Levantamiento del estado del arte sobre tendencias NoSQL

Juan Pablo Gómez Gil

Tecnología en Sistemas de Información

Jorge Iván Bedoya Restrepo

INSTITUTO TECNOLÓGICO METROPOLITANO

05/11/2016

RESUMEN

Las bases de datos juegan un papel fundamental, dando la oportunidad de convertir los datos en información y la información en insumo para la toma de decisiones. Hoy en día, generamos un gran volumen de información, desde fuentes anatómicas hasta uso de recursos electrónicos, dando pie a que los motores de procesamiento de datos requieran hacer uso de herramientas más poderosas, eficaces y robustas, reduciendo lo económico y lo accesible. Es en este momento, donde es necesario optimizar cada recurso y cada fuente para lograr la obtención de información de forma más rápida, efectiva y al menor costo, es por esto que las bases de datos no relacionales son una buena alternativa en los sistemas de información que surgen actualmente.

Esta última, llamada NoSQL o también conocida como “No solo SQL”, es mayormente reconocida por no usar SQL como lenguaje de consulta principal, también por no requerir esquemas de tablas fijas y por no soportar operaciones “Join”.

Las bases de datos NoSQL están desarrolladas para ejecutarse en múltiples sistemas con arquitectura de procesamiento versátil, el cual permite procesar cargas masivas de datos de forma más rápida y efectiva que otros motores estructurados que se usan actualmente para la misma labor y que no requieren tener prioridad en la consistencia en el tiempo.

El resultado final de este trabajo, devolverá un documento que enseña las diferentes alternativas que surgen en el medio para el almacenamiento y procesamiento de información, a través de bases de datos NoSQL. También, que ofrece el mercado actual, cuales son más flexibles para su implementación y de qué forma compite una con otra, mostrando los pros y contras de las opciones de software que se masifican en el medio por su compatibilidad con plataformas que emergen y son visibles en la actualidad.

Palabras clave: NoSQL, SQL, SGBD, Carga masiva de datos, Wearables, IoT, Procesamiento de Datos, Big Data, Cassandra, HBase, Redis, HadoopDB, MapReduce, Cloud Computing

RECONOCIMIENTOS

Quiero agradecer a mi asesor y maestro Jorge Iván Bedoya Restrepo, quien me dio la oportunidad de emprender este trabajo de grado a su lado y de motivarme a explorar nuevas tendencias y a apasionarme con mi carrera.

Agradezco inmensamente a mis padres, quienes desde el primer momento en que inicié mi recorrido por la academia, estuvieron acompañándome y apoyando mis ideas e iniciativas.

Agradezco al ITM, la cual se convirtió en el lugar donde pude emprender caminos que me han ayudado a crecer como persona y como un gran profesional.

Finalmente, quiero agradecer a la Biblioteca EPM y al equipo de trabajo de esta bella institución, quien no solo me apoyo en la realización del proyecto sino que me facilitó infraestructura y tecnología para lograr el sueño de ser un profesional integro.

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Las bases de datos NoSQL surgen para dar respuesta a los problemas que las bases de datos tradiciones no pueden abarcar de forma eficiente, con el fin de optimizar la capacidad de procesamiento de información y brindar escalabilidad y rendimiento.

Hoy en día las bases de datos NoSQL son una herramienta fundamental en el desarrollo de aplicaciones y tecnología móvil, cloud y demás sistemas de información que requiera tener un simplificado ecosistema de procesamiento de datos y que al tiempo procese de forma rápida y eficaz grandes volúmenes de información, tal y como pasa con las redes sociales, aplicaciones móviles, desarrollo web y Big Data, la cual migra hacia esta infraestructura tecnológica que se impone fuertemente.

El modelo de investigación pretende resolver interrogantes referentes a ¿Qué es NoSQL?, ¿Qué beneficios trae sobre otros modelos que han existido desde hace tiempo?, ¿Quiénes están trabajando con esta tendencia tecnológica?, ¿Qué posibilidades tiene dicha tendencia en entornos empresariales?, ¿Qué ventajas, desventajas y amenazas tiene dicha tecnología?, ¿Cuáles son las plataformas que se han impuesto en el mercado y de qué forma han alcanzado este logro? Y para resolverlos, se realizará una búsqueda de información respectiva en los últimos 10 años a partir de diferentes fuentes de información. La investigación iniciará en los países con mayor aporte en esta tendencia, llegando finalmente hasta Colombia, mostrando aplicaciones y uso de la tecnología.

Objetivo General:

Elaborar una revisión de la bibliografía referente al tema de bases de datos NoSQL, con el fin de tener recopilada la información del estado actual y aportes de dicha tecnología.

Objetivos Específicos:

- Comparar diferentes herramientas y alternativas que existen en el medio para el almacenamiento y procesamiento de información por medio de bases de datos NoSQL.

- Identificar las diferentes alternativas que surgen en el medio y qué ventajas tiene sobre otros modelos de bases de datos.

- Detectar tendencias de motores de bases de datos NoSQL.

2. MARCO TEÓRICO

NoSQL es una tendencia relacionada con el uso y aprovechamiento de bases de datos, el cual es incompatible con las estructuras presentadas por los motores SQL y es debido a sus características de rendimiento y fiabilidad. “En algunos casos motores NoSQL presentan características que permiten la importación de bases de datos SQL, pero no en su totalidad o al menos en la gran mayoría de los casos” [1]. Esto se debe a que la arquitectura manejada por ambos sistemas es internamente muy diferente, ya que NoSQL limita el manejo de almacenamiento de datos por esquemas fijos y en vez de eso, emplea esquemas no estructurados sirviendo por ejemplo en aplicaciones a gran escala web por el alto flujo de transaccional.

“NoSQL emplea diferentes tipos de bases de datos, de los cuales cuatro han sido exploradas a gran medida” [2], estas corresponden a las siguientes:

Bases de datos orientados a documentos; permiten anidar la información en forma de carpetas físicas. Se asemejan a los registros de bases de datos que se conocen, con la diferencia que son mucho más flexibles, puesto que carecen de esquemas.

En esta categoría existen diferentes motores que han sido reconocidos por ser lo suficientemente eficientes con este tipo de bases de datos, así como MongoDB y CoushDB.

Bases de datos orientas a columnas o column-family; estas bases de datos permiten asociar cada clave con más de un atributo o columna, de esta forma los datos pueden ser suministrados de forma rápida y eficiente, aunque tiene un inconveniente y es la perdida de consistencia de información. Una aplicación muy común es la minería de datos.

“Algunos de los motores más conocidos en dicho tipo son Cassandra, BigTable y HBase” [2].

Bases de datos claves-valor; los datos tienen diferentes características, uno corresponde a la clave y otro a la referencia de valor-clave. “Este tipo de base de datos tiene como posibilidad la escalabilidad, aunque repercute en otras características como consistencia y

además es difícil de interpretar por la carencia de esquema. Los motores de bases de datos que se caracterizan en dicho modelo son Redis, RIAK, y Amazon DynamoDB" [2].

Bases de datos gráficas o basadas en grafos; este modelo permite almacenar información en forma de nodos y relaciones entre cada uno. "Un motor muy conocido en dicha estructura es Neo4J" [2].

Cada una de estas tecnologías se usan en una gran variedad de aplicaciones y según las especificaciones o la fuente de la información, se usa uno u otro motor, aprovechando su especialidad.

Actualmente se han realizado diferentes pruebas en ambientes controlados, verificando el tiempo de respuesta y la eficiencia en diferentes motores SQL y NoSQL. Este es el caso de la Universidad Tecnológica Nacional de Argentina; en la cual se desarrolló un experimento usando una base de datos relacional, pero de fácil adaptación a estructuras NoSQL. "Se usó un total de un millón de datos para entidades correspondientes a "Personas", "Compras" y "Amigos" y de cien para la entidad Productos" [3].

Se usó un equipo con las siguientes características: "4 GB de RAM, procesador Intel Core i5 con velocidad de 2,30 Ghz y disco duro de 500 GB sobre un sistema operativo Ubuntu en versión 14.04", se realizó una carga masiva de datos en diferentes motores de la siguiente forma: "PostgreSQL como base de datos relacional, Redis como base de datos clave-valor, Cassandra para el modelo orientado a columnas, MongoDB para documentos y Noe4J como motor basado en gráficos" [3].

Se definieron estas tecnologías por ser las más optimas en cada uno de los tipos de modelos y por ser de licencia OpenSource. "Se realizó un monitoreo constante usando scrips en Python 2.7" [4].

El resultado concluyó que tanto las bases de datos tradicionales (SQL) como las documentales (NoSQL) se comportaron de forma muy similar. Pero "al momento de realizar cargas masivas, las bases de datos presentaron lentitud frente a las SQL dando un resultado desfavorable, pero al realizar consultas punto-clave, se observaron resultados muy favorables" [3].

En conclusión, las bases de datos SQL y NoSQL compiten en entornos con mucha más información, así como el BigData ya que a pesar que, en la carga masiva de datos SQL mostró ser mucho más ágil, el tiempo entre ambas tecnologías no fue muy diferente.

Dicha tendencia tecnológica, incorpora características que no son visibles en lenguajes SQL tradicionales, dando la posibilidad de trabajar en entornos altamente variados, “casos como el BigData, el cual tiene un ecosistema altamente robusto, mostrando a NoSQL como un ambiente propicio” [5].

Las bases de datos NoSQL han surgido para dar respuesta a los nuevos retos que ha introducido el BigData, aunque hoy en día es cuestionado por sacrificar la atomicidad, constancia, aislamiento y durabilidad de la información. “Muchos han empezado a incluirla en los proyectos por facilitar ventajas como rendimiento, escalabilidad física, representación de datos entre otras ventajas que son indispensables a la hora de trabajar con Big Data” [3].

Desde la llegada de la WEB 2.0, el hombre ha generado un gran volumen de datos usando blogs para la generación de contenido propio hasta la actualidad, donde el fenómeno de las redes sociales ha abarcado un gran porcentaje del internet, en el cual cada segundo se genera millones de datos que son usados para generar adaptabilidad en cada plataforma que se visita, mostrando publicidad o recomendando contenido único para cada usuario; esto también es llamado BigData.

Para soportar el fenómeno del Big Data, “se ha hecho necesario pensar en otras alterativas para almacenar y consultar volúmenes impresionantes de información, los cuales no son tan óptimos por medio de bases de datos tradicionales, sino que se hace necesario optar por motores NoSQL” [5].

Modelos relacionales, los cuales son básicamente “la estructura con la cual las bases de datos SQL trabajan, ya no son suficientes para abarcar las necesidades que se conocen hoy en día” [6].

Un ejemplo muy claro, son las aplicaciones que requieren consumir cantidades masivas de datos, los cuales son consultados y analizados en tiempo real, “casos como el internet, el cual transmite la gran mayoría de esta información por su naturaleza que actualmente se ha introducido en sensores, equipos cotidianos dando pie a nuevas tendencias como lo que hoy conocemos “IoT” o Internet de las Cosas” [7].

Hoy en día el mercado ha traído una variedad de plataformas de bases de datos, las cuales, por sus características específicas, son optadas por diferentes organizaciones; estas se dividen en cuatro categorías: relacionales, heredadas, en memoria y NoSQL.

Existen múltiples alternativas de bases de datos NoSQL que se han posicionado con el tiempo, “una de las más populares es “Cassandra”, la cual es usada en empresas que son potencias y las cuales su base es la información, tales como “Twitter, Netflix, Cisco, Rackspace, OpenX, Ooyala, entre otros”. Hasta hace tres años, un cluster de Cassandra podría alcanzar más de 300 TB de datos, los cuales estaban distribuidos en 400 máquinas” [7].

Los motores de bases de datos tradicionales se han constituido desde las últimas décadas como un insumo indispensable para la administración y almacenamiento de información usada por diferentes sistemas. Hoy ese hecho ha cambiado y se debe al crecimiento del internet.

A continuación, se presentan las principales diferencias entre las gestiones realizadas por dos bases de datos que son referencia desde el punto de vista del consumidor, las cuales corresponden a Oracle y MongoDB, una perteneciente a SQL y la otra a NoSQL.

“Una de las características que representan a este tipo de bases de datos, es el uso de esquemas JSON, permitiendo generar transacciones de forma más dinámicas y fluidas” [7].

Se pensó en esta comparación MongoDB, por ser una base de datos que a pesar de ser nueva entre las tecnologías existentes NoSQL. “Ha tomado fuerza entre plataformas tecnológicas de gran consumo de datos y que realizan miles de transacciones por manejar millones de registros, así como: MTV Networks, Craiglist, Disney Interactive Media Group,

Sourceforge, Wordnik, Yabblr, SecondMarket, The Guardian, Forbes, The New York Times, bit.ly, GitHub, FourSquare, Disqus, PiCloud, entre otras" [7].

Dicha tecnología ofrece a cada uno de los anteriores prestadores de servicios, la posibilidad de almacenar y procesar la información que se genera a partir del uso masivo por parte de la comunidad que utiliza las características de cada sistema.

Se pensó Oracle, por su antigüedad entre el grupo de bases de datos relacionales y que actualmente, domina un gran porcentaje de las empresas que generan y articulan una cantidad considerable de información a nivel global. Oracle DB, logró posicionarse desde hace casi 40 años, por su capacidad de sistema de gestión entre los diferentes mercados que contribuyen al uso y comercialización de datos para fines comerciales o particulares.

3. METODOLOGÍA

La metodología con la cual se pretende trabajar, dará como resultado identificar las tendencias en bases de datos no relacionales (NoSQL) y mostrar de qué forma han estado involucradas en diferentes aspectos a nivel tecnológico, tales como sistemas de información, generación de contenido, BigData, consulta de información y demás entornos que permiten el uso y aprovechamiento de las características de las bases de datos y procesamiento de información de forma masiva.

Con la información obtenida se pretende realizar un análisis con enfoque cualitativo, de tal manera que se puedan abordar de forma profunda las diferentes tendencias NoSQL, dando la posibilidad de afrontar desde varios frentes y dar aportes de forma crítica y explicativa acerca de los diferentes motores de bases de datos.

Suministrar información relacionada con dicha tecnología, incursionar en las diferentes herramientas que se conocen, ya sea de forma libre o de licenciamiento para comparar ventajas y desventajas, capacidades y revisar los diferentes tipos de bases de datos con las que se cuenta actualmente.

Dicho análisis se realizará por medio de un rastreo bibliográfico incursionando en bases de datos especializadas suministradas por empresas y demás instituciones, revistas académicas, libros de contenido técnico, entrevistas a personal que trabaja en el medio con diferentes herramientas tecnológicas y que han podido tener experiencias con bases de datos SQL y NoSQL, dando como resultado un estado del arte para ser usado en futuras investigaciones y benchmarking.

Para el rastreo de material bibliográfico se optará por consultar meta buscadores ofrecidos por instituciones que proporcionan información a comunidad académica e investigación de diferentes áreas. La búsqueda de información nacional e internacional se realizará por medio de bases de datos que se encuentran en bibliotecas especializadas en la ciudad y en

Universidades que tengan un enfoque tecnológico y científico. Se utilizarán herramientas y redes sociales con enfoque investigativo, así como Mendeley y usando materiales para la recolección de información tales como Zotero.

Se realizará una vigilancia tecnológica en herramientas tales como Google Académico, Google Patents, entre otras, las cuales permitirán identificar y recolectar información generada por universidades e instituciones que adelantan trabajos investigativos y que generan conocimiento a partir de la experimentación con motores NoSQL.

Se realizará un rastreo en diferentes instituciones públicas y tecnológicas, unidades de desarrollo e investigación local, empresas y otros centros de aprendizaje que han incluido bases de datos no relacionales dentro de su arquitectura tecnológica.

También se usarán otros mecanismos de recolección de información, tales como entrevistas a personal experto, que optaron por trabajar con bases de datos NoSQL, permitiendo consultar por qué el cambió de herramienta en la implementación de software, sistemas de información u otros medios que realizan labores transaccionales, aprovechando estas experiencias e incursionando de forma profunda la manera en que trabajan los diferentes motores de bases de datos.

Por medio de estas herramientas y mecanismos se buscará toda la información necesaria para el análisis de diferentes tecnologías y tendencias, permitiendo realizar una comparativa eficaz y determinar las mejores opciones para el almacenamiento de información y procesamiento de datos a diferentes escalas.

La información será recolectada de fuentes fiables y completas, bases de datos internacionales como: Academic Search Complete, la cual corresponde a una base de datos con acceso a textos completos en diferentes disciplinas académicas publicadas por EBSCO; Engineering Source, la cual está diseñada para investigadores de todo tipo; Fuente Académica Premier, la cual cuenta con publicaciones académicas en áreas de ciencia y tecnología; Science Direct, la cual cuanta con revistas editadas por Elsevier y otras editoriales con información de ciencia, tecnología e industria, entre otras.

Se tendrá acceso a meta buscadores de las principales instituciones con amplio acceso a información tales como: la Red de Bibliotecas del Área Metropolitana, Biblioteca EPM, Universidad EAFIT, ITM, Pascual Bravo y otras instituciones con enfoque tecnológico.

La información obtenida se segmentará por regiones e etapas temporales, dando la posibilidad de visualizar en que zonas se realiza más actividades con dicha tendencia, cuál ha sido el avance en los últimos 10 años mostrándose por épocas y cuales han sido los motores o herramientas que más han perdurado y su evolución por medio del trabajo con las mismas.

4. RESULTADOS Y DISCUSIÓN

Panorama Sistemas de Gestión Bases de datos NoSQL

Desde que el hombre tiene facultad de registrar cada una de las actividades de su cotidianidad, ha buscado múltiples alternativas para almacenar información de forma efectiva, clara y concisa. Sin embargo, al pasar el tiempo estos métodos han tenido que variar de forma rigurosa por diferentes factores.

Hoy tenemos posibilidades de poseer esta información a la mano gracias a la tecnología y a su vez la tecnología ha tenido que adaptarse a nuestras necesidades, ya sea por medio de herramientas, métodos, capacidades o en este caso otras alternativas no tradicionales, dando paso a tendencias.

“Las bases de datos comenzaron a aparecer a finales de 1950 y comienzos de 1960” [1], gracias a la estabilidad en la computación y el aprovechamiento de los diferentes medios de almacenamiento (almacenamiento en cintas y unidades de disco). Los lenguajes de consulta estructurada (SQL) no sólo representaban una alternativa, sino una obviedad al momento de almacenar información desde cualquier medio. Sin embargo, por las mismas razones en las cuales, la información ha resultado difícil de manejar por los altos volúmenes que se generan, se ha hecho necesario indagar en otras formas de contenerla sin que se vea afectada y más aún, buscando la manera en que se procese y visualice de forma más efectiva, generando nuevas tendencias, que con el paso del tiempo se han convertido en alternativas que perdurarán hasta que se generen otras necesidades.

“En los últimos años ha aumentado el interés por las bases de datos NoSQL (Not only SQL)” [2], un nuevo concepto a la hora de manejar y procesar la información que se genera en diferentes medios. Las bases de datos NoSQL resultan como una nueva posibilidad dentro del riguroso campo del almacenamiento de información, esto es, debido a que independientemente de la herramienta, dispositivo o mecanismo que se use, siempre la información resulta ser prioridad ya que representa un insumo necesario para la toma de decisiones y el actuar del día a día.

“En 1970 se propusieron por primera vez las bases de datos relacionales y las teorías subyacentes” [3], y una de ellas fue precisamente las bases de datos relacionales, las cuales resultaron ser una revolucionaria alternativa al momento de almacenar información y esto fue debido a que la metodología implicaba usar tablas separadas con información

segmentada para luego generar soluciones. Dichas consultas utilizan un lenguaje estructurado, de ahí su nombre común SQL (Structured Query Language), apoyado en el álgebra relacional y que permite generar consultas de forma interpretativa.

Hoy en día es usado este mecanismo para el almacenamiento y procesamiento de información, pero surge un nuevo paradigma que implica la transformación total de este modelo y que a su vez debe asumir el reto de satisfacer las nuevas necesidades que surgen en el medio. Estas necesidades aparecen en el momento en que la nueva era empieza a generar contenido, aparecen nuevos dispositivos y esto se debe al crecimiento del internet que requiere segundo tras segundo abastecerse de información que se genera y se calcula con cada clic y comportamiento humano. Para dar respuesta a estas necesidades, surgen alternativas y tendencias que son impulsadas por tecnologías existentes, es en este momento que aparece NoSQL. “Por otra parte, desde la aparición del término NoSQL existe un inconveniente conceptual” [1], ya que se definía como “No Relacional”, ocasionando que no fuese una alternativa lo suficientemente atractiva ni mucho menos fuera tomada como una oposición al SQL. Finalmente fue definido como NoSQL haciendo referencia a la definición “No solo SQL”, y que este mismo como lo define S. Drob en su escrito “NoSQL and CAP” es “un movimiento más que como una tecnología” [4].

De esta forma y como lo aclara Shashank Tiwari, John Wiley & Sons: “NoSQL es usado como un término general por todas las bases de datos y almacenes de datos que no siguen los populares y bien establecidos principios RDBMS (Relational Database Management System)” [5].

Todos estos conceptos nos ilustran que NoSQL es un conjunto de tecnologías que hacen referencia a un ecosistema de alternativas al SQL, donde la razón de ser es la velocidad, el manejo de grandes cantidades de datos y la posibilidad de tener un sistema distribuido.

“El paradigma de bases de datos NoSQL surge a causa del cambio que se ha dado en el manejo de la información durante las últimas décadas” [6], para el 2014 se hablaba de un crecimiento masivo en tráfico IP a nivel mundial, el cual estaba proyectado en 1 zetabyte por 4,8 zetabyte para el 2015; esto da cuenta de que hoy en día el volumen de información crece a un ritmo significativo y gracias a esto, la administración se torna mucho más compleja que años atrás. Las empresas dejaron de almacenar simplemente la información, sino que lo convirtieron en un insumo indispensable para la toma de decisiones, los usuarios requieren que la información se visualice de forma mucho más rápida y eficaz con cada clic, y la arquitectura de los sistemas ha tenido un cambio considerable. Estos factores justifican fácilmente el nacimiento del NoSQL; y como lo indican Alexander C., Juan Sebastián G.,

Mauro C. en la publicación “Databases NoSQL’s Utility and Functioning”; el porqué de la existencia de las bases de datos NoSQL se puede resumir en tres aspectos:

Tamaño y cantidad de información:

Los archivos han tenido una modificación en su formato y calidad, esto va de la mano con el tamaño que ocupa en espacio de almacenamiento; un ejemplo conciso es el formato de video. Años atrás un video podría ocupar hasta 4,7 GB de almacenamiento en un DVD tradicional (R o RW), pero con la evolución de este formato y el aumento en su calidad, ha migrado a formatos mucho más robustos como el Blu-ray y el 4K que actualmente tiene capacidad mayor a los 60 GB.

Las grandes industrias que operan cantidades impresionantes de información se dieron cuenta que con la infraestructura que contaban no podrían manejar la cantidad de información que se proyecta para dentro de los próximos cinco años, por esto, “son esas compañías las que están invirtiendo y creando soluciones NoSQL” [1]. Casos como los que presentan las redes sociales, donde eventos mundiales como los juegos olímpicos, reinados y mundiales esperen millones de usuarios activos mensuales o que a su vez sean responsables de decenas de terabytes de consumo en datos al día a partir de publicaciones y actividades por parte de los internautas [7].

Estos ejemplos dejan claro que “el aumento tanto en el tamaño de los archivos como en su cantidad ha incidido en el crecimiento del interés hacia tecnologías NoSQL” [1]; tomando como partida, las dificultas que tienen las soluciones actuales (SQL) en el tratamiento de la información y más aún en la escalabilidad de la infraestructura, ecosistemas menos productivos (aumento en datos y disminución en la productividad de los sistemas de gestión de bases de datos) y se conviertan finalmente en un problema para quienes administran este tipo de infraestructura tecnológica, que día a día es menos intuitiva gracias a que las sentencias se convierten en largas y poco precisas [1].

Velocidad:

Es notable que el crecimiento del internet ha aumentado de forma exponencial en los últimos años, por lo cual las empresas de telecomunicaciones han realizado un esfuerzo

para aumentar la velocidad en transmisión de datos. Este factor ha ocasionado que los usuarios tengan un grado de exigencia sobre el tiempo de espera en la respuesta en las peticiones realizadas sobre cualquier sistema de información. “Es la presión de los usuarios y el surgimiento de nuevas tecnologías (como el software en la nube y el streaming) las que han impulsado el crecimiento de las tecnologías NoSQL” [8]; con millones de usuarios por segundo requiriendo productos y servicios complejos, de forma rápida y efectiva, basados en almacenamientos y búsqueda de información, se hace necesario un paradigma en bases de datos orientado a la velocidad.

“Es un hecho que cualquier aplicación moderna va a tener una arquitectura distribuida” [9]; en la actualidad abundan aplicaciones WEB o aplicaciones basadas en consultas utilizando sistemas de información en línea, la obsolescencia programada que busca llevar el software de escritorio a la red, tienen muchos usuarios que al mismo tiempo exigen respuestas razonablemente ágiles y especificas basadas en intereses o acciones ejecutadas en las aplicaciones. Finalmente, el usuario desconoce el tipo de arquitectura que es usado para realizar este tipo de procesamiento, “como se demuestra en un experimento realizado para medir la paciencia de los usuarios en las consultas, el cual indicó que demoras de medio segundo tienen serias consecuencias en las métricas del negocio” [10]; esto da cuentas que el usuario no tiene interés en conocer a profundidad el mecanismo usado ni mucho menos la tecnología que es ejecutada para la realización de su tarea. Sin embargo, las empresas deben preocuparse por este hecho y como mejorar la ejecución de sus servicios ofertados al cliente final, sin dejar a tras otros conceptos indispensables como lo son la seguridad y la imagen que muestran sus sistemas de información para establecerse de forma intuitiva y sin que el prestador del servicio tenga la necesidad de interceder en las transacciones del cliente.

“Las bases de datos relacionales están diseñadas para ser “organizadas”; las tablas funcionan de tal forma que las podemos entender” [1]; Desafortunadamente, los desarrolladores y arquitectos de la época de creación de los sistemas de gestión de bases de datos, no pensaron en que los datos fueran a crecer, las necesidades de información fueran a ser mayores y que los métodos de procesamiento de datos pudieran tener la posibilidad de trascender en el tiempo e ir avanzando a la par con las necesidades del momento.

Escasez de innovación:

Las bases de datos estructuradas fueron creadas para suplir las necesidades que surgieron en una época específica (1950-1960), sin tener contemplado que en un futuro dicha información crecería.

“Es claro que las soluciones SQL están de alguna manera estancadas, además, para la comunidad “Open Source” ha habido noticias inquietantes últimamente” [1], “(la compra de MySQL por parte de Oracle)” [11]), eso sin contar con los costos de las soluciones más populares, lo cual hace que los desarrolladores miren hacia otro lugar: “Oracle te diría que con el grado justo de hardware y la configuración correcta de Oracle RAC (Real Application Clusters) y con el software mágicamente asociado, se puede lograr la misma escalabilidad. Pero ¿a qué precio?” [12].

Las posibilidades que se ofertan en soluciones basadas en tecnología SQL tradicional, se enfrascan en los motores actuales de proveedores como Microsoft u Oracle, los cuales no ofrecen nuevas ideas o alternativas, dejando un gran espacio entre las novedades que ofrece cada versión. “En este punto se puede concluir que el cambio del paradigma en el manejo de la información motivó el auge del movimiento NoSQL” [13]; también, hay que destacar la motivación de empresas gigantes a nivel de tecnología como lo es Google, que con la muestra de su investigación “Bigtable: A Distributed Storage System for Structured Data”, en el 2006, motivo a que la comunidad desarrolladora empezara a trabajar y a mejorar esta tendencia.

La tendencia “NoSQL” habla de un modelo eficiente para las necesidades de esta generación, con la capacidad de interacción con otra tecnología existente y en crecimiento (Computación en la Nube), a su escalabilidad, la desvinculación del modelo del hardware del modelo de datos y proporcionar el acceso a la información de forma más eficiente. “Debe destacarse que el tema no es nuevo, pero históricamente se puede decir que en la última década se ha hablado mucho de él” [14].

Principales características de NoSQL

NoSQL ha representado una tendencia entre los sistemas actuales de bases de datos y a pesar de su exploración y recorrido en sistemas actuales, hoy no es posible definir sus características generales. Sin embargo, se proponen 6 atributos que corresponden a este tipo de bases de datos [15]:

Escalabilidad horizontal: Este atributo hace referencia a su facilidad para añadir, eliminar o realizar diferentes operaciones con hardware del sistema, sin afectar el rendimiento.

Habilidad de distribución: Este hace referencia a la escalabilidad horizontal, pero haciendo énfasis en su soporte; para ello se tiene en cuenta la habilidad de replicar y distribuir los datos sobre los servidores.

Uso eficiente de recursos: Tiene la capacidad de hacer uso de nuevas tecnologías y además de forma eficiente, dentro de esta categoría se tienen los discos en estado sólido, aprovechamiento de la memoria RAM y demás sistemas distribuidos.

Libertad de esquema: Al carecer de un esquema estático, tiene la particularidad de permitir de forma sencilla el modelamiento de los datos; además facilita la integración con los lenguajes de programación orientados a objetos, lo que evita el proceso de mapeado.

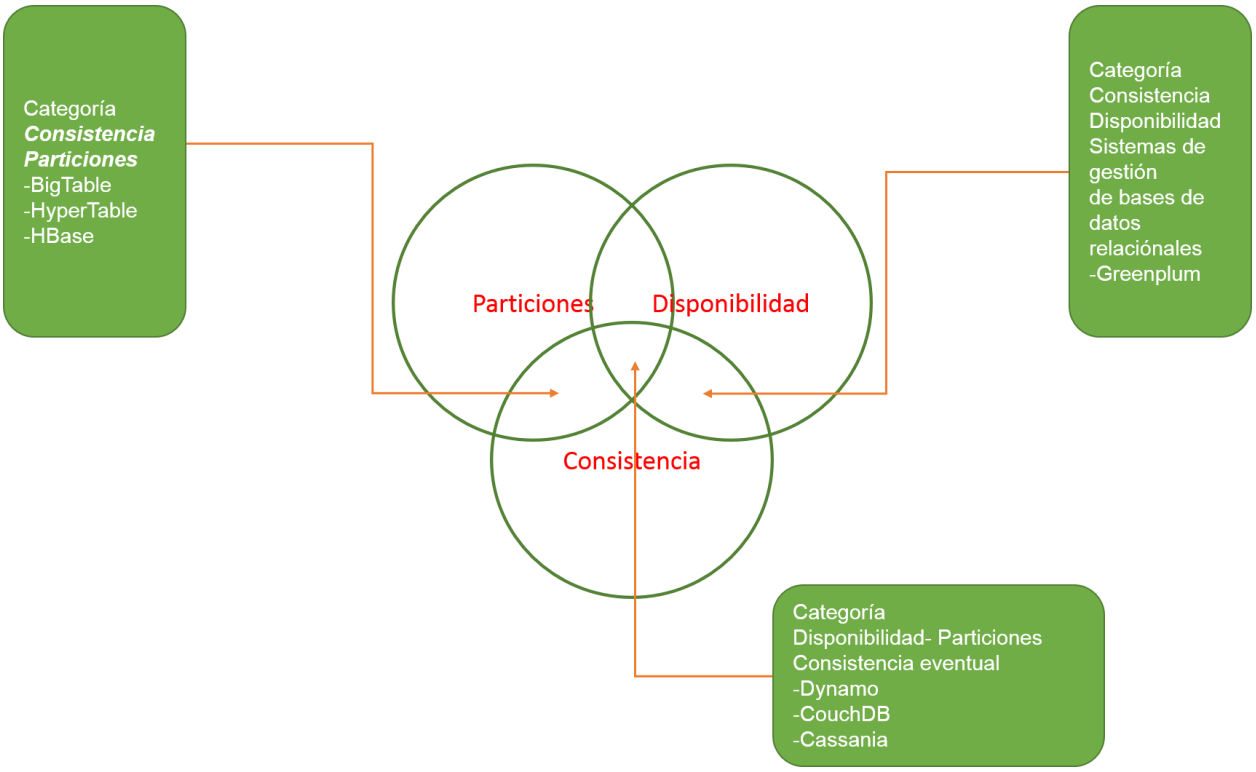
Modelo concurrencia débil: "No implementa ACID (Atomicity, Consistency, Isolation and Durability), que reúne las características necesarias para que una serie de instrucciones puedan ser consideradas una transacción, sin embargo, sí se tienen en cuenta algunas consideraciones para asegurar estos aspectos" [16].

Consultas simples: Las consultas son más concisas por lo cual requieren de menos operaciones, por lo cual para un desarrollador es mucho más sencillo entender el lenguaje natural que estos manejan.

Por otro lado, "se propone considerar cuatro aspectos teóricos muy importantes" [17]:

"Los sistemas NoSQL normalmente se apoyan en esquemas débiles de consistencia" [18], como es el caso de transacciones limitadas a elementos de datos simples o consistencia eventual. A continuación, se enseña de forma gráfica el teorema CAP [21]:

Teorema CAP



Definición Teorema CAP: (Consistency Availability Partition tolerance); en el 2000, Eric Brewer [20] propuso la idea de que en un entorno distribuido un sistema no puede mantener continuamente consistencia perfecta, disponibilidad y tolerancia partición simultáneamente.

Consistencia: Los usuarios tienen la posibilidad de acceder desde varios apuntadores a un mismo registro (al mismo tiempo).

Disponibilidad: La precaución de que cada petición recibirá una respuesta.

Tolerancia de reparto: El sistema sigue en funcionamiento a pesar de que éste sufra una pérdida parcial de información.

Estos aspectos permiten ver de forma global que para ganar velocidad las bases de datos modeladas en NoSQL deben sacrificar como mínimo de una característica.

CP: El sistema ejecutará las operaciones de forma consistente, aunque se pierda la comunicación entre nodos (partición del sistema), pero no se asegura que el sistema responda.

AP: el sistema siempre responderá a las peticiones, aunque se pierda la comunicación entre nodos (partición del sistema). Los datos procesados pueden no ser consistentes.

CA: el sistema siempre responderá a las peticiones y los datos procesados serán consistentes. En este caso no se permite una pérdida de comunicación entre nodos (partición del sistema).

ACID: ACID es muy importante en los sistemas de bases de datos relacionales; sin embargo, en los sistemas NoSQL se hizo evidente que, con el fin de proporcionar una gran escalabilidad, puede ser necesario redefinir algunas de las cualidades de este modelo, en particular consistencia y durabilidad.

Por ejemplo, en el caso de la consistencia, los sistemas tradicionales (sistemas de gestión de bases de datos SQL) utilizan un modelo que cuenta con consistencia completa, la cual ocasiona que el tiempo de ejecución y respuesta sea mucho más lento, esto es debido a que se implementa lo que es conocido como “Cerraduras”, la cual protege los registros de tal forma que dos usuarios no puedan alterar al mismo tiempo la estructura de un archivo, en el caso que se esté modificando o se cuente con el abierto por alguna operación; con NoSQL se opta por una consistencia casual, que permite a cada sistema hacer cambios a los datos y aprender de otras actualizaciones realizadas por otros sistemas dentro de un corto periodo de tiempo, sin ser totalmente coherente en todo momento. [16]

Este argumento reafirma y justifica la perdida de características en la estructura SQL tradicional, que en el caso de las estructuras NoSQL no pasaría. Sin embargo, ofrece nuevas características que en este caso las estructuras SQL no cuentan.

Las bases de datos han tenido en los últimos años una gran relevancia en los procesos de diferentes sistemas, por lo cual tanto investigadores, como desarrolladores expertos en el tema han planteado diferentes alternativas que cubren dicho "bache" dentro de la tendencia para no afectar de forma drástica a los diferentes sistemas que se intervienen:

Datos y modelo de acceso: El modelo de datos de los sistemas de gestión de bases de datos relacional con sus respectivas características (tablas, relaciones, vistas, filas y columnas), ha sido muy útil y proporcionado soluciones a las necesidades que han surgido en los últimos años; el orden que brinda y la estructuración son cuestiones que se destacan pero como se mencionó anteriormente, estos tienen problemas a la hora de una implementación, entre ellos, la relación con los lenguajes orientados a objetos y la dificultad para la adaptación a datos no estructurados. Por otro lado, el modelo de acceso a las bases de datos relacionales presenta dificultades en la gestión de las restricciones globales en un entorno distribuido, y los intenta solucionar con la creación de barreras para coordinar cambios; esto introduce sobrecarga de la red, y a veces puede detener el progreso en el sistema por su alto consumo en recursos y por ser eficaz a la hora de realizar tareas complejas de procesamiento.

Datos y procesos distribuidos: Las bases de datos NoSQL permiten la aplicación de procesos a los datos que se encuentran distribuidos sin necesidad de centralizarlos; este almacenamiento distribuido contribuye positivamente a la fiabilidad y escalabilidad de la base de datos, pero presenta muchos retos en su implementación, como los costos extra para el almacenamiento de datos duplicados y su correcta gestión.

Se hace necesario unificar criterios para estudiar a las bases de datos NoSQL, por eso se establecen tres características generales:

Descentralización de sistemas: Esto quiere decir que son sistemas distribuidos y que permiten una fácil escalabilidad horizontal; por eso, para futuras consideraciones se hace necesario revisar las características de sistemas distribuidos.

Esquema flexible: No quiere decir que no existan esquemas, sino que cada tecnología maneja un esquema que no es rígido. Estos esquemas están enfocados en las características de los sistemas actuales, y que el modelado sea sencillo y natural para quien realiza su implementación.

Cambio de modelo: Hay quienes pretenden que NoSQL mejore sus fallas sacrificando sus principios, lo que resultaría volver al modelo relacional; sin embargo, hay que recordar que este es un nuevo paradigma y necesita nuevas propuestas, un ejemplo es lo que pasó con ACID, en vez a implementarlo con modificaciones, se propone un sistema llamado "BASE (Basically Available, Soft State, Eventually Consistent), que es totalmente opuesto a ACID. Donde esta última forza la consistencia al final de cada operación, "BASE es optimista y acepta que la consistencia de base de datos estará en un estado de flujo" [19]. La disponibilidad de BASE se logra mediante el apoyo a fallos arbitrarios sin necesidad de esperar un fallo total del sistema.

Todas estas especificaciones de bases de datos dan pie para abrir un campo de investigación acerca de que tecnología es más viable para una unidad de negocios o incluso que tecnología es más efectiva según las necesidades presentes y la proyección de la institución o del sistema de información que se use.

"Reporte Claremont en investigación en bases de datos" [20], define cinco oportunidades de investigación:

Revisión de motores de bases de datos: Dentro de esta se incluye NoSQL, la cual nació justamente para fortalecer los diferentes sistemas de gestión de bases de datos según las necesidades que emergen en la era de la información.

Programación declarativa para plataformas emergentes: Los usuarios de NoSQL son capaces de escribir código robusto para entornos complejos debido a la cercanía entre el modelado en NoSQL y el modelado del negocio. En este cabe apuntar que la forma de inserción de información en estos sistemas, son justamente entradas de texto por JSON, XML entro otros conocidos en el mundo del desarrollo de software.

Interacción de datos estructurados y no estructurados: Una de las características principales de NoSQL, es el procesamiento de datos no estructurados, por lo cual se convierte en su fuerte sobre lenguajes de bases de datos estructurados.

Nube de servicios de datos: La computación en la nube (Cloud Computing) fue uno de los factores causales del resurgimiento de esta tendencia, ya que esta misma requiere de un alto volumen de procesamiento de información (una de muchas condiciones), característica que no es soportada de forma efectiva por modelos tradicionales.

Aplicaciones móviles y mundos virtuales: La necesidad de gestionar grandes cantidades de diversos datos generados por el usuario, "sintetizar de forma inteligente y de proporcionar servicios en tiempo real es satisfecha por las bases de datos NoSQL a la perfección" [16].

Modelos de bases de datos

NoSQL utiliza diferentes modelos para su sistema de almacenamiento, de los cuales se describirán 4 de los más usados:

1. **Bases de datos basados en depósitos clave / valor**: Este contenedor es análogo a un catálogo o mapa, en el que por cada llave hay un valor inscrito. El valor normalmente se guarda en forma binaria o más conocido como "BLOB", de esta forma es posible almacenar valores carentes de estructura. "No obstante, algunas implementaciones llave valor soportan listas como valores. Algunos de los beneficios de usar la aproximación llave-valor son su simplicidad y la facilidad para escalar" [22].

   Los contenedores llave-valor son útiles para realizar transacciones básicas, basadas en atributos llave únicamente. Consultas extensas o por rango resultan muy poco útiles en este tipo de contenedores. Esto se debe a que sus API permiten la recopilación de datos únicamente haciendo uso de operaciones get, put y delete. "Si se desean funcionalidades adicionales, están deben ser implementadas como

una capa superior, lo cual llevar a reducir el rendimiento y a aumentar la complejidad del sistema" [23].

Algunos ejemplos de sistemas de almacenamiento llave-valor son "DynamoDB, MemcacheDB, SimpleDB, Voldemort y Redis" [24].

2. **Bases de datos basada en Documentos**: Parecido a un sistema llave-valor, hay una llave única y una información. La diferencia con el modelo llave-valor radica en que los datos son incluidos con formato JSON o XML.

   Uno de los inconvenientes con este tipo de bases de datos es que la mayoría de sus implementaciones no pueden realizar uniones (JOIN) u operaciones que abarquen varias filas o documentos, por ello se hace necesario acudir a la des normalización de datos. "Esta restricción es deliberada, porque facilita a la base de datos realizar particionado automático, el cual es útil para escalar" [25]. "Aquí es Importante anotar que, aunque la de normalización facilita la escalabilidad, incrementa el costo de las actualizaciones a la base de datos y puede llevar a pérdida de consistencia y de características transaccionales. Además, requiere sincronización global para poder mantener consistentes los duplicados.

   "MongoDB y CouchDB son ejemplos de sistemas de almacenamiento de este tipo" [22].

3. **Bases de datos Tabulares u orientadas a Columnas**: Las bases de datos tabulares en vez de almacenar los datos en filas, se guarda en columnas. Las columnas pueden reunirse en un conjunto denominado "familia de columnas". Gracias a esto, es posible adquirir agilidad en las lecturas realizadas por el gestor de bases de datos. "Son ampliamente usadas para el cálculo de datos agregados" [22].

   "Cassandra" [26], "BigTable" [27] y "HBase" [28] son ejemplos pertenecientes a esta categoría.

4. **Bases de datos orientadas a Grafos**: Este tipo de bases de datos almacenan la información en forma de grafo, esto quiere decir, con nodos (entidades) y aristas (relaciones). Esto permite darle relevancia no solo a los datos, sino a las integraciones entre ellos. "De hecho, las relaciones también pueden tener atributos y es posible efectuar consultas directas a relaciones, en vez de a los nodos" [28].

Bases de datos tales como Neo4j, AllegroGraph y FlockDB ilustran este modelo.

Conclusiones definiciones generales

En conclusión, a pesar del alcance, trabajo y compatibilidad de estas tecnologías con las necesidades que han surgido, hay temas referentes a su desempeño que aún no han sido resueltos; uno de ellos es el escalamiento. “Esta limitación ha motivado al surgimiento de otro tipo de sistemas, denominados NoSQL” [22], los cuales pueden escalarse de forma óptima, gracias al aprovechamiento de sus características que se derivan del escalamiento horizontal, flexibilidad, particionamiento de datos (sharding), al buen manejo del concepto de redundancia y a la posibilidad de ejecutar tareas concurrentemente. Los sistemas NoSQL han hecho que los desarrolladores, para lograr un alto rendimiento, adopten soluciones de compromiso entre aspectos tales como alta disponibilidad mediante el uso de replicación, balanceo de carga, particionamiento y el modelo de consistencia. “En lo que respecta a consistencia, se destaca que estos sistemas son capaces de soportar modelos tales como consistencia eventual y consistencia débil” [22].

Comparativa y rendimiento entre Sistemas de gestión de bases de datos

NoSQL ha representado una alternativa para ambientes empresariales por la creciente demanda en la información que se genera. Años atrás se trataba de cantidades exorbitantes al hablar de Gigabytes e incuso de Terabytes de información, pero hoy en día se habla de Exabytes.

Es por esto que los sistemas de gestión de bases de datos (SGBD) no estructurados han surgido como alternativa para sustituir los modelos estructurados que se conservan actualmente y que no tienen capacidad distribuida para soportar la demanda que surge en el momento. Sin embargo, aún se preservan motores de bases de datos que han tenido recorrido en este tipo de gestiones, incluso han sobrevivido gracias a las mejoras o al aprovechamiento del crecimiento de la computación que hay en el momento.

Es importante delimitar las características y capacidades de ambos esquemas, como por ejemplo que NoSQL carece de operaciones “Join”, las cuales están presentes en esquemas

relacionales. Sin embargo, bajo la necesidad de este requerimiento y la amplia escases de alternativas, han surgido desarrollos que imponen soluciones a este problema, los cuales se detallarán de forma minuciosa en esta sección.

NoSQL ha representado un crecimiento importante en los últimos años, específicamente desde el año 2009, época en la cual se retomó dicho termino debido al crecimiento del internet, "este crecimiento ha originado el término Big Data [31]". Aunque existen varias definiciones, "Big Data se refiere a enormes volúmenes de datos (estructurados, semiestructurados y no-estructurados)" [32], esta información se traduce en Tweets, entradas de blogs, correos electrónicos, mediciones de sensores inteligentes (Internet de las Cosas) y demás medios de comunicación, difusión y generación de contenido proveniente de internet.

Para dar respuesta a las necesidades flotantes a causa del internet, se tienen en cuenta motores que contengan esquemas de datos que no estén predefinidos, que los valores de los atributos estén multivalorados, entre otras prestaciones que deben tener como componentes. Sin embargo, las características "esquemas de datos que no estén predefinidos y que los valores de los atributos estén multivalorados" [36] diferencian los esquemas NoSQL de los SQL.

Una manera de clasificar los SGBD NoSQL es la forma en que almacenan los datos:

Clave / valor: El almacenamiento se realiza por medio de distribuciones con dos componentes: una clave y un valor. Ejemplo: {"universidad": "ITM"}, donde la clave es "universidad" y su valor es "ITM". "Entre los SGBD NoSQL de este tipo están Redis, Dynamite y Voldemort DB" [33].

Columna o tabular: El almacenamiento se hace por medio de columnas. Esto quiere decir que, una fila de datos corresponde a un grupo de valores para un mismo atributo, como se muestra en la Tabla 1. En una base de datos relacional, una fila de datos corresponde a un grupo de valores, uno para cada atributo del esquema, como se muestra en la Tabla 2. "Entre los SGBD NoSQL de este tipo están Apache Hbase, Cassandra y Hypertable" [33].

Grafo: El almacenamiento se hace mediante grafos. En los nodos se almacenan objetos, las cuales se relacionan por medio de aristas, uno de ellos es HadoopDB.

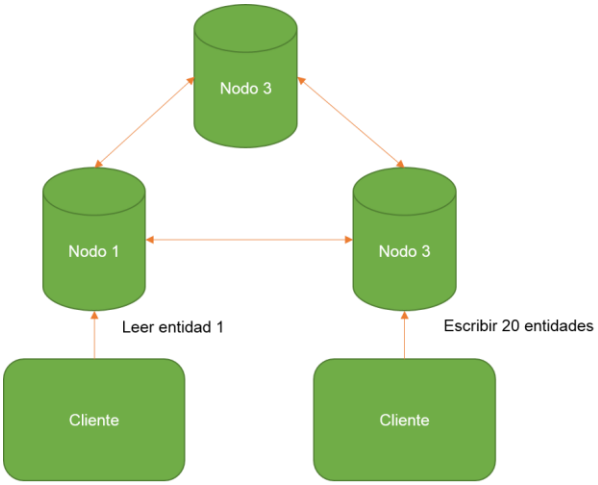
Documento: El almacenamiento se hace en repositorios de documentos. "Los documentos JSON (JavaScript Object Notation) los cuales se explican en la Sección 2. Entre los SGBD NoSQL de este tipo están MongoDB y Apache CouchDB." [33].

Comparación entre sistemas de bases de datos estructurados y no estructurados

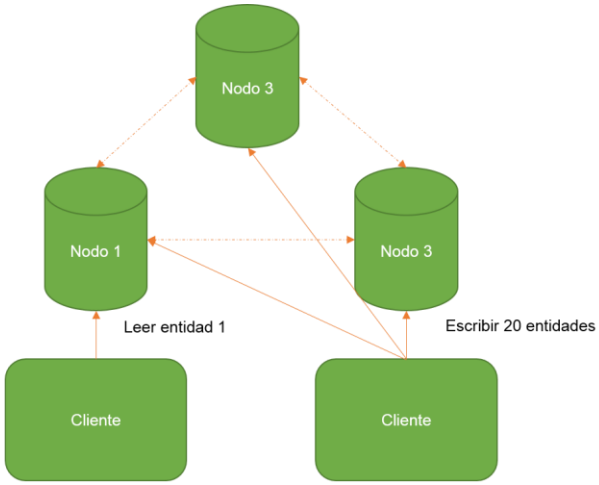
El modelo tradicional SQL ha traído desde su creación muchos beneficios tales como integridad, compatibilidad estandarización, fiabilidad de independencia de datos, y facilidad de implementación en lenguajes de programación orientados a objetos. Este esquema propone una infinidad de posibilidades para realizar transacciones, entre ellas unir tablas con base a relaciones, obtener agregados y seleccionar subconjuntos de datos con un criterio de búsqueda. La aparición de las aplicaciones web y las redes sociales han demostrado inconvenientes en estos modelos para trabajar este tipo de ambientes. "Es demasiado estricto, no da soporte a estructuras de datos complejas, la realización de JOINS es ineficiente" [22].

Como se mencionó anteriormente, los problemas de estos modelos son la carencia de distribución horizontal y carga. "A estas aplicaciones se les dificulta lograr sharding automático" [22]. Esta funcionalidad no es asignada por defecto. "El sharding requiere que entidades de datos diferentes se distribuyan a través de un número de nodos de la base de datos y que también éstas se procesen y escriban de forma independiente" [22].

A continuación, se muestra un gráfico comparativo entre ambos modelos de distribución de entidades en a) Bases de datos SQL y b) NoSQL. [29]:



a. Replicación, bloqueo y consistencia son verificados mediante Commit de 2 fases



b. Las escrituras son completamente paralelas y no están bloqueadas por comunicación por nodo

En la siguiente tabla, se analizan diferentes implementaciones de ambos sistemas [30]:

Característica	Base de datos Relacional	Base de datos No Relacional
Modelo de almacenamiento de datos y Esquema	Considera 3 aspectos: relaciones (tablas), atributos (columnas de una relación) y dominios (valores de un atributo). El esquema, estructura y tipos de datos se debe especificar antes de poder empezar a agregar datos. Si se requiere modificar el modelo es necesario alterar la base de datos completa, quedando durante este proceso en estado "fuera de línea".	Varía dependiendo del sistema de almacenamiento, siendo posible manejar información estructurada, no estructurada o semiestructurada. Se pueden almacenar datos disímiles juntos. Formatos como BLOB, XML, JSON, o cualquier otro son admisibles. No requieren un esquema definido, esto en general, puede hacerse de forma dinámica. Al no tener un modelo de datos definido pueden presentarse problemas en caso de migración.
Modelo de desarrollo	Código abierto código cerrado	Generalmente de código abierto.

Soporte para transacciones y recuperación de fallas	Puede ser configurado para que la operación finalice completamente o no se ejecute en absoluto.	Se da dependiendo de las circunstancias y en determinados niveles, por ejemplo a nivel de elementos de datos simples. Algunos sistemas dan soporte de transacciones ACID, mientras que otros tales como MongoDB no ofrecen transacciones, en su reemplazo tienen 2 opciones: operaciones atómicas y commit de 2 fases.
Manipulación de datos, interfaces e interoperabilidad	Ya está estandarizado. Hay un lenguaje específico, SQL.	Ofrecen diferentes APIs orientadas a objetos para interactuar con datos. Las interfaces para los servicios NoSQL no se han estandarizado, aún no se tiene semántica estándar.
Consistencia	Consistencia estricta	Es configurable dependiendo del producto. Algunos ofrecen consistencia estricta y otra consistencia eventual.
Escalabilidad	Usualmente vertical, puede escalar horizontalmente pero es más restrictivo	Horizontal
Soporte para almacenar datos a través de múltiples equipos, Autosharding	Capacidad no proporcionada de forma nativa, pero puede lograrse.	En su mayoría ofrecen soporte nativo y automático de auto-sharding.
Ámbito operacional	Operación sistemática y autónoma. Si	Si algo falla para diagnosticar el problema se requiere pasar por la cadena, incluso es posi
Replicación	Soportada	En su mayoría, este proceso es automático

Para ver de frente estas características sobre las ofrecidas por esquemas SQL, se realizará comparativo entre diferentes motores de bases de datos NoSQL y SQL, evidenciando particularidades, características, rendimiento, resultados entre otras anotaciones que se deriven de las pruebas en ejecución de volúmenes masivos de datos.

Para dicha comparativa, se realizarán operaciones CRUD (create, read, update y delete) y se enfoca para desarrolladores, investigadores y diseñadores que requieren un punto de inicio para elegir un ecosistema para sus aplicaciones, teniendo en cuenta que dicho análisis debe realizarse tomando en cuenta si la aplicación es de lectura o de escritura masiva.

En primera medida se evaluará un SGBD NoSQL del tipo “documental”, correspondiente a MongoDB, seguido de una base de datos del tipo “columna”, correspondiente a Cassandra, luego una base de datos del tipo “grafo”, correspondiente a HadoopDB y finalmente, una base de datos del tipo “clave valor” correspondiente a Redis.

1. Bases de datos tipo Documento

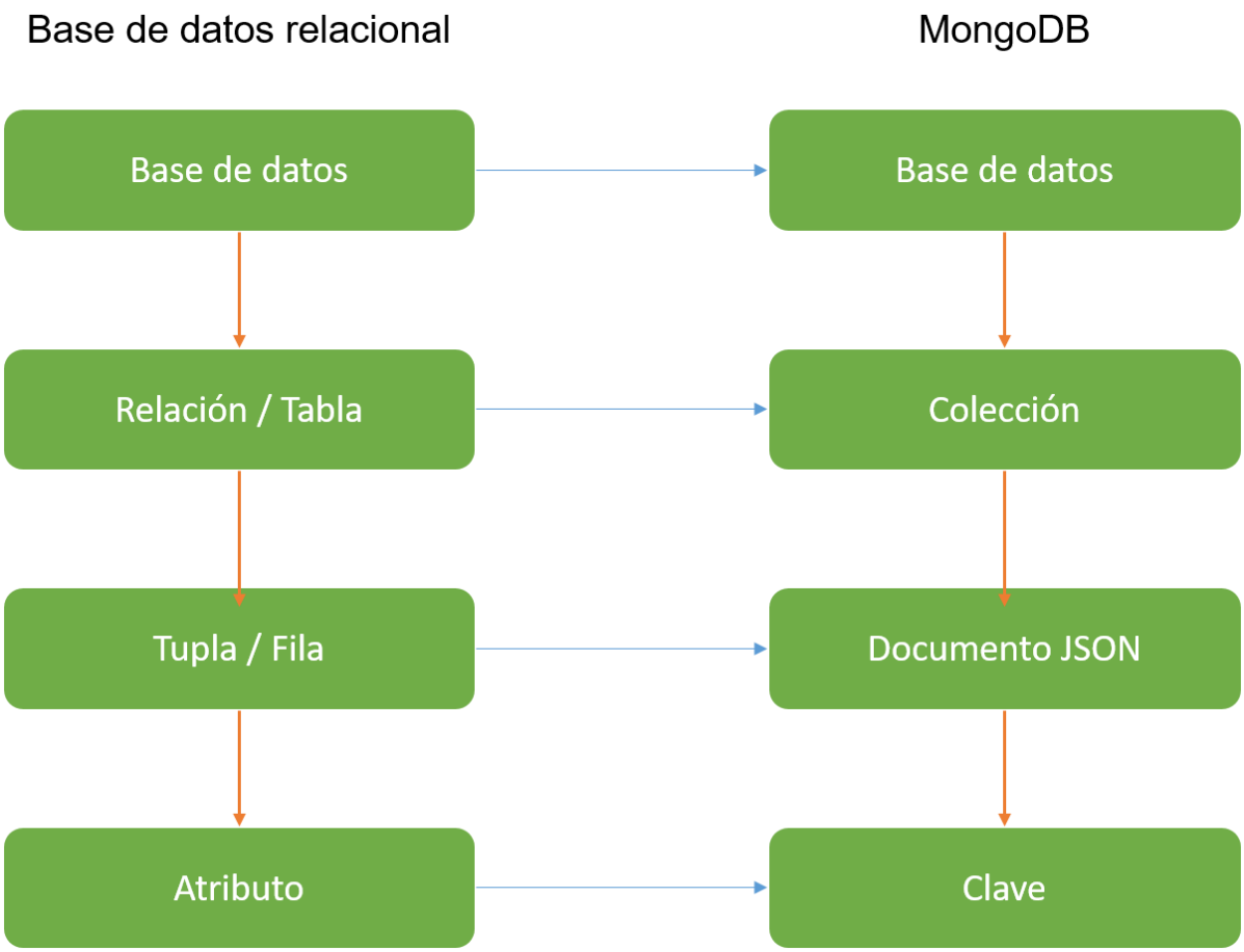
A. Elementos básicos de MongoDB

“MongoDB es un SGBD NoSQL de tipo documento, el cual usa documentos JSON” [34], JSON es un formato para el intercambio de datos análogo a XML, sin embargo, su composición es mucho más sencillo.

Un ejemplo de esta estructura es la siguiente:

```
{
        “id”: 1036,
        “nombre”: “Pablo”,
        “edad”: 24,
        “semestre”: 6,
        “dep”: 1
}
```

MongoDB utiliza los datos JSON para agruparlos en una colección, la cual es equivalente a una relación en un sistema de gestión de bases de datos relacional. En la siguiente grafica se establece un paralelo entre un sistema SQL y un NoSQL.



"Comparación de términos entre una base de datos relacional y una base de datos orientada a documentos: MongoDB" [35].

En la siguiente tabla se presenta la forma general de las sentencias básicas de definición del esquema de datos en SQL y en MongoDB:

OPERACIÓN	SQL	MONGODB
Creación base de datos	CREATE DATABASE nombreBD;	USE nombreBD
Creación tabla (en SQL) / Colección (en MongoDB)	CREATE TABLE nombreTabla (atributo1 tipo_de_dato restricciones <, atributo2…>);	db.createCollection(“nombreColección”)
Creación índice	CREATE INDEX nombreIndice ON nombreTabla (atributo1 <, atributo2… >);	db.nombreColeccion.ensureIndex({“atributo”: 1})
Destrucción tabla (en SQL) / Colección (en MongoDB)	DROP TABLE nombreTabla;	db.nombreColeccion.drop()

En la siguiente tabla se muestran ejemplos de estas sentencias. “La función ensureIndex() de MongoDB recibe como parámetro una pareja “clave”: valor. La clave indica el atributo que se indexará, y el valor puede ser 1 o -1” [35]. Donde el 1 indica que el orden es ascendente, y el -1 que es descendente.

“Formas generales de las sentencias de inserción, actualización y borrado de datos en SQL y en MongoDB” [35]:

Operación	SQL	MongoDB
Inserción	INSERT INTO nombreTabla VALUES (valorAtributo1, <, valorAtributo2…>);	db.nombreColeccion.insert({ atributo1: valorAtributo1 <, atributo2: valorAtributo2, ...>})
Actualización	UPDATE nombreTabla SET atributo = nuevoValor <WHERE condición>	db.nombreColeccion.update({<condición>}, {$set: {atributo: nuevoValor} }<, [true|false],[true|false]>)
Borrado	DELETE FROM nombreTabla <WHERE condición>	db.nombreColeccion.remove({<condición>})

“Ejemplos de las sentencias de inserción, actualización y borrado de datos en SQL y en MongoDB” [35]:

SQL	MongoDB	Resultados en MongoDB
INSERT INTO dpto VALUES(10, 'Ventas'); INSERT INTO empleado VALUES(1, 'Juan', 23, 8, 10);	db.dpto.insert({_id: 10, nom: "Ventas"}) db.empleado.insert({_id: 1, nombre: "Juan", edad: 23, grado: 8, dep: 10})	Los datos son insertados.
UPDATE empleado SET grado = 9 WHERE edad = 25;	db.empleado.update({edad:25}, {$set: {grado: 9}})	{"_id": 1, "nombre": "Juan", "edad": 23, "grado": 8, "dep": 10} {"_id": 2, "nombre": "Pablo", "edad": 25, "grado":9, "dep": 10} {"_id": 3, "nombre": "Ana", "edad": 11, "grado": 6, "dep": 20}
DELETE FROM empleado WHERE edad >= 18;	db.empleado.remove({edad: {$gt:18}})	{"_id": 3, "nombre": "Ana", "edad": 11, "grado": 6, "dep": 20}

Se puede visualizar que en la inserción de datos en MongoDB, se muestra el comportamiento de una clave foránea. Sin embargo, hay otra alternativa para insertar dos piezas en un solo documento JSON:

db.dpto.insert({

dep: 10, nom: "Ventas",

empleados: [{_id: 1, nombre: "Pablo", edad: 24, grado: 8, dep: 10}]

})

"La forma esencial de una consulta en SQL es:

SELECT <DISTINCT> [*atributo1 <, atributo2...>]

FROM relación1 <, relación2...>

WHERE condición

GROUP BY atributos

HAVING condición de grupo;

Y en MongoDB es:

db.collection.find(<condición> <, proyección>)

Donde “db” es la palabra clave para hacer operaciones sobre la base de datos actual, y “collection” indica la colección a consultar.

La función “find()” recibe dos parámetros: una condición y una proyección. “La condición es equivalente a la cláusula WHERE de SQL pero aplicada a los documentos” [35]

El flujo es equivalente a la sentencia “SELECT” de SQL, en la cual se especifican los atributos o claves que se desean incluir en el caso de MongoDB.

B. La reunión en MongoDB mediante MapReduce

MongoDB no posee un operador propio (nativo) para reunir (Join) dos colecciones. Como “MongoDB es un SGBD NoSQL y en este tipo de BD suele haber cierto tipo de redundancia (debido a la poca normalización) esto hace usualmente innecesaria la ejecución de operaciones como la reunión” [36].

Los vectores son otra alternativa y pueden hacer innecesarias las mencionadas reuniones. Sin embargo, si se requiere hacer la reunión de dos colecciones, “esta se puede lograr en MongoDB mediante MapReduce” [36].

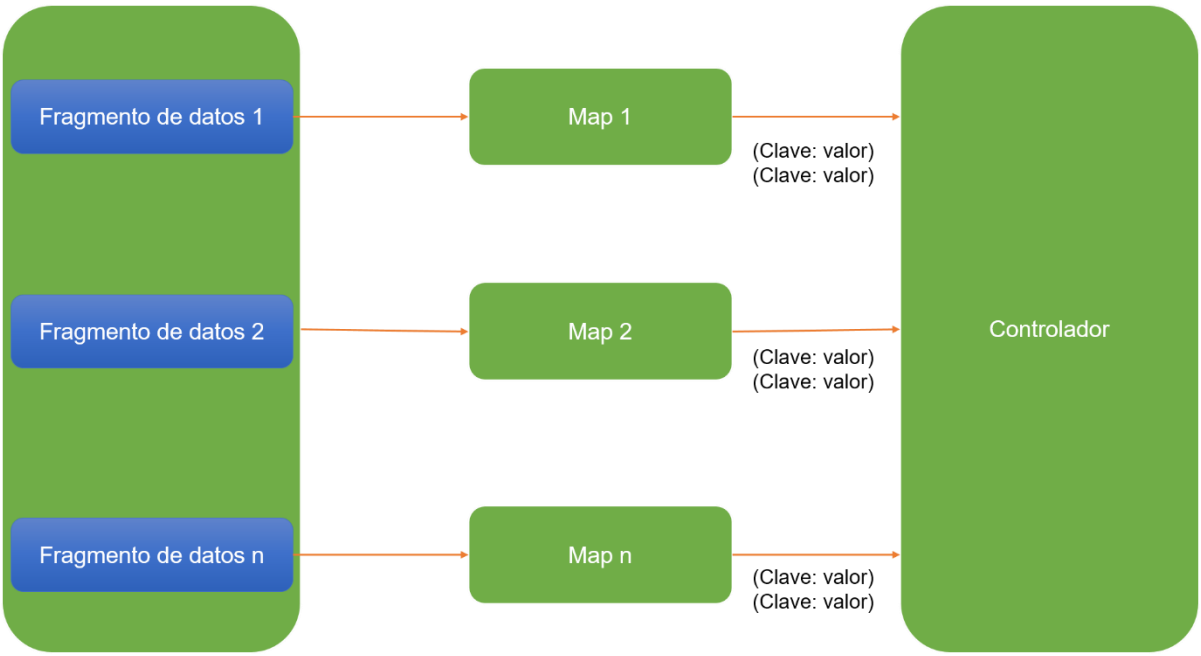
“MapReduce es un paradigma de programación paralela en arquitecturas distribuidas cuyo objetivo es la solución de problemas mediante algoritmos que exploten el paralelismo” [37].

Dicho paradigma es usado en la actualidad en aplicaciones que manejan un volumen muy alto de información.

Adicionalmente, tiene en esencia dos divisiones: “Map” y “Reduce”.

En la función Map, se aplica a un conjunto de datos y retorna una lista de parejas (clave: valor).

El conjunto nativo de datos se particiona en muchos fragmentos, cada fragmento se asigna a un procesador donde se ejecuta la función Map. “Como se muestra en la siguiente figura. Luego, la lista de parejas generada por cada función Map se envía a un proceso controlador” [37].



"El proceso controlador une todos los valores de una misma clave en un arreglo, i.e., se genera un conjunto de parejas (clave: arreglo de valores)" [37]. Este conjunto de duplas se particionan en muchos subconjuntos, cada subconjunto de duplas se asigna a un procesador donde se ejecuta la función Reduce.

En la función Reduce, "se aplica a cada pareja (clave: arreglo de valores) de cada subconjunto de parejas, y a partir de esta se genera una o varias parejas de la forma (clave: valor), véase en la siguiente figura" [37]:

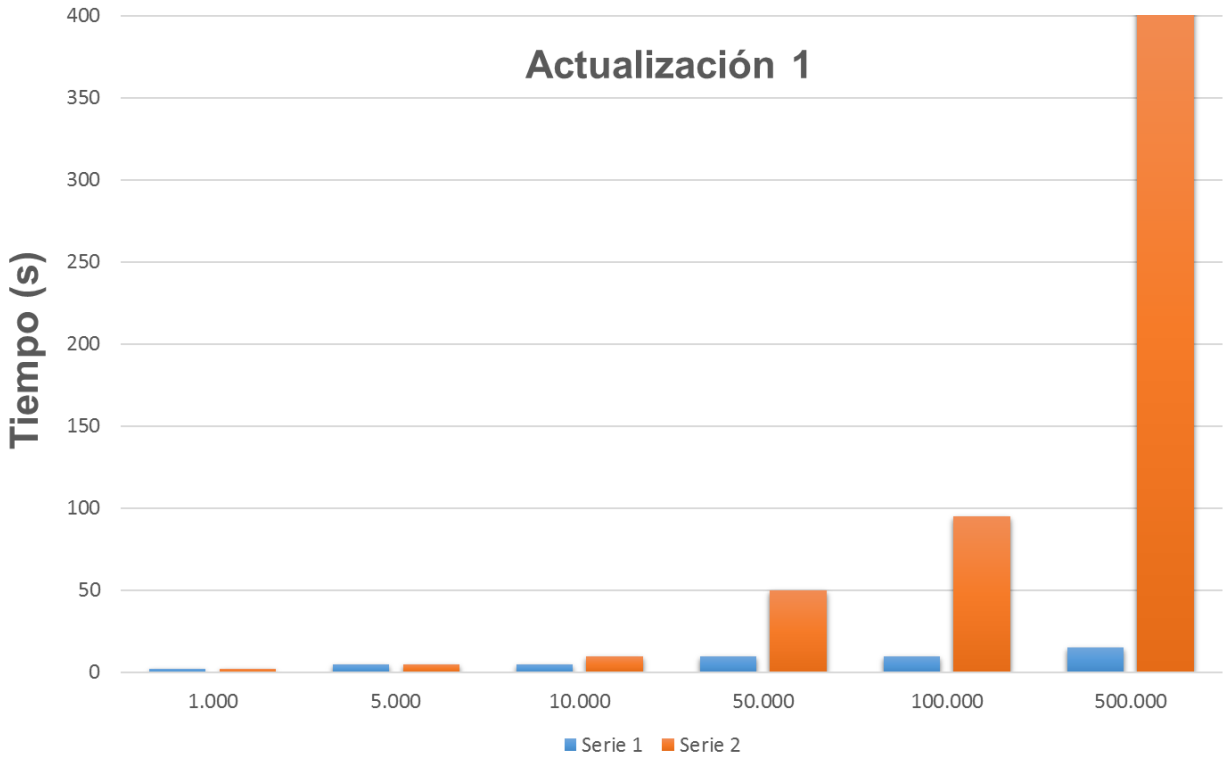
C. Experimentos y resultados

En la siguiente experimentación realizada, se usaron dos sistemas de gestión de bases de datos muy reconocidos actualmente (Uno SQL y otro NoSQL) como lo son MongoDB y Oracle; se pudo evidenciar el potencial de ambos en el procesamiento de datos a escala de motores del tipo documental (Para el caso de MongoDB). Para las pruebas se usó un equipo de cómputo con las siguientes características: Intel® Core (TM) i7 de 1.8 GHz y con 4 GB de memoria RAM. Se usó la versión 11 Express Edition de Oracle y la versión 3.2.4 de MongoDB. El sistema operativo seleccionado fue Windows 8. Para tratar de garantizar en lo posible igualdad de condiciones en la plataforma de pruebas, y ajustar la exactitud posible en los

resultados, se verificó según los sitios oficiales tanto de Oracle (www.oracle.com) como de MongoDB (www.mongodb.org), que este sistema operativo es uno de los recomendados para el desempeño óptimo de estos productos. “Las pruebas se hicieron para las cuatro operaciones: inserción, actualización, consulta y borrado. También se hicieron pruebas para la operación de reunión” [36].

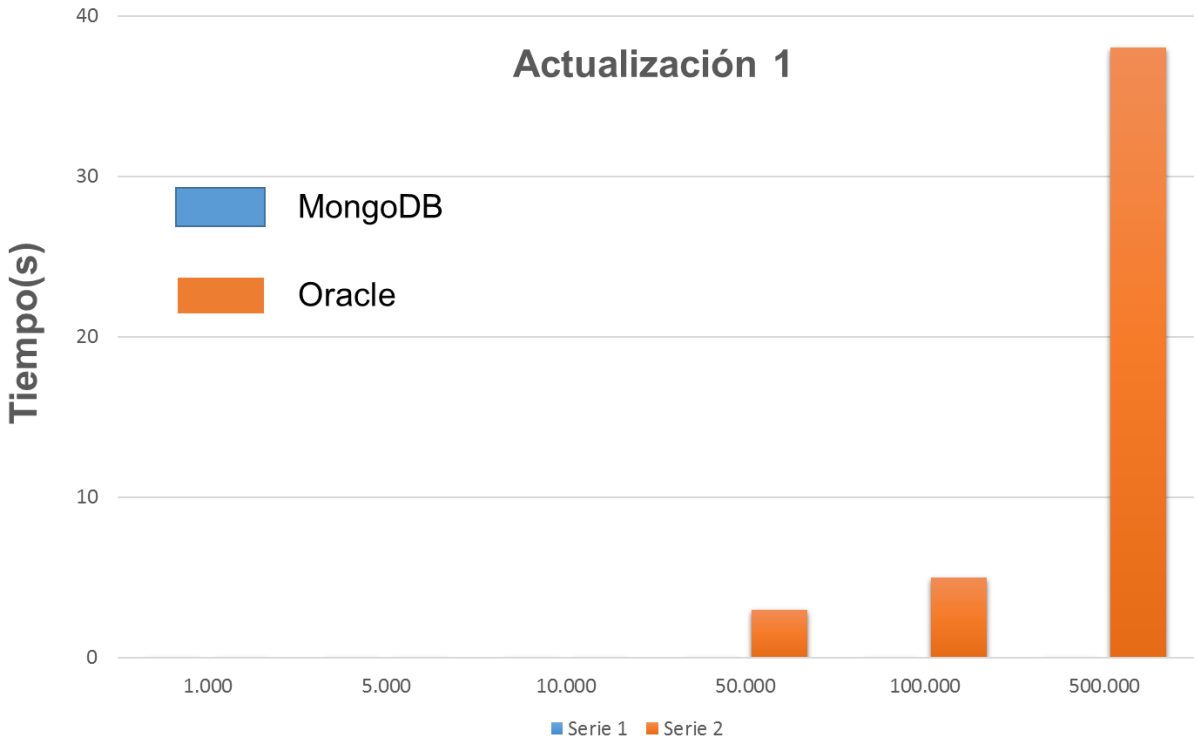
Cada test se ejecutó diez veces para cada tamaño de muestra elegido. Los tamaños fueron 1.000, 5.000, 10.000, 50.000, 100.000 y 500.000 filas y documentos; y se tomó el promedio del tiempo de ejecución para cada tamaño dado. “Se consideró, además, que cuando una misma consulta se ejecuta repetidamente en un SGBD, algunos de sus resultados pueden permanecer en la memoria caché” [36], esto ocasiona que el tiempo de ejecución de las últimas pruebas sea menor con respecto a las primeras. “Para evitar que los resultados se afectasen por este aspecto, se “limpió” la memoria caché antes de la ejecución de cada consulta” [36].

Resultados de las pruebas de actualización: caso 1



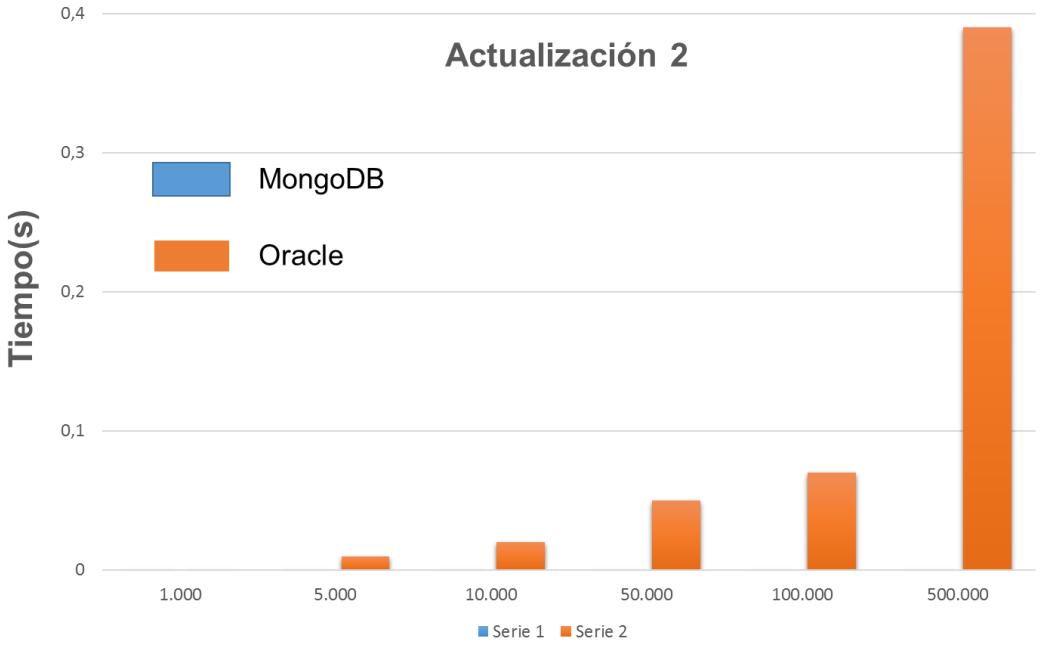
“Los resultados favorecieron a MongoDB. El mayor número de verificaciones de consistencia y de integridad (claves primarias, foráneas, checks de no nulidad, entre otras) y la gestión de transacciones podría explicar el mayor tiempo requerido por Oracle” [38].

Para la operación de actualización se evaluaron tres casos donde la operación afectaba (ver las siguientes figuras): 1) a todas las filas/documentos, 2) a un 20% de las filas / documentos y 3) a 10 de las filas / documentos:



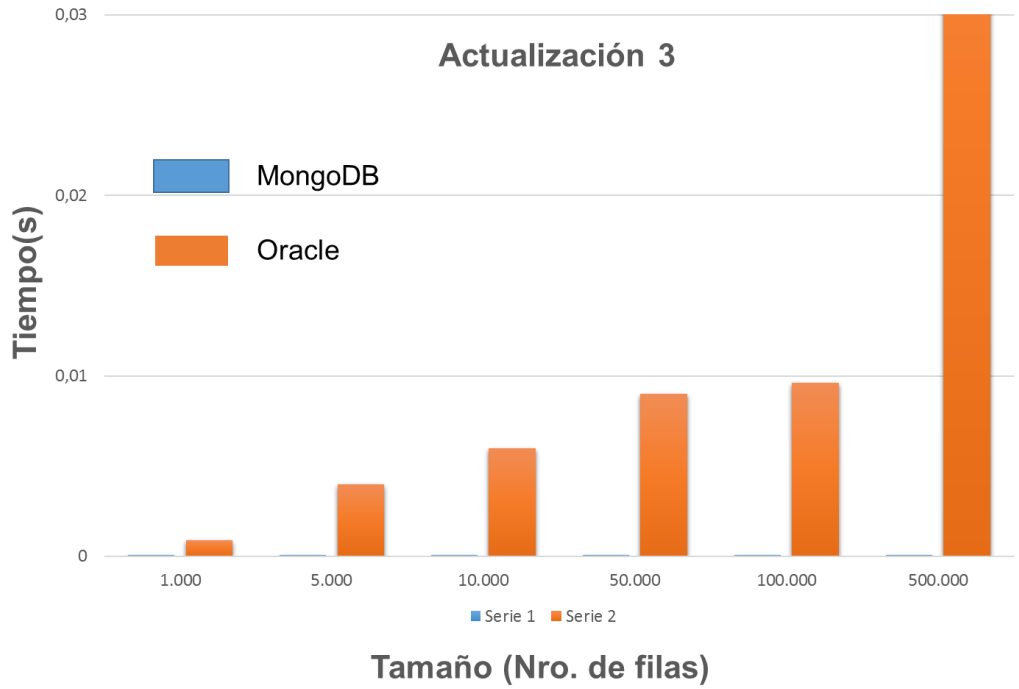
“Resultados de las pruebas de actualización: caso 1. Los resultados favorecieron de nuevo a MongoDB. De hecho, los tiempos para los seis tamaños de muestra fueron prácticamente los mismos” [38]. El mayor tiempo registrado por la base de datos ejecutada fue la correspondiente a Oracle, posiblemente se debe a la misma razón expuesta para los resultados de la operación de inserción.

En la siguiente figura, los resultados en MongoDB fueron similares a los del caso anterior. Aunque el tiempo registrado por Oracle fue mayor en todas las muestras, este mejoró con respecto al caso anterior.



Resultados de las pruebas de actualización: caso 2

“Esto se evidenció aún más en el siguiente caso, ver siguiente figura, aunque los tiempos de MongoDB siguieron siendo mejores” [38].



Resultados de las pruebas de actualización: caso 3

“Esta mejora en Oracle se puede deber al uso de índice sobre la clave primaria, el cual facilita la localización de las filas a actualizar en el segundo y tercer caso, y evita una exploración completa de la tabla (table access full) como se evidencia en los planes de ejecución obtenidos” [38].

Description	Object name	Cost	Cardinality	Bytes
UPDATE STATEMENT, GOAL ...		3	1000	12000
UPDATE	EMPLEADO			
TABLE ACCESS FULL	EMPLEADO	3	1000	12000

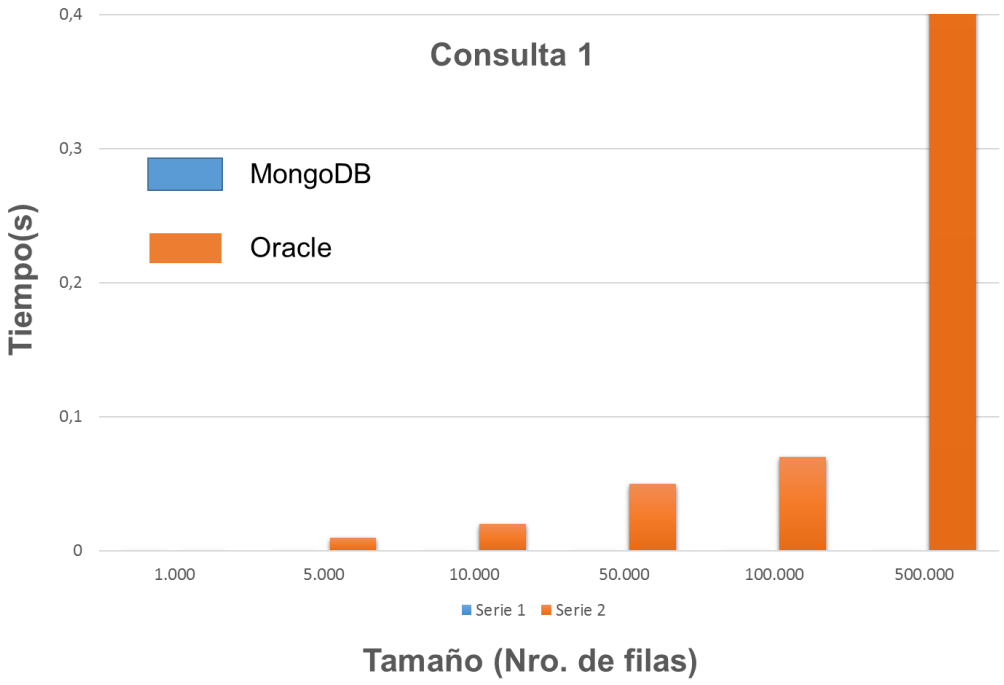
a)

Description	Object name	Cost	Cardinality	Bytes
UPDATE STATEMENT, GOAL ...		2	200	5000
UPDATE	EMPLEADO			
INDEX RANGE SCAN	SYS_C007949	2	200	5000

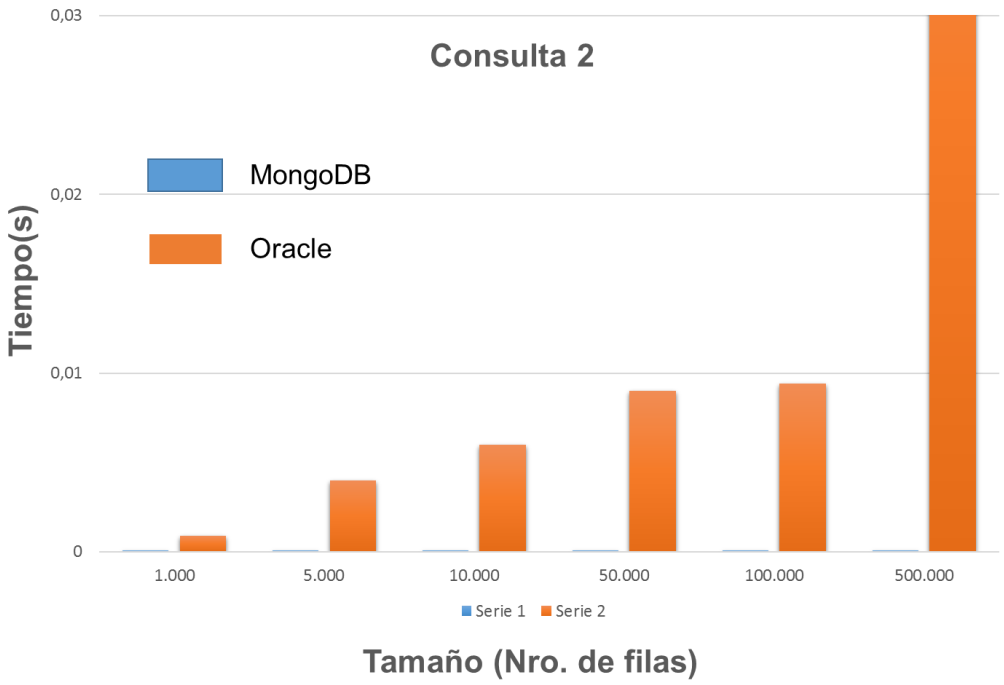
b)

“Planes de ejecución en Oracle para la actualización (tabla empleado con 1000 filas): a) de todas las filas y b) del 20% de las filas” [38].

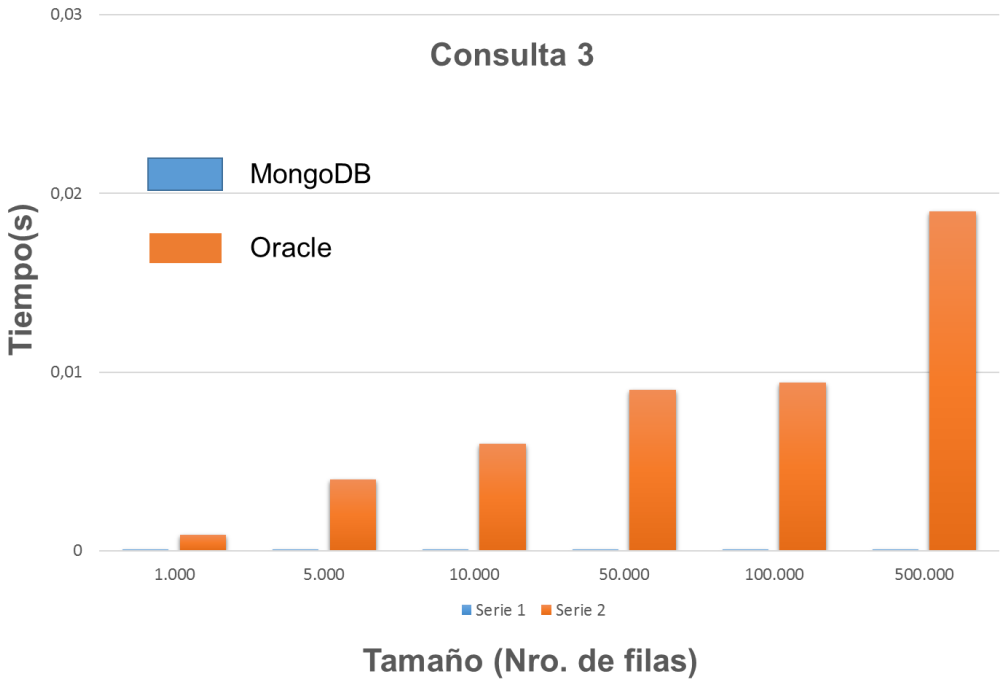
Para la operación de consulta se evaluaron tres casos donde la operación recuperaba (ver los siguientes gráficos): 1) una única fila/documento, 2) 20% de las filas/documentos, y 3) 10 de las filas/ documentos.



Resultados de las pruebas de consulta: caso 1



Resultados de las pruebas de consulta: caso 2



Resultados de las pruebas de consulta: caso 3

D. Conclusiones

En los tres casos de las consultas los resultados se mantuvieron prácticamente iguales en cada SGBD. De hecho, todas las consultas requirieron menos de un segundo. Aunque los resultados favorecieron a MongoDB, las diferencias fueron menores en comparación con la operación de inserción y con el caso 1 de actualización.

En conclusión, aunque se requieren experimentos más exhaustivos y muchos otros tipos de pruebas, los tiempos registrados para las operaciones de inserción, actualización, borrado y consultas sobre una sola relación / colección favorecieron a MongoDB. “Esto se debe posiblemente al mayor número de verificaciones de consistencia y de integridad, y a la gestión de transacciones realizadas por Oracle. Sin embargo, para operaciones de consulta binarias, i.e., que involucran dos relaciones/colecciones como la reunión, el tiempo de respuesta favoreció a Oracle” [38]. Esto se debe posiblemente

a los costos implicados para la ejecución de la reunión por medio de la programación adicional requerida (MapReduce) y los costos de comunicación entre los nodos (equipos) de la red.

"Debido a que el rendimiento de la programación en paralelo, como en el caso de MapReduce, depende del número de procesadores y de los costos de comunicación entre estos, un trabajo futuro es determinar si con un mayor número de procesadores los resultados de la operación de reunión siguen favoreciendo a Oracle" [38]. De igual forma, hay muchos puntos para retomar esta prueba y una de ellas es usando "MapReduce" en diferentes SGBD NoSQL y el de otras operaciones binarias.

2. Bases de Datos tipo Columnas

A. Elementos básicos de Cassandra

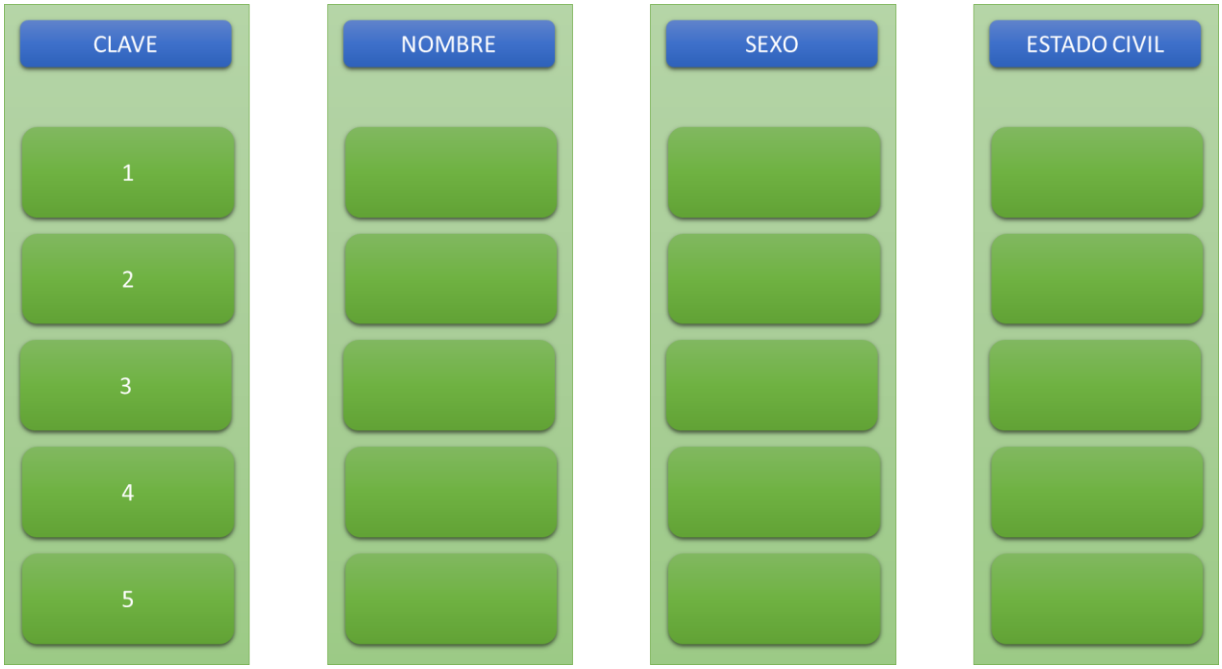
"Cassandra es un proyecto de Apache Software Foundation" [39]. Es un SGBD distribuida enfocada a una arquitectura orientada a columnas. "Inicialmente fue desarrollado por Facebook para ser un híbrido entre BigTable de Google" [40] y sistema de dínamo de Amazon [41].

En su composición primaria cuenta con dos modelos estructurales:

"Keyspace − > Column Family − > Row − > Column − > Value

"Keyspace − > Super Column Family − > Row − > Super Column − > Column − > Value" [42]

Actualmente, es usada por empresas que utilizan grandes volúmenes de datos para realizar tareas de cálculos, sugerencias y demás transacciones para el público en general. CERN, Comcast, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Reddit, The Weather Channel y más de 1500 empresas más que tienen acceso a dicho SGBD [43].



Escritura de una base de datos orientada a columnas [44]

Su estructura nativa de escritura, es muy semejante a la usada por motores SQL tradicionales, con la diferencia que usa documentos JSON para la inclusión de inserción de contenidos.

A continuación, se muestran ejemplos CRUD usando Cassandra para su respectiva ejecución [43]:

Create:

```
CREATE TABLE users (
   username text PRIMARY KEY,
   firstname text,
   lastname text,
   birth_year int,
   country text
)
```

```
CREATE INDEX ON users(birth_year);
```

Insert:

```
insert_statement ::=  INSERT INTO table_name ( names_values | json_clause )
                      [ IF NOT EXISTS ]
                      [ USING update_parameter ( AND update_parameter )* ]
names_values     ::=  names VALUES tuple_literal
json_clause      ::=  JSON string [ DEFAULT ( NULL | UNSET ) ]
names            ::=  '(' column_name ( ',' column_name )* ')'Update:
```

Update:

```
update_statement ::=  UPDATE table_name
                      [ USING update_parameter ( AND update_parameter )* ]
                      SET assignment ( ',' assignment )*
                      WHERE where_clause
                      [ IF ( EXISTS | condition ( AND condition )*) ]
update_parameter ::=  ( TIMESTAMP | TTL ) ( integer | bind_marker )
assignment       ::=  simple_selection '=' term
                    | column_name '=' column_name ( '+' | '-' ) term
                    | column_name '=' list_literal '+' column_name
simple_selection ::=  column_name
                    | column_name '[' term ']'
                    | column_name '.' `field_name
condition        ::=  simple_selection operator term
```

Delete:

```
delete_statement ::=  DELETE [ simple_selection ( ',' simple_selection ) ]
                      FROM table_name
                      [ USING update_parameter ( AND update_parameter )* ]
                      WHERE where_clause
                      [ IF ( EXISTS | condition ( AND condition )*) ]
```

B. La reunión en Cassandra mediante MapReduce

Una propiedad notable que es observada en las bases de datos orientadas a columnas y que además son intercedidas por medio del sistema "MapReduce", es que son totalmente independientes de las demás filas de datos. Esto significa que, para cualquier fila en la salida de datos, no requiere la existencia de cualquier otra forma de datos con el fin de transmitir toda su información sin ninguna inconsistencia en tiempo de ejecución.

Esta observación simplifica el problema inmenso, en el sentido de que la escala horizontalmente los datos ya no es un problema (Para los casos NoSQL). "Esto puede hacerse por ejemplo separando los datos al azar a través de un número arbitrario de los nodos" [45]. Cuando se procesa una transacción, se envía a cada nodo, y entonces el resultado final se obtiene tomando la reunión del resultado de cada nodo. Este modelo es posible independiente del SGBD y por ende simplifica los diseños de los sistemas de producción y los test que se realicen.

C. Experimentación y resultados

Para realizar la respectiva comparación entre SGBD NoSQL orientadas a columnas, que en este caso se escogió a Cassandra, se toman en cuenta (como fue en el caso de la comparación de MongoDB y Oracle) sus características y que ambos sistemas tengan propósitos similares para realizar el respectivo comparativo.

Por ahora, no hay existe una clasificación oficial para este tipo de software, aunque existen diferentes teorías, las cuales se han hablado al principio del documento. Para este caso, se

usará una teoría presentada por Stefan Edlich en una de sus papers [46], donde habla de las siguientes categorías:

a. "Núcleos en sistemas NoSQL: La mayoría de ellos creados como sistemas de componentes de servicios Web 2.0" [46] .

b. "Sistema de Gestión de base de datos: Actualmente existen diferentes familias, como las siguientes: Hadoop / HBase, Cassandra, Hypertable, Cloudata, Amazon SimpleDB, SciDB" [46].

El proyecto Apache Cassandra desarrolla una gran generación distribuida, "base de datos escalables, diseños completamente distribuidos del dínamo y de modelo de datos basado en ColumnFamily de Bigtable." [43]

Como un elemento de referencia, se utiliza MySQL como SGBD SQL (también opensource) para ver lo que se pierde y lo que es ganado por el uso de soluciones NoSQL.

Para este comparativo, se basó en diferentes referencias y pruebas realizadas por diferentes entidades. En las cuales tenían en común los siguientes criterios: 1. Persistencia, 2. Replicación, 3. Alta disponibilidad, 4. Transacciones, 5. Conciencia, 6. Lenguajes de implementación, 7. Influencias, 8. Tipo de licencia (La solución dual de licencias ya está disponible para MySQL, es el resultado de la serie de adquisiciones de los últimos años por parte de Sun) [11].

Los resultados se muestran la siguiente tabla [47]:

Feat	Cassandra	MySQL
1	SI	Sí (con otro tipo de conexión que el típico)
2	SI	SI
3	Distribuidos	Distribuido, disponible con MySQL Cluster
4	Eventualmente consistente	Consistente consistente (full realmente ácido)
5	Sí (heredado de Hadoop)	Sí (heredado de Hadoop) sí (con MySQL Cluster)
6	Java	ANSI C / ANSI C++
7	Dynamo y BigTable, Facebook/Digg/ Rackspace	Oracle
8	Apache 2.0	GPL+FLOSS / Propietario

Cuadro comparativo con las características de los tres productos seleccionados [47]

Mediciones de tamaño de instalaciones comunes

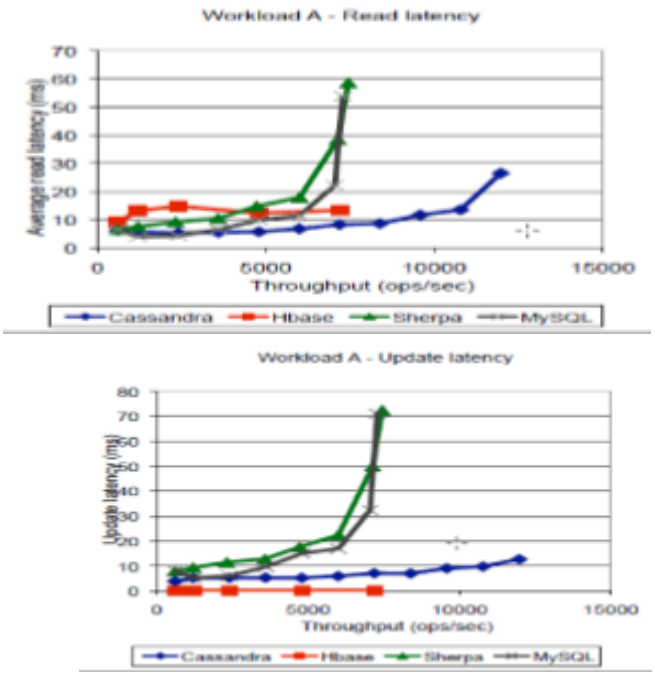
Aunque las bases de datos SQL han tenido representación por mucho tiempo en la historia de los motores de bases de datos, los SGBD NoSQL superan por mucho el almacenamiento de datos. “Es conocimiento común que las mayores instalaciones de MySQL no pueden ser más grandes, vamos a decir, 1 millón de registros de tamaño medio sin almacenamiento en caché de memoria y extendido sharding” [47]; más que limitar la inserción y procesamiento de información, el uso de SGBD SQL para estos temas se ha convertido en un dolor de cabeza tanto para los desarrolladores como a los usuarios finales a los cuales se dirigen las aplicaciones [48].

No hay ninguna unidad de medida oficial para el tamaño de una instalación de bases de datos, pero podemos tomar varios factores en cuenta: “Número de registros / filas /

documentos almacenados" [43], "diversas fuentes dan tamaños de 2 a 150 millones de registros para instalaciones diversas de Casandra" [43]; "algunos casos muestran tamaños máximos para instalaciones de 150 TB para Cassandra" [47].

Rendimiento en un ambiente intensivo de escritura

El número de escrituras es igual a las que se lee, el rendimiento alcanzado se aprecia en las siguientes figuras [47]:



En las 7000 transacciones realizadas (Lectura y escritura), se pudo visualizar que MySQL tiene un tiempo de latencia muy alto para una aplicación que corre en tiempo real; La gráfica indica el tiempo de ejecucación y aunque Cassandra tuvo mejores resultados que MySQL indica que el rendimiento de escritura de Cassandra puede mejorarse utilizando un disco de registro [49].

D. Conclusiones

Aunque MySQL y Cassandra tienen algunas características comunes sus comportamientos no son similares en tiempo de ejecución y procesamiento. Esto sugiere que no se pueden ligar o complementar para resolver cualquier tipo de problema.

Sin embargo, aunque MySQL demotro tener un tiempo de latencia demaciado alto al procesar grandes volumenes de datos, es una buena alternativa si los requierimientos puntuales del negocio no requieren tener este flujo de datos en curso.

De igual forma, las bases de datos enfocadas en columnas, suelen ser una segunda opción a las documentales; estas últimas son mucho mas eficaces y demuestran mayor eficiencia en procesamiento aunque tienen igual comportamiento en el uso de recursos.

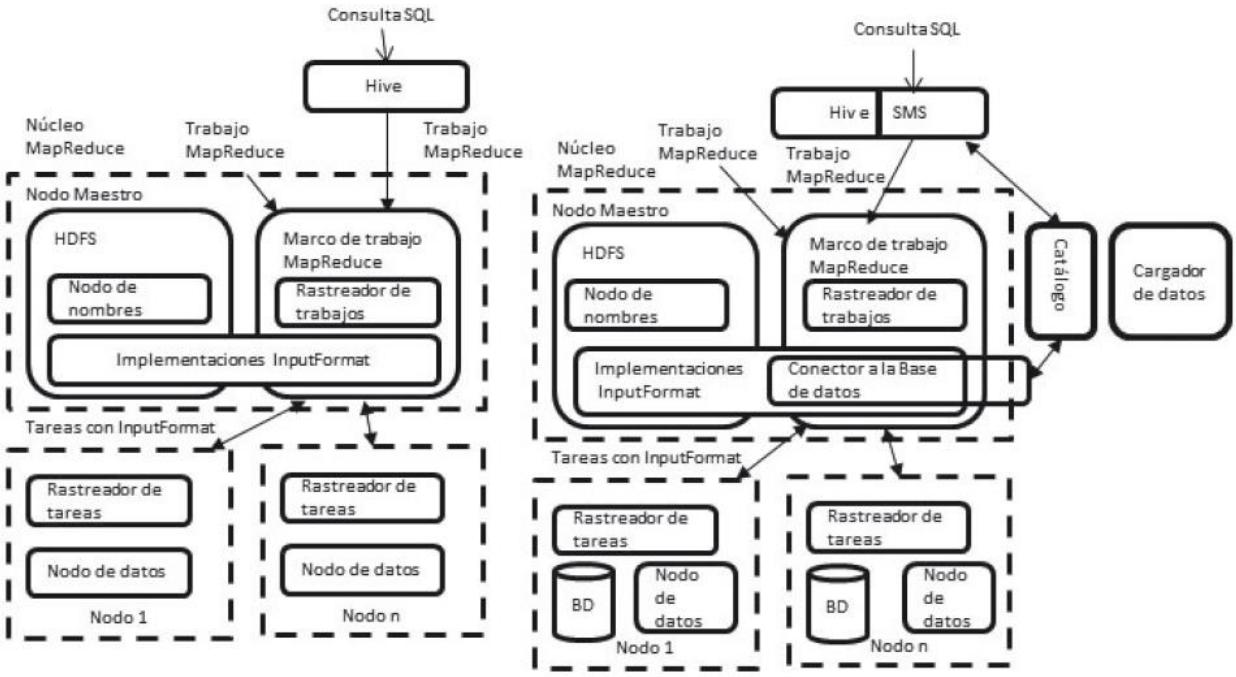
3. Bases de datos tipo Grafo

A. Elementos basicos de HadoopDB

Este sistema de gestion de bases de datos NoSQL pertenece a la categoria del tipo grafo, se ha caracterizado por ser uno de los mas utilizados en la insdustria por sus capacidades de analisis y por ser parte del Apache Fundation, lo que quiere decir que es de codigo abiero. Dicho sistema facilita el almacenamiento de información y proporciona caracteristicas que incluyen el procesamiento de consultas complejas en bases de datos existentes.

Los comienzos de Hadoop provienen desde el año 2004, donde Google comenzó a describir instructivos donde se describian tecnicas y mecanismos para el manejo de grandes volumenes de datos, reduciendo los problemas existentes en sistemas tradicionales de bases de datos. Sin embargo, dicha plataforma no fue explotada hasta el año 2008. Empresas como Yahoo (quien fue pionero con dicha heramienta), Facebook, Twitter o eBay usaron este SGBD.

HadoopDB contiene Postgres como parte de la capa de bases de datos en cada nodo, Hadoop / MapReduce como capa de comunicación para regular los nodos y Hive [50] como capa de interpretación. “La combinación de todos estos elementos da como resultado una base de datos paralela shared-nothing, en la cual es posible interactuar usando un lenguaje de tipo SQL” [51].



"Arquitectura de Hadoop (izquierda) comparada con arquitectura de HadoopDB (derecha)" [52].

B. La reunión en HadoopDB mediante MapReduce

HadoopDB extiende Hadoop para proveer 4 componentes [52]:

Catálogo: Contiene información acerca de las bases de datos. Dentro de estos parámetros de conexión y metadatos tales así como conjuntos de datos contenidos en el clúster, localización de las réplicas y propiedades de particionamiento de datos. "La información del catálogo es almacenada en un archivo XML en el HDFS" [53].

El cargador de datos: Particiona los datos e incorpora carga paralela para estos en los sistemas de gestiòn de bases de datos. El cargador de datos consta de dos componentes fundamentales: el hasher global y hasher local. "El primero de ellos ejecuta un trabajo MapReduce que se encarga de leer los datos del archivo almacenado en el HDFS y a continuación los particiona en tantos pedazos como número de nodos haya en el clúster"

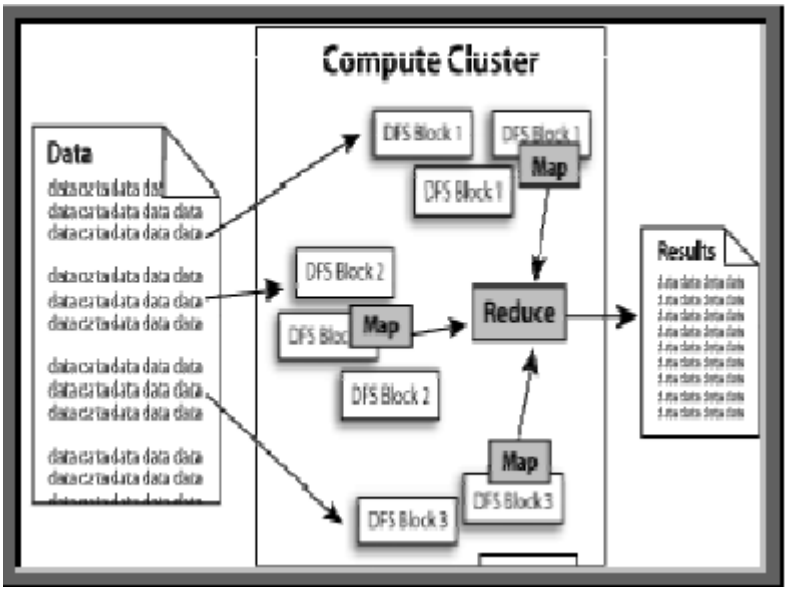
[53]. El hasher local reproduce cada partición desde el “HDFS” al sistema de archivos local de cada nodo y luego de esto divide el documento en pedazos de tamaño más diminutos, cada uno de ellos cerca de 1 GB. Estos datos son importados en las bases de datos locales.

Planificador SMS (SQL a MapReduce a SQL): Es una base de datos paralela que extiende Hive para suministrar una vista de consultas SQL. “Hive es un sistema de almacén de datos que facilita el análisis de grandes conjuntos de datos almacenados en sistemas de archivos compatibles con Hadoop” [54]. Hive suministra un módulo para realizar las reuniones a la base de datos por medio de una sintaxis similar a SQL, tambien conocido como “HiveQL”. Esta sintaxis resiste un subconjunto de transacciones SQL, entre ellas “selección, proyección, join, agregación y unión” [53]. No tolera borrado y actualización de filas en las respectivas tablas. “Las transacciones que son realizadas en HiveQL son compiladas en trabajos MapReduce” [55]. Hive posee un portafolio del sistema, tambien conocido como “Hive-Metastore”, el cual almacena metricas y esquemas, requeridos para la optimización de consultas y mineria de datos.

El conector a la base de datos: Es la vista que conecta sistemas de gestión de bases de datos independientes ubicados en los nodos del clúster y Hadoop.

C. Experimentación y resultados

Para la respectiva evaluación, se usó una base de datos SQL con información comercial almacenada desde al año 2003 y se realizarán diferentes tipos de consulta para generar reportes. Para la comparativa se usará HadoopDB, en esencia por su facilidad en manejo de grandesde volumenes de datos y por contener un Framework que ejecuta aplicaciones en grandes clústeres de hardware dedicado.



Generalidades Hadoop [56]

Implementación de la solución

Para la respectiva prueba y fase de analisis, se tomaron en cuenta 6 parametros:

1. Instalación del sistema operativo: Se realiza la instalación del sistema operativo CentOS 6.0.

2. Configuración de Hadoop: Luego de realizar las respectivas instalaciones y completar la configuracion de las demas aplicaciones (JAVA, Hadoop y Hive), se procede a la creación de los usuarios y roles dentro del SGBD.

3. Instalación de parámetros especiales: Para utlizar la base de datos existente, se procede a ejecutar diferentes parametros dentro del SGBD por medio de archivos planos que contienen los respectivos parámetros:

```
  <property>
     <name>dfs.replication</name>
     <value>1</value>
   </property>
  <property>
     <name>dfs.name.dir</name>
     <value>/home/hadoop/dfs/name</value>
   </property>
   <property>
     <name>dfs.data.dir</name>
     <value>/home/hadoop/dfs/data</value>
  </property>
```

```
    <property>
       <name>fs.default.name</name>
       <value>hdfs://FIEC-
SOFTWARE00:8020</value>
    </property>
```

```
 <property>
     <name>mapred.job.tracker</name>
     <value>FIEC-SOFTWARE00:8021</value>
</property>
<property>
     <name>mapred.system.dir</name>
<value>/home/hadoop/mapred/system</value>
</property>
```

```
 -  $HADOOP_HOME/bin/hadoop fs -mkdir /tmp
 -  $HADOOP_HOME/bin/hadoop fs -mkdir
/user/hive/warehouse
```

Configuración de prealistamiento Hadoop [56]

4. Replicar la información para los análisis de Hive: Para proceder con la importación de tablas desde la base de datos de origen (SQL), es necesario crear el ambiente en Hive.

```
CREATE TABLE sci_kardex (
cod_empresa int,
cod_local string,
nom_inventario string,
val_precio_distribuidor int,
por_desc_politica int,
cod_politica_descto string)
  ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
 STORED AS TEXTFILE;
```

Y finalmente se procede a cargar la información en Hive: [56]

```
hive> LOAD DATA LOCAL INPATH
'./bases/sci_kardex.txt'
    OVERWRITE INTO TABLE sci_kardex;
```

5. Ejecución de Queris requeridos: Se ejecutan los respectivos queris en Hive, teniendo en cuenta la información final requerida.

6. Visualización de las consultas en gráficos: Se procede a presentar las metricas generadas por el sistema al momento de ejecutar la secuencia de consultas. Esto permite visualizar en tiempo real, la cantidad de recursos consumidos por el respectivo SGBD.

Luego de implementar estos 6 parámetros, se proceden a realizar las pruebas en tiempo real.

	Nodos	Servidor 1	Servidor 2	Servidor 3 (PC)
CPU	Core i3 2,27Ghz	Xeon 2,4Ghz	Xeon 3,06Ghz	Core i3 2,2Ghz
Memoria	4GB	12GB	2GB	4GB
HDD	400GB	1,2TB Raid 5	80GB Raid 5	500GB

Caracteristicas de hardware en ambientes de pruebas [56]

	Hardware	Software	Inversión
Nodos	$450 c/u	$0 (CentOS, Hadoop, Hive)	$1800
Servidor1	$6366	$923 (Windows. Server)	$8597.99
Servidor2	$3500	$1095.68 (SQL Server)	$5731.99
Servidor3 (pc)	$460	$32.31 (CAL Windows) $181 (CAL SQL Server) *Total:* $2231.99	$2691.99

Costos aproximados (Dolares) para la implemetación de cada ambiente [56]

Nodo/ Query	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
1	60	326	561	223
2	52	237	477	156
3	37	202	384	134
4	35	193	318	124

Tiempos de respuesta de los queris ejecutados por Hive [56]

Servidor/ Query	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
Servidor 1	5	11	33	25
Servidor 2	67	101	371	182
Servidor 3 (pc)	70	271	529	168

Tiempos de respuesta para queris usando SQL Server [56]

	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
HIVE 4 nodos	35	193	318	124
Servidor 1	5	11	33	25
Servidor 2	67	101	371	182
Servidor 3 (pc)	70	271	529	168

Tiempos de respuesta para queris usando Hive vs SQL Server [56]

	Ventas x Sucursal	Ventas x Negocio	Ventas Full	Rentabilidad
HIVE 4 nodos	35	193	318	124
Servidor 1	5	11	33	25
Servidor 2	67	101	371	182
Servidor 3 (pc)	70	271	529	168

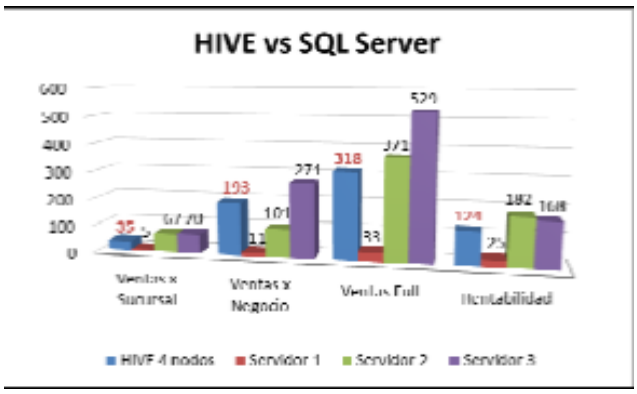
Tiempos de respuesta de los queris Hive vs SQL Server 2008 [56]

Análisis de los resultados

Con los resultados obtenidos a partir de esta prueba, podemos observar que a mayor numero de nodos, se optiene menor numero de respuesta al ejecutar las respectivas consultas.

Ejecución de Queris en diferentes nodos [56]

Al realizar la comparativa entre ambos sistemas (SQL y NoSQL) se observan mejoras en temas de configuración de servidores:



D. Conclusiones

La ejecución de transacciones sobre HadoopDB es de cierta manera muy superior frente a las realizadas sobre SQL (Usando arquitectura WIndows). Sin embargo, esto se debe en gran medida a que HadoopDB usando Hive sobre una plataforma Linux, soporta mucho mejor los queris y optimiza los nodos de los servidores.

Esto no arroja un resultado contundente, ya que Hadoop tiene un gran fuerte en el uso de caracteres espaciales y en este caso se usó una base de datos comercial que no explotaba estas caracteristicas.

El tipo de prueba realizada no fue muy exigente, ya que en primera medida estaba dirigida al comportamiento del hardware y de la cantidad de consumo. Ambos SGBD trabajaron con suficientes recursos, los cuales no tuvieron percanses a la hora de realizar sus respectivas transacciones.

De algna manera, la estructura SQL estaba preparada para garantizar el buen funcionamiento de la base de datos, a costa de generar lentitud a la hora de generar los reportes. Sin embargo, HadoopDB optimizó esta cuentión, solo que fue necesario preparar todo un ecosistema para su ejecución, aunque en producción esto implicaría una reducción en costos como se observa en las tablas, lo cual es muy psoitivo para la undiad de negocio que lo requiere.

Actualmente no se ha realizado una exploración muy profunda frente a este tipo de bases de datos, aunque grandes empresas como Facebook, Twitter o Ebay lo usan, no se ha demostrado en producción, realizando ejecución de bases de datos generales su real funcionamiento en consumo y transacción de datos.

4. Bases de datos tipo Clave / Valor

A. Elementos basicos de Redis

Redis es un sistema de gestión de bases de datos gestionado en memoria, y su arquitectura está basada en almacenamiento clave / valor o también conocido como “Haches”; tambien es usada como un SGBD persistente. Su composición está basada en ANSI C y es software libre.

Reputacionalmente es reconocida por su alta persistencia y por tener un control muy superior a otras bases de datos sobre volumenes muy altos de datos.

Redis, pone a disposición múltiples tipos de datos , que en muchos casos son de alta complejidad, “como listas, pilas, colas, conjuntos, conjuntos ordenados y operaciones atómicas definidas para esos tipos de datos” [57]. También cuenta con una base de datos en memoria pero consistente en disco duro, lo cual da una ventaja frente a otros sistemas como “memcached” que la perdida de energía de la máquina (Reinicio, desconexión u otros factores) expone la pérdida de información.

Dicho SGBD es usada comunmente en diferentes entornos, los cuales particularmente son: Caché de páginas web, Almacenamiento de sesiones de usuario, Almacenamiento de carritos de la compra, Caché de base de datos, Contadores y estadísticas, Listas de elementos recientes y Base de datos principal [58].

Algunos proveedores de herramientas WEB tienen consignada su información en este sistema, algunos de ellos son Twitter, Hulu, Pinterest, Flicker, Trello entre otros.

A continuación, se muestra una estructura general para las bases de datos clave / valor.

Estructura de una entrada de una base de datos clave-valor. [44]

B. La reunión en Redis mediante MapReduce

Hasta el momento, es dificil realizar procedimientos mediante MapReduce utilizando Redis. Sin embargo, existen otros mecanismos como por ejemplo “Jedis”. “Jedis es un cliente Java mediante el cual se puede leer / escribir datos desde / hacia Redis” [59].

En el siguiente ejemplo se observa la integración de Redis por medio de Jedis:

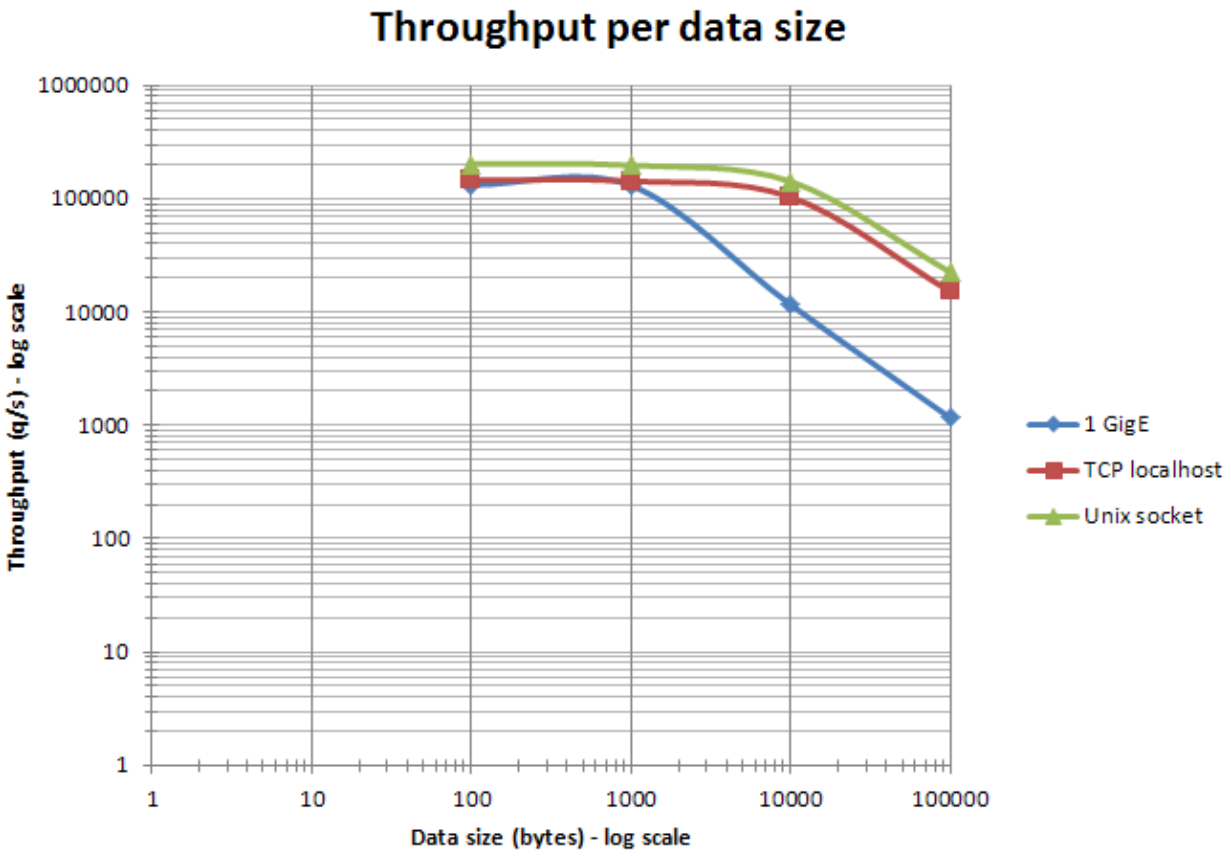
```
package com.redis.manager;
/*
 * Author: radhey
 * Date: 21-Dec-2013
 * Last Date of Modification: 21-Dec-2013 1:36:37 PM
 import redis.clients.jedis.Jedis;
 import redis.clients.jedis.JedisPool;
 import redis.clients.jedis.JedisPoolConfig;
public class RedisManager {
 private static final RedisManager instance = new RedisManager();
 private static JedisPool pool;
 private RedisManager() {}
 public final static RedisManager getInstance() {
 return instance;
 }
 public void connect() {
 // Create and set a JedisPoolConfig
 JedisPoolConfig poolConfig = new JedisPoolConfig();
 // Maximum active connections to Redis instance
 poolConfig.setMaxActive(20);
 // Tests whether connection is dead when connection
 // retrieval method is called
 poolConfig.setTestOnBorrow(true);
 /* Some extra configuration */
 // Tests whether connection is dead when returning a
 // connection to the pool
 poolConfig.setTestOnReturn(true);
 // Number of connections to Redis that just sit there
 // and do nothing
 poolConfig.setMaxIdle(5);
```

```
// Minimum number of idle connections to Redis
// These can be seen as always open and ready to serve
poolConfig.setMinIdle(1);
// Tests whether connections are dead during idle periods
poolConfig.setTestWhileIdle(true);
// Maximum number of connections to test in each idle check
poolConfig.setNumTestsPerEvictionRun(10);
// Idle connection checking period
poolConfig.setTimeBetweenEvictionRunsMillis(60000);
// Create the jedisPool
pool = new JedisPool(poolConfig, "localhost", 6379);
}
public void release() {
pool.destroy();
}
public Jedis getJedis() {
return pool.getResource();
}
public void returnJedis(Jedis jedis) {
pool.returnResource(jedis);
}
```

C. Experimentación y resultados

A pesar de ser una alternativa muy completa a la hora de seleccionar una herramienta para el manejo y uso de la información, en la actualidad carece de comparativos y pruebas en ambientes simulados. Sin embargo, si se cuenta con información referente a rendimiento en términos de arquitectura (Hardware).

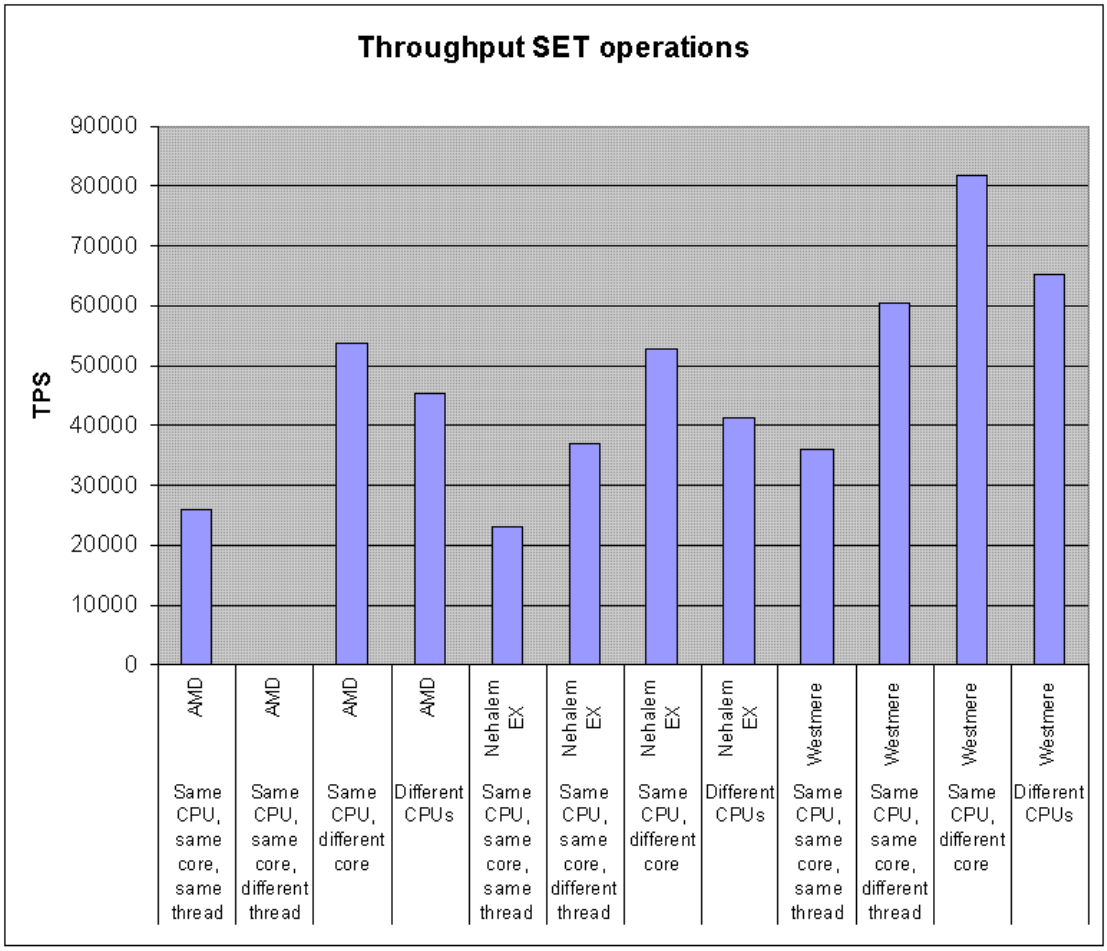
En el siguiente gráfico, se muestra el rendimiento frente a los canales de conexión, mostrando el alto rendimiento en la transferencia de información, independientemente de los puntos de flujo (10 bytes, 100 bytes o 1000 bytes) para el caso de una conexión Ethernet.



En esta gráfica se muestra el rendimiento por tamaño de datos [60].

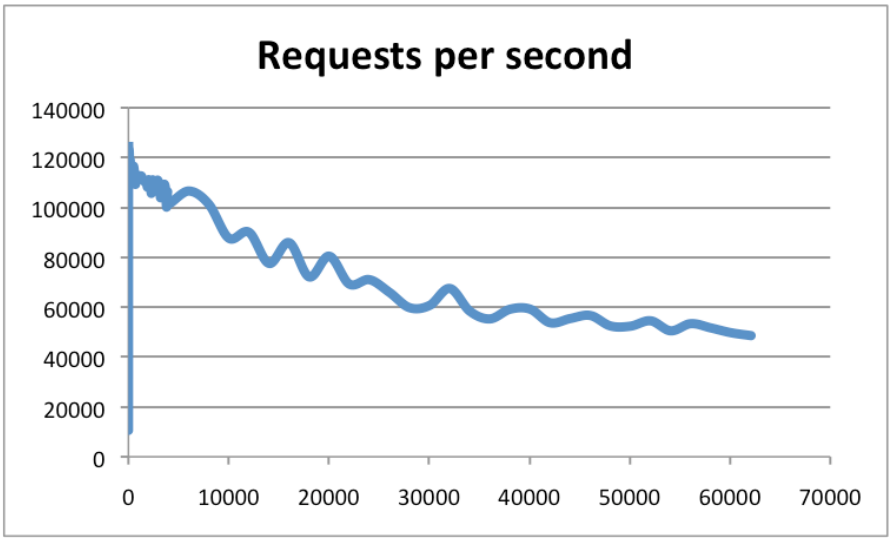
En el caso de los servidores, el rendimiento de Redis pasa a depender de la configuración y la ubicación de la NUMA de procesos [60].

En el siguiente apartado se observan algunos resultados de 4 KB, “punto de referencia establecidos para CPUs AMD 3 servidores (Estambul, Intel Nehalem EX, e Intel Westmere) con diferentes ubicaciones relativas” [60].



"Redis ya se ha evaluado en más de 60.000 conexiones, y todavía era capaz de sostener 50000 q / s en estas condiciones" [60].

Ahora se muestra un ejemplo de rendimiento de una instancia de Redis por cantidad de conexiones:



La prueba se realizó con 50 clientes simultáneos que realizan 2 millones de solicitudes en tiempo real. Redis en su versión 2.6.14 fue utilizada en todas las pruebas y adicionalmente fue ejecutado mediante la interfaz de bucle de retorno. La prueba fue ejecutada utilizando un espacio clave de 1 millón de llaves [60].

D. Conclusiones

Redis como SGBD NoSQL orientado a clave / valor, representa una alternativa muy eficiente si se requiere un sistema de procesamiento rápido y sobre todo optimo con recursos de hardware reducido. Sin embargo, falta indagar mucho en esta herramienta, ya que a pesar de ser libre y tener un respaldo por empresas que realizan transacciones robustas no se cuenta con la información necesaria para determinar en qué ambientes realmente es fuerte y en cuales debe ser suplido por otra herramienta.

Su distribución es muy similar a los sistemas que actualmente conocemos, maneja un lenguaje simple y una sintaxis muy partículas, como se mencionó, a pesar de no tener compatibilidad con MapReduce, cuenta con un homologo a través de Java, el cual le permite realizar transacciones de forma óptima y consultas de forma rápida y eficiente.

Los resultados fueron inconclusos, por lo cual no se puede tener un punto de comparación con los demás SGBD que fueron nombrados y que mostraron características en ejecución por medio de transacciones, ejecuciones e inyección directa de grandes volúmenes de información.

Otras implementaciones de bases de datos NoSQL

La computación en la nube o Cloud Computing, hace parte del despliegue de los sistemas de gestión de bases de datos NoSQL. Este paradigma se basa la ejecución de sistemas de información en la nube. Para el funcionamiento del mismo, se requiere la dispersión de información basados en datos obtenidos en bases de datos en la Nube; para esta tarea es muy optimo el uso de SGBD tipo columna, por la forma en que se distribuyen e inyectan los datos a los sistemas de información a petición. El trabajo "CDPort: A Portability Framework for NoSQL Datastores" [87].

En la actualidad existen múltiples sistemas de información basados en grandes volúmenes de datos, ya sea para la toma de decisiones o para la ejecución de tareas específicas. Esta el caso de las redes sociales, las cuales utilizan múltiples bases de datos, en especial NoSQL, para el almacenamiento y procesamiento de información en tiempo real (Gestión de cache, administración de entradas, administración de perfiles, entre otros). El trabajo "Designing Performance Monitoring Tool for NoSQL Cassandra Distributed Database", introduce al aprovechamiento de sistemas no estructurados para el monitoreo de aplicaciones usando bases de datos distribuidos por medio de Cassandra.

En el siguiente trabajo, se muestran ejemplos puntuales de gestiones usando basas de datos NoSQL para aplicaciones basadas en transacciones por internet. Además de la aplicación de sistemas QoS para la administración de prioridades en transacciones de Marketing Digital. Dichos sistemas son intermediados usando CassandraDB. "Managing Service Performance in the Cassandra Distributed Storage System" [89].

Una de las actividades más relevantes en un entorno TI, es la administración de la seguridad de la información y una de las ramas usadas para dicho sistema es por medio de sistemas basados en BackUp. NoSQL cumple con características que permiten el uso y aplicación de esta tarea y además ofrece la posibilidad de tener entornos gráficos usando SGBD basados en grafos para la fácil administración. "Remote Cloud Data Center Back up Using HBase and Cassandra with User-Friendly GUI" [90].

Proyectos en ejecución utilizando SGBD NoSQL

La tendencia NoSQL ha tenido mucha trascendencia en los últimos años, a tal punto que se ha convertido más que en una alternativa, en una herramienta indispensable para la ejecución y desarrollo de sistemas, los cuales tienen que ver con volúmenes muy altos de

información o empresas que requieren disminuir costos en arquitectura usando software libre.

En Colombia, dicha tecnología se ha tomado en cuenta, desde Universidades hasta propuestas que tienen que ver con la intervención de empresas y proyectos que actualmente se ejecutan.

A continuación, se presentará un modelo de proyecto usando bases de datos NoSQL orientada al sector de transporte, puntualmente, en el campo de la planificación de tráfico donde el volumen de datos para forjar la matriz O / D es muy alto.

Sin tomar en cuenta las bases de datos NoSQL, se puede optar por varias formas para generar dicha matriz, "la primera mediante la estimación matemática, la segunda de forma manual mediante el despliegue de personas y por último se busca que se realice de forma automática utilizando tecnologías avanzadas" [61]. Si observamos el sistema actual con el que funciona el transporte masivo en las ciudades principales, se puede notar que se realiza de forma manual. Sin embargo, y a pesar que esta forma ha sido usada desde siempre, no es realmente consistente ya que se generan errores y resultados malogrados, los cuales son un grave problema al momento de generar alternativas o soluciones a los problemas de movilidad actual.

Es en estos casos que surge como alternativa y como método de solución de problemas, las tendencias NoSQL las cuales representan un nuevo enfoque que aporta en la "gestión de la información para la generación de la matriz O/D que está siendo elaborada gracias a la utilización de tecnologías ITS avanzadas" [61].

Implementación de una base de datos NoSQL para la generación de la matriz O / D

El alcance del proyecto que se mencionará a continuación, analiza las diversas opciones de SGBD, para la implementación y ajuste de la base de datos más eficiente para la ITS enfocada a la generación de la matriz O / D.

La incorporación de tecnologías NoSQL puede traer ventajas al sector del transporte, "en función de esto es importante indicar algunos de los usuarios y/o stakeholders que puedan beneficiarse o utilizar estos sistemas en el país, estos pueden ser identificados en la tabla que se observa a continuación" [61].

No	Stakeholder	Descripción
1	ITS Colombia	Entidad sin ánimo de lucro que apoya la investigación tecnológica en tránsito, transporte y logística, su objetivo principal es fomentar y mejorar la eficiencia de los sistemas de movilidad y seguridad vial.[5]
2	Ministerio de Transporte de Colombia	Organismo Colombiano encargado de garantizar el desarrollo y el mejoramiento del transporte, tránsito y su infraestructura.
3	Organismos de Tránsito (OT)	Son las unidades administrativas municipales, distritales o departamentales que tienen por Ley, la función de organizar y dirigir lo relacionado con el tránsito y transporte en su respectiva jurisdicción[6]
4	Dirección de Tránsito y Transporte (DITRA)	Organización que debe velar por el control de las normas de tránsito y toso lo relacionado a él en aéreas transito urbano (Metropolitano y municipal) y carreteras.
5	Secretaría de movilidad	Organismo que formula, orienta, lidera y ejecuta las políticas del sector que garantizan mejores condiciones de movilidad en la ciudad
6	Empresas privadas de transporte	Empresas prestadoras del servicio de transporte público en Colombia

Stakeholders Del Proyecto [62]

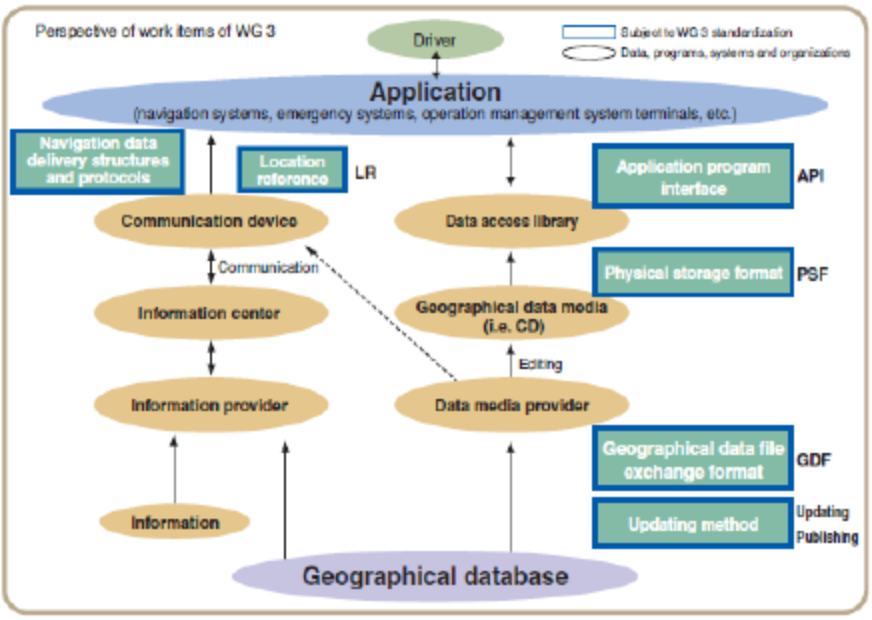
El sistema de transporte sugiere por sí mismo, múltiples medidas de mejora dentro de su ecosistema, ya que es una necesidad poblacional. Pero antes de iniciar cualquier implementación, es importante obtener información referente a necesidades, implicaciones, mejoras e innovación en el uso actual.

Algunas de esas aplicaciones podrían tratarse de prevención de accidentes, gestión de incidentes, notificaciones de señales de tránsito, gestión de tráfico, monitoreo de vías, información de localizaciones y ubicaciones, entretenimiento, entre otros [63].

El objetivo de llevar estas necesidades a una aplicación y que esta dependa de una base de datos NoSQL es que pueda generar un factor diferenciador, ofreciendo agilidad, facilidad al usuario en la visualización de alternativas y sugerencias (procesamiento de datos en tiempo real de forma personalizada), entre otros.

“El uso de tecnología para hacer del transporte vial una actividad más inteligente puede salvar vidas, ahorrar tiempo y dinero al disminuir la congestión, mejorar la seguridad y minimizar el consumo y las emisiones de combustible” [64].

A continuación, se observa una arquitectura estándar para la gestión de la información: [65]



Para el caso de este proyecto de investigación (Implementación de una base de datos NoSQL para la generación de la matriz O/D) se tienen en cuenta otros conceptos, tales como la tecnología Wifi.

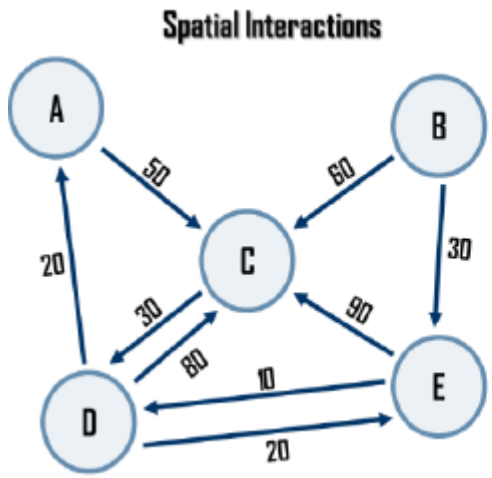
En este caso, el uso de la tecnología WIFI tiene gran relevancia por sus características principales, tales como la potencia del AccessPoint, la potencia del accesorio o dispositivo Wifi por el que nos conectamos y los obstáculos que la señal tenga que atravesar (muros o metal) [61]. En esencia, esto será usado para la generación de matriz origen destino [66].

Matriz origen/destino: La matriz de viaje Origen/destino (O/D), representa el número de usuarios con características similares de viajes que se desplazan entre zonas pares de transporte de la red de transporte, son un insumo fundamental para la planificación de la mayoría del transporte y los problemas de diseño [66].

La definición de matriz O/D, hace referencia al número de usuarios que realizan un desplazamiento entre un punto y otro, teniendo en cuenta que pueden retornar del punto final al punto inicial. Dichas matrices representan el flujo total en el desplazamiento y en todos los pares de orígenes y destinos de una red, “las matrices O/D se conocen también como matrices de tráfico” [67]. Su composición es bidimensional; las filas y columnas constituyen zonas de origen y destino, y el valor de sus elementos indica “el número de

viajes o desplazamientos realizados desde cada una de las zonas de origen a las de destino en un periodo de tiempo, tendrá tantas filas y columnas como zonas" [68].

A continuación, se muestra una gráfica que representa la construcción de una matriz O/D: [68]



O/D Matrix

	A	B	C	D	E	Ti
A	0	0	50	0	0	50
B	0	0	60	0	30	90
C	0	0	0	30	0	30
D	20	0	80	0	20	120
E	0	0	90	10	0	100
Tj	20	0	280	40	50	390

El fundamento del proyecto que se analiza en este momento, se basa en "la generación automática de la matriz O/D de una forma más eficiente" [61], ofreciendo una alternativa completa con soporte en la tecnología NoSQL.

Si bien se puede analizar en lo mencionado anteriormente, tanto las necesidades de la vida real, los modelos matemáticos que se requieren para la fabricación de la matriz y el uso de la tecnología, que para este caso se utilizó la NoSQL; se pueden inter relacional para construir una solución consistente en el área de transporte.

Para retomar el análisis de las bases de datos, se tomarán en cuenta los que mejor se acomodan al control de concurrencia y replicación de datos.

Concurrencia de los sistemas

En primera medida, será evaluada la concurrencia de los sistemas, la cual es altamente necesaria e importante debido a que esta misma es la que permite que diferentes sistemas se ejecuten al mismo tiempo y que potencialmente puedan interactuar entre sí.

Los siguientes SGBD soportan un alto volumen de concurrencia, permitiendo que todos los usuarios puedan leer y escribir los datos al mismo tiempo. Adicionalmente están orientadas al "clustering" y a tener nodos con replicas.

Como se podrá observar, muy pocos SGBD NoSQL bloquean las aplicaciones en el momento de presentarse una inconsistencia de concurrencia dentro de los sistemas, por lo cual proporcionan mayor disponibilidad en el momento de alto volumen de transacciones.

"En cuanto a los bloqueos óptimos, únicamente los motores basados en clave / valor proporcionan esta característica para controlar la concurrencia" [69]. El siguiente análisis tiene las siguientes convenciones: (-) No proporciona o (+) proporciona:

Database		Concurrency Control		
		Locks	*Optimistic Locking*	*MVCC*
Key Value Stores	*Voldemort*	-	+	-
	Redis	-	+	-
	Membase	-	+	-
Document Stores	*Riak*	-	-	+
	MongoDB	-	-	-
	CouchDB	-	-	+
Column Family Stores	*Cassandra*	-	-	-
	HBase	+	-	-
	Hypertable	-	-	+
Graph Databases	*Sesame*	+	-	-
	BigData	-	-	-
	Neo4J	+	-	-
	GraphDB	-	-	+
	FlockDB	-	-	-

Control de concurrencia en NoSQL [69]

Replica de datos

La segunda etapa es la medición de la réplica de datos. Esta propiedad es un método usado para propagar y aislar datos en un ambiente distribuido, con el objetivo de tener mejor performance y confianza, mediante la reducción de dependencias.

Muchos de los SGBD NoSQL utilizan la réplica y fragmentación de los datos y exploran la concordancia para potenciar la escalabilidad y la disponibilidad. A diferencia de los SGBD relacionales, "en este tipo de soluciones es posible añadir nuevos nodos para conseguir mayor capacidad. Esto se conoce como escalabilidad horizontal" [61]; esta propiedad es de gran relevancia dentro del almacenamiento de las bases de datos.

Database		Replication		
		Read from Replica	*Write to Replica*	*Consistency*
Key Value Stores	*Voldemort*	+	-	+/-
	Redis	+	-	-
	Membase	-	-	+
Document Stores	*Riak*	+	-	+/-
	MongoDB	+	-	+/-
	CouchDB	+	+	-
Column Family Stores	*Cassandra*	+	-	+/-
	HBase	-	-	+
	Hypertable	-	-	+
Graph Databases	*Sesame*	-	-	+
	BigData	+	-	+
	Neo4J	+	+	-
	GraphDB	-	-	+
	FlockDB	+	+	+/-

Replicación en NOSQL [70]

En esta tabla, se puede observar que solo 5 SGBD no proporcionan la lectura desde la réplica (Membase, Hbase, Hypertable, Sesame y GraphDB). “En cuanto a la escritura en la réplica muy pocos motores poseen esta característica, entre estos están CouchDB, Neo4J y FlockDB” [61].

La consistencia de los datos en la replicación es una peculiaridad muy trascendental que deben tener los SGBD, debido a que estos soportan seguridad al cliente. Muy pocos SGBD NoSQL carecen con esta característica, lo que los hace más endebles y poco manipulados en el mercado actual.

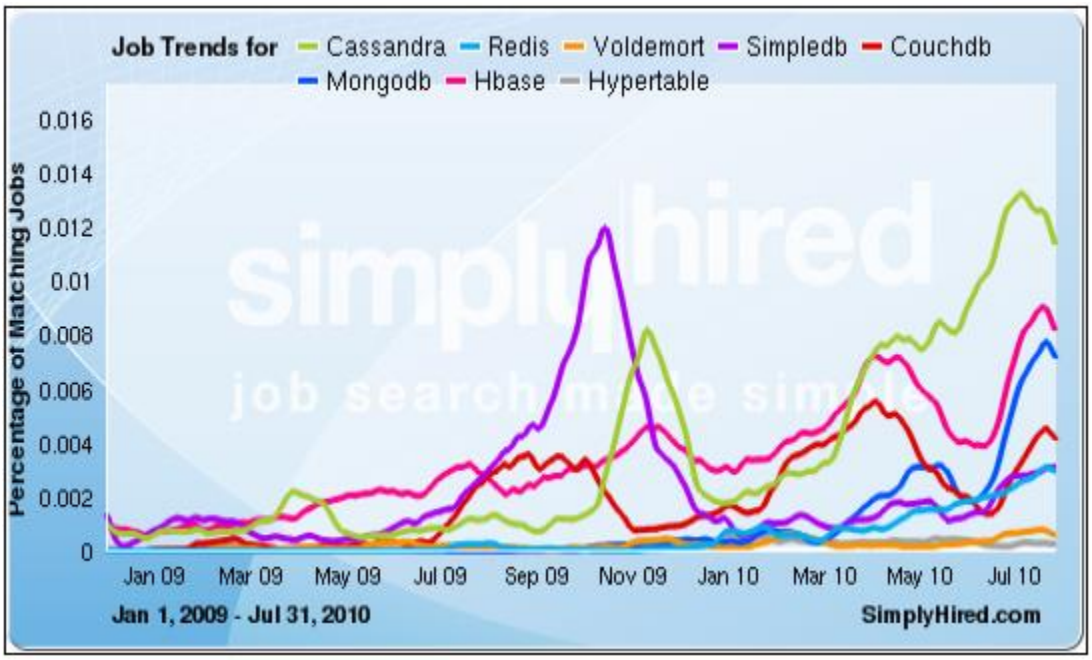
En la siguiente tabla, se muestran las bases de datos NoSQL con características de particionamiento: [71]

Database		Partitioning	
		Range based	*Consistent Hashing*
Key Value Stores	*Voldemort*	-	+
	Redis	-	+
	Membase	-	+
Document Stores	*Riak*	-	+
	MongoDB	+	-
	CouchDB	-	+
Column Family Stores	*Cassandra*	-	+
	HBase	+	-
	Hypertable	+	-
Graph Databases	*Sesame*	-	-
	BigData	-	+
	Neo4J	-	-
	GraphDB	-	-
	FlockDB	-	+

En el particionamiento se puede visualizar que los SGBD clave / valor y documentos como Riak y CouchDB se basan en consistencia “hash”, mientras que MongoDB, Hbase e Hypertable están basadas en rango de claves. Los motores basados en grafos tales como “Sesame, Neo4J y GraphDB”, no ofrecen ningún tipo de particionamiento, debido a que la partición de un grafo es inviable si carece de algoritmos complejos [61].

Dentro de estos SGBD, se encuentra consistencia a la partición en red, esta en sí, consiste en ofrecer al sistema distribuido operación continua, incluso ante posibles fallos en parte de la conectividad. Por ende, los almacenes de datos NoSQL usan normalmente particiones horizontales.

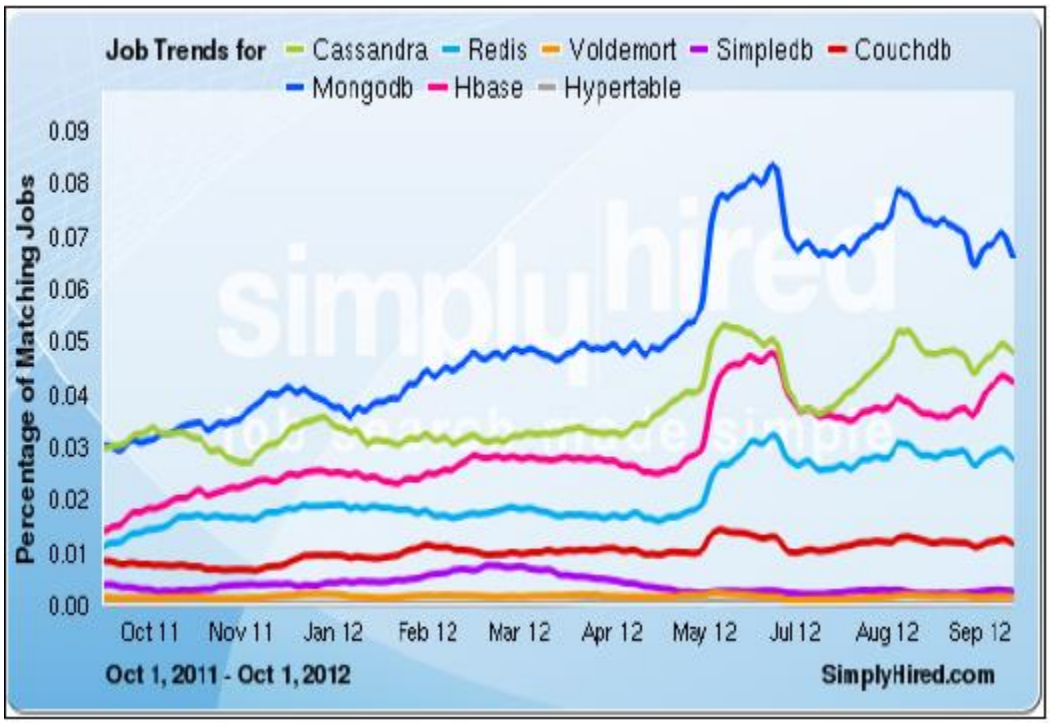
En el siguiente gráfico, se especificarán las bases de datos que se tomaron en cuenta para pruebas y desarrollos dentro del proyecto, mostrando su trascendencia en el tiempo [72]



CASSANDRA, REDIS, VOLDEMORT, SIMPLEDB, COUCHDB, MONGODB, HBASE, HYPERTABLE TRENDS 2009 – 2010

Los SGBD Simpledb y Cassandra han tenido un aumento muy significativo dentro de su rendimiento y demás características.

El siguiente gráfico, evalúa el potencial de búsqueda de algunos SGBD [73].



CASSANDRA, REDIS, VOLDEMORT, SIMPLEDB, COUCHDB, MONGODB, HBASE, HYPERTABLE JOB TRENDS 2011 – 2012

Los siguientes porcentajes muestran que Cassandra encabeza el grupo de SGBD relacionado con columna, luego Hbase cuya diferencia es mínima [72]:

Cassandra 61%

Redis 14,9%

Voldemort 3%

Simpledb 33%

Couchdb 36%

Mongodb 11,5%

Hbase 18,9%

Hypertable 8%

Descripción general del sistema

"Desde el punto de vista de desarrollo del sistema, se contemplan una serie de tecnologías tanto blandas como duras para el desarrollo del mismo" [61]. Para las tecnologías blandas, estas hacen referencia a prestaciones o servicios tecnológicos. En este caso, se utilizan herramientas basadas en tecnologías NoSQL, en cuanto a las tecnologías duras, se hace referencia a hardware que se usa, así como sensores que sirven para suministrar información que es el insumo básico para la generación de la matriz O / D.

Los sensores realizan la recolección de información dentro de la ciudad, utilizando señales inalámbricas. Actualmente se trabaja con dispositivos que permiten la recepción y almacenamiento de la señal Wifi, donde el proceso de recepción es el mismo utilizado para las señales Bluetooth tradicionales.

En la siguiente gráfica, se especifica un entorno para la recepción de información, captando señales emitidas por múltiples dispositivos de diferentes tipos, pasando por los sensores, proceso que mostrará automáticamente la matriz O / D.

Escenario de captura [61]

Arquitectura del sistema

La arquitectura del programa, se enfoca puntualmente en el SGBD NoSQL, en cual, analiza la clase de dispositivo y los resultados sobre la matriz O/D.

En esta investigación se analizarán modificaciones específicamente sobre los sistemas que tienen que ver con NoSQL y su manipulación dentro de los que se encuentran el subsistema de captación de señales y la construcción de una matriz O/D. “En cuanto a los demás subsistemas se realizó un cambio en el sistema operativo manejado, Linux específicamente Ubuntu 12.10 es el utilizado para todo el desarrollo” [61].

En el siguiente gráfico, se observa la información almacenada por los sensores, en donde se especifica por columnas la información que se tienen en cuenta. Esta consiste básicamente en fecha, hora y zona horaria donde se capturó la señal. Se adicionan los siguientes parámetros: “Timestamp, la MAC del dispositivo captado, además información en cuanto a la potencia que capta y finaliza con la clase del dispositivo” [61].

Collection Time	Timestamp	MAC Address	Strength	Type
Fri Mar 01 2013 18:20:04 GMT-0500 (COT)	1362180004.86	b8:c6:8e:17:ef:74	-61	Wifi

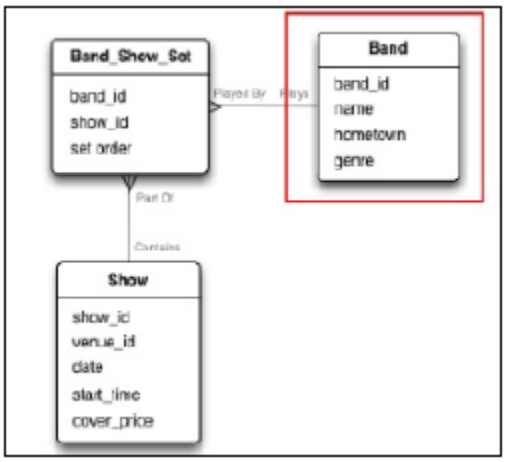
Información del dispositivo captado [61]

Para la generación de la matriz O / D, es indispensable primero que todo tener en cuenta que el usuario ingresa una “fecha y hora específica”, información relevante que involucra directamente las bases de datos.

Transformación de SQL a NoSQL

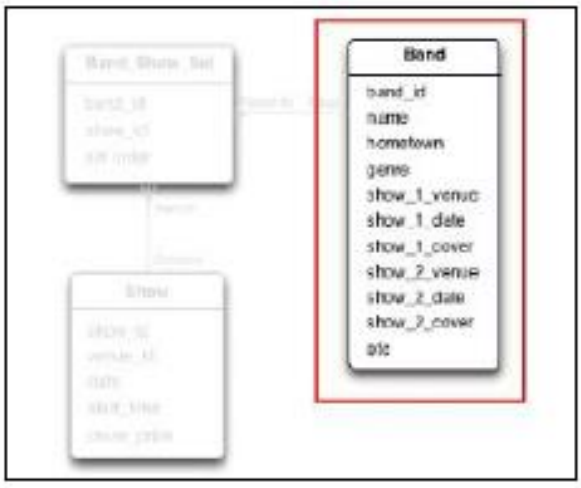
“Es importante tener en cuenta, que el proceso para transformar un modelo relacional a no relacional se llama des normalización” [74], este procedimiento es indispensable al momento de la implementación, ya que implica la optimización de las bases de datos que tienen implicación.

Para entender a fondo la transformación desde SQL, se presenta un ejemplo básico de un esquema de tres tablas en SQL:



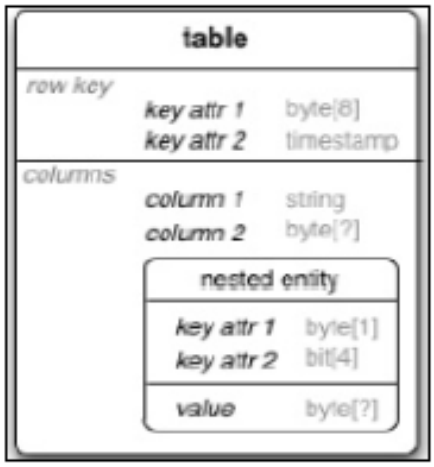
Modelo lógico SQL [74]

La transformación del esquema mencionado es posible mediante procedimientos estructurados; “seleccionando la tabla principal, es decir la que presenta mayor relevancia para en proceso de negocio, puesto que es convertida en la contenedora de todos los atributos de las tablas con las que tiene relación”, como se observa en la siguiente figura:



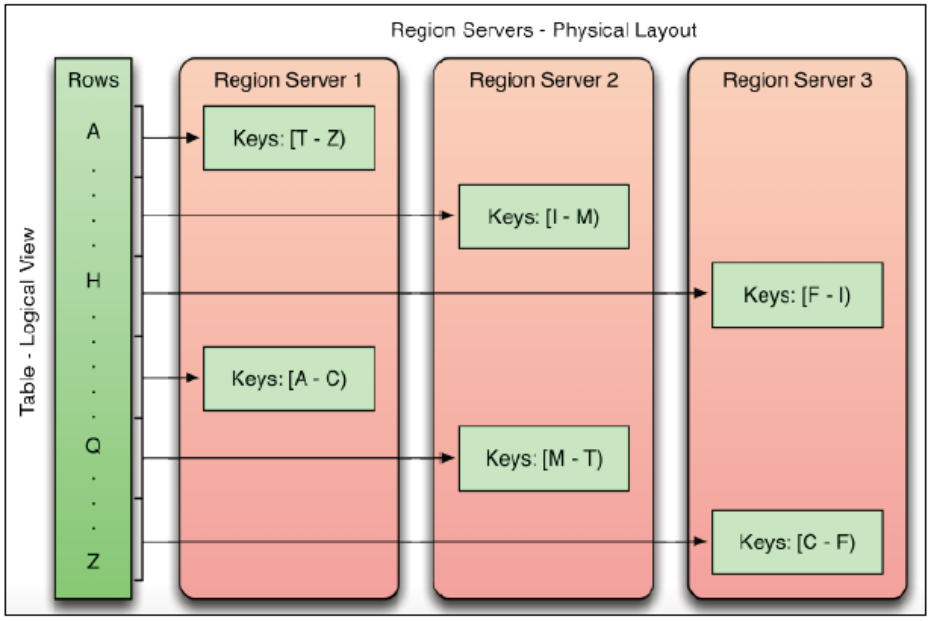
Modelo NoSQL sin anidación [74]

Este ejemplo puede presentar una transformación con “tablas anidadas”. Es muy particular este caso, sin embargo, puede observarse en ambientes orientados a la distribución de máquinas o clúster.



Modelo NoSQL con anidación [74]

En este modelo, la información se aloja por regiones y se contiene en servidores organizados por rangos de filas. En la siguiente gráfica se ejemplifica dicho modelo. "La región del servidor uno posee un rango de A hasta C y de T hasta Z y así sucesivamente como se muestra a continuación, además se debe resaltar que estas regiones vienen de una tabla principal" [75]



Ejemplo NoSQL por Regiones [75]

Ahora es posible realizar la respectiva transformación. Lo primero a considerar es el modelo entidad relación, el cual servirá para minimizar el tiempo de respuesta de la consulta para generar la matriz O / D. Para el caso de estudio, se pueden identificar varias tablas principales, las cuales pueden ir agrupándose: [61]

TRANS_Ruta contiene a TRANS_Parametros

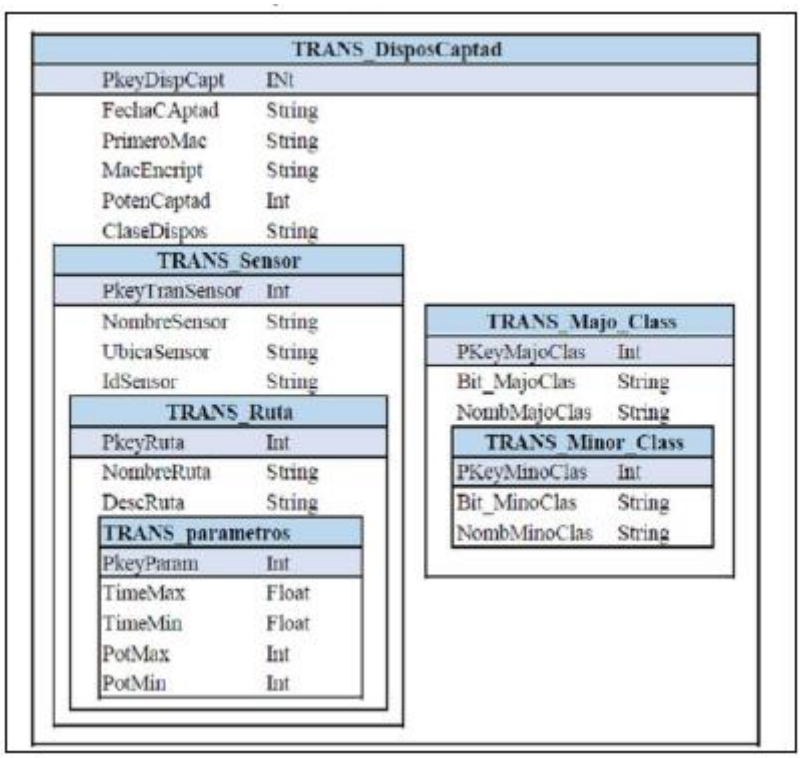
TRANS_Sentor contiene a TRANS Ruta

TRANS_Major_Class contiene a TRANS_Minor_Classs

TRANS_DisposCaptad contiene a TRANS_Major_Class y TRANS_Ruta

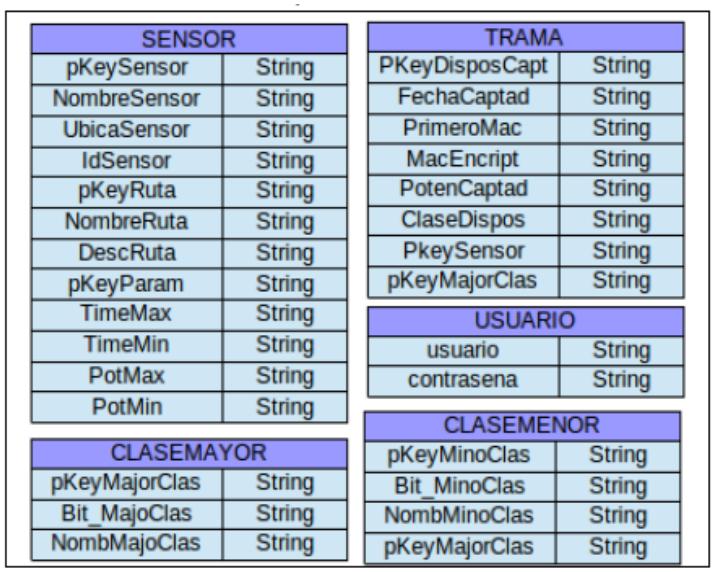
Para el caso de transformación por medio de anidaciones de tablas, se define un modelo que refiere una tabla contenedora: “TRANS_DisposCaptad”, “dentro de esta a su vez se crean tres regiones; la primera con sus atributos básicos, la segunda que contiene otra tabla contenedora en este caso TRANS_Sensor y la última región es para TRANS_Major_Class” [61].

“A continuación, se muestra de manera gráfica el modelo de tablas anidadas después del proceso de desnormalización para el caso de estudio” [61]:



Modelo NoSQL anidado caso de estudio [61]

El resultado de la respectiva transformación, arroja un modelo mucha más robusto que el obtenido originalmente desde la estructura SQL, generando respuestas a consultas de forma rápida usando datos escalables. Adicionalmente, las columnas anidadas tienen la característica de redundancia de información, que proporciona una mayor consistencia y seguridad de los datos. “Las familias de columna en su forma más simple, son sólo un elemento de agrupación para operaciones de lectura más eficientes” [61].



Modelo NoSQL sin anidación caso de estudio [61]

Este modelo cuenta con las siguientes tablas principales: [61]

Usuario: Utilizada para almacenar las diferentes personas que pueden ingresar al sistema, asociándoles a su vez una contraseña.
Sensor: es el resultado de la fusión entre las tablas que se encontraban anidades dentro de TRANS_Sensor, es decir cuenta con los atributos tanto de TRANS_Ruta como de TRANS_Parametros.
Trama: es la tabla que contiene los valores arrojados por el sensor, donde además se almacena información sobre el sensor y la clase del dispositivo.
Clase mayor y menor: estas tablas no presentaron cambios, son utilizadas como tablas paramétricas dentro del sistema.

Luego de validar ambos modelos (anidación y sin anidación), se optó por usar el modelo sin anidación, esto fue debido a que el estudio que se analiza actualmente no requiere dividir las tablas en regiones o tener más de una máquina para la aplicación. “Se realiza una estructura en XML, que busca dar solución al problema de expresar información estructurada de manera abstracta y reutilizable” [61].

En el siguiente gráfico se denota una estructura XML, referente al caso del modelo de transformación:

```
<table name="Band">
<Key>
<column name="band_id" type="int"/>
</Key>
<columnFamily name="cf1">
<column name="band_name" type="string"/>
<column name="hometown" type="string"/>
<entity name="show">
<Key>
<column name="show_id"/>
</Key>
<column name="date" type="date"/>
</entity>
</columnFamily>
</table>
```

Estructura XML [75]

El siguiente gráfico ya tiene que ver con la estructura XML para el trabajo que se analiza en este caso, esta primera parte incluye la tabla usuario y trama:

```
<table name="Usuario">
        <key>
                <column name="usuario" type=#String">
        </key>
        <columnFamily name="Administrador">
                <column name= "usuario"type="String/>
                <column name= "contrasena"type="String/>
        </columnFamily>
</table>

<table name="Trama">
        <key>
                <column name="PKeyDispositCapt" type=#String">
        </key>
        <columnFamily name="d1">
                <column name= "FechaCaptad"type="String/>
                <column name= "PrimeroMac"type="String/>
                <column name= "MacEncript"type="String/>
                <column name= "PotenCaptad"type="String/>
                <column name= "ClaseDispos"type="String/>
                <column name= "PkeySensor"type="String/>
                <column name= "pKeyMajorClas"type="String/>
        </columnFamily>
</table>
```

XML usuario y trama [61]

El siguiente gráfico XML corresponde al de las tablas restantes, es decir de la tabla sensor, clase mayor y clase menor:

XML tablas sensor, clase mayor y menor [61]

Pruebas comparativas entre SGBD NoSQL

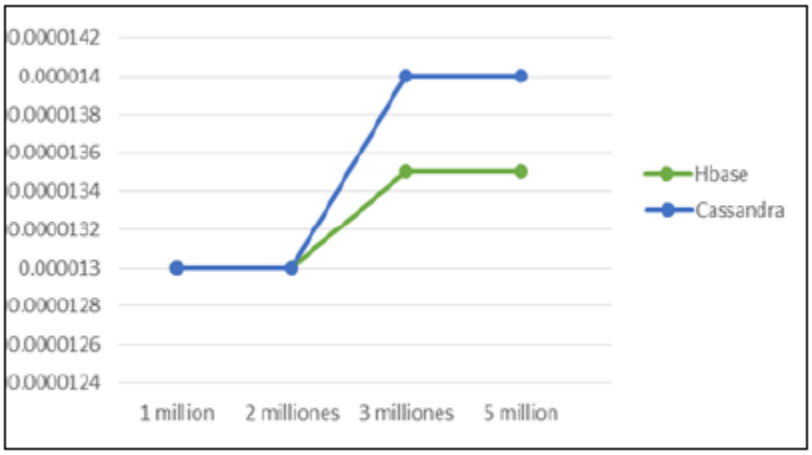
Durante el proceso previo de investigación, orientado en la búsqueda de herramientas NoSQL para la implementación del sistema que se analiza en esta sección, se realizaron pruebas entre los usuarios creados para Cassandra y HBase, “estas pruebas consistían en hacer un barrido de las inserciones, y tomar los tiempos de respuesta de ambos motores para observar si existía alguna diferencia significativa” [61].

Los resultados que arrojó dicha prueba, se analizaron y se graficaron respectivamente, generando diferentes indicadores. En la primera prueba Hbase demostró mayor fiabilidad, en el cual se pudo visualizar un tiempo mucho menor que el de Cassandra después de la inserción de un conjunto de datos.

	seconds	
Dataset Size	Hbase	Cassandra
1 million	0.000013	0.000013
2 milliones	0.000013	0.000013
3 milliones	0.0000135	0.000014
5 million	0.0000135	0.000014

Resultados de inserción de datos Hbase vs Cassandra [61]

Este ejercicio de prueba se basa en un artículo científico [76], "el cual presenta una serie de comparaciones y pruebas entre los motores mencionados en este capítulo, que permitirá otro punto decisivo para la elección final" [61].



Inserciones Hbase vs Cassandra [61]

Para la toma de la decisión final, acerca del motor de bases de datos que se debía usar en el proyecto, se tomó en cuenta tanto los resultados, como criticas de personal experto [61].

Adicionalmente, se desarrolló un apartado de características que influyeron en dicha decisión. Dicha información se puede detallar en las siguientes fuentes: [77] [78] [79] [80].

Diseño del sistema

El sistema en mención está orientado en la ejecución de un SGBD y su modelo de datos, “para esto se realizaron los diagramas de caso de uso, el diagrama de componentes y un diagrama de contexto que se observaran a continuación” [61].

El siguiente diagrama se refiere a los casos de uso que se encuentran presentes para el modulo del administrador. Este tiene la capacidad de realizar inserciones, modificaciones y eliminaciones sobre la información del SGBD NoSQL.

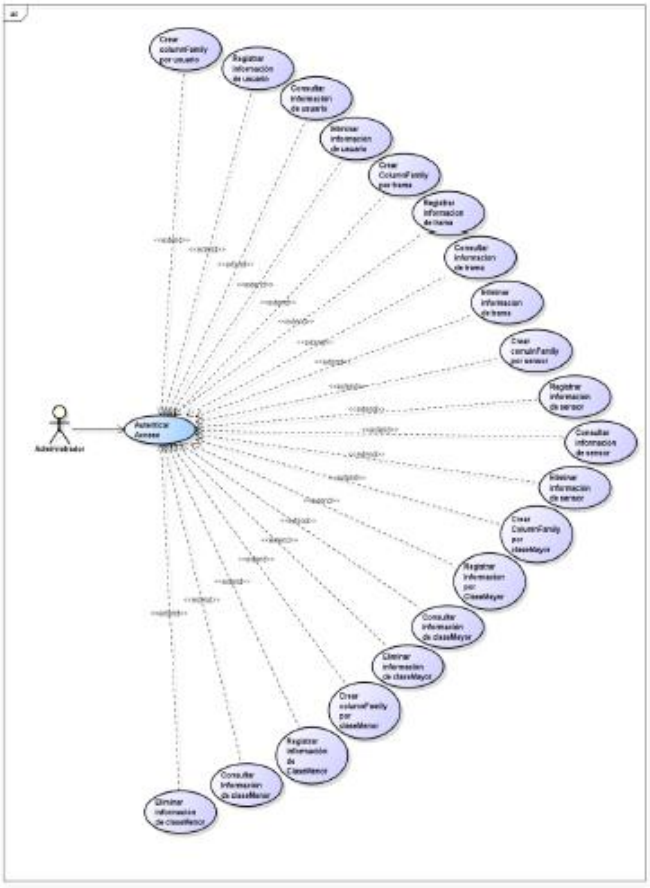


Diagrama caso de uso para el administrador [61]

En el siguiente diagrama, se denota el despliegue que realizará el sistema, donde se podrá observar la integración de la base de datos NoSQL que en este caso es Hbase.

Adicionalmente, permite observar las conexiones físicas entre el hardware y los componentes.

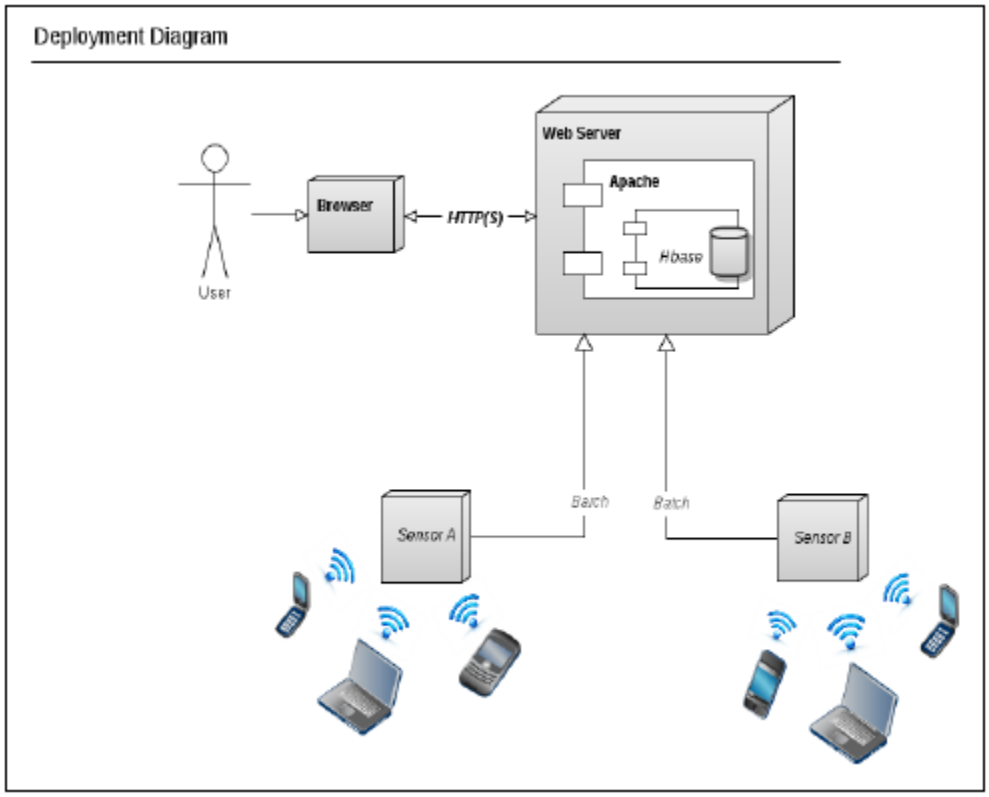


Diagrama de despliegue [61]

Desarrollo e implementación

En este apartado, se conocerá de forma global la intervención de factores como hardware, montaje de sensores y de forma específica (funcionamiento) el uso y aplicación del SGBD NoSQL que en este caso es Hbase.

Para el respectivo montaje se usaron los siguientes recursos:

- Energía: Fuente de voltaje: 5 Voltios con 2A.
- CPU: ARM 9 Processor.
- Memoria: 64 MB RAM applications 2 GB.
- Wireless: Habilitada Wifi (802.11 b/g).
- GSM Módem: módem celular integrado para la conectividad remota GPS sincronizar la hora y la ubicación utilizando el módulo GPS.

Adicional a esto, se usaron los diferentes componentes relacionados con la parte de software, así como el uso de esquemas XML para una integración del sistema. Adicionalmente, se integró un sistema de gestión de informes llamado Acyclica con el fin de calcular los tiempos de viaje y crear los respectivos informes.

En el tema de seguridad se utiliza un algoritmo MD5con el fin de cifrar las direcciones MAC para asegurar la privacidad de los usuarios. “Los dispositivos pueden ser emparejados de punto a punto, pero nunca descodificados para revelar el ID de dispositivo original” [61].

El sistema operativo usado para el despliegue del SGBD NoSQL fue Ubuntu 12.10 a 32 bits. Las demás aplicaciones usadas fueron el JRE de Java y el kit de desarrollo de java JDK. Para el desarrollo de la aplicación se usó el IDE de desarrollo “NetBeans”, además se manejó la librería WICKET 1.4.19 para el desarrollo del complemento WEB usado en este proyecto.

A continuación, se muestra la forma de recepción de la información recibida por los múltiples sensores usados durante el proyecto en ejecución:

Información en tiempo real [61]

En la siguiente imagen, se denota una prueba realizada en 20 minutos, donde se capta información proveniente de medios inalámbricos.

En esta se detalla información tal como fecha, hora, zona horaria, MAC, entre otra información:

Fri Mar 01 2013 18:19:42 GMT-0500 (COT)	1362179982.1	68:09:27:5f:61:c1	-65	Wifi
Fri Mar 01 2013 18:19:40 GMT-0500 (COT)	1362179980.13	6c:3e:6d:a6:98:9f	-58	Wifi
Fri Mar 01 2013 18:19:33 GMT-0500 (COT)	1362179973.95	00:26:ab:bb:50:4c	-47	Wifi
Fri Mar 01 2013 18:13:59 GMT-0500 (COT)	1362179639.42	00:0E:9F:AB:7B:B8	0	Bluetooth
Fri Mar 01 2013 18:09:35 GMT-0500 (COT)	1362179375.0	2C:30:68:03:BE:BA	0	Bluetooth
Fri Mar 01 2013 18:09:03 GMT-0500 (COT)	1362179343.27	2C:30:68:03:BE:BA	0	Bluetooth
Fri Mar 01 2013 18:07:06 GMT-0500 (COT)	1362179226.92	00:05:4F:CD:04:13	0	Bluetooth
Fri Mar 01 2013 18:04:17 GMT-0500 (COT)	1362179057.68	00:0E:9F:3A:DF:6E	0	Bluetooth

Recolección de datos [61]

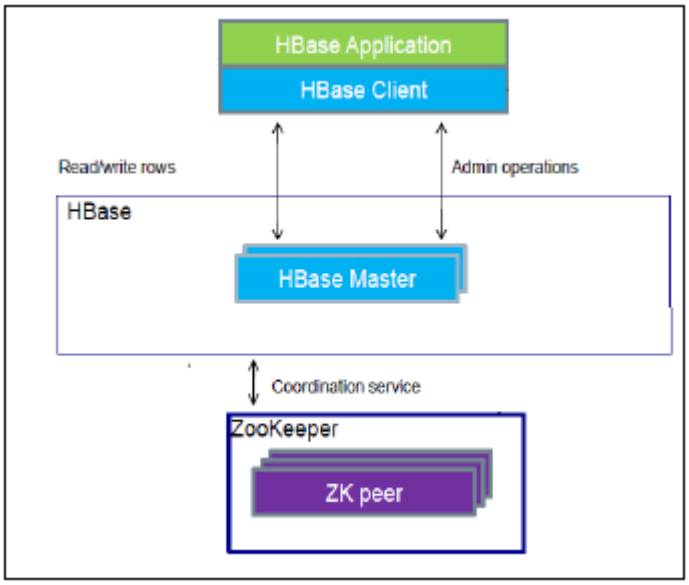
Configuración de la base de datos

Luego de la fase de análisis y pruebas para la selección del SGBD NoSQL que más se ceñía a las necesidades actuales del proyecto desarrollado, se mostrará los pasos que se usaron para el uso del mismo:

Luego de la instalación de HBase, se procede a su respectiva ejecución, tneiendo en cuenta sus tres modos de despliegue: “Independiente, pseudo-distribuida y completamente distribuida” [61]. Esto infiere que se está depurando HBase en un único proceso de Java, por ende, se requerirá de una máquina y una región para el caso de uso.

Esta implementación se definirá en tres divisiones que complementan todo el montaje del SQBD. En la primera se ubica la aplicación que interviene con Hbase cliente, la siguiente es encargada de aprobar lecturas, escrituras y otras operaciones y por último, el “ZooKeeper” [81] encargado de dirigir múltiples servicios.

A continuación, se muestra una gráfica con la información anteriormente mencionada:



Sistema HBase [61]

En la siguiente tabla, se muestra la respectiva configuración de HBase usando un archivo XML:

hbase-site.xml

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
-->
<configuration>
<property>
<name>hbase.rootdir</name>
<value>file:///home/andrea/hbase/</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/andrea/hbase/zookepeer</value>
</property>
</configuration>
```

Archivo XML HBase de configuración [61]

Aplicación web HBase

La aplicación WEB es la encargada de ser la intermediaria entre el usuario y la base de datos mencionada anteriormente (HBase), la cual se encarga de generar y almacena la matriz O / D. Las consultas básicas generadas hacia la base de datos a través del Web Service son las CRUD (Creación, Consulta, Eliminación y Actualización).

"A continuación, se observa un pequeño ejemplo de un filtro para buscar información en HBase" [61]:

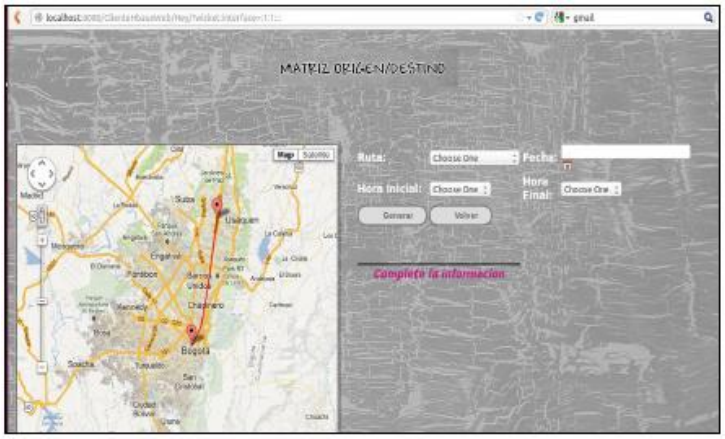
```
Scan s = new Scan();
s.addColumn(TWITS_FAM, TWIT_COL);
String expression = "ValueFilter(=,'regexString:.*TwitBase.*')";
ParseFilter p = new ParseFilter();
Filter f = p.parseSimpleFilterExpression(Bytes.toBytes(expression));
s.setFilter(f);
```

Filtros en HBase [82]

Pruebas finales de la aplicación basada en NoSQL (HBase)

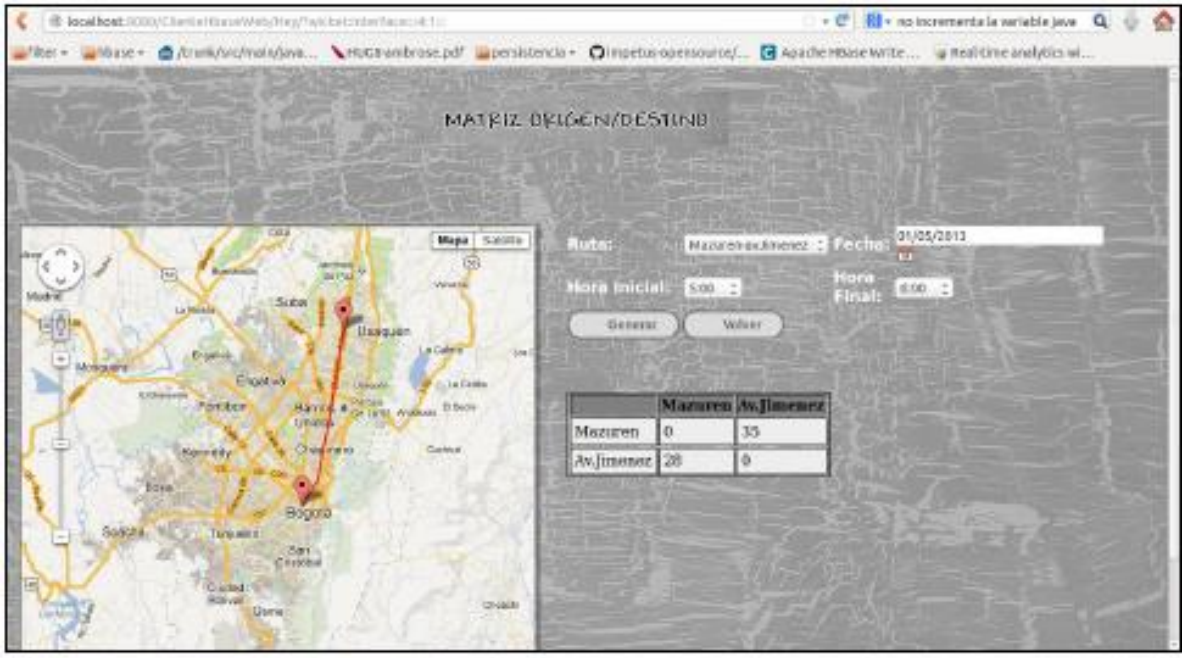
Al realizar múltiples consultas desde la aplicación hacia el servidor que aloja el sistema de gestión de base de datos NoSQL, se determinó que la conexión y la generación de la base de datos se realiza de forma correcta y sin ningún percance detectable.

Al ingresar a la aplicación WEB, se visualiza un entorno basado en Google Maps, mostrando las rutas ingresadas dentro de la aplicación y su respectiva descripción [61]:



En el momento que los campos están completos, se procede a invocar los métodos que contienen una cantidad de filtros, "con estos se realizan las consultas, comparaciones y cálculos necesarios para general finalmente el matriz origen destino" [61].

"En la siguiente figura se presenta la matriz O/D, que resulto luego de las consultas sobre la base de datos" [61].



Pruebas Matriz O / D aplicación cliente WEB [61]

Antes de la implementación final, se realizaron las respectivas pruebas para medir tiempos de consulta en cuanto a inserciones sobre la base de datos, estos se presentan a continuación en la tabla [61]:

Inserciones	Tiempo (Segundos)
20	3.575
40	4.895
100	6.831
160	7.753
320	10.248
640	15.967
1000	20.563
2000	35.563
5000	85.563

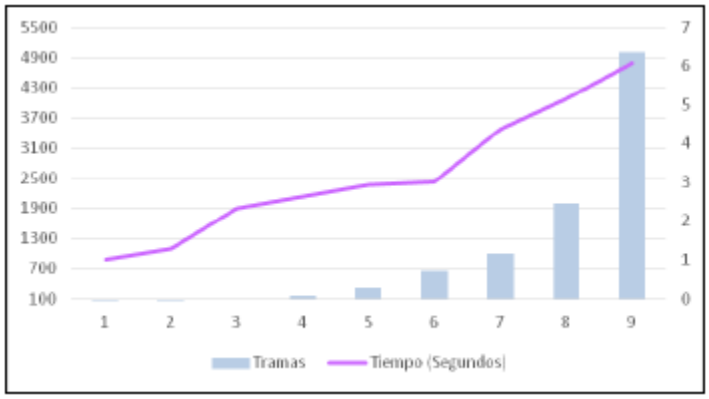
Inserciones finales sobre la base de datos [61]

Se realizan pruebas sobre la base de datos, midiendo el tiempo necesario para generar la matriz O / D a medida que las tramas se amplían.

	Generación Matriz O/D
Tramas	Tiempo (Segundos)
20	1,006
40	1,295
100	2,321
160	2,641
320	2,931
640	3,007
1000	4,364
2000	5,154
5000	6,061

Generación matriz O / D VS Tramas [61]

En la siguiente gráfica se detalla la información anteriormente mostrada:



Tiempos de consulta matriz O / D [61]

Conclusiones

Este trabajo muestra un ejemplo orientado a dar solución a un problema de la vida real usando bases de datos NoSQL en un sistema de transporte masivo. Además de esto, la plataforma usada para la implementación de esta base de datos es libre y sin requerir un alto volumen de recursos.

Dicha propuesta nace de un trabajo de grados de una universidad colombiana, lo cual supone el empeño en las instituciones del país en la construcción de sistemas novedosos por medio de una tecnología que es popular a nivel internacional y que se usa para diferentes ámbitos.

En este ejercicio, las bases de datos NoSQL muestran su potencial y la forma de implementación por medio de técnicas y métodos orientados a este tipo de tendencias (Des normalización y transformación), pero más allá de estos, validar la conveniencia del uso de estos y en qué casos es factible su incluso.

El proyecto actual evalúa esta tendencia, sin embargo, el volumen de información que se genera en la implementación no es suficiente para retar las opciones ofrecidas por los sistemas de gestión de bases de datos NoSQL, incluso, en este caso fue consideradas más eficientes las bases de datos clave / valor por su eficiencia en este tipo de datos y no por los alcances que pueda llevar el sistema en implementación.

Sin embargo, el desarrollo de este proyecto da muchos frentes para el trabajo en otro tipo de sistemas, así como el internet de las cosas, sistemas de información mediante la computación en la nube e incluso el manejo de altos volúmenes de datos por medio del BigData. Esta tecnología tiene mucho futuro en el manejo de cualquier tipo de sistema con cualquier entorno de hardware, ya que los actuales modelos estructurados, requieren de procesadores y maquinas con mucho más volumen para soportar su procesamiento, sin dejar olvidar su limitación con la cantidad de datos suministrados.

Si se habla desde la parte financiera, se puede concluir que independientemente de la cantidad de información que se maneje, la alternativa NoSQL es muy viable ya que se basa en su gran mayoría de tecnología libre y los accesorios que requiere en términos de hardware no son muy complejos o son muy económicos en términos de adquisición.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Otros proyectos NoSQL en ejecución:

La generación de sistemas de información usando ambientes NoSQL puede ser un tema que puede atraer mucha ambigüedad, sin embargo y como se ha mencionado en todo el proceso investigativo y análisis, dicha tendencia se ha transformando en una herramienta indispensable que en un futuro puede tomar dos rumbos, los cuales refieren a la sustitución de la tecnología actual (háblese de sistemas de gestión de base de datos estructurados) o a la transformación en una tecnología hibrida, como se habló en la sección de comparativas de diversos motores NoSQL vs SQL.

Además de esta tendencia, han surgido otros paradigmas como lo son el Big Data, el internet de las cosas, la computación en la nube y otros sistemas de información basados en grandes volúmenes de datos.

Esta nueva necesidad, ha ocasionado que se generen alternativas muy innovadoras para suplir inconvenientes o intervenir en baches que no alcanzaba a abarcar la tecnología actual.

El caso de la implementación de una base de datos NoSQL para la solución de un problema de movilidad, como se vio en el trabajo anteriormente mencionado “Implementación de una base de datos No SQL para la generación de la matriz O / D” [61], es un caso colombiano en la gestión y uso de información a través de este tipo de SGBD.

El caso del trabajo de grado “Diseño y desarrollo de una guía para la implementación de un ambiente Big Data en la Universidad Católica De Colombia” [83], abre también un mundo de alternativas usando NoSQL para la intervención de este tema que esta tan actual y que se asemeja más al NoSQL por sus características y naturaleza misma.

Otro ejemplo es “Diseño y elaboración de prácticas de laboratorio para la enseñanza de los conceptos fundamentales de bases de datos no relacionales - NoSQL” [84], ilustra acerca del uso y aplicación de los SGBD no estructurados en ambientes no tradicionales, dando paso a innovación en uso de tecnologías diferentes para suplir necesidades. También se habla acerca de comparativas a fondo de seguridad y fiabilidad en la estructura de los datos en diferentes usos y aplicaciones.

Otro ejemplo es el trabajo “Un viaje a través de bases de datos espaciales NoSQL” [85], donde se menciona el recién re aparición de este tipo de tendencia, la utilidad y su potencial para el cual está desarrollado.

Otro documento es el titulado “Análisis Bioinformáticos sobre la tecnología Hadoop” [86], en el cual se detalla la aplicación de las bases de datos no estructurados en la rama de la informática ligada a la medicina, dando lugar a concluir la inclusión de esta tecnología en múltiples ambientes.

Conclusiones generales:

Los resultados generados por todas las bases de datos especificadas y probadas fueron significativos en gran medida, independientemente de la aplicación o el entorno en que se aplique, superan en muchas características (midiéndose en resultados y en consumo de recursos en tiempo de ejecución) a las bases de datos tradicionales. Como se mencionó durante todo el documento, son muchos paradigmas los que surgen actualmente y es debido a las necesidades de información usando datos de forma masiva en diferentes entornos. Hoy no se puede medir por uso en transacciones comerciales o por uso y aplicación de sistemas de información empresariales, sino en lo que trasciende más allá.

Redes neuronales, robots y otros nuevos sistemas basados en algoritmos de gran procesamiento son quienes están midiendo las capacidades de los sistemas de gestión de bases de datos actuales. Y respecto a esto, tecnología que se conecta con otras tendencias, son quienes se quedan limitadas si se usan motores estructurados.

NoSQL se ha transformado para dar soporte a estos modelos, es por esto que a pesar de haber surgido hace años, se volvió a retomar poco menos de 8 años.

Aunque la variedad de modelos basados en NoSQL está incrementando, hoy no es posible determinar cuál es más óptimo para un montaje general de un sistema de información masivo o basado en minería de datos. Es por esto, que grandes del internet usan diferentes modelos para suplir sus necesidades (Gestión de cache, administración de Time Line, transacciones en tiempo real, entre otros). En laboratorio se han simulado muchos ambientes usando diversas tecnologías y esto mismo puede ser evidenciado en la sección donde se habla de la comparativa y rendimiento entre Sistemas de gestión de bases de datos; sin embargo, no fueron contundentes para determinar cuál sistema (Documento, Grafo, Clave / Valor, Orientada a columna) es mejor para suministrar contenido a un sistema de información, pero si se puede deducir que cada una de ellas tiene una especialidad según el tipo de información que se tenga, la arquitectura que se utilice y los requerimientos puntuales que se especifiquen.

“Las bases de datos con esquemas rígidos, son más adecuadas para manejar estructuras de datos con información descriptiva” [44]. Sin embargo y como se mencionó anteriormente, la cantidad de pruebas y la cantidad de datos utilizada, no dan una determinación final para afianzar este concepto.

Considerando las limitaciones con las que cuentan los sistemas de gestión de bases de datos SQL actuales y las necesidades que estas a su vez generan, dan como resultado la justificación en el uso y aprovechamiento de la tendencia NoSQL, la cual ha dado la posibilidad de que paradigmas como el BigData, IoT (Internet de las cosas), computación en la nube, entre otras, puedan ser aprovechadas de forma oportuna y que puedan trabajar de forma óptima con la información que capta y procesa en el medio.

Es Importante aclarar que aún existen muchas cuestiones y potencial para explorar en áreas de investigación acerca de las tendencias NoSQL. Como por ejemplo, los tipos de SGBD “clave-valor” presentan problemas para realizar consultas complicadas o que algunas de las bases de datos orientada a gráficos no pueden procesar particionamiento y la mayoría de

las implementaciones de bases de datos documentales son incompetentes para efectuar uniones o transacciones que incluyan varias filas o documentos.

En conclusión, no se descarta que, en un futuro ambos sistemas tanto como bases de datos relaciónales y sistemas de gestión de bases de datos NoSQL, se perfeccionen y que la tendencia sea la construcción de sistemas híbridos (como se mencionó en el apartado de Hadoop), combinados por diversos almacenes de datos, cada uno de ellos basado en diferentes principios.

REFERENCIAS

[1] D. Berndt et al. (2012, Apr.), “Site Wit Corporation: SQL or NoSQL that is the Question”, Grandon Gill’s Website [Online], Available: http://grandon.com/publications/SiteWit_NoSQL.pdf.

[2] Revista Facultad de Ingeniería - UPTC. 2012, Vol. 21 Issue 33, p21-32. 12p. .

[3] E. Codd, “A relational model of data for large shared data banks,” Commun.ACM, vol. 13, no. 6, pp. 377–387, Jun. 1970.

[4] S. Drobi (2012,Aug. 3), “RichHickey and Justin Sheehy about Datastores, NoSql and CAP”, InfoQ [Online], Available: http://www.infoq.com/interviews/rich-Hickeyand-justin-sheehy-aboutdatastores,-nosql-and-cap.

[5] Professional NoSQL, Shashank Tiwari, John Wiley & Sons, 2011.

[6] Cisco and/or its affiliates (2011, Nov.), “Cisco Global Cloud Index: Forecast and Methodology 2010–2015” [Online], Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns1175/Cloud_Index_White_Paper.pdf.

[7] M. Butter (2012, Aug.), “London Olympics Get Gold Medal for BigData (Infographic)”, NetApp [Online], Available: http://www.forbes.com/sites/netapp/2012/08/08/london-olympics-get-goldmedal-for-big-data-infographic/.

[8] J. Lecat (2010, Sep.), “Jérôme Lecat on scality vol. 1”, The NoSQL Tapes [Online], Available: http://nosqltapes.com/video/jerome-lecat-onscality.

[9] M. Loukides (2012, Feb.), "The NoSQL movement: How to think about choosing a database", O'Really Strata [Online], Available http://strata.oreilly.com/2012/02/nosql-nonrelational-database.html.

[10] E. Schurman and J. Brutlag, "The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search in", Velocity 2009 Conference [Online], 2009, Available: http://www.youtube.com/watch?v=bQSE51-gr2s.

[11] E. Arcos (2009, Apr.), "Oracle compra Sun", ALT140 La guía de geek [Online], Available: http://alt1040.com/2009/04/oracle-compra-sun.

[12] E. Lai (2009, Jul.), "No to SQL? Anti-database movement gains steam", ComputerWorld [Online], Available:
http://www.computerworld.com/s/article/9135086No_to_SQL_Anti_database_mov.ement_gains_steam_?taxonomyId=173&pageNumber=2.

[13] A. Marcus, "The NoSQL Ecosystem", The Architecture of Open SourceApplications, 2011.

[14] Google Trends (2012, Oct.), "Informe de búsquedas del término: NoSQL", Google [Online], Available: http://www.google.de/trends/?q=nosql&ctab=0&geo=all&date=all.

[15] R. Cattell, "Scalable SQL and NoSQL datastores", SIGMODRec., vol. 39, no. 4, pp. 12–27, May. 2011.

[16] Revista Facultad de Ingeniería, UPTC, Julio-Diciembre – CEDEC de 2012, Vol. 21, No. 33

[17] G.Burd, "NoSQL", Usenix, vol. 36, pp. 5-12,Oct. 2012.

[18] Brewer. (2000). Principles of Distributed Computing. Nineteenth ACM Symposium on Principles of Distributed Computing.

[19] D. Pritchett, "BASE: An Acid Alternative", Queue, vol. 6, no. 3, pp. 48–55, Mayo 2008.

[20] R. Agrawal et al., "The Claremont report on database research," SIGMOD Rec., vol. 37, no. 3, pp. 9–19, Sep. 2008.

[21] Init Developers. (2012). Introducción a NO-SQL: Cassandra y CouchDB. Recuperado (2014, junio 19) de http://blog.theinit.com/2012/04/24/introducciona-no-sql-cassandra-y-couchdb/

[22] Gerencia Tecnológica Informática, Septiembre 1, 2014

[23] Hecht, R., 8i Jablonskl, S. (2011). NoSQL evaluation a use case oriented survey. Cloud and Service Computing, International Conference, 336-341.

[24] Amazon Web Services. (2012). DynamoDB. Seattle, WA, Estados Unidos. Recuperado (2014, febrero 3) de http://aws.amazon.com/es/dynamodb/

[25] Kleppmann, M. (2009). Should you go Beyond Relational Databases. Estados Unidos: Treehouse Island, Inc. Recuperado (2014, enero 22) de http://thinkvltamln.com/code/should-you-go-beyondrelatlonal-data bases/

[26] Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2), 35-40.

[27] Chang, E, Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., & Gruber, R. (2008). Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS), 26(2), 4.

[28] Apache Software Foundation. (2012). HBase. Delaware, Estados Unidos. Recuperado (2014, febrero 03) de http://hbase.apache.org/

[29] Kopp Michael (2011). NoSQL or RDBMS? - Are we asking the right questions?. Recuperado (2014, junio 19) de http://apmblog.compuware.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/.

[30] Kopp Michael (2011). NoSQL or RDBMS? - Are we asking the right questions?. Recuperado (2014, junio 19) de http://apmblog.compuware.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/

[31] Joyanes, L. (2014). Big data: análisis de grandes volúmenes de datos en organizaciones. Barcelona: Marcombo Ediciones Técnicas.

[32] Moreno, A., Redondo, T. (2016). Text Analytics: the convergence of Big Data and Artificial Intelligence. En International Journal of Interactive Multimedia and Artificial Inteligence, 3(6).

[33] Scofield, B. (2009). NoSQL: Death to Relational Databases. Online RubyConference. En: http://es.slideshare.net/bscofield/nosql-death-to-relational-databases

[34] MongoDB CRUD (2015). Introduction - MongoDB Manual 3.0.1. En: http://docs.mongodb.org/manual/core/crudintroduction.(22 de septiembre de 2015).

[35] Ciencia e Ingeniería Neogranadina. 2016, Vol. 26 Issue 1, p109-129. 21p.

[36] F. J. Moreno Arboleda, J. E. Quintero Rendón, R. Rueda Vásquez. (2016). Una comparación de rendimiento entre Oracle y MongoDB. Ciencia e Ingeniería Neogranadina, 26 (1), pp. 109-129.

[37] Dean, J., Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1).

[38] Niemiec, R. (2012). Oracle Database 11g Release 2 Performance Tuning Tips & Techniques, McGraw-Hill.

[39] Avinash Lakshman and Prashant Malik. Cassandra - a decentralized structured storage system. Technical report, Cornell University, 2009.

[40] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: a distributed storage system for structured data. In OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation, pages 205–218, Berkeley, CA, USA, 2006. USENIX Association.

[41] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, pages 205–220, New York, NY, USA, 2007. ACM.

[42] JAKOB MATTSSON, June 2010. Examiner: GRAHAM KEMP. Department of Computer Science and Engineering. Chalmers University of Technology.

[43] What is Cassandra? (2016) http://cassandra.apache.org/

[44] Pollo, M., López, M & Daián, G. Cuaderno Activa N°6 Enero – Diciembre 2014 pg. 13

[45] Lith, A., Mattsson, J. (2010). Investigating storage solutions for large data. (Tesis de maestría). Chalmers University of Technology.

[46] Edlich, Stefan, "NoSQL, your ultimate guide to the non - relational universe!", http://nosql-database.org/, (unpublished)

[47] Bogdan G., Cristian B., A comparison between several NoSQL databases with comments and notes, Department for Economical Mathematics and Economical Informatics.

[48] Peters, Mike, "How to install Cassandra + Thrift (and why you should care)", http://www.softwareprojects.com/resources/programming/t-howto-install-cassandra-+-thrift-and-why-you-shou-1956.html, (unpublished).

[49] Cooper, Brian F., "Yahoo! Cloud Serving Benchmark", http://research.yahoo.com/files/ycsb-v4.pdf, (unpublished)

[50] Thusoo, A., Sarma, J. Jain, N., Shao, Z., Chakka, P., & Murthy, R. (2009). Hive: a warehousing solution over a Map-Reduce framework. Proceedings of the VLDB Endowment, 2(2), 1626-1629.

[51] Lorica. (2009). HadoopDB: An Open Source Parallel Database. Estados Unidos: O'Reilly Media, Inc. Recuperado (2014, enero 22) de http://strata.oreilly.com/2009/07/hadoopdb-an-open-source-paralleldatabase.html

[52] Abouzeid, A., Bajda- Pawlikowski, K., Abadi, D., Silberschatz, A., & Rasin, A. (2009). HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. ACM Proceedings VLDB Endowment, 2(1), 922-933.

[53] Jaramillo Valbuena, S. & Londoño, J. M. (2014). Sistemas para almacenar grandes volúmenes de datos. En R, Llamosa Villalba (Ed.). Revista Gerencia Tecnológica Informática, 13(37), 17-28.

[54] Brown, R. A. (2009). Hadoop at home: large-scale computing at a small college. SIGCSE '09 Proceedings of the 40th ACM technical symposium on Computer science education, 41(1), 106-110.

[55] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM - 50th anniversary issue: 1958 - 2008, 51(1), 107-113.

[56] Análisis de la información de una base de datos transaccional usando hive sobre hadoop. Alcívar, Mercedes; Espinoza, Iván; Cedeño, Vanessa.

[57] Base de datos Redis: Todo lo que debes saber: Borja On 18 octubre, 2012, Recuperado de: http://www.antweb.es/servidores/redis-todo-lo-que-debes-saber

[58] No sólo clave-valor. Redis te da alas: Vicente Ayestarán, 23 septiembre del 2015: Recuperado de: https://www.paradigmadigital.com/techbiz/no-solo-clave-valor-redis-te-da-alas/

[59] Información recuperada desde el foro de ayuda Redis: http://tiku.io/questions/1624859/mapreduce-read-input-from-redis

[60] How Fast is Redis? http://redis.io/topics/benchmarks

[61] Barragán Charry, A. M., & Forero Sanabria, A. (2013). Implementación de una base de datos NoSQL para la generación de la matriz O/D. (tesis de pregrado). Universidad Católica de Colombia, Bogotá.

[62] ITS COLOMBIA. [En línea] < http://www.its-colombia.org/> [citado en 4 de marzo de 2013].

[63] CINTEL. Intelligent Transportation systems-its- en Colombia: Estudio cualitativo. 2010 [en línea]. <http://www.interactic.org.co/> [citado en 8 de marzo de 2013].

[64] MERZ Sinclair. El futuro de alta tecnología para el transporte [en línea]. <http://www.globalskm.com/Insights/Achieve-Magazine/Issue3-09-ESP/article5.aspx> [citado en 18 de marzo de 2013]

[65] ITS Standardization Activities of ISO/TC204 - 2011 [En línea] <http://isotc204-publicdocuments.itsa.wikispaces.net/file/view/JSAE+TC204+Broch ure+(2011+Version).pdf> [Citado el 04de marzo de 2013].

[66] CASCETTA, E., POSTORINO, M., Fixed Point Approaches to the Estimation of O/D Matrices Using Traffic Counts on Congested Networks Transportation Sciencie informs, 2001, 35(2), p.134-147.

[67] TORGIL, Abrahamsson.Estimation of origin-destination matrices using traffic counts- a literatura survey. En: EconPapers [En línea]. <http://webarchive.iiasa.ac.at/Publications/Documents/IR-98-021.pdf > [citado en 10 de marzo re de 2013].

[68] RODRIGUEZ. Jean-Paul. Hofstra University, New York, [En linea] <http://people.hofstra.edu /geotrans/eng/methods/odmatrix.html >[Citado en 22 de marzo de 2013].

[69] HECHT Robin. JABLONSKi Stefan. NoSQL Evaluation A Use Case Oriented Survey. 2011 International Conference on Cloud and Service Computing. University of Bayreuth. Germany. P.339

[70] HECHT Robin. JABLONSKI Stefan. NoSQL Evaluation A Use Case Oriented Survey. 2011 International Conference on Cloud and Service Computing. University of Bayreuth. Germany. P.341

[71] HECHT Robin. JABLONSKI Stefan. NoSQL Evaluation A Use Case Oriented Survey. 2011 International Conference on Cloud and Service Computing. University of Bayreuth. Germany. P.340

[72] JOB TRENDS,Trend Observation [En Linea] http://trac.nchc.org.tw/grid/wiki/jazz/NoSQL 2010 [Citado el 06 de marzo de 2013]

[73] JOB TRENDS [En Linea]http://www.simplyhired.com/a/jobtrends/trend/q-Cassandra%2C+Redis%2 C+Voldemort%2C+Simpledb%2C+Couchdb%2C+Mongodb%2C+Hbase%2C+Hypertable [Citado el 06 de marzo de 2013]

[74] CLOUDERA,Hbase Schema Desing [En Linea]< http://www.slideshare.net/cloudera/5-h-base-schemahbasecon2012> [Citado el 15 de abril de 2013]

[75] LARS George. Hbase schema design. and Cluster Sizing Notes ApacheCon Europe, November 2012- p.20

[76] MCGLOTHLIN, P., KHAN-LATIFUR James,. Scalable Queries For Large Datasets Using Cloud Computing: A Case Study- p. 13

[77] CHALKIADAKi. Maria, MAGOUTIS Kostas. Managing service performance in NoSQL distributed storage systems- p.20

[78] MCGLOTHLIN P., KHAN-LATIFUR James. Op. Cit.

[79] WILLIAMS Dominic . HBase vs Cassandra: why we moved Marzo, 2011 [En línea] http://ria101.wordpress.com/2010/02/24/hbase-vs-cassandra-why-we-moved/ [Citado el 15 de abril de 2013]

[80] LI Pi. Caching in Apache HBase: SlabCache.Enero 2012[En línea] http://blog.cloudera.com/blog/2012 /01/caching-in-hbase-slabcache/ [citado el 18 de abril de 2013]

[81] IBM. What is ZooKeeper?. [En linea]< http://www-01.ibm.com/software/data/infosphere/hadoop/ zookee per/> [Citado el 2 de mayo del 2013].

[82] DIMIDUK. Nick, KHURANA. Amandeep. Hbase in action.2013.-p52

[83] Guerrero López, F. A., & Rodríguez Pinilla, J. E. (2014). Diseño y desarrollo de una guía para la implementación de un ambiente Big Data en la Universidad Católica de Colombia. (tesis de pregrado). Universidad Católica de Colombia, Bogotá.

[84] Bejarano Ocampo, C. A., & Montes Rozo, J. A. (2015). Diseño y elaboración de prácticas de laboratorio para la enseñanza de los conceptos fundamentales de bases de datos no relacionales–NoSQL. (tesis de pregrado). Universidad Católica de Colombia, Bogotá.

[85] Ramírez Arévalo, H. H., & Herrera Cubides, J. F. (2013). Un viaje a través de bases de datos espaciales NoSQL. Redes De Ingeniería, 4(2), 57-69.

[86] UNIVERSIDAD AUTÓNOMA DE BARCELONA. Análisis Bioinformáticos sobre la tecnología hadoop [en línea]. Madrid: Carmen Palacios Díaz [citado 10 agosto, 2013]. Disponible en internet: http://ddd.uab.cat/pub/trerecpro/2012/hdl_2072_196270/GutierrezMillaAlbertR-ETISa2010-11.pdf

[87] Alomari, E., Barnawi, A., & Sakr, S. (2015). CDPort: A Portability Framework for NoSQL Datastores, 2531–2553. http://doi.org/10.1007/s13369-015-1703-0

[88] Bagade, P., Chandra, A., & Dhende, A. B. (2012). Designing Performance Monitoring Tool for NoSQL Cassandra Distributed Database.

[89] Chalkiadaki, M. (2013). Managing Service Performance in the Cassandra Distributed Storage System. http://doi.org/10.1109/CloudCom.2013.16

[90] Chang, B. R., Tsai, H., Guo, C., Chen, C., & Engineering, I. (2015). Remote Cloud Data Center Backup Using HBase and Cassandra with User-Friendly GUI, 420–421.

FIRMA ESTUDIANTES Juan Pablo Gómez Gil

FIRMA ASESOR Jorge I. Bedoya

FECHA ENTREGA: 5/11/16