UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERIA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERA DE SISTEMAS

TESIS

ANÁLISIS COMPARATIVO DE RENDIMIENTO EN GESTORES DE BASES DE DATOS RELACIONALES Y NO RELACIONALES

PARA OPTAR EL TÍTULO DE INGENIERO DE SISTEMAS

Autor:

Bach. Campoverde Vega Victor Andree

ORCID: https://orcid.org/ 0000-0002-5107-7871

Asesor:

Mg. Ocampo Moreno Rosa Liliana

ORCID: https://orcid.org/ 0000-0001-8465-3751

Línea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú 2022

ANÁLISIS COMPARATIVO DE RENDIMIENTO EN GESTORES DE BASES DE DATOS RELACIONALES Y NO RELACIONALES

PRESENTADO POR:

CAMPOVERDE VEGA VICTOR ANDREE
Autor

Mg.Heber Ivan Mejia Cabrera
Asesor Metodológico

APROBADO POR:

Dr. Vasquez Leiva Oliver
Presidente de Jurado

Mg. Bravo Ruiz Jaime Arturo
Secretario(a) de Jurado

Mg. Bances Saavedra David Enrique
Vocal/Asesor de Jurado

DEDICATORIA

Esta tesis la dedico con especial cariño para mi madre, la persona encargada de que hoy pueda terminar mi etapa académico-profesional. Nada de esto hubiera sido posible sin su incondicional apoyo para ayudarme a cumplir todas las metas propuestas en la vida. A mis hijos, por ser también mi principal motivo para poder culminar con éxito esta tesis y a mi esposa, por su apoyo incondicional y ánimos para no desistir en el camino de la elaboración de la misma. A ustedes que se merecen todo, esto es solo algo significativo.

AGRADECIMIENTO

Agradecer a Dios Todopoderoso por todas las bendiciones puestas en mi vida. Agradecer también a mi asesor por poder encaminarme con éxito a la culminación de esta tesis y por toda la orientación prestada en el desarrollo de esta Tesis.

Agradezco a todos mis docentes por todos sus conocimientos dados en toda mi etapa universitaria los cuales han sido de mucha importancia en el desarrollo de esta tesis.

Un agradecimiento especial para nuestro asesor Metodológico el Mg. Heber Ivan Mejia Cabrera por su paciencia, tiempo y pautas prestadas para el desarrollo de esta tesis.

RESUMEN

El mundo moderno y los constantes avances tecnológicos que van apareciendo a diario han hecho que las bases de datos se conviertan en la parte fundamental de toda tendencia tecnológica, en lo que el almacenamiento de datos se pueda referir. Actualmente la mayoría de aplicaciones tecnológicas necesitan de una base de datos para poder funcionar, tal sea el caso de identidad digital, vehículos inteligentes, asistentes virtuales o dispositivos o que utilicen deep learning. Las bases de datos con el paso de tiempo pueden sufrir ciertos cambios que pueden afectar el rendimiento de las mismas, es aquí donde surge una nueva alternativa que son las bases de datos NoSQL, las cuales se presentan como la solución a problemas muy comunes de bases de datos relacionales. Esta alternativa viene sumando mayor cantidad de desarrolladores en todo el mundo. Esta investigación se basa en poder comparar el rendimiento de los gestores de bases de datos relacionales y NoSQL en equipos con recursos distintos y poder saber en qué caso realizar la mejor elección, para esta investigación se eligieron dos gestores de bases de datos, uno relacional y un NoSQL, la selección se hizo en un ranking elaborado por DB-Engines, donde se tomaron los 10 primeros puestos eligiendo un relacional y un NoSQL; para poder realizar las pruebas de rendimiento se utilizó una base de datos predial de la entidad Cofopri, la cual estaba separada por fechas; los datos se encontraban en formato xlsx, para ejecución de las pruebas fue necesario clasificar y normalizar los datos. Se crearon escenarios de pruebas donde se pudo definir el equipo y la cantidad de registros que iban a ser usados, los registros estaban en un rango de 1000 a 1000000. El consumo de recursos en CPU y memoria RAM fueron recopilados del monitor de recursos de Windows y el tiempo de respuesta fueron tomados de la ventana de salida del software utilizado para las pruebas, todos estos datos fueron documentadas en fichas de registro. Se analizaron los datos y se concluye que la base de datos NoSQL Mongodb puede ofrecer buenos resultados con pocos, medianos y gran cantidad de datos. Mariadb con un servidor con buenos recursos puede ser suficiente para desarrollar cualquier proyecto de base de datos.

Palabras Clave: Bases de datos, NoSQL, deep learning, tendencia, gestores de bases de datos, tuplas, ram, cpu, consumo de recursos, tiempo de respuesta

ABSTRACT

The modern world and the constant technological advances that are appearing daily have made the databases become the fundamental part of every technological trend, in what the data storage can refer. Nowadays, most technological applications need a database to be able to work, such as digital identity, intelligent vehicles, virtual assistants or devices or those that use deep learning. This alternative has been adding more developers around the world. This research is based on being able to compare the performance of the relational database managers and NoSQL in teams with different resources and to know in which case to make the best choice. For this research two database managers were chosen, one relational and one NoSQL, the selection was made in a ranking elaborated by DB-Engines, where the first 10 positions were taken by choosing a relational and a NoSQL; In order to carry out the performance tests, a property database of the Cofopri entity was used, which was separated by dates; the data was in xlsx format, and it was necessary to classify and normalize the data for the execution of the tests. Test scenarios were created where it was possible to define the equipment and the number of records to be used, the records were in a range of 1000 to 1000000. CPU and RAM resource consumption were collected from the Windows resource monitor and the response time was taken from the output window of the software used for testing, all these data were documented in log sheets. The data was analyzed and it is concluded that the Mongodb NoSQL database can provide good results with small, medium and large amounts of data. Mariadb with a server with good resources can be sufficient to develop any database project.

Keywords: Databases, NoSQL, deep learning, trend, database managers, tuples, ram, cpu, resource consumption, response time

ÍNDICE

1. INTRODUCCION 11

1.1. Realidad Problemática: 11

1.2. Antecedentes de Estudio 13

1.3. Teorías Relacionadas al Tema 17

1.4. Formulación del Problema 53

1.5. Justificación e Importancia del Estudio 53

1.6. Hipótesis 54

1.7. Objetivos 54

1.7.1. Objetivo General: 54

1.7.2. Objetivos específicos: 54

2. MATERIAL Y METODO 55

2.1. Tipo y Diseño de Investigación 55

2.2. Población y Muestra 56

2.3. Variables Operacionalización 57

2.4. Técnicas e Instrumentos de recolección de datos, validez y confiabilidad 58

2.5. Procedimientos de Análisis de Datos: 59

2.6. Criterios Éticos 62

2.7. Criterios de Rigor Científico 62

3. RESULTADOS 63

3.1. Resultados en Tablas y Figuras 63

3.2. Discusión de resultados: 70

3.3. Aporte práctico 71

4. CONCLUSIONES Y RECOMENDACIONES 103

REFERENCIAS 107

ANEXOS 110

TABLA DE FIGURAS:

Figura 1. Representación gráfica de entidad 19
Figura 2. Representación gráfica de Relación 20
Figura 3. Relación reflexiva solo relaciona una sola entidad consigo misma 20
Figura 4. Representación de cardinalidad..... 21
Figura 5. Representación gráfica de 1FN A) tabla sin 1FN B) tabla con 1FN 23
Figura 6. Representación gráfica de 2FN A) tabla sin 2FN B y C) tablas creadas por 2FN 23
Figura 7. Representación gráfica de 3FN A) tabla Alumnos sin normalizar B) primera tabla dividida de tabla principal C) segunda tabla dividida de tabla principal 24
Figura 8. Ejemplo de clave Valor..... 26
Figura 9. Ejemplo grafico de estructura tipo documento..... 27
Figura 10. Ejemplo de filas de empleados en una BD NoSQL orientada a columnas..... 27
Figura 11. Imagen de representación de un Grafo 28
Figura 12. Instrucciones SQL por Categorías 37
Figura 13. Proceso de replicación..... 40
Figura 14. Tipos de nodos secundarios y sus relaciones 41
Figura 15. Inicio del proceso de elección de un nuevo nodo primario 42
Figura 16. Distribución de datos en shards..... 43
Figura 17. Elementos de un sharded clúster..... 44
Figura 18. Inserción de Documentos en Mongodb A) Inserción de un documento B) Inserción de varios documentos mediante array 47
Figura 19. Uso del método find() A)método find() B)método find() complementado con pretty ... 49
Figura 20. Uso del método findOne() 49
Figura 21. Uso del método update con parámetros..... 50
Figura 22. Uso de remove y comando findOne para búsqueda de documento eliminado 51
Figura 23.Uso del método drop para eliminar colección libro 51
Figura 24. Tiempo de Respuesta para crear base de datos Mariadb..... 63
Figura 25. Tiempo de respuesta en creación de base de datos Mongodb 64
Figura 26. Consumo de CPU en proceso de creación de base de datos ambos gestores. 65
Figura 27. Uso de memoria RAM en proceso de creación de base de datos 66
Figura 28. Promedio de uso de memoria RAM en Equipo 1..... 67
Figura 29. Promedio de uso de memoria RAM en Equipo 2..... 68
Figura 30. Prueba de uso promedio de CPU en consultas equipo 1 69
Figura 31. Prueba de uso promedio de CPU en consultas equipo 2 69
Figura 32. Resumen de Aporte práctico..... 71
Figura 33. Descarga de datos para pruebas de rendimiento..... 80
Figura 34. Software Pentaho Data Integration comprimido 81
Figura 35. Ventana de bienvenida PDI 82
Figura 36. Creacion de nueva Transformación 83

Figura 37. Proceso de creación de tablas en PDI (A) Selección de Vista Diseño PDI (B) Datos de tabla a migrar (C) Total de tablas terminadas 84
Figura 38. Ventana de bienvenida Mondodb Compass Community 85
Figura 39. Datos para crear base de datos y colección Mongodb 85
Figura 40. Carga de proceso de creación de base de datos Mongodb 86
Figura 41. Creacion de colecciones Mongodb 86
Figura 42. Colecciones creadas para base de datos Mongodb 87
Figura 43.*Proceso de configuración de elemento Mongodb output (A)Elección de opción para relacionar elementos (B)Representación de relación PDI para generar salida (C)Datos de conexión con base de datos relacional y Mongodb* 88
Figura 44. Proceso de vista previa de transformación PDI (A)Selección de opción Mongo documents (B)Preview de campos de tabla relacional transformada a colección (C)Transformación terminada pendiente de ejecución 89
Figura 45. Verificacion de modelo PDI 90
Figura 46. Error de validación PDI 90
Figura 47. Opciones disponibles de Transformación 91
Figura 48. Ventana de verificación de transformación PDI 92
Figura 49. Limpieza de bandeja de salida Fuente: Elaboración propia 93
Figura 50. Ejecucion de script SQL para crear base de datos relacional 93
Figura 51. Tiempo de respuesta ejecución de estructura.sql en equipo 2 93
Figura 52. Monitor de recursos de Windows 95
Figura 53.Uso medio de CPU por segundos inicial Equipo 1 96
Figura 54. Uso medio de CPU por segundos final Equipo 1 96
Figura 55. Error de volcado de sistema operativo Windows 10 equipo 2 97
Figura 56. Monitor de recursos sección memoria 98
Figura 57. Fragmento de script SQL para tiempos de respuesta base de datos Mariadb 99
Figura 58. Fragmento de script SQL para tiempos de respuesta base de datos Mongodb 100
Figura 59. Fragmento de codigo SQL para ejecución de select con inner join 100
Figura 60. Fragmento de código No SQL para ejecución de select con map reduce 101

INDICE DE TABLAS:

Tabla 1 Privilegios globales Mariadb 31
Tabla 2 Privilegios de bases de datos 32
Tabla 3 Privilegios variados de Mariadb 33
Tabla 4 Límite de privilegios para usuarios en Mariadb 33
Tabla 5 Datos soportados en documentos Mongodb 46
Tabla 6 Doce primeros puestos ranking DB-Engines 57
Tabla 7 Muestra de gestores de bases de datos elegidos para la presente investigación 57
Tabla 8 Operacionalización de variables 58
Tabla 9. Discusión de resultados en tiempos recopilados en pruebas de rendimiento en consultas Nosql-Mongodb 70
Tabla 10 Ranking DB-Engines de los 10 primeros Gestores de bases de datos 73
Tabla 11 Evaluación de Características de los 5 primeros gestores de Bases de Datos elegida del ranking DB-Engines 74
Tabla 12 Propiedades de Gestores de Bases de datos elegidos 75
Tabla 13 Métricas según investigaciones 76
Tabla 14 Puntuación de métricas según aparición en investigaciones anteriores 77
Tabla 15. Equipo de trabajo 1 Servidor 78
Tabla 16. Equipo de trabajo 2 laptop convencional 78
Tabla 17 Tablas del modelo relacional después del proceso de normalización 80
Tabla 18. Tiempos de Respuesta en creación de base de datos relacional Mariadb 94
Tabla 19. Tiempos de respuesta en migración de base de datos Mariadb a Mongodb 94
Tabla 20 Tabla de procesos que se cerraron en Escenario 1. 95
T**abla 21** Tabla de procesos que se cerraron en Escenario 2 96
Tabla 22 Tabla de consumo de CPU para creación de base de datos 98
Tabla 23 Uso de memoria RAM para creación de base de datos 99
Tabla 24 Tiempo de rendimiento en segundos de consultas con delimitantes por equipos y gestores de base de datos 100
Tabla 25 Tiempo de respuesta en segundos para consulta con inner join en base de dato relacional y map reduce en Nosql 102
Tabla 26 Promedio de uso de RAM de consultas en bases de datos relacionales y Nosql 102
Tabla 27 Promedio de uso de CPU de consultas en bases de datos relacionales y Nosql 103

1. INTRODUCCION

1.1. Realidad Problemática:

El aumento y crecimiento de datos generados por las diversas aplicaciones informáticas que utilizan un gestor de bases de datos relacional para el almacenamiento de los mismos, son afectados con el crecimiento de los datos. El punto de quiebre de la mayoría de gestores de bases de datos relacionales es el alto consumo de procesador y tiempo de respuesta que pueda tener una consulta que implique una búsqueda extensa en una base de datos relacional. El consumo de uso procesador puede verse muy afectado si varios usuarios hacen consultas similares al mismo tiempo y el tiempo de respuesta pueda aumentar drásticamente, permitiendo así una mala experiencia al usuario final. Existe gran incertidumbre para manejar grandes volúmenes de datos; el poco conocimiento, costumbres y malas prácticas hacen de que la experiencia para el usuario final no sea muy agradable al momento de obtener algún dato y que esta consuma mucho tiempo para mostrar lo requerido. En la mayoría de instituciones gubernamentales los gestores de bases de datos relacionales se han convertido en la primera elección debido al tiempo, uso y simplicidad de los mismos. Se ha buscado desarrollar soluciones informáticas que apoyen la restructuración de las mismas como por ejemplo servicios, apirests o portales de datos libres. Todos estos ofrecen cantidades mínimas de información de grandes bases que funcionan en un gestor relacional. Pero qué pasa cuando la información que se necesita es histórica y es necesario tomar data en grandes volúmenes de datos, la mayoría de estas bases datos no están preparadas para esto y elevan el uso de procesador, tardando mucho tiempo en entregar una respuesta a lo solicitado. En el país todavía no se ha estandarizado algún método que permita la medición de recursos y tiempo de respuestas en bases de datos relacionales cuando estas tienen mucha o poca información, la manera de trabajo que tiene el personal responsable de estos gestores de bases de datos relacionales es que mientras funcionen y entregue datos todo está bien, sin importar los tiempos de respuesta y consumo de procesador que puedan tener.

Las bases de Datos NoSQL se convierten en una alternativa importante para dar la solución a los problemas de rendimientos presentados.Se afirma que una buena alternativa al crecimiento de datos generados por aplicaciones informáticas, es utilizar bases de datos NoSQL; estas bases de datos se caracterizan por ser una buena opción para trabajar con BigData (enormes volúmenes de Datos) (Moreno, Quintero, & Rueda, 2016).

Para aplicaciones que tienen usuarios conectados 24 horas los 7 días de la semana, los 365 días del año es necesario tener una base de datos que pueda soportar la conexión de millones de usuarios (Győrödi, Győrödi, G.Pecherle, & Olah, 2016).

Las bases de datos NoSQL están convirtiéndose en tendencia ya que están teniendo una gran demanda de uso con aplicaciones de redes sociales, fotos y aplicaciones en la Nube. Las bases de datos relacionales están quedando limitados con el constante crecimiento de la información convirtiéndose en un punto débil para los sistemas actuales. Para trabajar con sistemas donde se gestionen contenido, blogs y análisis de datos en tiempo real la mejor opción son las bases de datos NoSQL de tipo Documento. Una de las ventajas resaltantes de las bases de datos NoSQL escalan con facilidad y que para su uso se utilizan equipos de bajos costos.

Las bases de datos NoSQL pueden manejar grandes volúmenes de datos y generarlos más rápidos que las bases de datos relacionales. El método de enfoque que tienen las bases de datos tradicionales orientada a objetos no ofrece una escalabilidad flexible exigida para las nuevas tendencias tecnológicas (Benymol & Sajimon, 2017).

El valor agregado de las bases de datos NoSQL es que el rendimiento es muy óptimo al trabajar con grandes cantidades de datos, de las cuales los datos pueden cambiar continuamente; permitiendo tener una gran variedad de datos (estructurados, no estructurados, semiestructurados). Para cada tipo de datos a trabajar existe una base de datos NoSQL que calza a la medida de los mismos, proporcionando un alto rendimiento, escalabilidad y flexibilidad como características principales de estas (A.Gupta, S.Tyagi, N.Panwar, S.Sachdeva, & Saxena, 2017).

Existe la necesidad con respecto a poder encontrar una solución a los problemas presentados por los sistemas de base de datos relacionales, los sistemas de base de datos NoSQL están teniendo más participación en el mercado, hay muchas empresas en el mundo que están cambiándose del modelo de datos Relacional al NoSQL priorizando un mejor servicio a sus clientes y temas económicos de la empresa.

1.2. Antecedentes de Estudio

Russell (2015), en su investigación, mira como alternativa para solucionar problemas que las diversas bases de datos relacionales no puedan solucionar. Lo primero que él realizo fue evaluar bases de datos NoSQL y bases de datos Relacionales según criterios de popularidad, documentación, tipo de almacenamiento y costo; los manejadores elegidos fueron para NoSQL Cassandra y Mongodb y relacional fue MySQL. Posteriormente se encargó de realizar un análisis dividido en dos partes; parte cualitativa y cuantitativa apoyándose de conceptos obtenidos de diferentes fuentes. Para el análisis cuantitativo utilizó la herramienta YSCB para la cual se realizaron 3 cargas de lectura y escritura al (L:50% y E:50%),(L:95 y E:5%) y (L:100%),obteniendo resultados favorables para

Cassandra seguido por Mongodb a excepción de una sola prueba donde MySQL fue superior. Quispe concluye que las bases de datos NoSQL tienen un buen rendimiento en base de datos NoSQL.

Moreno, Quintero, & Rueda(2016) en su investigación," UNA COMPARACIÓN DE RENDIMIENTO ENTRE ORACLE Y MONGODB" realizaron una comparación de bases de datos Relacional vs NoSQL tomando como caso de estudio a Oracle vs Mongodb, ellos utilizaron procesos de base de datos creación, lectura, inserción, eliminación. Los datos eran recopilados y eran comparados de acuerdo a los tiempos de respuesta de cada proceso. Posteriormente se procedieron a comparar ambos resultados y ofrecer la mejor alternativa en rendimiento, la ganadora en este caso fue Mongodb. Ellos encuentran un error al realizar el segundo estudio donde se trató de ingresar en una base de datos NoSQL una muestra de 10 millones de registros. Ellos indicaron el máximo tamaño que tiene que tener un documento, es de 16 MB. Queda como referencia para probar una inserción de este tamaño para la investigación.

Leung & Zhou(2016) En su investigación, identifican la carga lenta en grandes consumos de datos es por ello que se encargan de evaluar el desempeño de bases de datos NoSQL y SQL, el método de trabajo de ellos fue utilizar un conjunto de datos de Tweets. Luego compararon el rendimiento de las bases de datos más populares Microsoft Sql Server, Postgresql, Mongodb y Redis; esta clasificación fue a priori, es decir, no se aplicó ningún criterio de clasificación. Para esta investigación utilizaron gran cantidad de datos entre 1 millón a 50 millones de Tweets. Ellos recomendaron el uso de Mongodb para trabajo con grandes datos y muchas consultas. Se tomó como referencia para esta investigación la conclusión a la que llegan los autores.

Ribeiro, Henrique, Ribeiro, & Neto (2017) en su investigación, creen que con el avance de las nuevas tendencias tecnológicas las bases de datos relacionales no cumplan con el rendimiento esperado, es por ello que se ve en la necesidad de comparar las bases de datos NoSQL versus las bases de datos relacionales y poder comparar si las bases de datos NoSQL superan en rendimiento a las bases de

datos relacionales. Él para esta investigación utiliza data de una empresa del rubro industrial que se encarga de manejar información de tratamiento de agua, pozos tabulares, depósitos, etc. Lo primero que realiza es selección de los gestores tomando como punto de referencia Rendimiento, Teorema CAP, características ACID, Integración con .NET Tipo de licencia. Él cree que el propósito de toda la investigación se basa en una pregunta muy puntual la cual es ¿Puede una empresa pequeña utilizar bases de datos no relacionales de forma óptima? Finalmente concluye su investigación mostrando resultados favorables NoSQL en este caso Couchdb, pero como punto importante observa la limitación de la misma que al ser una base de comercial esta necesita una licencia de uso. De este autor tomo referencia el criterio de evaluación de las bases de datos y la limitación que encuentra en Couchdb por ser una base de datos Comercial.

Kaur & Singh(2017), encuentran como problema la mala elección y de gestores NoSQL. Para entonces ellos utilizan un método de comparación bases de datos NoSQL basado en características de las mismas: escalabilidad, disponibilidad, rendimiento, consistencia, fiabilidad, flexibilidad, complejidad y mejor uso. Ellos proponen usos específicos para las bases de datos NoSQL como por ejemplo el uso de Mongodb para almacenamiento de información y para operaciones de mayor lectura y carga de datos Hbase.Tomo como referencia para mi investigación la conclusión de los autores al recomendar el uso de Mongodb para operaciones de mayor Lectura.

Benymol & Sajimon(2017) en su investigación encuentran como problema principal el poco conocimiento de las bases de datos NoSQL como una alternativa de las bases de datos relacionales. Exploran los méritos de utilizar bases de datos NoSQL y toman como caso de estudio Mongodb, ellos realizan una revisión a la literatura de conceptos de NoSQL y posteriormente realizan una breve comparación de Mongodb vs Oracle. Ellos concluyen que Mongodb puede ser una de las mejores alternativas para sistemas de gestión de contenidos, Plataformas Virtuales.

Krstića & Krstićb(2018) , en su investigación enfocan como principal problema por qué NoSQL todavía no es visto como una buena competencia ante las relacionales. Ellos buscan elaborar un análisis que permita conocer todas las ventajas que tiene las bases de datos No Relacionales, posterior a ello han seleccionado Mongodb y MySQL para realizar un análisis comparativo donde se modelan los datos en MySQL la base de datos Relacional y luego estos son migrados a la base de datos NoSQL. Para esta investigación se realizaron los siguientes análisis siguiendo el siguiente orden; primero se realizó el análisis de diagnóstico del modelo de datos relacional, luego se hizo el análisis de diagnóstico del modelo de datos no relacional, con los resultados de los dos primeros análisis elaboraron un análisis comparativo entre modelo de datos relacional y no relacional, posteriormente documentaron una propuesta de procedimiento de migración. Se realizó la construcción de un prototipo de herramienta basado en indicadores comparativos de tiempo de respuesta y escalabilidad. Para esta investigación los resultados fueron favorables para Mongodb sobre MySQL, para entonces los investigadores recomiendan el uso de bases de datos NoSQL como alternativa a una base de datos Relacional. Tomo como referencia el método utilizado donde los autores para tener un mejor panorama de comparación primero modelan la base de datos Relacional, posteriormente la alimentan con data y luego esta es migrada a una NoSQL.

JAEJUN & YOO (2018) investiga el principal problema que se genera al tratar de migrar de un modelo de datos relacional a un NoSQL. Ellos elaboran un modelo que se basa de dos metodos principales que son la desagregacion de claves principales y foráneas y la desagregacion a nivel de columna que consite en ir dejando sin efecto a los campos que no se consideran relevantes al momento de crear un modelo de datos no estructurado. Los resultados encontrados son favorables en los aspectos de rendimiento de consultas y tamaño de base de datos creada. En la investigacion se concluye que siguiendo este método la creación de base de datos NoSQL documental reduce en un 2% y es más efectivo que utilizar el método de desnormalizacion general o no estructurado.

1.3. Teorías Relacionadas al Tema

En el siguiente punto se describe teorías relacionados al tema investigado.

1.3.1. Bases de Datos:

Una base de datos es el conjunto de información que es organizada para que posteriormente pueda ser consultada, trabajada y transformada posteriormente. Los datos organizados pueden ser de diferentes tipos (Correa, 2015, p. 9). Las bases de datos tienen que cumplir tres puntos básicos:

1. **No redundancia:** Los datos son almacenados por única vez, aunque su uso se utilice para diferentes fines.

2. **Independencia:** Los datos no dependen de la aplicación que se encargará de tratarlos, estos se organizan independientemente.

3. **Concurrencia:** No importa el número de usuario para hacer uso de los mismos sin presentar problemas.

1.3.2. Sistema Gestor de Base de Datos:

Está compuesto por una serie de recursos los cuales permiten el manejo de diferentes volúmenes de información donde el único objetivo es trabajar como interfaz para que el usuario pueda hacer uso de las aplicaciones del sistema de base de datos. Los sistemas de gestión de bases de datos se organizan en tres niveles:

A) Nivel Interno: El usuario no se interesa por saber cómo y dónde se almacenan los datos, esta función es cumplida por el motor del sistema de gestión.

B) Nivel externo: El usuario maneja toda la información que se almacena en la base de datos enviada por aplicaciones informáticas sin saber la estructura de la base de datos.

C) Nivel intermedio: Ejecución de comandos para poder manipular los datos y realizar tareas de gestión.

Silberschatz, Korth, & Sudarshan (2014, p. 24) En una organización la parte fundamental es la colección de datos ya que en ella guarda información siempre relevante para cualquier organización. El grupo de herramientas informáticas conocidas como software que cumplen la función de permitir al usuario acceder y trabajar los datos es conocido como sistema gestor de base de datos.

1.3.3. Modelo de datos:

Ordoñez, Tapia, & Rivas (2015) describen al modelo de datos como un instrumento que tiene como función principal ayudar con la creación de la base de datos. Este modelo lo conforman normas, conocimientos y símbolos que representan la estructura de la información que será utilizada en la base de datos.

1.3.4. Modelo de Dato Conceptual:

En la actualidad el modelo de datos más utilizado es el modelo entidad relación. La característica principal de estos modelos son que pueden ser autónomos o semánticos de acuerdo al tipo de base de datos a crear (Ordoñez, Tapia, & Rivas, 2015).

1.3.5. Modelo de Datos Relacional:

Este modelo es utilizado para mostrar datos que están interrelacionados. Se denomina relacional ya que la función que realiza este modelo es de clasificar la información y crear relaciones entre las entidades, en los últimos años ha crecido la popularidad de este modelo (Arias, 2016). Para Peter Chen, creador de este modelo en el año 1976, los principales elementos de estos modelos son atributos, entidades y relaciones.

A) Entidad:

Una entidad es el objeto en donde se guardará información ya clasificada de la base de datos. Son todos los elementos principales que se ordenan y clasifican de acuerdo a un proceso que cumple un determinado fin y función. Si tomamos como ejemplo una escuela, las entidades principales serían aula, docente, profesor, alumno, apoderados. Para representar gráficamente una entidad tendríamos que dibujar un rectángulo y colocar en su interior el nombre de la entidad (Kroenke, 2003).

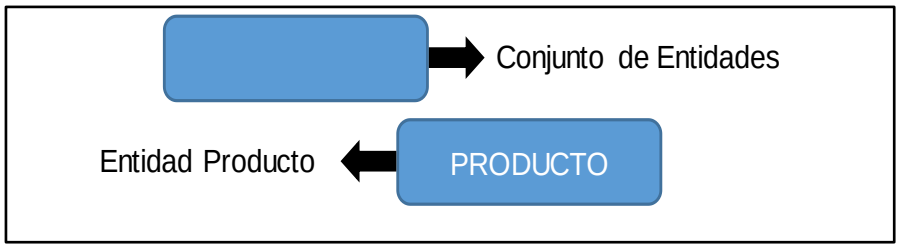


Figura 1. Representación gráfica de entidad **Fuente:** (Bender,Deco, Sanabria, Hallo, & Gallegos, 2014)

B) Relación:

La relación es el vínculo que guardan dos a más entidades, permitiéndose cualquier tipo de interacción entre las mismas. Un ejemplo claro de una relación se podría dar entre una Persona y sus familiares cercanos la relación que guardarían sería de parentesco. Esta puede ser representada mediante un rombo, en donde iría el nombre de la relación y las asociaciones entre entidades elegidas.

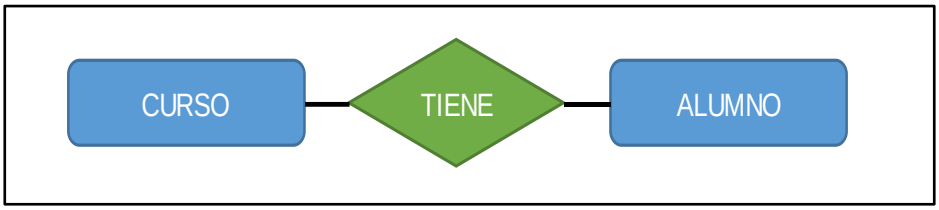


Figura 2.Representación gráfica de Relación **Fuente:** (C.Bender, C. Deco, Sanabria, Hallo, & Gallegos, 2014)

Las relaciones poseen las siguientes características:

NOMBRE: El nombre de la relación será único para poder distinguirse de las demás.

GRADO: Este es obtenido según el número de entidades que estén incluidas en la relación. Estas se pueden clasificar en relaciones binarias, reflexivas y de grado 3 o 4. A continuación, se explica la clasificación de manera gráfica:

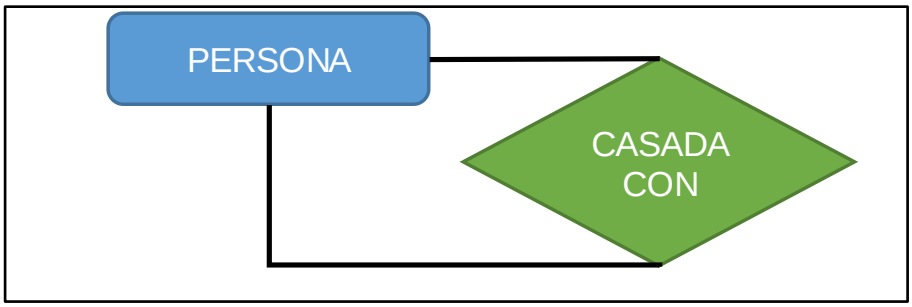


Figura 3.Relación reflexiva solo relaciona una sola entidad consigo misma **Fuente:(**C.Bender, C. Deco, Sanabria, Hallo, & Gallegos, 2014).

Es importante saber el número de asociaciones que puede tener una entidad con respecto al conjunto de relaciones del modelo, a este concepto se le conoce como cardinalidad (C.Bender, C. Deco, Sanabria, Hallo, & Gallegos, 2014).

Según Sánchez (2009) la cardinalidad puede conocerse de dos formas según las veces que una entidad puede aparecer en alguna relación.

Cardinalidad mínima: El valor puede tener un único valor, asumiendo que puede ser 0 o tal vez 1, si la cardinalidad que presenta es de cero o más de cero, el valor que tomaría será de uno.

Cardinalidad máxima: Concretamente el valor tiene que ser mayor a uno, dos o tres o cuando no conocemos el numero simplemente puede utilizarse la letra n minúscula.

Para Sánchez (2009) la forma más completa de expresar la cardinalidad en los modelos de datos es anotando el tipo de cardinalidad sea mixta o mínima de cada entidad en la relación.

Figura 4.Representación de cardinalidad **Fuente:** (Sánchez, 2009)

En la figura 4, la entidad jugador es de cardinalidad mínima cero, donde podría incluirse en un equipo y la máxima de uno porque no podría estar en dos equipos al mismo tiempo. La cardinalidad mínima de cada equipo es de uno como se explicó líneas más arriba, la cardinalidad puede ser mayor a uno, pero se toma uno como referencia.

C) Atributos:

Son las características o propiedades que conforman una relación o entidad. Por ejemplo, tomando como ejemplo la entidad cliente los atributos son id, descripción, dni y todos los atributos que contemplen información del cliente. Existen varios tipos de atributos, (C.Bender, C. Deco, Sanabria, Hallo, & Gallegos, 2014) los clasifican de la siguiente forma.

A) Atributo identificador principal AIP: Con este atributo se identifica cada registro de la entidad, necesariamente tiene que ser único ya que no podría existir dos registros con código similar.

B) Atributo identificador candidato AIC: Solo si la entidad tiene algún atributo que pueda tener la condición de ser un atributo identificador principal, a los demás atributos que no cumplan con esta condición se les clasificara como AIC.

C) Atributo identificador alternativo (AIA): Son todos los demás atributos que no cumplan con la condición de ser identificador candidato ni principal.

1.3.6. Formas Normales

En un modelo de base de datos relacional es suficiente trabajar con las formas normales definidas por Codd y continuadas por más autores con el paso del tiempo, por consiguiente (Ballester, y otros, 2007) afirman que es suficiente normalizar las tablas hasta la tercera forma normal (3FN) para que el diagrama presente una buena estructura.

A) Primera Forma Normal (1FN):

La mayoría de tablas relacionales cumplen esta primera forma normal, por lo tanto, se dice que una tabla se encuentra en 1FN cuando no deja que un atributo de cualquier fila que lo conforma pueda tomar más de un valor (Sanchez, 2009).

TRABAJADOR		
DNI	**NOMBRE**	**DEPARTAMENTO**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección Gestión

(A)

1FN

TRABAJADOR		
DNI	**NOMBRE**	**DEPARTAMENTO**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección
12345345G	Andrea	Gestión

(B)

Figura 5.Representación gráfica de 1FN **A)** tabla sin 1FN **B)** tabla con 1FN **Fuente:** (Sanchez, 2009)

B) Segunda Forma Normal(2FN):

Necesariamente la tabla tiene que encontrarse en primera forma normal y tiene que cumplirse que todos los atributos que conforman la tabla, que no son claves principales, no tendrían alguna dependencia con esta. Si existiera más de un atributo que dependa de la clave principal en la tabla, estos tendrían que formar una nueva tabla.

(A)

ALUMNOS				
DNI	**COD CURSO**	**NOMBRE**	**APELLIDO**	**NOTA**
12121219A	34	Pedro	Valiente	9
12121219A	25	Pedro	Valiente	8
3457775G	34	Ana	Fernandéz	6
5674378J	25	Sara	Crespo	7
5674378J	34	Sara	Crespo	6

2FN

ALUMNOS		
DNI	**NOMBRE**	**APELLIDO**
12121219A	Pedro	Valiente
3457775G	Ana	Fernandéz
5674378J	Sara	Crespo

(B)

DNI	COD CURSO	NOTA
12121219A	34	9
12121219A	25	8
3457775G	34	6
5674378J	25	7
5674378J	34	6

(C)

Figura 6.Representación gráfica de 2FN **A)** tabla sin 2FN **B y C)** tablas creadas por 2FN Fuente: (Sanchez, 2009)

C) Tercera Forma Normal(3FN):

Para que una tabla se encuentre en tercera forma normal es necesario que esta ya se encuentre en primera y segunda forma y que si en la tabla existen atributos que no guarden relación con la clave primaria, es necesario crear una nueva tabla con estos atributos (Sanchez, 2009).

(A)

ALUMNOS				
DNI	NOMBRE	APELLIDO	COD PROVINCIA	PROVINCIA
12121349A	Salvador	Velasco	34	Palencia
12121219A	Pedro	Valiente	34	Palencia
3457775G	Ana	Fernandéz	47	Valladolid
5674378J	Sara	Crespo	47	Valladolid
3456858S	Marina	Serrat	8	Barcelona

3FN

DNI	NOMBRE	APELLIDO	COD PROVINCIA
12121349A	Salvador	Velasco	34
12121219A	Pedro	Valiente	34
3457775G	Ana	Fernandéz	47
5674378J	Sara	Crespo	47
3456858S	Marina	Serrat	8

(B)

COD PROVINCIA	PROVINCIA
34	Palencia
34	Palencia
47	Valladolid
47	Valladolid
8	Barcelona

(C)

Figura 7.Representación gráfica de 3FN A) tabla Alumnos sin normalizar B) primera tabla dividida de tabla principal C) segunda tabla dividida de tabla principal **Fuente:** (Sanchez, 2009)

1.3.7. CPU:

Es la parte más importante de la computadora, también conocido como procesador. Aquí es donde se trabaja con los datos, instrucciones de programas y mayoría de control de los elementos o componentes. Se encarga de coordinar, priorizar y controlar todas las operaciones del sistema u ordenador.

Así como evolucionaron los computadores también evolucionó la estructura de la CPU, incrementando sus capacidades para que pueda adecuarse a las necesidades de la informática que aparecerían con el avance del tiempo, básicamente los componentes básicos de la CPU estarían compuestos por la unidad aritmética -lógica(AUL) la cual es la encargada de realizar todas las operaciones matemáticas y la unidad de control (CU) conocido como el acompañante el cual tiene como única función distribuir y coordinar información con los componentes del computador (Rubio, Gomez, Letón, Rodrigo, & Chaos, 2017).

El cálculo de potencia en velocidad de un microprocesador, depende del tiempo en el que tiene para poder realizar las diversas operaciones básicas y es marcada por el reloj interno que posee. (Ordoñez, Tapia, & Rivas, 2015).

1.3.8. Memoria RAM:

El dispositivo encargado de almacenar información no permanente o permanente en un computador es la memoria. La capacidad de almacenamiento, la velocidad de escritura, el tiempo de acceso y la velocidad de lectura, será variable en función a las necesidades que se tenga.

Este dispositivo puede pasar por el proceso de escritura y lectura las veces que sea necesario y su mejor característica es que esta es volátil, es decir, pierde la información almacenada siempre y cuando no esté alimentada a la corriente eléctrica. (Rubio, Gomez, Letón, Rodrigo, & Chaos, 2017).

1.3.9. Clasificación de Sistemas de Bases de Datos NoSQL:

Benymol & Sajimon(2017) afirman que los sistemas gestores de bases de datos NoSQL pueden ser clasificados por la forma que tienen para poder organizar y almacenar los datos.

A) Bases de Clave-valor:

Son las más sencillas y populares debido a la estructura simple que utilizan. En estas bases para cada valor le corresponde una llave única.

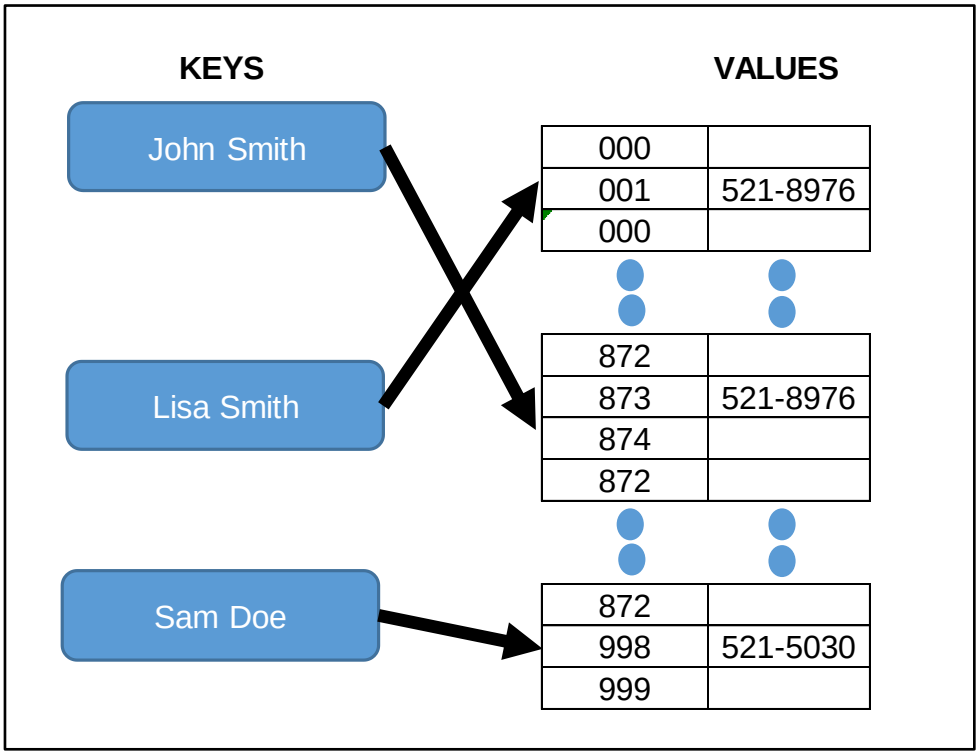


Figura 8.Ejemplo de clave Valor **Fuente:(**Sánchez, 2009)

B) Bases Documentales:

Para estas bases de datos el tipo de almacenamiento que realiza es como lo indica su nombre, del tipo documento. Son las más versátiles según la clasificación elaborada.

Chávez (2014) afirman que “Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales”.

Figura 9.Ejemplo grafico de estructura tipo documento **Fuente:** (Sánchez, 2009)

C) Base de datos tipo Columna:

En estas bases de datos el almacenamiento se realiza por columnas. Para cada grupo de valores representan un mismo atributo.

1	2	3	Una fila de **ids.**
Juan	Pablo	Ana	Una fila de **Nombre**
23	25	21	Una fila de **Edades.**
8	10	6	Una fila de **Grados.**
10	10	20	Una fila de **Deps.**

Figura 10.Ejemplo de filas de empleados en una BD NoSQL orientada a columnas **Fuente:** (Moreno, Quintero, & Rueda, 2016).

D) Base de datos tipo Grafo:

Para estas bases de datos se utilizan grafos. Para que estás bases de datos puedan tener un buen rendimiento estas tienen que estar normalizadas.

R.Rai & Chettri(2017, p. 197-198) resumen que una base de datos de gráficos es básicamente una variedad de nodos y bordes en los que cada nodo significa una entidad (como una persona o empresa) y cada borde representa una conexión o relación entre dos nodos.

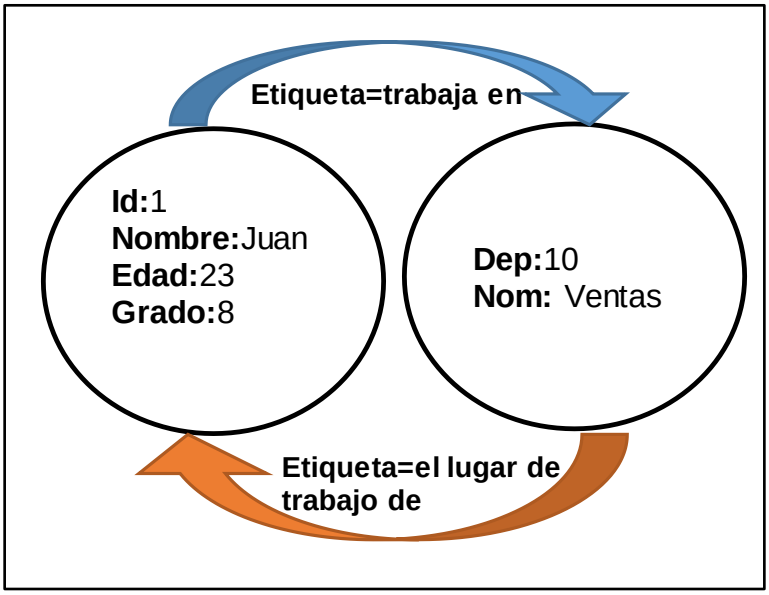


Figura 11.Imagen de representación de un Grafo **Fuente:** (Moreno, Quintero, & Rueda, 2016)

1.3.10. Diferencia de NoSQL vs Sql

Moreno, Quintero, & Rueda (2016) definen a las bases de datos NoSQL como listas de datos almacenados en una sola tabla sin definir relaciones entre los registros. En el caso de las relacionales reparten los datos en varias tablas más pequeñas eliminando datos duplicados y asegurando consistencia y estableciendo restricciones y relaciones con otras tablas por medio de claves primarias y foráneas.

Brito Zhunio(2011, p. 18) describe que “La unidad básica de almacenamiento se define como un "documento" que sería el equivalente a un registro en una base de datos relacional. Los "documentos" con contenidos en una estructura de mayor jerarquía denominada "colección" que almacena los datos de similar tipo; las colecciones serían el equivalente las tablas en un sistema relacional. Finalmente, las colecciones se agrupan en una base de datos que las contendrá”.

1.3.11. Mariadb

Después de haber comenzado como un fork de Mysql que tenía como principal fundador del proyecto a Michael “Monty” Widenius el cual teniendo claro las intenciones de Oracle actual dueño de Mysql que la única intención que tenía era eliminar la única competencia en bases de datos de código libre la compra de Mysql lideraba todos los rankings por muy encima de Oracle. El equipo encargado de realizar el desarrollo de Mariadb basó su producto en mejoras al código fuente de Mysql, a estas mejoras las llamó series y se basaron en las buenas prácticas de Debían, lo cual consiste en tener una versión estable y otra de desarrollo constante (Bartholomew, 2013).

A) Arquitectura del Sistema:

La arquitectura de Mariadb es muy diferente a la de Mysql, se encarga de separar el procesamiento de datos y las tareas de mantenimiento del servidor (Bartholomew, 2013). A continuación se detalla de manera gráfica la arquitectura de este gestor y posteriormente la explicación de la misma.

El grupo de conexiones está conformado por todos los equipos que tengan conexión directa al gestor de base de datos y la forma de conexión por la cual se realice, ya sea utilizando conectores para aplicaciones desarrolladas, api rest o distintas formas de conexión.

El grupo de conexiones es el encargado de ordenar a los clientes para el paso al siguiente nivel. En el caché de consultas se busca que los comandos que pretendan ser ejecutados por el cliente, se encuentren en la caché del servidor, caso contrario serán almacenados para posibles nuevas ejecuciones. El analizador sql es el encargado de asignarle un identificador interno a la consulta y tomar toda la información del motor de almacenamiento.

Mariadb en las últimas series se ha encargado de hacerle mejoras a los motores para poder optimizar tiempos de ejecución, consumo de recursos, procesamiento de datos y replicación.

B) Seguridad:

Es recomendable utilizar buenas prácticas para mejorar la seguridad de Mariadb, una buena contraseña alfanumérica mantendrá el gestor de base de datos seguro. Es importante crear roles y asignar permisos a los usuarios para que el entorno de trabajo se encuentre seguro y evitar cambios perjudiciales (Bartholomew, 2013). Los privilegios en Mariadb se clasifican en tres categorías:

1. Administrativos globales:

Son los privilegios que se aplican a todas las bases de datos en conjunto. Se detallan a continuación.

Tabla 1

Privilegios globales Mariadb **Fuente:** (Bartholomew, 2013)

PRIVILEGIO	DESCRIPCION
CREATE USER	La capacidad de crear un usuario
FILE	La capacidad de usar la declaración LOAD DATA INFILE y la función LOAD_FILE().
PROCESS	La capacidad de utilizar la lista de procedimientos
RELOAD	La capacidad de usar la declaración de FLUSH.
REPLICATION CLIENT	La capacidad de usar el SHOW MASTER STATUS
REPLICATION SLAVE	La capacidad de obtener actualizaciones de la
SHOW DATABASES	La capacidad de listar todas las bases de datos del
SHUTDOWN	La capacidad de apagar el servidor usando el
SUPER	La capacidad de utilizar comandos de superusuario que permiten realizar cambios en tablas maestras, purgar registros, asignar variables globales y terminar hilos secundarios

2. De bases de datos:

Estos privilegios pueden ser aplicados a una base de datos o alguna tabla que la conforme.

Tabla 2

Privilegios de bases de datos **Fuente:** (Bartholomew, 2013)

PRIVILEGIO	DESCRIPCION
ALTER	La capacidad de cambiar los índices y las tablas.
ALTER ROUTINE	La capacidad de cambiar o borrar procedimientos y funciones almacenadas.
CREATE	La capacidad de crear bases de datos y tablas.
CREATE ROUTINE	La capacidad de crear procedimientos y funciones almacenadas.
CREATE TEMPORARY TABLES	La capacidad de crear tablas temporales.
CREATE VIEW	La capacidad de crear vistas.
DELETE	La capacidad de eliminar filas de las tablas.
DROP	La capacidad de eliminar bases de datos y tablas enteras.
EVENT	La capacidad de alterar, crear y eliminar eventos del planificador de eventos.
EXECUTE	La capacidad de ejecutar funciones y procedimientos almacenados.
INDEX	La capacidad de crear o borrar índices.
INSERT	La capacidad de insertar nuevas filas de datos en una tabla.
LOCK TABLES	La capacidad de bloquear y desbloquear tablas.
SELECT	La capacidad de leer datos de una tabla.
SHOW VIEW	La capacidad de usar la declaración SHOW CREATE VIEW.
TRIGGER	La capacidad de usar las declaraciones CREATE TRIGGER y DROP TRIGGER.
UPDATE	La capacidad de modificar las filas de una tabla.

Se tienen los privilegios de columnas individuales y solo existen 3 que son INSERT, UPDATE, y SELECT.

3. Variados:

Son considerados todos los privilegios que no encajan en las anteriores categorías.

Tabla 3

Privilegios variados de Mariadb **Fuente:** (Bartholomew, 2013)

PRIVILEGIO	DESCRIPCION
USAGE	No concede nada real, pero puede ser usado para cambiar las opciones globales de un usuario.
ALL PRIVILEGES	Puede utilizarse para conceder todos los privilegios disponibles a un usuario. No otorga el privilegio de GRAN USUARIO. Puede ser acortado a TODOS.
GRANT OPTION	Le da a un usuario la capacidad de dar a otros usuarios los privilegios que tienen. Esto se da al final de la declaración de CONCESIÓN. Consulte la sección Concesión de permisos de este capítulo para ver algunos ejemplos.

También hay límites que se le pueden asignar a las cuentas de usuario, los más usados son detallados a continuación.

Tabla 4

Límite de privilegios para usuarios en Mariadb **Fuente:** (Bartholomew, 2013)

LIMITE	DESCRIPCION
MAX_QUERIES_PER_HOUR	El número de sentencias o consultas SQL que la cuenta de usuario puede emitir por hora, incluyendo actualizaciones.
MAX_UPDATES_PER_HOUR	El número de sentencias de actualización de SQL (no consultas) que la cuenta del usuario puede emitir por hora.
MAX_CONNECTIONS_PER_HOUR	El número de conexiones que el usuario puede utilizar por hora.
MAX_USER_CONNECTIONS	El número de conexiones simultáneas que realiza el usuario al servidor de base de datos. Si se pone a cero, el número será igual al ajuste configurado en el campo max_connections. Si el ajuste de max_connections es también cero, entonces no hay límite para el número de conexiones simultáneas.

C) Motores de Almacenamiento

Es el encargado de poder trabajar el almacenamiento, manejo y recuperación de toda la información en una tabla. En el caso de Mariadb la elección que tengamos que realizar para utilizarlos en cualquier de los casos dependerá al 100 por ciento del ambiente de uso que se les dé (Fanta, 2019).

1. **Aria:** A partir de la versión 10.5 en adelante, está siendo utilizado de forma predeterminada al momento de crear base de datos. Inicialmente fue creado con el nombre Maria, su creación fue basada en las características del motor de Mysql MyISAM, mejorando el almacenamiento en cache. Es mayormente utilizada para aplicaciones que no tienen una estructura definida (Fanta, 2019).

2. **InnoDB / XtraDB:** A partir de la versión 10.3.7 de Mariadb, Innodb no es incluido como motor de almacenamiento, todas las versiones posteriores a 10.3.7 utilizan XtraDB conservando por razones de compatibilidad las variables de sistema. Mariadb se basó en la forma de trabajo XtraDB de Percona Server, es utilizado para poder trabajar con modelos transaccionales, soporta las características ACID y la arquitectura MVC, quiere decir que está diseñado para escalar mejor en muchos núcleos, para usar la memoria de manera más eficiente y para ser más conveniente y útil (Mariadb, 2020).

3. **MariaDB ColumnStore:** Es el motor que utiliza el almacenamiento en columnas o también conocido como almacenamiento horizontal, está diseñado para trabajar con grandes cantidades de datos y ofrecer buenos tiempos de respuesta (Mariadb, 2020).

4. **Archive:** Este motor tiene como forma de trabajo comprimir las filas utilizando gzip, es utilizado para poder almacenar grandes cantidades de datos sin índices. Admite comandos INSERT, SELECT pero no UPDATE ni DROP.La selección de datos en este motor es extremadamente lenta, ya que lo primero que tiene que hacer es descomprimir la fila y luego presentar los datos, este motor no cuenta con caché (Mariadb, 2020).

5. **Blackhole:** La forma de trabajo que utiliza este motor no almacena ningún registro y el resultado que mostrará siempre será vacío. El uso que se le da a este motor es el campo de la replicación de tablas esclavo-maestro (Mariadb, 2020).

6. **Connect:** Este motor fue creado para poder transformar información, poder agrupar la mayor cantidad de información clasificada o no clasificada y convertirla en información valiosa y útil. Connect es una mezcla de trabajar con datos de aplicaciones relacionales y también de otros tipos, entre los más conocidos JSON, XML (Mariadb, 2020).

7. **CSV:** Este motor dejó de ser mejorado por Mariadb ya que Connect surgió como reemplazo. Este motor tiene como limitaciones principales que las tablas que lo utilizan no pueden ser indexadas ni particionadas (Mariadb, 2020).

Estos han sido los motores de almacenamiento que siguen estables en la serie de desarrollo de Mariadb, también existen otros motores que se han estancado debido a que ya su uso es muy antiguo o menos requerido por los usuarios (Correa, 2015).

D) Control de Concurrencia:

Mariadb utiliza el control de concurrencia para que cundo se ejecute una instrucción SELECT de alguna tabla, la instrucción se encargará de bloquear en las demás sesiones de usuario que las tablas en consulta no puedan ser escritas, pero si puedan ser leídas. La concurrencia en un sistema gestor de base de datos funcionará como un semáforo sincronizado el cual se encargará de ordenar las sentencias a ejecutar, permitiendo la ejecución de múltiples sentencias en un mismo gestor de base de datos sin que la información se vea afectada (Kroenke, 2003).

E) Instrucciones SQL:

Fanta (2019) Clasifica las instrucciones SQL en función a las tareas que se realizan y en el momento en el cual son ejecutadas.

Instrucciones DDL:

Lenguaje de definición de datos, son las instrucciones encargadas de realizar todas las gestiones de forma y organización de los objetos de bases de datos, entre las principales crear, eliminar, consultar, asignar permisos, renombrar.

1. Instrucciones DML:

Lenguaje de manipulación de datos, son las instrucciones encargadas de gestionar los datos almacenados en las entidades existentes en la base de datos, entre las principales instrucciones encontramos las de inserción, actualización, eliminación y bloqueo.

2. Instrucciones TCL:

Lenguaje de Control de Transacción, son las instrucciones encargadas de velar por todos los procesos realizados por las instrucciones DML.

3. Instrucciones SCL:

Lenguaje de control de sesión, son todas las instrucciones encargadas de realizar toda la administración de sesiones y usuarios de la base de datos. Las instrucciones más importantes de esta clasificación son las encargadas de alterar las sesiones y asignar permisos.

4. Instrucciones Embedded SQL 2.0

Fanta (2015) afirman que son las instrucciones que permiten agrupar todas las instrucciones DDL,DML y TCL.

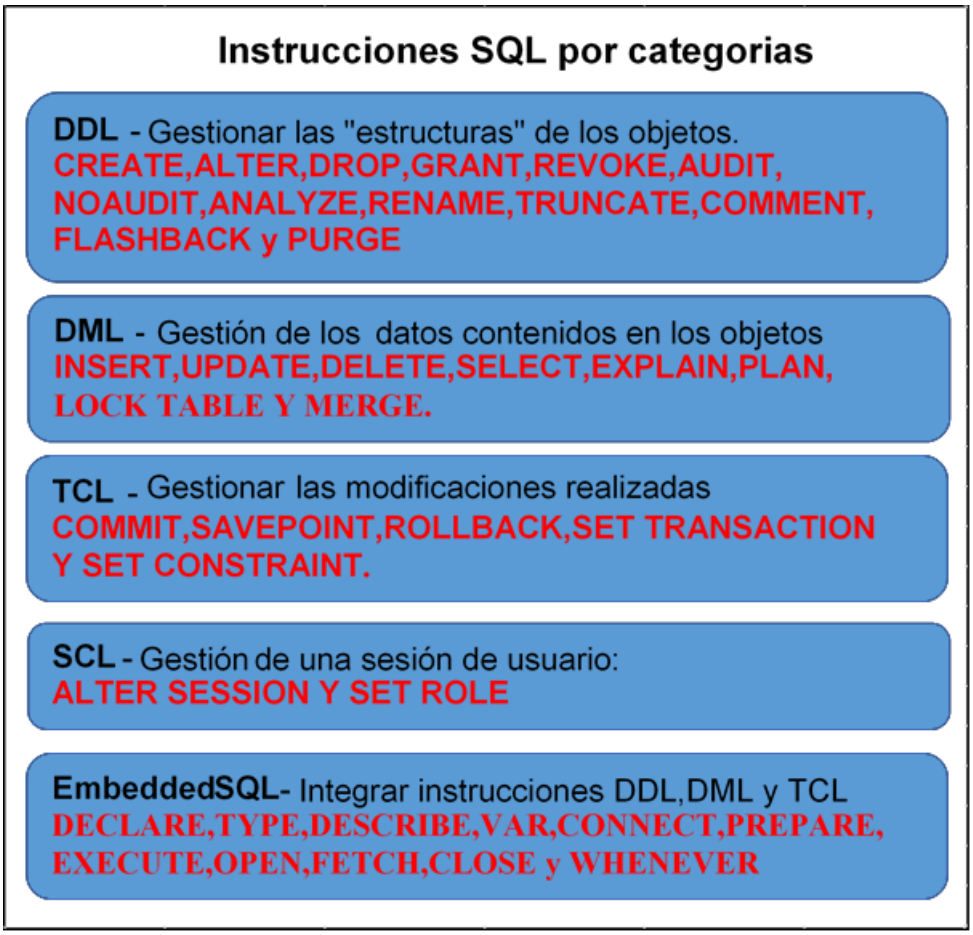


Figura 12.Instrucciones SQL por Categorías **Fuente** (Fanta, 2019)

1.3.12. Mongodb

Este gestor de base de datos fue desarrollado por la compañía 10gen3 a mediados del 2007 y publicado en el inicio del 2009, su nombre en inglés hace referencia a la palabra homongus que en español tendría un significado de grande. Utiliza una licencia open source en el estándar GNU AGPL 3.0. Es un gestor de base de datos de tipo documental y utiliza el formato de datos BSON para el tratamiento de datos. La popularidad alcanzada por este gestor puede ser comparada con MySQL a la fecha (Graterol, 2014).

A) Arquitectura:

Este gestor de base de datos está desarrollado para soportar documentos tipo JSON, estos documentos son tanto intuitivos como flexibles para ofrecer una mayor productividad y velocidad de los datos. El modelo de datos de documental se ha convertido en la alternativa más popular al modelo tabular utilizado por las bases de datos relacionales tradicionales. La arquitectura de este gestor se basa en la replicación y el proceso sharding (Sarasa, 2016).

1. Replicación:

Es el proceso por el cual la información que dispone una base de datos, son sincronizados a través de servidores distintos al servidor que hospeda la base de datos. Entre las ventajas más relevantes de la replicación sobresale el poder incrementar el rendimiento de todos los procesos de selección y lectura que ejecuta el cliente, reduciendo considerablemente tiempos de latencia (Sarasa, 2016).

A. Replica sets: Es un conjunto de instancias de Mongodb que guardan los mismos datos. Una de las instancias llamada primaria, es la que se encarga de recibir y gestionar los procesos de escritura. Las demás instancias, llamadas secundarias, realizan los procesos que la instancia primaria les ordene, de modo que ambas instancias dispongan del mismo conjunto de datos.

La réplica set solo tiene un nodo primario, y puede tener entre uno o más nodos secundarios. Para cada réplica es recomendado contar con 3 nodos y un principal y 2 secundarios. El proceso de lectura es realizado en el nodo primario, en caso este no tenga un estado disponible para realizar el proceso uno de los secundarios asumirá la función de primario.

B. Proceso de Replicación:

El nodo primario tiene el conjunto de logs denominados oplogs, donde almacena todos los cambios que se hicieron sobre el conjunto de datos de la replicación. Los nodos secundarios se encargan de replicar cada oplog del nodo primario, también verifican que, si existe alguna operación pendiente que este contenida con los mimos datos, de modo que el nodo primario sea replicado.

El trabajo de operación del nodo primario sobre los nodos secundarios es de manera asíncrona, esto no afecta en nada el conjunto de replicación. El único problema que podría presentarse si sucediera algún error, sería que alguno de los nodos secundarios no devuelva la información actualizada (Sarasa, 2016).

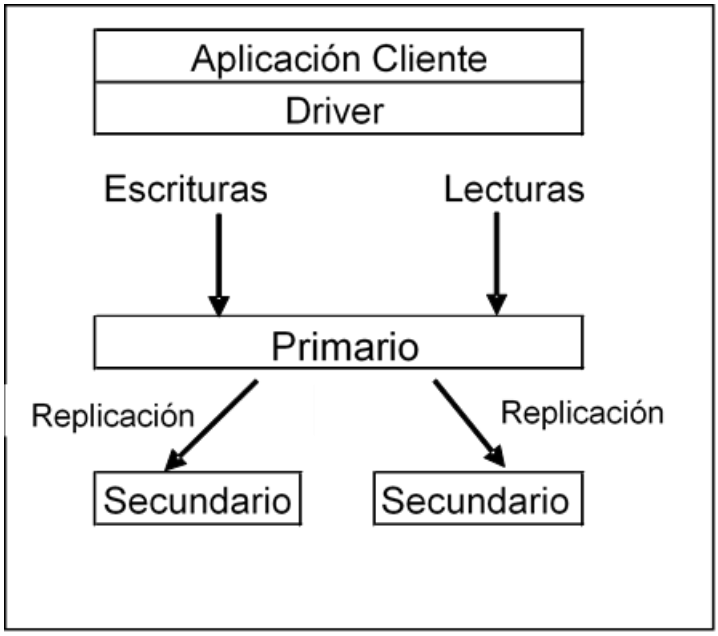


Figura 13.Proceso de replicación **Fuente:** (Sarasa, 2016)

C. Configuración de nodos secundarios:

Los parámetros que necesitan ser configurados en los nodos secundarios son los siguientes.

1.- Prioridad: Con este parámetro se establece la preferencia del nodo para poder elegirse como primario en el proceso de elección. Si se le asigna la prioridad de 0 a un nodo, este no podrá ser elegido como primario.

2.- Visibilidad: Si un nodo tiene la propiedad hidden activa, significa que se encargará de mantener la información que es replicada por los demás nodos, también no podría entrar a ninguna de las elecciones para elegir los nodos primarios.

Existen varios tipos de nodos secundarios.

I) Nodos retardados:

Es el nodo secundario que guarda estados anteriores de la base de datos, con esto si existe una operación de recuperación por fallos, este nodo será encargado de otorgar esta información. Se le puede configurar el tiempo de retardo que tendrá necesita que tenga la propiedad hidden activa y tener como prioridad 0, si puede participar en la elección de nodo primario.

II) Nodos árbitros:

Este nodo no realiza ninguna función de replicación, no almacena ni procesa información. Únicamente se crea cuando en la elección se presenta un empate entre nodos principales; con esto se le evita la sobrecarga de algún otro nodo que necesite replicar datos del nodo principal. La elección siempre tiene que realizarse de forma impar, en caso la elección se encuentre en forma par, automáticamente se creara el nodo árbitro para evitar el empate.

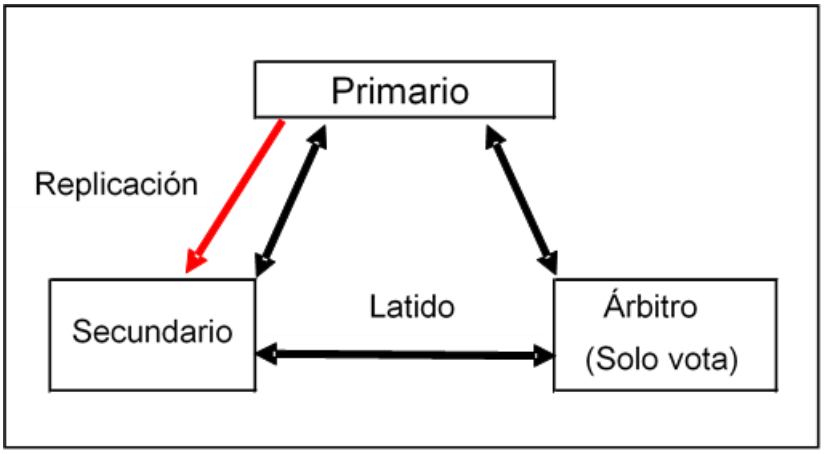


Figura 14.Tipos de nodos secundarios y sus relaciones **Fuente:** (Sarasa, 2016)

D. Recuperación ante fallo del nodo primario:

Cuando en la replicación ya transcurrió un lapso de tiempo mayor a 10 segundos sin poder comunicarse entre nodos, se inicia el proceso de elección para seleccionar un nuevo nodo primario. Mientras no se defina el nodo primario no se permitirá que se ejecuten procesos de escritura sobre los nodos.

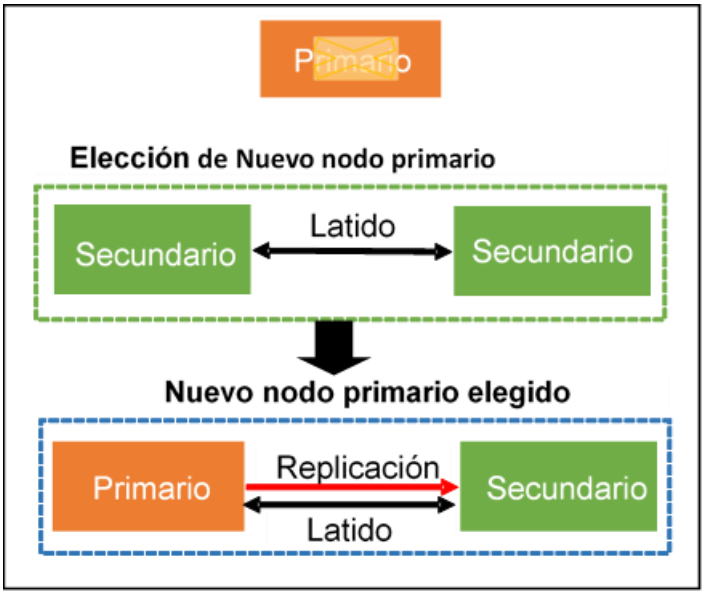


Figura 15.*Inicio del proceso de elección de un nuevo nodo primario* ***Fuente:*** *(Sarasa, 2016)*

Un replica set el máximo de miembros que puede tener es de 50, de los cuales solo pueden votar de 1 a 7 miembro en las elecciones. Si hay una caída en el servidor primario, los cambios se pueden perder si ninguno de estos se transfirió a cualquier otro miembro del replica set.

Si cualquier otro miembro asume la función de primario, este no conocerá todos los cambios que se hayan hecho.

Una vez restablecida la conexión, se deben descartar todos los cambios que no fueron afectados y sincronizar con el primario. Si se desea restablecer algún cambio no sincronizado tendrá que hacerse de forma manual.

2. Sharding:

El mecanismo implementado que permite procesar los datos distribuidos se le conoce como sharding. Este mecanismo surge a partir de la necesidad de trabajar con datos extremadamente grandes donde los problemas identificados se encuentran en el uso de recursos en base a la información que se ejecuta en el momento (Sarasa, 2016).

Para poder solucionar el problema de recursos existen 2 formas de escalado; el escalado horizontal que es el más costoso y más rápido, ya que solo es necesario mejorar los recursos del servidor de base de datos, puede ser en memoria, procesador, disco duro, placa madre. La otra forma es el horizontal, es más barato por que se utilizarían más equipos de recursos aceptables.

El proceso de sharding consiste en que la información sea partida y colocada en diferentes equipos, para que este proceso tenga éxito es necesario que la infraestructura de trabajo sea de servidores y que la configuración de distribución tenga los parámetros correctos.

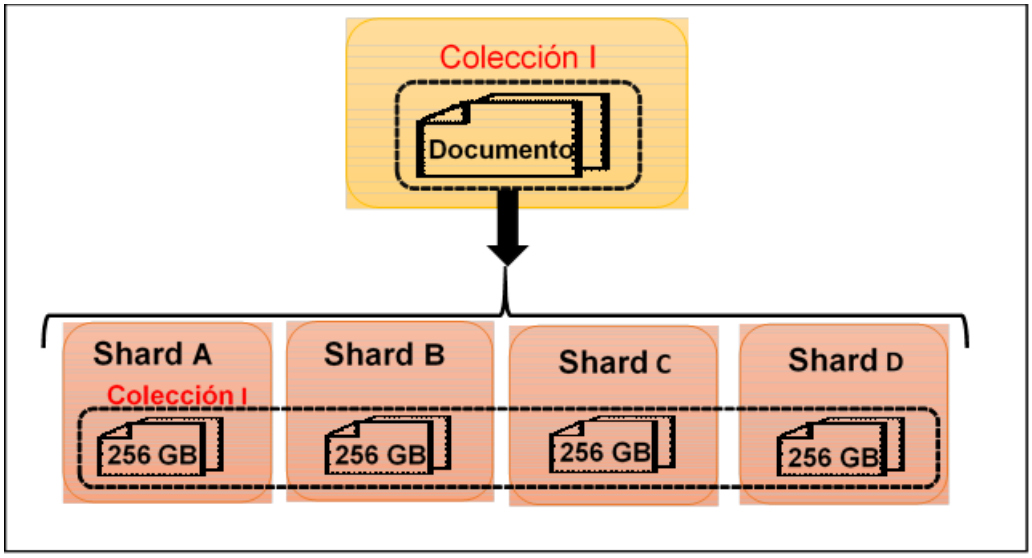


Figura 16.Distribución de datos en shards **Fuente:** (Sarasa, 2016)

2.1 Elementos que conforman un sharded clúster:

Existen 3 tipos de máquinas que son usadas en un sharded clúster.

2.1.1 Shards: Son los encargados de organizar y almacenar toda la información, suelen tener más de 1 nodo y casi siempre son réplica set.

2.1.2 Enrutadores de consultas: No cumplen la función de almacenar algún dato, su misión es trabajar como puente entre las operaciones que realiza el cliente y la redirección hacia los shards donde esta almacenada la información. En un sharded clúster pueden existir más de un enrutador.

2.1.3 Servidores de configuración: Son los encargados de almacenar la metadata del clúster, son consultados por los enrutadores para saber la ruta de redirección de todas las solicitudes. Estos necesitan tener siempre un estado disponible para cualquier cambio que se presente.

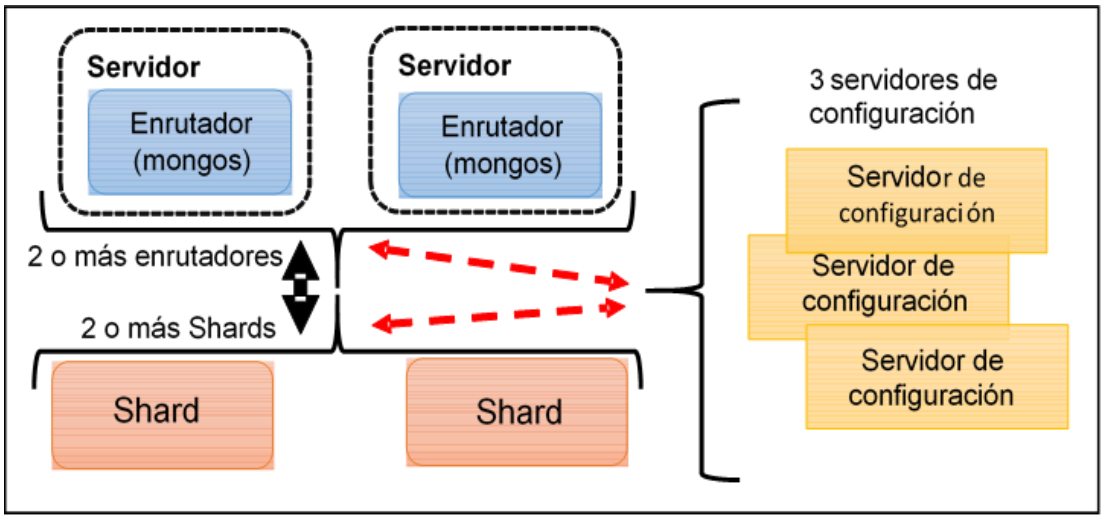


Figura 17.Elementos de un sharded clúster **Fuente:** (Sarasa, 2016)

B) Modelo de Datos

El modelo de datos que utiliza Mongodb es conocido como flexible, porque le da la libertad al usuario que pueda definir la estructura de los datos que se ingresan, a diferencia de las relacionales que la estructura ya está predefinida (Graterol, 2014). Una base de datos puede contener de 1 a más colecciones, estas funcionan como contenedoras de documentos que traen información, es muy igual a la función de una tabla en una base de datos relacional.

DOCUMENTOS: Es la unidad básica de almacenamiento de información para Mongodb, el rol que desempeña es muy similar al que realiza una tupla en una base de datos relacional. Consta de valores los cuales están asociados mediante una clave. Hay algunas excepciones que deben ser tomadas en cuenta. El punto y el símbolo de dólar es necesario que no se incluya en los registros, las mayúsculas y minúsculas tienen que ser diferenciadas en caso esto no se haga Mongodb las considerará como datos diferentes, para cada clave se le asignará un valor si en caso se desea asignarle más de uno el campo valor tendría que ser un arreglo (Sarasa, 2016).

C) Tipos de datos:

Los datos soportados por Mongodb en sus documentos son los detallados a continuación.

Tabla 5
Datos soportados en documentos Mongodb ***Fuente:*** *(Sarasa, 2016)*

Tipo de dato	Descripción	Ejemplo
Nulo	Valor que no existe	{“x”:null}
Booleanos	Puede tomar dos valores	{“x”:true}
Números	Pueden ser números reales o enteros	{“x”:3.14} {“x”:45}
Cadenas	Conformada por caracteres	{“x”:”Ejemplo”}
Fechas	La fecha es almacenada en milisegundos	{“x”:new Date()}.
Expresiones Regulares	Son más utilizadas para consultas en el gestor	
Arrays	Es representado por una lista de valores	{“x”:[“a”,”b”,”c”] }
Documentos Embebidos	Los documentos pueden contener documentos embebidos como valores de un documento padre.	{“x”:{“y”:45}}
Identificadores de objetos	es un identificador de 12 bytes para un documento	{“x”: OjectId()}
Datos binarios	Sirve para poder representar cadena de caracteres no UTF8	
Código JavaScript	Los documentos y las consultas pueden contener código JavaScript.	{“x: function () {…}}

D) CRUD en Mongodb:

Estos comandos sirven para gestionar la información en la base de datos. Las bases de datos no solo funcionan para almacenar información, también puede manipular y trasformar los datos almacenados.

Si hablamos de bases de datos en general, se sabe que existen 4 funciones básicas para el tratamiento de información Create, Read, Update y Delete (Graterol, 2014).

1 CREATE:

Crear es la función elemental de una base de datos, el manejo de información se realiza a través de las colecciones que poseen documentos que aceptan la inserción de información. En Mongodb un documento puede por los siguientes métodos insert, save y también por update pero usándolo de una forma diferente (Graterol, 2014).

1.1 Insert:

El método insert permite añadir a una colección un arreglo de documentos o un único documento, es muy fácil su uso y permite agregar un documento sin identificador o con identificador.

A)

```
MongoDB shell version: 2.4.6
connecting to: test
> documento = {" titulo": "MongoDB" ,
... "autores": [ "Yohan Graterol" , "Flavio Percoco" ] }
{
        " titulo" : "MongoDB",
        "autores" : [
                "Yohan Graterol",
                "Flavio Percoco"
        ]
}
> db.libro.insert(documento)
>
```

B)

```
MongoDB shell version: 2.4.6
connecting to: test
> arreglo = [
... {"documento": 1},
... {"documento": 2}
... ]
[ { "documento" : 1 }, { "documento" : 2 } ]
> db.varios.insert(arreglo)
```

Figura 18.Inserción de Documentos en Mongodb A) Inserción de un documento B) Inserción de varios documentos mediante array **Fuente:** (Graterol, 2014)

1.2 Save

A diferencia del insert este comando permite actualizar un documento si ya está creado.

2 READ

El método read es el encargado de realizar la búsqueda y selección de documentos en la colección. Existen dos métodos utilizados en el read que son el find para buscar y leer todos los documentos que existen en una colección y el método findone que solo nos permite leer uno (Graterol, 2014).

2.1 Find: Selecciona y devuelve todos los elementos de la colección en formato JSON. El formato de salida en que se muestran los datos no es muy claro, para mejorar la salida del mismo tiene que complementarse con el comando pretty seguido del find.

A)

```
MongoDB shell version: 2.4.6
connecting to: test
> db.libro.find()
{ "_id" : ObjectId("5382c84842519556b7de00c5"), " titulo" : "Mon
goDB", "autores" : [  "Yohan Graterol",  "Flavio Percoco" ] }
{ "_id" : ObjectId("5382ca8de6214875e080740a"), "titulo" : "Mong
oDB Básico", "autores" : [  "Yohan Graterol",  "Flavio Percoco"
], "licencia" : "CC", "date" : "2014" }
>
```

B)

```
MongoDB shell version: 2.4.6
connecting to: test
> db.libro.find().pretty()
{
        "_id" : ObjectId("5382c84842519556b7de00c5"),
        " titulo" : "MongoDB",
        "autores" : [
                "Yohan Graterol",
                "Flavio Percoco"
        ]
}
{
        "_id" : ObjectId("5382ca8de6214875e080740a"),
        "titulo" : "MongoDB Básico",
        "autores" : [
                "Yohan Graterol",
                "Flavio Percoco"
        ],
        "licencia" : "CC",
        "date" : "2014"
}
>
```

Figura 19.Uso del método find() A)método find() B)método find() complementado con pretty
Fuente: (Graterol, 2014)

2.2 Find one: La forma de trabajo de este método es similar al de find, la única diferencia es que este método solo devolverá un único valor. Si no se indica ningún parámetro de búsqueda, el método devolverá el primer elemento ingresado de la colección**.**

```
MongoDB shell version: 2.4.6
connecting to: test
> db.libro.findOne()
{
        "_id" : ObjectId("5382c84842519556b7de00c5"),
        " titulo" : "MongoDB",
        "autores" : [
                "Yohan Graterol",
                "Flavio Percoco"
        ]
}
>
```

Figura 20.Uso del método findOne() **Fuente:** (Graterol, 2014)

3 UPDATE:

Mediante esta función se puede realizar modificaciones la data de cualquier documento. Esta función soporta dos métodos, el método save para actualizaciones simples y update para actualizaciones completas.

3.1 Update: Este método solo trabajará en base a un solo documento para ser actualizado, pero al igual que el comando find se le puede asignar diversos parámetros, por ejemplo si esta activada la opción multi en el método update, este actualizará en todos los documentos el campo o los campos que sean indicados. Las actualizaciones en Mongodb son de naturaleza anatómicas, esto quiere decir que las actualizaciones serán ejecutadas por orden de llegada.

```
db.collection.update (
<consulta - criterios>,
<documento_modificado >,
{upsert: true | false, multi : true | false }
)
```

Figura 21.Uso del método update con parámetros **Fuente:** (Graterol, 2014)

4 DELETE:

Para poder eliminar documentos y colecciones en Mongodb, existen 2 métodos que pueden ser utilizados el drop y el remove. Según (Graterol, 2014) por un tema de rendimiento para eliminar colecciones es mejor utilizar drop.

4.1 Remove: Se utiliza para eliminar uno o más documentos que tengan el mismo id en una colección. En caso no se le indicara algún id, este eliminará toda la colección.

```
> db.libro.remove({_id: ObjectId("5382c84842519556b7de00c5")})
> db.libro.findOne()
{
        "_id" : ObjectId("5382ca8de6214875e080740a"),
        "titulo" : "MongoDB Básico",
        "autores" : [
                "Yohan Graterol",
                "Flavio Percoco"
        ],
        "licencia" : "CC",
        "date" : "2014"
}
>
```

Figura 22.Uso de remove y comando findOne para búsqueda de documento eliminado **Fuente:** (Graterol, 2014)

4.2 Drop: Es el método más sencillo y práctico cuando se necesita eliminar toda una colección.

```
MongoDB shell version: 2.4.6
connecting to: test
> db.libro.drop()
true
> db.libro.find()
>
```

Figura 23.Uso del método drop para eliminar colección libro **Fuente:** (Graterol, 2014)

1.3.13. Definición de la Terminología

Fork: en la informática se le conoce a fork como la base que se toma de algo ya creado (software, página web, aplicaciones) para darle un enfoque y finalidad distintos a lo que está ya creado.

Bigdata: se refieren a grandes conjuntos de datos que no pueden ser procesados por procesos o herramientas tradicionales.

JSON (javascript object notation): es un formato de texto, documento de texto ligero que permite el almacenamiento e intercambio de datos, puede ser utilizado en diferentes lenguajes de programación.

CSV (delimitado por comas): es un archivo de datos que permite el trabajo de datos separados por comas. Es uno de los formatos de texto más populares en la actualidad.

DBMS: en español significa Sistema Manejador de Bases de Datos (SGBD), si hablamos de DBMS nos referimos a la colección de Software que sirve como interfaz entre las aplicaciones la base de datos y el usuario.

Blog: es un sitio web que permite al autor o propietario compartir contenidos de su interés personal con los lectores en la red.

Tupla: se refiere básicamente a un registro o fila de una base de datos, representa un objeto con una estructura definida.

Código abierto: es la manera de referirse al software que está basado para la colaboración gratuita, donde la principal ventaja a diferencia del software free es poder modificar el código fuente.

Paradigma: sinónimo de ejemplo o modelo a seguir para usarse en un determinado contexto.

Benchmark: conocida también como punto de referencia, es una técnica que se encarga de medir el rendimiento de un sistema.

1.3.14. Formulación del Problema

¿Cuál sistema gestor de bases de datos NoSQL o Relacional tendrá mejor rendimiento en la gestión de información utilizando pequeños y grandes volúmenes de datos?

1.4. Justificación e Importancia del Estudio

1.4.1. Justificación Social:

La investigación será realizada para que el usuario de hoy en día pueda ser el más beneficiado al momento de encontrar soluciones a los problemas de rendimiento presentados en gestores de bases de datos relacionales, ante el constante crecimiento de los datos. Es poder ofrecer una nueva alternativa que pueda ser tomada en cuenta.

1.4.2. Justificación Académica:

La presente investigación surge de la preocupación por mejorar el rendimiento de gestores de bases de datos donde las personas que intervienen usuarios y desarrolladores web, estudiantes y docentes quienes son los que apoyaran en el estudio e implementación de la misma y será guía de futuros trabajos de investigación a realizar en lo referente a la evaluación de gestores de bases de datos NoSQL los cuales están apareciendo como una opción nueva.

1.4.3. Justificación Tecnológica:

Lo más relevante de esta investigación será evaluar el paradigma que la mayoría de usuarios informáticos tienen en mente sobre los gestores de base de datos NoSQL, donde la mayoría de usuarios que trabajan con gestores de bases de datos relacionales, piensan que los gestores de bases de datos NoSQL no son tan eficientes como los gestores de bases de datos Relacionales, o que solo están hechos para poder trabajar con grandes volúmenes de datos. La mayoría de profesionales inmersos en esta especialidad tienen esta idea en mente, debido al poco conocimiento que tienen de todas las ventajas de usar bases de datos NoSQL.

1.4.4. Hipótesis

Ho= Que el Gestor de base de datos relacional sea mejor en rendimiento que el sistema de base de datos NoSQL.

Ha= Que el Gestor de base de datos NoSQL sea mejor en rendimiento que el sistema de base de datos relacional.

1.5. Objetivos

1.5.1. Objetivo General:

Comparar el rendimiento de bases de datos relacionales y no relacionales.

1.5.2. Objetivos específicos:

A. Seleccionar gestores de bases de datos.

B. Establecer métricas para medir el rendimiento de gestores elegidos.

C. Definir Equipos de trabajo con bases de datos Seleccionadas.

D. Ejecutar pruebas de Rendimiento.

2. MATERIAL Y METODO

2.1. Tipo y Diseño de Investigación

2.1.1 Tipo de la Investigación:

El tipo de investigación es Cuantitativa debido que en el presente proyecto los indicadores están representados a través de cantidades numéricas, mediante estos se medirán resultados para poder emitir estadísticas.

2.1.2 Diseño de Investigación:

El diseño es de tipo experimental, debido a que en la investigación se harán experimentos manipulando las variables dependientes. Para la evaluación de las pruebas de rendimiento y consumo de recurso se utilizará dos gestores de base de datos, un relacional y un NoSQL, los cuales tendrán data de pueblos jóvenes de un portal de datos abiertos del estado peruano, mayor información en el **ANEXO 5**.

A) EQUIPO DE TRABAJO:

El software utilizado como gestor de base de datos relacional fue la versión **MARIA DB SERVER 10.6.5 64 bits**. Para la administración de la base de datos de manera grafica se utilizó **HeidiSQL versión 11.2.0.6392**.

B) CANTIDAD DE REGISTROS:

Se trabajo con 1090000 tuplas.

C) PRUEBAS DE RENDIMIENTO:

Para poder realizar la recopilación de las pruebas de rendimiento y consumo de CPU fue necesario el uso de herramientas del Sistema Operativo. Se utilizo el Monitor de Recursos y el manejador HeidiSQL donde los datos fueron ingresados de manera incremental de menor a mayor, los registros fueron de 1000 a 1090000.

D) METODO DE POBLACION DE BASE DE DATOS:

Fue necesario realiza inserción de datos aleatorios para tener una cantidad mayor de tuplas, el proceso de población se detalla en el **ANEXO 4,** donde fue necesario montar un servidor local **XAMP** donde se hizo uso de **APACHE y PHP** versión 7 para poder correr los scripts.

Para el desarrollo de esta investigación fue importante saber que gestor de base de datos NoSQL se iba utilizar, para que pueda ser una comprobación justa fueron necesarios los siguientes criterios.

A) ALMACENAMIENTO DE DATOS:

Las bases de datos de tipo documental son las que mejor guardan datos de manera semiestructurada y con diferentes tipos de información en un solo registro.

B) RAPIDO TIEMPO DE CONSULTA:

Las bases de datos de tipo documental utilizan el paradigma de programación **MAP-REDUCE** para mejorar tiempos de consulta y disminuir el consumo de recursos del equipo.

Para el caso de NoSQL el gestor de base de datos elegido fue MONGO DB.

2.2. Población y Muestra

2.2.1 Población:

La población está conformada por todo el universo de gestores de bases de datos relacionales y NoSql, ya sean de tipo comercial o de código libre. En los últimos años se han mantenido en los primeros puestos los gestores detallados a continuación:

Tabla 6
Doce primeros puestos ranking DB-Engines **Fuente:** (DB-Engines, 2020)

Rank	Gestor de Base de datos	Tipo
1	Oracle	Relacional,Multi-modelo
2	MySQL	Relacional,Multi-modelo
3	Microsoft SQL Server	Relacional,Multi-modelo
4	PostgreSQL	Relacional,Multi-modelo
5	MongoDB	Documental,Multi-modelo
6	IBM Db2	Relacional,Multi-modelo
7	Redis	Llave-valor, Multi-modelo
8	Elasticsearch	Motor Busquedad, Multi-modelo
9	SQLite	Relacional
10	Cassandra	Columna
11	Microsoft Access	Relacional
12	Mariadb	Relacional,Multi-modelo

2.2.2 Muestra:

Para la presente investigación la muestra fue necesario seleccionar dos gestores de bases de datos un relacional y un NoSQL.

Tabla 7
Muestra de gestores de bases de datos elegidos para la presente investigación. **Fuente:**Elaboración Propia

Gestor de Base de datos	Tipo
Mariadb	Relacional,Multi-modelo
Mongo db	Documental,Multi-modelo -NoSQL

2.3. Variables Operacionalización

2.3.1. Variable Dependiente

Rendimiento de gestores de base de datos

2.3.2. Variable Independiente

Gestores de bases de datos Relacionales y NoSQL

2.3.3. Operacionalización

Tabla 8
Operacionalización de variables

Variable	Dimensiones	Indicador	Formulas	Técnica e instrumento de recolección de datos
Independiente: Gestores de Bases de datos	Recursos	Uso de RAM	$\boldsymbol{UMR} = \boldsymbol{UMF} - \boldsymbol{UMI}$	Ficha de registro
		Uso de CPU	$\boldsymbol{UCP} = \boldsymbol{UCPF} - \boldsymbol{UCPI}$	Ficha de registro
Dependiente: Rendimiento de Gestores de base s de datos	Tiempo	Tiempo de Respuesta	$\boldsymbol{TR} = \boldsymbol{tiempoxejecuci}$ó	Ficha de registro

2.4. Técnicas e Instrumentos de recolección de datos, validez y confiabilidad

2.4.1. Técnicas de Recolección de Datos:

En la investigación se aplicó la observación científica y el análisis documental, de manera que la información recolectada pueda ser utilizada para el pronóstico del estado actual del rendimiento de los gestores de bases de datos elegidos a estudiar.

A. La observación científica:

Esta técnica consiste en poder ver todos los eventos que se vienen desarrollando durante la investigación, donde el principal ejecutor de estas es el investigador.

B. Análisis Documental:

Esta técnica consiste en registrar todos los datos obtenidos de los sucesos presentados durante la observación científica, con el fin de presentar resultados en todo el desarrollo de la investigación.

2.5. Procedimientos de Análisis de Datos:

En la presente investigación los datos fueron recolectados de la siguiente forma:

2.5.1. Observación:

Se realizó una observación estructurada del rendimiento de los gestores de bases de datos, teniendo claro los aspectos ya definidos que se tomarán en el estudio para un posterior análisis documental. Se pudo observar el comportamiento y creación de tablas del modelo ER, ejecución de comandos y Consultas anidadas.

2.5.2. Ficha de Registro:

Los datos que se documentaron en esta investigación fueron los tiempos de respuestas en creación de bases de datos en los gestores seleccionados, errores en proceso de migración del gestor de base de datos relacional SQL Server a Mariadb., errores de proceso de creación de base de datos NoSQL Mongodb, documentado de prototipos creados para las pruebas respectivas, documentado de procesos de ambas bases de datos.

Fue necesario aplicar el uso de fórmulas a los datos registrados en la ficha de registro, para poder presentar los resultados finales, a continuación, se detallan las fórmulas utilizadas en la investigación. Mayor detalle en **ANEXO 2**.

A) Uso de memoria RAM:

Con esta fórmula se pudo medir la cantidad de memoria RAM asignada por el sistema, la cual no es compartida con ningún otro proceso, también conocida como memoria privada, lo cual permitió saber el consumo que tenga cada gestor al realizar cualquier acción, ya sea de creación, inserción, actualización o borrado. Fue necesario que para cada interacción realizada se borre la memoria caché y archivos temporales del sistema operativo, logrando así poder obtener información precisa del proceso ejecutado. Los datos son obtenidos en KB.

$$\boldsymbol{UMR = UMF - UMI}$$

En donde:

UMI =El uso de Memoria RAM utilizado al inicio de la ejecución del proceso.

UMF =El uso de Memoria RAM utilizado al término de la ejecución del proceso.

B) Uso de CPU:

Mediante esta fórmula se pudo saber el uso que tiene el microprocesador del computador al realizar cualquier instrucción en el SGDB, los datos obtenidos son en función al porcentaje de consumo medio de la CPU a 60 segundos por proceso.

$$\boldsymbol{UCP = UCPF - UCPI}$$

En donde:

UCPF=Porcentaje de uso de CPU al terminar la ejecución del proceso

UCPI=Porcentaje de uso de CPU inicial al comenzar la ejecución del proceso.

C) Tiempo de Respuesta:

Mediante esta fórmula se obtendrá el tiempo de respuesta en milisegundos que los gestores entregan al momento de ejecutar cualquier proceso.

$$\boldsymbol{TR = tiempoxejecuci\acute{o}n}$$

En donde:

tiempoxejecución=El tiempo que el gestor demore en procesar cualquier proceso.

2.6. Criterios Éticos

2.6.1. Responsabilidad:

Para la presente investigación se tomará en cuenta el uso de los datos con los que se trabajará ya que es información predial del estado, donde prima el compromiso para poder salvaguardar y hacer uso de los mismos con un fin educativo el cual permita aportar conocimientos a la comunidad científica.

2.6.2. Social:

En esta investigación se tiene como propósito realizar la evaluación de rendimiento de sistemas de bases de datos, contando con el apoyo de profesionales para obtener mejores resultados.

2.7. Criterios de Rigor Científico

2.7.1. Coherencia Metodológica:

El proyecto cumplirá con las expectativas expresadas en su contenido y su implementación con el estándar y políticas para su desarrollo.

2.7.2. Validez:

Datos obtenidos para la investigación serán evaluados y tratados para poder hacer semejante la realidad en el crecimiento de información usada en una base de datos en la actualidad.

2.7.3. Consistencia: La investigación representará material consistente y certificado por la comunidad científica. Los cuales han sido recopilados y seleccionados para aportar un mejor enfoque a la investigación.

3. RESULTADOS

3.1. Resultados en Tablas y Figuras

Para poder evaluar los resultados encontrados en esta investigación se clasificaron del mismo orden a como se desarrolló el aporte practico.

A) CREACION DE BASES DE DATOS:

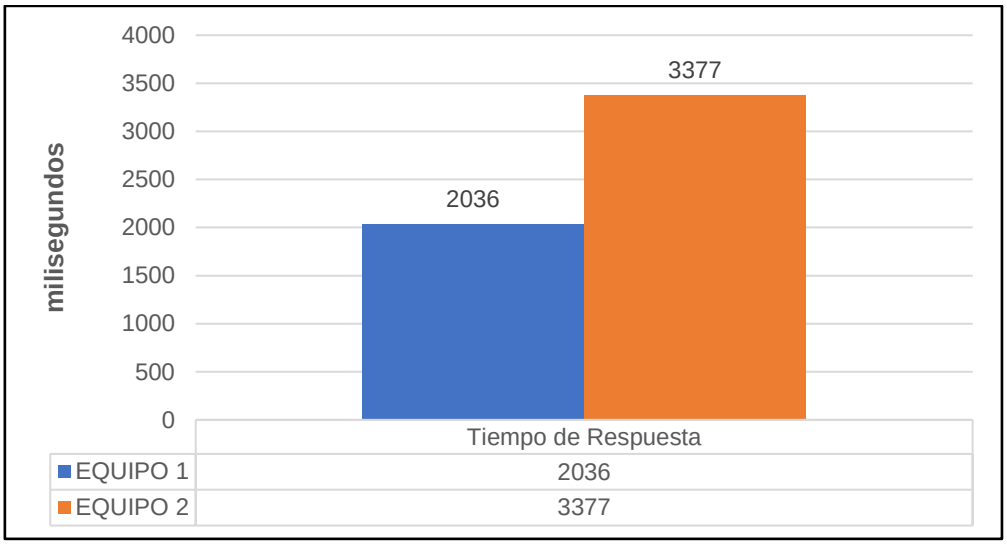


Figura 24.*Tiempo de Respuesta para crear base de datos Mariadb* ***Fuente:*** *Elaboración Propia*

La diferencia de tiempo encontrada en los dos equipos donde se medía el tiempo de respuesta de la base de datos relacional Mariadb al momento de ejecutar el script de su creación, nos indica que la variación de tiempo no depende de los recursos con los que cuente el equipo, si no con el tamaño de líneas que este tenga.

En ambos casos se utilizó el mismo archivo y solo hubo una diferencia de 1341 milisegundos.

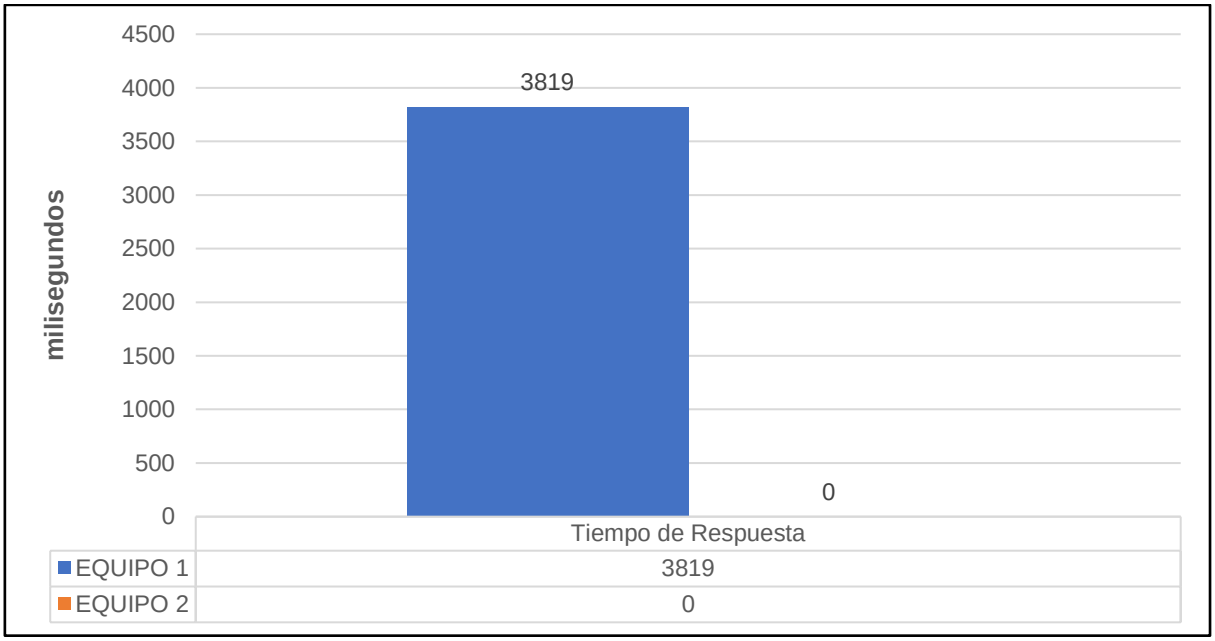


Figura 25.*Tiempo de respuesta en creación de base de datos Mongodb* ***Fuente:*** *Elaboración propia*

Para el equipo 2 se encuentra con un valor de 0, no se logró recolectar ninguna información porque el proceso de migración no pudo ser ejecutado en este equipo debido a las prestaciones en recursos que el software Pentaho Data Integration necesitaba, al no tener los recursos suficientes se evitó desarrollar este proceso en este equipo, la información solo fue recolectada para el equipo 1 que es el de mayores prestaciones.

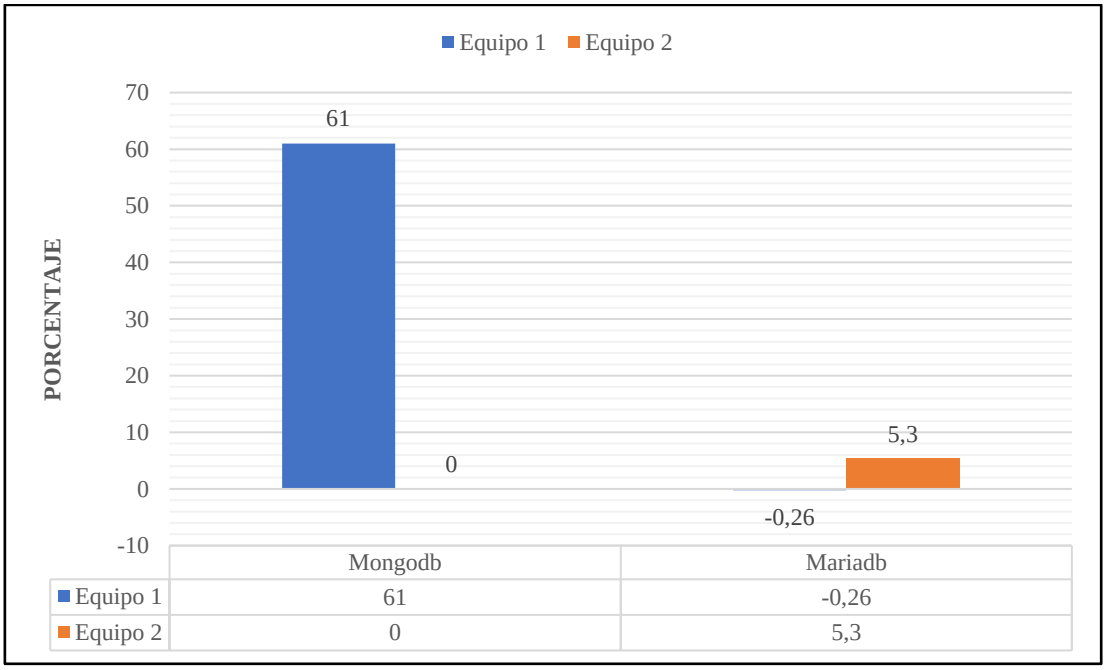


Figura 26.*Consumo de CPU en proceso de creación de base de datos ambos gestores.* ***Fuente:*** *Elaboración propia*

En esta figura el valor de Mariadb en el equipo 1 tiene un valor negativo, este valor aparece debido a la diferencia que se genera cuando el consumo final de CPU es menor al consumo inicial. Esta variación solo la encontramos en Mariadb debido a las nuevas funciones incorporadas en el meto asíncrono del motor InnoDB, la cual es encargada de generar una mini cache en el motor de almacenamiento para disminuir el uso de recursos en el gestor. El consumo de CPU para la creación de base de datos en Mongodb fue mayor en el equipo que presentaba mejores recursos.

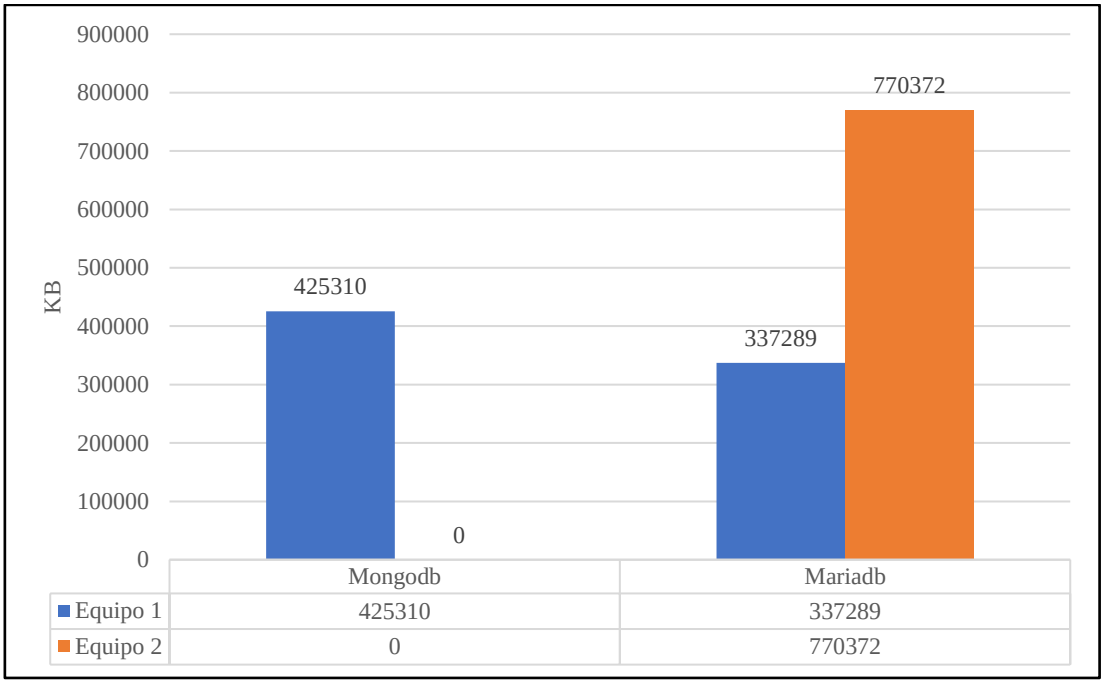


Figura 27.Uso de memoria RAM en proceso de creación de base de datos **Fuente:** Elaboración propia

En el uso de la RAM para ambos gestores fue mucho mayor para Mariadb,en el caso de Mongodb fue mínimo pero comenzó con un uso alto de 425310 KB sobre 3373289 KB sobre Mariadb.El uso disminuyó considerablemente cuando tomo unos segundos de comenzar a ejecutar la sentencia.

C) PROMEDIO DE USO DE RAM EN INSERCIONES (BYTES):

Para un mejor análisis se separó los gráficos de acuerdo a los equipos en los que fueron ejecutados.

1) EQUIPO 1:

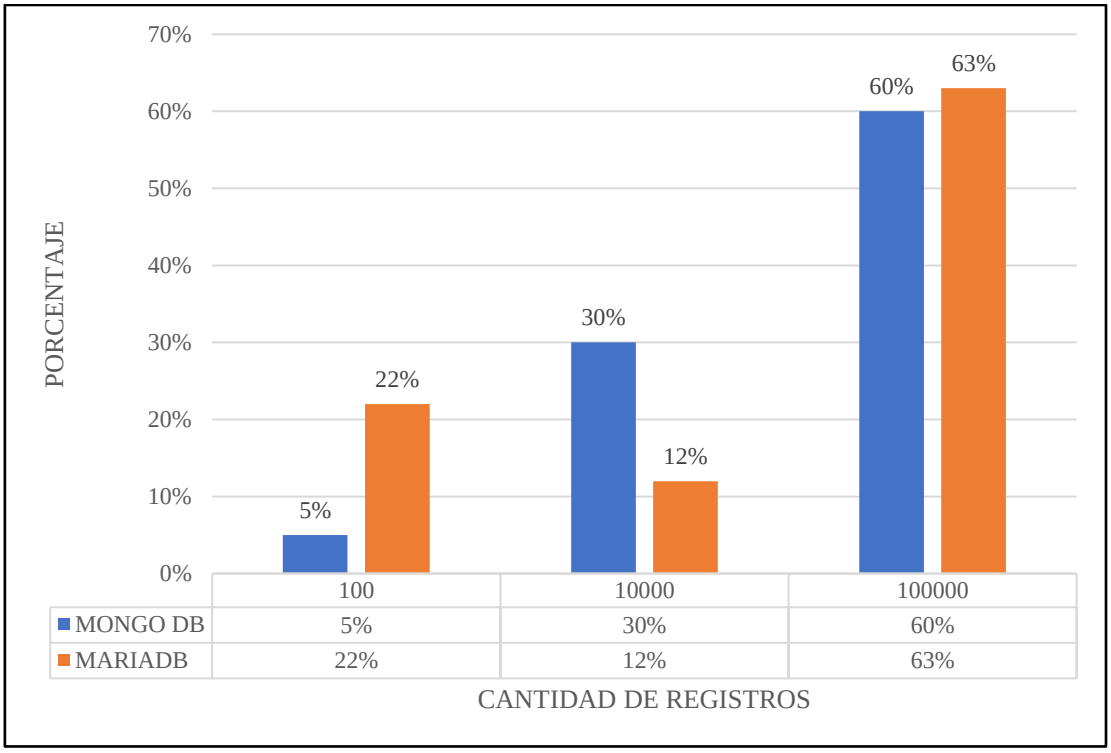


Figura 28.Promedio de uso de memoria RAM en Equipo 1 **Fuente:** Elaboración propia

En esta figura se observa que Mariadb disminuye el uso de la RAM cuando se realizan inserciones después de la segunda etapa, para Mongodb el uso no disminuye si no aumenta.

2) EQUIPO 2

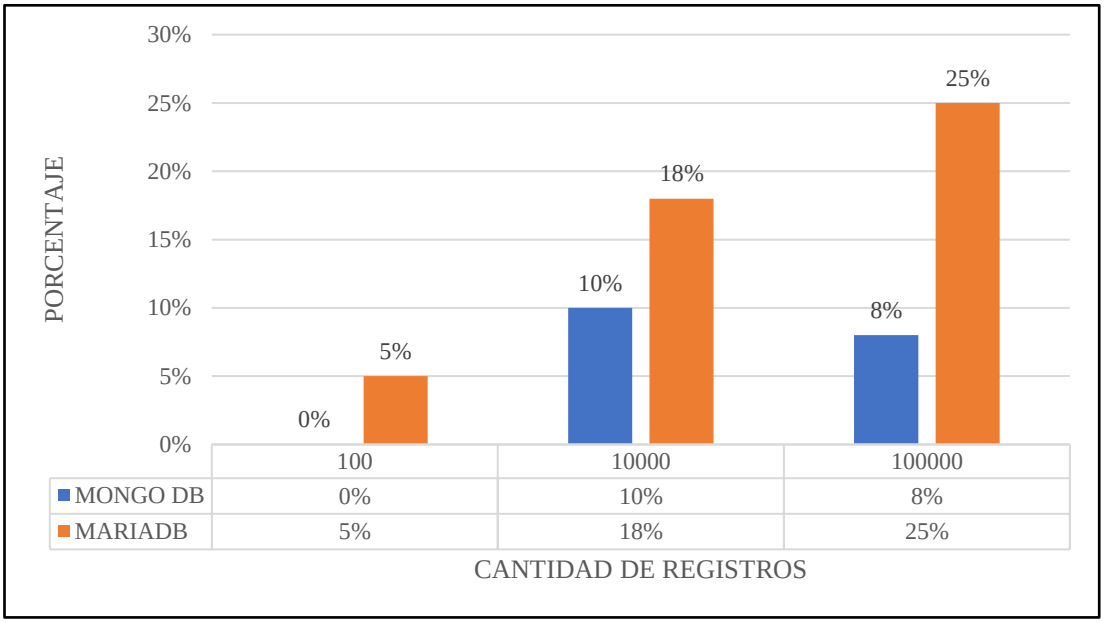


Figura 29.Promedio de uso de memoria RAM en Equipo 2 **Fuente:** Elaboración propia

En este equipo que tiene menos recursos que el equipo 1, Mariadb utiliza menor uso de RAM que Mongodb; aquí Mongodb disminuye el uso cuando se maneja gran cantidad de datos.

C) PROMEDIO DE USO DE CPU EN CONSULTAS (PORCENTAJE)

1) EQUIPO 1:

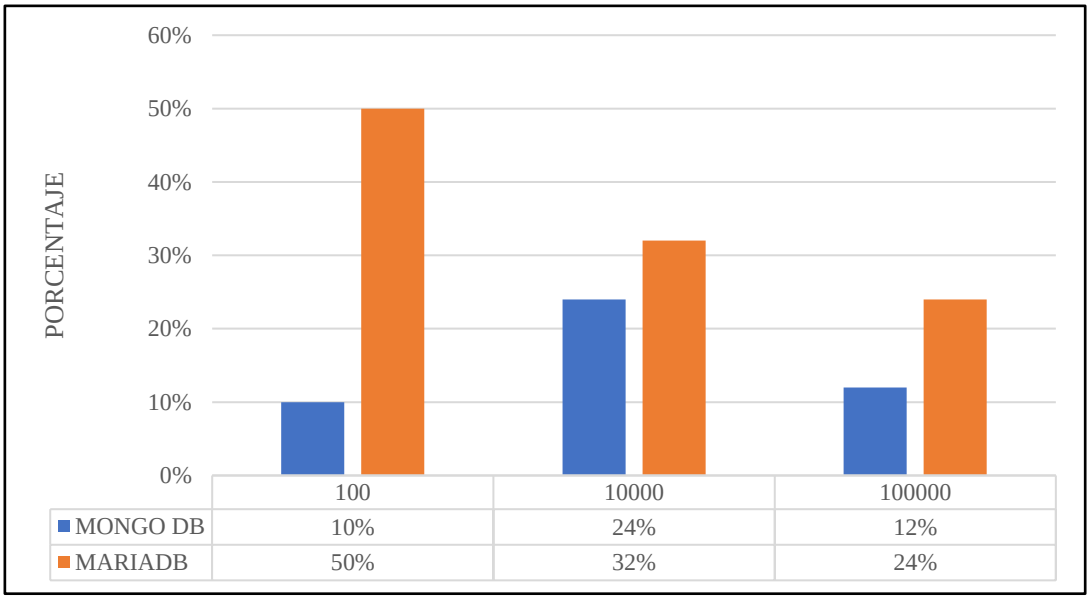


Figura 30.Prueba de uso promedio de CPU en consultas equipo 1 **Fuente:** Elaboración propia

2) EQUIPO 2:

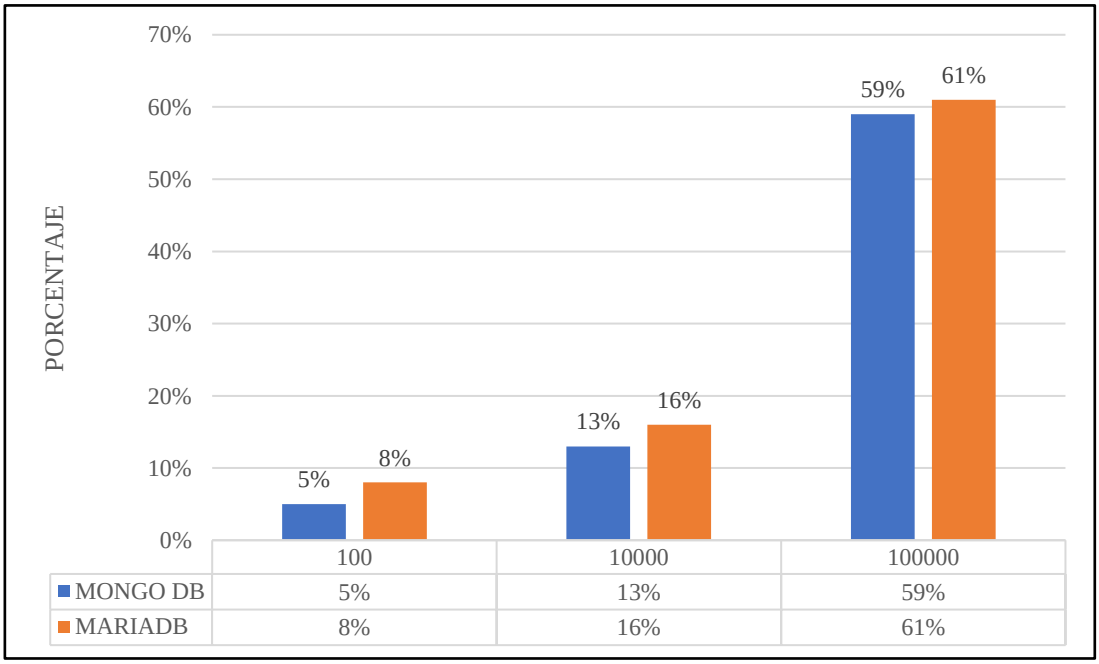


Figura 31.Prueba de uso promedio de CPU en consultas equipo 2 **Fuente:** Elaboración propia

El uso en promedio de CPU en el equipo 2 va muy parejo en ambos gestores, pero Mongodb va por muy debajo de Maridb, mongodb mejora su rendimiento considerablemente cuando se ejecuta en un computador con procesador AMD.

3.2. Discusión de resultados:

Comparando el tiempo de respuesta en segundos obtenido de las consultas del gestor Mongodb con las demás investigaciones que ejecutaron la misma cantidad de datos, se detalla lo siguiente.

Tabla 9.
Discusión de resultados en tiempos recopilados en pruebas de rendimiento en consultas Nosql-Mongodb ***Fuente:*** *Elaboración propia*

INDICADOR	AUTOR	PRUEBAS REALIZADAS		
		100	**10000**	**100 000**
Tiempo de Respuesta en Consultas	Investigación Actual (Moreno, Quintero, & Rueda, 2016)	1.13 250.69	6.398 439.398	16.5 950.39
	(Győrödi, Győrödi, G.Pecherle, & Olah, 2016)	15.39	198.36	790.121
	(Krstića & Krstićb, 2018)	1.39	8.654	29.065

La diferencia que existe en cuanto a segundos entre las investigaciones de Moreno y Pecherle es debido a que estos dos usan equipos convencionales de sobremesa con procesadores y poca memoria RAM, en cuanto a Krstića y la presente utilizan servidores para sus pruebas. Hay una variación considerable a la investigación de Krstića,debido a que usaron versiones anteriores al software utilizado en esta investiagacion, cada version trae mejoras nuevas y parches de caracteristicas nuevas. Krstića utiliza el método de desagregado de columnas del modelo relacional para migrar la base de datos relacional a Nosql y la insercion de datos mediante colecciones complementarias ayudadas por una aplicación de escritorio, en esta investigacion los datos fueron trabajados directamente en los gestores, no se utilizó ninguna aplicación de escritorio, ni web para obtener tiempos de respuesta mas exactos.

3.3. Aporte práctico

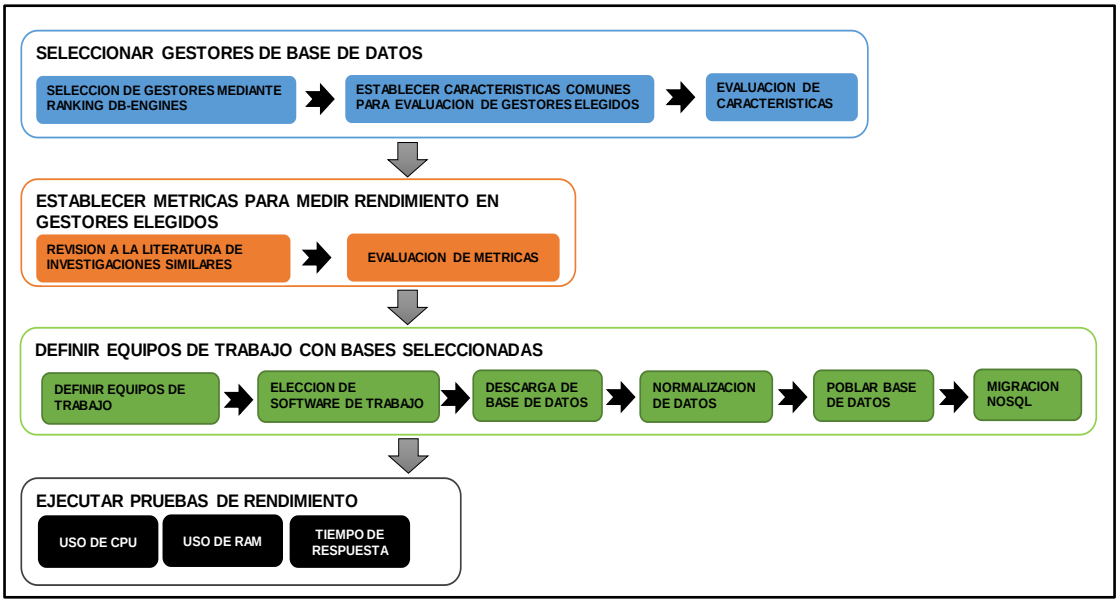


Figura 32. Resumen de Aporte práctico **Fuente:** Elaboración propia

En esta investigación se tomó como referencia el ranking de gestores de bases de datos que ofrece mensualmente la página DB-Engines00. Se eligió trabajar con este ranking debido a que es respaldado por la mayoría de gestores de bases de datos comerciales y no comerciales; además cuenta con un blog donde promueve la colaboración activa de miembros, una enciclopedia donde se clasifican a los gestores de acuerdo a su forma de trabajo.

Esta web se encarga de promocionar eventos, noticias y publicar artículos nuevos para la comunidad, es un producto de SolidIT GmbH, empresa austriaca constituida en el 2011 dedicada a consultoría informática y desarrollo empresarial con certificaciones ISO/ IEC 25012, ISO/IEC 25024 y CompTIA Network. La página elabora el ranking mensual, promediando valores obtenidos de los siguientes criterios.

A. Número de menciones del gestor en sitios web: Se utiliza Google y Bing para poder medir las consultas en los gestores de búsquedas ingresadas por los usuarios. Para tener resultados exactos.

B. Interés general en el gestor: Este criterio es medido obteniendo la frecuencia de búsquedas del gestor utilizando la herramienta Google Trends.

C. Frecuencia de las discusiones técnicas sobre el sistema: Aquí la página se ayuda de todas las preguntas hechas por los usuarios en dos de los mejores sitios de ayuda y respuestas de soporte de TI, Stack Overflow y DBA Stack Exchange.

D. Número de ofertas de trabajo en las que se menciona el sistema: El valor se obtiene de las ofertas publicadas en los principales motores de búsqueda de empleo, entre ellos Simply Hired,Indeed.

E. Número de perfiles en redes profesionales: La página se apoya en menciones hechas en la red de trabajo profesional LinkedIn, la cual tiene una reputación ganada a nivel internacional.

F. Relevancia en redes sociales: Se toma todas las menciones hechas en Twitter, en los que se menciona el nombre del gestor de base de datos.

Tabla 10
Ranking DB-Engines de los 10 primeros Gestores de bases de datos **Fuente** (DB-Engines, 2020)

Ranking	Sistema Gestor de Base de Datos	Modelo de Base de Datos	Puntuación
1.	Oracle	Relational, Multi-model	1369.36
2.	MySQL	Relational, Multi-model	1264.25
3.	Microsoft SQL Server	Relational, Multi-model	1062.76
4.	PostgreSQL	Relational, Multi-model	542.29
5.	MongoDB	Document, Multi-model	446.48
6.	IBM Db2	Relational, Multi-model	161.24
7.	Redis	Key-value, Multi-model	151.86
8.	Elasticsearch	Search engine, Multi-model	150.50
9.	SQLite	Relational	126.68
10.	Cassandra	Wide column	119.18

Este ranking también fue utilizado por (Győrödi, Győrödi, G.Pecherle, & Olah, 2016), (Moreno, Quintero, & Rueda, 2016) y (Krstića & Krstićb, 2018).En los cinco primeros puestos del ranking, se encontraron 4 gestores de bases de datos relacionales y uno NoSQL de tipo documental. El gestor de base de datos NoSQL Mongodb ubicado en el puesto cinco, es el que mejor se asemeja a uno de tipo relacional. Se realizó una evaluación de los gestores de acuerdo a su estructura y su forma de trabajo;(A.Gupta, S.Tyagi, N.Panwar, S.Sachdeva, & Saxena, 2017, p. 296) tambien utilizaron la misma evaluacion en su investigacion.

De las características elegidas para la evaluación se seleccionaron las más relevantes y se les asigno una X a los gestores que si cumplían con esta característica y a los que no cumplían simplemente se dejó el campo vacío. Posteriormente se realizó una suma de todas las X para poder asignarles un valor a cada característica; las características elegidas fueron las siguientes.

Código libre: También conocido como código libre nos permitirá conocer de los sistemas gestores de bases de datos elegidos cuales son de ámbito comercial y necesitan de una licencia para poder utilizarlos.

Filas: Es el modo de almacenamiento que tienen los gestores de bases de datos para poder agrupar y guardar la información.

Soporte XML: Se evalúa la posibilidad de que se soporte el manejo de estructuras XML para operaciones de inserción de datos, haciendo más fácil la importación desde el gestor, sin utilizar algún software de terceros para completar la operación.

Disparadores: Característica que permite desencadenar una o más secuencias al realizar alguna operación de inserción, actualización o eliminado de datos. Se evalúa esta característica ya que se piensa poblar la base de datos duplicando registros con un proceso automático.

SQL: Se evalúa el uso del lenguaje estructurado para el proceso de creación de la base de datos y poblado de la muestra, esta característica solo aplicaría para los gestores de bases de datos relacionales.

Tabla 11
Evaluación de Características de los 5 primeros gestores de Bases de Datos elegida del ranking DB-Engines Fuente Elaboración Propia

CARACTERITICAS	**ORACLE**	**MySQL**	**MICROSOFT SQL SERVER**	**POSTGRESQL**	**MONGODB**
Código libre		X		X	
Filas	X	X	X	X	
Soporte XML	X	X	X		
Disparadores	X	X	X	X	X
SQL	X	X	X	X	
TOTAL	**4**	**5**	**4**	**4**	**1**

En base a los resultados expuestos anteriormente en la tabla 4, MySQL fue el de mejor puntuación obteniendo 5 puntos precedidos por Oracle, Microsoft SQL Server y PostgreSQL, la mayoría de investigaciones a la fecha han utilizado MySQL ya que es un gestor que ha venido consolidándose con el tiempo; en la actualidad existe un gestor de base datos relacional llamado Mariadb que es un fork de MySQL con mejoras hechas a los mecanismos de almacenamientos de datos, creado a partir de la compra de Oracle a MySQL.

Mariadb utiliza la misma sintaxis de MySQL, para esta investigación se eligió trabajar con Mariadb en remplazo de MySQL.Se elaboró un cuadro de propiedades

generales para conocer un poco más a detalle los gestores de bases de datos elegidos.

Tabla 12
Propiedades de Gestores de Bases de datos elegidos Fuente Elaboración Propia

Propiedad	**Mongodb**	**Mariadb**
Modelo	Orientado a Documentos	Orientado a Filas
Tipo Registro	JSON	Fila
Licencia	Open Source	Open Source
Sistemas Operativos	Linux	FreeBSD
Servidor	OS X	Linux
	Solaris	OS X
	Windows	Solaris
		Windows
Lenguaje de desarrollo	C ++	C y C ++
Áreas de uso	Almacenamiento y registro de eventos, Sistemas gestión de contenidos (CMS), bases de datos históricas	Sistemas empresariales, publicidad, páginas web
Código Abierto	SI	SI

Se realizó una revisión a la literatura de investigaciones similares a esta, donde se pudo identificar las métricas utilizadas por diversos autores y que sirvieron para el desarrollo de esta investigación, a continuación, se presentan las métricas identificadas las cuales fueron sometidas a una puntuación basada en la aparición por investigación. Se utilizó un color único para cada métrica, para que en la tabla de puntuación sea más fácil de poder ubicar el puntaje obtenido.

Tabla 13
Métricas según investigaciones Fuente Elaboración Propia

N	Articulo	Métrica	Autores
1	Una Comparación del rendimiento de base de datos de MariaDB y MySql con la carga de trabajo OLTP	Consumo de CPU Consumo de RAM Rendimiento en carga soportada	S. Tongkaw A. Tongkaw
2	A Comparative Study: MongoDB vs. MySQL	Tiempo de respuesta en Consultas	C. Gyrödi, R. Gyrödi, G. Pecherle, A Olah
3	Un análisis comparativo de rendimiento de consultas en Data Warehouses en MongoDB vs SQL Server	Tiempo de respuesta en Consultas Almacenamiento en Disco	Pereira, P. Oliveira ,F. Rodríguez
4	Base de Datos NoSQL: MongoDB vs. Cassandra en operaciones CRUD	Tiempo de respuesta en Consultas	J. Castillo, J. Garcés, M. Navas, D. Segovia
5	Comparación Del Rendimiento De Los Comandos Insert, Select Y Delete En Los Sistemas Gestores De Bases De Datos Oracle Y Mysql	Tiempo de respuesta en Consultas	J. Portilla Tovar, M. Bernal Gómez
6	Uso de bases de datos NoSQL documentales para crear sitios web de alto rendimiento	Rendimiento en carga soportada Consumo de CPU	L. Mancilla
7	La Evaluación del rendimiento de las operaciones CRUD en una base de datos orientada a documentos replicados de forma asíncrona.	Tiempo de respuesta en consultas Almacenamiento en Disco	C. Truica, F. Radulescu, Boicea A., and Bucur I.
8	Análisis comparativo de los sistemas de gestión de bases de datos relacionales seleccionados.	Tiempo de respuesta de consultas	Poljak, R.; Poščić, P., Jakšić, D.
9	Una comparación de rendimiento entre Oracle y Mongodb	Consumo de RAM Consumo de CPU	F. Moreno, J Quintero, R. Rueda
10	NoSQL: Death to Relational Databases	Almacenamiento en Disco Consumo de RAM Consumo de CPU	Scofield

Tabla 14
Puntuación de métricas según aparición en investigaciones anteriores Fuente Elaboración Propia

METRICA	1	2	3	4	5	6	7	8	9	10	PUNTUACION
Consumo de CPU	X								X	X	3
Consumo de RAM	X					X			X	X	4
Tiempo de respuesta en Consultas		X	X	X	X		X	X			6
Almacenamiento en Disco			X				X				2

Entre las métricas más utilizadas por los autores se encontraron tiempo de respuesta en consultas, consumo de RAM y consumo de CPU, estás fueros las elegidas para analizar el rendimiento de los gestores en esta investigación.

Fue necesario poder preparar los escenarios físicos de trabajo en el cual se iba a desarrollar la investigación. En esta preparación los parámetros que se tomaron en cuenta fueron los siguientes.

A) HARDWARE: Para poder medir el rendimiento de gestores de base de datos elegidos, se seleccionaron dos equipos, uno con grandes recursos y otro con recursos menores, con el único fin de que se pueda saber si los resultados sean variables en función a los recursos del equipo. El equipo de trabajo 1 es un servidor Lenovo y el equipo de trabajo 2 es una laptop Hp convencional considerada de gama alta/media.

Tabla 15.
Equipo de trabajo 1 Servidor ***Fuente:*** *Elaboración Propia*

CARACTERISTICA	DESCRIPCION
PROCESADOR	Intel Xeon(R) CPU E3-1245 v5 3.50GHz
MEMORIA RAM	24 Gb
DISCO DURO	1 Tb
SISTEMA OPERATIVO	Windows Server 2016 Datacenter
MARCA	Lenovo
MODELO	ThinkServer 1075

Tabla 16.
Equipo de trabajo 2 laptop convencional ***Fuente:*** *Elaboración Propia*

CARACTERISTICA	DESCRIPCION
PROCESADOR	AMD Ryzen 7 3700 U (8 procesadores lógicos)
MEMORIA RAM	16 Gb
DISCO DURO	500 Gb SSD
SISTEMA OPERATIVO	Windows 10 Home Single Language 64 bits(compilación 19041)
MARCA	Hp
MODELO	Pavilion 15

B) SOFTWARE UTILIZADO: Para todos los procesos realizados como las pruebas de rendimiento, recolección de datos e interpretación de resultados se utilizó software libre para evitar el uso de licencias comerciales.

1. MongoDB 4.0 Community Edition: Se necesita realizar la instalación del servidor Mongodb, se siguió con toda la documentación citada en la página oficial del software. También fue necesaria la instalación de Mongodb Compass Community para fines de conexión.

2. Mariadb 10.4 X64: La instalación para Mariadb fue sencilla, configurándose como instalación mínima predefinida por el usuario, en caso el puerto de comunicación 3309 que es el utilizado por defecto este ocupado tendría que cambiarse por otro al realizar la instalación.

3. **Robo3T 1.3.1:** Este cliente fue utilizado para poder obtener los resultados obtenidos en las pruebas de rendimiento en gestores de bases de datos NOSQL.

4. **HeidiSql 10.2.0.5599:** Software utilizado para obtención de tiempos de ejecución en milisegundos en la ventana de salida a la ejecución de comandos. Para futuras investigaciones no es recomendable que se instale la versión portable de este programa, ya que puede presentar errores cuando se hagan pruebas de rendimiento con grandes cantidades de datos.

5. **Sublime Text Build 3176:** Este software fue utilizado para poder realizar la adaptación de datos de la muestra. Se utilizó este editor de texto por referencias de otras investigaciones para procesar gran cantidad de datos,(Győrödi, Győrödi, G.Pecherle, & Olah, 2016) también usaron este editor de texto.

6. **PDI (Pentahoo Data Integration):** Este software es el más importante en esta investigación ya que permitió realizar el proceso de migración de estructura relacional al modelo NOSQL.

C) PREPARAR BASE DE DATOS

La preparación de los datos fue trabajada en el escenario 1, ya que este contaba con mejores recursos que el escenario 2. Se trabajó con una base de datos abierta predial de la Entidad Cofopri la cual contenía información de los pueblos y caseríos formalizados de todo el Perú. Para mayor información de la procedencia de los datos se detalla un informe adjuntado en el **ANEXO 5.**

Se procedió con la descarga de toda la data, esta se encontraba clasificada por fechas fue necesario realizar la descarga de todas las partes para poder unirla.

Figura 33. Descarga de datos para pruebas de rendimiento **Fuente:** Elaboración propia

Al momento que se realizó la apertura del archivo XLSX los archivos tenían columnas en común entre todos los archivos descargados, se encontraron publicaciones donde los archivos se les había agregado nuevas columnas, Se realizó una normalización de datos, obteniendo así un total de 6 tablas; las cuales son:

Tabla 17
Tablas del modelo relacional después del proceso de normalización ***Fuente:*** *Elaboración Propia.*

N°	ENTIDAD
1	departamento
2	provincia
3	distrito
4	clasificación
5	depreciación
6	pueblo

Para un mayor detalle de los campos que conforman el modelo relacional revisar el **ANEXO 3.**

D) POBLADO DE DATOS:

Luego de haber ingresado y creado las entidades fuertes se procedió a crear las entidades débiles e ingresar todos los datos correspondientes de la data trabajada, fue necesario poblar la base de datos para tener una mayor cantidad de registros para las pruebas de rendimiento en consultas.

La inserción de los datos se realizó mediante un proceso en php, utilizando una estructura repetitiva y haciendo uso de valores generados de forma aleatoria siguiendo la lista de distritos ya ingresados con anterioridad.Se ejecutaba este código en 10 repeticiones para cada repetición tomo el tiempo de 2 horas con 45 minutos y 28 segundos.El código fuente puede ser observado en el **ANEXO 4** a mayor profundidad.

E) PROCESO DE CREACIÓN DE BASE DE DATOS NOSQL MONGODB:

Este es el punto más importante de toda la investigación ya que es necesario que la base de datos Nosql migre con los mismos nombres que tiene en la base de datos relacional. En la presente investigación se hizo uso de una herramienta de Big Data para hacer el proceso más simple y rápido.

Fue necesario realizar la descarga del software desde https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform/pentaho-data-integration.html y se descomprimió el archivo .zip descargado.

Figura 34.Software Pentaho Data Integration comprimido **Fuente:** Elaboración propia

Luego de haber descomprimido el archivo nos dirigimos a la carpeta descomprimida y se ejecutó el archivo Spoon.bat. Fue necesario crear una nueva transformación, aquí registramos los datos solicitados para la trasformación; indicamos el tipo de base de datos relacional a migrar que en este caso sería Mariadb junto a las credenciales de usuario y numero de puerto.

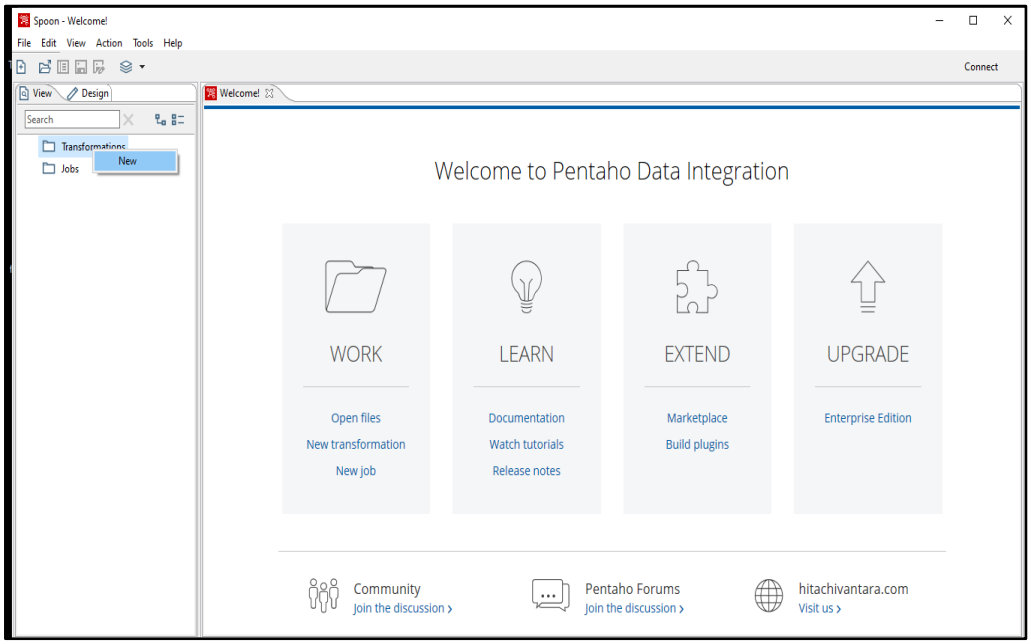


Figura 35.Ventana de bienvenida PDI **Fuente:** Elaboración Propia

Fue necesario contar con el conector JDBC de Mariadb para poder realizar la conexión, caso contrario nos mostrara un error donde indicara que no encuentra el conector, este tiene que estar ubicado en la carpeta lib donde se realizó la extracción del software. El conector puede ser descargado desde la página: https://mariadb.com/kb/en/library/about-mariadb-connector-j/

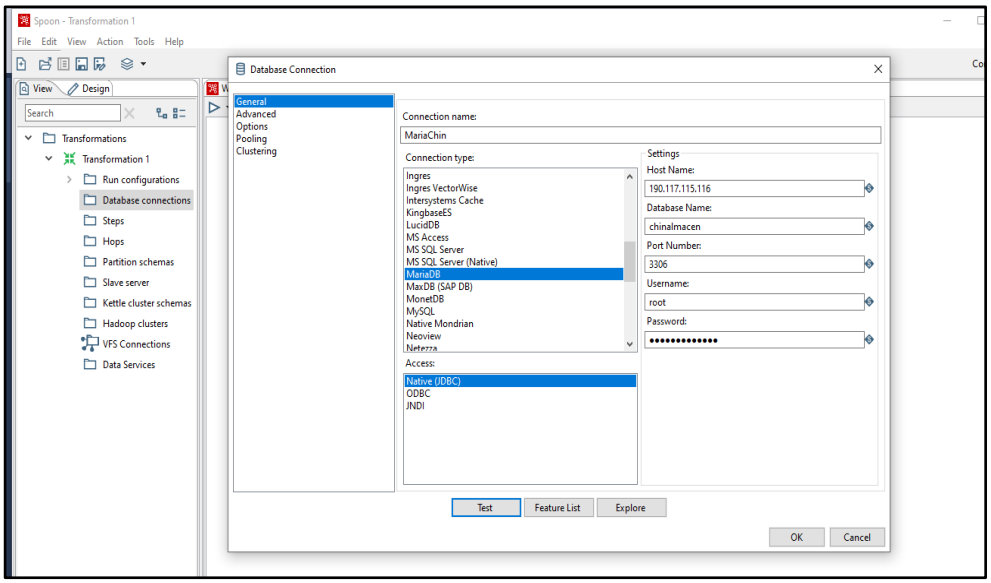


Figura 36.Creacion de nueva Transformación **Fuente:** Elaboración propia

Se eligió en la vista Design >Input >Table Input. Aquí se indicó el nombre de la tabla que se va importar a Mongodb,en el campo Connection se eligió la conexión creada, en el campo SQL se indicó la instrucción SQL, en este caso la instrucción tuvo que ser general ya que se migraron todos los registros sin excepción. Se repitió el mismo proceso para todas las tablas a migrar.

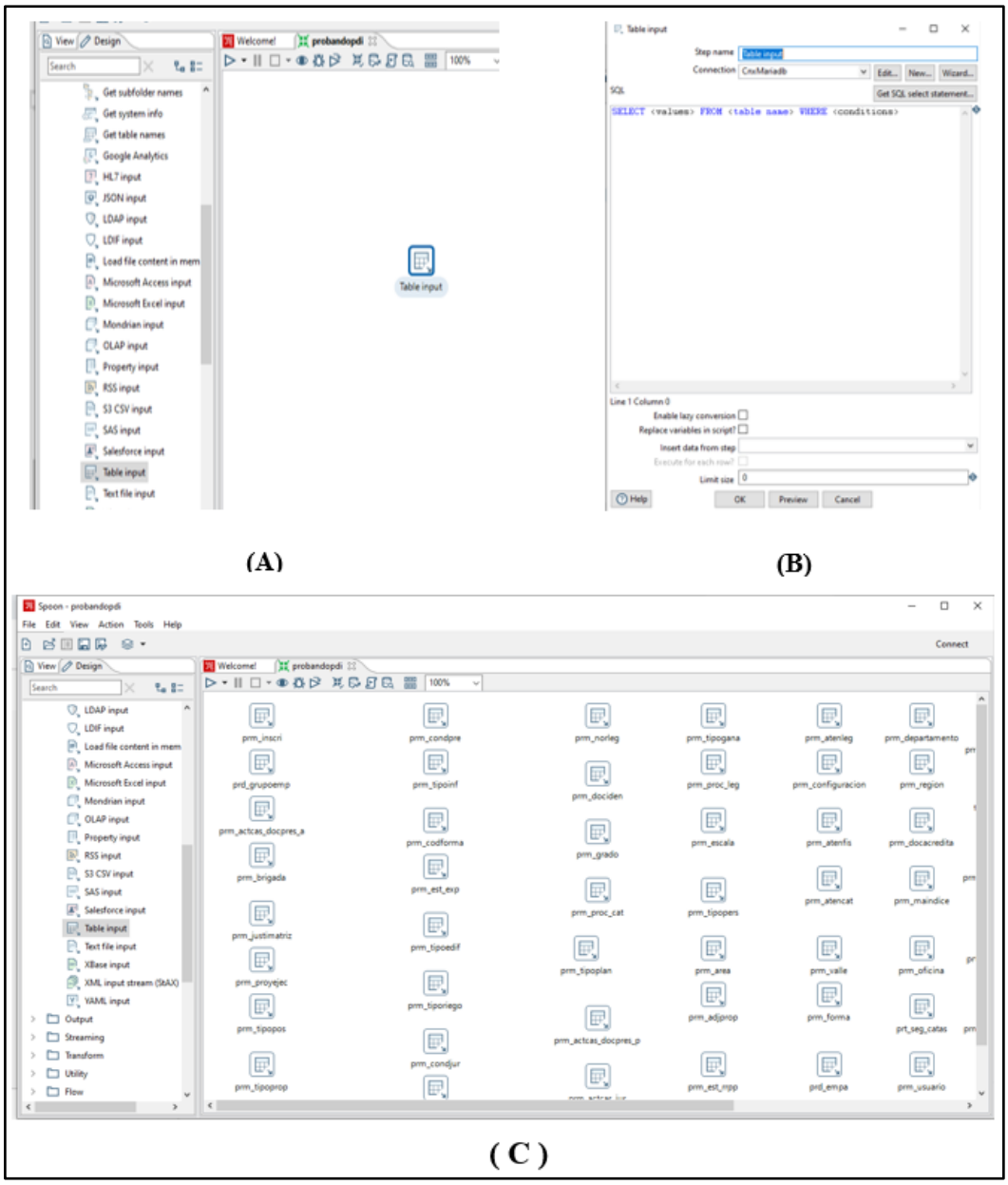


Figura 37.*Proceso de creación de tablas en PDI (A) Selección de Vista Diseño PDI (B) Datos de tabla a migrar (C) Total de tablas terminadas Fuente: Elaboración propia*

Se creó la base de destino de la migración en este caso Mongodb, para el proceso de creación utilizamos Mongodb Compass Community. Se dio clic en CREATE DATABASE; se le indicó el nombre de la base de datos por lo general que tenga el mismo nombre de la base de datos relacional bdtesis, para la creación de la base de datos fue necesario indicarle el nombre de una colección, entonces se tomó el nombre de la primera tabla creada en el PDI.

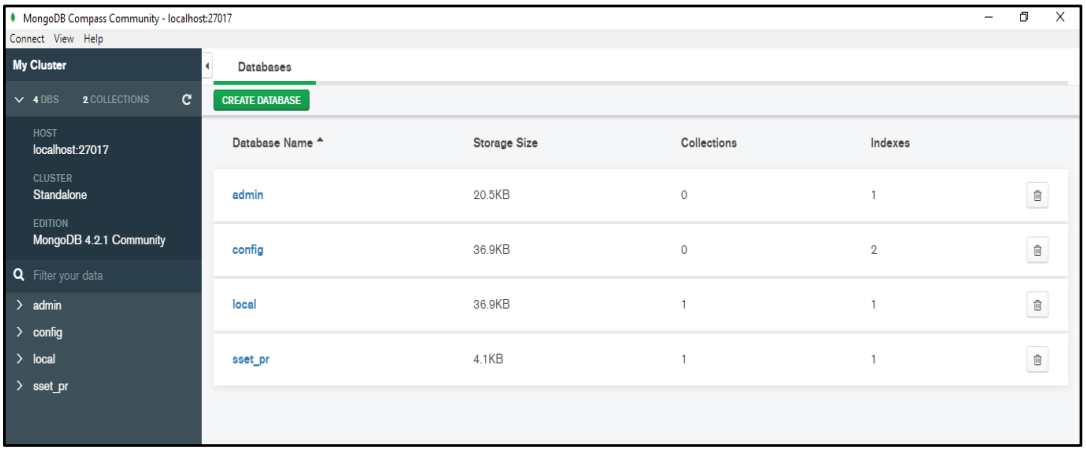


Figura 38. Ventana de bienvenida Mondodb Compass Community Fuente: Elaboración propia

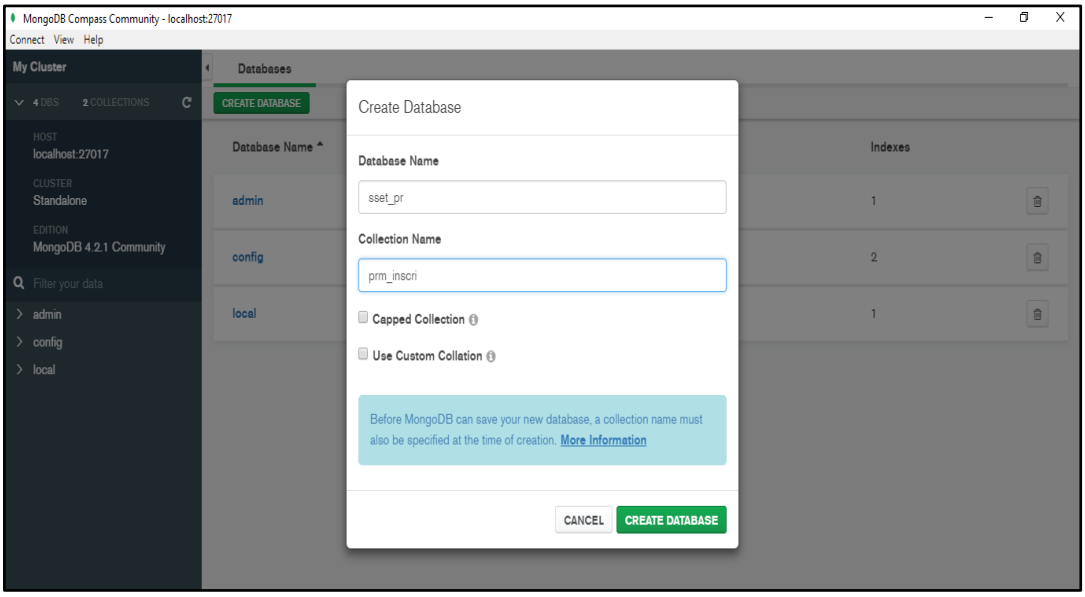


Figura 39. Datos para crear base de datos y colección Mongodb **Fuente:** Elaboración propia

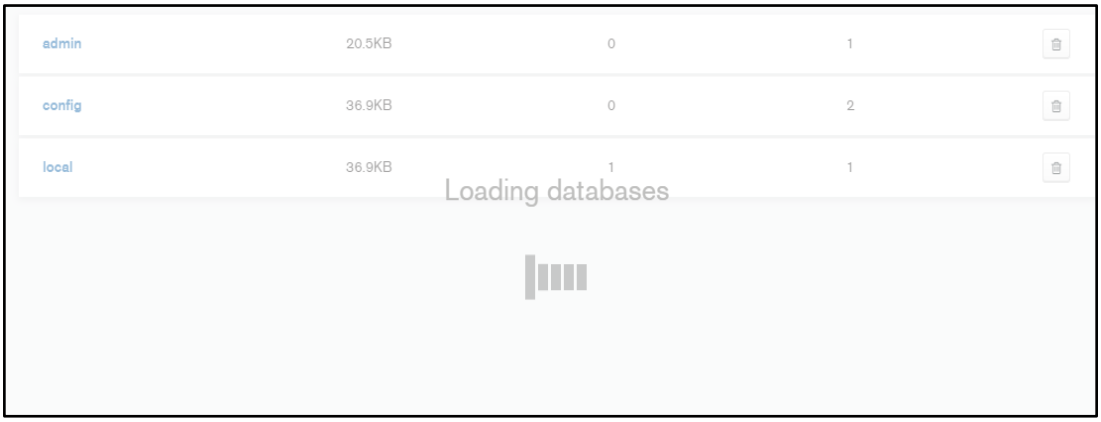


Figura 40.Carga de proceso de creación de base de datos Mongodb **Fuente:** Elaboración propia

Por cada tabla de la base de datos Mariadb que se iba a migrar fue necesario crear una colección nueva en la base de datos Mongodb.

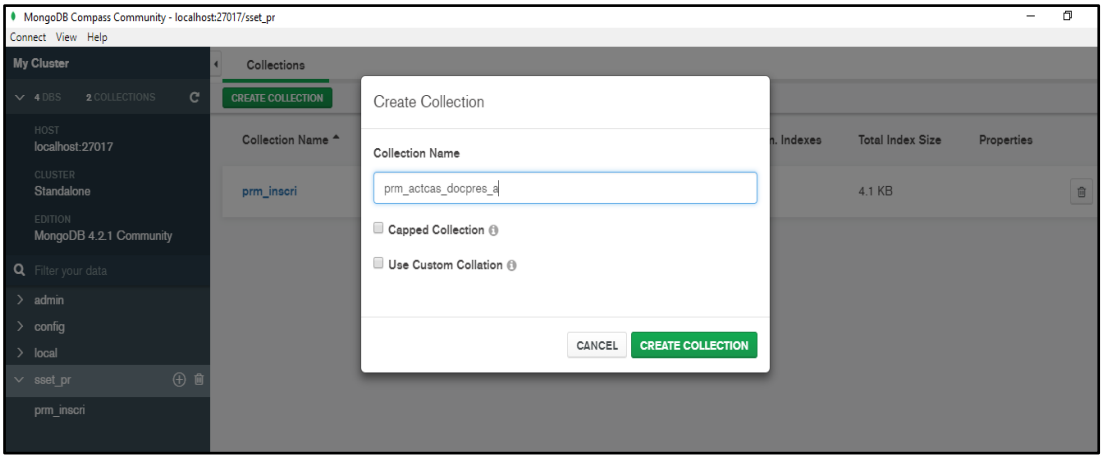


Figura 41.Creacion de colecciones Mongodb **Fuente:** Elaboración propia

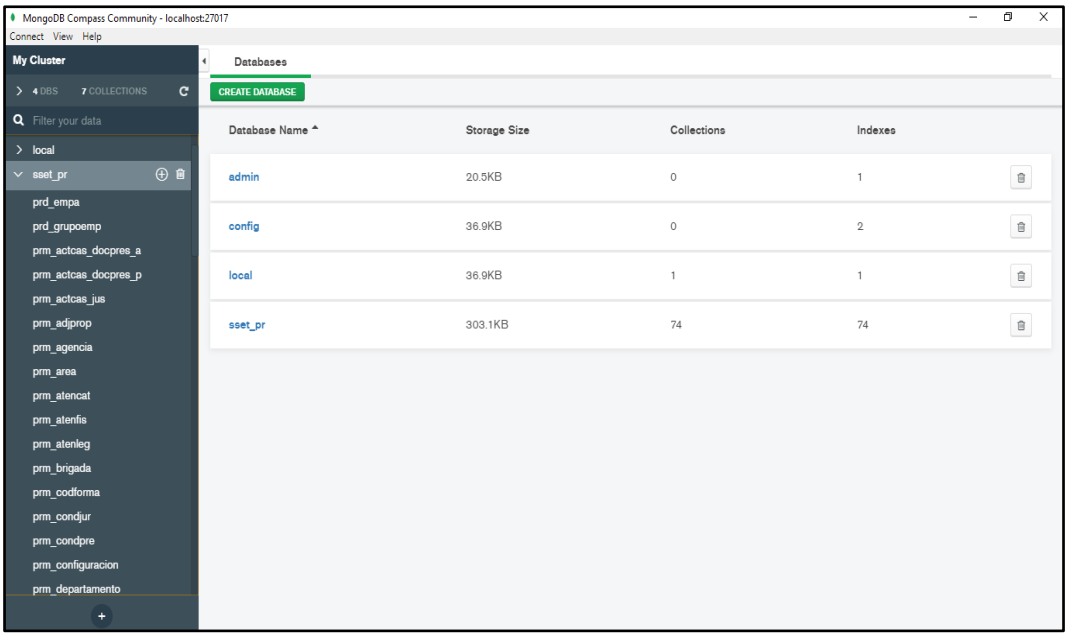


Figura 42.Colecciones creadas para base de datos Mongodb **Fuente:**Elaboración propia

En el software al PDI, se arrastró el elemento Mongodb Output, este lo encontramos en la vista BigData.Aquí se indicaron los datos de conexión y nombre de la colección, para poder llevar un mejor orden se trabajó con el mismo nombre de la tabla relacional agregándole el discriminante _mdb para identificar la estructura de mongodb.

Seleccionamos en el penúltimo icono de Derecha a Izquierda, seguido de ello nos aparecerá una flecha la cual será llevada hasta el componente MongoDB input creado con anterioridad. Para poder obtener los datos de la base de datos, se seleccionó en Get DBs y se eligió en el combo desplegable el nombre de la base de datos y posteriormente la Colección donde serán migrados.

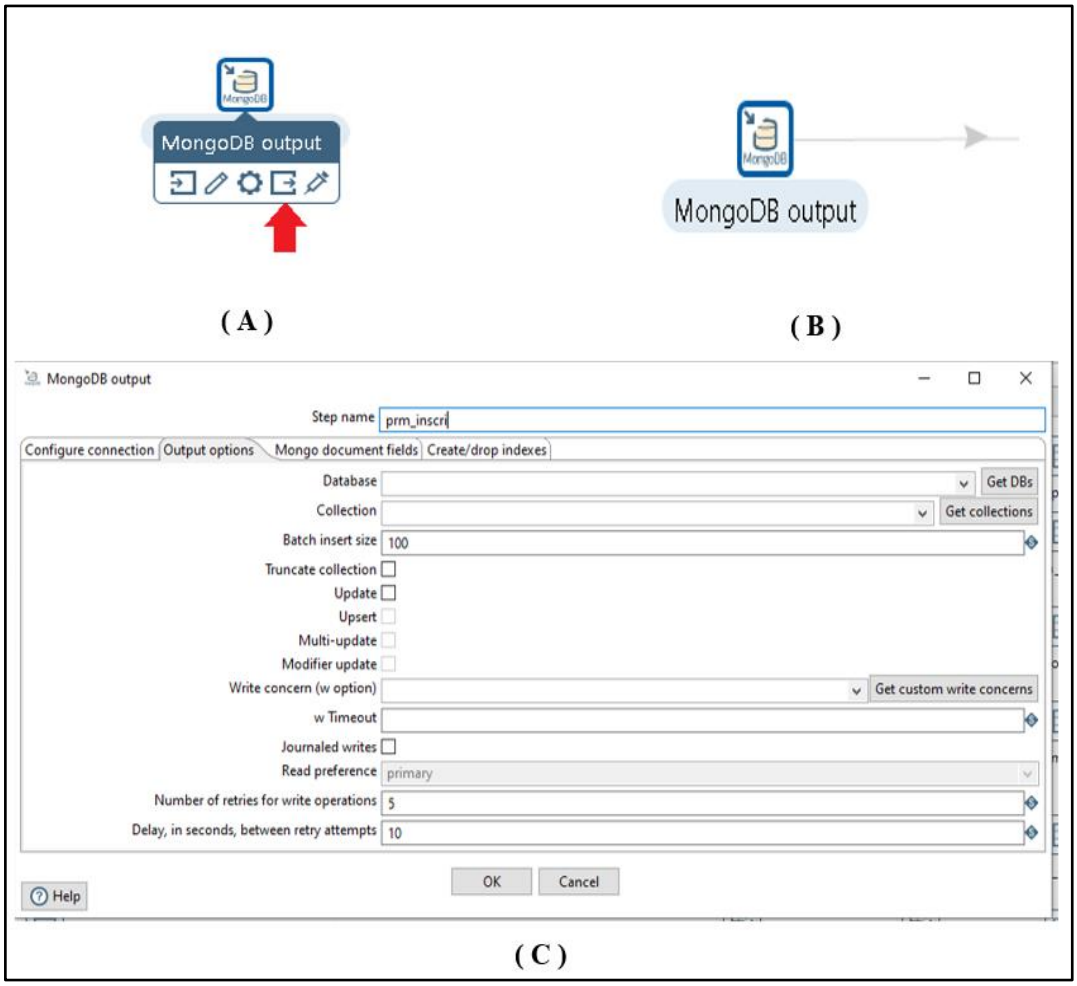


Figura 43.*Proceso de configuración de elemento Mongodb output (A)Elección de opción para relacionar elementos (B)Representación de relación PDI para generar salida (C)Datos de conexión con base de datos relacional y Mongodb* ***Fuente:*** *Elaboración propia*

En el apartado Get Fiels se pudo obtener una vista previa de cómo quedará la estructura del documento que será migrado. Este paso es importante, de lo contrario el documento en Mongodb no podrá obtener la estructura definida de la base de datos relacional. Repetir pasos para todas las demás tablas a migrar definidas en la transformación.

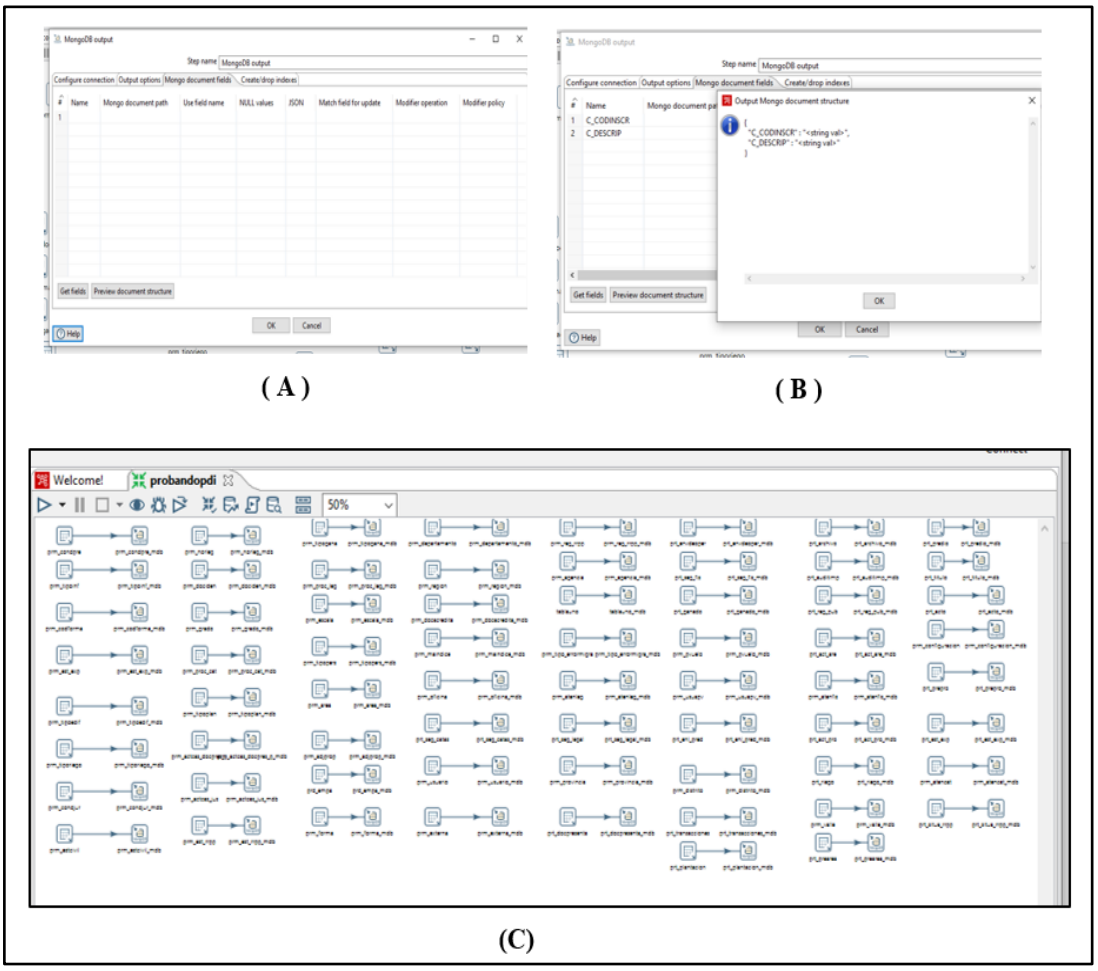


Figura 44.Proceso de vista previa de transformación PDI (A)Selección de opción Mongo documents (B)Preview de campos de tabla relacional transformada a colección (C)Transformación terminada pendiente de ejecución **Fuente:** Elaboración propia

Se procedió a realizar la verificación y validación, previo pasó para la transformación de datos.

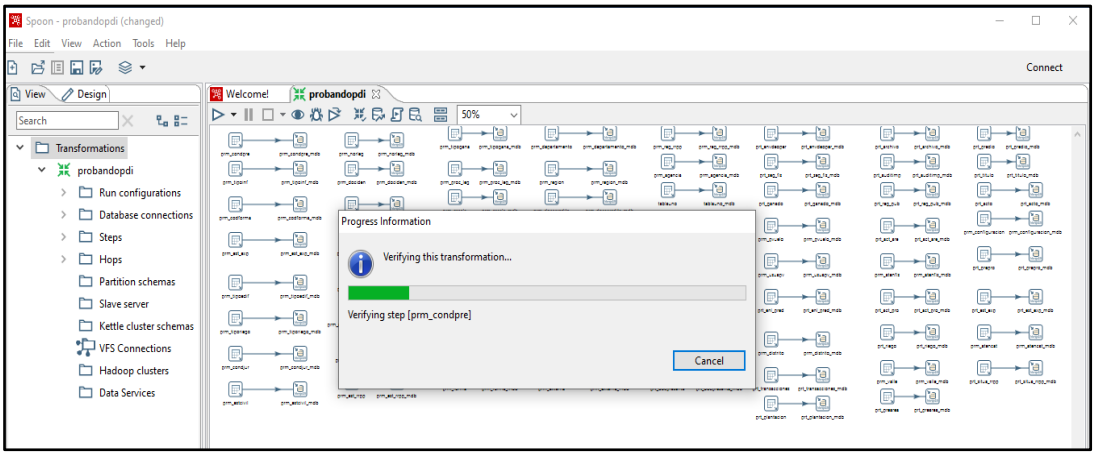


Figura 45.Verificacion de modelo PDI **Fuente:** Elaboración propia

Si tenemos algún problema el PDI nos indicará en que entidades se presentan para corregirlos y seguir con la transformación. Es muy importante prestar atención a los mensajes marcados con rojo.

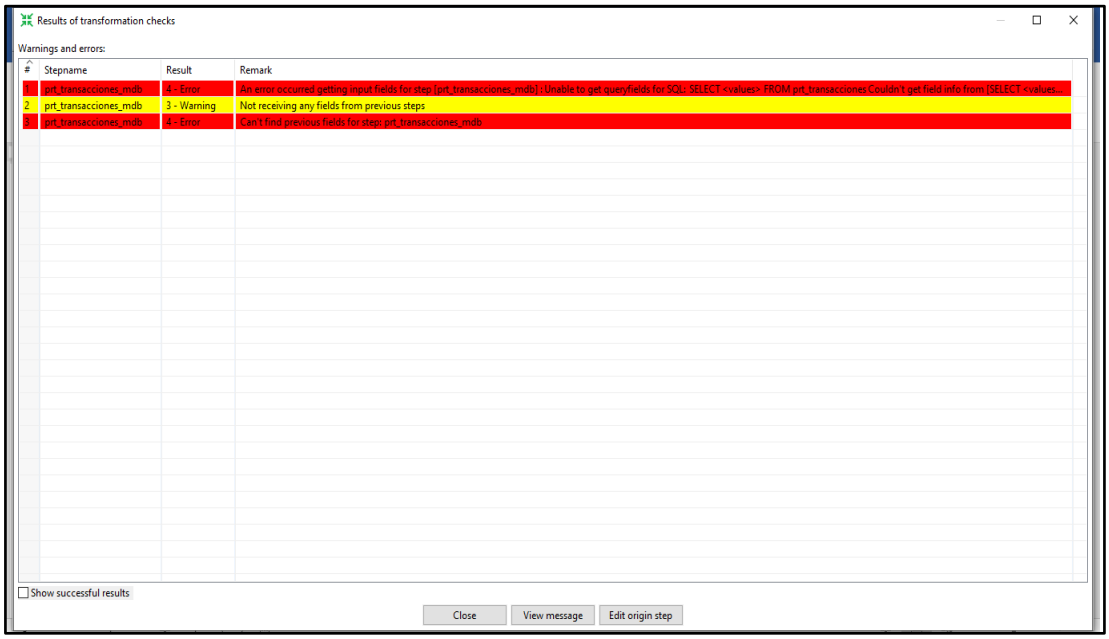


Figura 46.Error de validación PDI **Fuente:** Elaboración propia

Una vez que se realizaron todas las validaciones se procedió a realizar la migración. Se eligió el método Quick Launch y el proceso duro 63 minutos con 39 segundos, el software tiene la característica de marcar con un check las entidades ya importadas y las que estarán procesándose se enmarcaran en guiones y las entidades que presenten errores marcaran un negativo en rojo.

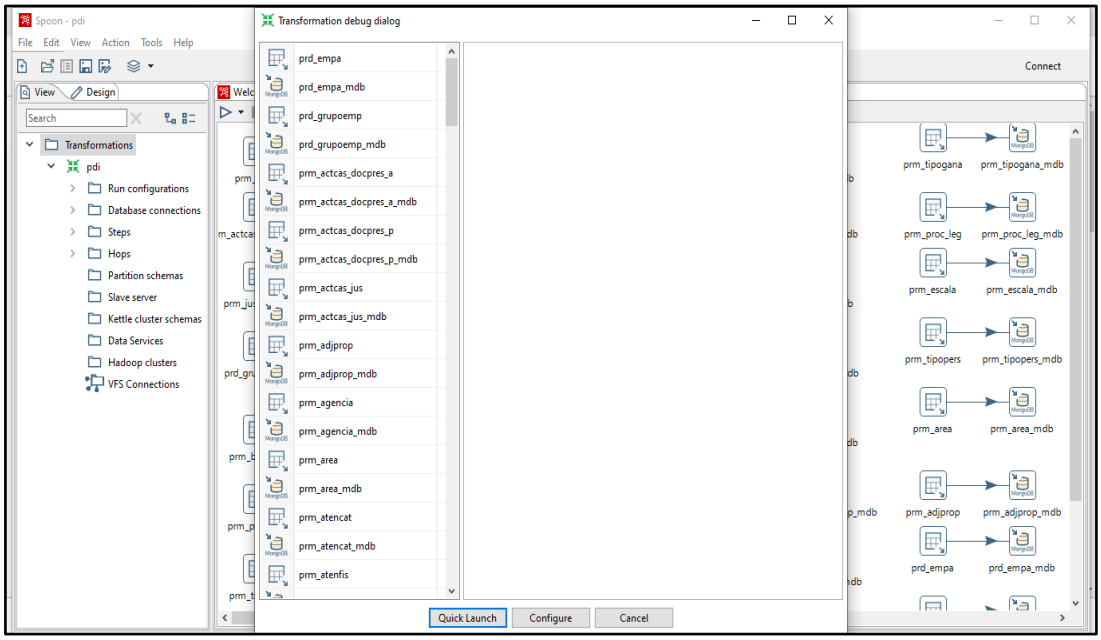


Figura 47.Opciones disponibles de Transformación **Fuente:** Elaboración propia

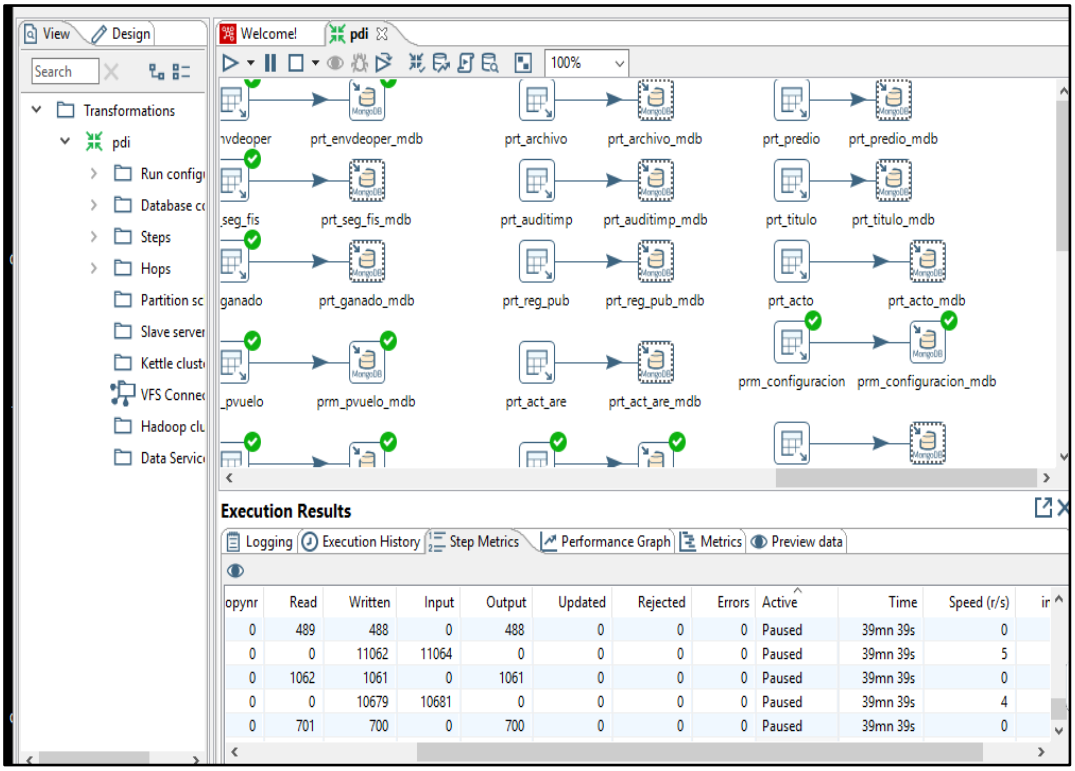


Figura 48.Ventana de verificación de transformación PDI **Fuente:** Elaboración propia

Teniendo la base datos relacional y la Nosql creadas y con los datos insertados; se seleccionó la entidad pueblo que era la que contaba con mayor número de registros para las pruebas de rendimiento

Las pruebas de rendimiento fueron ejecutadas en los softwares Heidi SQL y Mongodb Compass, para la recolección de datos utilizamos la bandeja de salida del software y el monitor de recursos de Windows 10 y Windows Server 2016, la información fue documentada en las fichas de procesos para posteriormente ser analizada.

A) PRUEBA DE RENDIMIENTO PARA CREAR BASES DE DATOS: Para la creación de base de datos relacional se procedió con la ejecución del script estructura.sql en el software Heidi SQL, el cual en pasos anteriores fue creado para evitar su desbordamiento. Fue necesario limpiar la bandeja de salida del software para poder obtener los tiempos de ejecución.

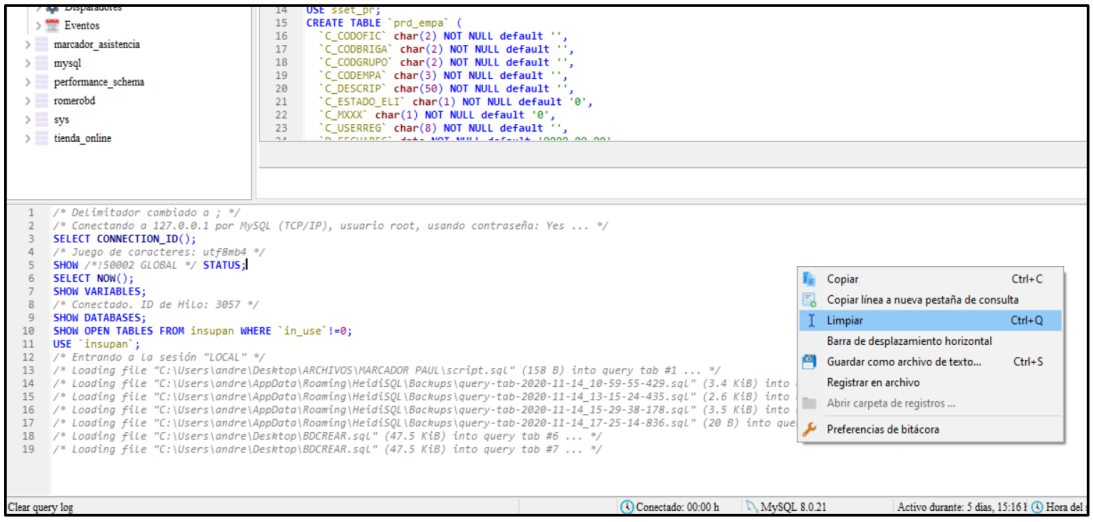


Figura 49.Limpieza de bandeja de salida Fuente: Elaboración propia

Se ejecutó el script y se procedió a recopilar los tiempos mostrados en la bandeja de salida.

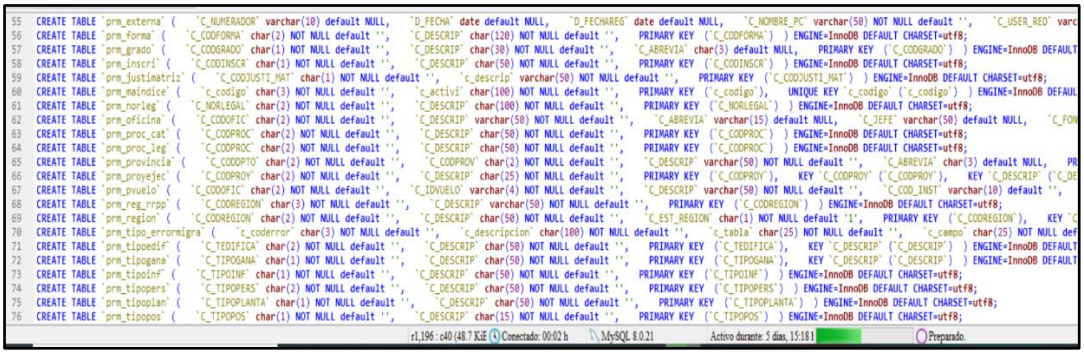


Figura 50.Ejecucion de script SQL para crear base de datos relacional **Fuente:** Elaboración propia

El archivo ejecutado tenía 1196 líneas, conformadas por 83 entidades y se ejecutaron 93 consultas en SQL que permitió la creación de la base de datos relacional.

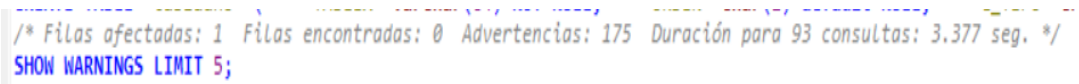


Figura 51.*Tiempo de respuesta ejecución de estructura.sql en equipo 2* ***Fuente:*** *Elaboración propia*

Se repitió el proceso para el equipo 1 y los tiempos recolectados son detallados en la siguiente tabla.

Tabla 18.
Tiempos de Respuesta en creación de base de datos relacional Mariadb ***Fuente:*** *Elaboración propia*

EQUIPO	N° Consultas	Tiempo de Respuesta
EQUIPO 1	93	2036 seg
EQUIPO 2	93	3377 seg

El proceso de creación de la base de datos Nosql como se detalló anteriormente se realizara mediante la migración, igualmente se recopiló la información del tiempo utilizado por el software Pentano Data Integration. Cabe resaltar que el software se encargó de migrar la base de datos relacional Mariadb a Mongodb de forma automática.

Tabla 19.
Tiempos de respuesta en migración de base de datos Mariadb a Mongodb ***Fuente:*** *Elaboración Propia*

EQUIPO	N° Consultas	Tiempo de Respuesta
EQUIPO 1	13185	3819 seg
EQUIPO 2	13185	0 seg

Para poder medir el consumo de recursos de los equipos, el software necesario fue el monitor de recursos que vienen con Windows, tanto para Windows 10 y para Windows Server.

Se utilizó el monitor de recursos por que permitió separar el uso de los recursos por cada proceso seleccionándolo, este también cuenta con ventanas separadas para obtener el uso de cpu, memoria, disco y red.

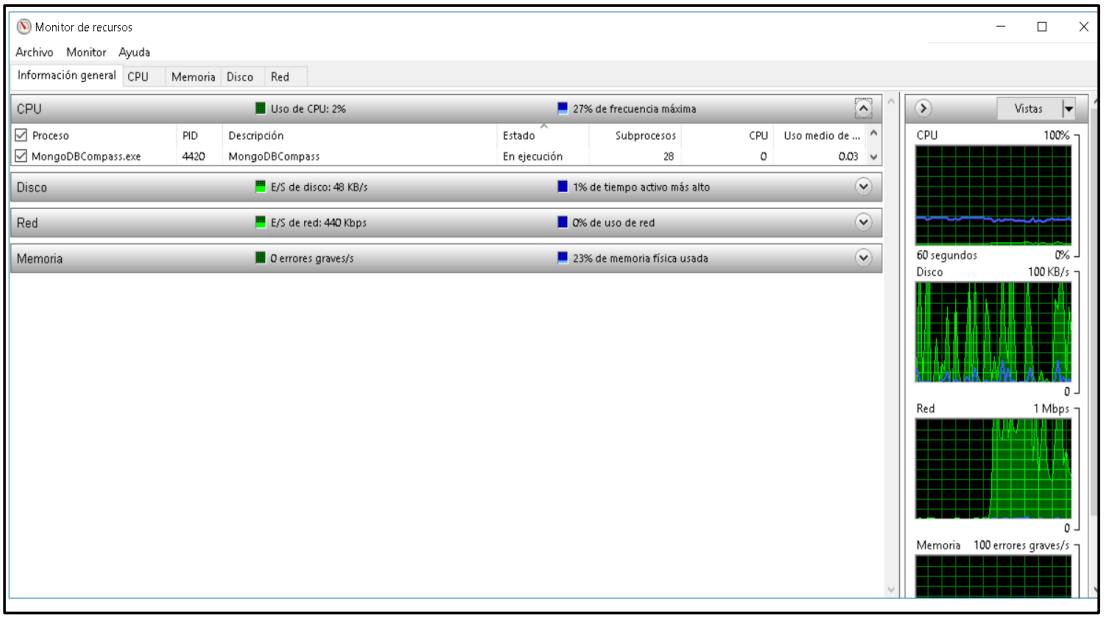


Figura 52.Monitor de recursos de Windows Fuente: Elaboración propia

Para el caso de uso de CPU y de memoria RAM se tomaron los datos iniciales y se aplicó cada formula con los datos obtenidos. Fue necesario identificar y cerrar todos los procesos que estaban trabajando en su momento y los q más recursos consumían para que el proceso de recolección de datos sea más exacta, se detalla a continuación todos los procesos, software consumo de CPU, consumo de memoria RAM y función que desempeñan por equipo.

Tabla 20
Tabla de procesos que se cerraron en Escenario 1 ***Fuente:*** *Elaboración propia.*

Proceso	Software	Cpu	Memoria	Función
AnyDesk.exe	AnyDesk	2%	15.7 MB	Asistencia Remota
svcr.exe	TS plus	1.10%	12.6MB	Acceso Remoto Terminal Services
cbInterface.exe	Cobian Backup	3.50%	10.3MB	Generador de Copias de Seguridad
RvRvpnGui.exe	Radmin VPN	1.45%	9.6MB	Red VPN

Tabla 21
Tabla de procesos que se cerraron en Escenario 2 ***Fuente:*** *Elaboración propia.*

Proceso	Software	Cpu	Memoria	Función
AcroRd32.exe	Adobe Reader	6%	180 MB	Lector de Archivos PDF
MsMpEng.exe	Microsoft	5.85 %	100MB	Malware
TouchpointAnalyticsClientService.exe	Hp	4.50 %	41.38MB	Servicio de análisis de datos de cliente hp
nextcloud.exe	Nextcloud	1.26 %	17.2MB	Sincronizador de nube

1) CONSUMO DE CPU: Inicialmente se tuvo que quitar los subprocesos que ejecutaban y consumían mayor uso de CPU, fue importante identificar qué proceso se finalizó, ya que tuvieron que finalizarse procesos que no guarden relación con Heidisql para no afectar su funcionamiento.

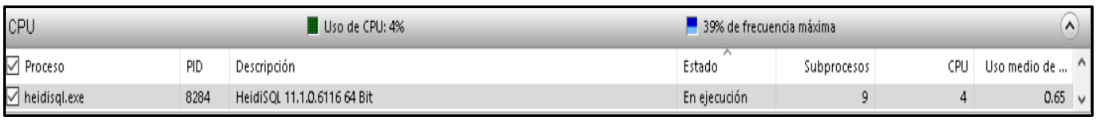


Figura 53*.Uso medio de CPU por segundos inicial Equipo 1* ***Fuente:*** *Elaboración Propia*

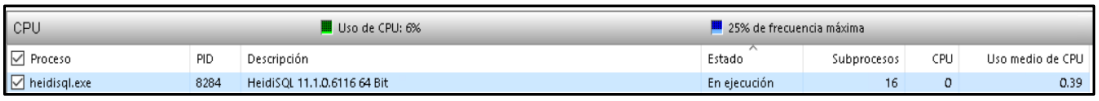


Figura 54.*Uso medio de CPU por segundos final Equipo 1 Fuente: Elaboración Propia*

Se intentó ejecutar el software Pentaho data integration en el equipo 2 para realizar la migración de la base de datos, pero por las exigencias del mismo el sistema se volcó generando un error de sistema o mejor conocido como pantallazo azul. Por lo tanto, solo se trabajó la migración en el equipo 1 y posteriormente fue llevada como un backup al equipo 1.

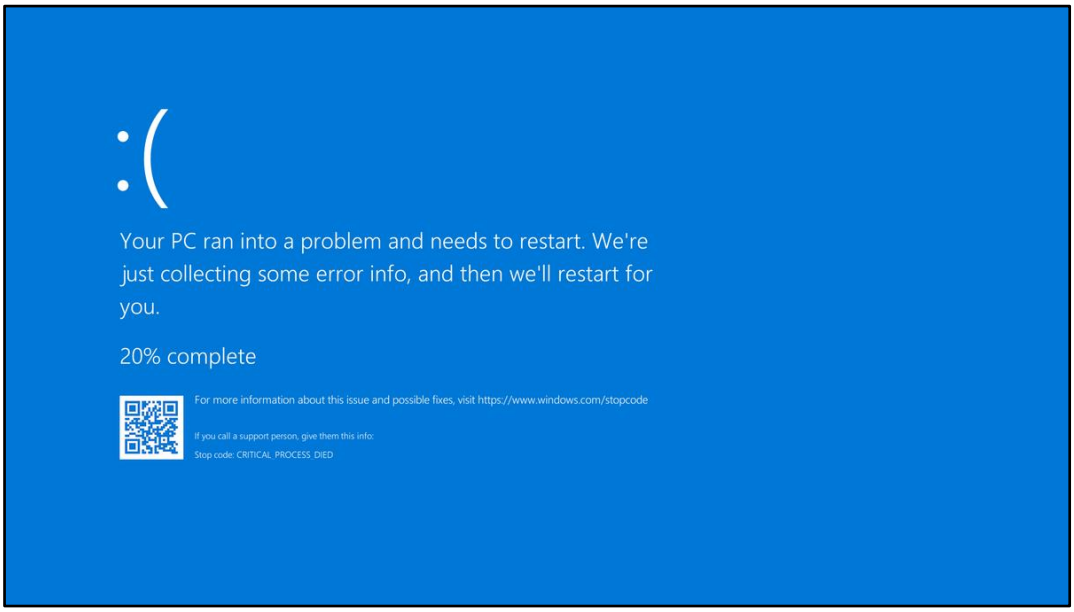


Figura 55.*Error de volcado de sistema operativo Windows 10 equipo 2* ***Fuente:*** *Elaboración propia*

Los tiempos utilizados para el consumo de CPU se detallan en la siguiente tabla donde se recoge el uso promedio inicial de cada proceso utilizado por el software ejecutado.

$$\boldsymbol{UCP = UCPF - UCPI}$$

En donde:

UCPF=Porcentaje de uso de CPU al terminar la ejecución del proceso

UCPI=Porcentaje de uso de CPU inicial al comenzar la ejecución del proceso.

Tabla 22
Tabla de consumo de CPU para creación de base de datos ***Fuente:*** *Elaboración propia*

Equipo	Gestor de BD	UCPI	UCPF	UCP
EQUIPO 1	Mariadb	0.65	0.39	-0.26
	Mongodb	38.00	99.00	61.00
EQUIPO 2	Mariadb	4.60	9.63	5.03
	Mongodb	0.00	0	0.00

2) CONSUMO DE MEMORIA RAM: Los datos también fueron obtenidos mediante el monitor de recursos sección memoria aquí fue necesario obtener la cantidad de espacio de trabajo utilizada por cada proceso, se recopilo la siguiente información.

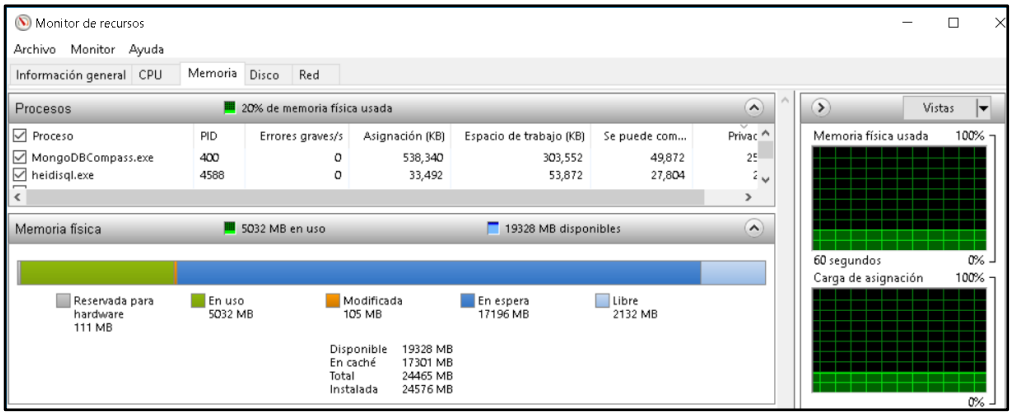


Figura 56.Monitor de recursos sección memoria **Fuente:** Elaboración propia

$$UMR = UMF - UMI$$

En donde:

UMI =El uso de Memoria RAM utilizado al inicio de la ejecución del proceso.

UMF =El uso de Memoria RAM utilizado al término de la ejecución del proceso.

Tabla 23
Uso de memoria RAM para creación de base de datos ***Fuente:*** *Elaboración propia*

Equipo	Gestor de BD	UMF	UMI	UMR
EQUIPO 1	Mariadb	390321KB	53032KB	337289KB
	Mongo db	520361KB	95051KB	425310KB
EQUIPO 2	Mariadb	856695KB	86323KB	770372KB
	Mongo db	0.00KB	0KB	0KB

B) PRUEBA DE RENDIMIENTO PARA CONSULTAS SQL

1) TIEMPO DE RESPUESTA EN CONSULTAS:

Se realizará una consulta de selección general de la tabla prt_situa_rrpp la cual consta de 194572 registros y poder recopilar los tiempos de respuesta.

```
//SELECCIONAR BASE DE DATOS
USE `bdtesis`;
//HACIENDO UN SELECT GENERAL SE ESTARAN CONSULTANDO 194572
REGISTROS
SELECT * FROM `pueblo`;
//PARA LIMITAR CANTIDAD DE REGISTROS
SELECT * FROM `pueblo` limit 100;
SELECT * FROM `pueblo` limit 10000;
SELECT * FROM `pueblo` limit 1000000;
```

Figura 57.Fragmento de script SQL para tiempos de respuesta base de datos Mariadb **Fuente:** Elaboración propia

```
//SELECCIONAR BASE DE DATOS
use `bdtesis`;
//HACIENDO UN SELECT GENERAL SE ESTARAN CONSULTANDO 194572
REGISTROS
db.getCollection("pueblo").find({}).limit();
//PARA LIMITAR CANTIDAD DE REGISTROS
db.getCollection("pueblo").find({}).limit(100);
db.getCollection("pueblo").find({}).limit(10000);
db.getCollection("pueblo").find({}).limit(1000000);
```

Tabla 24
Tiempo de rendimiento en segundos de consultas con delimitantes por equipos y gestores de base de datos ***Fuente:*** *Elaboración propia*

ESCENARIO	GESTOR DE BASE DE DATOS	CANTIDAD DE REGISTROS UTILIZADOS POR CONSULTA		
		100	**10000**	**100000**
Equipo 1	Mongodb	0.015	0.098	1.03
	Mariadb	0.060	0.1035	2.63
Equipo 2	Mongodb	0.025	0.0128	0.121
	Mariadb	0.000	0.0155	0.084

Figura 58.Fragmento de script SQL para tiempos de respuesta base de datos Mongodb **Fuente:** Elaboración propia

2) PRUEBA DE RENDIMIENTO PARA MEDIR TIEMPO DE RESPUESTA EN CONSULTAS CON INNER JOIN Y MAP REDUCE:

Para obtener el tiempo de respuesta utilizaremos un inner join en caso de Mariadb y para Mongodb la función Map Reduce. Utilizaremos dos entidades participantes prt_situa_rrpp, prm_reg_rrpp para esta prueba la cantidad de registros que contiene la tabla es de 409021 y no abra ningún limitante en la cantidad de registros a consultar la única condición será utilizar el valor 3 como código de región, se usa el valor 3 porque hay mayor cantidad de registros con este valor.

```
//SELECCIONAR BASE DE DATOS
USE `sset_pr`;
//EJECUCION DE CONSULTA CON INNER JOIN

SELECT t1.C_IDPREDIO,t1.N_IDACTO,t1.D_FECEST,t2.C_DESCRIP
 FROM prt_situa_rrpp t1 INNER JOIN prm_reg_rrpp t2 ON
t1.C_CODREGION=t2.C_CODREGION WHERE t2.C_CODREGION='003';
```

Figura 59.Fragmento de código SQL para ejecución de select con inner join **Fuente:** Elaboración propia

```
//SELECCIONAR BASE DE DATOS
use `sset_pr`;
//HACIENDO UN SELECT CON INNER JOIN
db.getCollection("prt_situa_rrpp").aggregate(
    [
        {
            "$project" : {
                "_id" : NumberInt(0),
                "t1" : "$$ROOT"
            }
        },
        {
            "$lookup" : {
                "localField" : "t1.C_CODREGION",
                "from" : "prm_reg_rrpp",
                "foreignField" : "C_CODREGION",
                "as" : "t2"
            }
        },
        {
            "$unwind" : {
                "path" : "$t2",
                "preserveNullAndEmptyArrays" : false
            }
        },
        {
            "$match" : {
                "t2.C_CODREGION" : "003"
            }
        },
        {
            "$project" : {
                "t1.C_IDPREDIO" : "$t1.C_IDPREDIO",
                "t1.N_IDACTO" : "$t1.N_IDACTO",
                "t1.D_FECEST" : "$t1.D_FECEST",
                "t2.C_DESCRIP" : "$t2.C_DESCRIP",
                "_id" : NumberInt(0)
            }
        }
    ],
    {
        "allowDiskUse" : true
    }
);
```

Figura 60. *Fragmento de código No SQL para ejecución de select con map reduce* ***Fuente:*** *Elaboración propia*

Tabla 25
Tiempo de respuesta en segundos para consulta con inner join en base de dato relacional y map reduce en Nosql ***Fuente:*** *Elaboración Propia*

Equipo	Gestor de BD	Tiempo de respuesta en segundos
EQUIPO 1	Mariadb	74.390
	Mongo db	75.36
EQUIPO 2	Mariadb	45.30
	Mongo db	48.95

3) PROMEDIO DE USO DE RAM EN CONSULTAS: Fue necesario eliminar temporales y borrar la cache de ambos gestores por cada ejecución que se realizaba. Para estas pruebas el dato obtenido será recopilado del monitor de recursos.

Tabla 26
Promedio de uso de RAM de consultas en bases de datos relacionales y Nosql ***Fuente:*** *Elaboración propia*

ESCENARIO	GESTOR DE BASE DE DATOS	CANTIDAD DE REGISTROS UTILIZADOS POR CONSULTA		
		100	**10000**	**100000**
Equipo 1	Mongodb	5%	30%	60%
	Mariadb	22%	12%	63%
Equipo 2	Mongodb	8%	40%	89%
	Mariadb	24%	56%	96%

4) PROMEDIO DE CPU EN CONSULTAS:

Tabla 27
Promedio de uso de CPU de consultas en bases de datos relacionales y Nosql Fuente: Elaboración propia

ESCENARIO	GESTOR DE BASE DE DATOS	CANTIDAD DE REGISTROS UTILIZADOS POR CONSULTA		
		100	**10000**	**100000**
Equipo 1	Mongodb	10%	24%	12%
	Mariadb	50%	32%	24%
Equipo 2	Mongodb	12.56%	36.30%	71.69%
	Mariadb	16.32%	52.39%	92.36%

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES:

Una buena comparación de rendimiento, es sinónimo de una buena selección de gestores que serán comparados, la mejor elección NoSQL para ser comparado con un gestor relacional son los gestores de tipo documental, debido a que pueden almacenar cualquier dato e incluso estructuras ya definidas. Se eligió a Mariadb en remplazo de MySQL por ser software libre y tener una puntuación de 1264.25 puntos y la colocaba en el segundo puesto del ranking.

El modelo de datos relacional tuvo que ser normalizado hasta la tercera forma normal para que exista una buena optimización de consultas. Se logro una reducción del 40% en el modelo relacional, debido a que la mayoría de consultas eran en 3 tablas principales, las demás al no contar con relaciones validas fueron eliminadas.

Se trabajo con las métricas elegidas luego de pasar un proceso de puntuación donde los resultados fueron basados en investigaciones anteriores a estas, el resultado de dicha evaluación tuvo como ganadores a Consumo de RAM con 4 puntos y Tiempo de Respuesta en Consultas con 6 puntos; las demás métricas tuvieron un total de 3 puntos para Consumo de CPU y 2 de Almacenamiento en Disco.

En las pruebas de tiempo de respuesta para creación de ambas bases Mongodb quedo vencido por Mariadb ya que logro un tiempo de 2036 milisegundos y Mongodb 3819.En las pruebas de consumos de recursos el consumo de RAM tuvo a un Mariadb con un consumo de 337289KB y a MongoDB con un consumo de 425310KB teniendo como vencedor a Mariadb.Para el proceso de inserción y medir el uso de la RAM ,Mongodb logro una muy buena ventaja logrando un 5% sobre 22% en 100 registros en Mariadb,para 10000 y 100000 la ventaja fue para MONGO DB logrando el 30% sobre el 60% logrado por Mariadb.El uso de CPU también tuvo como vencedor a Mongodb logrando un 12% sobre un 24% que uso Mariadb.Se concluye que Mongodb sería la mejor opción para el manejo de información en pocas y grandes cantidades de datos, los tiempos de respuesta y consumo de recursos fueron menores para este sistema gestor de base de datos NoSQL.

4.2. RECOMENDACIONES:

A) Se recomienda utilizar el ranking de Db-Engines ya que las clasificaciones y puestos son actualizados mensualmente y te ofrece un seguimiento de los últimos 3 meses de todos los gestores en el ranking.

B) Seleccionar Mariadb como gestor de base de datos en remplazo de Mysql para futuras investigaciones; este gestor viene escalando en el ranking y apareció como la solución para evitar el pago de licencias comerciales, se sabe que Oracle compro todas las acciones de Mysql y por ende ha dejado de invertir en el desarrollo de esta, enfocándose en soluciones netamente empresariales.

C) Para el proceso de separación de estructura y datos en archivos .sql donde existe gran cantidad de líneas, se recomienda el uso del editor de Texto SublimeText en su versión instalable, evitar utilizar la versión portable para evitar cierres forzosos del software.

D) Utilizar Heidisql debido a la simplicidad que tiene el software para poder gestionar diversas bases de datos y también porque permite extraer información netamente del proceso ejecutado desde la ventana de salida.

E) Escoger equipos que sean catalogados como gama media para alta y que ambos tengan diferencias de memoria y disco duro. Esto con el fin de que los resultados obtenidos por la replicación de esta investigación no varíen.

F) Utilizar Mongodb compass ya que este no necesita licencia comercial para poder utilizarlo y además viene con el instalador de Mongodb Server Community.

G) Utilizar Pentaho Data Integration para realizar la migración de una base de dato relacional a una NoSql, este proceso es menos complicado que el método de desagregación de columnas que también puede realizarse para hacer una migración. Es muy importante seguir a detalle todos los pasos de transformación detallados en el aporte practico.

H) Fueron analizadas las pruebas de rendimiento obtenidas de acuerdo al proceso generado, se recomienda utilizar discos duros sólidos para la ejecución de consultas con gran cantidad de datos y discos duros mecánicos para los procesos de inserción.

REFERENCIAS

A.Gupta, S.Tyagi, N.Panwar, S.Sachdeva, & Saxena, U. (2017). NoSQL databases: Critical analysis and comparison. *International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, (págs. 293-299). Gurgaon, India.

Arias, A. (2016). *Fundamentos de Programación y Bases de Datos.* España: IT Campus Academy.

Ballester, E., Barco, P., Pozo, P., Cuento, A., Guijarro, A., & Boro, E. (2007). *Bases de Datos 1.* Alicante: Escuela Politécnica Superior de Alicante.

Bartholomew, D. (2013). *Getting Started with Mariadb.* Reino Unido: Packt Publishing Ltd.

Bas Abad, V. J. (18 de 09 de 2015). *Riunet.* Obtenido de Repositorio Institucional de la Universitat Politècnica de València: https://riunet.upv.es/bitstream/handle/10251/55530/Memoria.pdf?sequence=1

Benimol. (23 de 05 de 2017). *Exploring the merits of nosql: A study based on mongodb.* Obtenido de Ieeexplore: https://ieeexplore.ieee.org/document/8076778/

Benymol, J., & Sajimon, A. (2017). Exploring the merits of nosql: A study based on mongodb. *International Conference on Networks & Advances in Computational Technologies (NetACT).* Thiruvanthapuram, India.

C.Bender, C. Deco, C., Sanabria, J., Hallo, M., & Gallegos, J. (2014). *Tópicos Avanzados de Bases de Datos.* Proyecto LATIn.

Chávez, S. (2014). Bases de Datos NoSql en Cloud Computing. *XV WORKSHOP DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACION* (págs. 166-170). Parana: Entre Rios.

DB-Engines. (27 de 09 de 2020). *DB-Engines*. Obtenido de DB-Engines Ranking: https://db-engines.com/en/ranking

Fanta. (28 de 06 de 2019). *Introducción a MariaDB.* Obtenido de Fanta Binario: Introducción_a_MariaDB1.pdf

Graterol, Y. (2014). *Mongodb en Español.* Bogota: Yohan Graterol.

Győrödi, C., Győrödi, R., G.Pecherle, & Olah, A. (2016). A Comparative Study: MongoDB vs. MySQL. *IEEE Xplore*.

Kaur, A., & Singh, D. K. (17 de 02 de 2017). *Analysis of NoSQL Databases: A Comparative Study.* Obtenido de EBSCOHOST: http://web.a.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=ec5e346a-a66c-4179-bd80-c3bfc04f4fc9%40sessionmgr4007

Kroenke, D. (2003). *Procesamiento de Base de Datos.* Mexico: PEARSON EDUCACIÓN.

Krstića, L. J., & Krstićb, M. S. (2018). TESTING THE PERFORMANCE OF NoSQL DATABASES VIA THE DATABASE BENCHMARK TOOL. *MILITARY TECHNICAL COURIER , VOL 66*, 614-639.

Leung, F., & Zhou, B. (2016). NoSQL databases: Evaluación del rendimiento de los conjuntos de datos de Twitter en SQL y NoSQL DBMS. *EBSCOHOST*, http://web.a.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=1c3f2870-8f83-4abb-a5e1-bd9e014b9d62%40sessionmgr4009 .

Mariadb. (12 de 09 de 2020). *Documentación Mariadb*. Obtenido de Mariadb: https://mariadb.com/kb/en/storage-engines/

Mongodb. (14 de 06 de 2019). *Mongodb Documentation.* Obtenido de Mongodb manual: https://docs.mongodb.com/manual/

Moreno, J., Quintero, E., & Rueda, R. (2016). UNA COMPARACIÓN DE RENDIMIENTO ENTRE ORACLE Y MONGODB. *CIENCIA E INGENIERÍA NEOGRANADINA*.

Ordoñez, Z., Tapia, J., & Rivas, W. (2015). *Fundamentos de Base de datos.* Machala: UTMACH.

R.Rai, & Chettri, P. (2017). Advances in Computers, Volume 109. En R.Rai, & P. Chettri, *NoSQL Hands On* (págs. 197-198). India: Elsevier Inc.

Rubio, M., Gomez, S., Letón, E., Rodrigo, S., & Chaos, D. (2017). *INTRODUCCIÓN A LA INFORMÁTICA BÁSICA.* Madrid: Universidad Nacional de Educación a Distancia.

Russell, J. (2015). *Learning MySQL and MariaDB.* Cambridge: O'reilly.

Sanchez, J. (20 de Febrero de 2009). *Sistemas Gestores de Base de Datos.* España: Creative Commons. Obtenido de Apuntes y Materiales de Gestión de Bases de Datos: https://jorgesanchez.net/manuales/abd/bases-sgbd.html

Sarasa, A. (2016). *Introducción a las Bases de datos NoSQL usando MongoDB.* Barcelona: Oberta UOC Publishing, SL.

Silberschatz, A., Korth, F., & Sudarshan, S. (2014). *FUNDAMENTOS DE BASES DE DATOS.* Madrid: McGraw-Hill.

ANEXOS

ANEXO 1-RESOLUCION DE APROBACION DE PROYECTO DE INVESTIGACION:

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°0018-2022/FIAU-USS

Pimentel, 3 de febrero de 2022

VISTO:
El Acta de reunión N°3101 - 2022 del Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS remitida mediante Oficio N°0004-2022/FIAU-IS-USS de fecha 2 de febrero de 2022, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, mediante documentos de vistos, el Comité de investigación de la referida Escuela profesional acordó aprobar la ampliación de la vigencia de las tesis que se detallan en el Acta de reunión N°3101 - 2022, de la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y AMBIENTE, a cargo de egresados del Programa de estudios INGENIERÍA DE SISTEMAS, hasta el 31 de julio de 2022.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO ÚNICO: AMPLIAR VIGENCIA, de la Tesis a cargo de los egresados del Programa de estudios de **INGENIERÍA DE SISTEMAS** que se detallan en el anexo de la presente Resolución, hasta el 31 de julio de 2022.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Mg. Víctor Alexei Tuesta Montoya
Decano (e) / Facultad De Ingeniería, Arquitectura Y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

MBA. María Noelia Sialer Rivera
Secretaria Académica / Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Cc: Interesado, Archivo

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°0018-2022/FIAU-USS

Pimentel, 3 de febrero de 2022

ANEXO

N°	APELLIDOS Y NOMBRES	TEMA DE TESIS	RESOLUCIÓN DE APROBACIÓN/MODIFICACIÓN/AMPLIACIÓN TEMA DE TESIS	FECHA RESOLUCIÓN DE APROBACIÓN TEMA DE TESIS
1	CAMPOVERDE VEGA VICTOR ANDREE	ANÁLISIS COMPARATIVO DE RENDIMIENTO EN GESTORES DE BASES DE DATOS RELACIONALES Y NO RELACIONALES	2325-2020/FIAU-USS	17 de Noviembre de 2020 (*)
2	CLAVO TAFUR CRISTIAN JESUS	ANÁLISIS COMPARATIVO DE TÉCNICAS DE MITIGACIÓN DE ATAQUE DE DDOS EN CLOUD COMPUTING	2325-2020/FIAU-USS	17 de Noviembre de 2020 (*)
3	DIAZ CASTAÑEDA RUDY SLAYTONW	MODELO DE PROCESOS PARA EL DESARROLLO DE SOFTWARE CON CARACTERÍSTICAS DE SEGURIDAD PARA VULNERABILIDADES MÁS RECURRENTES	0933-2019/FIAU-USS	22 de julio de 2019
4	VENEGAS BRAVO JOSE ALEXANDER	ANÁLISIS COMPARATIVO DE RENDIMIENTO DE GESTORES DE BASE DE DATOS NOSQL DOCUMENTALES	2325-2020/FIAU-USS	17 de noviembre de 2020 (*)

(*) La Resolución en referencia ampliaba la vigencia hasta el 31 de julio 2021

Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

ANEXO 2-FICHA DE PROCESO PARA RECOLECCION DE DATOS:

FICHA DE PROCESOS

N:

PROYECTO: _____ **ENCARGADO:** _____
_____ **FECHA:** _____
_____ **INSTITUCION:** _____

TIPO DE OPERACIÓN: ( )CREAR
( )ACTUALIZAR
( )ELIMINAR
( )CONSULTAR
( )OTRO: _____

FECHA EJECUCION: _____ **HORA INICIO:** _____
N° ENTIDADES: _____ **HORA FIN:** _____

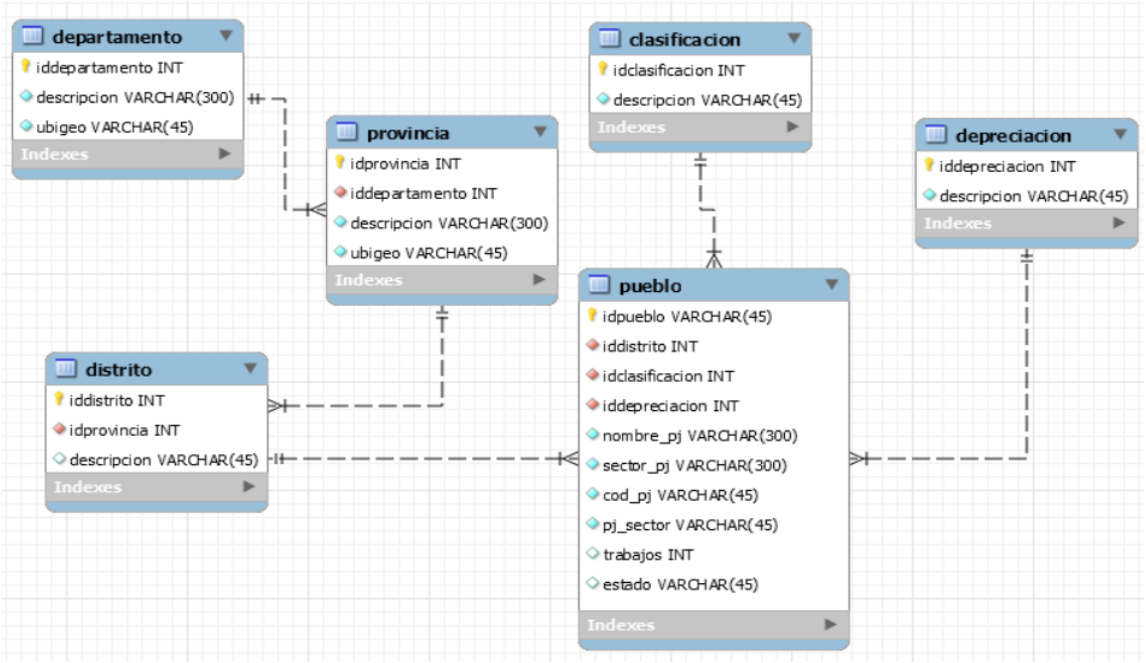
N° REGISTROS: _____
SISTEMA GESTOR DE BASE DE DATOS: _____

SOFTWARE UTILIZADO: _____

DESCRIPCION DEL PROCESO: _____
_____
_____
_____
_____
_____
_____
_____

OBSERVACIONES:
1.LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2.ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3.EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4.ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

ANEXO 3- MODELO RELACIONAL BASE DE DATOS:



ANEXO 4:

CODIGO ARCHIVO Conexión.php

```
<?php

$driver="mysql";

$host  ="localhost";
$puerto="3306";
$dbname="bdtesis";
$usuario="root";
$clave  ="sistemas";


$chs="utf8";
$cadena="$driver:host=$host;port=$puerto; dbname=$dbname; charset=$chs";
$con = new PDO($cadena, $usuario, $clave);

?>
```

CODIGO ARCHIVO Repoblar.php

```
<?php

for ($i = 1; $i <= 100000; $i++) {
        $distrito=mt_rand(1,1874);
        $clasificacion=mt_rand(1,5);
        $depreciacion=mt_rand(1,5);
        $nombrepj=substr(md5(time()), 0, 16);
        $sectorpj=substr(md5(time()), 0, 16);
        $codpj=substr(md5(time()), 0, 16);
        $pjsec=substr(md5(time()), 0, 16);
        $trabajos=mt_rand(1,10);

        try {
         $sql = "insert pueblo()values
         (:iddistrito,:idclasificacion,:iddepreciacion,:nombre_pj,:sector_pj,:cod_pj,:pj_sector,:tr
abajos)";
         $stmt = $con->prepare($sql);
         $stmt->bindParam(':iddistrito', $distrito, PDO::PARAM_INT);
         $stmt->bindParam(':idclasificacion', $clasificacion,PDO::PARAM_INT);
         $stmt->bindParam(':iddepreciacion', $depreciacion,PDO::PARAM_INT);
         $stmt->bindParam(':nombre_pj',  $nombrepj, PDO::PARAM_STR);
         $stmt->bindParam(':sector_pj', $sectorpj, PDO::PARAM_STR);
```

```
        $stmt->bindParam(':cod_pj',$codpj, PDO::PARAM_STR);
        $stmt->bindParam(':pj_sector',$pjsec, PDO::PARAM_STR);
        $stmt->bindParam(':trabajos',$idserco, PDO::PARAM_INT);
        $stmt->execute();
        $stmt->closeCursor();

       }
       catch (PDOException $pe) {
           die("Error occurred:" . $pe->getMessage());
       }

}
echo 'DATOS INGRESADOS TERMINADO';

?>
```

ANEXO 5

INFORME DE BASE DE DATOS:

La base de datos es una base de datos predial de todos los pueblos que ha venido siendo publicada desde el año 2017 en el portal de datos abiertos del gobierno en Perú. El encargado del tratamiento de esta información es el ORGANISMO DE FORMALIZACION DE LA PROPIEDAD INFORMAL o mayormente conocido como COFOPRI.

Esta información se comenzó a trabajar en Junio de 1997 cuando el responsable era el Ministerio de Agricultura del Perú; para el 12 de junio del 2007 el ministerio de Agricultura ceda y entregue la información a Cofopri para que ellos continúen con el trabajo.

En Septiembre del 2018 el ministerio de Agricultura solicita la data nuevamente y asigna que el organismo supervisor de la información sea el Ministerio de Vivienda, Construcción y Saneamiento y que ellos sean los que supervisen que la data que es formalizada a la actualidad sea correcta.

A continuación se detalla aspectos generales de la base datos.

Lenguaje	Español
Licencia	Open Data Commons Attribution License
Autor	ORGANISMO DE FORMALIZACION DE LA PROPIEDAD INFORMAL
Tipo de Datos	Los campos contienen datos numéricos y alfanuméricos
Persona de Contacto	Daniel Lizarraga López Cesar Cisneros Vargas
Email de Contacto	dlizarraga@cofopri.gob.pe ccisneros@cofopri.gob.pe
Repositorio	Plataforma Nacional de Datos Abiertos
Url de Repositorio	https://www.datosabiertos.gob.pe/

La base de datos fue publicada de acuerdo a la fecha de formalización de pueblos por región y departamento.

1. **Publicación de Pueblos Formalizados al 08 de Junio del 2017**

Tamaño del archivo	670.25 KB / 13589 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2017_06_08.xlsx
Fecha de publicación	12/06/2017
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-08-de-junio-del-2017/resource/c48318a7-5e78-4db9-b0d7-d08e1fd59c80

2. **Publicación de Pueblos Formalizados al 10 de Julio del 2017**

Tamaño del archivo	631.69 KB / 13613 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2017_07_10.xlsx
Fecha de publicación	19/07/2017
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-10-de-julio-del-2017/resource/38597421-fa43-4d45-a818-d1a41ad80d9f

3. **Publicación de Pueblos Formalizados al 01 de Setiembre del 2017**

Tamaño del archivo	2.03 MB/ 11785 registros
Extensión de archivo	XLS
Nombre del archivo	Pueblos Formalizados 2017_09_01.xls
Fecha de publicación	22/09/2017
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-01-de-setiembre-del-2017/resource/5089cdbc-3e35-44d8-a343

4. **Publicación de Pueblos Formalizados al 21 de Noviembre del 2017**

Tamaño del archivo	663.23 KB/ 13768 registros
Extensión de archivo	XLS
Nombre del archivo	Pueblos Formalizados 2017_11_21.xlsx
Fecha de publicación	22/11/2017
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-21-de-noviembre-del-2017/resource/83c48327-f59b-49b6-bfe3

5. **Publicación de Pueblos Formalizados al 28 de Diciembre del 2017**

Tamaño del archivo	664.15 KB/ 13768 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2017_12_28.xlsx
Fecha de publicación	27/04/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-28-de-diciembre-del-2017/resource/0271f068-4cc1-449d-9849

6. **Publicación de Pueblos Formalizados al 25 de Enero del 2018**

Tamaño del archivo	712.6 KB/ 13860 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2017_01_25.xlsx
Fecha de publicación	18/05/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-25-de-enero-del-2018/resource/31ed0454-0e87-4c18-b2d2-0548f6747cdb

7. **Publicación de Pueblos Formalizados al 09 de Marzo del 2018**

Tamaño del archivo	671.57 KB/ 13860 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2018_03_09.xlsx
Fecha de publicación	28/05/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-09-de-marzo-del-2018/resource/9f6eb143-e34a-4488-b47c-f16d92be6401

8. **Publicación de Pueblos Formalizados al 01 de Abril del 2018**

Tamaño del archivo	714 KB/ 13891 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2018_04_01.xlsx
Fecha de publicación	28/05/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-01-de-abril-del-2018/resource/7cc057f9-b370-4ca6-9db4-f79360e32a5e

9. **Publicación de Pueblos Formalizados al 01 de Mayo del 2018**

Tamaño del archivo	713.43 KB/ 13909 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2018_05_01.xlsx
Fecha de publicación	21/05/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-01-de-mayo-del-2018/resource/983f16be-df82-4d06-8037-af657af8a03e

10. Publicación de Pueblos Formalizados al 6 de Junio del 2018

Tamaño del archivo	725.51 KB/ 13969 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2018_06_06.xlsx
Fecha de publicación	03/08/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-16-de-julio-del-2018/resource/0a753192-aaa2-47a5-8fbe-488811cbfc4f

11. Publicación de Pueblos Formalizados al 16 de Julio del 2018

Tamaño del archivo	725.51 KB/ 13969 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2018_06_06.xlsx
Fecha de publicación	03/08/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-16-de-julio-del-2018/resource/0a753192-aaa2-47a5-8fbe-488811cbfc4f

12. Publicación de Pueblos Formalizados al 03 de Agosto del 2018

Tamaño del archivo	542.44 KB/ 14004 registros
Extensión de archivo	ZIP
Nombre del archivo	Pueblos_Formalizados_3Ago_2018.zip
Fecha de publicación	03/08/2018
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-03-de-agosto-del-2018/resource/ef86f857-8e3d-4d07-b8fb-a0b03144a4fa

13. Publicación de Pueblos Formalizados al 30 de Octubre del 2018

Tamaño del archivo	546.39 KB/ 13349 registros
Extensión de archivo	ZIP
Nombre del archivo	Pueblos Formalizados 2017.10.30.zip
Fecha de publicación	24/01/2019
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-30-de-octubre-del-2018/resource/76d00823-a960-4bfa-ac93-86dc07492488

14. Publicación de Pueblos Formalizados al 10 de Enero del 2019

Tamaño del archivo	664.7 KB/ 14182 registros
Extensión de archivo	ZIP
Nombre del archivo	Pueblos Formalizados 2019.01.10.zip
Fecha de publicación	24/01/2019
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-10-de-enero-del-2019/resource/66c72f7c-9ccd-41ef-b0a0-f2a112fba737

15. Publicación de Pueblos Formalizados al 06 de marzo del 2019

Tamaño del archivo	589.48 KB/ 14196 registros
Extensión de archivo	ZIP
Nombre del archivo	Pueblos Formalizado 2019.03.06.zip
Fecha de publicación	22/04/2019
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-06-de-marzo-del-2019/resource/73d81574-6a36-4466-a78c-75c83b1fba01

16. Publicación de Pueblos Formalizados al 22 de mayo del 2019

Tamaño del archivo	827.2 KB/ 14224 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos Formalizados 2019.05.22.xlsx
Fecha de publicación	22/05/2019
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-22-de-mayo-del-2019/resource/97065739-9e28-4678-8e67-51aea474c62f

17. Publicación de Pueblos Formalizados al 10 de febrero del 2020

Tamaño del archivo	881.09 KB/ 14291 registros
Extensión de archivo	XLSX
Nombre del archivo	Pueblos_Formalizados_10Feb_2020.xlsx
Fecha de publicación	18/02/2020
URL	https://www.datosabiertos.gob.pe/dataset/pueblos-formalizados-al-10-de-febrero-del-2020/resource/3c135a35-bf6a-4d13-84b8-ac555d326de9

Resumen de Registros por Publicación:

N°	PUBLICACION	N° DE REGISTROS
1	Publicación de Pueblos Formalizados al 08 de Junio del 2017	13589
2	Publicación de Pueblos Formalizados al 10 de Julio del 2017	13613
3	Publicación de Pueblos Formalizados al 01 de Setiembre del 2017	11785
4	Publicación de Pueblos Formalizados al 21 de Noviembre del 2017	13768
5	Publicación de Pueblos Formalizados al 28 de Diciembre del 2017	13768
6	Publicación de Pueblos Formalizados al 25 de Enero del 2018	13859
7	Publicación de Pueblos Formalizados al 09 de Marzo del 2018	13860

8	Publicación de Pueblos Formalizados al 01 de Abril del 2018	13891
9	Publicación de Pueblos Formalizados al 01 de Mayo del 2018	13909
10	Publicación de Pueblos Formalizados al 6 de Junio del 2018	13969
11	Publicación de Pueblos Formalizados al 16 de Julio del 2018	13969
12	Publicación de Pueblos Formalizados al 03 de Agosto del 2018	14004
13	Publicación de Pueblos Formalizados al 30 de Octubre del 2018	13349
14	Publicación de Pueblos Formalizados al 10 de Enero del 2019	14182
15	Publicación de Pueblos Formalizados al 06 de marzo del 2019	14196
16	Publicación de Pueblos Formalizados al 22 de mayo del 2019	14224
17	Pueblos Formalizados al 10 de febrero del 2020	14291
	TOTAL	**234226**

ANEXO 6:

CREACION DE BASE DE DATOS RELACIONAL:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,N
O_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema bdtesis
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema bdtesis
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bdtesis` DEFAULT CHARACTER SET utf8 ;
USE `bdtesis` ;

-- -----------------------------------------------------
-- Table `bdtesis`.`departamento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`departamento` (
  `iddepartamento` INT NOT NULL AUTO_INCREMENT,
  `descripcion` VARCHAR(300) NOT NULL,
  `ubigeo` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`iddepartamento`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bdtesis`.`provincia`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`provincia` (
  `idprovincia` INT NOT NULL AUTO_INCREMENT,
  `iddepartamento` INT NOT NULL,
  `descripcion` VARCHAR(300) NOT NULL,
  `ubigeo` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idprovincia`),
  INDEX `fk_provincia_departamento_idx` (`iddepartamento` ASC) VISIBLE,
  CONSTRAINT `fk_provincia_departamento`
    FOREIGN KEY (`iddepartamento`)
    REFERENCES `bdtesis`.`departamento` (`iddepartamento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bdtesis`.`distrito`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`distrito` (
  `iddistrito` INT NOT NULL AUTO_INCREMENT,
  `idprovincia` INT NOT NULL,
  `descripcion` VARCHAR(45) NULL,
```

```
  PRIMARY KEY (`iddistrito`),
  INDEX `fk_distrito_provincia1_idx` (`idprovincia` ASC) VISIBLE,
  CONSTRAINT `fk_distrito_provincia1`
    FOREIGN KEY (`idprovincia`)
    REFERENCES `bdtesis`.`provincia` (`idprovincia`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bdtesis`.`clasificacion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`clasificacion` (
  `idclasificacion` INT NULL AUTO_INCREMENT,
  `descripcion` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idclasificacion`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bdtesis`.`depreciacion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`depreciacion` (
  `iddepreciacion` INT NOT NULL AUTO_INCREMENT,
  `descripcion` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`iddepreciacion`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bdtesis`.`pueblo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdtesis`.`pueblo` (
  `idpueblo` INT NULL AUTO_INCREMENT,
  `iddistrito` INT NOT NULL,
  `idclasificacion` INT NOT NULL,
  `iddepreciacion` INT NOT NULL,
  `nombre_pj` VARCHAR(300) NOT NULL,
  `sector_pj` VARCHAR(300) NOT NULL,
  `cod_pj` VARCHAR(45) NOT NULL,
  `pj_sector` VARCHAR(45) NOT NULL,
  `trabajos` INT NULL,
  `estado` VARCHAR(45) NULL DEFAULT 'REGISTRADO',
  INDEX `fk_pueblo_distrito1_idx` (`iddistrito` ASC) VISIBLE,
  PRIMARY KEY (`idpueblo`),
  INDEX `fk_pueblo_clasificacion1_idx` (`idclasificacion` ASC) VISIBLE,
  INDEX `fk_pueblo_depreciacion1_idx` (`iddepreciacion` ASC) VISIBLE,
  CONSTRAINT `fk_pueblo_distrito1`
    FOREIGN KEY (`iddistrito`)
    REFERENCES `bdtesis`.`distrito` (`iddistrito`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pueblo_clasificacion1`
    FOREIGN KEY (`idclasificacion`)
    REFERENCES `bdtesis`.`clasificacion` (`idclasificacion`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pueblo_depreciacion1`
    FOREIGN KEY (`iddepreciacion`)
```

```
  REFERENCES `bdtesis`.`depreciacion` (`iddepreciacion`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

ANEXO 7:

HISTORICO DE FICHAS DE PROCESO:

FICHA DE PROCESOS

N: 01

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Víctor Campoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipan

TIPO DE OPERACIÓN:
( )CREAR
( )ACTUALIZAR
( )ELIMINAR
(X)CONSULTAR
(X)OTRO: Uso promedio de CPU

FECHA EJECUCION: 01 / 12 / 2021

HORA INICIO: 07:00 pm

Nº ENTIDADES:

HORA FIN: 12:30 AM

Nº REGISTROS: 100 / 10000 / 1000 000

SISTEMA GESTOR DE BASE DE DATOS: Mariadb / Mongodb

SOFTWARE UTILIZADO: Monitor de Recursos windows

DESCRIPCION DEL PROCESO: Se descubre que Mongodb hace un consumo mínimo cuando se utiliza un procesador AMD. Sigue ejecutando la prueba al momento de ver el porcentaje de consumo contra Mariadb

OBSERVACIONES:

1 LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2 ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3 EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK
4 ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 02

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de data relacionales y no relacionales

ENCARGADO: Victor Campoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipan

TIPO DE OPERACIÓN:
( )CREAR
( )ACTUALIZAR
( )ELIMINAR
(X)CONSULTAR
(X)OTRO: Promedio de uso CPU /Equipo 1

FECHA EJECUCION: 07 / 11 / 2021

HORA INICIO: 03:00 pm

Nº ENTIDADES:

HORA FIN: 06:50 pm

Nº REGISTROS:

SISTEMA GESTOR DE BASE DE DATOS: Mongodb / Mariadb

SOFTWARE UTILIZADO: Monitor de recursos de windows

DESCRIPCION DEL PROCESO: Mariadb ofrece un consumo alto de CPU con la cantidad de recursos que maneja. Para cada ejecución Mongodb fue menor.

OBSERVACIONES:

1 LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE
2 ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3 EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4 ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 03

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Víctor Campoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipán

TIPO DE OPERACIÓN:
( )CREAR
( )ACTUALIZAR
( )ELIMINAR
( )CONSULTAR
(X)OTRO: Superbanes uso de RAM

FECHA EJECUCION: 05/11/2021

HORA INICIO: 08:00 pm

Nº ENTIDADES: 3

HORA FIN: 10:00 pm

Nº REGISTROS: 100 / 10000 / 100000

SISTEMA GESTOR DE BASE DE DATOS: Mariadb / Mongodb

SOFTWARE UTILIZADO: Monitor de Recursos windows

DESCRIPCION DEL PROCESO: De acuerdo a los tres procesos efectuados Mongodb fue el que menor consumo de RAM tuvo en el primer y ultimo proceso en el segundo proceso aumento en un 18% a Mariadb.

OBSERVACIONES:

1.LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2.ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3.EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4.ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 04

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Víctor Campoverde

FECHA: 2021

INSTITUCION: UNA. Señor de Sipan

TIPO DE OPERACIÓN: (X)CREAR
( )ACTUALIZAR
( )ELIMINAR
( )CONSULTAR
(X)OTRO: Uso de Memoria RAM / Creación

FECHA EJECUCION: 04/ 11 /2021

HORA INICIO: 03:50 pm

Nº ENTIDADES:

HORA FIN: 06:00 pm

Nº REGISTROS:

SISTEMA GESTOR DE BASE DE DATOS: Maria db / Mongodb

SOFTWARE UTILIZADO: Monitor de Recursos de windows

DESCRIPCION DEL PROCESO: Mongodb. hizo mayor consumo de memoria RAM en ambos casos pero cuando entra al 1ha 18 el consumo de RAM baja y se mantiene en ese rango. Maria db se aumento el uso de memoria RAM va de forma creciente.

OBSERVACIONES:

1.LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2.ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3.EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4.ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 05

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Víctor Campoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipán

TIPO DE OPERACIÓN:
( )CREAR
( )ACTUALIZAR
( )ELIMINAR
(X)CONSULTAR
(X)OTRO: Consumo de CPU – Creación de ambas bases

FECHA EJECUCION: 03/11/2021

HORA INICIO: 10:42 AM

Nº ENTIDADES:

HORA FIN: 10:50 AM

Nº REGISTROS:

SISTEMA GESTOR DE BASE DE DATOS: MariaDB / MongoDB

SOFTWARE UTILIZADO: Manejador de recursos de windows

DESCRIPCION DEL PROCESO: MariaDB marca un valor negativo debido a que el consumo final de CPU fue menor al consumo inicial. En el caso de MongoDB consumió mayor CPU en el equipo que tenía mejores prestaciones de hardware. En el segundo equipo el consumo fue menor.

OBSERVACIONES:

1. LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2. ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3. EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4. ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 06

PROYECTO: Analisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Victor Compoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipan

TIPO DE OPERACIÓN: (X)CREAR
( )ACTUALIZAR
( )ELIMINAR
( )CONSULTAR
(X)OTRO: Proceso de Migracion mediante PDI

FECHA EJECUCION: 03/11/2021

HORA INICIO: 10:42 AM

Nº ENTIDADES:

HORA FIN: 10:50 AM

Nº REGISTROS:

SISTEMA GESTOR DE BASE DE DATOS: Mariadb / Mongodb

SOFTWARE UTILIZADO: Pentaho Data Integration / Monitor de Recursos

DESCRIPCION DEL PROCESO: Solo se pudo ejecutar este proceso en el Equipo 1 ya que el Equipo 2 nos arrojo un error de memoria al intentar ejecutar el PDI de migracion.
El proceso de migracion tardo en 3819 milisegundos

OBSERVACIONES:

1.LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE
2.ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO.
3.EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK
4.ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

FICHA DE PROCESOS

N: 07

PROYECTO: Análisis comparativo de rendimiento en gestores de bases de datos relacionales y no relacionales

ENCARGADO: Yector Campoverde

FECHA: 2021

INSTITUCION: Univ. Señor de Sipan

TIPO DE OPERACIÓN: (X)CREAR
( )ACTUALIZAR
( )ELIMINAR
( )CONSULTAR
( )OTRO:

FECHA EJECUCION: 03/11/2021

HORA INICIO: 10:00 AM

Nº ENTIDADES: → Serept a Sql

HORA FIN: 10:21 AM

Nº REGISTROS:

SISTEMA GESTOR DE BASE DE DATOS: Mariadb.

SOFTWARE UTILIZADO: Heidi Sql / Administrador de recursos windows

DESCRIPCION DEL PROCESO: La diferencia de tiempo encontrada en ambos equipos donde se medió el tiempo de respuesta para crear la base de datos relacional, la variación no fue notoria a pesar que los dos equipos contaban con software diferente y hardware diferente. Sólo se encontró una diferencia de 1341 milisegundos.

OBSERVACIONES:

1 LA FICHA TIENE QUE SER LLENADA CON LETRA LEGIBLE.
2 ES NECESARIO QUE SE REGISTRE EL ENCARGADO QUE EJECUTA CADA PROCESO
3 EL TIPO DE OPERACIÓN TIENE QUE MARCARSE CON UNA X O UN CHECK.
4 ES NECESARIO QUE SE DETALLE LA DESCRIPCION DEL PROCESO

2022

ANEXO 8:

PLAN DE PRUEBAS DE RENDIMIENTO:

VICTOR CAMPOVERDE VEGA

INDICE

1 INTRODUCCION: 2
1.1 OBJETIVO: 2
2 CUADRO RESUMEN DEL PLAN DE PRUEBAS: 3
3 ENTORNO: 3
3.1 BASE DE DATOS: 4
3.2 SOFTWARE 4
4 ESTRATEGIA 5

1 INTRODUCCION:

PROYECTO	TIPO DE PROYECTO
Análisis comparativo de rendimiento en gestores de bases de datos relacionales y NoSQL	Investigación para obtener grado académico.
DOCUMENTOS DE RECOLECCION DE DATOS	
FICHA DE PROCESOS	
ENCARGADO DE PRUEBAS	
Bach. Victor Campoverde Vega	

1.1 OBJETIVO:

El propósito de este documento es proporcionar pautas y definir las estrategias que debe seguir para ejecutar pruebas de rendimiento en Gestores de base de datos.

El objetivo general del plan es establecer las condiciones para la ejecución de las pruebas de rendimiento; de manera que permita obtener datos para posteriormente elaborar comparativas y saber que gestor de base de datos tiene mejor rendimiento sobre otro.

2 CUADRO RESUMEN DEL PLAN DE PRUEBAS:

OBJETIVOS DE LAS PRUEBAS	• Recolectar información de tiempos de respuesta y consumos de recursos. • Documentar todas las pruebas realizadas. • Encontrar errores de ejecución.
DETALLE DEL ORDEN DE EJECUCION DE PRUEBAS	Las pruebas serán de manera incremental para cada proceso que se realice. Se probarán los procesos de creación, inserción, actualización. Se comenzará por 100 registros siendo el numero menor y 2350853 registros el numero mayor.
RESPONSABILIDAD DE LAS PRUEBAS	Las pruebas son responsabilidad del encargado de pruebas.

3 ENTORNO:

Para poder realizar las pruebas se necesita que estén disponibles los siguientes entornos:

A) SERVIDOR DE PRODUCCION

CARACTERISTICA	DESCRIPCION
PROCESADOR	Intel Xeon(R) CPU E3-1245 v5 3.50GHz
MEMORIA RAM	24 Gb
DISCO DURO	1 Tb
SISTEMA OPERATIVO	Windows Server 2016 Datacenter
MARCA	Lenovo
MODELO	ThinkServer 1075

B) EQUIPO LAPTOP

CARACTERISTICA	DESCRIPCION
PROCESADOR	AMD Ryzen 7 3700 U (8 procesadores lógicos)
MEMORIA RAM	16 Gb
DISCO DURO	500 Gb SSD
SISTEMA OPERATIVO	Windows 10 Home Single Language 64 bits(compilación 19041)
MARCA	Hp
MODELO	Pavilion 15

3.1 BASE DE DATOS:

Se trabajó con una base de datos abierta predial de la Entidad Cofopri la cual contenía información de los pueblos y caseríos formalizados de todo el Perú. Para mayor información de la procedencia de los datos se detalla un informe adjuntado en el **ANEXO 4.**

3.2 SOFTWARE

Para todos los procesos realizados como las pruebas de rendimiento, recolección de datos e interpretación de resultados se utilizó software libre para evitar el uso de licencias comerciales.

1. **MongoDB 4.0 Community Edition:** Se necesita realizar la instalación del servidor Mongodb, se siguió con toda la documentación citada en la página oficial del software. También fue necesaria la instalación de Mongodb Compass Community para fines de conexión.

2. **Mariadb 10.4 X64:** La instalación para Mariadb fue sencilla, configurándose como instalación mínima predefinida por el usuario, en caso el puerto de comunicación 3309 que es el utilizado por defecto este ocupado tendría que cambiarse por otro al realizar la instalación.
3. **Robo3T 1.3.1:** Este cliente fue utilizado para poder obtener los resultados obtenidos en las pruebas de rendimiento en gestores de bases de datos NOSQL.

4. **HeidiSql 10.2.0.5599:** Software utilizado para obtención de tiempos de ejecución en milisegundos en la ventana de salida a la ejecución de comandos. Para futuras investigaciones no es recomendable que se instale la versión portable de este programa, ya que puede presentar errores cuando se hagan pruebas de rendimiento con grandes cantidades de datos.

5. **Sublime Text Build 3176:** Este software fue utilizado para poder realizar la adaptación de datos de la muestra. Se utilizó este editor de texto por referencias de otras investigaciones para procesar gran cantidad de datos,(Győrödi, Győrödi, G.Pecherle, & Olah, 2016) también usaron este editor de texto.

6. **PDI (Pentahoo Data Integration):** Este software es el más importante en esta investigación ya que permitió realizar el proceso de migración de estructura relacional al modelo NOSQL.

4 ESTRATEGIA

Se requiere documentar todos los resultados entregados por el monitor de recursos de Windows y por los diversos softwares administrativos de base de datos. Todo el desarrollo de las pruebas programadas se está ejecutando en **WINDOWS**.

PRIMERA ETAPA: Ejecución de código SQL para creación de base de datos relacional.

SEGUNDA ETAPA: Ejecución de procesos para migración de base de datos relacional a MongoDB (NoSql).

TERCERA ETAPA: Ejecución de código SQL y NOSQL para inserción de datos.

CUARTA ETAPA: Ejecución de código SQL y NOSQL para consultas de datos en cantidades incrementales de menor a mayor cantidad.

ANEXO 9:

INFORME DE CONEXIÓN A SERVIDOR DE PRODUCCION:

Para el desarrollo de investigación el equipo 1 al ser un servidor de producción trabajando 24 horas al día los 7 días de la semana, necesita una conexión estable de internet que permita la configuración y el mantenimiento de bases de datos.

CARACTERISTICAS DE EQUIPO PARA CONEXION:

CARACTERISTICA	VALOR
IP PRIVADA	190.117.115.116
UBICACION	
TERMINAL SERVER	SI
SISTEMA OPERATIVO	Windows Server 2016 Datacenter
ACTIVE DIRECTORY	SI
DOMINIO	www.grupoasiu.com
FECHA TERMINO LICENCIA TERMINAL SERVER	12/06/20222
ANTIVIRUS	ESET Server Security para Microsoft Windows Server

A) INTERNET UTILIZADO:

El servicio prestado por la Compañía CLARO SAC, donde se ofrece por contrato 100 mb pero solo llegan reales 60mb.Todas las pruebas y actualizaciones al software utilizado se realizaron por la tarde para evitar el uso de la red por los trabajadores de la empresa.

B) ROUTER PRINCIPAL DE CONEXIÓN:

La conexión se realiza mediante el router de claro modelo SAGEMCOM 3284 el cual viene la conexión desde la toma central externa al mismo mediante cable coaxial tecnología HFC,la cual distribuye desde el mismo el servicio de VOZ para teléfono y DATOS para internet repartido por sus 4 puertos de color amarillo.

C) ROUTER SECUNDARIO PARA ADMINISTRACION DE INTERNET:

Se utiliza el router modelo MICROTIK RouterBOARD 951Ui 2HnD el cual se utiliza para realizar la configuración y apertura de puertos utilizados en la red. La conexión con el router principal es mediante un patch cord donde se conecta el puerto 1 con cualquier puerto de RED de color amarillo del router principal de CLARO.

D) SWITCH DE RED:

Se hace uso de un switch DLINK de 24 puertos modelo DGS-1024D.La conexión para este equipo es mediante un patch cord del router secundario al puerto 1de este.