UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: Flexibilidad en bases de datos NoSQL sobre ambientes Web Mining
Autores: Lisandro della Croce – Jorge Adrián Salinas
Director: Javier Bazzocco
Carrera: Licenciatura en Sistemas

Resumen

Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación, es decir, con el conocido modelo relacional. A su vez, las bases de datos NoSQL son conocidas principalmente por el hecho de mejorar tiempos de respuesta al manipular datos masivos.
En la actualidad, los avances en el desarrollo Web como así también las redes sociales con millones de usuarios, provocaron en general inconvenientes de alta escalabilidad y agregaron complejidad en la representación de los datos; es en estos casos principalmente donde el uso de NoSQL comenzó a utilizarse con más presencia en la última década. El ámbito Web Mining (metodología de recuperación de la información que utiliza herramientas de Data Mining para extraer información del contenido de la web) permite abordar en esta tesis un caso de análisis particular en una base de datos relacional y otra NoSQL sobre un modelo de log mediante operaciones comunes que usuarios podrían hacer sobre un servidor.

Palabras Claves

-Base de datos
-NoSQL
-Web Mining
-BigData
-MongoDB
-Log
-Servidor
-MySQL
-JDBC

Trabajos Realizados

-Investigación sobre concepto, diseño y desarrollo sobre base de datos NoSQL.
- Diseño y representación de un modelo de Log en una base de datos relacional y NoSQL.
- Desarrollo e implementación de patrones Web Mining sobre una base de datos relacional y NoSQL.
-Análisis de resultados obtenidos sobre los patrones Web Mining ejecutados sobre una base de datos relacional y NoSQL.

Conclusiones

Mediante la investigación y el análisis de las pruebas realizadas en el escenario de la tesis, se demuestra que es altamente recomendable el uso de NoSQL para operaciones comunes sobre un modelo de Log. El modelo y el desarrollo trabajado en conjunto con los resultados obtenidos derivan a la utilidad y adaptabilidad para cualquier usuario y/o administrador de los mismos. El modelo desarrollado se presenta como base para continuar investigaciones en otras direcciones.

Trabajos Futuros

- Representación del modelo de Log en otra base de datos distinta a NoSQL - Orientada a Documentos y comparar los tiempos.
- Análisis de factibilidad para implementar patrones Web Mining aplicados a este tipo de base de datos NoSQL en un ambiente distribuido.
- Investigar el comportamiento (almacenamiento y recuperación) de documentos embebidos para bases de datos NoSQL orientado a documentos sobre patrones Web Mining.

Fecha de la presentación: Marzo 2016

Tesina de la carrera Licenciatura en Sistemas

Título: "*Flexibilidad en Bases de Datos NoSQL sobre ambientes Web Mining".*

Integrantes:

- della Croce, Lisandro – Legajo: 5656/1
- Salinas, Jorge Adrián – Legajo: 5151/9

Director: Mg. Javier Bazzoco

ÍNDICE

CAPÍTULO I – Base de Datos a Gran Escala

Introducción

Una Base de datos a gran escala o también llamada “masiva”, permite el almacenamiento y manipulación de un orden alcanzable a petabytes de cualquier tipo de información diaria [1]. El crecimiento exponencial de información es directamente aplicable a estas bases de datos [2]. Por ejemplo, la consolidación de las redes sociales sumado a su utilización de forma diaria y masiva por parte de sus millones de usuarios, llevan a tener que realizar modificaciones que faciliten la administración de los datos [3].

Si bien no todas las aplicaciones necesitan almacenar y procesar datos de la misma manera, la arquitectura de almacenamiento debería pensarse de forma acorde a las necesidades. Si se pretende desarrollar una aplicación que requiera la lectura/escritura de millones de datos y pueda dar servicio a millones de usuarios sin perder rendimiento (escalabilidad), entonces debe plantearse el uso de una base de datos NoSQL. Pues, el término NoSQL se refiere al procesamiento de información en bases de datos que intentan solventar las limitaciones que el modelo relacional encuentra en entornos de almacenamiento masivo de datos, y concretamente al momento de escalar, donde es necesario disponer de servidores de alta performance y de balanceo de carga.[4].

Como se menciona en [3], para saber si es recomendable utilizar una base de datos NOSQL, se requiere un análisis previo de las condiciones existentes y de los objetivos que se pretenden lograr con el uso de la misma. Cuando se trata de aplicaciones de escritura escalables, como las redes sociales Facebook , Twitter, o el propio Google, donde es fundamental un medio de almacenamiento de información, las bases de datos relacionales no son recomendadas, ya que la normalización de datos, los joins y las transacciones ACID (las cuales son un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción) suelen presentar soluciones negativas que presentan más problemas para la escritura escalable[5]. No obstante, después de haber decidido que se va a usar una base de datos no relacional, es necesario elegir qué tipo de base de datos NoSQL se utilizará. “*Cap. 3 – Clasificación de BBDD NoSQL*”.

En el caso de Twitter, por ejemplo (*“sus usuarios envían 2 millones de tweets por día, lo que equivale a un libro con 10 millones de páginas”* [6]), se generan 12

terabytes diarios y aproximadamente 4 petabytes cada año [7]. Para manejar este tipo de situaciones y otros propósitos para los cuales la base de datos MySQL no es ideal, se aplican soluciones NoSQL.

Existe un área específica abocada al análisis sobre contenidos web en general y también aplicable a servidores en lo que se refiere a todas las actividades llevadas a cabo por los mismos (por ejemplo a escrituras y lecturas) que utiliza el concepto *Data Mining* de base de datos para obtener una mayor adecuada representación y extracción de información. Dicho concepto es denominado "*Web Mining"* que apunta al proceso global de descubrir información o conocimiento potencialmente útil y previamente desconocido a partir de datos de la web [12]. También es considerado un campo multidisciplinar donde además del datamining convergen áreas como la recuperación de la información, la estadística, la visualización de datos, lenguajes de etiquetas y tecnología web, con el objetivo de descubrir diferentes relaciones existentes en la W3, utilizando su información desestructurada o semiestructurada. En estos aspectos las tecnologías NoSQL y BigData son adecuadas y directamente aplicables para obtener en una amplia variedad de casos una mejor optimización de la información, principalmente en cuanto a la representación y extracción de la misma [3].

NoSQL

¿Qué es NoSQL?

Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación, es decir, con el conocido modelo relacional.

El término NoSQL, independientemente del concepto "Base de datos" es interpretado comúnmente como "Not Only SQL" (no solo SQL). También se lo puede conocer como una combinación de dos palabras: "No" y "SQL", es decir que el término no posee un único significado.

El creador del término "NoSQL" fue Carlo Strozzi, quién en 1998 [8] lo utilizó para referirse a una base de datos específica, a la cual utilizaba como base de estudio. Dicha base de datos, no ofrecía SQL para manipular consultas, pero sí utilizaba el modelo relacional. Strozzi sugirió también el nombre "NoREL", en alusión a bases de datos no relacionales, el cual no tuvo mucha repercusión, porque no incluía lo referente a clasificaciones de bases de datos, definiciones, referencias, arquitecturas y una

variedad de elementos semánticos. Luego, el concepto NoSQL fue reintroducido por varios investigadores, entre los más destacados se encuentran Eric Evans y Johan Oskarsson, ambos en 2009. Eric Evans se refirió a NoSQL para referirse específicamente a bases de datos no relacionales. Johan Oskarsson definió el término refiriéndose a la variedad de sistemas de bases de datos distribuidas y no relacionales que evolucionan constantemente [9].

En cuanto a sus orígenes en los diferentes entornos, la aparición de NoSQL se debe, en gran medida, a Cloud Computing o también conocida como “nube” de Internet.

Con las bases de datos NoSQL se pretende escalar servidores de bases de datos. Algo similar sucede con el concepto “Big data” (mencionado en la sección anterior *“Base de Datos a Gran escala”)* que es considerado como uso de las bases de datos NoSQL.

El concepto de NoSQL no está enfocado a un solo modelo de base de datos [10], ya que las bases de datos NoSQL se clasifican en diferentes tipos, como trataremos en el Capítulo III. De ésta manera el concepto también se identifica a un conjunto de otros tipos de base de datos, que no corresponde a las relacionales. Esto significa que las bases de datos NoSQL son clasificadas en cuanto a cómo es almacenada la información.

Actualmente, el término NoSQL se identifica en su mayor medida con las clases de bases de datos que no utilizan un RDBMS (relational database management system).

Ventajas y Desventajas

Se comienza a describir las ventajas y desventajas entre las bases de datos relacionales y las bases de datos NoSQL, teniendo presente que las comparaciones no se deben considerar en forma directa. Pues, el volumen de información y la problemática particular, son dos cuestiones en las que impactan directamente.

Se procede entonces a describir los beneficios y ventajas, para luego describir las desventajas y limitaciones de este modelo.

Beneficios y Ventajas de las BBDD NoSQL

Los sistemas de bases de datos NoSQL, responden en su mayoría a las necesidades de escalabilidad horizontal, cuando se trata de manipular con información que crece de manera exponencial. La existencia y el agregado de nodos (generalmente servidores) para distribuir el tráfico de la información, se caracteriza por ser sencillo. Debido a que es posible redirigir el tráfico de un nodo caído a nodos en funcionamiento, permitiendo de esta manera el trabajo normal de la red de información. La "filosofía" de este modelo, se basa en tener nodos que no tengan como impedimento lo económico en su adquisición, es decir, que básicamente esto último no sea un problema.

En cuanto a la implementación (a diferencia de las bases de datos SQL o relacionales), por ejemplo, no es necesario utilizar "joins" (las sentencias de una consulta se reducen considerablemente), sino que la información se organiza en los distintos servidores, dividiendo los datos, transparente para el desarrollador, mediante el uso de APIs.

Se dispone de amplia libertad para elegir, mejorar e innovar. Los desarrolladores pueden tener acceso (y cambiar) la lógica interna del manejo de datos en los sistemas, bajo las posibilidades que brinda el OpenSource [10]. Actualmente, se dispone de una amplia variedad de lenguajes de programación y motores de base de datos que soportan NoSQL.

Otra cuestión a destacar, es que existen diversos tipos de base de datos NoSQL, los cuales se describen en el Capítulo III – "Tipos de Bases NoSQL". Esto último ofrece un gran "abanico" de posibilidades para analizar de qué manera es más conveniente almacenar la información. En muchos casos, apuntado a la performance; a tal punto, que se podría llegar a tener diferentes BBDD NoSQL para distintos proyectos, de acuerdo a la necesidad particular.

En contraste con las bases de datos relacionales, la elección de optar por una u otra base de datos, está vista desde otro enfoque. Los distintos tipos de bases de datos NoSQL, impactan de manera directa en la elección del diseño de la base de datos, lo que se puede considerar como una ventaja, ya que a su vez, también brinda distintas maneras de representar la información. En las bases de datos relacionales ésto último no es posible. Existe un solo tipo (relacional) y, en cuanto a la elección, el análisis sólo podría quedar reducido a qué RDBMS utilizar, cuestiones de licencia (gratuito o pago), tipos de datos y funciones que soporta.

Desventajas y Limitaciones de las BBDD NoSQL

Como se ha mencionado en la introducción a este tema; las bases de datos NoSQL poseen diversas ventajas y beneficios, pero a su vez, no todo lo que ofrece abarca todas las necesidades, pues posee también diversos inconvenientes y limitaciones.

Las bases de datos relacionales proveen y garantizan ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) [11], en cambio, las bases de datos NoSQL no garantizan esta propiedad con lo cual se impide el uso de transacciones. Tampoco brinda el manejo de excepciones mediante fallos. [10]

Al priorizar por la disponibilidad y la inmediatez del dato para el usuario, trabajan con datos distribuidos, y en algunas ocasiones se podría contar con información replicada. Si el dato es muy volátil (como por ejemplo, la información que está constantemente en memoria caché), puede generar inconsistencia o distintas versiones para el mismo. [8]

Como la mayoría de los sistemas NoSQL no están diseñados para asegurar la integridad de datos, actualmente no se encontró la fiabilidad (consistencia) de las bases de datos relacionales. Sin embargo, en el caso de ser necesario, se podrían forzar modelos de consistencia fuertes bajo la presencia por ejemplo de restricciones de integridad, concurrencia, y múltiples actualizaciones simultáneas.

Es decir, se baja el nivel de fiabilidad (consistencia de información) en pro de una mejor disponibilidad del dato ante una consulta.

Todo aquel que haya trabajado en un nivel intermedio/avanzado con bases de datos relacionales, cuenta con muchas posibilidades para realizar transacciones. Algunos ejemplos: posibilidad de utilizar JOIN's, conjuntos de datos (group by), un completo manejo de fechas (intervalos, sumas, restas, comparaciones…), integridad de los datos, transacciones, excepciones, etc. Las bases de datos NoSQL no garantizan integridad de los datos e integridad referencial, están ausentes las constraints, índices y foreign keys.

En algunos casos (dependiendo la problemática particular) podría suceder que tarde o temprano se tenga que hacer algo más que leer y escribir datos, y la encargada de realizar esa tarea sería la aplicación (Java, PHP, Python, etc). Esto podría suceder por ejemplo, en un manejo específico de información. Con lo cual, la eficiencia del programa podría caer y todos los milisegundos que se habían ganado podrían volverse en contra.

Es una tecnología que actualmente carece de estándares, lo que acarrea diversas alternativas (como se ha mencionado en las ventajas mediante OpenSource).

Por ejemplo, SQL es un estándar abalado por el ANSI (Instituto Nacional EstadoUnidense de Estándares). Cualquier desarrollador/administrador de base de datos que denomine el estándar ”SQL-92” o “SQL2”, podrá manejar, casi en su totalidad, aquellas operaciones que brinda cualquier motor de base de datos (exceptuando algunas características particulares de cada motor, como por ejemplo, manejo de fechas, soporte a datos numéricos u otros tipos de datos). Esta característica no se presenta en base de datos NoSQL.

Existe en algunos casos, grados de incompatibilidad entre herramientas; cada base de datos NoSQL tiene sus propias interfaces y APIs. Por otra parte, al no tener un estándar, tampoco existe un soporte y documentación totalmente completa y confiable (más aún en herramientas no tan utilizadas).

Organización del trabajo

Este trabajo de investigación aplicada comienza describiendo el concepto de “Bases de Datos”, su evolución, su historia y luego se continúa con un enfoque orientado a las bases de datos a gran escala. Se mencionan las ventajas y desventajas que brindan las bases de datos NoSQL, con el fin de obtener una visión precisa al momento de tenerlas en cuenta como opción para trabajar con estas herramientas.

Se concluye este capítulo determinando el objetivo principal, el alcance del trabajo y la motivación acerca del tema.

A continuación se detallan los demás capítulos que conforman el trabajo de tesis:

- **Capítulo II – Estado del arte**

En este capítulo se realiza un informe específico respecto de las bases de datos NoSQL. Se comentan sus características más importantes, describiendo las razones que llevaron a aplicarlo desde sus comienzos hasta la actualidad. Se analizan los resultados que han obtenido las aplicaciones que utilizan las bases de datos NoSQL como parte de su funcionamiento. Además, se realiza una comparación entre ACID y BASE, para determinar la capacidad de las bases de datos NoSQL ante las propiedades más relevantes que debe cumplir un DBMS. Se concluye este capítulo, denominando algunos ejemplos de aplicaciones web conocidas que utilizan las bases de datos NoSQL, y explicando la visión innovadora que se pretende lograr en este trabajo.

● Capítulo III – Clasificación de los Tipos de Bases de Datos NoSQL

Se realiza un informe de la clasificación de los tipos de bases de datos NoSQL, en función de su modelo de almacenamiento. Se detallan las características de cada uno de estos modelos, se describe la forma en la que procesan y almacenan los datos, su estructura, las aplicaciones que utilizan cada tipo de base de datos y los lenguajes que utiliza NoSQL. Se muestran algunos ejemplos de cada modelo, describiendo las ventajas y desventajas de cada tipo de base de datos NoSQL.

● Capítulo IV – MongoDB

En este capítulo se realiza un informe acerca de la tecnología MongoDB, su importancia en el procesamiento de los datos y su forma de representar la información. Además, se detallan las ventajas, desventajas y usos prácticos actuales de dicha tecnología.

● Capítulo V – Escenario

En principio, se realiza la presentación del concepto Web Mining en conjunto con sus tipos de clasificaciones (Web Content Mining, Web Structure Mining y Web Usage Mining).

Dichos conceptos son abordados para presentar el nexo entre el concepto general Web Mining en relación con un modelo de servidor mencionado en la sección *Base de Datos a Gran Escala* del presente capítulo.

A continuación de presentar este nexo, se realiza un análisis detallado de los resultados obtenidos de las ejecuciones efectuadas sobre distintos modelos de almacenamiento, entre los cuales, se encuentra un modelo aplicado a una base de datos relacional (RDBMS) y el otro, aplicado a una base de datos NoSQL.

Las pruebas consisten en registrar los resultados y los tiempos de respuesta que se obtienen en cada operación asociada a reglas específicas sobre el modelo de servidor mencionado en la sección *Base de Datos a Gran Escala* del presente capítulo.

● Capítulo VI – Conclusiones y trabajo a futuro

En este capítulo se realiza un informe detallado (conclusiones) en base a las pruebas y resultados obtenidos que se realizaron el Capítulo V. Además, se menciona sobre las ventajas y desventajas en cuanto a las tecnologías utilizadas, detallando diversas cuestiones que serán útiles para ser abordadas en posibles futuros trabajos de investigación.

Alcance

Para una mejor comprensión se clasifica el alcance en:

- Ø Alcance Global
- Ø Alcance Específico
- Ø Fuera de alcance

● Alcance Global:

En este trabajo se abarcará los temas relacionados al modelo NoSQL, con una fuerte introducción al mismo, es decir, haciendo hincapié en sus utilidades, características y la relación con otras tecnologías con las que convive. Por otro lado, se enfocará en los diferentes tipos de sistemas de bases de datos NoSQL, analizando sus estructuras, sus funcionalidades y características principales que los identifica dentro del modelo NoSQL.

Luego, nuestras intenciones se abocarán en generar un escenario particular y pruebas del mismo, en los cuales se corresponderán con el alcance específico. Al finalizar, expondremos las conclusiones y ciertas características no detalladas en el presente trabajo, que según nuestra visión, permitirán establecer trabajos a futuro.

- **Alcance Específico:**

Se analizará el comportamiento puntual en una base de datos NoSQL y en una base de datos relacional, partiendo de un conjunto de datos considerable para un contexto reducido. Para esto, estableceremos un máximo de capacidad de 1GB en base de datos para al momento de realizar las pruebas.

Esto es muy útil, ya que al brindar esta característica, permite desde el punto de vista práctico; de una manera simple y específica, satisfacer el alcance de pruebas reales. Es simple, al menos por el hecho que cualquier interesado en el tema, podrá acceder al resultado de las pruebas.

La especificidad, proviene que mostraremos el resultado del comportamiento respecto del modelo relacional, de la manera más puntual y objetiva posible, teniendo como base un esquema de información que permita ser comparable en ambos modelos. Dicha justificación radicará en los fundamentos principales del modelo relacional y NoSQL, que se expondrán en el Capítulo 5(Escenario).

Las pruebas (escenario) se elaborarán mediante el siguiente marco, en el cual, se divide en los siguientes puntos:

Herramientas NoSQL:

Se utilizarán herramientas que soportan lenguajes que trabajan con la arquitectura NoSQL.

Las herramientas elegidas podrán ser compiladas y ejecutadas en cualquier SO, es decir que no se analizarán cuestiones en cuanto al comportamiento sobre diferentes SO.

Herramientas SQL:

Se abarcará un solo motor de base de datos: MySQL, por el hecho de ser opensource y simple, lo cual no influye en el momento de casos de estudio, ya que las herramientas NoSQL que se utilizarán, tendrán su propio almacenamiento de base de datos, independientemente del motor de base de datos que se optará.

Método de desarrollo:

- No hace al objetivo principal de estudio, será lo más simple posible y enfoca a un escenario puntual, solo se mostrarán entradas y salidas por un informe de log estándar.

- El lenguaje utilizado será Java con el sistema de base de datos MongoDB, debido a su simplicidad de uso y configuración, que no hace en la mayoría de los casos a los análisis de respuesta sobre ciertas características que se expondrán como pruebas en el Capítulo V: “Escenario y Pruebas”.

Volumen de Información

- Volumen actualmente considerable de datos para pruebas razonables en un contexto reducido, donde el límite que estableceremos será de 1 GB (GigaByte).

- El tamaño en razón de GB que se realizarán en las pruebas en los esquemas SQL y NoSQL diferirá naturalmente (no serán iguales), ya que ambos modelos implican la persistencia en disco de manera distinta y más aún como ya mencionamos, cuando algunas herramientas ni siquiera utilizan un motor específico, sino que tienen su propio almacén de datos como es la de Google AppEngine. Este parámetro nos será muy útil al momento de comparar.

- En los casos de estudio la cantidad de registros generados para cada modelo será la misma, con lo cual, el conjunto de datos, en cuanto a volumen es totalmente comparable. Es decir, si estamos almacenando 20.000.000 de personas, contaremos con esa misma cantidad para los dos modelos. Esto es trivial, ya que de otra manera la comparación entre ambos modelos no tendría relación alguna de estudio de comparación.

Hardware

PC/Notebooks con hardware de uso actual, ya sea de uso corporativo/hogareño.

Sistema

Windows (solo este S.O., razón explicada en “Herramientas NoSQL”).

Esquema/Modelos de Base de Datos:

En la actualidad, en la mayoría de los casos, no hay correspondencia directa de un modelo de datos NoSQL hacia un modelo de datos relacional, por el hecho de que son dos modelos distintos de almacenar información. Pero, se optará un modelo relacional que se asemejará en la mayor medida posible a lo que correspondería a su equivalencia en un modelo NoSQL. De esta forma al momento en que se realizará los análisis de comportamiento de cada uno y sus diferencias, se estarán estudiando casos de modelos que se aproximan en la mayor medida posible. También se permitirá obtener relaciones entre los resultados obtenidos.

Fuera de alcance:

- Concepto ACID (Atomicity, Consistency, Isolation and Durability): Atomicidad, Consistencia, Aislamiento y Durabilidad. Si bien algunos consideran que las bases de datos NoSQL aseguran ACID de manera parcial, esto no es totalmente correcto.
La gran mayoría de arquitecturas NoSQL frecuentemente aportan escasas garantías de consistencia, tales como de eventos o transaccional restringida a ítems únicos de datos. Estos temas serán abordados en el capítulo 2, en el punto ACID vs BASE. Es por eso que como la arquitectura básica del modelo no soporta ACID, se decide no ampliar en este tema.

- Comportamiento en cuanto a cambios de hardware como por ejemplo memoria RAM, disco, procesador, etc.

- Importación/Exportación de base de datos NoSQL.

- Comportamiento y/o diferencias de las bases de datos NoSQL en ambientes distribuidos y con respecto a los sistemas paralelos.

- Comportamiento de NoSQL en sistemas móviles.

    - Big Data y Cloud Computing con sistemas distribuidos.

    - Estudio y/o ejemplo de comportamiento en cuanto a concurrencia.

- Comparación entre los distintos modelos MongoDB, Cassandra, Google AppEngine, etc.

- Estudios específicos de conceptos de base de datos relacionales y su equivalencia exacta con el modelo NoSQL, ya que en su mayoría no existen correspondencia exacta de conceptos entre ambos modelos. Ejemplos: formas normales, triggers, transacciones, etc.

- Análisis y utilización con motores de base de datos que no sean MySQL (razón explicada en "Alcance Específico").

- Desarrollo Web o Aplicación de escritorio en el escenario

CAPÍTULO II – Estado del Arte

Estado del Arte

En esta etapa, se pretende mostrar un enfoque que aporte sentido al trabajo de tesis. Se comenzará en primer lugar, con un análisis general, para luego centralizarse en lo que respecta en particular a las bases de datos NoSQL ("Not only SQL)"; que representan bases de datos de gran escalabilidad (consideradas como una alternativa al modelo relacional) y pensadas para manipular datos que crecen exponencialmente.

El modelo NoSQL data de aproximadamente una década atrás. Si bien es un concepto actual y moderno, el concepto "NoSQL" (a veces llamado "no solo SQL"), es una clase amplia de sistemas de gestión de bases de datos que difiere en distintos aspectos del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS); por no utilizar SQL como el principal lenguaje de consultas. Como se ha mencionado en el Capítulo I, los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad). Asimismo, debe tenerse presente, que habitualmente escalan horizontalmente.

Desde el momento en que los sistemas de bases de datos NoSQL comenzaron a tener repercusión, la idea básica fue al menos disponer de ellos como una alternativa más de almacenamiento de información. Su objetivo central es ofrecer mejores tiempos de respuesta en comparación a los sistemas de bases de datos relacionales (RDBMS). Además, el impulso que tuvo como modelo de base de datos, arrojó como consecuencia que se lo compare con las bases de datos relacionales. Las comparaciones entre ambos tipos de bases de datos, no se enfocan solamente por lograr un mejor tiempo de respuesta o tiempo de acceso, pues existen otros factores tenidos en cuenta al momento de elegir uno u otro. Estos factores están relacionados con la información a almacenarse o el sistema de información que se desea modelar. Por ejemplo, las bases de datos relacionales tradicionales además de permitir ACID, nos permiten definir la estructura de un esquema que demanda reglas rígidas, lo cual seguramente puede satisfacer las necesidades por ejemplo de un sistema bancario. Sin embargo, las aplicaciones web modernas presentan desafíos muy distintos a los que ofrecen los sistemas empresariales tradicionales. Ellos son, generalmente, garantizar datos a escala web, alta frecuencia de lecturas y escrituras, cambios de esquema de datos frecuentes. En el caso de las aplicaciones web sociales, probablemente no necesiten el mismo nivel de ACID que por ejemplo, un sistema bancario.

Cabe mencionar también, que según artículos y publicaciones donde se ha entrevistado a profesionales de la informática de diferentes universidades [13], se puede inferir, la aplicabilidad del sistema de bases de datos NoSQL en los sistemas Web masivos actuales. Algunos de ellos son las redes sociales, como por ejemplo, Twitter, Facebook, Google+, Linkedin, Yahoo, cuyas implementaciones en la manipulación de datos son mediante base de datos NoSQL. Esto es debido a que el incremento de información crece exponencialmente, no sólo a nivel base de datos, sino también a nivel almacenamiento de archivos de cualquier tipo. En consecuencia, se estima que la velocidad es uno de los puntos fundamentales. Es por eso, que este concepto empezó a despegar y mostrar toda su utilidad en estos sistemas de redes de información.

Influencias de NoSQL en las tecnologías

En lo que se refiere al avance de las tecnologías, se debe señalar que se han abierto las puertas hacia un nuevo enfoque de entendimiento y toma de decisiones, las cuales se utilizan tanto para datos estructurados como no estructurados. De esta manera, surgen las infraestructuras Big Data, Cloud Computing y Cloud Database. Y, a su vez, diversas plataformas tecnológicas que se acoplan a ellas, como lo son: HBase, Hadoop y MapReduce (estos dos últimos se tratarán en el Capítulo III).

Tanto las infraestructuras como las plataformas tecnológicas mencionadas, surgen por diversos factores. Entre ellos, el crecimiento de información de manera exponencial (como se menciona en la sección del *Capítulo I - Beneficios y ventajas de las BBDD NoSQL*). Asimismo, este crecimiento de información impacta en el paradigma de la arquitectura computacional (principalmente distribuida y paralela) y en los mecanismos de procesamiento de datos a gran escala. Las principales herramientas tecnológicas en las que NoSQL se destaca en estas dos cuestiones son: Big Data y Cloud Database. En consecuencia, se procederá a continuación, a definir estos conceptos, relacionando las influencias y usos de NoSQL a las referidas herramientas tecnológicas.

Big Data

El concepto BigData, tuvo sus orígenes en ciencias como el análisis del negocio, la astronomía y otras diversas ramas científicas, como por ejemplo, las médicas; donde se acuñó el término *Big Data* como “datos masivos” [13]. La compañía SAS (Statistical Analysis System, una de las más destacadas empresas en desarrollar soluciones software y servicios de analítica empresarial), define Big Data como un término para *describir el crecimiento exponencial y la disponibilidad de información, tanto estructurada como no estructurada. Abocada a su vez para el negocio y la sociedad* [14].

Se entiende que en principio, no existe una rigurosa definición de los datos masivos, ya que ello depende principalmente, del escenario y del tipo de información. Sin embargo, la información estructurada y sobre todo no estructurada, de al menos del orden de terabytes puede considerarse como Big Data. Variedades de herramientas dedicadas a investigar temas relacionados con la gestión de información, afirman sobre este orden de capacidad. Ejemplo de ello, es la metodología para administrar información empresarial opensource MIKE 2.0 [14]. Si bien inicialmente se centró en torno a los datos estructurados, su objetivo es el de proporcionar una metodología completa para cualquier forma de desarrollo de información.

El concepto de NoSQL, se encuentra relacionado directamente con las diversas herramientas de mercado (ya sean por ejemplo APIs o lenguajes de programación), que permiten manipular información de tipo Big Data [15]. De esta manera, a continuación se hará mención sobre algunos de los productos con relevancia en los que se destaca BigData, con base de datos NoSQL:

- Varias compañías de la talla de Oracle, ya cuentan de manera oficial con su propia implementación de base de datos NoSQL para el almacenamiento de información del estilo Big Data. La plataforma se denomina “Big Data Appliance” [16]. La implementación se corresponde al tipo clave-valor. Con esta base de datos, Oracle promete una baja latencia para acceder al dato, teniendo en cuenta el orden de petabytes de información.

- La compañía Teradata que se dedica a soluciones analíticas y Data WareHouse, también logró acoplarse a NoSQL. Para esto último, NoSQL impuso mediante MongoDB una alianza con Teradata, permitiendo integrar sus sistemas a través del desarrollo de una componente basada en JavaScript Object Notation (JSON). De esta forma, Teradata afirma que se permite a los usuarios o cualquier subsistema incorporar sus datos de una manera más sencilla y mejorar las operaciones con inteligencia estratégica [17].
- En el ámbito Business Intelligence, la herramienta opensource Pentaho permite la integración de información para la toma de decisiones de las organizaciones. También abarca un ambiente visual para modelar, visualizar y explorar conjunto de datos almacenados en base de datos NoSQL [18].

□ Cloud Database

El concepto Cloud Database, se refiere a las bases de datos que se ejecutan en plataformas de Cloud Computing. Previamente, para comprender y definir el término Cloud Database se debe mencionar el significado de Cloud Computing, que abarca varios conceptos además de los relacionados con base de datos. Atendiendo a la definición dada por el NIST (National Institute of Standards and Technology), Cloud Computing es un modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red de un conjunto compartido de recursos. Donde los mismos son recursos compartidos (por ejemplo: redes, servidores, equipos de almacenamiento,

aplicaciones y servicios), que pueden ser aprovisionados y liberados con un esfuerzo de gestión reducido o interacción mínima con el proveedor del servicio [19]. Otra definición complementaria es la aportada por el RAD Lab de la Universidad de Berkeley, desde donde se explica que el Cloud Computing se refiere tanto a las aplicaciones entregadas como servicio a través de la Web, como el hardware y el software de los centros de datos que proporcionan estos servicios. Los servicios anteriores son conocidos como Software as a Service (software como servicio, SaaS) [20].

Cloud Database se refiere a las bases de datos que se ejecutan en plataformas de Cloud Computing. La filosofía de Cloud Database, es tomar el concepto de base de datos como servicio (de la misma manera que se ha mencionado para Cloud Computing). Con lo cual, dicho servicio debe ser mantenido por un proveedor en la nube (o cloud). A su vez, cada proveedor debe brindar alguna herramienta de acceso, así es como por ejemplo, ClearDB ofrece al motor de base de datos MySQL [21].

De las bases de datos disponibles en la nube, algunas utilizan el modelo SQL, o bien híbrido (SQL y NoSQL). Las más populares que utilizan NoSQL son: Google BigTable y Amazon SimpleDB. La influencia de NoSQL en este caso es la utilidad y su evolución constante en diferentes APIs, donde una base de datos es pensada como un servicio de persistencia.

Características de NoSQL

Como se ha mencionado en la sección del Capítulo I *"Ventajas y Desventajas del modelo"*, mientras las tradicionales bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones ACID, las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas (no imponen un esquema pre-fijado de tablas). También en dicha sección, se mencionó que estas bases de datos no proveen ACID, con lo cual las actualizaciones son una de las características más complejas del modelo. Ya que, por ejemplo, no es posible propagar de manera transaccional las actualizaciones de múltiples nodos en todo el sistema.

Adicionalmente, en el Capítulo I, se ha comentado la cuestión de escalabilidad enfocado a la escalabilidad horizontal, no solo aplicable para sistemas NoSQL sino también para Cloud Computing. Se añaden más recursos (como por ejemplo, puede ser un disco duro o añadir más máquinas) a un nodo particular dentro de un sistema. Este concepto de escalabilidad horizontal se dirige hacia la alta disponibilidad del hardware; en la mayoría de los casos, no es confiable, con lo cual, es necesario ampliar recursos.

Además de la carencia de un esquema predeterminado, la principal característica de las bases de datos NoSQL, es que están pensadas para manipular cantidades de información que crecen de manera muy rápida y de cantidad exponencial. También proveen diversos tipos de esquemas de representación de la información, como lo son, el esquema orientado a documentos, clave/valor, grafos y

columnas. Estos esquemas se tratarán en el Capítulo III, en ese sentido se puede afirmar que provee flexibilidad de esquemas.

En NoSQL, generalmente los datos son recuperados de manera más rápida que en un RDBMS. Sin embargo, las consultas que se pueden hacer son más limitadas y requieren trasladar complejidad a la aplicación. Es decir, para aquellas aplicaciones que generan informes empleando consultas complejas, NoSQL a priori no sería inmediatamente adecuado.

Con respecto a la arquitectura de las bases de datos NoSQL, a menudo ofrecen garantías de consistencia débiles, como por ejemplo, eventual consistency, o transacciones restringidas a elementos de datos simples. Las bases de datos NoSQL emplean una arquitectura distribuida, donde los datos se guardan de modo redundante en distintos servidores, la mayoría de las veces, usando tablas hash distribuidas.

En relación a lo mencionado en el Capítulo I, en la sección de ventajas y desventajas, afirmamos que los principales problemas de las bases de datos NoSQL son su falta de madurez (se desconoce su reacción manipulando datos de sistemas con antigüedad mayor a 10 años) y su complejidad (los modelos de datos usados, la instalación de algunas bases de datos NoSQL y las consultas).

El blog Facility9 [22], se refiere a un proyecto creado por un conjunto de investigadores de nombre sobre el comportamiento de las bases de datos. En dicho blog, se refiere básicamente a NoSQL con respecto a RDBMS convencionales. Se debe mencionar que el uso de las bases de datos NoSQL sorprende, ya que muchos departamentos de TI siguen evaluando pasar de SQL Server 2000 a SQL Server 2005, es decir, bases de datos relacionales; mientras que otras empresas buscan bajar costos y mejorar el rendimiento. Por este motivo, estudian los beneficios de la aplicación y el uso de las bases de datos no relacionales, agrupándolos de la siguiente forma:

- **Analíticas:**

Una de las razones para considerar utilizar una base de datos NoSQL a nivel empresarial y de negocio, es que muchas bases de datos NoSQL son muy adecuadas para la realización de consultas analíticas. Los desarrolladores pueden utilizar los mismos lenguajes de consulta para realizar consultas analíticas en vez de realizar consultas atómicas. Normalmente esto será alguna variación de una query MapReduce. Los términos Map Reduce intentan decir "agrupar por y seleccionar" al estilo "group by / select" de SQL, aclarando esto último, para no confundir a las personas que están acostumbradas al lenguaje SQL. Más adelante, específicamente en el Capítulo III, se detallará más en profundidad los conceptos de Map Reduce.

Muchos sistemas NoSQL cuentan con muy buen rendimiento de escritura. Por ejemplo: Redis. Esta es una base de datos NoSQL que se destaca por su rendimiento altamente elevado; pues se considera una base de datos en memoria con persistencia para datos. Además, consume muy pocos recursos (utiliza aprox. 7.5MB de disco en ejecución). [24]

□ Escalabilidad

Las Bases de datos NoSQL están diseñadas para escalar (principalmente horizontalmente como se ha mencionado en el presente Capítulo). Es una de las principales razones por las que se elige optar por una base de datos NoSQL. Por lo general, con una base de datos relacional como SQL Server u Oracle, la escalabilidad es lograda con la compra de servidores y de almacenamiento más grande y más rápido, o con el empleo de especialistas para proporcionar un ajuste adicional. A diferencia de las bases de datos relacionales, las bases de datos NoSQL están diseñadas para la escalabilidad en forma transparente. En este aspecto, están preparadas para mantener la misma velocidad de respuesta independientemente de la capacidad utilizada de almacenamiento.

□ Disponibilidad

La disponibilidad de un sistema es el resultado de la disponibilidad de los componentes requeridos para una tarea específica. Las bases de datos NoSQL se construyen principalmente sobre la base de alcanzar alta disponibilidad y escalabilidad, a través de la distribución sobre hardware accesible o económico (commodity).

□ Redundancia

Las bases de datos NoSQL también están diseñadas para trabajar con datos replicados, lo cual puede entenderse como tener redundancia en la información. Esta característica la hemos mencionado en el *Capítulo I -Ventajas / Desventajas* del modelo NoSQL. Ahora bien, el hecho de contar con redundancia de información no implica que sea una falla grave de las bases de datos NoSQL, como sí podría serlo en las bases de datos relacionales (en el caso que exista algún escenario en que se pueda presentar).

□ Flexibilidad

Si bien los problemas de representación de datos son muy diferentes en bases de datos NoSQL, hay una flexibilidad en cómo se almacenan los datos para no comprometer los tiempos de ejecución y los tiempos de respuesta. Bases de datos NoSQL, del estilo de Bigtable y Cassandra, son flexibles al momento de almacenar los datos en el disco. Por ejemplo, es posible crear familias de columnas derivadas, favoreciendo la velocidad en respuesta de aquellas consultas más frecuentes. Las basadas en el modelo de Bigtable también tienen otra ventaja, fuera de la estructura fundamental, que es posible almacenar una variedad de datos dispares. Otro ejemplo de flexibilidad la constituyen las bases de datos del tipo clave-valor en cuanto a la

información que manipulan. Los datos se almacenan como un valor arbitrario, es posible almacenar imágenes, documentos de texto, cadenas, números enteros, y los objetos serializados dentro de la misma base de datos. Esto requiere una mayor responsabilidad y el pensamiento creativo por parte de los desarrolladores de aplicaciones y arquitectos (lo cual es una desventaja), pero también permite a los diseñadores de software, una solución personalizada que puede cubrir sus necesidades (lo cual es una ventaja).

□ Desarrollo Amplio Distribuido (Cloud Computing)

El desarrollo de sistemas en forma distribuida está relacionado al concepto Cloud Computing, término que ya se ha expuesto en el presente capítulo en la sección "Influencias de NoSQL en las tecnologías".

Si bien las bases de datos NoSQL son independientes a la arquitectura de procesamiento, se están utilizando en diferentes ambientes y hacia un foco específico, como lo son: almacenamiento en clúster, servidores replicados o distribuidos, refactorizaciones y migraciones de datos.

Respecto a cloud computing, existen algunos fundamentos a tener en cuenta al momento de evaluar soluciones NoSQL en memoria:

- **Escalabilidad Automatizada**: Pues, una solución a un problema debería ser lo más escalable posible, dado que no se debería trabajar con nodos, cluster ni operaciones escalables, ya que la solución debería ser transparente.
- **Evitar la pérdida de datos**: Teniendo en cuenta que los nodos en memoria pueden presentar anomalías (por ejemplo, una falla en un servicio web), en estos casos, podrían perderse todos los datos almacenados en ellos. La solución debería incluir almacenamiento persistente, tolerancia ante fallos y capacidades de backup.
- **Asegurar alta performance**: Para ello, se sugiere utilizar servidores y procesar los datos de manera distribuida. Por ejemplo, Google Apps que incorpora desde un navegador hasta el almacenamiento de datos en sus servidores. De esta manera, los programas se ejecutan en los servidores, mientras que el acceso a los servicios e información se realiza desde la Web. [24]
- **Mejorar la administración**: Utilizando, en la mayoría de las operaciones (por ejemplo: Upgrades, clustering y recuperación ante fallas) que deberían automatizarse completamente.
- **Coste acorde a la necesidad**: Ello se lograría, buscando la mejor relación carga/tiempo de información. Muchas soluciones cuentan con varios servidores cluster, por lo cual termina siendo muy superior la capacidad y el costo, que lo necesario para almacenar la información.

Utilidades de una base de datos NoSQL

El uso de una base de datos NoSQL se debería plantear a partir de la cantidad de datos, operaciones de lectura y escritura, como así también sobre la disponibilidad de información a brindarse en el orden de millones de usuarios sin perder rendimiento. Para solventar algunas soluciones específicas, es posible recurrir a sistemas mixtos que combinan los clásicos sistemas relacionales (en general, fácilmente manipulables con el lenguaje SQL) con soluciones NoSQL, para aquellas funcionalidades que requieren millones de consultas en tiempo real. Por ejemplo, Microsoft hace hincapié en las bases de datos híbridas que combinan datos tanto en las instalaciones del cliente como en la nube Azure a través de SQL Data Sync. Esta empresa, tiene un servicio de nube alojada en base de datos NoSQL llamada Tables, mientras que Blobs (almacenamiento de objetos binarios) se ha optimizado para archivos multimedia, como por ejemplo, audio y video [31]. De esta manera, se puede afirmar que las bases de datos NoSQL no fueron construidas con el fin de rivalizar a las bases de datos relacionales, en cuanto a rendimiento, integridad y escalabilidad; pues solamente vienen a ofrecer una funcionalidad que las bases de datos relacionales, por definición, no pueden soportar.

El sistema tradicional que utiliza el modelo relacional, también soporta el almacenamiento, lectura y escritura de una considerable cantidad de información, por ejemplo del orden de terabytes. Pero, las bases de datos NoSQL resultan más interesantes en este aspecto. Es por ello, que aplicaciones de la talla de Wikipedia y Twitter se hayan planteado en la migración de sus servicios a estas mismas. [26]

Existen cientos de aplicaciones funcionando con bases de datos NoSQL, por ejemplo, todas las de la plataforma GAE (Google App Engine). Desde este punto de vista, funcionan como una caché persistente conjugable con bases de datos SQL tradicionales, según el requerimiento específico. Así mismo, las bases de datos NoSQL tienen un contexto particular y no son aplicadas a todas las empresas y compañías. Es por esto, que lo utilizan aquellas que tienen y manipulan volúmenes de datos que crecen de manera exponencial. En estos casos, es optada para resolver los problemas de cuellos de botella que se originan dando mayor escalabilidad horizontal.

En cuanto a la utilidad de los sistemas NoSQL sobre diferentes ámbitos, podemos mencionar como más relevantes a las redes sociales, desarrollos de software (enfocados en su mayoría a la web) y Big Data. A continuación, se procederá a detallar cada uno de ellos:

Redes Sociales

Actualmente, para una red social, una base de datos NoSQL es altamente recomendable. Debido a las redes sociales, esta tecnología comenzó a despegar y mostrar utilidad en el campo de la informática y la estadística. Las redes sociales más conocidas, que usan NoSQL son Facebook, Twitter, Google+ y Linkedin. Facebook, creó en 2007 una herramienta, denominada Cassandra, para poder potencializar la

búsqueda en su inbox, que actualmente es mantenida por Apache. Luego, la referida herramienta fue utilizada por otras redes sociales. Utiliza DHT (tablas de hash distribuidas), también conocidas como una clase de sistemas distribuidos descentralizados tipo hash, donde pares (clave-valor) son almacenados y cualquier nodo puede recuperar de forma eficiente el valor con una clave (key). Esto permite que las DHTs puedan escalar a cantidades de nodos a un número muy considerable. Para tener una idea de esa cantidad, podemos estar refiriéndonos tranquilamente de clusters de más de 100 TB (terabytes) de información en un orden de cientos de máquinas.

Linkedin, utiliza Voldemort desde el 2009 [23], como herramienta para acceso a bases de datos NoSQL, bajo licencia Apache. Es un sistema distribuido de tipo clave valor. Según la página web del proyecto, esta herramienta es usada para resolver algunos problemas de alta escalabilidad para Linkedin, donde el simple particionamiento funcional no es suficiente. No es usada como una única base de datos, ya que es un proyecto que está bajo desarrollo, y requiere de contribuciones de reporte de ideas, bugs y parches.

Google+ desde el 2004, utiliza Big Table [27], donde ellos mismos fueron los creadores como un sistema de gestión distribuido de base de datos. Es un mapa distribuido multidimensional. Está construido sobre GFS (Google File System), en el que mediante el formato de archivo SSTable (que es un mapa clave-valor ordenado) se realizan la persistencia de la información.

□ Desarrollos

Debido a que en muchos casos la información crece de manera exponencial, particularmente en la Web y el tiempo de acceso al dato es un punto fundamental; la tendencia a la elección es NoSQL. Como consecuencia, existen variados tipos de usos para el desarrollador, al momento de realizar tal elección. En éste ámbito, los usos en los que se recomendaría evaluar la misma (relegando un poco el concepto directo sobre “red social” de información que habitualmente se asocia a NoSQL); son: administración de catálogos, sistemas que realicen lógicas aplicadas a inventarios de productos (por ejemplo, carritos de compra); como asimismo administración de historiales de cierta información específica. A continuación se detallarán las herramientas tecnológicas, las cuales son consideradas las más relevantes para el desarrollador.

Google AppEngine (como se ha mencionado en el presente Capítulo al definir “Cloud Computing”), es una plataforma considerada como un servicio que utiliza tecnologías que son familiares para el desarrollo (por el hecho que soporta lenguajes populares como PHP y Java). Y, además, utilizan la misma infraestructura que Google. Provee una base de datos, definida por Google como una base de datos NoSQL, que utiliza el almacenamiento de “Google Cloud Datastore”, basado en HRD (Google App Engine High Replication Datastore). HRD pretende replicar la información en múltiples centros de datos para obtener una muy baja latencia para acceder al dato. Google afirma y promete con esta plataforma: alta disponibilidad en lecturas y escrituras, consistencias fuertes, soporte de monitores por ingenieros de Google [27].

En cuanto a servicios web, existen web services comerciales, un ejemplo es el de Amazon Web Service (AWS). También conocido como SimpleDB. IBM lo considera como alternativas de sistemas NoSQL [28]. Es una colección de servicios web, que en su conjunto son considerados una plataforma en la nube, ofrecidas a través de *Amazon.com*. Utilizan la estructura de clusters en las bases de datos de los servidores.

Referido a nivel de integración y facilidad de manipulación de APIs, en algunos casos esto podría impactar directamente en la complejidad de la adaptación de la base de datos NoSQL al sistema de información. Como, por ejemplo, Cassandra con lenguaje Java, o MongoDB con PHP.

Por último, haciendo mención a otras perspectivas hacia soluciones sobre problemáticas específicas para un desarrollador, podemos notar que en varios casos particulares, se encuentra que NoSQL puede ser una alternativa de uso para administrar y hacer más ágil el uso de cierto tipo de información. Por ejemplo, el uso de logs, como lo son los de Apache o MySQL. Al ser no transaccional, para este tipo de enfoque suele ser utilizado en diversos casos particulares. Una posibilidad para su implementación puede ser almacenar en una base de datos de tipo clave valor toda la información de un logueo de una aplicación, donde la clave puede ser la fecha y hora, y el valor el conjunto de detalles de archivos asociados.

□ Big Data

Como se ha mencionado en "*Influencias de NoSQL en las tecnologías"* del presente capítulo, NoSQL es también una opción muy recomendable cuando se manipula con volúmenes de información del orden de BigData.

En cuanto al ámbito de negocio y Business Intelligence, en algunos casos particulares, NoSQL promete ser una posibilidad que hoy en día puede ser comparable con mucha imposición sobre las base de datos relacionales. Esto ha cambiado considerablemente en la última década; debido a las ventajas que se han mencionado en el Capítulo I: escalabilidad, alta disponibilidad en tiempo de acceso al dato, diversidad de elección en tipos de base de datos y evolución constante. Y más aún, sumado por ejemplo, que existen grandes compañías como Oracle que cuentan con su propia base de datos NoSQL.

Desde un enfoque hacia otros sistemas web, usos comunes también pueden ser sistemas de tiempo real, donde la baja latencia en tiempo es crítica. Un ejemplo típico para este caso pueden ser los juegos online masivos en red. En los desarrollos de la programación, se cuentan con herramientas que son utilizadas específicamente, como pueden ser Cassandra o MongoDB [29], entre otras. Actualmente, en la medida que las herramientas avanzan en su funcionalidad, se realizan diversas conferencias y debates, al respecto de las utilidades, ventajas, desventajas y actualidades de NoSQL.

También sobre otros distintos aspectos más amplios y específicos, a los mencionados anteriormente; así como también relacionada al enfoque empresarial y del negocio.

ACID vs BASE

En lo que abarca fuera del alcance, se ha expuesto en el Capítulo I, que no se analizan las características ACID ya que las mismas no son garantizadas por las bases de datos que utilizan NoSQL. Si bien el concepto ACID es meramente para bases de datos relacionales, debemos considerar diversos conceptos asociados al mismo pero para base de datos NoSQL, ya que siguen en constante surgimiento y evolución.

Existe un acrónimo análogo a ACID para NoSQL que prima disponibilidad frente a consistencia. Dicho concepto es BASE (Basically Available, Soft state, Eventual consistency).

- Basically Available (Disponibilidad Básica): el sistema responderá aún con datos viejos.
- Soft State (Estado de Soft): el estado puede que no sea estable y puede ser corregido.
- Eventually Consistency (Consistencia Eventual): se garantiza que al cabo de un tiempo el sistema quedará en un estado consistente, es decir, que eventualmente se conseguirá un estado consistente.

La consistencia eventual es relativamente nueva, pero dicho concepto no lo es como idea [30]. Generalmente, BASE está asociado con las bases de datos NoSQL. Variados servicios, como Google App Engine DataStore y Amazon S3 [28], poseen consistencia eventual.

Previamente a BASE, existe un teorema, el teorema CAP (Consistency, Availability, Partition Tolerance). Dicho teorema no está enfocado específicamente a base de datos NoSQL, sino a las características de garantías de almacenamiento en sistemas distribuidos. No obstante, se debe mencionar, ya que son conceptos que luego se utilizaron y derivaron al estudio de las garantías de almacenamiento de base de datos NoSQL.

Las siglas significan:

- Consistency (Consistencia): todos los nodos vean la misma información al mismo tiempo.
- Availability (Disponibilidad): la garantía de que cada petición a un nodo reciba una confirmación si ha sido o no resuelta satisfactoriamente.
- Partition Tolerance (Tolerancia a Fallos): que el sistema siga funcionando a pesar de algunas pérdidas arbitrarias de información o fallos parciales del sistema.

El teorema establece que es imposible para un sistema distribuido garantizar simultáneamente estas tres características, es decir, no puede tener más de dos de

estas tres características simultáneamente. Por eso se dividen sistemas distribuidos de acuerdo a estos términos:

AP (Disponibilidad y Tolerancia a fallos): Cassandra y CouchDB.
CP (Consistencia y Tolerancia a Fallos): HBase y Paxos.
CA (Tolerancia a fallos y consistencia): RDBMS.

Niveles de consistencia para Servicios de Base de Datos NoSQL

También se puede optar por el nivel de consistencia con diferentes características que a continuación describiremos, actualmente ofrecidas por ejemplo por Amazon's SimpleDB (servicio de base de datos). Esto impacta en los sistemas NoSQL, como así también en Big Data y Cloud Computing.

Windows Azure Storage [31] (utilizado para Cloud Computing), provee un tipo llamado "consistencia inmediata". Este concepto también es llamado "consistencia fuerte" o "consistencia estricta". Significa que se puede leer independientemente si está escribiendo y que además todas las operaciones de lectura deben obtener las últimas modificaciones de una operación completa de escritura. Esto implica que las operaciones de lectura y escritura para un conjunto determinado de datos, deben ser ejecutadas en el mismo nodo, o bien que la consistencia es real en un escenario de transacción distribuido. Pero, una consistencia con estas restricciones, no siempre puede ser asegurada con lo mencionado anteriormente por el teorema de CAP, en lo que respecta a disponibilidad y tolerancia a fallos. Es justificable que pueda suceder que no se cumplan estas restricciones, por el simple hecho que las operaciones de escritura y lectura, podrían residir en otro nodo y no siempre impactar en el mismo.

La consistencia eventual (que también hemos mencionado en el presente capítulo, cuando se definió CAP), actualmente provee variados tipos de garantías para las consistencias eventuales. No nos detendremos en cada una de ellas, ya que el concepto CAP está enfocado específicamente en lo que respecta a sistemas distribuidos y es por eso que no se incluirá dentro del contenido del trabajo de tesis.

Por último, se puede afirmar que la existencia de los distintos niveles de consistencia permite esclarecer la idea de que existen bases de datos NoSQL que soportan consistencia eventual; y que en algunos casos soportan algún nivel de atomicidad, pero nunca garantizable.

CAPITULO III – Clasificación de tipos de bases de datos NoSQL

Introducción

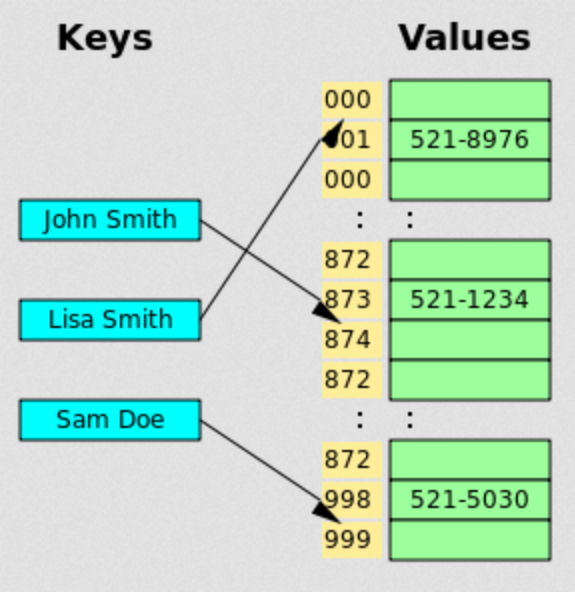
En este capítulo se describe la clasificación de los tipos de bases de datos NoSQL con el objetivo de entender su definición, mostrar algunos ejemplos, detallar sus características principales, estructura, utilidades e implementaciones actuales. De esta manera, aportará en el conocimiento al momento de elegir alguna de ellas para aplicarlo a un contexto determinado.

Bases de Datos Orientadas a Clave/Valor

Definición

Los sistemas de base de datos clave/valor son la implementación más básica y fundamental de los tipos NoSQL, debido a que estas bases de datos operan de forma similar a un diccionario para mapear claves a valores y así no necesitar estructura alguna ni tampoco una relación. Almacenan registros que contienen una clave y su valor. Cuando se tiene que recuperar un dato, se busca por su clave y se recupera el valor. [34]

A continuación se muestra la forma de almacenamiento de estos sistemas. A cada clave se le asocia un valor (clave = valor):



Origen

El precursor de las bases de datos NoSQL orientadas a clave/valor fue Amazon Dynamo, basadas en DHT (Distributed Hash Tables). [32]
En el año 2007 se publicó un paper con el diseño y la especificación a grandes rasgos de Dynamo, rompiendo con conceptos como la consistencia o modelo relacional. Su objetivo se diseñó en base a escalabilidad y disponibilidad, ambos términos definidos en el *capítulo 2- "Características de NoSQL"*.
A lo largo del paper "**Dynamo: Amazon'sHighlyAvailable Key-value Store**" [33] se especifica esta base de datos como una gran tabla de hash distribuida (DHT) y accesible mediante un mecanismo de "Clave-Valor".

Características

La naturaleza poco estructurada de las bases de datos NoSQL orientadas a clave/valor asegura una buena escalabilidad. [36] Los sistemas de bases de datos NoSQL orientados a clave/valor ofrecen un buen rendimiento (por ejemplo en consultas y escrituras) a cambio de renunciar a las funcionalidades ya mencionadas en el *"Capítulo I – Desventajas y Limitaciones de Bases De NoSQL"*. Las validaciones de los datos se delegan completamente en la aplicación cliente, siendo la base de datos

simplemente el lugar donde se guardan los datos. No se verifican integridades, todo esto se ha de implementar a nivel de aplicación en el código del cliente. [37]

Estructura

Un sistema clave/valor está formado por contenedores, también se les llama cabinets, donde se pueden almacenar tantos pares clave/valor como se requiera. En cada contenedor se pueden guardar datos del mismo tipo o familia (por ejemplo: productos, pedidos, clientes) o totalmente diferente (se podría tener un contenedor por cliente). Ambas representaciones dependerán de cómo lo requiera la aplicación cliente. [5]

Las bases de datos clave/valor ofrecen un modelo de consultas limitado que puede imponer costes de desarrollo y requisitos a nivel de aplicación para ofrecer un modelo de consultas más adecuado. Un ejemplo de esto son los índices, que deben ser gestionados por el propio usuario o programador. [38]

Utilidades

Las bases de datos orientadas a clave/valor son recomendables para usarse en casos donde se necesite mejorar el tiempo de respuesta en el acceso a los datos, o bien cuando se tienen muchos campos en una sola estructura (tabla) que requieren ser procesados una y otra vez y tienen valores cambiantes, por ejemplo, lista de los productos más vendidos, gestión de sesiones, rango de venta y catálogo de productos. [37]

Algunos esquemas de almacenamiento que utilizan clave/valor son los siguientes:

- Oracle Berkeley DB: es un almacenamiento puro que surgió a mediado de los ’90, donde claves y valores son array de bytes aunque no abarca el significado puro de clave/valor, sino que trabaja con pares de array de bytes y retorna el mismo retorno en la llamada cliente. Permite “cachear” información en memoria. [39]
- EH Cache: robusto sistema de caché distribuido, open source. Es considerado como una solución NoSQL y es usado en aplicaciones Java. [40]
- MemCached: es un almacén de tipo clave-valor en memoria para los datos arbitrarios (como strings y objetos) de los resultados de las llamadas a bases de datos, llamadas a la API o la renderización de páginas. [41]

Implementaciones

Algunas de las implementaciones de bases de datos orientadas a clave/valor son las siguientes:

- **Redis**: es un motor de base de datos libre de tipo clave-valor (key-value) persistente, que residen en memoria RAM y posteriormente vuelca el conjunto de datos almacenados al disco duro. Redis es cliente/servidor por lo que levanta su servicio y responde peticiones, cuenta con interfaz de red lo cual hace posible conectar clientes o nodos desde otros host. Redis es key/value esto significa que solo entiende Claves y Valores, es decir, cadenas, listas, hash, conjuntos y conjuntos ordenados, de igual manera Redis provee mecanismos para manipular y consultar estas estructuras de datos. Al no pertenecer al mundo relacional automáticamente entra en el grupo de bases de datos libres de tipo **NoSQL** (Not Only SQL). [42]

- **DynamoDB**: es una base de datos NoSQL que ha desarrollado y probado internamente Amazon. DynamoDB puede ser gestionada desde la consola de administración de AWS (Amazon Web Services). Desde esta consola se podría crear una nueva tabla DynamoDB, aumentar o disminuir los recursos, según las necesidades consultando las estadísticas de rendimiento. DynamoDB no tiene un esquema fijo, sino que cada elemento de datos puede tener un número diferente de atributos. Se pueden usar varios tipos de datos como strings, números o conjuntos. Amazon DynamoDB admite operaciones GET/PUT mediante la utilización de una clave principal definida por el usuario. La clave principal es el único atributo obligatorio para los elementos de la tabla e identifica tales elementos de forma exclusiva. La clave principal se especifica cuando se crea una tabla.

Una clave principal puede ser una clave hash como atributo exclusivo o una clave de rango hash compuesto. Una clave principal hash como atributo exclusivo podría ser, por ejemplo, "UserID". Esto podría permitir leer y escribir datos con mejoras generalmente en el tiempo de respuesta para un elemento asociado con un ID de usuario determinado. [43]

Ejemplos

En el ejemplo se utiliza un contenedor para los usuarios y en otro contenedor se pueden tener los datos de los usuarios:

[users.cab]
pepe=patata
juan=minino
mario=ferrari

[user_data.cab]
pepe_nombre=JoseAlberto
pepe_email=jose@alberto.com
pepe_fecha=19700315
juan_nombre=JuanAntoni
juan_email=juanAntoni@yopmail.com
juan_fecha=19800218
mario_nombre=MarioGarcia
mario_email=mgarcia@micorreo.es

Bases de Datos Orientadas a Documentos

Definición

Una base de datos de datos orientada a documentos es un programa diseñado para almacenar, recuperar y gestionar información semiestructurada orientada a documentos [44] Se denomina información semiestructurada a la información similar de una estructura implícita, pero no tan regular como para poder ser gestionada y automatizada como la información estructurada. [56]
Un documento encapsula información en un formato estándar (por ejemplo: XML, YAML, JSON y BSON). Los documentos en una base de datos orientada a documentos son similares a los registros de las BBDD relacionales, pero no requieren un esquema estándar con las mismas secciones, huecos, partes o claves. Por lo general, estos documentos son direccionables por una clave que los representa unívocamente. Además de la búsqueda por clave de documentos, estas BBDD suelen ofrecer una API o lenguaje de consultas que permite recuperar documentos en base a su contenido. [43]

Origen

Esta categoría de base de datos NoSQL fue inspirada en Lotus Notes [55], donde el concepto central es la noción de documento, siendo este la unidad atómica de almacenamiento.

Características

Las bases de datos NoSQL orientadas a documentos consisten en una estructura que no manipula esquemas al igual que los demás tipos de bases de datos NoSQL (y al contrario que las bases de datos relacionales). No tienen restricciones a la hora de elaborar la estructura de los documentos. Proveen flexibilidad para agregar información adicional en la base de datos sin tener que rediseñar la misma o modificar los datos que han sido almacenados con anterioridad [46]. Por todo esto último, se definen a las bases de datos NoSQL orientadas a documentos como *"libres de esquemas"* [47]. Además, estos sistemas de base de datos permiten el versionado de documentos y el control de dichas versiones. Proveen herramientas para reducir las consultas, funciones, vistas e índices secundarios y las limitaciones en las consultas son menos restrictivas que en el resto de los sistemas NoSQL.
Existe la posibilidad que los datos requeridos para ciertas consultas se encuentren en un único documento sin tener la necesidad de hacer joins o transacciones entre objetos. [45]

Estructura

Los documentos tienen una estructura para almacenar información libre (por ser semiestructurada), por lo cual puede darse el caso de un conjunto de documentos en el que la estructura de información entre los mismos sea muy diferente para modelar alguna situación particular. A su vez, los sistemas de bases de datos NoSQL orientados a documentos manipulan los documentos no como objetos separados clave/valor, sino como unidad, es decir, mantiene juntos los datos relacionados en los documentos a diferencia de una base de datos relacional donde los datos relacionados se separan en varias tablas. [57] Este tipo de sistemas permite indexar documentos a partir de su identificador, y en función del lenguaje de consultas utilizado es posible recuperar documentos por sus propiedades.

Para describir los documentos se utiliza una estructura de árbol jerárquico implementada a través de mapas, colecciones y valores escalares. Todos los documentos almacenados son iguales pero no necesariamente deben tener el mismo contenido.
Dentro de las colecciones se agrupan documentos que pueden ser almacenados de manera aleatoria, aunque se suelen almacenar documentos similares para facilitar el indexado y de esta manera, producir una mejora en el rendimiento y reducir los efectos de la concurrencia. [45]

Utilidades

Los documentos que manejan estos sistemas de almacenamiento son un conjunto de datos identificados por etiquetas, internamente pueden ser documentos JSON o de otro tipo que se recuperan mediante claves primarias. También suelen disponer de funcionalidades para combinar documentos al estilo de las JOINs de SQL. Los almacenes de documentos pueden ser muy convenientes en casos en los que por ejemplo, hay que almacenar formularios web rellenados por el usuario donde estos formularios cambian periódicamente. En lugar de tener que añadir una columna a una tabla por cada campo nuevo en un formulario diferente, o crear una tabla distinta para cada tipo de formulario, los almacenes de documentos permiten simplemente convertir el formulario a JSON o XML y guardarlo en este formato indexándolo por los campos que se desee. Los dos productos Open Source más populares son CouchDB y Mongo DB. [53]

Implementaciones

Las implementaciones de bases de datos NoSQL orientadas a documentos más conocidas son: *MongoDB, CouchDB y Terrastore***.** Debido a que MongoDB se detallará en el capítulo 4, se detallará a continuación las bases de datos *CouchDB* y *Terrastore*:
CouchDB [49] es una base de datos de código abierto orientada a documentos.
Entre sus características destacan que:

- Almacena los datos con JSON.
- Permite consultar los datos con JavaScript.
- Permite la replicación de datos maestro-maestro. [51]

Terrastore [50] es una base de datos de código abierto orientada a documentos. Sus características fundamentales son:

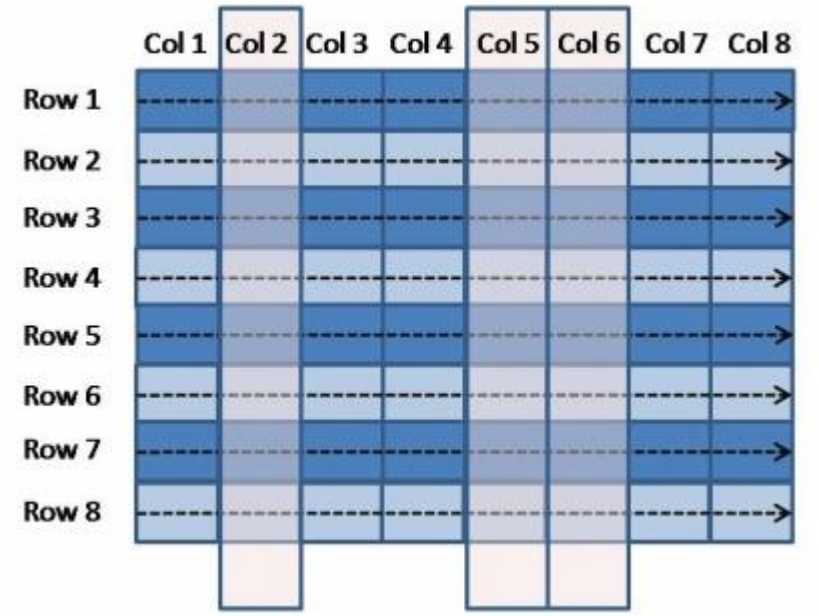
- Accesible mediante HTTP.

- Distribuida: permite distribuir los nodos por cualquier servidor de la red.
- Elasticidad: permite quitar y poner nodos sin necesidad de parar el servicio o cambiar la configuración.
- Escalabilidad en la capa de datos: los documentos se dividen y reparten por los nodos con balanceo automático y unión de las partes.
- Escalabilidad computacional: las consultas (select) y modificaciones (update) se distribuyen por los nodos reduciendo el tráfico.
- Consistencia: ofrece consistencia para los documentos, garantizando que siempre se obtiene el último valor de un documento, con aislamiento de la lectura de posibles “commits” de modificación.
- Las consultas y procesamiento de funciones Map/Reduce y, la actualización de funciones del lado del servidor. [51]

Bases de Datos Orientadas a Columnas

Definición

Las bases de datos orientadas a columnas tienden a ser un híbrido de las clásicas bases de datos relacionales y la tecnología orientada a columna. La columna es la base, es un elemento compuesto de un nombre, un valor y una marca de tiempo. [62]
Este tipo de modelo considera un registro o “fila” para tener acceso a los atributos necesarios de una columna, como se muestra en el dibujo:



Origen

Las Bases de Datos Orientadas a Columnas se introdujeron por primera vez en 1970 en productos como Model 204 y ABABASS [59]. En el año 2004 Google lanza un sistema de almacenamiento distribuido llamado BigTable [60], el cual es en la actualidad el precursor de este tipo de base de datos. Este sistema de base de datos NoSQL ha demostrado la capacidad de manejar un volumen datos del orden de petabytes a procesarse entre servidores distribuidos. [61]

Características

Las bases de datos orientadas a columnas almacenan toda la información en columnas, de esta forma se pretende que las lecturas mejoren tiempos de respuesta a diferencia de otros tipos de bases de datos, pero es factible que para las escrituras las mejoras no sean siempre las mismas. [63]

La base de datos orientada a columna almacena sus datos (físicamente por familia de columnas) de manera que se pueda mejorar el tiempo de agregación de una nueva columna, con menos actividad de entrada y salida. Una desventaja de almacenar datos en columnas es que los datos de una entidad se fragmentan entre varias columnas; por lo tanto, la inserción y la actualización o lectura del contenido completo de una entidad puede ser más lenta y compleja que en una base de datos relacional. [62]

Este modelo de almacenamiento orientado a columna permite clasificar y ordenar información. Además, es utilizado para soluciones ad-hoc y aplicaciones analíticas. La estructura de estas bases de datos favorece eficientemente en el sentido que permiten realizar búsquedas por clave de fila. [65]

Estructura

Una base de datos orientada a columnas almacena su contenido por columnas, en lugar de por filas. Este modelo de datos orientado a columnas es el que expone Big Table de Google [69]. Permite almacenar la información de manera tal que evita espacio innecesario en los valores nulos para una columna.
En una base de datos NoSQL orientada a columna, cada unidad de dato es pensada como un juego de pares clave/valor donde esta unidad es identificada con la ayuda de un identificador primario, frecuentemente referenciada como una clave primaria. BigTable suele llamar a esta como “row-key” (clave de fila). Las unidades de datos son clasificadas y ordenadas en la base de “row-key” (clave de fila).
La familia de columna actúa como una clave para columnas y el conjunto de clave de fila (row-key) como la clave del conjunto entero de datos. De esta manera, los datos que pertenecen a una clave de fila son almacenados en conjunto.
Los datos en BigTable son almacenados de manera secuencial. Como los datos crecen para almacenarse en un nodo, este se distribuye en múltiples nodos. La información es clasificada y ordenada no solo en cada nodo, sino a través de nodos que proporcionan un juego de datos ordenado. Los datos son persistidos de una manera tolerante a fallos donde tres copias de cada juego son mantenidas. [65]

Utilidades

Las bases de datos NoSQL orientadas a columnas se recomiendan para aquellos almacenes de datos cuya lectura es más frecuente que su escritura, y es necesario realizar operaciones (que crecen de manera exponencial) con los atributos de las entidades, como en el manejo de datos estadísticos [62]. Por ejemplo, en una compañía los datos crecen permanentemente porque las operaciones de negocios se expanden y por otro lado, las compañías tienen que interactuar con más recursos de datos y almacenar información de forma online. En la actualidad, existen usuarios (como analistas estadísticos del INDEC o censistas nacionales) que necesitan un mejor tiempo de respuesta en el acceso al historial de datos, para obtener resultados para fines analíticos que servirán para la toma de decisiones de una compañía. [64]

Implementaciones

Algunas de las implementaciones más conocidas de los sistemas de almacenamiento orientados a columnas son los siguientes:

- **Cassandra:** es una base de datos de código abierto cuya principal característica es que fusiona Dynamo de Amazon con BigTable de Google, siendo ambas implementaciones de código cerrado. Es decir, que sigue un modelo híbrido entre clave/valor y orientado a columna.
  Posee una tabla de datos por cada instancia Cassandra, cada familia de columnas puede contener o bien columnas o bien supercolumnas. Las supercolumnas son columnas con agrupación de n-columnas. Cada columna contiene elementos de la forma “Clave-Valor-Tiempo”, donde el valor del campo Tiempo es definible por el usuario. [66]

- **HBase:** Es una base de datos NoSQL de código abierto, NoSQL y distribuida que ha sido desarrollada como un subproyecto de Hadoop [67] y que usa HDFS [36] como su sistema de almacenamiento de archivos. Es una base de datos orientada a columnas, distribuida y escalable. Existe un API Java que permite el acceso de clientes a los datos. También se puede acceder mediante servicios RESTful que soportan formatos XML, protobuffer o codificaciones binarias. [68]

Ejemplos

Consideremos una tabla simple con información referida a personas.

Nombre: John
Apellido: Lennon
Codigo Postal: 1900
Sexo: Masculino

Otro conjunto de datos en la misma tabla podría ser:

Nombre: Paul
Codigo Postal: 94303

Con estos conjuntos de datos se asigna el valor 1 a la clave de fila del primer conjunto de datos y como 2 al segundo conjunto de datos. De esta manera, se clasifican los datos y se ordenan si el registro con clave de fila 1 está almacenado antes que el registro con clave 2.

Una posible familia de columnas del ejemplo presentado puede ser “nombre” con columnas “primer_nombre” y “apellido” como miembros. Otra familia de columnas puede ser “ubicación” con “codigo_postal” como miembro. Una tercera familia puede ser “perfil”.

Las familias de columnas son definidas típicamente cuando se está iniciando un modelado de información. Las columnas que se definen no necesitan a priori ninguna definición o declaración.

Las columnas pueden almacenar cualquier tipo de datos y pueden ser persistidos como serie de objetos.

El almacenamiento de este ejemplo consiste de tres partes: nombre, ubicación y perfil. Solo son almacenados los pares clave/valor con valores válidos.

For row-key: 1
Primer Nombre: John
Apellido: Lennon

For row-key: 2
Primer Nombre: Paul

La familia de columna: “ubicación” es la siguiente:

For row-key: 1
Codigo Postal: 10001

For row-key: 2
Codigo Postal: 94303

La familia de columna “perfil” tiene valores sobre el conjunto de datos con clave de fila 1, esta almacena solo lo siguiente:

For row-key: 1
Sexo: Masculino

Bases de Datos Orientadas a Grafos

Definición

Una base de datos orientada a grafos es un sistema de almacenamiento que permite la adyacencia libre de índice, es decir, que cada elemento contiene un puntero directo a sus elementos adyacentes por lo cual no es necesario realizar consultas por índices. [96]
Además, se la define como una base de datos que utiliza la topología de un grafo con nodos como vértices y relaciones como aristas y propiedades, utilizada para almacenar y representar datos. [97]
La siguiente figura (Figura 1) representa un ejemplo de una base de datos de grafo que almacena la información de una pequeña red social:

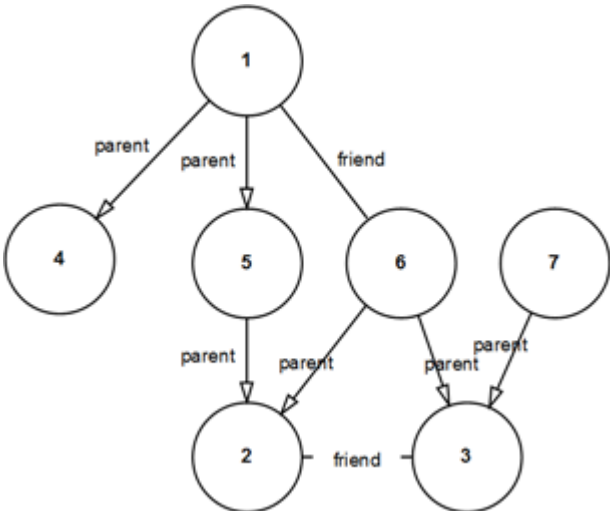


Figura 1. Ejemplo de Base de Datos de Grafo.

En el ejemplo, los nodos están enumerados del 1 al 7, se representa un identificador único para los nodos, mientras que los arcos están etiquetados dependiendo si representan la relación *parent* o *friend*. La semántica del grafo está dada por la etiqueta de los arcos, ya que el arco (1, *parent*, 5) se interpreta como “1 es parte de 5”. Este paradigma permite modelar ciertas situaciones (topología de red, estructura familiar) que no son descritas de forma natural con bases de datos relacionales, principalmente por la naturaleza de los datos.

Origen

Las bases de datos orientadas a grafos surgieron en la década de los 80's [70] con el objetivo de modelar información cuya estructura es un grafo [71]. Después de algunos años, la cantidad de trabajos publicados referentes a las bases de datos de grafos fue disminuyendo considerablemente hasta que a mediados de la década de los 90's cesó por completo. La razón del declive del área fue la aparición de los datos semiestructurados (definido en *"Bases de datos Orientadas a Documentos"*); la aparición de XML [72] capturó toda la atención en las estructuras de datos de árbol, por el hecho que este modelo es suficiente para una determinada cantidad de aplicaciones. Este tipo de base de datos está diseñado para almacenar millones de datos relacionados entre sí, que es natural representarlos en un grafo, tales como redes sociales y viales.

Características

Las bases de datos orientadas a grafos están diseñadas para los datos cuyas relaciones son representadas en forma de grafo, es decir, los datos son elementos interconectados con un número no determinado de relaciones entre ellos. La información a gestionar por este tipo de almacenamiento pueden ser las relaciones sociales, el transporte público, mapas de carreteras o topologías de red, entre otros ejemplos [96]. El almacenamiento está optimizado para hacer recorridos a través del grafo sin utilizar un índice para las relaciones entre nodos. Por lo cual, está optimizado para hacer consultas sobre datos próximos partiendo de uno o más nodos, más que para consultas globales. [97]

Estructura

Las bases de datos orientadas a grafos representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda recorrer la base de datos ya que esta puede describir atributos de los nodos (entidades) y las aristas (relaciones).
No existe un consenso general sobre la terminología existente en el área de grafos pues hay muchos tipos diferentes de modelos de grafos. Sin embargo, se están realizando algunos esfuerzos para crear el Modelo de Grafo de Propiedad que unifica la mayoría de las diferentes implementaciones de grafos. De acuerdo con este Modelo, la información en un grafo de propiedad se modela utilizando tres elementos básicos: el

nodo (vértice), la *relación* (arista) con dirección y tipo (etiquetado y dirigido) y la *propiedad* (atributo) en los nodos y las relaciones. [96]

Utilidades

El uso de las bases de datos orientadas a grafos se corresponde al volumen de datos (alcanzando los miles de millones de datos). [93] Por ejemplo, las empresas indicadas en el *"Capítulo II - Estado del Arte"*.
A continuación se detalla un listado de aplicaciones reales más comunes que utilizan a las bases de datos de grafos en la actualidad [73]:

Redes Sociales

Las redes sociales actuales, tales como Facebook y Twitter, están compuestas por grandes cantidades de datos modelados por un gran grafo [74]. Estos grafos representan el conjunto de actores como el conjunto de nodos, mientras que los arcos representan las relaciones o flujos entre los actores. Las operaciones más comunes en grandes grafos sociales son la búsqueda de caminos más cortos y las de agrupamiento (*clustering*) donde los algoritmos para resolver estas operaciones proveen un análisis de la relación entre dos o más individuos dentro de la red social [75].

Química

En química los datos son modelados como grafos [76] en donde los átomos se representan como nodos y los enlaces entre ellos se representan como arcos. En biología los datos también son representados por grafos [77] en donde los aminoácidos se representan como nodos y los enlaces también se representan como arcos. Estos grafos son utilizados para el análisis y el descubrimiento de fármacos.

Implementaciones

En los últimos años ha habido un incremento en la cantidad de implementaciones de bases de datos de grafos, existiendo en la actualidad variedades de proyectos [76], cada uno con distintas características, ventajas y desventajas. Algunos de los componentes que deben proveer una implementación de base de datos de grafos [86] son los siguientes: interfaces externas, lenguajes de bases de datos, optimización de consultas, motor de almacenamiento y manejo de transacciones.

Algunas de las implementaciones que satisfacen en su mayoría las condiciones anteriormente mencionadas son las siguientes:

- **AllegroGraph:**

AllegroGraph es uno de los precursores de la actual generación de bases de datos de grafo. Aunque fue creada inicialmente como una base de datos de grafo, actualmente su desarrollo está orientado a los estándares de la Web Semántica. Además, AllegroGraph provee características para análisis de redes sociales y para datos geográficos. [87]

DEX

DEX [88] es según sus creadores una biblioteca para gestionar grandes grafos o redes. En concreto, soporta multigrafos dirigidos y etiquetados con atributos. Desarrollada como una librería para Java, posee una serie de métodos para gestionar y consultar grafos, aunque no proporciona un lenguaje de consulta, por lo que su uso es a bajo nivel.

- **Neo4j**

La base de datos *Neo4j* [92] es una base de datos en grafo NoSQL. Ofrece una API orientada a objetos, un almacén nativo para manejar grafos en disco y un framework para recorrer grafos.

CAPITULO IV - MongoDB

Introducción

MongoDB (proviene de <<*humongous*>>, que significa "extremadamente") es un sistema de base de datos orientada a documentos, es decir que utiliza el tipo de base de datos que se describió en el "*Capítulo III - Tipos de Base de Datos NoSQL"*. Este sistema será empleado en el "*Capítulo V - Escenario*", principalmente por ser opensource y por varias razones que serán detalladas en este capítulo, como por ejemplo simplicidad en su uso.

Origen de MongoDB

El desarrollo de MongoDB fue trabajado con la empresa de software 10gen [98] en 2007 cuando se estaba desarrollando una plataforma como servicio (PaaS) similar a Google App Engine (mencionado en el *Capítulo II -Estado del Arte*) [112]. En 2009 MongoDB fue lanzado como un producto independiente y publicado bajo la licencia de código abierto AGPL. [100]

Estructura y Modelado

La estructura que utiliza MongoDB para almacenar información es correspondiente a la del tipo de base de datos orientada a documentos, como se ha mencionado en la introducción del presente capítulo. Esto quiere decir que cada registro o unidad básica se la denomina documento y cada conjunto de documentos se los denomina colección [100].
MongoDB provee el modelo JSON (*JavaScript Object Notation*) como formato para la representación de información en documentos. JSON es utilizado en diversas herramientas como CouchDB y por diversas compañías como Teradata, las cuales han sido mencionadas en el "*Capítulo II – Estado del Arte"* y en el *"Capítulo III – Tipos de Base de Datos"* respectivamente.
MongoDB se basa sobre JSON, pero actualmente la información no se almacena específicamente en dicho formato, sino en BSON (*Binary JSON*) debido a que BSON es una representación serializada de los documentos JSON lo que permite utilizar documentos embebidos, arrays dentro de documentos y además realiza una extensión de tipos de datos que no son parte de la especificación JSON como los "Date type" y "BinData type". [101]

Como se ha mencionado en el "*Capítulo III - Tipos de Base de Datos NoSQL"*, los documentos en MongoDB son un conjunto de par clave/valor, libre de esquemas. Por ejemplo, un documento posible en Mongo podría ser el siguiente:

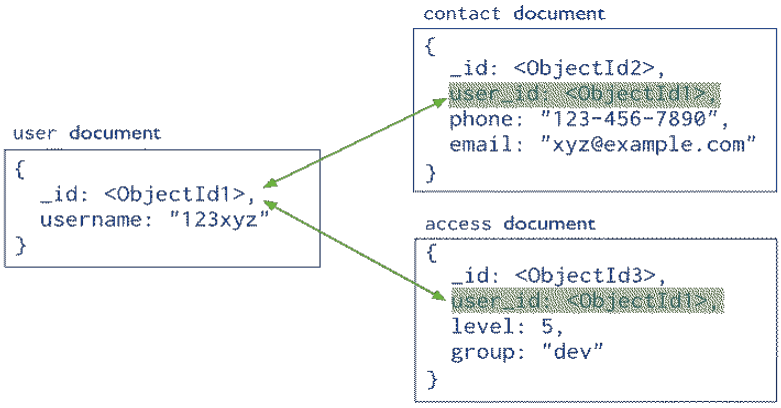
```
{
"firstname": "Peter",
"lastname": "Membrey",
"phone_numbers": [
"+852 1234 5678",
"+44 1234 565 555"
]
}
```

Como los documentos son libres de esquemas, MongoDB no requiere que en una lista de documentos aparezcan siempre la misma cantidad de campos para un documento. Esto quiere decir que en el ejemplo si no se conocería ningún teléfono de una persona, directamente la lista "phone_numbers" no aparecería [104].

Estructura del modelo

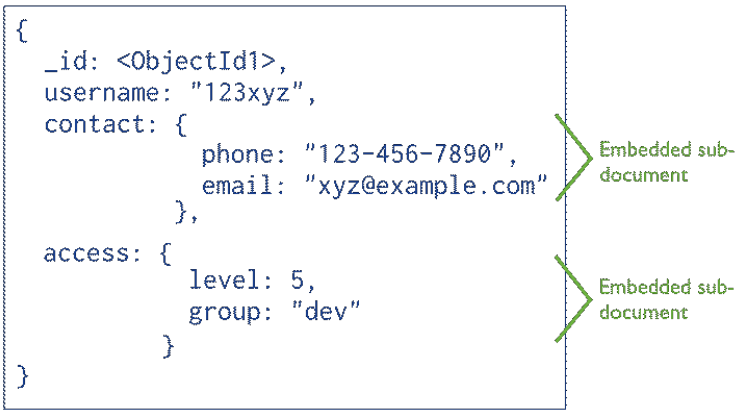
En MongoDB existen dos modelos de representación para las relaciones entre documentos: referencias y documentos embebidos. La relación por referencia consiste en que en los documentos existe un campo que identifica a otro documento que contiene datos específicos, también se lo denominan modelos de datos normalizados por el hecho de que con la referencia se obtendrá toda la información del documento.
La relación por documento embebido significa que podemos incluir uno o más documentos dentro de un mismo documento, también se los denominan modelos de datos desnormalizados por el hecho de que se pueden almacenar más de un documento dentro de un mismo documento sin realizar ninguna relación hacia otros documentos [105]. A continuación se presentan ambos ejemplos:

- Documentos por referencia:



En este ejemplo, los documentos “contact” el campo “user_id” incluye la referencia al documento “user”.

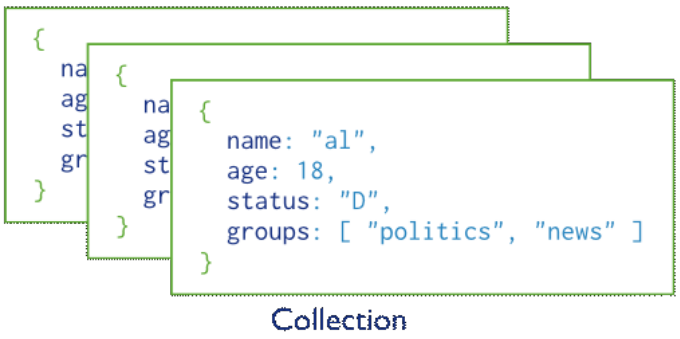
- Documentos embebidos:



En este ejemplo, los documentos “contact” y “access” son documentos embebidos de un documento particular.

Colecciones

Una colección en MongoDB es un contenedor de documentos, análogo a lo que es una tabla en un RDBMS [106]



Relaciones

Las relaciones en MongoDB se entienden como la información que se puede encontrar asociada en un mismo documento (información embebida) o referenciada, cuestión que se mencionó en la “Estructura de modelo” del presente capítulo [107].
En MongoDB existen relaciones 1 a 1 y relaciones 1 a N. Si bien se pueden modelar relaciones N a N, no se abarcará este tipo de relaciones ya que no se incluye en la documentación oficial y tampoco será necesario en el caso de estudio para el “*Capítulo V –Escenario”.*

Relaciones 1 a 1

Estas relaciones se utilizan cuando tenemos relacionada un conjunto de información que corresponde a una entidad de datos. Dicha entidad de datos puede ser relacionada o embebida [106].Es decir que podemos representar de dos maneras relaciones 1 a 1, presentadas en los siguientes ejemplos:

```
{  _id: "joe",
  name: "Joe Bookreader"
}
```

```
{  patron_id: "joe",
  street: "123 Fake Street",
  city: "Faketon",
  state: "MA",
  zip: "12345"
}
```

En este ejemplo (información relacionada) se muestra que para acceder al nombre que se encuentra en el primer documento, se debe acceder a “patron_id” y luego acceder al primer documento para acceder al nombre. La desventaja que tiene que se debe realizar una consulta adicional para obtener una información que es propia de una misma entidad.

```
{
 _id: "joe",
 name: "Joe Bookreader",
 address: {
            street: "123 Fake Street",
            city: "Faketon",
            state: "MA",
            zip: "12345"
          }
}
```

En este ejemplo (información embebida) la información que corresponde a una misma entidad se encuentra en un mismo documento. Se puede obtener toda la información de una misma entidad en una misma consulta.

Relaciones 1 a N

Estas relaciones se utilizan cuando se relaciona un documento que posee referencias a más de un documento [107]. Se muestran los siguientes ejemplos que muestran la manera de representar estos tipos de relaciones.

```
{
        _id: "joe",
        name: "Joe Bookreader"
      }

      {
        patron_id: "joe",
        street: "123 Fake Street",
        city: "Faketon",
        state: "MA",
        zip: "12345"
      }

      {
        patron_id: "joe",
        street: "1 Some Other Street",
        city: "Boston",
        state: "MA",
        zip: "12345"
      }
```

Este ejemplo representa en un modelo normalizado que los documentos “address” contienen referencias al documento “patron”. En este caso en particular, necesitamos al menos dos consultas para obtener todos los datos completos relacionados.
En el caso de documentos embebidos (desnormalizado), con una consulta podemos obtener toda la información necesaria. El ejemplo quedaría de la siguiente manera:

```
{_id: "joe",
  name: "Joe Bookreader",
  addresses: [
            {
              street: "123 Fake Street",
              city: "Faketon",
              state: "MA",
              zip: "12345"
            },
            {
              street: "1 Some Other Street",
              city: "Boston",
              state: "MA",
              zip: "12345"
            }
          ]
}
```

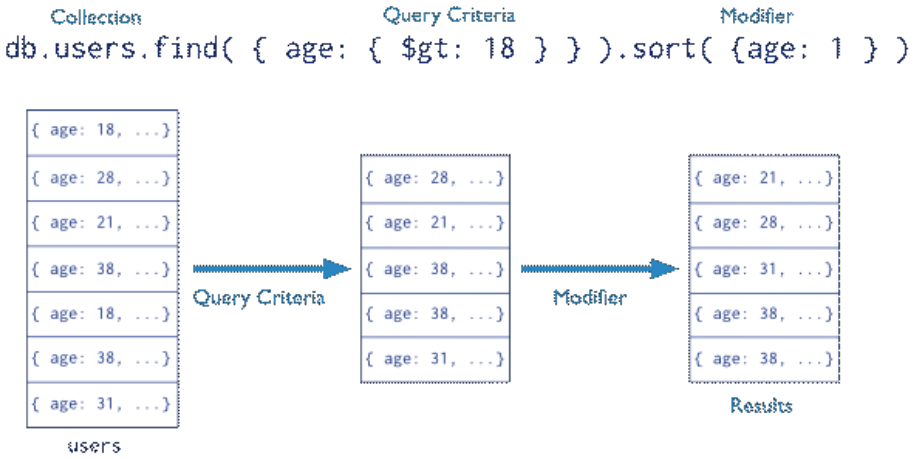
Operaciones sobre la base de datos

MongoDB provee la semántica necesaria para realizar operaciones CRUD (*create, read, update and delete*) [108].

Consultas

En cuanto a operaciones de consulta (read), en MongoDB el resultado de una query es una colección de documentos. Las querys especifican condiciones que identifican los documentos en los que MongoDB retorna hacia los clientes. Una query produce un retorno de documentos que incluye condiciones que "matchean" campos específicos. Se incluyen operadores condicionales, expresiones, filtros, límites sobre resultados de consultas, operaciones de agregación y ordenaciones [109].

En el siguiente diagrama el proceso de consulta especifica un criterio y un modificador:

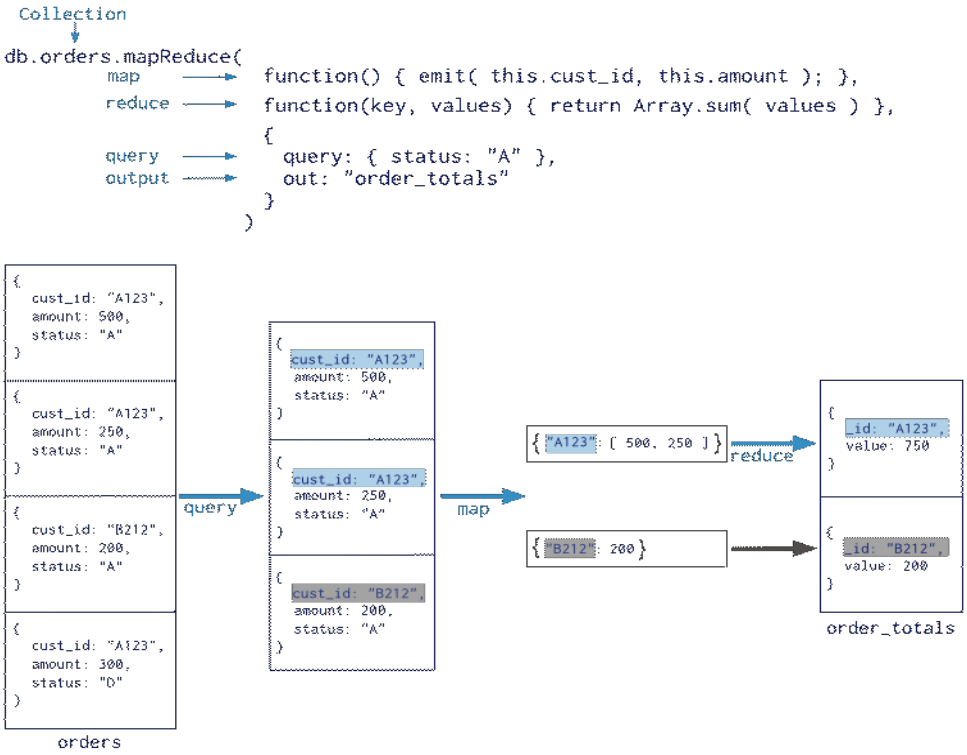


Map Reduce

El término "Map Reduce" ha sido mencionado en los Capítulos II y III como una característica analítica de operación de agregación y como herramienta en diversos productos respectivamente.

El término MapReduce es un paradigma de procesamiento sobre volúmenes de información que crecen de manera exponencial involucrando resultados con

operaciones de agregación. MongoDB provee el comando “mapReduce” para realizar estas operaciones [110].



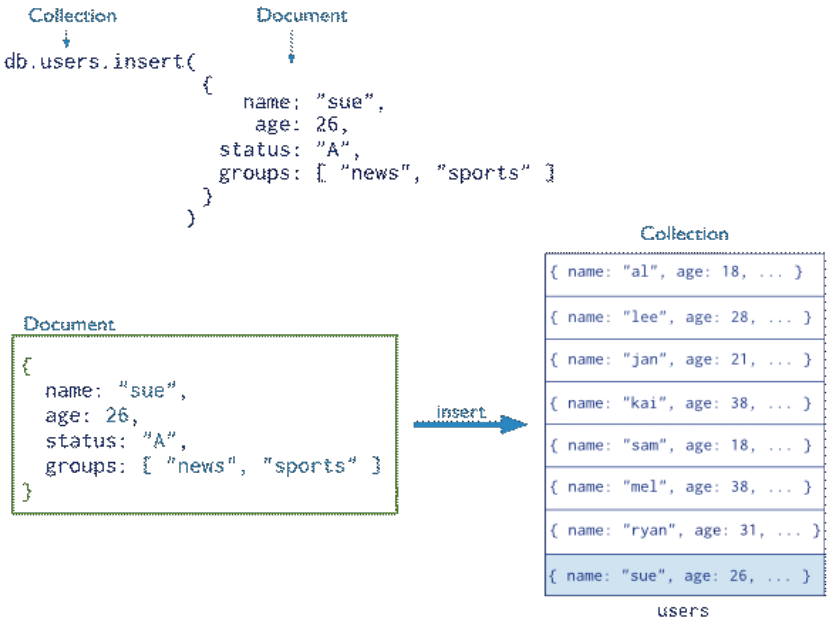
En esta operación Map Reduce, MongoDB aplica la fase “map” para cada documento de entrada (documentos en una colección que “matchean” una condición). La función “map” provee pares clave valor, de las cuales estas claves pueden contener múltiples valores. MongoDB aplica para este caso la fase “reduce” que recolecta la información y realiza la operación en cuestión (en este ejemplo una suma) y la almacena en una colección. El nombre de “reduce” es justamente porque reduce la operación en cuestión.

Las operaciones MapReduce en MongoDB son funciones JavaScript [111] ejecutadas sobre el proceso *mongodb* [112]. Pueden retornar los resultados en un documento o en colecciones.

Las operaciones MapReduce también se utilizan en entornos distribuidos para obtener mejores rendimientos en las aplicaciones [113], pero no se detallará en este tema ya que sistemas distribuidos queda fuera del alcance, como ya se ha mencionado en el “*Capítulo I – Introducción”.*

Modificación de datos:

En cuanto a modificación de datos (create, update o delete), estas operaciones modifican los datos de una colección. Para actualización y eliminación de datos se puede especificar el criterio para seleccionar los documentos en los que se desea realizar esas operaciones [114].
En el siguiente diagrama la operación de inserción agrega un nuevo documento a la colección “users”.



Una operación *update* modifica un documento existente de una colección.
Ejemplo:

```
db.users.update(                    <-- collection
   { age: { $gt: 18 } },            <-- update criteria
   { $set: { status: "A" } },       <-- update action
   { multi: true }                  <-- update option
)
```

Una operación *delete*, elimina un documento existente de una colección.
Ejemplo:

```
db.users.remove(        <-- collection
   { status: "D" }      <-- remove criteria
)
```

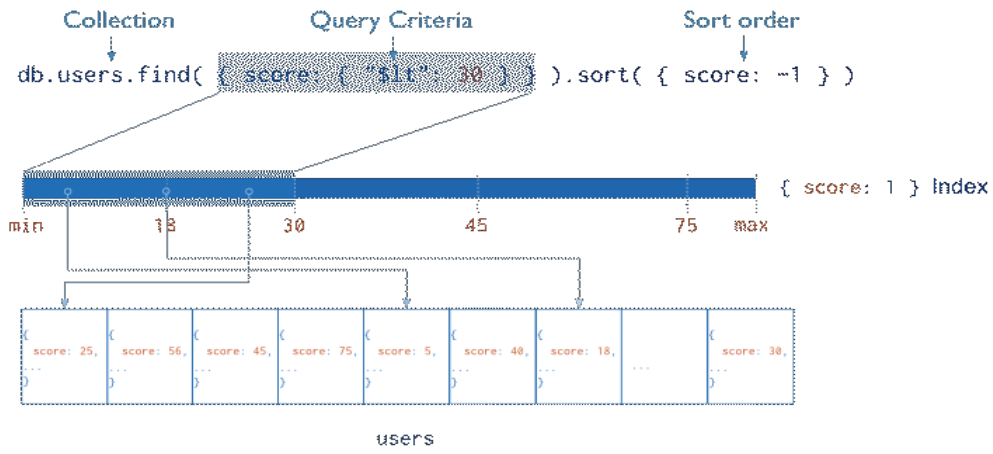
Operaciones Adicionales

Además de las operaciones ya mencionadas que soporta MongoDB, existen otras adicionales de las cuales las de mayor relevancia son: creación de índices [115] (para mejorar tiempos de respuesta en consultas), replicaciones [116] (para mejorar la disponibilidad en distintos servidores) y funciones específicas para manipular información compartida en diferentes clusters [117].

Índices

MongoDB soporta índices para mejorar la ejecución de consultas. Sin índices, MongodDB debe realizar la búsqueda en toda una colección para seleccionar los documentos que coincidan con una condición de búsqueda. En estos casos es apropiado usar índices para limitar el número de documentos en los que se debe realizar la búsqueda [115].
A grandes rasgos, los índices en MongoDB funcionan del mismo modo que en un RDBMS y son básicamente un tipo de estructura de datos que almacena una porción global de los datos de cada colección, almacenando el valor de uno o varios campos de forma ordenada según un criterio. De esta forma, al ejecutar una query contra MongoDB se examinará si existe un índice que concuerde los parámetros de búsqueda y el resultado esperado debería mejorar los tiempos de respuesta. MongoDB utiliza para su implementación de índices árboles B como estructura de datos [118].

El siguiente diagrama ilustra el uso de un índice con una consulta y una ordenación:



Se debe tener en cuenta que una mala administración de índices (como por ejemplo generar más cantidad de la necesaria) podría generar sobrecarga en operaciones de insert, update y remove, ya que no solo se debe realizar dichas operaciones sino que también debe realizar el registro de los índices en la colección en la que se involucra la información[119].

GridFS

GridFS [126] es una herramienta adicional que provee MongoDB para almacenar y obtener archivos que exceden los 16MB por documento almacenado (como por ejemplo archivos de audio, foto o video). GridFS divide un archivo en partes o también llamadas “chunks” [127] y almacena cada uno de ellas en un documento separado. GridFS también es utilizado cuando se quiere acceder a un archivo sin necesidad de cargar todo el mismo en memoria.

Usos de MongoDB

Las bases de datos MongoDB son utilizadas por diversas compañías como MTV Network [24], Craiglist [121] y Foursquare [122]. También desde la página oficial de MongoDB existen experiencias de empresas dedicadas a arquitecturas BigData en las que presentan sus experiencias beneficiosas en el uso de MongoDB [123]. El uso de MongoDB se enfoca hacia casos de administración de e-commerce (comercio electrónico), gaming, aplicaciones móviles, aplicaciones analíticas y logging (administración de históricos) [130].

Sistemas Operativos y Lenguajes de Programación Soportados

En cuanto a sistemas Operativos, MongoDB es soportado para sistemas UNIX, OS X [124] y Windows [125].
Los lenguajes de programación que soporta son: C, C++, Java, C#, .NET, Java, Node.js, Perl, PHP, Python, Ruby, Scala. Para manipular base de datos MongoDB con estos lenguajes existen “manejadores” (drivers) para cada uno de ellos que son utilizados como librerías adicionales.

Ventajas y Limitaciones

La principal ventaja de MongoDB es que se trata de un modelo NoSQL orientado a documentos que intenta mejorar los tiempos de respuesta con respecto al modelo relacional cuando se trabajan con datos no estructurados que crecen de manera exponencial [128]. En cuanto a soporte al desarrollador es opensource [129], soporta variedad de lenguajes de programación mencionados en el presente capítulo en "*Sistemas Operativos y Lenguajes de Programación Soportados";* así como también diversas herramientas adicionales propias para mejorar la administración de base de datos orientada a documentos.

MongoDB contiene algunas limitaciones, la más relevante es que no permite el manejo de transacciones y no asegura completamente ACID (como ya se ha mencionado en los capítulos anteriores que es lo que sucede en cualquier tipo de base de datos NoSQL) [130]. Tampoco permite almacenar documentos mayores a 16MB como se ha mencionado en el presente capítulo en *"GridFS".* [131]

CAPITULO V - Escenario

Introducción

En la actualidad, los avances en el desarrollo Web como así también las redes sociales con millones de usuarios, provocaron en general inconvenientes de alta escalabilidad y agregaron complejidad en la representación de los datos [132]. Esto trae aparejado que en muchos casos no se necesite un esquema prefijado de datos, debido a que los datos a almacenar crecen de manera exponencial y no se encuentran homogeneizados [133]. En este sentido, los cambios de estructuras relacionados a modificaciones en las bases de datos no deberían impactar directamente en la estructura de la información y el esfuerzo en el desarrollo de la programación. Como se hizo mención en el *Capítulo 3 - Clasificación de las Bases de datos NoSQL*, debido a esta cuestión se ofrecen posibilidades para analizar de qué manera es más conveniente almacenar estructuras de información de acuerdo a un requerimiento específico [134]. En bases de datos relacionales los cambios de estructuras estarían referidos, por ejemplo, a algunas cuestiones como: agregar campos en una tabla con su tipo de datos correspondiente, modificar el tamaño de algunos campos (texto o numérico), incorporar índices en las tablas, agregar vistas, procedimientos almacenados, entre otras. Por otro lado, estos cambios son transparentes en las bases de datos NoSQL. Estas cuestiones principalmente llevan a pensar diversos planteos sobre distintas representaciones de la información, en particular asociadas al concepto Web Mining mencionado en el *Capítulo I - Bases de Datos a Gran Escala.* Este concepto abarca técnicas para la extracción y recuperación de la información [139]. Si estas técnicas pudieran ser aplicadas a un modelo específico que estuviese representado tanto en una base de datos relacional como en una base de datos NoSQL, se podrían obtener diferentes resultados al momento de acceder a la información, como por ejemplo, en tiempos de respuesta, acceso a caché o procesamiento en disco. La presencia de Web Mining garantiza el registro de aquellas acciones que realizan los usuarios sobre la web, por lo tanto esto podría llegar a ser un campo de estudio interesante para representar en una base de datos NoSQL y relacional. Por tal motivo, se opta por Web Mining como el ámbito central de escenario del presente capítulo.

Web Mining

El concepto de Web Mining, presentado en el *Capítulo I - Bases de Datos a Gran Escala* de la presente tesina, es considerado una metodología de recuperación de la información que utiliza herramientas de Data Mining para extraer información del contenido de la web, la estructura de las relaciones entre las páginas web (enlaces) y el registro de navegación de los usuarios.

Algunos de los objetivos principales del Web Mining son los siguientes:

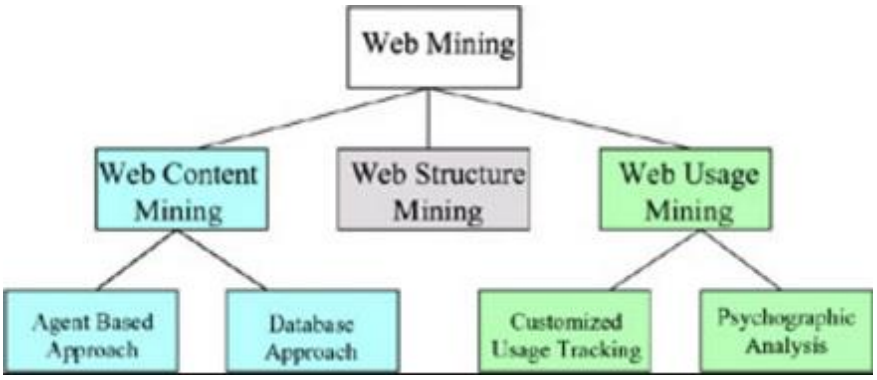
- Tener representaciones gráficas que muestren los cambios sufridos y/o representar la estructura general de la red.
- Analizar el comportamiento de los usuarios y de esta manera mejorar la estructura y contenido de los sitios web más visitados.
- Descubrir recursos, extraer información, analizar datos e inferir generalidades.
- Encontrar información relevante.
- Obtener nuevos conocimientos provenientes de la información disponible en la W3.
- Personalizar la información.
- Saber más sobre usuarios y clientes. [139]

Tipos de Web Mining

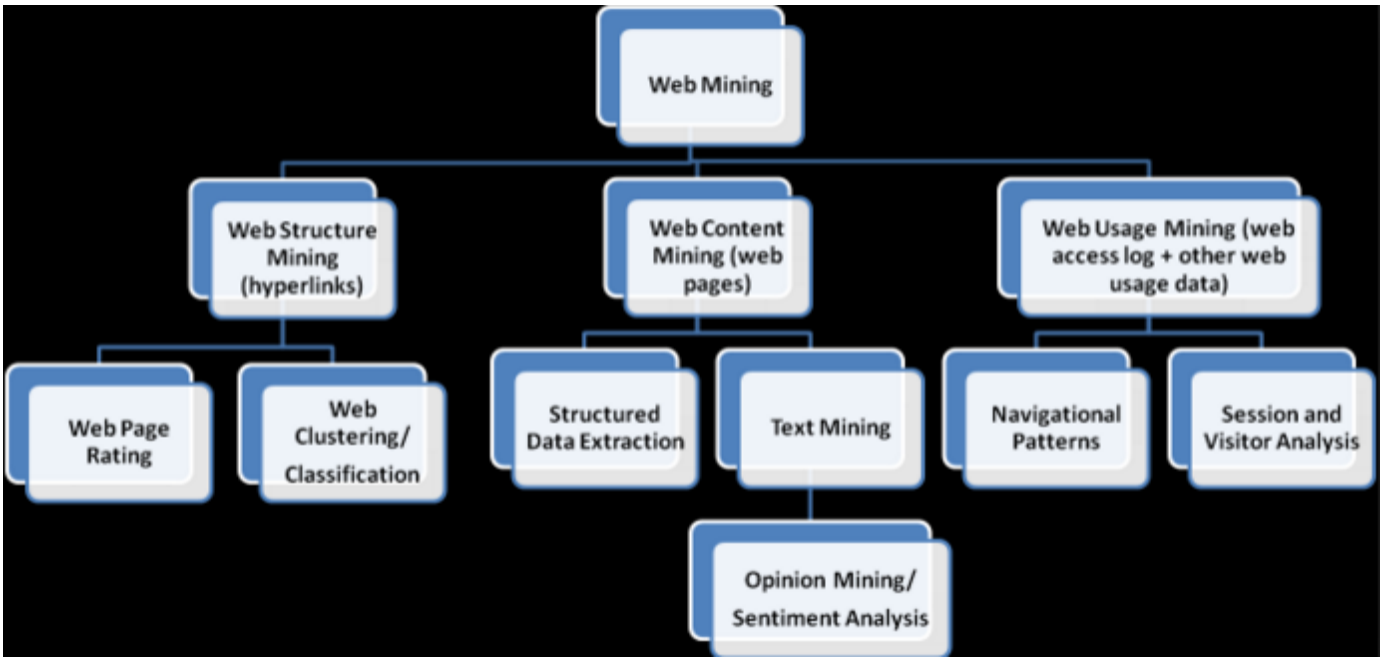
El Web Mining facilita la tarea de descubrir información, encontrar documentos relacionados y mostrar temáticas. Según la finalidad deseada, la actividad de buscar en la web se desglosa en tres dominios de extracción de conocimiento de acuerdo con la naturaleza de los datos [133].

- ➢ Web Content Mining (minería de contenido web)
  - Web Page Content Mining
  - Search Result Mining
- ➢ Web Structure Mining (minería de estructura web)
- ➢ Web Usage Mining (minería de uso web)
  - General Access Pattern Tracking
  - Customized Usage Tracking

Por lo tanto, la jerarquía quedaría definida de la siguiente manera:



En la siguiente figura se muestra una subclasificación más detallada:



Web Content Mining:

La minería de contenido tiene como principal objetivo otorgar datos reales o finales a los usuarios que interactúan con la web, de manera tal, que los usuarios puedan obtener cierta información del contenido de la web.
Generalmente la información disponible suele encontrarse de forma no estructurada (minería de texto), semiestructurada y estructurada como tablas HTML generadas automáticamente con la información de la base de datos.
La minería de contenido puede ser diferenciada desde dos puntos de vista: Recuperación de la información (IR) y desde la vista de base de datos (DB). [138]
Es decir, asistir en el proceso de recuperación de la información o mejorar la información encontrada por los usuarios, usualmente basada en las solicitudes hechas por ellos mismos (IR). Desde el punto de vista de DB principalmente trata de modelar los datos e integrarlos en la web a través de queries.

Web Content Mining: Extracción y Recuperación de Información

La extracción de información (IE) se centra principalmente en la estructura o la representación de un documento mientras que la recuperación de la información (IR) considera al texto en un documento como una “bolsa” de palabras desordenadas [140]

La recuperación de información es el proceso de encontrar el número apropiado de documentos relevantes de acuerdo a una búsqueda hecha en una colección de documentos. La IR y la web mining tienen diferentes objetivos, es decir la web mining no busca reemplazar este proceso. La web mining pretende ser utilizada para incrementar la precisión en la recuperación de la información y mejorar la organización de los resultados extraídos. [139]
La recuperación de la información es popular en diferentes empresas del mundo web las cuales hacen uso de este tipo de sistemas, las máquinas de búsquedas (por ej.: Google), directorios jerárquicos (por ej.: Yahoo) y otros tipos de agentes y de sistemas de filtrado colaborativos.

Se puede asegurar que dichas técnicas (IR e IE) son complementarias una de otra y usadas en combinación pueden generar valor agregado.
Los pasos para la extracción de información en documentos no estructurados son el reconocimiento de objetos de dominios tales como, nombres de personas y compañías, análisis sintáctico y etiquetado semántico.

Existe una tecnología llamada Text Mining que hace referencia principalmente al proceso de extracción de información y conocimiento interesante, no trivial desde documentos no estructurados.
Las principales categorías de la Web Text Mining son las siguientes: Text Categorization, Text Clustering, association analysis, trend prediction.

- *Text Categorization*: dada una determinada taxonomía, cada documento de una categoría es clasificada dentro de una clase adecuada o más de una. Es más conveniente y más sencillo realizar búsquedas especificando clases que buscando en documentos. Actualmente existen varios algoritmos de text categorization, dentro de los cuales encontramos, K-nearest, neighbor-algorithm y naive bayes algorithm.

- *Text Clustering:* el objetivo de esta categoría es el de dividir una colección de documentos en un conjunto de clústeres tal que la similitud intra-cluster es minimizada y la similitud extra-cluster es maximizada. Se puede hacer *Text Clustering* a los documentos que fueron extraídos por medio de una máquina de búsqueda. Las búsquedas de los usuarios referencian directamente a los clusters que son relevantes para su búsqueda. Existen dos tipos de Text Clustering: Clustering Jerárquico y Clustering Particional (G-HAC y k-means). [141]

Web Content Mining: Base de Datos

El objetivo que tiene la Web Content Mining desde el punto de vista de la base de datos es que busca representar los datos a través de grafos etiquetados.

La publicación de datos semiestructurados y estructurados en la web ha evolucionado principalmente con las “hidden web”, páginas ocultas las cuales son generadas automáticamente con datos de base de datos a través de consultas hechas por usuarios. La extracción de estos datos permite agregar valor agregado a los servicios, por ejemplo, en los comparativos de compras, meta búsquedas, etc. Existen varios enfoques para la extracción de información estructurada; manual wrapper, wrapper induction y el enfoque automático. [142] El primero consiste en escribir un programa para extracción de información de acuerdo con los patrones encontrados en un sitio Web en específico. Los segundos consisten en identificar un grupo de páginas de entrenamiento y un sistema de aprendizaje generará reglas a partir de ellas, finalmente dichas reglas serán aplicadas para obtener objetos identificados dentro de páginas Web. Finalmente el método automático tiene como objetivo principal identificar patrones de las páginas web y luego usarlas para extraer información. Este último método es el más utilizado para extraer información de la Web.

Web Structure Mining:

El *Web Structure Mining* intenta descubrir la organización de los enlaces del conjunto de hiperenlaces dentro del documento para generar un informe estructural sobre la página y el sitio web. Se obtiene información acerca de si los usuarios encuentran la información, si es suficiente el tamaño de la estructura de sitio, si los elementos están colocados en los lugares adecuados dentro de la página, si la navegación se entiende, cuales son las secciones con menos visitas y su relación con el lugar que ocupan en la página central. [144]

Según el objetivo a estudiar, se pueden dar tres tipos de informes:

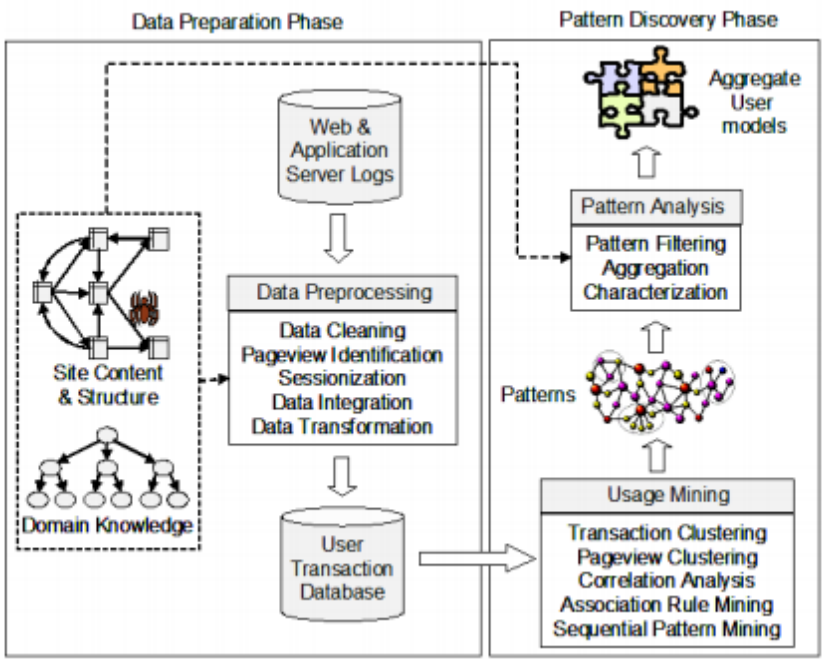
- Basándose en los hiperenlaces, clasifica las páginas web y genera el informe.
- Revelando la estructura web en sí.
- Descubriendo la naturaleza de la jerarquía o de la red de hiperenlaces del sitio Web de un dominio particular.

La Web Structure Mining se centra principalmente en la estructura de los hiperlinks de la web [141], basada en la entrada y salida de links de las páginas. Los links que apuntan a una página puede sugerir la popularidad de la misma, mientras que los links que salen de la página demuestran los tópicos o la riqueza de contenido.

Web Usage Mining:

El Web Mining de uso es la aplicación de las técnicas de data mining para descubrir pautas de conducta a la hora de utilizar la web por parte de los usuarios.

Esta extracción se refiere a patrones de navegación que se pueden descubrir en los usuarios y que puede servir para mejorar la misma. Por ejemplo, si el 80% de los usuarios recurren al campo de búsqueda cuando entran al sitio entonces debería ponerse énfasis en la mejora de esa interfaz y que el motor que se encuentre detrás devuelva la información deseada. Este proceso se basa en el uso de logs de los accesos al web. [144]



En definitiva, se trata de seguir una serie de pautas sobre:

- El acceso que utilizan los clientes cuando consultan el sitio web de una empresa.
- Los usuarios que interrogan a una aplicación que precede a una base de datos
- Los individuos que navegan por páginas determinadas.

A partir de datos secundarios derivados de interacciones automáticas de los usuarios mientras navegan por la web se pueden cubrir mejor las necesidades que se solicitan a través de aplicaciones basadas en protocolos W3.
Los logs que se generan constantemente en los servidores debido a los requerimientos de los usuarios, generan un alto volumen de datos provenientes de dichas acciones. Recientemente este gran volumen de información relevante empezó a usarse para obtener datos estadísticos, analizar accesos inválidos y analizar problemas que se produjeran en el servidor.
Los datos almacenados en los logs siguen un formato standard. Una entrada en el log siguiendo este formato contiene entre otras cosas, lo siguiente: dirección IP del cliente, información del usuario, fecha y hora de acceso, requerimiento, URL de la página accedida, el protocolo utilizado para la transmisión de los datos, un código de error, agente que realizó el requerimiento, y el número de bytes transmitidos. Esto es

almacenado en un archivo de texto separando cada campo por comas (“,”) y cada acceso es un renglón distinto.
Algunas de las técnicas de data mining que se aplican en los servidores web son:

- Clasificación
- Reglas de asociación (Association Rules)
- Patrones secuenciales (Sequential Patterns)
- Clustering

Patrones aplicables a servidores web

Los patrones determinan la forma en que los usuarios interactúan con la web y lo realizan a través de logs que se encuentran en los servidores web.
Los beneficios más interesantes son: [145]

- Mejorar la navegación de un sitio.
- Permitir realizar campañas de publicidad.
- Estructurar mejor el contenido de un sitio web.
- Dirigir avisos de ofertas a grupos de usuarios.

A continuación se detallan algunos de los patrones:

Clasificación:

Las técnicas de clasificación permiten desarrollar un perfil para los ítems pertenecientes a un grupo particular de acuerdo a los atributos comunes. Este perfil luego puede ser utilizado para clasificar nuevos ítems que se agreguen en la base de datos. En el contexto de Web Mining, las técnicas de clasificación permiten crear un perfil para clientes que acceden a páginas o archivos del servidor, basado en información disponible de los mismos.

Por ejemplo: Clientes de agencia del estado o gobierno que visitaron el sitio tienden a interesarse en la página */myCompany/productos/producto1.html*; 50% de los clientes que colocaron una orden online en */myCompany/productos/producto2.html*, pertenecen al grupo 25-30 años y viven en La Plata.

Reglas de asociación:

El objetivo de estas técnicas es descubrir las correlaciones entre referencias a varios archivos disponibles en el servidor hechas por un mismo cliente.

A través de esta regla se conocen las transacciones realizadas por cada usuario, las cuales están contenidas en el log. Teniendo en cuenta que “cada transacción está compuesta por un conjunto de URL”. Esto permite encontrar las similitudes entre los diferentes ítems, como por ejemplo:

El 60% de los clientes que acceden a la página con *URL /myCompany/productos/,* también acceden a la página /myCompany/productos/producto1.html.

Patrones secuenciales:

Estas técnicas permiten a las organizaciones basadas en la web predecir los patrones de visita de los usuarios y los ayuda a dirigir las publicidades a grupos de usuarios basándose en estos patrones.
En general en las bases de datos transaccionales se tienen disponibles los datos en un período de tiempo y se cuenta con la fecha en la que se realizó la transacción; la técnica de patrones secuenciales se basa en descubrir patrones en los cuales la presencia de un conjunto de ítems es seguido por otro ítem en orden temporal.

Por ejemplo:
El 60% de los clientes que emitieron una orden online en *myCompany/productos/producto1.html*, también emitieron una orden online en */myCompany/productos/producto4.html* dentro de los siguientes 15 días.

Estas técnicas también conocidas como “Episodios Frecuentes” [146] pueden utilizarse para el descubrimiento de tendencias, comportamiento de usuarios, secuencia de eventos, etc. Esta información puede ser utilizada tanto en el aspecto comercial (pensar una campaña de marketing) como en el aspecto técnico (mejorar los tiempos de acceso).

Clustering:

El clustering permite agrupar clientes y elementos de datos que tienen características similares. A través de esta técnica se analizan los logs de transacciones con el fin de desarrollar y ejecutar futuras estrategias de marketing, como por ejemplo el envío de correo electrónico a los clientes.

Presentación del escenario

El desarrollo que se ha realizado hasta el momento en todos los capítulos de la presente tesina de grado, permiten abordar al estudio particular que se estudiará en el presente capítulo para el escenario en cuestión sobre un modelo de log en el ámbito Web Content Mining.

Además de lo expuesto dentro de la *Propuesta de Tesis* acerca de *Web Content Mining* [145] y [146], se agrega el concepto de *Web Usage Mining* [144] por el cual se identifican patrones que determinan la forma en que los usuarios interactúan con la web, estos patrones se representan a través de logs que están ubicados en los servidores web. Los archivos de logs son una grabación de la actividad de un servidor o de un sitio web a lo largo de un período de tiempo determinado.

El presente escenario abocará al proceso de manipulación de este tipo de información que tiene un significado útil a partir de los registros que son generados por un servidor; en particular el historial de información que es utilizada por un usuario. Dicho historial generalmente no posee una estructura de datos prefijada y rígida. El uso de NoSQL en este aspecto es directamente recomendable [137].

Modelo del diseño Web Mining

En función de la importancia de recuperar y extraer la información del contenido de los documentos de la web (*Web Content Mining*) y su vez, intentar extraer información a partir de sesiones y comportamientos de los usuarios (*Web Usage Mining*), se presenta un modelo que puede contener el historial de las acciones que realizan los usuarios sobre un servidor web, y por el cual se podrán obtener resultados interesantes a partir de búsquedas implementadas con querys que surgen de patrones de Web Mining de uso aplicables en un servidor web.

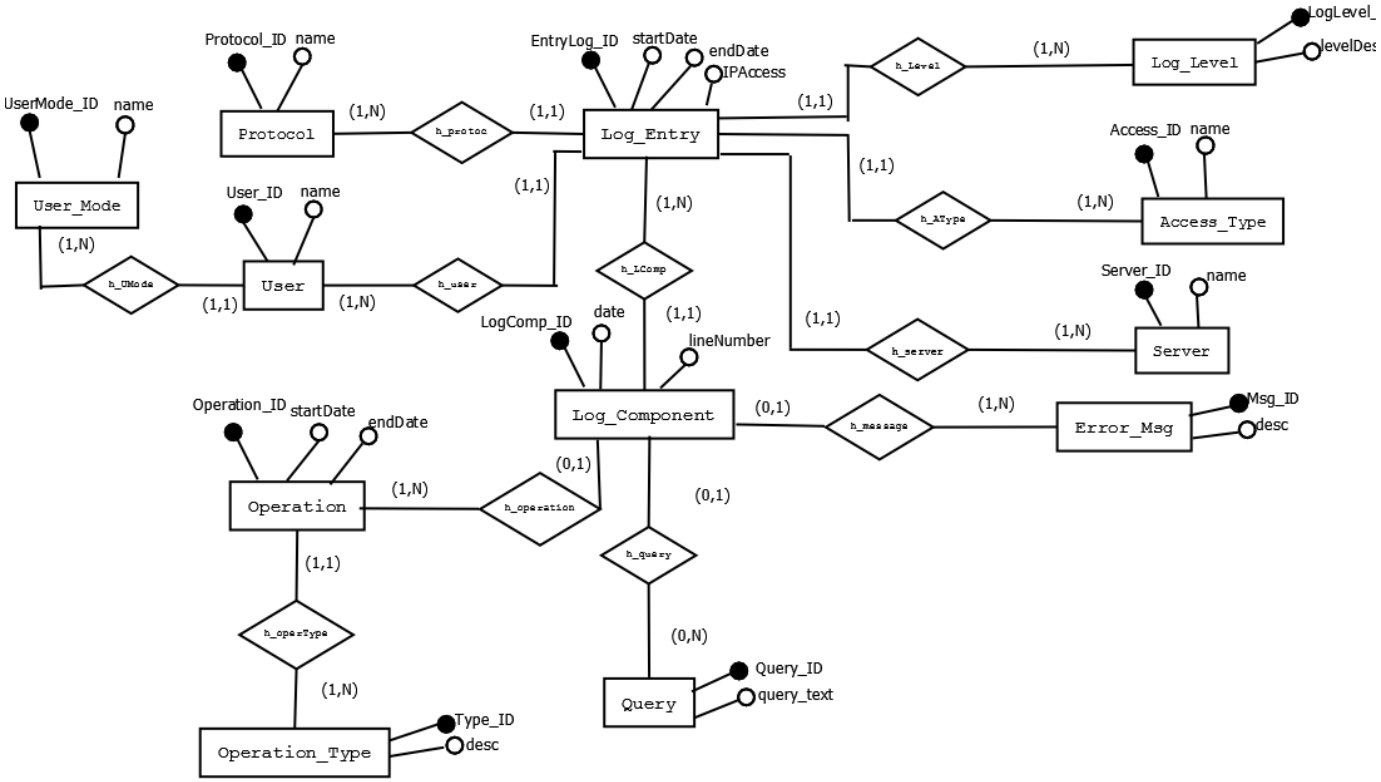
Luego, este diseño centrado en el caso de estudio *Web Content Mining* es abordado en un escenario real y específico de manipulación de logs.

Los modelos se realizaron para la representación relacional sobre el esquema entidad-relación, modelo lógico y su correspondiente modelo físico de datos (pasaje a tablas). Para el caso de NoSQL, se muestra el modelo de documentos JSON como ya se ha mencionado en el *"Capítulo IV (MongoDB)"*

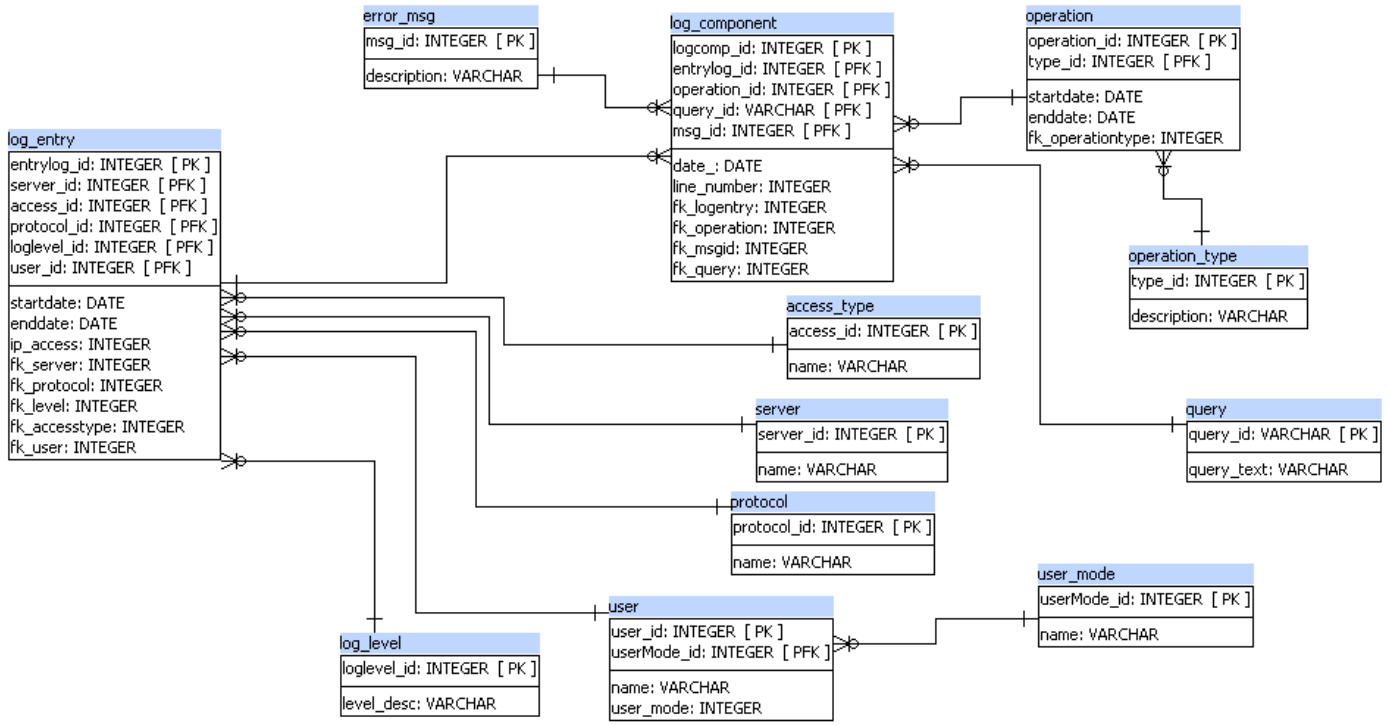
Relacional

A continuación, se presenta el modelo E-R, el modelo lógico (con las relaciones entre entidades) y su correspondiente modelo físico (pasaje a tablas):

Modelo E-R


Protocol_ID
name
Protocol
(1,N)
h_protoc
(1,1)
EntryLog_ID
startDate
endDate
IPAccess
Log_Entry
(1,1)
h_Level
(1,N)
Log_Level
LogLevel_
levelDes
UserMode_ID
name
User_Mode
(1,N)
h_UMode
User_ID
name
User
(1,1)
(1,N)
h_user
(1,1)
(1,N)
h_LComp
(1,1)
(1,1)
h_AType
(1,N)
Access_ID
name
Access_Type
Server_ID
name
(1,1)
h_server
(1,N)
Server
LogComp_ID
date
lineNumber
Log_Component
(0,1)
h_message
(1,N)
Error_Msg
Msg_ID
desc
Operation_ID
startDate
endDate
Operation
(1,N)
h_operation
(0,1)
(0,1)
h_query
(1,1)
h_operType
(1,N)
Operation_Type
Type_ID
desc
(0,N)
Query
Query_ID
query_text

Modelo Lógico:



Modelo Físico:

El diccionario de datos queda conformado de la siguiente manera:

access_type(#access_id, name)

error_msg(#msg_id, description)

log_component(#logcomp_id, date_, line_number, fk_logentry, fk_operation, fk_msgid, fk_query)

log_entry(#entrylog_id, startdate, enddate, ip_access, fk_server, fk_protocol, fk_level, fk_accesstype, fk_user)

log_level(#loglevel_id, level_desc)

operation(#operation_id, startdate, endDate, fk_operationType)

operation_type(#type_id, description)

protocol(#protocol_id, name)

query(#query_id, query_text)

server(#server_id, name)

user(#user_id, name, user_mode)

user_mode(#userMode_id, name)

➢ NoSQL

A continuación, se presenta un modelo de ejemplo del modelo en JSON (NoSQL), basado en documentos.

```
{
  "_id" : 22593,
  "log_component_id" : NumberLong(36),
  "date" : ISODate("2015-10-12T00:11:25.975Z"),
  "lineNumber" : NumberLong(22593),
  "query" : {
      "query_id" : NumberLong(1),
     "query_text" : "select e.emp_id, emp_name,
            emp_salary,address_line1, city, zipcode from Employee e, Address
    a where a.emp_id=e.emp_id\n",
      "_id" : ObjectId("561afaabc7366f19e81eac40")
  },
  "errorMsg" : {
```

```
        "msg_id" : NumberLong(10),
              "desc" : "Java exception occurred: java.sql.SQLException: Could
        not retrieve transation read-only status server\nat
        com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1078)\nat
        ….     },
    "operation" : {
        "operation_id" : NumberLong(27),
        "startDate" : ISODate("2015-10-12T00:11:21.837Z"),
        "endDate" : ISODate("2015-10-12T00:11:21.864Z"),
        "operation_type" : {
            "operationType_id" : NumberLong(2),
            "desc" : "[READ]"
        }
    },
    "logEntry" : {
        "log_entry_id" : NumberLong(1),
        "start_date" : ISODate("2015-10-12T00:11:25.877Z"),
        "end_date" : ISODate("2015-10-13T00:11:25.877Z"),
        "ip_access" : "145.5.88.2",
        "logLevel" : {
            "loglevel_id" : NumberLong(1),
            "levelDesc" : "DEBUG"
        },
        "accessType" : {
            "access_type_id" : NumberLong(1),
            "name" : "[LOCAL]"
        },
        "protocol" : {
            "protocol_id" : NumberLong(2),
            "name" : " [htpps]"
        },

        "user" : {
            "user_id" : NumberLong(1),
            "name" : "[lisancroce]",
            "user_mode" : {
                "_id" : ObjectId("561afaadc7366f19e81eac81"),
                "user_mode_id" : NumberLong(1),
                "name" : "[ADMIN]"
            },
            "_id" : ObjectId("561afaadc7366f19e81eac83")
        },
        "server" : {
            "server_id" : NumberLong(1),
            "name" : "fs_Tandil"
        },
        "_id" : ObjectId("561afaadc7366f19e81eac8d")
    }
}
```

El modelo en NoSQL está conformado por un conjunto de documentos *"LogComponent".* Notar que los documentos *"Operation*", *"Operation_Type*", "*Query*" y "*ErrorMsg*" se encuentran embebidos en el documento *LogComponent* para poder

manipular de una manera más simple las consultas (cuestión que se tratará a continuación) sobre los patrones. Los documentos *"Protocol", "User", "User_Mode", Log_Level" "Server" y "Access_Type"* se encuentran embebidos en el documento *"Log_Entry"*.

Patrones Web Mining Aplicado al escenario

De acuerdo a los modelos presentados en el presente capítulo (*Modelo E-R* para el modelo relacional y JSON para NoSQL), las operaciones que se realizarán sobre dichos modelos se dividirán en: Clasificación, Reglas de Asociación y Episodios Frecuentes, las cuales corresponden a algunas de las técnicas consideradas patrones Web Mining de uso (explicados en el inciso: "*Patrones aplicables a servidores web"* del presente capítulo*).*
Estas operaciones son consideradas como patrones en un ámbito Web Mining debido a que permiten resolver los requerimientos para los usuarios, navegantes y administradores de logs en escenarios específicos de servidores y clusters. [134]

A continuación se detallarán los requerimientos agrupados por técnica Web Mining que se aplican tanto al modelo Relacional como al modelo NoSQL, donde ambos representan una herramienta de manipulación de logs correspondientes al historial de operaciones realizadas por los usuarios en un servidor web:

Clasificación:

- Obtener los usuarios que realizan siempre consultas que involucran un tipo de operación determinada (Query1_C).
- Obtener las operaciones que se llevan a cabo con el usuario "[lisancroce]" (Query2_C).
- Obtener los LogComponents que tienen en la Query como parte el texto "pTexto"(Query3_C).

Reglas de asociación:

- Obtener la cantidad de operaciones [DELETE] que realizó el usuario "[lisancroce]" durante las últimas 2hs (esta regla de asociación determinará si el usuario Lisandro estuvo eliminando datos de la base de datos en dicho transcurso de tiempo) (Query1_Asoc).
- Obtener la cantidad de Log Entry generados con un tipo de acceso determinado (HTTPS, HTTP o FTP) durante los últimos 90 minutos. (Esta regla de asociación

entre Log Entry y el tipo de acceso determinará qué protocolos fueron los más utilizados recientemente) (Query2_Asoc).
- Obtener la cantidad de Log Component que poseen un error del tipo “unique constraint” en la última hora. (Esta regla de asociación determinará la cantidad de errores de restricción) (Query3_Asoc).

Episodios Frecuentes:

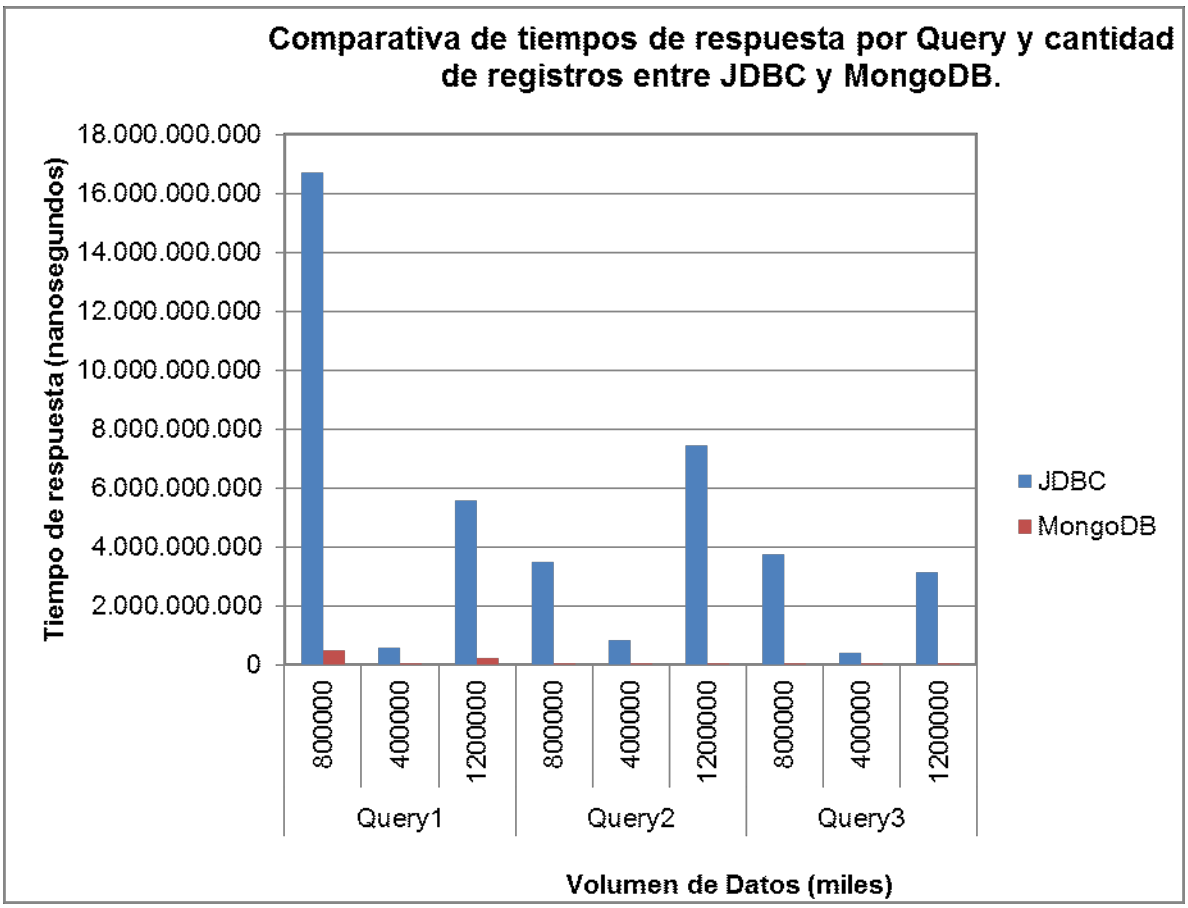
- Obtener la cantidad de operaciones que se realizaron los últimos 30 minutos donde la query lleva la palabra "La Plata" (Query1_EpiF).
- Obtener la cantidad de querys ejecutados en una entrada de log en diciembre del 2015 (Query2_EpiF).
- Obtener para cada tipo de operación su cantidad involucrada en una entrada de log diaria (no hay que trabajar con funciones para cálculo entre fechas se supone que toda la información a trabajar es la que se genera en un día) (Query3_EpiF).
- Idem anterior en los últimos X minutos (Query4_EpiF).

Pruebas sobre los patrones en MongoDB y MySQL

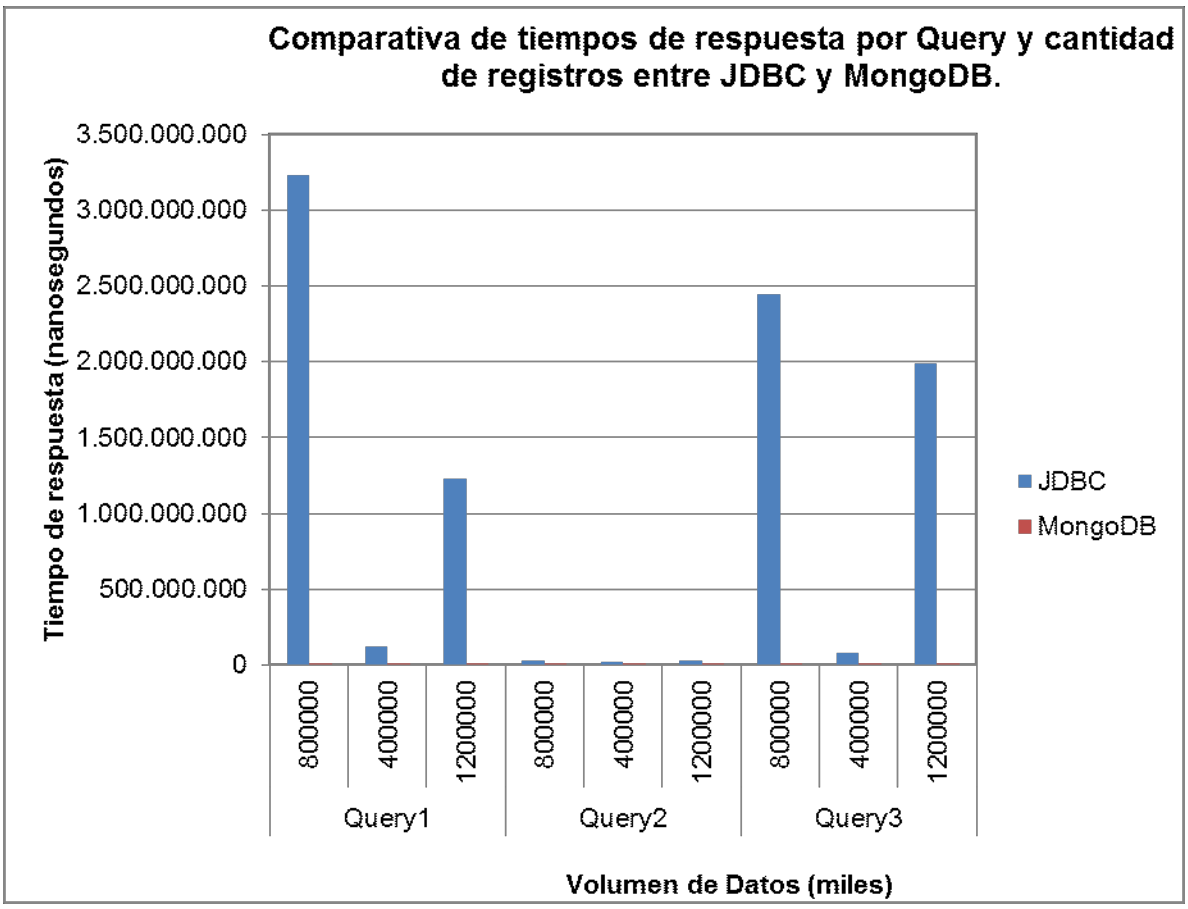
En función de los patrones mencionados anteriormente, se han llevado a cabo las pruebas específicas sobre los mismos en las tecnologías MongoDB (para NoSQL) y JDBC (para el modelo relacional). Las pruebas han sido basadas a que cada query se corresponde a un patrón, donde cada uno de ellos es realizado con la tecnología MongoDB y JDBC realizando la misma lógica para obtener el mismo resultado. De esta manera, es posible analizar el tiempo de ejecución de cada query con cada tecnología (MongoDB y JDBC). La finalización de la ejecución de cada query implica tener disponible la información que responde a cada patrón, con lo cual no se tiene como objetivo de análisis el tiempo de respuesta sobre el procesamiento de la salida de cada una de ellas sino solo la demora reflejada por el llamado al inicio y finalización de ejecución de las mismas.
Como el volumen de información en el tiempo de ejecución influye en el tiempo de respuesta de una query, se analizaron las mismas con tres tipos de volúmenes de información por cada patrón. El tiempo de ejecución sobre las pruebas han sido medidos en el orden nanosegundos, por el hecho de que en MongoDB en algunas querys arrojaron tiempo nulo en milisegundos, con lo cual las gráficas representadas en nanosegundos representan de una manera más marcada la brecha entre la ejecución sobre ambas tecnologías (JDBC y MongoDB). A continuación se muestran los resultados de las pruebas por cada patrón.

Resultados y comparativa de ejecución de los patrones

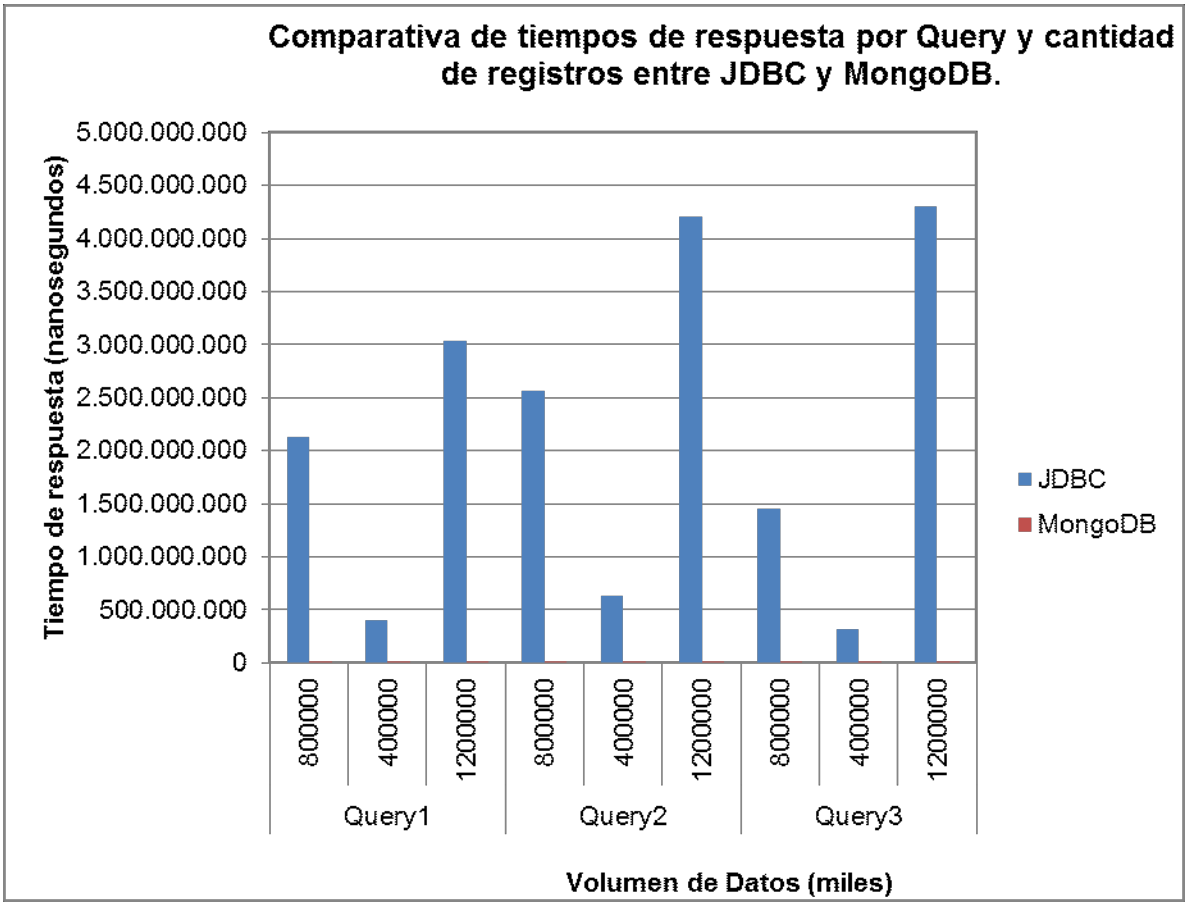
CLASIFICACION



REGLAS DE ASOCIACION



EPISODIOS FRECUENTES



CAPÍTULO VI
Conclusiones y Trabajo a Futuro

Introducción

En esta tesina se han desarrollado a lo largo de los capítulos de manera gradual diversos temas asociados a Base de Datos a Gran Escala, específicamente NoSQL, lo que resultó en un análisis final de un escenario específico basado en los resultados sobre pruebas ejecutadas en SQL (como tecnología de base de datos relacional) y MongoDB (como tecnología sobre base de datos NoSQL) sobre un modelo específico de log.

A continuación, se procederá a describir los principales aportes de la tesina y luego las conclusiones generales sobre los temas abordados en la presente tesis. Al haberse tratado diferentes aspectos (que han sido detallados del Capítulo 1 al 5), la clasificación sobre las conclusiones serán consideradas sobre un nivel teórico y nivel práctico.

Principales aportes de la tesina

El propósito de la tesina fue darle mayor importancia a la investigación de las bases de datos NoSQL por sobre las bases de datos a gran escala. Este aspecto incentivó a la investigación de temas como el comportamiento y uso de las bases de datos NoSQL en la actualidad. Los diferentes tipos de bases de datos NoSQL ayudan a determinar si una herramienta perteneciente a uno de ellos puede acoplarse con menos o más facilidad a un determinado modelo a representar. Es decir, en el caso que un analista necesite modelar un conjunto de datos podrá optar por el tipo de base de datos NoSQL que se asemeje más al modelo en cuestión.
Desde el punto de vista de las características y diferencias con las bases de datos relacionales, uno de los aportes más importantes está relacionado con la terminología BASE respecto a las bases de datos NoSQL, en contrapunto con ACID que se corresponde a las bases de datos relacionales. De esta manera, se tienen en cuenta ambos conceptos al momento de trabajar con transacciones.

Por otro lado, la flexibilidad en las bases de datos NoSQL es la característica que se tuvo en cuenta al momento de representar el modelo de logs, de esta manera, se logró mostrar la ventaja que brindan las bases de datos NoSQL sobre cualquier

esquema de datos al momento de almacenar y recuperar la información. Esto último sumado al hecho de poder almacenar la información en dos modelos distintos, permitió analizar y confeccionar consultas, las cuales fueron agrupadas dentro de los patrones Web Mining existentes. La importancia de llevar a cabo una comparación entre dos modelos diferentes (que si bien representaron los mismos datos aunque de diferente manera y llevarlo a un ámbito como lo es Web Mining) ha logrado obtener resultados en favor de NoSQL en cuanto a tiempo de respuesta de las consultas.

Conclusiones

A continuación se describirán las conclusiones divididas en un nivel teórico y práctico sobre todo lo trabajado en la presente tesina. El nivel teórico se corresponde a las conclusiones propias realizadas en función de definiciones, herramientas encontradas y casos de estudio que han sido investigados, analizados y extraídos de bibliografías oficiales y/o confiables. Por otro lado, el nivel práctico corresponde a las conclusiones realizadas en base al escenario y al concepto Web Mining presentado en el *Capítulo V - Escenario* en conjunto con las herramientas investigadas para agilizar el análisis, desarrollo y pruebas realizadas con los patrones Web Mining sobre el modelo de Log.

Nivel Teórico

En principio, la investigación que se ha realizado sobre NoSQL y los usos que se han presentado en la presente tesis, deriva a la conclusión que dicho concepto ha avanzado considerablemente en el último tiempo (por ejemplo los últimos dos años). Esta consideración es justificable debido a que organizaciones de nombre como por ejemplo Oracle, han consolidado productos NoSQL referido a un motor de base de datos NoSQL, en este caso denominado Oracle NoSQL (lanzado en el 2012) [146].
Además, las bases de NoSQL no solo fueron utilizadas por aplicaciones de Internet como las redes sociales (Twitter y Facebook) sino que también son utilizadas por algunas áreas de la informática como lo son Big Data, Data Warehouse o Business Intelligent [150], las cuales utilizan herramientas NoSQL para la manipulación de los datos. Esto último justifica la variedad de herramientas NoSQL que existen desde el año 2000 hasta el momento y sus utilidades más conocidas.
Esto quiere decir que si se remonta al menos a una década el panorama era distinto, en el sentido que cuando se mencionaba sobre NoSQL se trataba sobre una tecnología aún con falta de madurez.

En cuanto a los tipos de base de datos que se han investigado, se puede concluir que cada tecnología o herramienta utiliza un tipo de base de datos particular.

Se puede notar que si bien hay diferentes tipos de base de datos particulares, hay algunas que suelen ser más utilizadas, la más común es la orientada a documentos ya que reemplazó a las bases de datos orientadas a grafos. Al haber investigado las utilidades y herramientas de desarrollo, se puede afirmar que las bases de datos orientadas a documentos son actualmente las más utilizadas en las diferentes tecnologías que existen, por mencionar algunas como las de MongoDB y la mayoría de las ofrecidas por Google (como se ha mencionado en el *Capítulo III - Base de Datos Orientada a Documentos y Capítulo IV - MongoDB).* La popularidad que ganó el formato JSON hizo posible también el optar por este tipo de base de datos, tanto para las tecnologías mencionadas como para el desarrollo del escenario de la tesina.
Además, se optó por MongoDB para la utilización NoSQL pues maneja una única representación: el Documento, y soporta lenguajes de programación que se utilizan en la actualidad, como por ejemplo: Java, NodeJS, Python y PHP, lo cual hace que este trabajo de tesis pueda ser utilizado por futuros trabajos de investigación que deseen utilizar el modelo de log presentado y realizar operaciones con estos lenguajes de programación u otros IDE's adaptados a estos últimos. De todas maneras, pudo haberse utilizado CouchDB que también es una base de datos orientada a Documento, pero se tuvo en cuenta a MongoDB que favorece la Consistencia [152], mientras que CouchDB favorece a la Disponibilidad [153]

En cuanto a la investigación sobre las tecnologías que utilizan NoSQL, se puede notar que constantemente se lanzan nuevas mejoras y funcionalidades, lo que lleva a estar continuamente a la expectativa sobre las novedades de dichas tecnologías sobre lenguajes de programación y herramientas que utilizan NoSQL, como por ejemplo MongoDB (tratado en el *Capítulo IV-MongoDB* de la presente tesina).

Nivel Práctico

El modelo de Log que se ha presentado en el *Capítulo V* (del cual ha sido basado el escenario) permitió comprender un diseño sencillo por el hecho de tener un número menor a las 20 entidades que pueden resolver una situación real para el análisis de log. Con lo cual, se puede concluir que ha sido considerablemente útil para el análisis de las respuestas basadas en resultados de tiempos de ejecución. A su vez, dicho modelo también puede ser adaptable a una situación similar sobre análisis de servidor, sitio web, entre otros similares. El modelo relacional, al menos como panorama inicial para visualizar el diseño presentado, resultó más legible que el modelo JSON, ya que para representar un modelo JSON se debe remitir a algún ejemplo de información ya cargada.

Previamente a las pruebas realizadas sobre el escenario presentado en el Capítulo V, se realizaron cargas masivas de datos sobre los modelos en las tecnologías MongoDB y JDBC. Dichas cargas fueron realizadas en base a procesos randomizados para obtener datos reales de prueba, donde se obtuvieron resultados de

tiempos de respuestas ampliamente superiores (en algunos casos mayores al 50%) en el modelo relacional con la tecnología JDBC con respecto al modelo NoSQL con la tecnología MongoDB. Por ejemplo, para carga de datos de una posible migración, si el objetivo fundamental es mejorar los tiempos de respuesta no es recomendable utilizar las cargas que se trabajaron en este capítulo.
Para las pruebas que se realizaron en el *Capítulo V - Escenario* se consideró el concepto *Web Mining* que engloba a patrones de búsquedas y consultas, de los cuales se tomaron los patrones Clasificación, Reglas de asociación y Episodios Frecuentes pertenecientes a la rama *Web Usage Mining,* por tratarse de un proceso de extracción de la información a partir de los registros del servidor y el historial de los usuarios. Esto último resultó interesante para aplicarlo a un modelo Log específico, adaptando consultas pertenecientes a los patrones Web Mining para luego llevarlas a JDBC y MongoDB. Luego de las distintas pruebas realizadas para diferentes cantidades de registros, los resultados fueron favorables en todos los casos al modelo NoSQL con la tecnología MongoDB, con lo cual en algunas querys los resultados de tiempo de ejecución arrojan tiempo 0 (cero) en milisegundos y por este motivo fue necesario informarlos en el orden de nanosegundos. Esta cuestión resultó interesante por el hecho de que se manipularon como máximo 1.200.000 registros y, se podría deducir que en el caso que se tratara por ejemplo de al menos cientos de millones (almacenados en servidores), la diferencia sería aún más notoria, pues se estaría manipulando volúmenes de información mayores al 1.000 %.
Las gráficas presentadas en el Capítulo V visualizan los resultados sobre los tiempos de ejecución y se puede concluir que para reglas de asociación y episodios frecuentes los tiempos de ejecución son muy favorables en MongoDB, al menos hasta los volúmenes de información que se trabajaron (1.200.000 registros de Log Components). Para las reglas de clasificación puede notarse una mayor cantidad de procesamiento sobre las querys con respecto al resto de los otros patrones, ya que para MongoDB se alcanza a visualizar su gráfico de barra correspondiente; de todas maneras el tiempo de ejecución continúa siendo muy favorable para MongoDB (de manera similar a las querys de los otros patrones). Con todas estas pruebas, se llega a la conclusión de que es altamente recomendable el uso de NoSQL al menos en el procesamiento de todas las querys involucradas sobre los patrones encontrados sobre el modelo de log presentado.

En cuanto a las conclusiones sobre las herramientas utilizadas en el *Capítulo V-Escenario*, se procede a mencionarlas por cada herramienta:

- **XAMPP**: utilizado como gestor de la base de datos MySQL y servidor Web Apache. Resultó simple en el sentido que se utilizó la interfaz gráfica para crear la base de datos relacional y diseñar las tablas correspondientes. A su vez, ha aportado la funcionalidad de verificar la carga de datos, realizar las consultas SQL, validar y verificar los resultados obtenidos en cada query.

- **Eclipse IDE** ha resultado de mucha utilidad al desarrollo de los métodos que interactúan con las librerías JDBC y MongoDB. Brindó la posibilidad de utilizar el

lenguaje Java (entre otros), aportar visualización a errores de conexión, DEBUG y cálculo en tiempos de respuesta de cada operación. También resultó útil para asociar la librería de control de versiones (SVN) [151] para desarrollar de manera compartida en un repositorio común para que el código pueda ser compartido por desarrolladores.

- **JDBC** y **MongoDB:** JDBC ha sido utilizado como herramienta para trabajar sobre las operaciones sobre el modelo relacional realizadas en el escenario. Sobre dicha herramienta se utilizó SQL, ya que JDBC es una API que trabaja con SQL como dialecto para conectarse a la base de datos. JDBC resultó simple en el sentido que no se necesitó ninguna librería adicional para operar sobre el modelo relacional y solo con tener conocimientos básicos en SQL y en el motor de base de datos MySQL alcanzó para realizar todas las operaciones realizadas en el *"Capítulo V - Escenario".* Por otro lado, MongoDB ha permitido trabajar sobre el modelo NoSQL y también resultó simple su utilización en el sentido que sin tener conocimientos previos de utilización de MongoDB, la documentación básica oficial permitió desarrollar las pruebas del escenario. En el uso de MongoDB se intentó aplicar los índices para las búsquedas, pero estos no mejoraron considerablemente el tiempo de ejecución y además, producen un consumo de memoria adicional que para este caso no fue conveniente, por lo tanto, se procedió a no aplicarlos. En el modelo relacional con JDBC se tuvo que adaptar la codificación al momento de trabajar con fechas, es decir, con campos del tipo DATE y por otro lado, modificar el tipo de dato en las tablas con campos DATE para reemplazarlos por TimeStamp debido a que no se realizaba la actualización correspondiente en estos campos. Es decir, el almacenamiento se realizaba con valores nulos y, en otros casos, solo la fecha sin hora, minuto y segundo. En cambio, en el modelo NoSQL se verificó que MongoDB asegura el almacenamiento del valor con el formato indicado sin necesidad de especificar un tipo de dato (Date o TimeStamp) como ocurre en una base de datos relacional, pues NoSQL trabaja con datos semiestructurados.

- **RoboMongo:** permitió la visualización global del modelo NoSQL (representado en modelo JSON). Es útil en el sentido que puede obtenerse un panorama visual de todos los documentos del modelo sin necesidad de exportar la base de datos. A su vez la misma herramienta permite realizar consultas sencillas sobre el modelo sin necesidad de programar y, por ejemplo, en el caso de borrar colecciones de documentos resultó más rápido el tiempo de ejecución en esta herramienta en comparación del resultado que se obtuvo por medio de la programación.

Futuras líneas de investigación

El marco desarrollado y diferentes cuestiones que se han presentado sobre la presente tesis contribuyen a despejar algunas incógnitas y a su vez generar nuevas preguntas, ideas y/o abrir nuevas vías de trabajo. En esta sección se presentan algunas líneas de investigación que pueden ser objeto de interés en lo que se refiere a todo el trabajo expuesto en la presente tesis:

- En relación con el modelo de log presentado en el *Capítulo V,* la metodología descrita puede aplicarse a todo tipo de aplicaciones, en el sentido del marco de diseño y tecnologías de desarrollo utilizadas. Puede ser interesante obtener modelos de otros tipos de aplicaciones y efectuar un estudio detallado de su funcionamiento, pues existen diversas aplicaciones de redes sociales, carritos de compra, entre otras que siguen una línea similar de diseño al modelo de log presentado. De esta manera es probable descubrir particularmente un "standard" de diseño sobre estos tipos de aplicaciones.

- Como se ha mencionado en el presente capítulo en la sección *Pruebas realizadas* en referencia al alta de datos masivos (que llevaron a la carga sobre el modelo presentado en el *Capítulo V -Escenario)*, existe la inquietud sobre la posibilidad de mejorar los tiempos de respuesta de MongoDB a diferencia con MySQL. Algunas alternativas pueden consistir en utilizar el lenguaje MongoDB nativo [147] sin necesidad de utilizar alguna librería adicional asociado a un lenguaje de programación en particular, esto se asume que genera un tiempo extra de ejecución pero debería probarse para verificar realmente una mejora considerable. Otra alternativa es utilizar un mapeador específico de MongoDB, como por ejemplo *Morphia* [148]. Los mapeadores en la mayoría de los casos utilizan cachés que incrementan el rendimiento en la ejecución de operaciones CRUD [149]. Como última alternativa se podría probar con alguna versión futura posterior a la actual *v3.2* de MongoDB para poder verificar una posible mejora, pero ello implicaría esperar el tiempo necesario hasta el lanzamiento de una nueva versión.

- En cuanto a documentos embebidos en MongoDB existe el límite de 16MB por documento sin utilizar ningún mapeador para esta tecnología. Al embeber documentos se tiene una opción adicional para diseñar, una idea interesante sería medir el tiempo de respuesta utilizando un mapeador para MongoDB (pues por ejemplo *Morphia [148]* admite documentos mayores a 16MB) y otro para MySQL sobre las pruebas realizadas en el *Capítulo V - Escenario*. Analizando los resultados obtenidos se podría verificar si existen algunas ventajas para unos casos y/o desventajas para otros.

- Con respecto a las pruebas realizadas, las mismas fueron ejecutadas en un contexto reducido (tratado en el *Capítulo V*). Podría considerarse un trabajo a futuro realizar otras pruebas en un ambiente distribuido y pudiendo manipular un

volumen de datos mayor a 5 millones de registros, con el objetivo de obtener otros tiempos de respuesta. Se debería armar un entorno particular para realizar este tipo de pruebas.

Desde el punto de vista de los tipos de bases de datos NoSQL descrito en el *Capítulo 3-Clasificación de bases de datos NoSQL* sería una buena práctica representar el modelo de log en otra base de datos NoSQL que no sea una del tipo orientada a documentos, sino en otra base de datos, como por ejemplo, Cassandra (orientada a clave/valor) o HBase (orientada a columna) y poder comparar los tiempos de respuesta.

Bibliografía:

[1] Big data : la revolución de los datos masivos, 2013, Ed:Turner, Viktor Mayer Schönberger, pp19.

[2] Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL and Graph, 2013, Ed: Elsevier, David Loshin, pp34.

[3] Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL and Graph, 2013, Ed: Elsevier, David Loshin, pp15.

[4] Big Data Imperatives: Enterprise 'Big Data' Warehouse, 'BI' Implementations and Analytics, 2013, Ed: Apress, Soumendra Mohanty, Madhu Jagadeesh, Harsha Srivatsa, pp14.

[5] Handbook of Research on Cloud Infrastructures for Big Data Analytics, 2014, Ed: IGI Global, Raj, Pethuru, pp52.

[6] Architecting High Performing, Scalable and Available Enterprise Web Applications, 2014, Ed: Morgan Kaufmann, Shailesh Kumar Shivakumar, pp38.

[7] Official Blog Twitter [online].
Disponible en:
https://blog.twitter.com/2011/200-million-tweets-day

[7] Diario WhashingtonPost [online]
http://www.washingtonpost.com/blogs/faster-forward/post/twitter-users-send-200-million-tweets-per-day/2011/07/01/AGEFBUtH_blog.htmlhttp://www.washingtonpost.com/blogs/faster-forward/post/twitter-users-send-200-million-tweets-per-day/2011/07/01/AGEFBUtH_blog.html

[8] "NoSQL: a non-SQL RDBMS".
Disponible en:
http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page

[9] Evans, Eric. "Erics Evans' Web Log" – "NoSQL: What's in a name?".
Disponible en:
http://blog.sym-link.com/2009/10/30/nosql_whats_in_a_name.html

[10] Gaurav Vaish, "Getting Started with NoSQL". 2013. Ed: Packt Publishing, pp 9.

[11]Abraham Silberschatz, Henry F. Korth, S. Sudarshan | McGraw-Hill: Fundamentos de bases de datos (4ta. edición), 2002, pp 367.

[12] Etzioni, O. (1996), "the World Wide Web: Quagmire or Gold Mine?". Comunications of the ACM, november 1996, Vol.39, No.11

[13] Revista Gerencia: NoSQL: ¿el fin del reinado de las bbdd relacionales?
http://www.emb.cl/gerencia/articulo.mvc?xid=105

[14] Disponible en:
http://mike2.openmethodology.org/wiki/Big_Data_Definition

[15] Disponible en:
http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[16] Disponible en:
http://www.oracle.com/es/products/database/big-data-appliance/overview/index.htmll

[17] Teradata: Disponible en:
http://www.teradata.com.sa/News-Releases/2014/Teradata-and-MongoDB-Inc-Empower-Big-Data-Strategies-with-JSON-Integration/?LangType=1025&LangSelect=true

[18]
Pentaho. Disponible en:
http://www.pentaho.com/big-data-analytics/nosql-databases

[19] Cloud Computing. Disponible en:
http://www.nist.gov/itl/cloud/

[20] Software como Servicio (SaaS):
http://netforbeginners.about.com/od/s/f/what_is_SaaS_software_as_a_service.htm

[21] ClearDB
https://www.cleardb.com/

[22] Jeremiah Peschka. “Facility9”.
Disponible en:
http://facility9.com/2010/09/five-reasons-to-use-nosql/

[23] Voldemort. “Project Voldemort – A distribuited database”.
Disponible en:
http://www.project-voldemort.com/voldemort/

[24] Redis - NoSQL de alto rendimiento
http://altenwald.org/2012/06/21/redis-nosql-de-alto-rendimiento/

[25] Cloud Computing - nueva era de desarrollo
http://www.maestrosdelweb.com/cloud-computing-nueva-era-de-desarrollo/

[26] Bases de datos Libres

http://tubasededatoslibre.org/site/bases-de-datos-nosql/

[27] Google- Google App Engine.
Disponible en:
https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=es

[28] Amazon – Amazon S3.
Disponible en:
http://aws.amazon.com/es/s3/

[29] MongoDB
Disponible en:
http://www.mongodb.com/solutions

[30] Bill Wilder, “Cloud Architecture Patterns” . Ed. O’Reilly,, pp 54.

[31] Microsoft. “Windows Azure”.
Disponible en:
http://www.windowsazure.com/en-us/documentation/services/storage/

[32] NoSQL: Introducción a las bases de datos no estructuradas, escrito por Dr. Diego López de Ipiña González de Artaza. Disponible en: http://es.slideshare.net/dipina/nosql-introduccin-a-las-bases-de-datos-no-estructuradas

[33] Amazon.com, “Dynamo: Amazon’s Highly Available Key-value Store”. Giuseppe Decandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels.

[34] Luis Rubén Artavia y Mario Villalobos, *“Relación entre las bases de datos NoSQL y Big Data”*. Disponible en: http://bb9.ulacit.ac.cr/tesinas/Publicaciones/043241.pdf

[35] G. DeCandia et al., “Dynamo: amazon’s highlyavailable key-value store,” SIGOPS Oper. Syst.Rev., vol. 41, no. 6, pp. 205–220, Oct. 2007.

[36] TALENS, JAH: “Bases de datos clave-valor”. Disponible en:
http://softinspain.com/desarrollo/bases-de-datos-clave-valor/.

[37] ondho, “Claves para elegir tu base de datos NoSQL”. Disponible en: https://www.ondho.com/claves-para-elegir-tu-base-de-datos-nosql/

[38] Oracle Technology Network - Oracle Berkeley DB.
Disponible en:
http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

[39] EH Cache. Disponible en: http://ehcache.org/

[40] Memcached. Disponible en: http://memcached.org/

[41] Redis. Disponible en: http://redis.io/

[42] Amazon WebServices, *"DynamoDB"*
Disponible en: http://aws.amazon.com/es/dynamodb/faqs/

[43] Fundación Universidad Rey Juan Carlos (Madrid), Red de Innovación y Transferencia en Gestión de Datos, *"Bases de datos No Relacionales (NoSQL)"*, Pág.38.
Disponible en: http://es.slideshare.net/dipina/nosql-cassandra-couchdb-mongodb-y-neo4j?related=1

[44] Analisis y desarrollo de MongoDB y Redis en Java, Escrito por Francisco Javier Ruano Vázquez. Capítulo 3. Versión Kindle. Edición 2014.
Disponible en: http://libros.nom.es/java

[45] Analisis y desarrollo de MongoDB y Redis en Java, Escrito por Francisco Javier Ruano Vázquez. Capítulo 2.2.1. Versión Kindle. Edición 2014.
Disponible en: http://libros.nom.es/java

[46] Anthony R. Sotolongo León, "*Bases de datos orientadas NoSQL a orientadas a documentos".*
Disponible en: http://es.slideshare.net/asotolongo/bases-de-datos-nosql-orientadas-a-documentos (Pág. 3)

[47] Jonhatan Wiesel, *"MongoDB desde cero"*.
Disponible en: http://codehero.co/mongodb-desde-cero-modelado-de-datos/

[48] MONGODB. 2012. MongoDB. *Sitio Ofical del proyecto MongoDB.*
Disponible en: [www.mongodb.org].

[49]COUCHDB-1. CouchDB. *Why Large Hadron Collider Scientists are Using CouchDB.*
Disponible en: [http://www.readwriteweb.com/enterprise/2010/08/lhc-couchdb.php].

[50] TERRASTORE. Terrastore. *Sitio ofical del proyecto Terrastore.*
Disponible en: [http://code.google.com/p/terrastore].

[51] Yudisney Vazquez ortiz, Anthony Rafael Sotolongo León, *"Mirada a bases de datos NoSQL de código abierto orientadas a documentos".*
Disponible en: [http://publicaciones.uci.cu/index.php/SC/article/view/1382]

[52] Sergio Montoro, "NoSQL para no programadores"
Disponible en: http://lapastillaroja.net/2012/02/nosql-for-non-programmers/ Sitio Oficial del libro: "La pastilla roja", Editorial Open:Service, Alfredo Romeo y Juantomás Garcia.

[53] MONGODB CRUD OPERATIONS, "Modelo de Interrelación entre documentos".
Disponible en: http://docs.mongodb.org/manual/applications/data-models-relationships/

[54] Dr. Diego López de Ipiña Gonzalez de Artaza, Universidad de Deusto, *"NoSQL: Introducción a las bases de datos no estructuradas"*, Pág. 20.
Disponible en: http://es.slideshare.net/dipina/nosql-introduccin-a-las-bases-de-datos-no-estructuradas?next_slideshow=1

[55] MarketingDirecto.com.
Disponible en: http://www.marketingdirecto.com/actualidad/marketing/datos-simples-semiestructurados/

[56] Yoselin Salazar C., Daniel Herrera C., Rebeca Brenes R., "MongoDB".
Disponible en: https://prezi.com/izjqg7o_aqqn/mongodb/

[57] Hadoop. *"MapReduce Tutorial"*.
Disponible en: http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[58] MongoDB. "GridFS"
Disponible en: http://docs.mongodb.org/manual/core/gridfs/

[59] VALERO, Nelica. Bases de Datos Columnares. 2009. Disponible en: http://www.gravitar.biz/index.php/bi/base-datos-columnar/

[60] NoSQL: Introducción a las bases de datos no estructuradas, escrito por Dr. Diego López de Ipiña González de Artaza. Disponible en: http://es.slideshare.net/dipina/nosql-introduccin-a-las-bases-de-datos-no-estructuradas, Pág. 19.

[61] Google Inc, "BigTable: A Distributed Storage System for Structured Data".
Disponible en: http://www.nosql.es/blog/wp-content/uploads/2010/04/bigtable-osdi06.pdf

[62] T. Wellhausen (2012, May),"Highly Scalable, Ultra-Fast and Lots of Choices: A Pattern Approach toNoSQL", Tim-Wellhausen [Online]. Disponible en: http://tim-wellhausen.de/papers/NoSQL-Patterns.pdf

[63] René E. Cuevas Valencia, Angeles Cruz Manjarrez Antaño, José Mario Martínez Castro. *"Migración de bases de datos SQL a NoSQL".* Revista Tlamati, Número Especial 3 CICOM 2014.

[64] Gheorghe MATEI, Romanian Commercial Bank, Bucharest, ROMANIA, "Column-Oriented Databases, an Alternative for Analytical Environment", Vol. I, no. 2/2010.

[65] David Loshin. "Gaining the Performance EdgeUsing a Column-OrientedDatabase Management System". Disponible en: http://www.caci.com/contracts/ites/gainingtheperformanceedge_sybase.pdf

[66] NoSQL.es. Cassandra. Disponible en: http://www.nosql.es/blog/nosql/cassandra.html

[67] IBM. What's Hadoop?. Disponible en: http://www-01.ibm.com/software/data/infosphere/hadoop/

[68] APACHE HBASE. HBase. Disponible en: https://sitiobigdata.com/que-es-hbase/

[69] Komadinovic Vanja, Vast Platform team, "Google BigTable". Disponible en: http://es.slideshare.net/vanjakom/google-bigtable-paper-presentation-5078506?next_slideshow=2

[70] Kuper, G. M., & Vardi, M. Y. (1993). The logical data model. ACM Transactions on Database Systems (TODS), 18(3), 379-413.

[71] Bondy, John Adrian, and Uppaluri Siva Ramachandra Murty. Graph theory with applications. Vol. 290. London: Macmillan, 1976.

[72] Bray, Tim, et al. Extensible markup language (XML). World Wide Web Journal 2.4 (1997): 27-66.

[73] Buerli, M., & Obispo, C. P. S. L. (2012). The Current State of Graph Databases.

[74] Ramas, Mauro San Mart¬n, Claudio Gutierrez, and Peter T. Wood. SNQL: Social Networks Query Language. (2011).

[75] Wasserman, Stanley, and Katherine Faust. Social network analysis: Methods and applications.
Vol. 8. Cambridge university press, 1994.

[76] Trinajstiac, N. Chemical graph theory. Boca Raton, FL: CRC press, 1983.

[77] Graves, M., Bergeman, E. R., & Lawrence, C. B. (1995). Graph database systems. Engineering in Medicine and Biology Magazine, IEEE, 14(6), 737-745.

[78] Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., & Tomkins, A. S. (1999). The web as a graph: Measurements, models, and methods. In Computing and combinatorics (pp. 1-17). Springer Berlin Heidelberg.

[79] de Solla Price, Derek J. Networks of scienti.c papers. Science 149.3683 (1965): 510-515.

[80] Introducing the Knowledge Graph: things, not strings.
http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html,
2012.

[81] Bollacker, Kurt, et al. Freebase: a collaboratively created graph database for structuring human knowledge. Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008.

[82] Kasneci, Gjergji, et al. The YAGO-NAGA approach to knowledge discovery. ACM SIGMOD Record 37.4 (2009): 41-47.

[83] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scienti.c american, 284(5), 28-37.

[84] World Wide Web Consortium. Home. http://www.w3.org/, 2013.

[85] Graph Database. Wikipedia. http://en.wikipedia.org/wiki/Graph_database, 2013.

[86] Angles, Renzo. A comparison of current graph database models. En Data Engineering Work-
shops (ICDEW), 2012 IEEE 28th International Conference on. IEEE, 2012. p. 171-177.

[87] AllegroGraph 4.10. http://www.franz.com/agraph/allegrograph/, 2013.

[88] Martínez-Bazan, Norbert, et al. Dex: high-performance exploration on large graphs for information retrieval. Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. ACM, 2007.

[89] Sparsity-Technologies. Home. http://www.sparsity-technologies.com, 2013.

[90] HyperGraphDB. Home. http://www.hypergraphdb.org/index, 2012.

[91] In.niteGraph. Home. http://www.in.nitegraph.com, 2012.

[92] Neo4j. Home. http://neo4j.org, 2013.

[93] INFOTECHNOLOGY.COM, *"Oficial: Twitter tiene 4,7 millones de usuarios activos en la Argentina".*
Disponible en: http://www.infotechnology.com/internet/Oficial-Twitter-tiene-47-millones-de-usuarios-activos-en-la-Argentina-20140611-0004.html

[94] Arume, Desenvolvementos Informáticos, *"Estructura Informática en bases de datos jerárquicas".*
Disponible en: http://www.arumeinformatica.es/blog/estructuras-jerarquicas-de-datos-en-bases-de-datos-relacionales/

[95] C.J. Date, *"Introducción a los sistemas de bases de datos"*, Editorial Pearson Educación, Año 2001, Cap. 3 – Bases de Datos basadas en la lógica.

[96] Ing Hansel Gracia del Busto, Ing Osmel Yanes Enríquez, *"Bases de datos NoSQL",* TELEMATICA – Revista Digital de las tecnologías de la información y las comunicaciones. Pág. 29 y 30.
Disponible en: http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/74/74

[97] Ariel Andres Nazca, "Introducción a NoSQL – Graph Database Neo4j"
Disponible en: https://prezi.com/xaitejgkpqsv/introduccion-a-nosql-graph-database-neo4j/

[98] 10gen
Disponible en:
http://techcrunch.com/tag/10gen/

[99] GNU - AGPL.
Disponible en:
http://www.gnu.org/licenses/licenses.es.html

[100] MongoDB: Data Modeling Introduction
Disponible en:
http://docs.mongodb.org/manual/core/introduction/

[101] BSON:
Disponible en:
http://bsonspec.org/

[102] David Hows, Peter Membrey, Eelco Plugge,"MongoDB Basics", Ed.Apress, pp10.

[103] MongoDB: Data Modeling introduction.
Disponible en:
http://docs.mongodb.org/manual/core/data-modeling-introduction/

[104]MongoDB: Collection
Disponible en:
https://www.mongodb.org/about/introduction/

[105]MongoDB: Data Model Design
Disponible en:
http://docs.mongodb.org/manual/core/data-model-design/

[106] MongoDB: Model One-to-One Relationships with Embedded Document
Disponible en:
http://docs.mongodb.org/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

[107] MongoDB: Model One-to-Many Relationships with Embedded Document
Disponible en:
http://docs.mongodb.org/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/

[108] MongoDB CRUD Operation
Disponible en:
http://docs.mongodb.org/manual/core/crud-introduction/

[109]MongoDB CRUD Operations
Disponible en:
http://docs.mongodb.org/manual/crud/

[110]MongoDB MapReduce
Disponible en:
http://docs.mongodb.org/manual/core/map-reduce/

[111] JavaScript
Disponible en:
http://www.w3schools.com/js/

[112]Proceso *mongod*
Disponible en:
http://docs.mongodb.org/manual/reference/program/mongod/#bin.mongod

[113]Perform incremental MapReduce
Disponible en:
http://docs.mongodb.org/manual/tutorial/perform-incremental-map-reduce/

[114] MongoDB: Write Operations Overview.
Disponible en:
http://docs.mongodb.org/manual/core/write-operations-introduction/

[115]MongoDB: Index Introduction
Disponible en:
http://docs.mongodb.org/manual/core/indexes-introduction/

[116]MongoDB: Replication Introduction
Disponible en:
http://docs.mongodb.org/manual/core/replication-introduction/

[117]MongoDB:Sharding Introduction
Disponible en:
http://docs.mongodb.org/manual/core/sharding-introduction/

[118] Kristina Chodorow, MongoDB: The Definitive Guide, 2nd Edition, O'Reilly Media ,pp 72

[119] Kristina Chodorow, MongoDB: The Definitive Guide, 2nd Edition, O'Reilly Media ,pp 67

[120] MTV Networks
Disponible en:
https://www.mongodb.com/customers/mtv-networks

[121] Craiglist Foundation
Disponible en:
http://craigslistfoundation.org/

[122] Foursquare
Disponible en:

https://es.foursquare.com/

[123] Conoce MongoDBMadrid.
Disponible en:
https://www.mongodb.com/events/conoce-mongodb-madrid

[124] OS X
Disponible en:
https://www.apple.com/es/osx/

[125] Windows
Disponible en:
https://www.microsoft.com/en-us/windows

[126]MongoDB:GridFS
Disponible en:
http://docs.mongodb.org/manual/core/gridfs/

[127]MongoDB: chunks
Disponible en:
http://docs.mongodb.org/manual/core/gridfs/#chunk-disambiguation

[128] David Hows, Peter Membrey, Eelco Plugge,"MongoDB Basics", Ed.Apress, pp11.

[129] MongoDB: Source
Disponible en:
https://www.mongodb.org/about/source-code/

[130]FAQ:MongoDB Fundamentals
Disponible en:
http://docs.mongodb.org/manual/faq/fundamentals/

[131]Gaurav Vaish, Getting Started with NoSQL, 2013, Packt Publishing, pp 53.

[132]Computerworld: No to SQL? Anti-database movement gains steam. June 2009. [online]
Disponible en:
http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_-movement_gains_steam

[133]Computerworld: No to SQL? Anti-database movement gains steam. June 2009. [online]
Disponible en:
http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_-movement_gains_steam

[134] Shashank Tiwari, Professional NoSQL, 2011, Ed. Wiley, pp5.

[135] Dan McCreary & Ann Kelly, Making Sense of NoSQL , 2014, Manning Shelter Island, pp6

[136] Alexander Graubner-Muller, Web Mining in Social Media, 2011, Social Media Verlag, pp11.

[137] Kosala, R. and Blockeel, H. Web Mining Research. (2000). A.Survey. ACM SIGKDD Explorations, Newsletter of the Special Interest Group on Knowledge Discovery and Data Mining. Page 1-9, 2000.

[138] Etzioni, O. (1996), "the World Wide Web: Quagmire or Gold Mine?". Comunications of the ACM, november 1996, Vol.39, No.11

[139] Wilks, Y. (1997). Information Extraction as a Core Language Technology Source Lecture Notes In Computer Science; Vol. 1299 Pages: 1-9, 1997.

[140] Wang, Y. (2008). “Web Mining and Knowledge discovery of usage patterns - A survey”.

[141] Liu, B. and Chen-Chuan Chang, Kevin. (2005). Editorial: “Special Issue on Web Content Mining”. WWW 2005 Tutorial, Page 1-4, 2005.

[142] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. Journal of Knowlegde and Information Systems, 1(1):5–32, 1999.

[143] Srivastava J, Cooley R., Deshpande M., Tan P-N., Web Usage Mining: Discovery and aplications of usage patterns from web data. In WEBKDD (1999) Pág. 163 - 182.

[144] Z. Markov and D. T. Larose. Data Mining the Web: Uncovering Patterns in Web Content, Structure and Usage. John Wiley and Sons, New York, USA, 2007

[145] H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovering frequent episodes in sequences,” in In Proceedings of the 1st International Conference on Knowledge Discovery in Databases and Data Mining, 1995, pp. 210–215

[146] Disponible en: http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html

[147] Disponible en:

https://github.com/mongodb/node-mongodb-native

[148]Disponible en:
https://mongodb.github.io/morphia/

[149]Disponible en:
http://mongodb.github.io/morphia/1.0/javadoc/org/mongodb/morphia/mapping/cache/EntityCache.html

[150] Disponible en:
http://businessintelligence.com/tag/nosql/

[151]Disponible en:
http://subversion.apache.org/

[152] Disponible en:
https://docs.mongodb.org/manual/core/replication-introduction/

[153] Disponible en:
http://docs.couchdb.org/en/latest/intro/consistency.html

ANEXO

- **Características de Hardware utilizado en Escenario:**

    - Procesador: **Core i3**
    - RAM: **4 GB**
    - Capacidad de disco: **1 TeraByte**
    - Sistema Operativo: **Windows 7 – 64 bits**

- **Java JDK 1.7:**
  Disponible en:
  http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

- **Eclipse (IDE de desarrollo) Kepler Service Release 2**
  Build id: 20140224-0627
  Disponible en:
  https://eclipse.org/downloads/

- **DIA 0.97.2(editor de diseño de diagramas):**
  Disponible en:
  http://dia-installer.de/

- **JDBC MySQL Connector Driver v5.1.8:**
  Disponible en:
  http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.8

- **XAMPP v5.6.14(entorno utilizado para el desarrollo sobre BBDD MySQL)**
  Disponible en:
  https://www.apachefriends.org/es/index.html

- **MongoDB Java Driver 3.0.4:**
  Disponible en:
  https://docs.mongodb.org/ecosystem/drivers/java/

- **RoboMongoDB** (herramienta de visualización sobre Base de Datos MongoDB):
  http://robomongo.org/