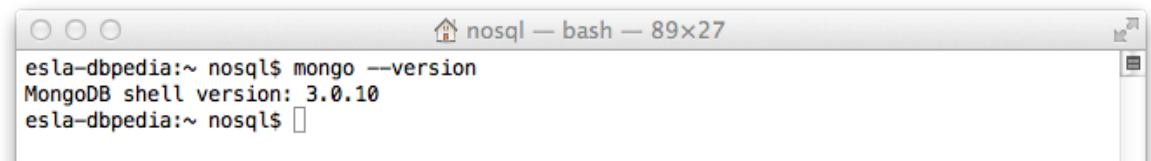
4. Agrega los archivos ejecutables de mongodb al \$PATH para acceder directamente a través del terminal de comandos.

- a. Se procede a abrir el archivo ~/.bash\_profile en el directorio del usuario y agrega al PATH la dirección del directorio de mongodb.



```
nosql — nano — 80x24  
GNU nano 2.0.6 File: /Users/nosql/.bash_profile  
export MONGO_PATH=/Users/nosql/mongodb  
export PATH=$PATH:$MONGO_PATH/bin
```

5. Verifica la instalación de mongodb, comprobando su versión.



```
nosql — bash — 89x27  
esla-dbpedia:~ nosql$ mongo --version  
MongoDB shell version: 3.0.10  
esla-dbpedia:~ nosql$
```

## ANEXO C. Instalación de MongoDB en Ubuntu

1. Importa la clave pública que ofrece mongodb para que sea utilizada por el sistema de gestión de paquetes de Ubuntu

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
7F0CEB10
```

2. Crea el archivo donde se configura el repositorio de mongo con el siguiente comando.

```
echo "deb http://repo.mongodb.org/apt/ubuntu precise/mongodb-org/3.0  
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
```

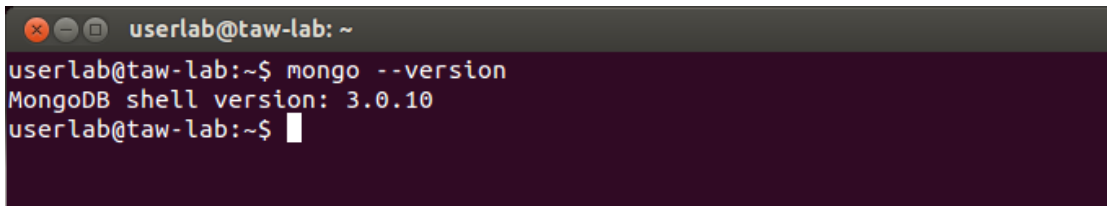
3. Actualiza el repositorio con el siguiente comando:

```
sudo apt-get update
```

4. Instala mongodb

```
sudo apt-get install -y mongodb-org
```

5. Verifica la instalación, comprobando la versión.

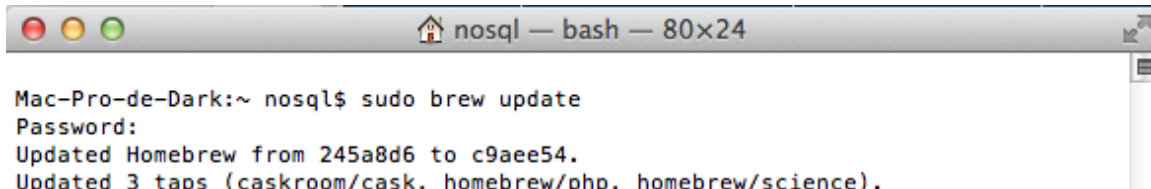


```
userlab@taw-lab: ~  
userlab@taw-lab:~$ mongo --version  
MongoDB shell version: 3.0.10  
userlab@taw-lab:~$
```

## ANEXO D. Instalación de couchdb en Mac OS X

La instalación en Mac se la puede realizar por medio del terminal a través de la herramienta HomeBrew con los siguientes pasos:

1. Actualiza los paquetes de homebrew



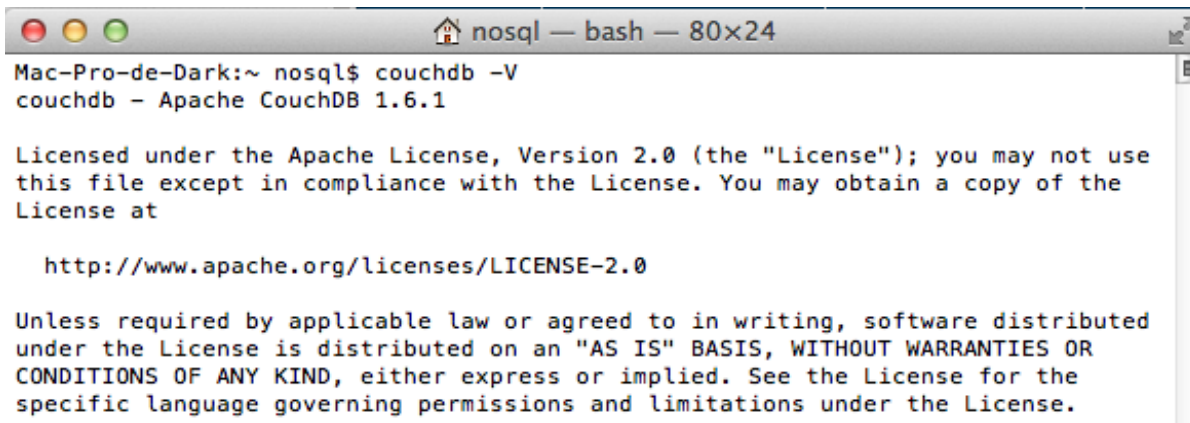
```
Mac-Pro-de-Dark:~ nosql$ sudo brew update
Password:
Updated Homebrew from 245a8d6 to c9aee54.
Updated 3 taps (caskroom/cask, homebrew/php, homebrew/science).
```

2. Instala couchdb a través de brew

```
sudo brew install couchdb
```

NOTA: Al momento de ejecutar este comando también se instalan todos los paquetes necesarios para que couchdb pueda funcionar correctamente

Por último verificamos la versión de CouchDb instalada



```
Mac-Pro-de-Dark:~ nosql$ couchdb -V
couchdb - Apache CouchDB 1.6.1

Licensed under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License. You may obtain a copy of the
License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
```

## ANEXO E. Instalación couchdb en ubuntu

1. Instalamos CouchDB a través de la consola con el siguiente comando:

```
sudo apt-get install couchdb
```

NOTA: Al momento de ejecutar este comando también se instalan todos los paquetes necesarios para que couchdb pueda funcionar correctamente.

2. En algunas versiones de Ubuntu no se instala la última versión, para ello se procede actualizar por medio del siguiente procedimiento:

### 2.1 Instala las últimas propiedades de Python

```
sudo apt-get install python-software-properties -y
```

### 2.2 Agrega la última clave pública de CouchDB (PPA)

```
sudo add-apt-repository ppa:couchdb/stable -y
```

### 2.3 Actualiza los paquetes

```
sudo apt-get update -y
```

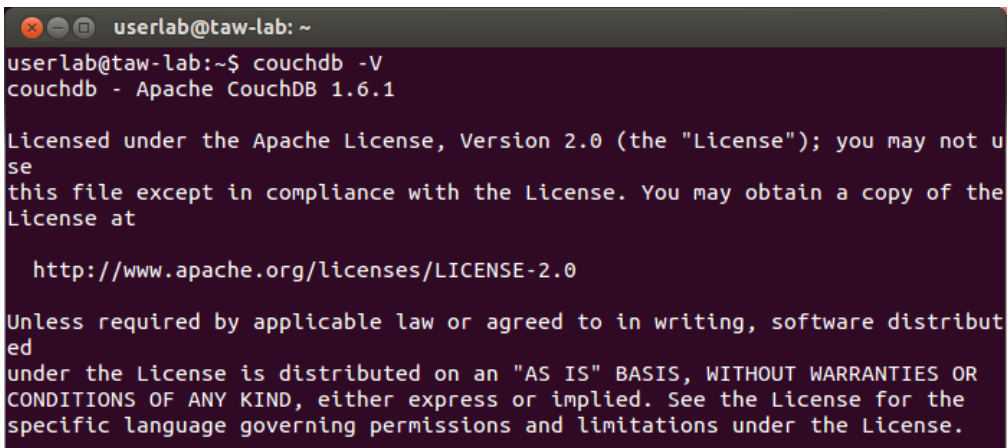
### 2.4 Elimina la versión anteriormente instalada

```
sudo apt-get remove couchdb couchdb-bin couchdb-common -yf
```

### 2.5 Instala couchDB 1.6

```
sudo apt-get install couchdb
```

3. Verificar la actualización de CouchDB.

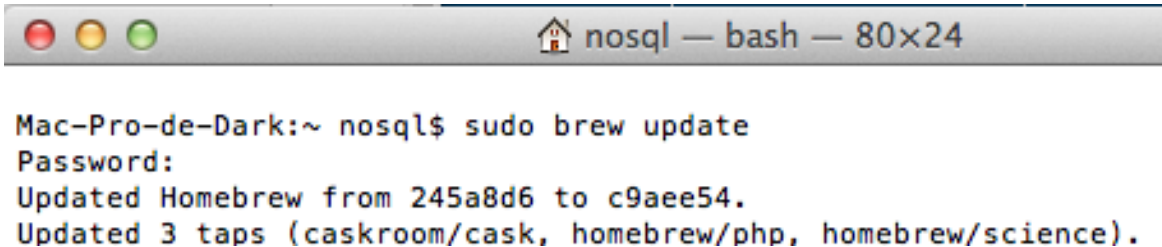


```
userlab@taw-lab: ~  
userlab@taw-lab:~$ couchdb -V  
couchdb - Apache CouchDB 1.6.1  
  
Licensed under the Apache License, Version 2.0 (the "License"); you may not use  
this file except in compliance with the License. You may obtain a copy of the  
License at  
  
http://www.apache.org/licenses/LICENSE-2.0  
  
Unless required by applicable law or agreed to in writing, software distributed  
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR  
CONDITIONS OF ANY KIND, either express or implied. See the License for the  
specific language governing permissions and limitations under the License.
```

## ANEXO F. Instalación de rethinkdb en Mac OS X

La instalación en Mac se la puede realizar por medio del terminal a través de la herramienta HomeBrew con los siguientes pasos:

1. Actualiza los paquetes de homebrew



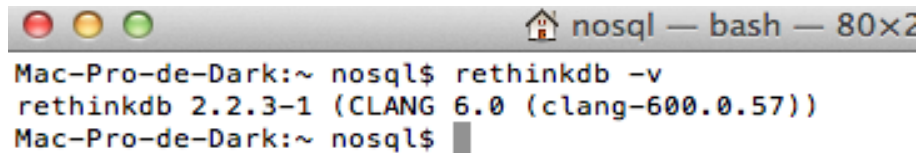
```
Mac-Pro-de-Dark:~ nosql$ sudo brew update
Password:
Updated Homebrew from 245a8d6 to c9aee54.
Updated 3 taps (caskroom/cask, homebrew/php, homebrew/science).
```

2. Instala rethinkdb a través de brew

```
sudo brew install rethinkdb
```

NOTA: Al momento de ejecutar este comando también se instalan todos los paquetes necesarios para que rethinkdb pueda funcionar correctamente.

3. Verifica la instalación, comprobando la versión instalada



```
Mac-Pro-de-Dark:~ nosql$ rethinkdb -v
rethinkdb 2.2.3-1 (CLANG 6.0 (clang-600.0.57))
Mac-Pro-de-Dark:~ nosql$
```

## ANEXO G. Instalación rethinkdb en ubuntu

- a. Agrega el repositorio de rethinkdb al paquete de Ubuntu

```
source /etc/lsb-release && echo "deb http://download.rethinkdb.com/apt
$DISTRIB_CODENAME main" | sudo tee /etc/apt/sources.list.d/rethinkdb.list
```

- b. Se realiza la descarga de la llave pública de RethinkDB.

```
wget -qO- https://download.rethinkdb.com/apt/pubkey.gpg | sudo apt-key add -
```

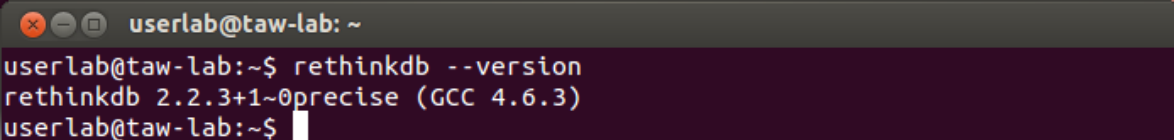
- c. Actualiza el repositorio de Ubuntu

```
sudo apt-get update
```

- d. Instala Rethinkdb

```
sudo apt-get install rethinkdb
```

- e. Verifica la versión de RethinkDB



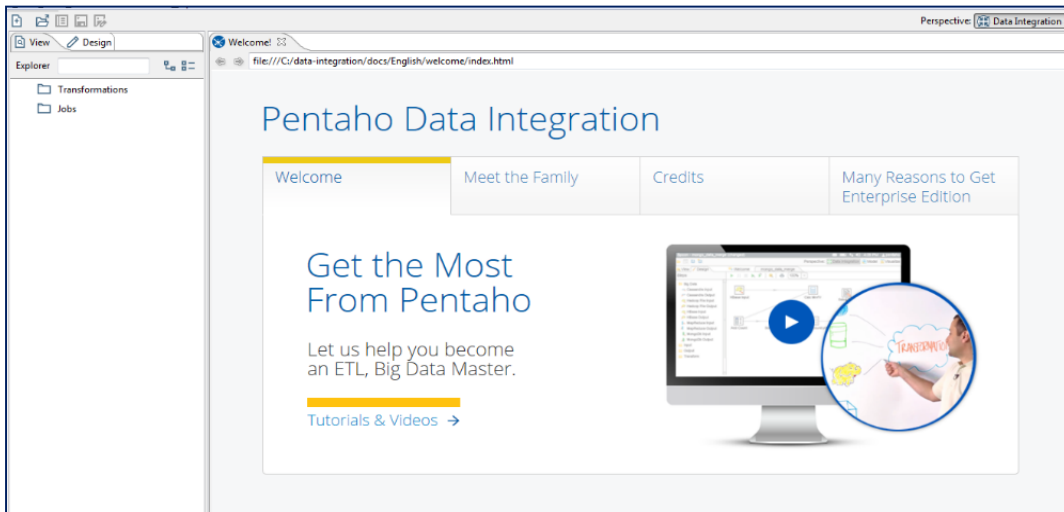
```
userlab@taw-lab: ~
userlab@taw-lab:~$ rethinkdb --version
rethinkdb 2.2.3+1~0precise (GCC 4.6.3)
userlab@taw-lab:~$
```

## ANEXO H. Transformación de datos Relacional a No Relacional

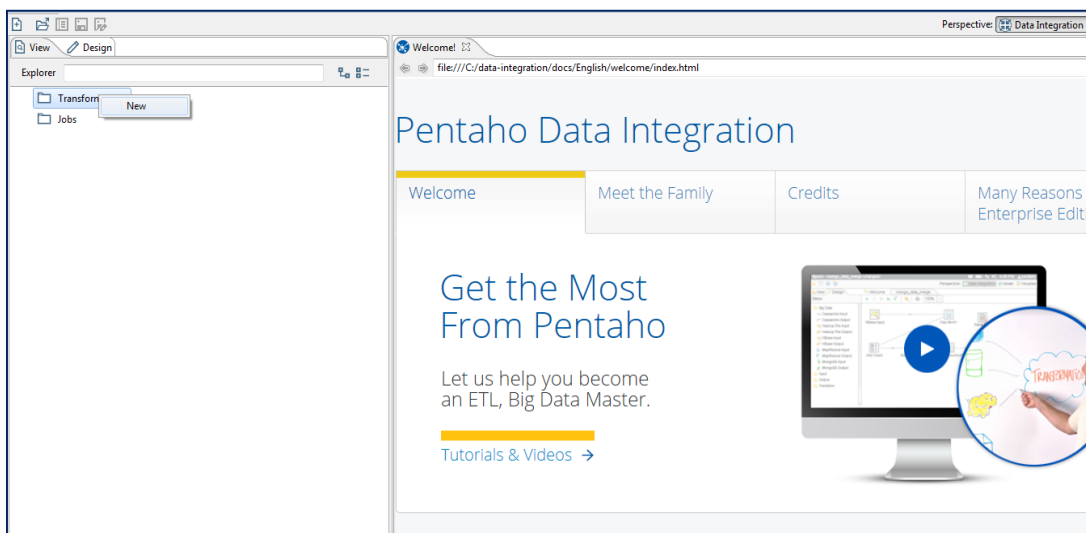
- 1) Ejecutar la aplicación Pentaho, para ello dentro de la carpeta “data-integration” se encuentra el archivo spoon.sh y a través de la consola de comandos ingresar la siguiente sentencia:

```
sh spoon.sh
```

Después se abrirá la siguiente interfaz

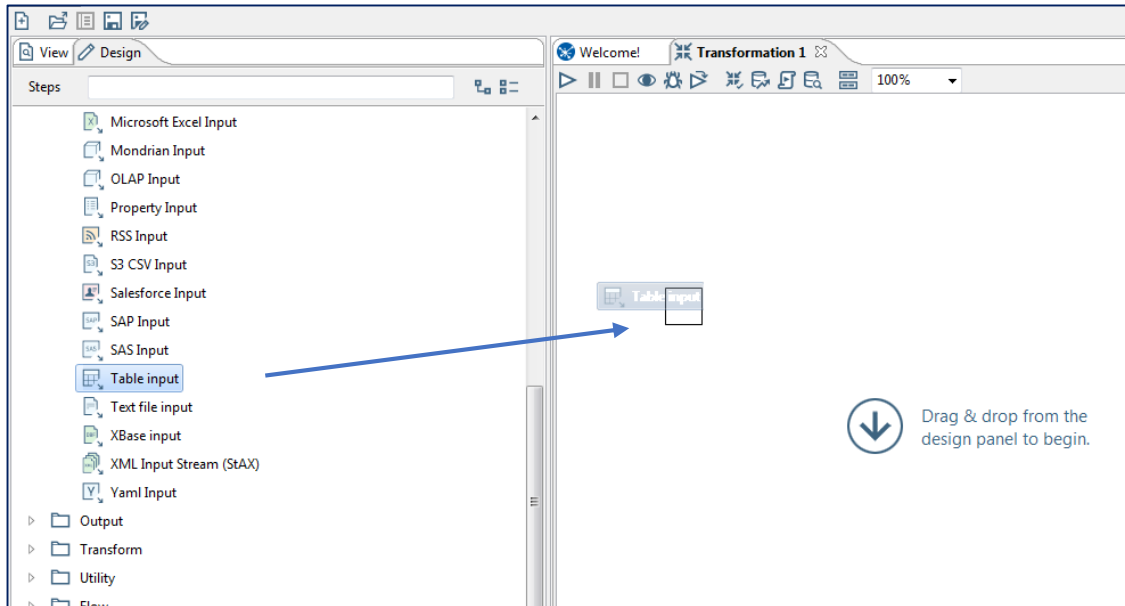


- 2) Click derecho en la carpeta “Transformations” y seleccionamos “New”

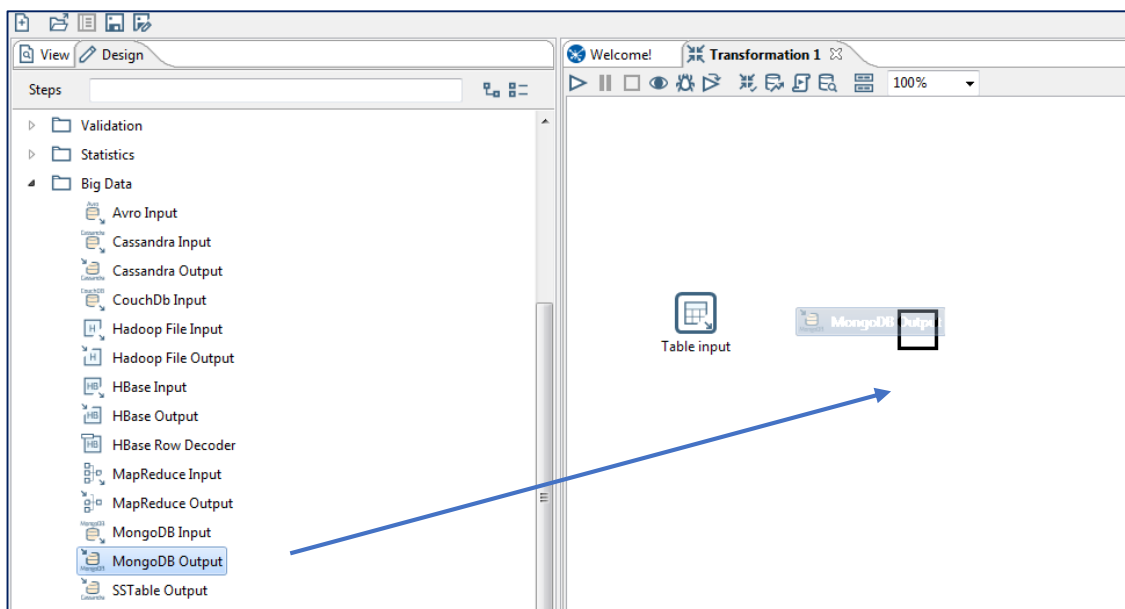




- 3) A continuación aparecerá la interfaz correspondiente al panel de diseño, aquí se configura los pasos necesarios para realizar la transformación, en primera instancia seleccionamos la carpeta Input y arrastramos la opción “Table Input” hacia la parte izquierda de la pantalla.



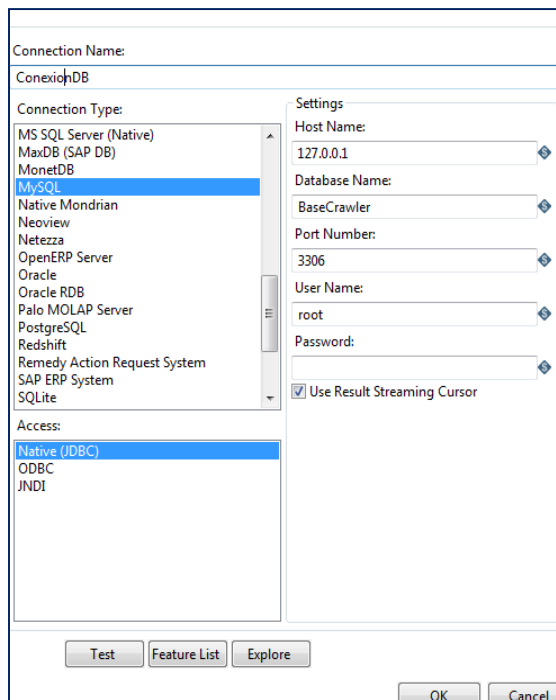
Así mismo seleccionamos la carpeta BigData y arrastramos la opción “Mongo Output” hacia la parte izquierda de la pantalla



- 4) Ahora se procede a configurar cada paso, para ello hacemos doble click en “Table input” y creamos una nueva conexión seleccionando en “New”



Colocamos un nombre para identificar la nueva conexión y los datos para tener acceso a la Base de datos SQL(host, puerto, etc)



Cabe recalcar que también podemos colocar sentencias SQL en este paso, en nuestro caso resulta necesario para poder extraer los 20 millones de registros en lugar de toda la Base de datos, la sentencia sql que utilizamos fue la siguiente:

```
SELECT * FROM Tweets LIMIT 0,20000000
```

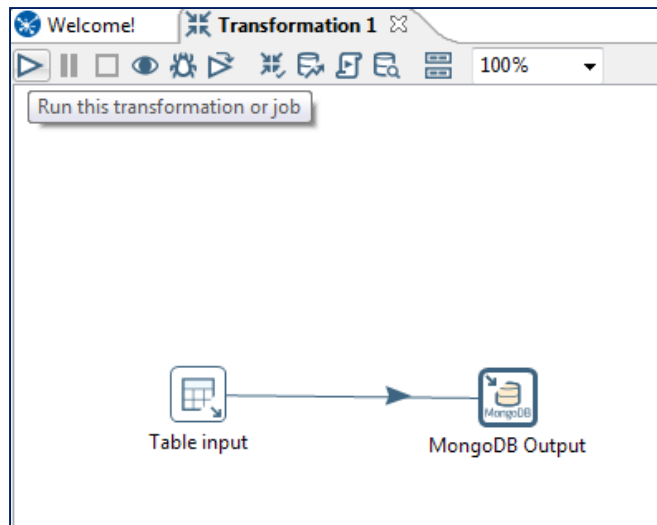
- 5) Ahora corresponde configurar el siguiente proceso, para ello hacemos doble click en MongoDB Output y colocamos los datos para poder acceder a MongoDB (host, puerto, etc).

Step name		MongoDB Output
Configure connection	Output options	Mongo document fields
Host name(s) or IP address(es)		localhost
Port		27017
Use all replica set members/mongos		<input type="checkbox"/>
Username		
Password		
Authenticate using Kerberos		<input type="checkbox"/>
Connection timeout		
Socket timeout		

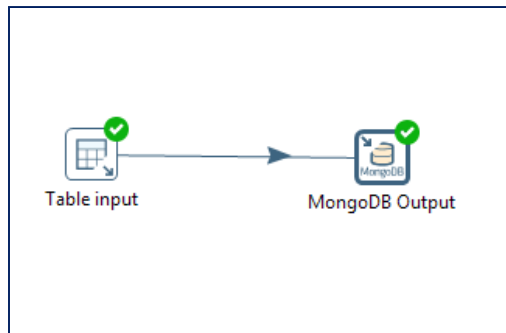
En la pestaña “Output options” colocamos el nombre de la Base de datos y la colección en donde queremos que se guarden los datos.

Step name		MongoDB Output
Configure connection	Output options	Mongo document fields
Database		BaseCrawler
Collection		tweets
Batch insert size		100

6) Por último guardamos la transformación y hacemos click en el ícono de ejecutar



Si no existen errores, en cada paso configurado se marcará con un visto, en señal de la transformación fue exitosa



## ANEXO I. Configuración de fragmentación de datos en MongoDB.

Para implementar la funcionalidad de fragmentación de datos en MongoDB se debe realizar los siguientes pasos:

- 1) Crear directorios para el almacenamiento de las instancias de los config Servers y Shards.
- 2) Iniciar cada instancia correspondiente a los config server, a través del siguiente comando, por defecto se inicia en el puerto 27019.

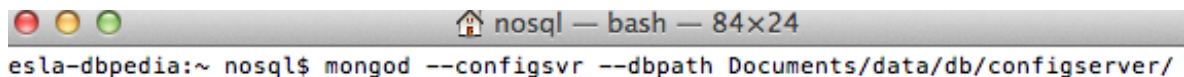
```
mongod -configsvr -dbpath directorio anteriormente creado
```

**Donde:**

**configsvr:** Indica el rol que tiene dicha instancia.

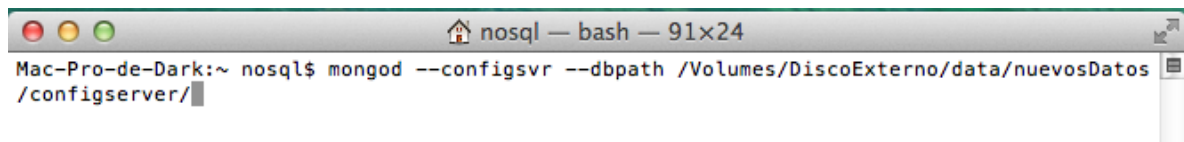
**dbpath:** Directorio donde se almacena los datos.

### Inicio de instancia Config Server en el Primer Servidor.



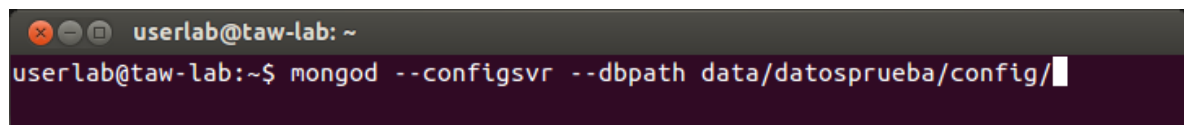
```
esla-dbpedia:~ nosql$ mongod --configsvr --dbpath Documents/data/db/configserver/
```

### Inicio de instancia Config Server en el Segundo Servidor.



```
Mac-Pro-de-Dark:~ nosql$ mongod --configsvr --dbpath /Volumes/DiscoExterno/data/nuevosDatos /configserver/
```

### Inicio de instancia Config Server en el Tercer Servidor.



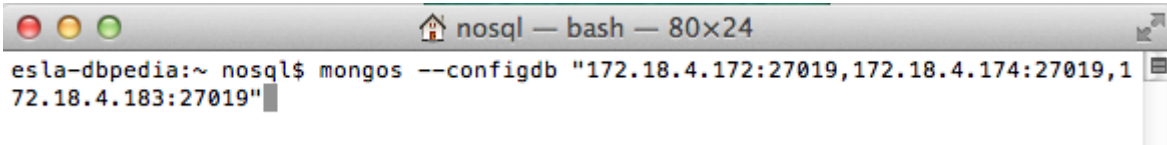
```
userlab@taw-lab:~$ mongod --configsvr --dbpath data/datosprueba/config/
```

- 3) Sincronizar cada instancia config Server iniciada con el siguiente comando, este proceso solo se debe aplicar en el servidor Principal.

```
mongos -configdb hostname1:puerto, hostname2:puerto,  
hostname3:puerto
```

**Donde:**

**Configdb:** Indica la configuración de la base de datos para un entorno distribuido, donde se debe especificar las instancias config servers.



```
esla-dbpedia:~ nosql$ mongos --configdb "172.18.4.172:27019,172.18.4.174:27019,172.18.4.183:27019"
```

4) Iniciar cada instancia Shard a través del siguiente comando:

```
mongod -shardsvr -dbpath directorio anteriormente creado -port puerto
```

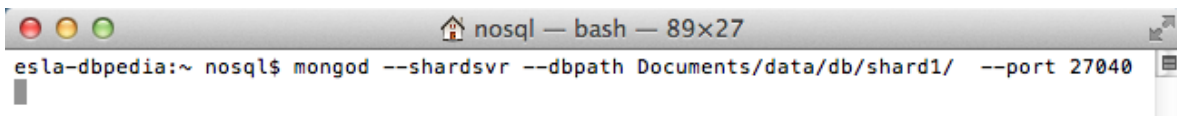
**Donde:**

**shardsvr:** Indica el rol que tiene dicha instancia.

**dbpath:** Directorio donde se almacena los datos.

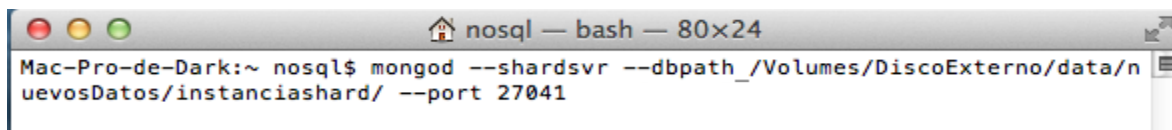
**port:** Puerto donde inicia dicha instancia.

#### Inicio de instancia shard en el Primer Servidor.



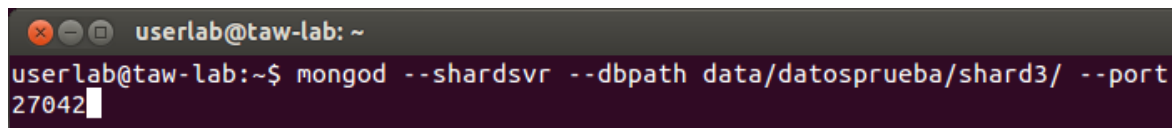
```
esla-dbpedia:~ nosql$ mongod --shardsvr --dbpath Documents/data/db/shard1/ --port 27040
```

#### Inicio de instancia shard en el Segundo Servidor.



```
Mac-Pro-de-Dark:~ nosql$ mongod --shardsvr --dbpath_/Volumes/DiscoExterno/data/nuevosDatos/instanciashard/ --port 27041
```

#### Inicio de instancia shard en el Tercer Servidor.



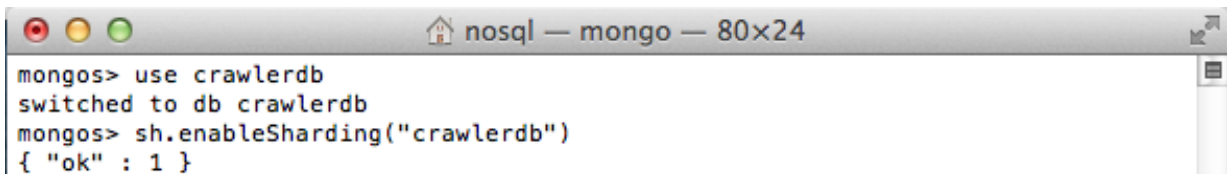
```
userlab@taw-lab:~$ mongod --shardsvr --dbpath data/datosprueba/shard3/ --port 27042
```

5) Conectarse a la instancia donde se realiza la sincronización de los config Servers aplicada en el servidor Principal (**mongos**), a través del siguiente comando: mongo

- 6) Agregar los nodos que conforman el entorno distribuido, solo instancias shards

```
mongos> sh.addShard("172.18.4.172:27040")
{ "shardAdded" : "shard0000", "ok" : 1 }
mongos> sh.addShard("172.18.4.174:27041")
{ "shardAdded" : "shard0001", "ok" : 1 }
mongos> sh.addShard("172.18.4.183:27042")
{ "shardAdded" : "shard0002", "ok" : 1 }
```

- 7) Habilitar la fragmentación en la base de datos a distribuir



```
nosql — mongo — 80x24
mongos> use crawlerdb
switched to db crawlerdb
mongos> sh.enableSharding("crawlerdb")
{ "ok" : 1 }
```

- 8) Habilitar la colección a distribuir, e indicar cuál es el campo clave

En la documentación oficial de MongoDB se proporciona recomendaciones<sup>21</sup> para seleccionar el campo clave (shardkey), tomando en cuenta dichas consideraciones, se determina que el campo “\_id” sea elegido como clave, principalmente por su cardinalidad al tener valores únicos y porque crece constantemente conforme se ingresen nuevos documentos, para garantizar una distribución uniforme el campo clave tiene que ser de tipo “hashed”.

```
mongos> sh.shardCollection("crawlerdb.tweets",{"_id":"hashed"})
{ "collectionsharded" : "crawlerdb.tweets", "ok" : 1 }
```

---

<sup>21</sup> Consideraciones para escoger shardkey: <https://docs.mongodb.org/v3.0/tutorial/choose-a-shard-key/>