Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Arquitectura de datos para la Red Social de la Universidad de las Ciencias Informáticas

Autor:

Karelia Reyes Sierra

Tutores:

Ing. Cesar González Hernández

Ing. Damarys Cano López

La Habana, Junio de 2012

"Año 54 de la Revolución"

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas, para que haga el uso que estime pertinente.

Para que así conste firmo la presente declaración a los _____ días del mes de _____ del año ____.

Karelia Reyes Sierra

_____
Firma del Autor

Ing. Cesar González Hernández

_____
Firma del Tutor

Ing. Damarys Cano López.

_____
Firma del Tutor

A mi madre, Jacqueline, porque para ti no hay nadie mejor en el mundo que tu hija mayor.

A mi padre, Bárbaro, porque cuando me hablas siento que no hay ningún problema que no puedas resolver ni nada que yo no pueda hacer.

A mi hermana Katia, porque eres la niña de mi alma, mi primera hija y porque no hago nada sin pensar en ti y en todo el amor que me das.

A mi hermanito Fernandito, por decir siempre que soy tu hermana favorita.

A mi tía Jessica, por ser mi tía, mi madre, mi hermana y mi confidente.

A mi abuela Caridad que en paz descanse, por ser mi ángel de la guarda.

A mis abuelos Evaildo y Zenaida, porque no conozco mayor muestra de amor y cariño que la que me dan ustedes.

A mi hermano Aquiles, porque aunque para todos seamos primos yo te quiero como a un hermano.

A Pavel, porque siempre estuvo, está y estará cuando lo necesito.

A Ana Rosa, Yaimara, Yanisbel, Evelín, Zoilén, Yanet, Anisley, Walter, Carlos, Jose, Arístidez y Yeniel, amigos como ustedes sólo se encuentran a tres metros sobre el cielo.

Resumen

La Universidad de las Ciencias Informáticas es una institución donde se trabaja en la creación de una red social universitaria. Tiene entre sus principales metas permitir que alrededor de 10 mil usuarios de la universidad se conecten constantemente y consuman los servicios que se brindan, así como registrar su actividad al acceder a cada uno de estos con el mayor rendimiento posible sin que colapse el sistema. La actual investigación surgió por la necesidad de desarrollar la arquitectura de datos de la red social universitaria de la Universidad de las Ciencias Informáticas que permita almacenar el gran volumen de información, traducido en datos, que crece paulatinamente y se genera de forma vertiginosa. Se definió como objetivo general, proponer una arquitectura de datos que permita garantizar alto rendimiento en la capa de persistencia de datos en la red social universitaria de la Universidad de las Ciencias Informáticas, donde existe alta concurrencia de usuarios y se almacena gran volumen de información. Para dar cumplimiento al mismo se realizó un análisis de los diferentes modelos de base de datos existentes y las tendencias actuales en el almacenamiento de información en las redes sociales. Se obtuvo como principal resultado el modelo de datos NoSQL como base para el desarrollo de la arquitectura de datos y la herramienta para el almacenamiento de información HBase. La solución propuesta fue validada a través de un conjunto de pruebas de rendimiento que permitieron dar cumplimiento al objetivo planteado para la investigación.

Palabras clave: *red social universitaria, arquitectura de datos, rendimiento*.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

En sus inicios Internet no brindaba la posibilidad de interactuar directamente con los usuarios ni entre ellos ya que se basaba en la simple publicación de contenidos por los llamados *webmasters*[1], a esta web se le denominó Web 1.0. El constante desarrollo provocó la necesidad de mejorar este producto inicial y junto a él los servicios que brindaba, por lo que para compensar las nuevas demandas se hizo necesaria la evolución a la Web 2.0, generándose una serie de webs basadas en la creación de contenidos producidos y compartidos por los propios usuarios de la red.

Las redes sociales aunque puedan parecer un fenómeno reciente surgieron en 1997 (ROS-MARTÍN, 2009) antes del nacimiento de la Web 2.0, no son más que espacios en Internet que permiten interactuar con otras personas y organizaciones. Estas estructuras sociales compuestas por personas agrupadas por uno o más tipos de relaciones se han desarrollado como estrategia para acortar las distancias en materia de comunicación, convirtiéndose en una necesidad del hombre almacenar y administrar toda la información que circula y crece en la red.

La Universidad de las Ciencias Informáticas es una institución en la que la interacción de todo su personal y la divulgación de la información constituyen elementos fundamentales para fomentar su desarrollo. En el Centro de Informatización Universitaria se trabaja en la creación de una red social universitaria, que brindará a los usuarios las funcionalidades de las redes sociales convencionales y otras más de corte académico para su beneficio y el de la institución. Será un espacio en el que se podrán encontrar de forma centralizada los servicios que se ofrecen en ella y los de las redes sociales convencionales. Esta red tiene como objetivos principales ampliar las comunicaciones entre los usuarios, fomentar la investigación y la colaboración científica y aumentar el trabajo comunitario con una tendencia creciente al uso y desarrollo de servicios.

Constituye un reto garantizar que alrededor de 10 mil usuarios de la universidad se conecten constantemente y consuman los servicios que serán integrados a la red social como: el correo electrónico, mensajería instantánea, gestión y notificación de eventos, gestión de *software*, gestión de archivos media,

[1] Es el responsable de diseñar, desarrollar y gestionar todos los aspectos de contenido de un sitio web, la conectividad y los recursos humanos. ROWLAND, E. *The Decline of the "Webmaster".* [Consultado el: 07 de febrero de 2012]. Disponible en: http://www.clickfire.com/the-decline-of-the-webmaster/.

periódico digital; así como registrar su actividad al acceder a cada uno de los servicios con el mayor rendimiento posible sin que colapse el sistema.

El volumen de información crecerá paulatinamente y la cantidad de información traducida en datos que se generará de forma vertiginosa demandará un sistema de base de datos potente, que sea capaz de almacenar todos los datos.

Por la problemática mencionada anteriormente se enuncia el siguiente **problema a resolver**: ¿Cómo garantizar alto rendimiento en la capa de persistencia de datos en la red social universitaria de la Universidad de las Ciencias Informáticas, donde existe alta concurrencia de usuarios y se almacena gran volumen de información?

Queda definido como **objeto de estudio:** la arquitectura de datos.

Se determina como **campo de acción:** la arquitectura de datos para redes sociales.

Con el fin de dar cumplimiento a esta investigación se plantea como **objetivo general**: proponer una arquitectura de datos que permita garantizar alto rendimiento en la capa de persistencia de datos en la red social universitaria de la Universidad de las Ciencias Informáticas, donde existe alta concurrencia de usuarios y se almacena gran volumen de información.

El objetivo general planteado anteriormente se desglosa en los siguientes **objetivos específicos**:

- Analizar los diferentes modelos de base de datos existentes y las tendencias actuales en el almacenamiento de datos en las redes sociales.
- Proponer la arquitectura de datos para el almacenamiento de información de la red social universitaria de la Universidad de las Ciencias Informáticas y las pruebas de sistema para la solución.
- Validar la solución teniendo en cuenta las pruebas propuestas.

Para dar cumplimiento al objetivo planteado se utilizarán los siguientes **métodos científicos de investigación**:

Métodos teóricos.

- **Análisis.** Se pone de manifiesto al dividir el objeto de estudio que es la arquitectura de datos en componentes o partes más pequeñas que son los modelos de datos, para facilitar el estudio de los aspectos específicos del mismo.
- **Síntesis.** A través de este método se establece la unión de los componentes o partes previamente analizadas, brindando la posibilidad de establecer las características generales y las relaciones

esenciales entre ellas, sintetizándolas para poder lograr una serie de conclusiones correctamente definidas.

- **Histórico.** Permite analizar la trayectoria completa de los modelos de bases de datos en las redes sociales, las tendencias de su uso a lo largo de la historia, revelando las etapas fundamentales de su desenvolvimiento.
- **Lógico.** Pone de manifiesto, después de analizar anteriormente la trayectoria histórica de los modelos de bases de datos, la lógica interna de su desarrollo, permitiendo la unión entre el estudio de la estructura del objeto de estudio y su concepción histórica.

Métodos empíricos.

- **Observación.** A través de este se observan las características del almacenamiento de datos en las redes sociales así como las herramientas que utilizan para realizar este proceso.

Resultados esperados

La presente investigación se realiza por la necesidad de estudiar las características del almacenamiento de datos en las redes sociales con el fin de proponer una solución que defina, teniendo en cuenta las particularidades de la red social universitaria de la Universidad de las Ciencias Informáticas, una arquitectura de datos idónea. Permitirá sostener la red social y garantizar sus funciones, beneficiando a todos sus usuarios. Por la recopilación de conocimiento acerca de los modelos de base de datos existentes y las herramientas más utilizadas para el almacenamiento de datos en redes sociales, constituirá un material de estudio y se podrá utilizar como documento de apoyo a los cursos de Base de datos. Se pretende obtener los manuales de Instalación y Configuración de las herramientas a utilizar.

Estructura del trabajo de diploma

Capítulo 1. Arquitectura de datos y redes sociales. En el presente capítulo se plantean las bases teóricas que permiten dar solución al problema enunciado. Contiene el análisis de las tendencias que existen en el almacenamiento de información en algunas de las redes sociales más populares del mundo así como las herramientas que utilizan.

Capítulo 2. Descripción y validación de la herramienta HBase. En el presente capítulo se describen las particularidades de la herramienta HBase, sus características, arquitectura, instalación y seguridad. Se lleva a cabo el diseño y ejecución de las pruebas, se analizan los resultados obtenidos y se realiza la validación de la herramienta.

Capítulo 1: Arquitectura de datos y redes sociales

Almacenar la gran cantidad de información que circulará en la red social constituye una tarea de gran envergadura. En el presente capítulo se plantean las bases teóricas que permiten dar solución al problema enunciado. Contiene el análisis de las tendencias que existen en el almacenamiento de información en algunas de las redes sociales más populares del mundo así como las herramientas que utilizan.

1.1 Principales conceptos asociados a la investigación

En el presente epígrafe se expresan algunos de los principales conceptos asociados a la investigación que deben ser tratados para un mejor entendimiento de la misma.

Dato

Según el diccionario de la Real Academia Española un dato es un "*Antecedente necesario para llegar al conocimiento exacto de algo o para deducir las consecuencias legítimas de un hecho.*" En el ámbito informático se define como "Información dispuesta de manera adecuada para su tratamiento por un ordenador." (ESPAÑOLA, 2011b). Sin embargo, un dato por sí solo no contiene información, sino que se almacena en las tablas de la base de datos para luego extraerse junto con otros y formar una información de la cual se obtiene el conocimiento. Un dato puede significar un número, una letra, un signo ortográfico o cualquier símbolo que represente cantidad, una medida, una palabra o una descripción. Algunos ejemplos son: cuando se dice "cinco", "a", ":", "km", "Matanzas"; estos no representan por sí solos ningún tipo de información, sino que son simplemente datos que se almacenan como atributos de una entidad. Los datos al extraerse y conjugarse pueden brindar una información, por ejemplo: "Matanzas: a cinco km" y luego al interpretarse es que se obtiene un conocimiento, se puede decir que la ciudad de Matanzas se encuentra a cinco km. Se asume que un dato no tiene sentido por sí mismo.

Base de datos (BD)

Una BD es una estructura en la que se almacenan datos de forma sistemática, que están relacionados el uno con el otro y a la vez pertenecen al mismo contexto para que puedan utilizarse en el futuro. Según el diccionario de la Real Academia Española es un "*Conjunto de datos organizado de tal modo que permita obtener con rapidez diversos tipos de información*" (ESPAÑOLA, 2011a). Las bases de datos pueden ser

estáticas (cuando los datos almacenados no varían pese al paso del tiempo) o dinámicas (los datos se modifican con el tiempo, requieren de actualizaciones periódicas).

Sistemas de gestión de bases de datos (SGBD)

Los SGBD *"son software que dirigen y controlan todas las gestiones que realizan las bases de datos"* (RUIZ, 2008). Sin embargo estos son más que software, constituyen un conjunto de programas a través de los cuales se puede definir, manipular y utilizar la información que contienen las bases de datos, realizar todas las tareas de administración necesarias para mantenerlas operativas, mantener su integridad, confidencialidad y seguridad. Se pueden utilizar como servidor de datos para programas complejos realizados en cualquier lenguaje de programación, además se le considera como la interfaz entre el usuario y la BD.

Arquitectura de datos

La arquitectura de datos es una guía general a través de la cual se puede definir la forma de almacenar y mantener los datos en un SGBD determinado. Esta indica la estructura, funcionamiento e interacción entre las partes que componen dicho sistema, guardando especial relación con los modelos de datos (ABREU *et al.*).

Abraham Silberschatz[2] en el libro "*Fundamentos de bases de datos"* plantea que la arquitectura de un sistema de bases de datos está influenciada en gran medida por el sistema informático subyacente en el que se ejecuta, en particular por aspectos de la arquitectura de la computadora como la conexión en red, el paralelismo y la distribución (SILBERSCHATZ *et al.*, 2002):

- La conexión en red de varias computadoras permite que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en los sistemas clientes. Esta división de trabajo ha conducido al desarrollo de sistemas de bases de datos cliente-servidor.
- El procesamiento paralelo dentro de una computadora permite acelerar la actividad del sistema de base de datos, proporcionando a las transacciones unas respuestas más rápidas así como la capacidad de ejecutar más transacciones por segundo. Las consultas pueden procesarse de manera que se explote el paralelismo ofrecido por el sistema informático subyacente. La necesidad

[2] Científico informático estadounidense. Obtuvo su doctorado de la Universidad Estatal de Nueva York en Stony Brook. Profesor de Ciencias de la Computación en la Universidad de Yale. Antes de llegar a la Universidad de Yale en 2003 era el vice presidente del Centro de Investigación de Ciencias de la Información en los Laboratorios Bell. Anteriormente, había ocupado una cátedra dotada de la Universidad de Texas en Austin, donde enseñó hasta 1993 BIOGRAFÍADE. *Abraham Silberschatz* [Consultado el: 10 de junio de 2012]. Disponible en: http://www.biografias10.com/s/Abraham-Silberschatz/1/.

del procesamiento paralelo de consultas ha conducido al desarrollo de los sistemas de bases de datos paralelos.

- La distribución de datos a través de las distintas sedes o departamentos de una organización permite que estos datos residan donde han sido generados o donde son más necesarios, para continuar siendo accesibles desde otros lugares o departamentos. El hecho de guardar varias copias de la base de datos en diferentes sitios permite que puedan continuar las operaciones sobre la base de datos aunque algún sitio se vea afectado por algún desastre natural como una inundación, un incendio o un terremoto. Se han desarrollado los sistemas distribuidos de bases de datos para manejar datos distribuidos geográfica o administrativamente a lo largo de múltiples sistemas de bases de datos.

Modelos de datos (MD)

Los modelos de datos establecen las reglas y conceptos que dan la posibilidad de representar las relaciones entre varios datos así como las operaciones que se realizan sobre los mismos, cumpliendo en todo momento un conjunto de restricciones. Poseen como objetivo principal facilitar la representación de los datos en el sistema de información dado. A través de los mismos se pueden estructurar los datos de forma que se capte la semántica de estos. Un MD será más expresivo en la medida en que sea capaz de representar dentro de él, la mayor cantidad de información posible sobre los datos.

Abraham Silberschats en su libro "Fundamentos de bases de datos" plantea que *"Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos y conceptos matemáticos"(SILBERSCHATZ et al., 2002).*

Red social

Las redes sociales se han convertido en una forma de comunicación en línea que permite a los usuarios interactuar con las otras personas que forman parte de las redes. Las mismas *"son formas de interacción social, definidas como un intercambio dinámico entre personas, grupos e instituciones"*(PALACIOS, 2010). Una red social es un sistema abierto y en construcción permanente el cual permite que se identifiquen conjuntos de personas que poseen las mismas necesidades.

Sitios de redes sociales (por sus siglas en inglés, SNS), tales como *Friendster, Cyworld* y *MySpace* permiten a los individuos presentarse a ellos mismos, articular sus redes sociales y establecer o mantener las conexiones con los demás. Estos sitios se pueden orientar hacia el trabajo relacionado con los contextos (por ejemplo, *LinkedIn.com*), el inicio de una relación romántica (la meta original de

Friendster.com), la conexión de las personas con intereses comunes como la música o la política (por ejemplo, *MySpace.com*), o el estudiante universitario de la población (la encarnación original de *Facebook.com*). Los participantes pueden usar los sitios para interactuar con las personas que ya conoce en línea o para conocer gente nueva (ELLISON *et al.*, 2007).

1.2 Modelos de datos: características, ventajas y desventajas de su uso

Existen varias formas de clasificar los modelos de datos, sin embargo, se decide utilizar la representada en la Figura **1,** porque permite analizar de cada modelo seleccionado para la investigación, las características que favorecen los criterios establecidos. Los modelos de datos utilizados durante la investigación se agrupan en dos criterios, funcionalidad y rendimiento.

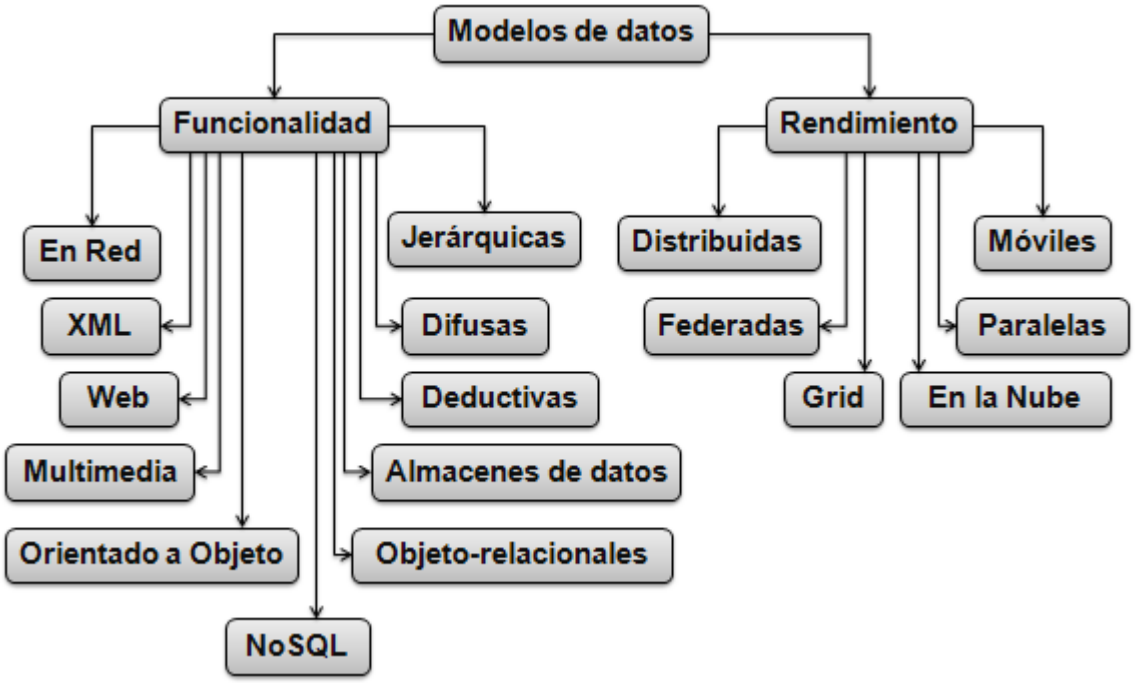

Figura 1. **Modelos de datos.**

Funcionalidad: todos los modelos pueden ser caracterizados de acuerdo con su funcionalidad, debido a que cada uno posee una forma particular de representar las relaciones y operaciones que se llevan a cabo entre y sobre sus datos, sin embargo, los modelos que aparecen representados en este criterio son los elegidos para analizar las características específicas en cuanto a su funcionamiento.

Jerárquicas

Se deriva de los sistemas de gestión de información de los años 1950 y 1960, fue adoptado por muchos bancos y compañías de seguros que todavía los utilizan. Los sistemas de base de datos jerárquicos se pueden encontrar en algunos departamentos de instituciones públicas y hospitales para gestionar el inventario y la contabilidad. Se basa en almacenar los datos en una serie de registros, los cuales tienen campos asociados a ellos. Para crear enlaces entre los tipos de registros, utiliza las relaciones padre-hijo, correspondencias 1: N entre tipos de registro. Esto se realiza mediante el uso de árboles. A diferencia de otros modelos, como el modelo En Red, el modelo jerárquico representa precisamente eso, todas las relaciones están jerarquizadas en un árbol (ver Figura **2**), por lo que no es capaz de establecer enlaces entre hijos o entre capas, si no es padre-hijo. La ventaja del modelo jerárquico es su gran estructuración, que anteriormente se veía como una gran ventaja para mejorar el rendimiento de las transacciones (inserción, modificación y borrado de registros), así como para simplificar la interfaz para los usuarios.

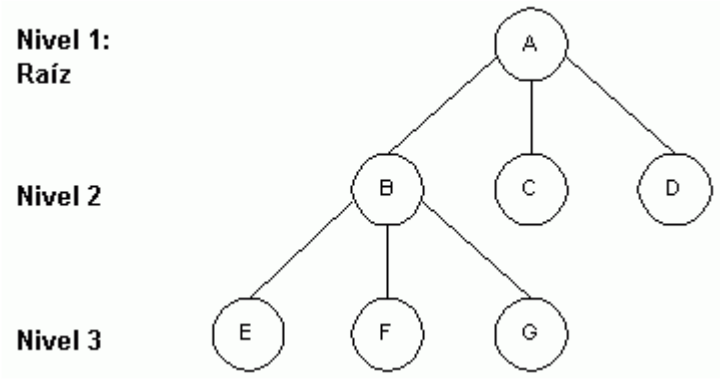

Figura 2. **Modelo de base de dato jerárquico (ORALLO, 2002).**

En red

El modelo En Red se estandarizó a finales de 1960 mediante un informe de CODASYL (*COnference on Data SYstems and Languages*) *Data Base Task Group* (DBTG). Por eso, en ocasiones se le conoce como el modelo DBTG o el modelo CODASYL. El primer informe de CODASYL ya incluía la sintaxis y semántica de un lenguaje de definición de datos (DDL, *Data Definition Language*) y de un lenguaje de manipulación de datos (DML, *Data Manipulation Language*). Siguiendo muchos comentarios, revisiones de expertos y usuarios se realizó un nuevo informe, en el que ya se incluía la posibilidad de definir vistas para los distintos grupos de usuarios. El término base de datos en red no se refería (al contrario de lo que se

entendería hoy en día) a que la base de datos estuviera almacenada en una red de ordenadores, sino por la manera en la que los datos se enlazaban con otros datos (ver Figura **3**). Se llama, por tanto, modelo en red porque representa los datos que contiene en la forma de una red de registros y conjuntos (en realidad listas circulares llamadas *sets*) que se relacionan entre sí, formando una red de enlaces. Para hacer esto utiliza registros, tipos de registro y tipos de conjunto. El modelo en red no es muy utilizado hoy en día y si subsiste es como consecuencia del mantenimiento de un sistema todavía no reconvertido o no portado a modelos y SGBD más modernos. Aunque este modelo permite más flexibilidad que el modelo jerárquico, y en algunos casos se adapta muy bien a algunos tipos de transacciones, se considera superado por otros modelos, como el relacional, o asumido en parte por modelos más modernos, como el objetual.

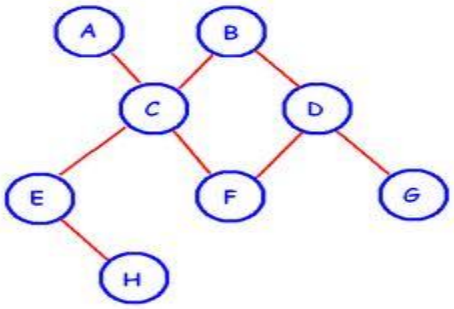

Figura 3. **Modelo de base de dato en red (ORALLO, 2002).**

Los modelos Jerárquico y En Red, con el paso del tiempo se pueden considerar como modelos puente hacia el modelo relacional, ya que se incorporan en los primeros sistemas de gestión de bases de datos que introducen un mayor nivel de independencia respecto a la estructura interna, pero siguen teniendo una estructura de bajo nivel y de compleja manipulación. Por tales motivos no se tendrán en cuenta para tomar una decisión respecto a la solución.

Multimedia

Las bases de datos multimedia almacenan una gran variedad de tipos de datos como texto, imágenes, audio y vídeo. Hasta hace unos años estas bases de datos eran difíciles de implementar, debido al tamaño de los objetos y la complejidad de los datos. El añadir metadatos a los ficheros multimedia resolvía parcialmente el problema. Esta cabecera incluye datos del formato, el creador, el contenido, etc. El problema es que esta información no suele estar indexada convenientemente y no se puede utilizar para realizar consultas.

Hay dos métodos principales para incluir metadatos en una base de datos multimedia: mediante análisis automático o mediante análisis manual. Aunque el análisis manual es más efectivo, porque permite anotar

aquellas características del objeto multimedia que son importantes, requiere mucho tiempo y por tanto es muy costoso. Además es difícil conseguir homogeneidad en los criterios que se utilizan para agregar estos metadatos. El análisis automático es mucho más rápido, pero las técnicas todavía son limitadas en algunos casos especialmente en imagen y sonido). Su mayor ventaja es que proporciona una descripción consistente de los datos y no se ve afectada por estilos individuales (SUBRAHMANIAN y SCHULZRINNE, 1998).

Si hace pocos años, las bases de datos multimedia no eran asequibles para los usuarios de ordenadores personales, el abaratamiento de discos con gran capacidad de almacenamiento, hace que cualquier ordenador personal pueda contener una biblioteca de imágenes, piezas musicales y de películas. Otro problema es el uso de estas bases de datos para distribución por Internet (especialmente VoD, *Video on Demand*), para el cual el ancho de banda todavía tiene que crecer enormemente (ORALLO, 2002).

Orientado a Objeto

Las bases de datos Orientado a objeto son una colección persistente y compatible de objetos definida por un modelo de datos orientado a objetos, en las mismas su modelo de datos almacena y recupera objetos en los que se almacena estado y comportamiento. Su origen se debe a que en los modelos clásicos de datos existen problemas para representar cierta información, puesto que aunque permiten representar gran cantidad de datos, las operaciones que se pueden realizar con ellos son bastante simples. Permiten aprovechar las ventajas del paradigma orientado a objetos en el campo de las bases de datos y evitan transformaciones entre modelos de datos (usar el mismo modelo de objetos) (MUÑOZ *et al.*, 2008).

En bases de datos orientados a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica de forma separada y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones (ORALLO, 2002).

A continuación se describen varias características de las base de datos orientado a objetos (MUÑOZ *et al.*, 2008):

- Objetos: cada entidad del mundo real se modela como un objeto.

- Identificador de objetos (OID): único para cada objeto. Generalmente este identificador no es accesible ni modificable para el usuario. Los OID son independientes del contenido, es decir, si un objeto cambia los valores de atributos, sigue siendo el mismo objeto con el mismo OID. Si dos objetos tienen el mismo estado pero diferentes OID, son equivalentes pero tienen identidades diferentes.
- Encapsulamiento: cada objeto contiene y define procedimientos (métodos) y la interfaz mediante la cual se puede acceder a él y otros objetos pueden manipularlo. La mayoría de los Sistemas Gestores de Base de Datos Orientado a Objeto (SGBDOO) permiten el acceso directo a los atributos incluyendo operaciones definidas por el propio SGBDOO, las cuales leen y modifican los atributos para evitar que el usuario tenga que implementar una cantidad considerable de métodos, cuyo único propósito sea el de leer y escribir los atributos de un objeto. Generalmente, los SGBDOO permiten al usuario especificar qué atributos y métodos son visibles en la interfaz del objeto y pueden invocarse desde afuera.

La utilización del modelo de datos multimedia se recomienda para la implementación de bases de datos para el almacenamiento de archivos media. Para el desarrollo de la presente investigación no cumple objetivo tener en cuenta este modelo ya que a través del mismo se representa un área de información muy específica. De igual forma el modelo de datos orientado a objeto, aunque permite representar gran cantidad de datos, las operaciones que se pueden realizar en el mismo son bastante simples y no satisfacen las necesidades de la investigación.

Objeto-relacionales

Éste es uno de los modelos más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970, no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados *tuplas*[3]. Pese a que ésta es la teoría de las bases de datos relacionales, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es viendo cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las *tuplas* y campos (las columnas de una tabla). En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de

[3] Conjunto de elementos de distintos tipos que se guardan de forma consecutiva en memoria. Sirven para agrupar, como si fueran un único valor, varios valores que, por su naturaleza, deben ir juntos. BLADE, L. *Tupla* [Consultado el: 10 de junio de 2012]. Disponible en: http://es.diveintopython.net/odbchelper_tuple.html.

otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información (MUÑOZ *et al.*, 2008).

Las características más importantes de este modelo se muestran a continuación (ORALLO, 2002):

- Se pueden crear nuevos tipos de datos que permiten gestionar aplicaciones más complejas con una gran riqueza de dominios.
- Se soportan tipos complejos como: registros, conjuntos, referencias, listas, pilas, colas y arreglos.
- Se pueden crear funciones que posean un código en algún lenguaje de programación como por ejemplo: *SQL*[4], Java, C, etc.
- Existe una mayor capacidad expresiva para los conceptos y asociaciones. Se pueden crear operadores asignándole un nombre y existencia de nuevas consultas con mayor capacidad consultiva.
- Soporta el encadenamiento dinámico y herencia en los tipos *tupla* o registro.
- Se pueden compartir varias bibliotecas de clases ya existentes, esto es lo que se conoce como reusabilidad.
- Posibilidad de incluir el chequeo de las reglas de integridad referencial a través de los *triggers*[5].
- Soporte adicional para seguridad y activación de la versión cliente-servidor.

El inconveniente que tienen las bases de datos objeto-relacional (BDOR) es que aumentan la complejidad del sistema y por tanto ocasiona un aumento del coste asociado.

Almacenes de datos

"Un almacén de datos es una colección de datos orientada al dominio, integrada, variante en el tiempo y no volátil, diseñada para dar apoyo al proceso de toma de decisiones en una organización" (KIMBALL y ROSS, 2011).

[4] Por sus siglas en inglés *Structured Query Language*, lenguaje de consulta estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores SQL.ORG. *SQL* [Consultado el: 10 de junio de 2012]. Disponible en: http://www.sql.org/.

[5] Es un bloque que se almacena en la BD y se ejecuta de forma implícita cuando ocurre el suceso que lo activa, no acepta argumentos. El suceso que activa el disparo puede ser: una operación DML sobre una tabla o vista de la BD, un suceso del sistema: inicio, desconexión, entre otros. Existen tres tipos fundamentales de disparadores: disparadores DML, de sustitución y de Sistema.

Los almacenes de datos integran información procedente de múltiples fuentes de datos independientes en una única base de datos, funcionando como un repositorio de información histórica que puede ser consultado directamente para detectar tendencias y anomalías dentro de las actividades del negocio, conocer el estado actual de áreas de interés de la organización y tomar decisiones de futuro.

Usualmente el almacén de datos se encuentra separado de otros sistemas y aplicaciones de la organización en una base de datos propia. Una razón obvia para ello reside en el hecho de que en las organizaciones es frecuente que existan diferentes repositorios de datos soportados por tecnologías diferentes, lo que dificulta el acceso integrado a la información. Otras razones son (LINOFF y BERRY, 2011):

- Los objetivos de uso: los sistemas operacionales están orientados a los procesos (procesamiento de transacciones) mientras que los sistemas de almacenes de datos están orientados al análisis de los datos.
- Las características de mantenimiento: los sistemas operacionales cambian constantemente, mientras que los sistemas de almacenes de datos no son volátiles, sólo crecen incrementalmente.

El uso de los sistemas de almacenes de datos ha puesto al relieve las ventajas e inconvenientes que el desarrollo de este tipo de sistemas reporta a las organizaciones.

Ventajas:

- Integrar datos históricos sobre la actividad de la organización o negocio en un único repositorio.
- Analizar los datos del negocio desde la perspectiva de su evolución en el tiempo.
- Prever tendencias de evolución del negocio.
- Identificar nuevas oportunidades de negocio y tomar decisiones estratégicas.
- Reducir los costes materiales y humanos en la toma de decisiones.

Inconvenientes:

- Riesgo de fracaso en la construcción del sistema, al subestimar los costes de captura y preparación de los datos.
- Riesgo de fracaso en la construcción del sistema por cambios continuos en los requisitos de los usuarios.
- Problemas con la privacidad de los datos.

En resumen, aunque los almacenes de datos actualmente constituyen un área de gran interés, se puede considerar como una rama establecida de la tecnología de las bases de datos, especialmente vigente en

grandes organizaciones e industrias de la distribución, y suele funcionar en relación con los sistemas de toma de decisión (DSS, *Decision Support System*) y los Sistemas de Información Ejecutiva (EIS, *Executive Information System*). Es relevante constatar que el uso de los almacenes de datos es el original de los sistemas de información: recopilar datos para analizarlos y tomar decisiones a partir de ese análisis (ORALLO, 2002).

En la presente investigación se destaca este modelo de datos por las facilidades que brinda, proporcionando un sistema de almacenamiento que permite el análisis de grandes volúmenes de datos almacenados sin correr riesgo de pérdida.

XML

El Lenguaje de Marcas Extendido (XML, *eXtended Mark-up Language*) es un lenguaje de marcas inspirado también en el Lenguaje de Marcas Generalizado Estandarizado (SGML, *Standardized Generalized Markup Language*) con el objetivo de permitir el intercambio de información de diversos tipos. El XML permite tratar datos semi-estructurados de la Web, organizar colecciones de datos de distintas fuentes y formatos e intercambiar datos entre diferentes sitios/organizaciones.

En lo que concierne a las bases de datos, el XML permite integrar sistemas de información hasta el momento separados:

- Sistemas de información basados en documentos o ficheros: tienen estructura irregular, anidados profundamente, utilizan tipos de datos relativamente simples y dan gran importancia al orden.
- Sistemas de información estructurados (por ejemplo las bases de datos relacionales): tienen una estructura muy regular, son relativamente planos, utilizan tipos de datos relativamente complejos y dan poca importancia al orden.

El hecho de que XML permita expresar información de características y estructuras muy diversas, no quiere decir que sustituya a las bases de datos tradicionales. Cuando por XML se entiende un documento, su DTD (en español, Definición de tipo de documento) asociada, la interpretación del mismo respecto a una semántica y todas las tecnologías que lo rodean, se puede llegar a comparar la tecnología XML con las bases de datos.

A continuación se describen las similitudes que posee respecto a las bases de datos:

- La tecnología XML usa uno o más documentos (ficheros) para almacenar la información.
- Define esquemas sobre la información (DTD y otros lenguajes de esquema XML).

- Tiene lenguajes de consulta específicos para recuperar la información requerida (*XQL*[6], *XML-QL*[7], etc.).
- Dispone de interfaces de programación (*SAX*[8], *DOM*[9]).

Las principales diferencias que existen entre el XML y las bases de datos son:

- La tecnología XML carece de algunas características comunes que existen en las bases de datos, almacenamiento y actualización eficientes, índices, seguridad, transacciones, integridad de datos, acceso concurrente, disparadores, etc.

Pese a ser un lenguaje de marcas, XML permite representar la mayoría de modelos de datos existentes: en red, jerárquico, relacional, objetual, etc. Es decir, se puede representar la información contenida en una base de datos relacional en uno o más documentos XML (ORALLO, 2002).

A pesar de lo anteriormente expuesto, XML no representa una solución viable para resolver el problema de la investigación planteado ya que carece de características como seguridad, transacciones, integridad de datos, acceso concurrente, entre otras, todas estas de suma importancia para la arquitectura de datos de la red social en construcción.

Web

Con el advenimiento de la Web, la gestión de datos se ha ramificado para tratar con una variedad de información disponible en la *World Wide Web*[10]. La necesidad creciente de colocar información en la misma, haciendo que esté disponible desde cualquier lugar del mundo, ha dado lugar a la proliferación de sitios web con datos, en muchas ocasiones, duplicados y mal organizados. Este tipo de organización casual, así como las redundancias de información, dificulta el mantenimiento de los sitios, provocando incoherencias, datos difícilmente accesibles, etc. La comunidad de bases de datos sabe que un modo de evitar este tipo de problemas es, precisamente, aprovechar las facilidades que ellas brindan. Pero el proceso de creación de una base de datos web no es exactamente igual al proceso de desarrollo de una

[6] Lenguaje de consulta diseñado específicamente para documentos XML. ROBIE, J. *XQL (XML Query Language)* [Consultado el: 10 de junio de 2012]. Disponible en: http://metalab.unc.edu/xql/xql-proposal.xml

[7] Es un acrónimo de Lenguaje de Marcado Extensible-Lenguaje de consulta.

[8] Simple API for XML (SAX) es una interfaz simple para aplicaciones XML. Fácil e intuitiva, se usa es pecialmente en situaciones en las que los archivos XML ya están en una forma que es estructuralmente similar a la que se desea obtener. LAPUENTE, M. J. L. Hipertexto: el nuevo concepto de documento en la cultura de la imagen. *http://www. hipertexto. info/documentos/web_tecnolog. htm. Consultado em*, 2007, vol. 19, nº p. 06-07. [Consultado el: 10 de junio de 2012]. Disponible en: http://www.hipertexto.info/documentos/dom.htm.

[9] Modelo de Objetos del Documento o Document Object Model (DOM) es un modelo de objetos estandarizado para documentos HTML y XML. Es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento. Ibid.

[10] Literalmente es la Red de Alcance Mundial.

tradicional. Por ello es necesario también adaptar los métodos de desarrollo de bases de datos clásicas (tanto el proceso, como las técnicas, notaciones y modelos) a la Web.

Una base de datos web es aquella cuyos datos pueden ser accesibles a través de la red. Su mantenimiento puede realizarse o no por medio de páginas web. Su creación es similar a una de carácter tradicional, pero no igual. Una vez que se ha creado una base de datos web es posible y extremadamente útil, aprovechar esa experiencia para tratar de sistematizar ese proceso.

Las empresas y organizaciones han visto la importancia de que ciertos contenidos de sus bases de datos estuvieran disponibles a través de la red. En un primer momento, esto permitía publicar información hipermedia que podría ser recuperada desde cualquier lugar. Realizar esto era aparentemente algo tan simple como enlazar varias páginas *HTML*[11] con el contenido deseado. Sin embargo ese sistema de publicación ha dado lugar a sitios donde la información se organiza de una manera casual y habitualmente caótica, dando lugar a los ya conocidos problemas de mantenimiento, tales como: inconsistencias, información obsoleta, problemas al relacionar datos, etc. (MARCOS y VELA, 2002).

Actualmente existen múltiples bases de datos web de dominios muy específicos de información, son espacios reducidos en los que se lleva a cabo, de forma sistemática, todo el proceso de mantenimiento, permitiendo encontrar la información detallada y precisa pero solo de algunas ramas de la ciencia. De forma general la información en la Web se encuentra desorganizada.

Deductivas

Un sistema de bases de datos deductivas, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática (SOARES, 2005).

Las bases de datos deductivas utilizan relaciones y reglas escritas específicamente pensadas para ellas, por lo que la apariencia es totalmente diferente a las bases de datos ordinarias. Gracias a ello, se puede preguntar sobre información implícita que será extraída al aplicar las reglas a los datos o hechos. Las reglas, basadas sobre todo en transitividad y recursividad, pueden llegar a ser notablemente complejas, para lo cual se hace necesario que el usuario tenga una gran capacidad de abstracción. Básicamente, su

[11] Siglas de HyperText Markup Language, lenguaje de marcado de hipertexto. RAGGETT, D.; LE HORS, A., *et al.* HTML 4.01 Specification. *W3C recommendation*, 1999, vol. 24, nº [Consultado el: 10 de junio de 2012].

aplicación se reduce a entornos experimentales, aunque su potencial comercial es grande (MÉXICO y LATINA, 2011).

Difusas

Un sistema basado en reglas difusas es un sistema basado en reglas donde la lógica difusa es utilizada como una herramienta para representar diferentes formas de conocimiento acerca del problema a resolver, así como para modelar las interacciones y relaciones que existen entre sus variables. Debido a estas propiedades, los sistemas basados en reglas difusas han sido aplicados de forma exitosa en varios dominios en los que la información imprecisa emerge en diferentes formas. Permiten recuperar datos con tan solo una vaga descripción de lo que se desea obtener. Esto es un importante avance al acercar el lenguaje natural a la organización formal de los datos. Puede implementarse, sin dificultades, en SGBD ordinarios y el lenguaje de acceso a esta es una extensión de SQL. Sin embargo, requiere una gran cantidad de parámetros, lo que hace el manejo muy incómodo, lo cual constituye la principal causa que propicia su impopularidad (MUELA *et al.*, 2009).

NoSQL

NoSQL ("no sólo SQL") es una amplia clase de sistemas de gestión de base de datos que difieren del modelo clásico del Sistema de Bases de Datos Relacionales (sus siglas en inglés son RDBMS) de manera significativa. Estos almacenes de datos pueden no requerir esquemas fijos de tabla, por lo general evitan las operaciones de combinación y la escala horizontal. El término *NoSQL*, que fue acuñado en 1998, se refería a una base de datos relacional de código abierto que no usaba un lenguaje de consultas SQL (*Structured Query Language*) (CALDERÓN, 2011).

Cuando se habla de *NoSQL* únicamente no se refiere a un tipo de base de datos, sino a diferentes soluciones dadas para almacenar datos cuando las bases de datos relacionales generan problemas. Las mismas son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación, no imponen una estructura de datos en forma de tablas y relaciones entre ellas, en ese sentido son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor (similar a tablas *Hash*[12]), Mapeo de Columnas, Documentos o Grafos.

[12] Estructura de datos que asocia *llaves* o *claves* con *valores*. La operación principal que soporta de manera eficiente es la *búsqueda*: permite el acceso a los elementos almacenados a partir de una clave generada. BESEMBEL, I. y ROBERTS, S. Comparación entre métodos de acceso espaciales y de puntos multidimensionales. *V Jornadas Científico-Técnicas de la Facultad de Ingeniería, ULA. Mérida (Venezuela)*, 1998, nº [Consultado el: 10 de junio de 2012].

Además de la carencia de un esquema predeterminado, la principal característica de las bases de datos *NoSQL* es que están pensadas para manipular enormes cantidades de información de manera muy rápida. Para ello suelen almacenar toda la información que pueden en memoria (utilizando el disco como una mera herramienta de persistencia), y están preparadas para escalar horizontalmente sin perder rendimiento. Suelen funcionar bastante bien en hardware de bajo coste y permiten el escalado horizontal añadiendo nuevas máquinas cuando el sistema se encuentra en funcionamiento (sin necesidad de reinicio del sistema).

Las bases de datos *NoSQL* parten de la base en la que las "tablas" no existen como tal, sino que la información se almacena de forma distinta, generalmente como clave-valor, como una tabla en la que las columnas son dinámicas. Las mismas pueden cambiar sin perder la agrupación de la información, por lo que pueden tener 'personas' con más atributos que otras y pueden cambiar la estructura de la información dinámicamente sin tener que re-diseñar todo de nuevo.

Características:

- Consistencia eventual. No se implementan mecanismos rígidos de consistencia como los presentes en las bases de datos relacionales, donde la confirmación de un cambio implica una comunicación del mismo a todos los nodos que lo repliquen. Esta flexibilidad hace que la consistencia se dé, eventualmente, cuando no se hayan modificado los datos durante un período de tiempo. Esto se conoce también como BASE (*Basically Available Soft-state Eventual Consistency*, o consistencia eventual flexible básicamente disponible).
- Estructura distribuida. Generalmente se distribuyen los datos mediante mecanismos de tablas de *hash* distribuidas.
- Escalabilidad horizontal. Consiste en la posibilidad de aumentar el rendimiento del sistema simplemente añadiendo más nodos, sin necesidad en muchos casos de realizar ninguna otra operación más que indicar al sistema cuáles son los nodos disponibles. Muchos sistemas *NoSQL* permiten utilizar consultas del tipo *MapReduce*[13], las cuales pueden ejecutarse en todos los nodos

[13] Es un marco de trabajo introducido por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras. Se utiliza como modelo de programación para el procesamiento de datos. Se han escrito implementaciones de *MapReduce* en C++, Java, Python y otros lenguajes. DEAN, J. y GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 2008, vol. 51, nº 1, p. 107-113. [Consultado el: 10 de junio de 2012]. ISSN 0001-0782.

a la vez (cada uno operando sobre una porción de los datos) y reunir luego los resultados antes de devolverlos.

- Tolerancia a fallos y redundancia.
- No generan cuellos de botella. El problema de fondo de los sistemas SQL, es que deben de transcribir cada sentencia para poder ser ejecutada y cada sentencia compleja requiere de un nivel de ejecución más concreto para poderse llevar a cabo, por lo que constituye un punto de entrada común, único y conflictivo en base a rendimiento.
- Alta velocidad de respuesta a peticiones.
- Solo lo estrictamente necesario. Son sistemas simples que no tienen un sistema de consulta complejo ni con capacidad declarativa para en una sola línea realizar una cantidad interna de operaciones gigantescas.
- Estructura dinámica. La primera característica es que los datos no tienen una definición de atributos fija, es decir, cada registro puede contener una información con diferentes formas cada vez, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras de datos complejas en un sólo documento, como por ejemplo almacenar la información sobre una publicación de un blog (título, cuerpo de texto, autor, etc.) junto a los comentarios y etiquetas vertidos sobre el mismo, todo en un único registro. Hacerlo así aumenta la claridad (al tener todos los datos relacionados en un mismo bloque de información) y el rendimiento (no hay que hacer una intersección de conjuntos o JOIN para obtener los datos relacionados, pues éstos se encuentran directamente en el mismo documento).

Ventajas:

- Es de código abierto. Los productos de código abierto proporcionan a los desarrolladores grandes beneficios, sobre todo por su estado sin costo alguno. El software de código abierto tiende a ser más confiable, seguro y rápido de implementar que las alternativas propietarias.
- Escalamiento sencillo. *NoSQL* sustituye al antiguo "escalar" el mantra de los gestores de las bases de datos con una nueva manera en lugar de añadir más servidores para manejar más carga de datos, una base de datos *NoSQL* permite distribuir la carga entre varios *hosts* a medida que aumenta la carga.

Inconvenientes:

- La falta de experiencia. La novedad de *NoSQL*
- Problemas de compatibilidad. A diferencia de las bases de datos relacionales, que comparten ciertos estándares, las bases de datos *NoSQL* tienen pocas normas en común. Cada base de datos *NoSQL* tiene su propia API[14], las interfaces de consultas son únicas y tienen peculiaridades. Esta falta de normas significa que es imposible cambiar simplemente de un proveedor a otro por si no quedara satisfecho con el servicio.

De manera general el término *NoSQL* se refiere a una gran cantidad de bases de datos que intentan arreglar las limitaciones que posee el modelo relacional cuando se encuentra en entornos de almacenamiento masivo de datos y concretamente en el momento de escalar, donde es necesario disponer de servidores muy potentes y de balanceo de carga. No significa que el modelo relacional desaparezca, pero sí que en determinados entornos donde se necesita escalar rápidamente, es una solución muy buena, en especial por el gran rendimiento que ofrece. Hoy en día se utilizan, no sólo como almacenamiento primario, sino también como sistema de persistencia para guardar cachés, analíticas de uso y otros datos para los que lo primordial es la velocidad (GUERVÓS y ESPARCIA, 2012).

Hasta este momento se han analizado todos los modelos agrupados por el criterio funcionalidad en el esquema, destacándose por sus particularidades el modelo Objeto-relacional y el modelo *NoSQL.* A continuación se analizan los modelos agrupados por el criterio rendimiento.

Rendimiento: los modelos que aparecen agrupados en este criterio, además de poseer un funcionamiento particular que será tomado en cuenta, son los escogidos para analizar todas aquellas características que fomentan el rendimiento en los sistemas de base de datos que son implementados teniéndolos como base.

Paralelas

El concepto de paralelismo en las bases de datos se puede definir como “la partición de la base de datos (normalmente a nivel de relaciones) para poder procesar de forma paralela en distintos discos y con distintos procesadores una sola operación sobre la base de datos” (CARRION, 1999).

[14] Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

La carga de datos en paralelo desde fuentes externas, es un requisito importante si se van a tratar grandes volúmenes de datos entrantes. Un gran sistema paralelo de bases de datos debe abordar también los siguientes aspectos de disponibilidad:

- El poder de recuperación frente al fallo de algunos procesadores o discos.
- La reorganización interactiva de los datos y los cambios de los esquemas.

Los sistemas paralelos de bases de datos de gran escala se diseñan para operar incluso si falla un procesador o un disco. Los datos se replican en al menos dos procesadores. Si falla un procesador se puede seguir teniendo acceso desde los demás procesadores a los datos que guarda. El sistema hace un seguimiento de los procesadores con fallos y distribuye el trabajo entre los que funcionan. Las peticiones de los datos guardados en el emplazamiento con fallo se desvían automáticamente a los emplazamientos de las copias de seguridad que guardan una réplica de los datos.

Cuando se gestionan grandes volúmenes de datos (del orden de *terabytes*), las operaciones sencillas, como la creación de índices, y los cambios en los esquemas, como añadir una columna a una relación, pueden tardar mucho tiempo (quizás horas o incluso días). Por tanto, es inaceptable que los sistemas de bases de datos no estén disponibles mientras se llevan a cabo tales operaciones. Los sistemas paralelos de bases de datos permiten que tales operaciones se lleven a cabo interactivamente, es decir, mientras el sistema ejecuta otras transacciones (CARRION, 1999).

Ventaja:

- Aumenta la capacidad de proceso.

Inconveniente:

- Limitaciones en la cantidad de memoria que el sistema direcciona.

Los sistemas de bases de datos paralelos, como casi toda la tecnología paralela, fue acuñada como la tecnología del futuro en cuanto a altas prestaciones. Hoy en día la postura es más realista y se reconoce su uso en sistemas de muy altas prestaciones, aunque para sistemas de uso corriente, incluso grandes empresas, su uso es más limitado. No obstante, por muy paralelo que fuera el sistema, todo ordenador tiene su límite (ORALLO, 2002).

Distribuidas

“Una Base de Datos Distribuida (llamada en adelante BDD), es una colección de datos relacionados lógicamente, pero dispersos sobre diferentes sitios de una red de computadoras. Cada sitio en la red tiene

capacidad de procesamiento autónomo y puede ejecutar aplicaciones locales." (CERI y PELAGATTI, 1984).

En la Figura **4**se presenta un esquema que ejemplifica claramente el concepto de una BDD.

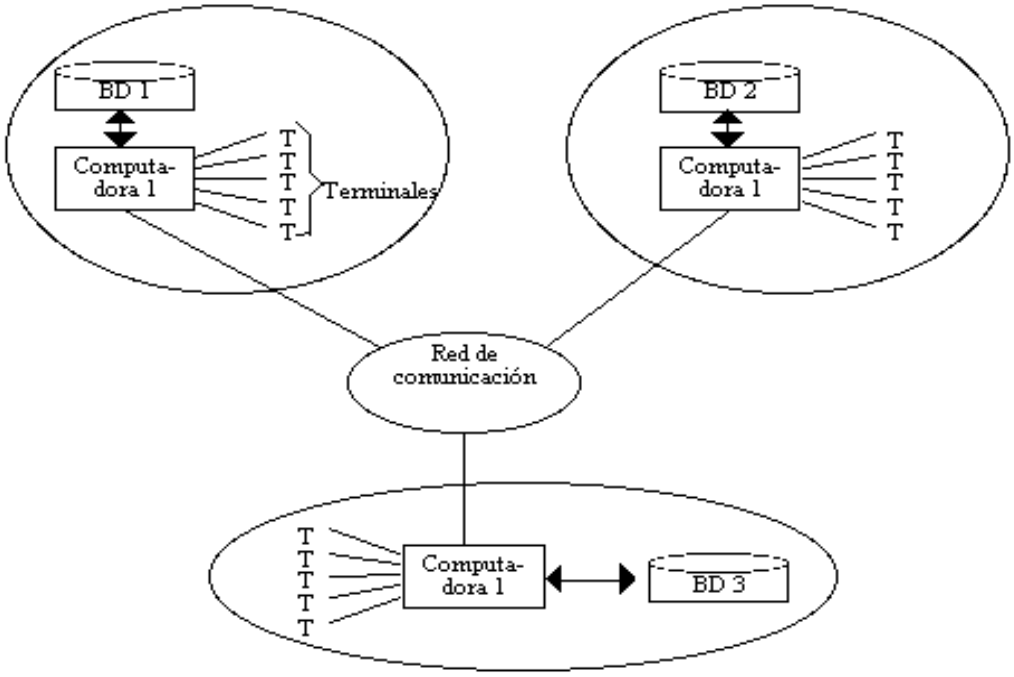

Figura 4. **Una BDD en una red geográficamente dispersa.**

Las BDD constituyen un grupo de datos que pertenecen a un sistema pero a su vez está repartido entre ordenadores de una misma red, ya sea a nivel local o cada uno en una diferente localización geográfica, cada sitio en la red es autónomo en sus capacidades de procesamiento y es capaz de realizar operaciones locales y en cada uno de estos ordenadores debe estar ejecutándose una aplicación a nivel global que permita la consulta de todos los datos como si se tratase de uno solo (CARRION, 1999).

Un Sistema de Bases de Datos Distribuida (SBDD) es un sistema en el cual múltiples sitios de bases de datos están ligados por un sistema de comunicaciones de tal forma que, un usuario desde cualquier sitio, puede acceder a los datos en cualquier parte de la red exactamente como si los datos estuvieran en un mismo servidor de bases de datos. Los procesadores de un sistema distribuido pueden variar en cuanto a su tamaño y función. Los mismos pueden incluir microcomputadores pequeños, estaciones de trabajo y sistemas de computadores grandes de aplicación general. No comparten la memoria principal ni el reloj.

Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red para transmitir datos y órdenes entre los nodos.

Los doce objetivos de las base de datos distribuidas son:

1. Principio fundamental: para el usuario un sistema distribuido debe ser igual que uno centralizado.
2. Autonomía local: los sitios distribuidos deben ser autónomos, es decir que todas las operaciones en un sitio dado se controlan en ese nodo.
3. No dependencia de un sitio central: no debe de haber dependencia de un sitio central para obtener un servicio.
4. Operación continua: nunca debería apagarse para que se pueda realizar alguna función, como añadir un nuevo nodo.
5. Independencia con respecto a la localización: no es necesario que los usuarios sepan dónde están almacenados físicamente los datos, sino que el usuario lo debe de ver como si solo existiera un nodo local.
6. Independencia con respecto a la fragmentación: la fragmentación es deseable por razones de desempeño, los datos, pueden almacenarse en la localidad donde se utilizan con mayor frecuencia de manera que la mayor parte de las operaciones sean sólo locales y se reduzca el tráfico en la red.
7. Independencia de réplica: si una relación dada (es decir, un fragmento dado de una relación) se puede presentar en el nivel físico mediante varias copias almacenadas o réplicas, en muchos sitios distintos.
8. Procesamiento distribuido de consultas: el objetivo es convertir transacciones de usuario en instrucciones para manipulación de datos y así reducir el tráfico en la red implica que el proceso mismo de optimización de consultas debe ser distribuido.
9. Manejo distribuido de transacciones: tiene dos aspectos principales, el control de recuperación y el control de concurrencia, cada uno de los cuales requiere un tratamiento más amplio en el ambiente distribuido.
10. Independencia con respecto al equipo: el SGBDD debe ser ejecutable en diferentes plataformas de hardware.

11. Independencia con respecto al sistema operativo: el sistema debe ser ejecutable en diferentes Sistemas Operativos.
12. Independencia con respecto a la red: el sistema debe poder ejecutarse en diferentes redes (GARCÍA y FERNÁNDEZ, 2009).

La principal ventaja de los sistemas distribuidos es la capacidad de compartir y acceder a la información de forma fiable y eficaz. Además de:

- Utilización compartida de los datos y distribución del control. La ventaja principal de compartir los datos por medio de la distribución es que cada localidad pueda controlar hasta cierto punto los datos almacenados localmente.
- Fiabilidad y disponibilidad. El sistema sigue funcionando aún en caso de caída de uno de los nodos. Si se produce un fallo en una localidad de un sistema distribuido, es posible que las demás localidades puedan seguir trabajando. En particular, si los datos se repiten en varias localidades, una transacción que requiere un dato específico puede encontrarlo en más de una localidad. Así, el fallo de una localidad no implica necesariamente la desactivación del sistema.
- Agilización del procesamiento de consultas. Si una consulta comprende datos de varias localidades, puede ser posible dividir la consulta en varias sub-consultas que se ejecuten en paralelo en distintas localidades. Sin embargo, en un sistema distribuido no se comparte la memoria principal, así que no todas las estrategias de intersección se pueden aplicar en estos sistemas. En los casos en que hay repetición de los datos, el sistema puede pasar la consulta a las localidades más ligeras de carga.

El inconveniente principal de los sistemas distribuidos es la complejidad que se requiere para garantizar una coordinación adecuada entre localidades.

- Coste del desarrollo de software: es más difícil estructurar un sistema de bases de datos distribuido, por lo tanto su coste es mayor.
- Mayor posibilidad de errores: puesto que las localidades del sistema distribuido operan en paralelo, es más difícil garantizar que los algoritmos sean correctos.
- Mayor tiempo extra de procesamiento: el intercambio de mensajes y los cálculos adicionales son una forma de tiempo extra que no existe en los sistemas centralizados.
- Aumento de la sobrecarga en las actualizaciones: el sistema debe asegurar que todas las réplicas de la tabla sean consistentes (DATE, 2001).

Fragmentación de datos

El problema de fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red. Esto presenta diferentes interrogantes a la hora de aplicar este método ya que se debe definir cuál es la unidad razonable distribución. Se puede considerar que una relación completa es lo adecuado ya que las vistas de usuario son subconjuntos de las relaciones. Sin embargo, el uso completo de relaciones, no favorece las cuestiones de eficiencia sobre todo aquellas relacionadas con el procesamiento de consultas. La otra posibilidad es usar fragmentos de relaciones (sub-relaciones) lo cual favorece la ejecución concurrente de varias transacciones que acceden a porciones diferentes de una relación. En resumen, el objetivo de la fragmentación es encontrar un nivel de particionamiento adecuado en el rango que va desde *tuplas* o atributos hasta relaciones completas.

Tipos de fragmentación

Dado que una relación se corresponde esencialmente con una tabla y la cuestión consiste en dividirla en fragmentos menores, inmediatamente surgen dos alternativas lógicas para llevar a cabo el proceso: la división horizontal y la división vertical.

La división o fragmentación horizontal trabaja sobre las *tuplas*, dividiendo la relación en sub-relaciones que contienen un subconjunto de las *tuplas* que alberga la primera.

La fragmentación vertical, en cambio, se basa en los atributos de la relación para efectuar la división. Estos dos tipos de partición podrían considerarse como fundamentales y básicos.

Sin embargo, existen otras alternativas. Fundamentalmente, se habla de fragmentación mixta o híbrida cuando el proceso de partición hace uso de los dos tipos anteriores. La fragmentación mixta puede llevarse a cabo de tres formas diferentes: desarrollando primero la fragmentación vertical y posteriormente, aplicando la partición horizontal sobre los fragmentos verticales (denominada partición VH), o aplicando primero una división horizontal para luego, sobre los fragmentos generados, desarrollar una fragmentación vertical (llamada partición HV), o bien, de forma directa considerando la semántica de las transacciones.

Otro enfoque distinto y relativamente nuevo, consiste en aplicar sobre una relación, de forma simultánea y no secuencial, la fragmentación horizontal y la fragmentación vertical; en este caso, se generara una rejilla y los fragmentos formaran las celdas de esa rejilla, cada celda será exactamente un fragmento vertical y un fragmento horizontal (nótese que en este caso el grado de fragmentación alcanzado es máximo, y no por ello la descomposición resultará más eficiente) (ROB y CORONEL, 2003).

Federadas

"Los sistemas de bases de datos federadas (SBDF) son colecciones de componentes cooperativos pero autónomos de sistemas de bases de datos convencionales" (SHETH y LARSON, 1990).

Un SBDF es un sistema múltiple de base de datos, en el cual, cada nodo en la federación mantiene su autonomía en los datos y define un conjunto de esquemas de exportación, a través de los cuales se hacen disponibles los datos a otros nodos (RABELO *et al.*, 1999).

Dependiendo del enfoque, un sistema federado debe cumplir con ciertas características, por ejemplo, en (SHETH y LARSON, 1990) se presentan las siguientes:

- Distribución. Los datos pueden estar ubicados entre múltiples bases de datos.
- Heterogeneidad. Se debe permitir diferencias en el *hardware*, *software* y en los sistemas de comunicación.
- Heterogeneidad de semántica. Ocurre cuando hay discrepancias acerca del significado, interpretación o pretensión de utilización de los mismos datos o datos relacionados.
- Autonomía. Se define como la capacidad de manejar su propio sistema de base de datos, es decir, que tengan control separado e independiente.

En las base de datos federadas el sistema está conformado por un conjunto de bases de datos heterogéneas con un alto grado de autonomía local. Cada nodo en el sistema tiene sus propios usuarios, aplicaciones y datos locales y es el sistema el que trata con ellos directamente y sólo conecta con otros nodos en busca de información que no tiene. A diferencia de los sistemas homogéneos, los sistemas heterogéneos pueden incluir diferentes SGBD en los nodos. Pueden o no tener diferentes sistemas operativos, diferente *hardware*, diferentes gestores de bases de datos, diferentes modelo de datos (en red, relacional, orientada a objetos), diferentes estructura de datos. El modelo de base de datos federada deberá tener un optimizador de recursos para aprovechar correctamente todos los componentes. Pueden ser físicamente distribuidas en diferentes lugares e incluso en lugares muy lejanos (SOARES, 2005).

La importancia de las bases de datos federadas se encuentra principalmente en su doble comportamiento, es decir, en su capacidad de atender consultas globales, al mismo tiempo que permite que los componentes de la base de datos sigan atendiendo a sus aplicaciones locales. El componente de bases de datos está interconectado a través de una red de computadoras el cual puede estar descentralizado geográficamente.

Móviles

Una base de datos móvil es “una base de datos donde los usuarios pueden acceder a la información lejos de donde se encuentra almacenada la base de datos, se hace utilizando una conexión inalámbrica” (LAWRENCE, 2009).

Todas las bases de datos móviles tienen una arquitectura similar (ver Figura **5**), donde se distinguen una serie de elementos característicos de este tipo de sistema:

- Servidor de base de datos corporativo y Sistema Gestor de Base de Datos Móvil (SGBDM) que gestiona y almacena los datos corporativos y proporciona aplicaciones corporativas.
- Base de datos remota y SGBDM que gestiona y almacena los datos móviles. Son las bases de datos que deben estar implementadas en los dispositivos móviles.
- Plataforma de base de datos móvil, que puede ser un ordenador portátil u otro dispositivo de acceso a Internet, es decir, los dispositivos móviles en cuestión.
- Enlaces de comunicación bidireccionales entre el SGBDM corporativo y el SGBDM móvil, que pueden ser redes inalámbricas de distinta naturaleza, comunicaciones vía satélite, etc.

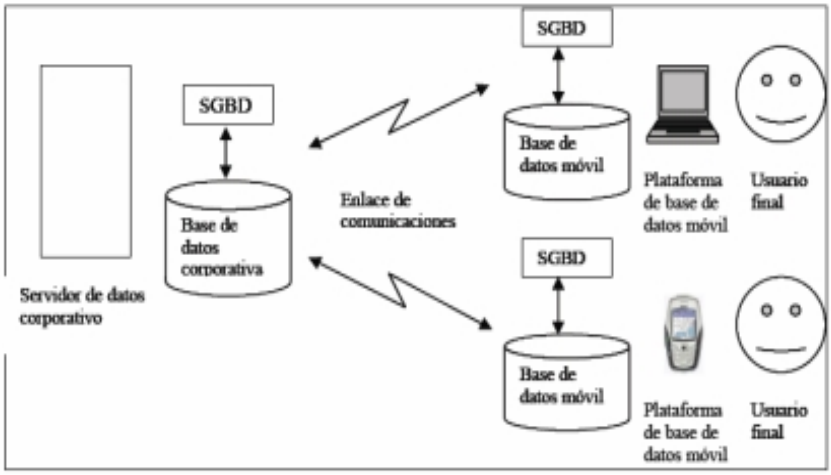

Figura 5. **Arquitectura de una base de datos móvil.**

En una BDM, la comunicación entre los dispositivos es una parte importante, ya que es imprescindible una buena comunicación para el acceso a los datos. La arquitectura de comunicaciones más utilizada consiste en tener una o varias estaciones base en contacto con la base de datos corporativa y una serie de estaciones móviles que acceden a los datos a través de las estaciones base. Una cuestión importante relacionada con el acceso o la localización puede ser localizar una estación móvil que contenga los datos

que se necesiten en un momento determinado. Para solucionar dicho problema existen varias soluciones propuestas, por ejemplo, que cada estación móvil esté asociada a una estación base principal la cuál conoce en todo momento la localización de la estación móvil debido a que la principal recibe notificaciones de los movimientos de la estación móvil.

Ventajas:

- Permiten la movilidad de los usuarios, por lo que no es necesario estar físicamente en la organización para acceder a sus datos. Éstos pueden ser accedidos remotamente.
- El mercado potencial de este tipo de bases de datos es bastante amplio, ya que muchas empresas poseen trabajadores que necesitan acceder a los datos de la compañía mientras se encuentran en localizaciones remotas.
- Estas bases de datos poseen un gran ámbito de aplicación ya que en principio, cualquier base de datos relacional puede ampliarse para ofrecer los servicios de las bases de datos móviles.

Inconvenientes:

- Los enlaces de comunicaciones juegan un papel importante en el desarrollo de estos sistemas, por lo que su dependencia puede suponer un freno para ellos.
- Los datos pueden estar replicados, por lo que la consistencia y coherencia de los mismos son fundamentales y puede generar conflictos importantes.
- El tratamiento de fallos es un aspecto delicado ya que al tratarse de un entorno distribuido, los fallos de transmisión de datos deben de solucionarse y detectarse de forma eficiente para que no produzcan errores en la información tratada.
- La capacidad de procesado de los dispositivos móviles, ya que dicha capacidad no es la misma para todos los dispositivos (LAWRENCE, 2009).

Grid

Es una tecnología que surgió como una nueva forma de computación distribuida. Esta tecnología se basa en la utilización de recursos externos además de los locales, logrando con ello una mayor disponibilidad de recursos para la realización de una tarea.

Como consecuencia del uso de una red Grid, un usuario puede hacer uso de recursos libres situados en los computadores que se encuentren dentro de esta red Grid, sin importar la localización del mismo. De este modo, el usuario dispone de un computador ficticio con la potencia, disco duro o memoria RAM necesitada. Por otro lado, se puede decir que con Grid, no se pone atención en los datos que se

transmiten en sí, como es el caso de los sistemas cliente-servidor sino que el punto de interés y estudio son los recursos computacionales y el uso que se hace de ellos. Otro avance de Grid es que genera un incremento de las posibilidades del uso de Internet ya que proporciona un incremento de su usabilidad. De este modo se obtiene una mayor velocidad de procesamiento así como la facilidad de tener bases de datos de mayor tamaño.

Características:

- Capacidad de balanceo de sistemas. La capacidad se puede reasignar desde la granja de recursos a donde se necesite.
- Alta disponibilidad. Si un servidor falla, se reasignan los servicios en los servidores restantes.
- Reducción de costes. Los servicios son gestionados por granjas de recursos. No es necesario disponer de grandes servidores y se puede hacer uso de componentes de bajo coste.

Ventajas:

- Nunca queda obsoleta, se integran diferentes tipos de máquinas y de recursos y todos los recursos se aprovechan.
- Brinda la posibilidad de compartir, acceder y gestionar información, mediante la colaboración y la flexibilidad operacional, aunando no sólo recursos tecnológicos dispares, sino también personas y aptitudes diversas.
- Tiende a incrementar la productividad, otorgando a los usuarios finales acceso a los recursos de computación, datos y almacenamiento que necesiten, cuando los necesiten.
- La tolerancia a fallos, si una de las máquinas que forman parte del Grid colapsa, el sistema lo reconoce y la tarea se reenvía a otra máquina, con lo cual se cumple el objetivo de crear infraestructuras operativas flexibles y resistentes, así se obtiene una tecnología más robusta y resistente, capaz de responder a desastres.

Inconveniente:

- Necesita para mantener su estructura, de diferentes servicios como Internet, conexiones de 24 horas, los 365 días, con banda ancha, servidores de capacidad, seguridad informática, redes virtuales privadas (en inglés VPN), *firewalls*, encriptación, comunicaciones seguras, políticas de seguridad, normas *ISO*[15], entre otras.

[15] Organización Internacional de Normalización.

La computación Grid está creada con el fin de ofrecer una solución a determinadas cuestiones, como problemas que requieren de un gran número de ciclos de procesamiento o un acceso a una gran cantidad de datos (CARRION, 1999).

En la nube

Básicamente la computación en la nube consiste en los servicios ofrecidos a través de la red tales como correo electrónico, almacenamiento, uso de aplicaciones, etc., los cuales son normalmente accesibles mediante un navegador web. Al utilizar estos servicios, la información utilizada y almacenada, así como la mayoría de las aplicaciones requeridas, son procesados y ejecutados por un servidor en Internet.

A partir de que Google y Amazon construyeran su propia infraestructura surgió una arquitectura: un sistema de recursos distribuidos de manera horizontal, introducidos como servicios virtuales de Tecnología de Información (TI) escalados masivamente y manejados como recursos agrupados y configurados continuamente.

El modelo de esta arquitectura tiene como base a las granjas de servidores. Estas eran similares en su arquitectura al procesamiento en red (Grid), sin embargo, mientras que las redes se utilizan para aplicaciones de procesamiento técnico con un acoplamiento débil (consistentes en un sistema compuesto de subsistemas con cierta autonomía de acción que mantienen una interrelación continua entre ellos formando una “supercomputadora virtual” para realizar grandes tareas), la nube orientó sus aplicaciones a los servicios de Internet.

Algunas de las características que poseen las bases de datos en la nube son:

- No es necesario disponer de un equipo potente. Tan solo se necesita un equipo con conexión a Internet; esto debido a que el dispositivo del usuario no realizará ningún proceso complejo y los ficheros pueden guardarse en la nube.
- No hay necesidad por parte del usuario de conocer la infraestructura detrás de esta. Pasa a ser una abstracción, una nube donde las aplicaciones y servicios pueden fácilmente crecer, funcionar rápido y con pocas fallas.
- Es auto reparable. En caso de surgir un fallo, el último respaldo (*backup*) de la aplicación se convierte automáticamente en la copia primaria y a partir de esta se genera uno nuevo.
- Es escalable. Todo el sistema y su arquitectura son predecibles y eficientes. Se establece un nivel de servicios que crea nuevas instancias de acuerdo a la demanda de operaciones existente de tal forma que se reduzca el tiempo de espera y los cuellos de botella.

- Virtualización. Las aplicaciones son independientes del hardware en el que se ejecuten, incluso varias aplicaciones pueden correr en una misma máquina o una aplicación puede usar varias máquinas a la vez.
- Posee un alto nivel de seguridad. El sistema está creado de tal forma que permite a diferentes clientes compartir la infraestructura sin preocuparse de ello y sin comprometer su seguridad y privacidad; de esto se ocupa el sistema proveedor que se encarga de cifrar los datos.
- Disponibilidad de la información. No se hace necesario guardar los documentos editados por el usuario en su computadora o en medios físicos propios ya que la información radicará en Internet permitiendo su acceso desde cualquier dispositivo conectado a la red (con autorización requerida).

Ventajas:

- Ahorro en licencias, en la administración del servicio y en los equipos necesarios. Si se cuenta con una infraestructura 100 % basada en nube computacional no se requiere instalar ningún tipo de hardware, solo los terminales.
- Implementación rápida y baja en riesgos. Gracias a una infraestructura de *Cloud Computing* (computación en la nube), es posible comenzar a trabajar rápidamente. No es necesario esperar mucho tiempo e invertir grandes cantidades de dinero antes de que un usuario inicie sesión en su nueva solución.
- Actualizaciones automáticas. No afectan negativamente a los recursos de TI. Si se actualiza a la última versión de la aplicación, la nueva tecnología no obliga al consumidor a decidir entre actualizar o conservar su trabajo, porque las personalizaciones e integraciones se conservan automáticamente durante la actualización.
- Portabilidad de información. Aunque en un principio la mayoría de los proveedores en la nube dirigían sus servicios a los usuarios corporativos, con el paso del tiempo los usuarios particulares han comenzado a usar este concepto de forma masiva y casi sin darse cuenta con el uso de servicios para teléfonos móviles (*Smart phones* particularmente), *tablets*, etc.

Inconvenientes:

- Problemas de confiabilidad.
- Fuga de información. Un problema común dado la variedad de los datos que los proveedores en la nube almacenan, lo que implica que en cualquier fuga de información puede ocurrir un significativo impacto.

- Disponibilidad de las aplicaciones. Está atada a la existencia de acceso a Internet. Si un consumidor decide tener todos sus servicios informáticos en la nube, queda sujeto a la cobertura de red.
- Problema de escalabilidad a largo plazo. A medida que más usuarios comiencen a compartir la infraestructura de la nube, la sobrecarga en los servidores de los proveedores aumentará.

La computación en la nube, de forma general, es un paradigma en el que la información se almacena de manera permanente en servidores que se encuentran conectados a Internet y se envía a cachés de clientes, de manera temporal, lo cual permite, que la información visualizada se mantenga en los servidores que prestan estos diferentes servicios, lo que incluye equipos de escritorio, portátiles, móviles, etc. La computación en la nube permite diversificar y elevar el número de servicios sustentados en la red. Esto beneficia en gran parte a los proveedores, que pueden brindar, de manera más rápida y eficiente, una mayor cantidad de servicios, para los usuarios que tienen la posibilidad de acceder a ellos, de manera rápida, transparente y eficiente (MEJIA, 2011).

Luego de ser analizados cada uno de los modelos agrupados en la categoría de rendimiento se estimó que el SGBD debe ser distribuido por los beneficios que brinda la distribución de los datos. Por la necesidad que existe en la red social de la Universidad de las Ciencias Informáticas de almacenar la gran cantidad de *logs* que se generan a partir de la actividad de cada usuario en dicha red social, se selecciona como modelo de datos el *NoSQL*, el mismo permitirá desarrollar una herramienta para gestionar la base de datos capaz de almacenar el gran volumen de datos que se generará, así como acceder a la información de forma concurrente con un alto rendimiento.

1.3 Almacenamiento de datos en las redes sociales

Las redes sociales poseen SGBD que permiten gestionar y almacenar los datos que se generan en las mismas, cada uno de estos sistemas se construye sobre la base de un modelo de datos o la combinación de varios de estos. Para el desarrollo de la presente investigación se seleccionaron cinco de las redes sociales que mayor impacto han tenido mundialmente, con el fin de analizar los SGBD que utilizan cada una de estas y utilizar esta información como guía para tomar una decisión respecto a cuál herramienta utilizar para la propuesta de solución. Al analizar cada SGBD se tiene en cuenta aspectos como: su funcionamiento, si es distribuido, su capacidad de respaldo, el manejo de transacciones y de concurrencia, el rendimiento y la seguridad.

MySpace

Creada en el año 2003, creció gracias al apoyo que recibió por parte de las bandas de música *indie*[16], ofrecía la posibilidad de personalizar las páginas de los propios usuarios (ROS-MARTÍN, 2009). Su objetivo es que un usuario tenga una página principal que podrá controlar completamente. Muchos grupos de música (comerciales y no comerciales) aprovechan *MySpace* para publicitarse y hacerse un hueco en la Web 2.0 (WEB, 2007). *MySpace* posee 30 millones de usuarios registrados (hasta diciembre de 2011), gestiona 8 mil millones de relaciones de amistad, posee 4.4 millones de usuarios concurrentes en las horas pico y almacena alrededor de 34 mil millones de mensajes de correo electrónico. Utiliza como sistema gestor de base de datos *SQL Server 2005* (MYSPACE, 2012).

SQL Server 2005 es un sistema para la gestión de bases de datos creado por *Microsoft* el cual se basa en el modelo relacional, constituye una plataforma global que posee un motor de base de datos relacional seguro, confiable, escalable y altamente disponible con mejor rendimiento y compatible para datos estructurados y sin estructura (XML). Incluye entre sus herramientas el servicio de réplica de datos para aplicaciones de procesamiento de datos distribuidas o móviles, alta disponibilidad de los sistemas, concurrencia escalable con almacenes de datos secundarios para soluciones de información empresarial e integración con sistemas heterogéneos, incluidas las bases de datos Oracle existentes. Sus principales características y desventajas son (CORPORATION, 2005):

Características

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Permite administrar información de otros servidores de datos.

Desventajas

[16] El término proviene de la palabra independiente, es cualquier género, panorama, subcultura y atributo estilístico y cultural con un acercamiento autónomo y un nivel de planteamiento que se reduce al lema: "Hazlo tú mismo". HESMONDHALGH, D. Indie: The institutional politics and aesthetics of a popular music genre. *Cultural studies*, 1999, vol. 13, nº 1, p. 34-61. [Consultado el: 10 de junio de 2012]. ISSN 0950-2386.

- Usa *Address Windowing Extension* (AWE) para hacer el direccionamiento de 64-bit. Esto le impide usar la administración dinámica de memoria y sólo le permite alojar un máximo de 64 GB de memoria compartida.
- No manipula compresión de datos (excepto la versión 2008 Enterprise Edition, que sí lo hace), por lo que las bases de datos pueden llegar a ocupar mucho espacio en disco.
- Requiere de un sistema operativo Microsoft Windows, por lo que no puede instalarse, por ejemplo, en servidores Linux, por esta razón.

SQL Server 2005 proporciona funciones de copia de seguridad y restauración de alto rendimiento. El componente de copia de seguridad y restauración de SQL Server ofrece una protección esencial para los datos críticos almacenados en las bases de datos de SQL Server. La implementación de una estrategia correctamente planeada de copia de seguridad y restauración contribuye a la protección de las bases de datos de la pérdida de datos derivada de daños causados por diferentes errores. La comprobación de la estrategia mediante la restauración de las copias de seguridad y la recuperación de la base de datos permite responder de forma eficaz ante un desastre (MSDN, 2007a).

Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo la cual debe exhibir cuatro propiedades, conocidas como propiedades de atomicidad, coherencia, aislamiento y durabilidad (ACID), para ser calificada como transacción.

Los programadores de SQL son los responsables de iniciar y finalizar las transacciones en puntos que exijan la coherencia lógica de los datos. Deben definir la secuencia de modificaciones de datos que los dejan en un estado coherente en relación con las reglas corporativas de la organización e incluir esas instrucciones de modificación en una sola transacción de forma que, SQL Server 2005 *Database Engine* (Motor de base de datos de SQL Server 2005) pueda exigir la integridad física de la misma.

Database Engine (Motor de base de datos) proporciona (MSDN, 2007b):

- Servicios de bloqueo que preservan el aislamiento de la transacción.
- Servicios de registro que aseguran la durabilidad de la transacción. Aunque se produzca un error en el hardware del servidor, el sistema operativo o la instancia de *Database Engine*, utiliza registros de transacciones al reiniciar, para revertir automáticamente las transacciones incompletas al punto en que se produjo el error del sistema.
- Características de administración de transacciones que exigen la atomicidad y coherencia de la transacción. Una vez iniciada una transacción, debe concluirse correctamente; en caso contrario,

la instancia de *Database Engine* deshará todas las modificaciones de datos realizadas desde que se inició la transacción.

De forma general, SQL Server 2005 es una base de datos relacional compatible con procesadores multi núcleo, con memoria limitada a un máximo compatible con el sistema operativo, posee compatibilidad para bases de datos a gran escala, procesamiento paralelo de operaciones de indexación. Constituye una solución avanzada de alta disponibilidad que incluye conmutación rápida por error y re direccionamiento automático del cliente, solución de copia de seguridad y recuperación de datos. Optimiza automáticamente la base de datos para obtener un rendimiento óptimo, incorpora cifrado de datos para lograr seguridad avanzada de datos, brinda la posibilidad de replicar el *backup* y de ejecutar la restauración *online.*(MSDN, 2007b)

Hi5

Hi5 es una de las redes sociales más conocidas en el mundo, a pesar de que ha bajado su popularidad, debido al impulso que ha tomado *Facebook*(PALACIOS, 2010). Nace en el 2003 y es una idea que revoluciona la forma de conocer y socializar con amigos, familiares y conocidos. Es un espacio en el que el usuario, con comandos muy sencillos, modifica una página en Internet para compartir información personal acerca de sus gustos o preferencias. Para el 2007 ya tenía más de 70 millones de usuarios registrados (sobre todo en Latinoamérica) (PALACIOS, 2010). Hi5 posee actualmente 230 millones de miembros registrados y continúa aumentando, esto se debe a que fue adquirida por *Target*, la misma utiliza como sistema gestor de base de datos *EnterpriseDB*.

EnterpriseDB es una base de datos relacional construida sobre *PostgreSQL*, lo cual le permite desarrollar y desplegar aplicaciones basadas en *PostgreSQL* que escalan por todos lados, desde soluciones integradas *Data Warehouses*[17]. Posee la fiabilidad y escalabilidad necesaria para el manejo de altos volúmenes de datos. Se caracteriza además por su gran rendimiento para lo que utiliza el *Index Advisor*, caché infinita *DynaTune*, un cargador de alta velocidad, seguridad, disponibilidad y usabilidad. El verdadero nombre del producto es *EnterpriseDB Postgres Plus*, el cual es capaz de ofrecer tiempos de respuesta muy rápidos e incluir características avanzadas de seguridad que garantizan la protección de las bases de datos en todo momento, tanto desde el robo de datos hasta la pérdida accidental de datos. Para realizar la distribución de la base de datos de forma uniforme utiliza la herramienta *GridSQL*,

[17] Almacenes de datos.

permitiéndole a los desarrolladores ignorar las complejidades de la distribución de datos y la recuperación de información (ENTERPRISEDB, 2010).

Netlog

En el año 2004 es creada por *Lorenz Bogaert*[18] y *Toon Coppens*[19], es un servicio de redes sociales muy popular, especialmente en Europa. Netlog es la preferencia de millones de personas para contactar a sus amigos y conocer personas a través de Internet. Desde el comienzo, el sitio fue pensado para la juventud europea. A nivel mundial, tiene más de 40 millones de miembros, la mayoría de Europa. Netlog posee más de 40 millones de miembros activos, más de 50 millones de visitantes únicos por mes, 5 millones de páginas vistas al mes más 6 mil millones de minutos en línea al mes. Utiliza como sistema gestor de base de datos Sphinx (ECURED, 2011).

Sphinx es un motor de búsqueda de texto completo, técnicamente es un paquete de software independiente que proporciona una rápida y relevante funcionalidad de búsqueda de texto completo a aplicaciones cliente. Fue diseñado especialmente para integrarse bien con bases de datos SQL que almacenan los datos y acceden fácilmente a lenguajes de *script*. Sin embargo, Sphinx no depende ni requiere ninguna base de datos específica para funcionar. Las aplicaciones pueden acceder al demonio de búsqueda de Sphinx (*searchd*) utilizando cualquiera de los tres métodos de acceso diferentes: a través de la API nativa de búsqueda (*SphinxAPI*), a través de la implementación propia de Sphinx del protocolo de red de MySQL (con un pequeño subconjunto de SQL llamado *SphinxQL*), o a través del servidor de MySQL con un motor de almacenamiento conectable (*SphinxSE*). A continuación se muestran algunas de las características que posee Sphinx, entre las que se encuentran (SERVER, 2012):

- Posee alta velocidad de indexación (hasta 10-15 MB / s por núcleo en un punto de referencia interno),
- búsqueda de alto rendimiento (hasta 150-250 consultas / s por núcleo en contra de 1.000.000 documentos, 1,2 GB de datos en un punto de referencia interno),
- avanzado conjunto de post-procesamiento (SELECT con las expresiones, WHERE, ORDER BY, GROUP BY, etc. sobre los resultados de búsqueda de texto),

[18] Nacido en 1976, es un empresario de los medios de comunicación en línea belga. Es co--fundador de los medios de comunicación masiva, propietario de las marcas digitales de una compañía de medios de comunicación social, tales como servicio de redes sociales Netlog y la comunidad social Twoo.

[19] Director de marketing para el servicio de monedero móvil de Alcatel-Lucent.

- probada escalabilidad con búsquedas distribuidas de hasta miles de millones de documentos, terabytes de datos y miles de consultas por segundo (mayores índices del clúster conocidos de más de 3.000.000.000 de documentos, y mayor actividad en picos de más de 50.000.000 de consultas al día),
- fácil integración con fuentes de datos SQL y XML, *SphinxAPI*, *SphinxQL* o interfaces *SphinxSE* de búsqueda.

Facebook

Construida en febrero de 2004 por *Mark Elliot Zuckerberg*[20]. Originalmente era un sitio para estudiantes de la Universidad de Harvard, sólo se podía acceder a ella si se disponía de un correo electrónico del centro universitario. Posteriormente fue ampliándose en el año 2006 hacia otras universidades, empresas y finalmente, a cualquiera que dispusiese de un correo electrónico. Los usuarios pueden participar en una o más redes sociales, en relación con su lugar de trabajo o región geográfica. Actualmente Facebook posee más de 700 millones de usuarios registrados alrededor de todo el mundo, 5 mil millones de fotos de usuarios, más de 83 millones de fotos subidas a diario, posee 500 000 aplicaciones en el sitio. Utiliza como sistemas gestores de base de datos una combinación entre *HBase* y *Cassandra*, ambos desarrollados por la *Apache Software Foundation* (en español Fundación de Software de Apache) (PÉREZ, 2011).

Cassandra es una base de datos distribuida que responde al modelo de datos *NoSQL*, la misma está escrita en Java y es de código abierto, orientada a columnas, basada en la estructura de *Dynamo* de Amazon y en el modelo de datos de *Bigtable* de Google. Está diseñada para ser altamente escalable tanto en términos de volumen de almacenamiento como de rendimiento de solicitud, sin ser sujeto a ningún punto único de fallo. Se caracteriza por: realizar consulta por la columna, poseer rango de claves, contar con índices secundarios y escribir mucho más rápido de lo que lee. Para la tolerancia a fallos se replican automáticamente los datos a varios nodos y es compatible la replicación a través de múltiples centros de datos. Se pueden reemplazar nodos fallidos sin tiempo de inactividad. Posee una estructura descentralizada, no hay cuellos de botella lo que genera una alta disponibilidad. Cada nodo del clúster es el mismo.

[20] Nacido en 1984, empresario y programador informático estadounidense. Creador y presidente de la comunidad virtual Facebook.

Una instancia de *Cassandra* es una colección de nodos independientes que están configurados para formar un clúster. En un clúster, todos los nodos son iguales, es decir, no hay ningún nodo maestro o proceso de gestión centralizada. Un nodo se une a un clúster de Cassandra sobre la base de ciertos aspectos de su configuración. Cassandra utiliza un protocolo llamado *Gossip* para descubrir la ubicación y la información de estado acerca de los otros nodos que participan en un clúster.

Cuando se inicia un grupo de Cassandra, se debe elegir cómo los datos se dividen a través de los nodos del clúster. Esto se hace mediante la elección de un particionador para el clúster. El *RandomPartitioner* es la estrategia de partición por defecto para un clúster de Cassandra.

La replicación es el proceso de almacenamiento de copias de los datos en varios nodos para garantizar la fiabilidad y tolerancia a fallos. Cuando se crea un espacio de claves en Casandra, se debe decidir la estrategia de colocación de réplicas: el número de réplicas y cómo esas réplicas se distribuyen entre los nodos del clúster. La estrategia se basa en la replicación del clúster configurado, *snitch,* para ayudar a determinar la ubicación física de los nodos y su proximidad entre sí. El número total de réplicas de todo el grupo se refiere a menudo como el factor de replicación. Un factor de replicación de valor 1 significa que sólo hay una copia de cada fila. Un factor de replicación de valor 2 significa dos copias de cada fila. Todas las réplicas son igualmente importantes, no hay réplica primaria o principal en términos de cómo leer y escribir peticiones (CASSANDRA, 2012).

HBase es una base de datos distribuida orientada hacia las columnas, de código abierto, no relacional y escrito en Java. Se desarrolla como parte del proyecto *Apache Hadoop* de la *Apache Software Foundation* y se ejecuta en la parte superior de HDFS (Sistema de archivos distribuido *Hadoop*), proporcionando, para *Hadoop,* capacidades similares a las de *BigTable,* en su esencia es un *hash-map*[21], es decir, se proporciona una forma de almacenar grandes cantidades de datos dispersos tolerante a fallos (HBASE, 2012).

Un sistema HBase comprende un conjunto de tablas. Cada tabla contiene filas y columnas, como una base de datos tradicional. Cada tabla debe tener un elemento definido como clave principal, y todos los intentos de acceso a las tablas de HBase deben utilizar esta clave principal. Una columna de HBase representa un atributo de un objeto. HBase permite muchos atributos que se agrupan en lo que se conoce como familias de columna, de tal manera que los elementos de una familia de columnas se almacenan

[21] Tipo de dato abstracto compuesto por una colección de llaves y una colección de valores, donde se asocia a cada llave con un valor.

todos juntos. Con HBase se debe predefinir el esquema de la tabla y especificar las familias de las columnas, sin embargo, es muy flexible en que se puedan añadir nuevas columnas a las familias en cualquier momento, haciendo que el esquema sea flexible y por lo tanto capaz de adaptarse a los requisitos cambiantes de la aplicación.

Así como HDFS tiene un *NameNode* y nodos esclavos, y *MapReduce* tiene *JobTracker* y esclavos *TaskTracker*, HBase se basa en conceptos similares. En HBase un nodo maestro gestiona el clúster, almacena las porciones de las tablas en las regiones del servidor (en inglés *RegionServer)* y realiza el trabajo en los datos. A continuación se manifiestan algunas de las características que posee HBase, las mismas son (IBM, 2012):

- Orientada a columnas semi-estructurada, almacén de datos,
- distribuido en muchas máquinas,
- conocida por escalar a más de 1000 nodos,
- tolerante a fallos, no existe ningún punto único de fallo,
- consistencia fuerte,
- las filas de bytes almacenados ordenadas en forma lexicográfica,
- tabla dinámica dividida en regiones,
- regiones alojadas en un *RegionServer*,
- optimización de las consultas en tiempo real.

Twitter

Sitio web lanzado el 13 de julio de 2006, permite a sus usuarios enviar y leer micro-entradas de texto de una longitud máxima de 140 caracteres. El envío de estos mensajes se puede realizar tanto por el sitio web de Twitter, como vía SMS[22] desde un teléfono móvil o desde programas de mensajería instantánea. Fue oficialmente lanzado al público en octubre del mismo año. El 4 de noviembre de 2009, Twitter se puso en disponibilidad al español. Posee más de 200 millones de usuarios registrados, generando 65 millones de *tweets* al día y maneja más de 800 000 peticiones de búsqueda diarias. Twitter utiliza FlockDB como sistema gestor de base de datos para el mapeo de identificadores o lo que es lo mismo para almacenar gráficos sociales (quién sigue a quién, quién bloquea a quién) y los índices secundarios; utiliza *Cassandra* para escrituras de alta velocidad (SÁEZ, 2010).

[22] Servicio de mensajes cortos.

FlockDB inicialmente desarrollada para la red social Twitter, es una base de datos en grafo para la gestión de datos a escala web. El sistema está construido para ser totalmente distribuido, tolerante a fallos y bajo la propuesta teórica *MapReduce*. Almacena los datos del grafo, pero no es una base de datos optimizada para las operaciones de recorrido de grafo. En su lugar, está optimizado para las listas de adyacencia muy grandes, rápidas lecturas y escrituras. Posee tiendas de clúster de 13 billones de bordes y sostiene a los picos de demanda de 20 mil escrituras y 100 mil lecturas por segundo. Liberada como software libre bajo una licencia de tipo Apache, es mucho más simple que el resto de bases de datos en grafo existentes. Almacena grafos como conjuntos de aristas entre nodos identificados por enteros de 64 bits (TWITTER, 2010).

Analizar las tendencias que existen en el almacenamiento de datos en las principales redes sociales, caracterizadas anteriormente, brinda la posibilidad de determinar hacia donde se dirigen las redes sociales más populares del mundo en materia de almacenamiento de datos. Permite establecer una comparación entre las herramientas que se investigan, teniendo en cuenta las características que se desean tener en la arquitectura de datos para la red social universitaria de la Universidad de las Ciencias Informáticas, arrojando como resultado principal la selección de la herramienta adecuada para construir el SGBD para dicha red.

1.4 Comparación

Algunas de las características que debe tener el SGBD que se proponga se convierten en criterios que dan la posibilidad de comparar las herramientas analizadas y permiten tomar una decisión que satisfaga las demandas planteadas.

Criterios:

- **Modelo de datos**. Establece el modelo de datos que utiliza el software a través del cual se representan las relaciones existentes entre los datos así como las operaciones que se llevan a cabo sobre ellos.
- **Soporta búsqueda distribuida**. Permite decidir si el software es capaz o no de realizar búsquedas distribuidas.
- **Fiabilidad**. Se evalúa midiendo la tolerancia del sistema ante los fallos que no es más que la frecuencia y gravedad de los fallos, la capacidad de recuperación de un fallo y la capacidad de predicción del programa.

- **Usabilidad**. Facilidad de empleo o uso. Se valora considerando factores humanos, la estética, consistencia y documentación general.
- **Rendimiento**. Se mide por la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia.

Cada uno de los criterios anteriormente descritos se analizó a medida que fueron caracterizadas las herramientas. Existe una amplia variedad de estrategias e implementaciones para el almacenamiento y recuperación de base de datos, la mayoría de las soluciones, específicamente las que responden al modelo relacional, no están construidas para ser sistemas con gran escala y distribución. Las herramientas que responden a este modelo de dato son: *SQL Server 2005* y *EnterpriseDB*, independientemente de las características favorables que poseen en el resto de los criterios, no se tomaron en cuenta para la solución por las limitaciones del modelo relacional. A estas se une *Sphinx*, motor de búsqueda que se integra con bases de datos SQL y que a pesar de poseer gran rendimiento, probada escalabilidad y alta velocidad de indexación, adquiere las características de una base de datos relacional y junto a estas sus limitaciones.

Por otra parte, aparece *FlockDB* como una propuesta interesante. Esta base de datos en grafo, además de estar construida para ser usada en un ambiente totalmente distribuido, ser tolerante a fallos y almacenar grandes volúmenes de datos, no está optimizada para las operaciones de recorrido de grafo. Estas operaciones son la clave para realizar consultas de inferencia en la bases de datos, las cuales ofrecen la información que se desea obtener de los datos almacenados. Esta desventaja la excluye de la propuesta de solución.

Cassandra y *HBase* por su parte responden al modelo de datos *NoSQL,* se construyen para explotar sus ventajas en entornos distribuidos, son sistemas con alta tolerancia ante los fallos y de gran escalabilidad, fáciles de emplear y entender a la vez y clasificados como sistemas de alto rendimiento por la capacidad de leer y escribir miles de millones de *tuplas* en ínfimos tiempos. En el caso particular de *HBase* permite realizar lecturas a grandes volúmenes de datos y *Cassandra* por su parte realiza escritura con un alto rendimiento. Las dos constituyen buenas candidatas, sin embargo por su capacidad de realizar rápidas lecturas se escoge *HBase* como herramienta para la propuesta de solución.

El estudio realizado permitió seleccionar HBase como herramienta para la gestión de los datos en la capa de persistencia de la red social de la Universidad de las Ciencias Informáticas. La misma se utilizará para almacenar en sus tablas los *logs* que se generan a partir de la actividad de los usuarios al acceder a los

servicios que se brindan en la red y a su vez para acceder a dicha información de forma concurrente y con gran rendimiento.

En el presente capítulo se analizaron los modelos de datos representados en el esquema y se seleccionó el modelo *NoSQL* para representar la arquitectura de datos de la red social de la Universidad de las Ciencias Informáticas. Se compararon las herramientas para el almacenamiento de información que utilizan algunas de las redes sociales más populares en el mundo y se seleccionó HBase para almacenar los logs que se generan en la red social, convirtiéndose en la propuesta de solución para la arquitectura de datos de la red social universitaria de la Universidad de las Ciencias Informáticas.

Capítulo 2: Descripción y validación de la herramienta HBase

En el presente capítulo se describen las particularidades de la herramienta HBase, sus características, arquitectura, instalación y seguridad. Se lleva a cabo el diseño y ejecución de las pruebas, se analizan los resultados obtenidos y se realiza la validación de la herramienta.

2.1 HBase

El proyecto HBase se inicia a finales del año 2006 por *Chad Walters* y *Jim Kellerman* de *Power Set*[23]. Fue modelado después del *Bigtable* de Google: sistema de almacenamiento distribuido para estructurar datos. En febrero de 2007, *Mike Cafarella* hizo una disminución de código de un sistema de trabajo que luego *Jim Kellerman* lo llevaría adelante.

2.1.1 Conceptos

En esta sección se ofrece una visión general y rápida de los conceptos básicos de HBase para facilitar el entendimiento de la herramienta.

Componentes del modelo de datos en HBase

Las tablas están hechas de filas y columnas. Las celdas de las tablas y la intersección de filas y columnas coordinadas están versionadas. Por defecto, su versión es una marca de tiempo asignado automáticamente por HBase en el momento de introducción de la celda. El contenido de una celda es una matriz sin interpretar de *bytes*.

Las claves de fila de tablas son también matrices de *bytes*, en teoría cualquier combinación de letras y números puede servir de clave de fila de las cadenas a las representaciones binarias de posiciones largas o incluso las estructuras de datos serializados. Las filas de las tablas están ordenadas según la clave de fila, la clave principal de la tabla, por defecto, el tipo es ordenado por *bytes*. Todos los accesos a las tablas son a través de la clave primaria. Las columnas de filas se agrupan dentro de familias de columnas. Todos los miembros de la familia de columnas tienen un prefijo común, por lo que, las columnas de la temperatura: el aire y temperatura: *dew_point*, son miembros de la familia de columna temperatura.

[23] Compañía que pertenece actualmente a Microsoft y tiene su sede en San Francisco, New York. Tiene como objetivo cambiar la forma en que los seres humanos interactúan con las computadoras a través del lenguaje, es decir, mejorar la búsqueda por indexación de las páginas web basado en el significado que se expresa en ellas en lugar de sólo las palabras literales.

Físicamente todos los miembros de la familia de columnas se almacenan juntos en el sistema de archivos. Así que, aunque anteriormente se ha descrito HBase como una tienda orientada a columnas, sería más exacto si se describe como un almacén de las columnas orientado a la familia. Debido a las afinaciones y las especificaciones de almacenamiento, que se realizan a nivel de la familia de columna, se aconseja que todos los miembros de la familia de columna tengan el mismo patrón de acceso general y las características de tamaño.

En resumen, las tablas HBase son como las de un RDBMS, sólo se versionan las células, las filas se ordenan y las columnas se puede añadir sobre la marcha por el cliente, siempre y cuando la familia de columnas a la que pertenecen preexiste.

Implementación

Justo como HDFS y *MapReduce* están construidas con los clientes, los esclavos y un maestro de coordinación *namenode* y *datanodes* en HDFS, *JobTracker* (rastreador de trabajos) y *Tasktrackers* (rastreador de tareas) en *MapReduce*, HBase se caracteriza por poseer un nodo maestro HBase, conformando un conjunto de uno o más esclavos en la *RegionServer* (ver Figura 6). El maestro HBase es responsable del arranque de una instalación nueva, para la asignación de las regiones para *RegionServer* registrados y para la recuperación de fallos de *RegionServer*. Los *RegionServer* llevan cero o más regiones y peticiones al cliente de campo de lectura / escritura. También aplican división de regiones informando al máster HBase sobre las nuevas regiones hijas para que sea administrado el recubrimiento de la región padre y la asignación de las hijas de reemplazo. HBase depende de *Zookeeper* y por defecto maneja una instancia de *Zookeeper* como la autoridad en estado del clúster.

HBase, a nivel interno, mantiene las tablas de catálogo especial llamado -***ROOT***- y -***META***- dentro de los cuales mantiene la lista actual, el estado, la historia reciente y la ubicación de todas las regiones a flote en el clúster. La tabla -***ROOT***- contiene la lista de las regiones de la tabla -***META***-. La tabla -***META***- contiene la lista de todas las regiones del espacio de usuario. Las entradas de estas tablas tienen la forma adecuada usando la fila de la región de inicio. Las llaves de fila como se señaló anteriormente, se ordenan a fin de encontrar la región que aloja una fila particular, a partir de una búsqueda para encontrar la primera entrada cuya clave es mayor o igual a la de la clave de la fila solicitada. Dado que las regiones de transición se dividen en, deshabilitado y habilitado, eliminado, redistribuido por el balanceador de carga de una región o redistribuido debido a un accidente de *RegionServer* las tablas de catálogo se actualizan de modo que el estado de todas las regiones en el clúster se mantiene al día.

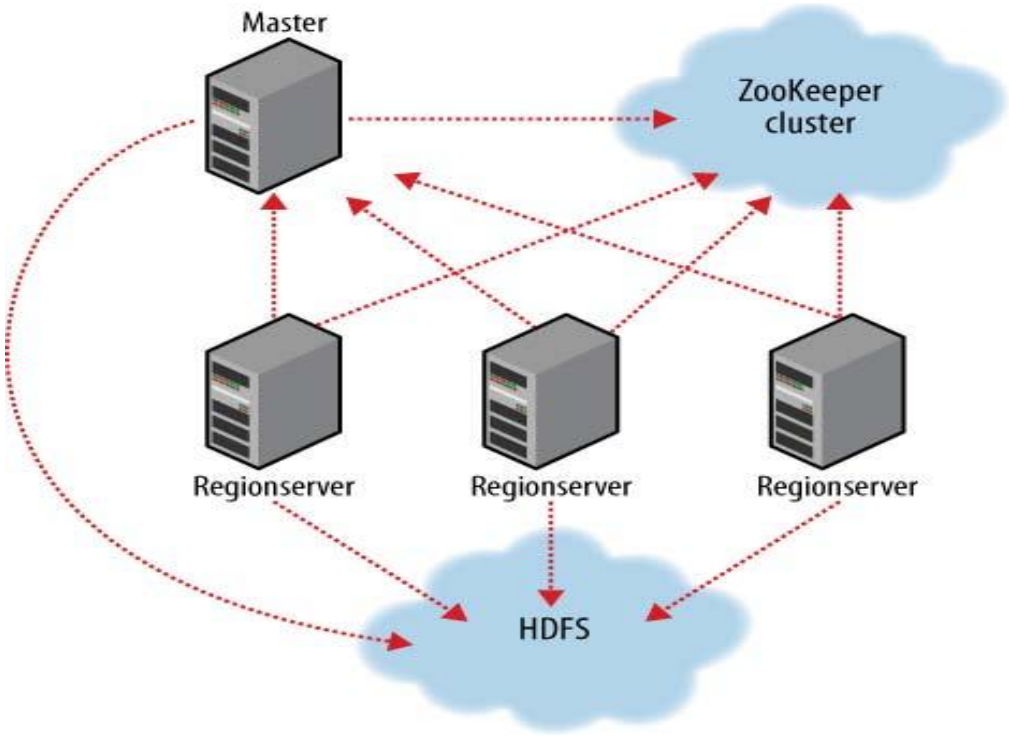

Figura 6. **Los miembros del clúster HBase.**

2.1.2 Características

La primera liberación de HBase fue incluida como parte de Hadoop 0.15.0. Al inicio de 2008, HBase se convirtió en un sub-proyecto de Hadoop, que ha estado en uso en la producción de *Power Set* desde finales de 2007. Entre los usuarios de producción de HBase se incluyen *WorldLingo*, *Streamy.com*, *OpenPlaces*, y los grupos de Yahoo y Adobe (WHITE, 2010).

HBase es un tipo de base de datos *NoSQL*. Es una base de datos distribuida, técnicamente hablando es más un "almacén de datos" que "una base de datos", ya que carece de muchas de las características que se encuentran en un sistema gestor de base de datos relacional (SGBDR), como columnas escritas, los índices secundarios, disparadores y los lenguajes avanzados de consulta, etc. (HBASE, 2012)

Es una distribución de base de datos orientada a columnas, construida en la cima de HDFS (*Hadoop File System*). HBase es la aplicación Hadoop para usar en caso de requerir en tiempo real, lectura y escritura de acceso aleatorio a los conjuntos de datos muy grandes.

HBase está construido de la base para arriba para escalar linealmente, simplemente añadiendo nodos. HBase no es relacional y no es compatible con SQL, pero dado el problema de espacio adecuado es capaz de hacer lo que un sistema gestor de base de datos relacional (sus siglas en inglés RDBMS) no puede: organizar hábilmente grandes tablas de baja densidad en los clústeres de hardware elaborados a partir de hardware de bajo costo. Ofrece búsquedas rápidas de registros (y actualizaciones) para tablas grandes. Esto a veces puede ser un punto de confusión conceptual. HBase internamente pone sus datos en indexados *StoreFiles* que existen en HDFS, de alta velocidad las búsquedas. HDFS es un sistema de archivos distribuido adecuado para el almacenamiento de archivos de gran tamaño. No es, sin embargo, un sistema de propósito general de archivos y no proporciona rápidas búsquedas de registros individuales en los ficheros(WHITE, 2010).

HBase tiene varias características que le permiten soportar escalado. Los grupos HBase se expanden mediante la adición de *RegionServer* que están alojados en los servidores de clase de los productos básicos. Si un grupo se expande de 10 a 20 *RegionServer*, por ejemplo, se duplica tanto en términos de almacenamiento como la capacidad de procesamiento. Un SGBDR puede escalar bien, pero sólo hasta cierto punto y para un mejor desempeño requiere hardware especializado y dispositivos de almacenamiento.

Características de HBase:

- Totalmente compatible lee/escribe. HBase no es un almacén de datos eventualmente consistente. Esto hace que sea adecuado para tareas tales como la agregación de alta velocidad del contador.
- Compartición automática. Las tablas HBase se distribuyen en el clúster a través de las regiones que son divididas automáticamente y se re-distribuyen tanto como los datos crecen.
- Integración de Hadoop/HDFS. HBase apoya HDFS, ya que es el encargado de realizar la distribución a través del sistema de archivos.
- MapReduce. HBase soporta el procesamiento masivamente paralelizado a través de MapReduce para el uso de HBase tanto como fuente y sumidero.
- API de Java. HBase apoya utilizar la API de Java para un fácil acceso mediante programación.
- API Thrift[24] / REST[25]. HBase utiliza las API Thrift y REST en caso de no utilizar Java.

[24] Lenguaje de definición de interfaz. Se utiliza como marco de trabajo para definir y crear servicios para numerosos lenguajes. THRIFT, A. *Thrift* [Consultado el: 04 de junio de 2012]. Disponible en: http://thrift.apache.org/.

[25] Estilo de arquitectura para sistemas hipermedia distribuidos. FIELDING, R. y TAYLOR, R. N. *Representational State Transfer.* Ph. D. Thesis. UCI, 2000, [Consultado el: 11 de junio de 2012].

- Caché de bloques y filtros Bloom. HBase es compatible con una caché de bloques y filtros Bloom para la optimización de consultas de alto volumen.
- Gestión operativa. HBase ofrece construir páginas web para obtener una visión operativa, así como mediciones JMX[26] (*Java Management Extensions*, en español *Administración de Extensiones Java*).

HBase no es apto para todos los problemas, pero si se tienen miles de millones de filas entonces se convierte en un candidato idóneo, además de asegurarse de que existe suficiente hardware. HDFS no funciona bien con menos de 5 *DataNodes* (debido a aspectos como la replicación del bloque HDFS que tiene por defecto), además de un *NameNode*.

HBase se puede ejecutar de dos modos, independiente o distribuido. Por defecto el modo de ejecución es independiente, en el cual no utiliza HDFS sino el sistema de archivo local. El modo distribuido por su parte se puede subdividir en distribuido, donde todos los demonios se ejecutan en un solo nodo, también conocido como pseudo-distribuido y totalmente distribuido, donde los demonios se reparten entre todos los nodos en el clúster (HBASE, 2012).

2.1.3 Mecanismos para realizar consultas a los datos

De forma general, para administrar HBase, crear, eliminar, listar y modificar tablas, se utiliza *HBaseAdmin*. Una vez creada una tabla, el acceso a la misma es a través de una instancia de *HTable* y se puede añadir contenido a la tabla en una fila a la vez. Para insertar, se crea una instancia de un objeto, se especifica el valor, la columna de destino y opcionalmente, una marca de tiempo. La actualización se realiza a través de *HTable.put* (actualizar). Para recuperar el valor insertado se utiliza *Get*, el mismo se puede especificar en sentido amplio (obtener todo en una fila en particular) o reducido (retorna sólo un valor de celda única). Después de crear una instancia de *Get*, se invoca con *HTable.get* (obtener). Para configurar un escáner se utiliza *Scan* (un acceso similar al cursor). Después de la creación y configuración de la instancia del *Scan*, se invoca *HTable.getScanner* (escanear) y luego se invoca al siguiente objeto devuelto. Tanto *HTable.get* (obtener) y *HTable.getScanner* (escanear) devuelven un resultado. Un resultado es una lista

[26] Tecnología que proporciona las herramientas para crear soluciones distribuidas, basadas en la Web, las soluciones modulares y dinámicas para la gestión y supervisión de dispositivos, aplicaciones y redes orientadas a servicios. ORACLE. *Java Management Extensions (JMX) Technology* [Consultado el: 03 de junio de 2012]. Disponible en: http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html.

de *KeyValues*. Para eliminar el contenido de una tabla ya sean células individuales o familias enteras se utiliza *DELETE,* se ejecuta *HTable.delete* (eliminar).

El código de acceso del cliente a un clúster se localiza en el clúster mediante la consulta del *ZooKeeper* esto significa que el quórum *ZooKeeper* a utilizar debe ser en el cliente *CLASSPATH*. Por lo general, esto significa asegurarse de que el cliente encuentre su HBase-site.xml (FOUNDATION, 2012).

Existen numerosas opciones de clientes que permiten manipular, realizar consultas a los datos e interactuar con un grupo HBase, los mismos son: la API de Java, *MapReduce*, *Thrift* y *REST.* A continuación se explica brevemente cómo se utilizan cada uno de estos clientes.

API de Java

Una vez que se tiene un HBase corriendo, una manera de conectar una aplicación en Java al mismo es utilizando la API de Java. Ver en el **Anexo 1** un ejemplo de lo que puede ser un cliente sencillo.

MapReduce

Las clases y utilidades de HBase que aparecen en el paquete *org.apache.hadoop.HBase.mapred* facilitan el uso de HBase como fuente y / o sumidero en los puestos de trabajo *MapReduce*. La clase *TableInputFormat* realiza divisiones en los límites de la región para que los mapas sean entregados a una sola región para trabajar. El *TableOutputFormat* escribe el resultado de *reduce* en HBase. La clase *RowCounte*, puede ser encontrada en el paquete *mapred* de HBase. Para contar las filas se ejecuta una tarea *map* utilizando *TableInputFormat* (WHITE, 2010).

Thrift

Es un lenguaje de definición de interfaz, se utiliza para definir y crear servicios para numerosos lenguajes y como un marco de trabajo de llamada a procedimiento remoto (RPC). Fue desarrollado en Facebook para "escalar entre lenguajes de desarrollo de servicios". En el mismo se combina una pila de software con un motor de generación de código para construir servicios que funcionan de manera eficiente a un mayor o menor grado y sin problemas entre C + +, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C #, Cacao, JavaScript, Node.js, Smalltalk, OCaml, Delphi y otros lenguajes (THRIFT, 2011).

Vea el **Anexo 2** para obtener más información acerca de cómo se utiliza el marco de trabajo Thrift en HBase.

REST

Es un estilo híbrido de arquitectura para sistemas hipermedia distribuidos, derivado de varios de los estilos arquitectónicos basados en la red y en combinación con las restricciones adicionales que definen a un conector de interfaz uniforme. (FIELDING y TAYLOR, 2000).

HBase REST expone tablas, filas, celdas, y los metadatos HBase como recursos URL (FUNDATION, 2012). Ver el **Anexo 3** para obtener mayor información acerca de los métodos que utiliza HBase REST.

2.1.4 Arquitectura

En esta sección se explica todo lo referente a la arquitectura y el funcionamiento de HBase, se describe la función de los miembros o componentes de HBase a partir de la Figura 7 teniendo en cuenta el papel que desempeñan en el clúster.

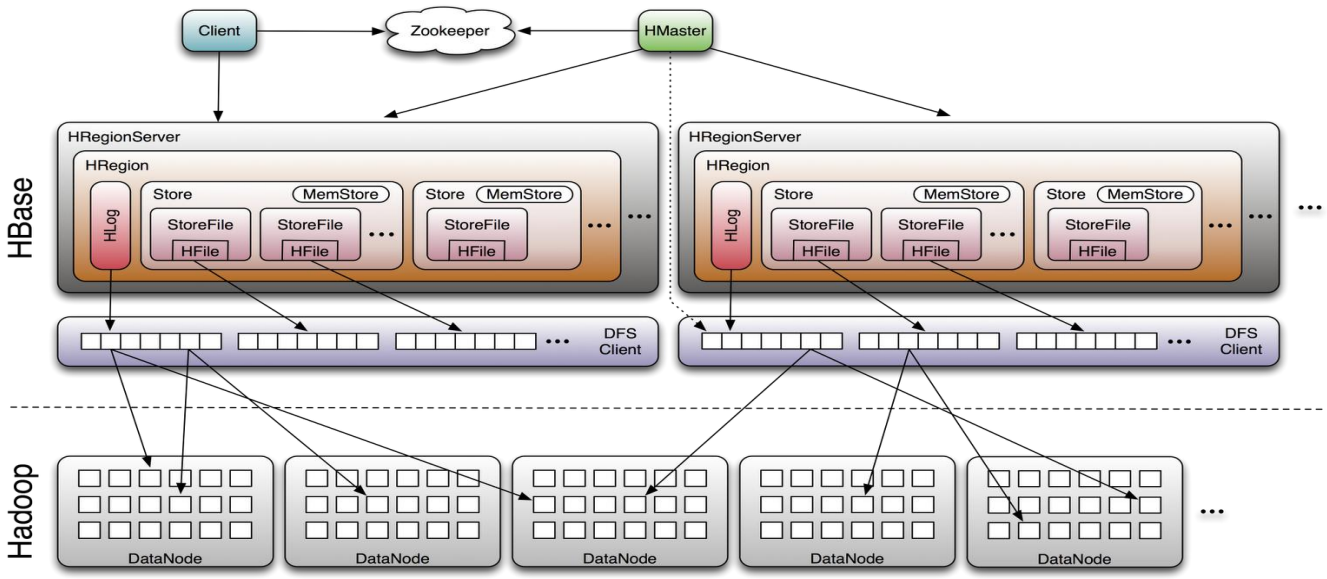

Figura 7. **Arquitectura de HBase.**

Master

Hmaster es la implementación del servidor maestro. El mismo es responsable de supervisar todas las instancias *RegionServer* en el grupo y es la interfaz para todos los cambios en los metadatos. Si se ejecuta en un entorno multi-Master, todos los maestros competirán para administrar el clúster. Si el maestro del principio activo pierde su contrato de arrendamiento en Zookeeper (o el maestro se apaga), entonces se disputan los maestros restantes para asumir la función de maestro.

Los métodos expuestos por *HMasterInterface* son principalmente orientados a los metadatos de los métodos:

- *Table (CreateTable, modifyTable, RemoveTable, enable, disable)*
- *ColumnFamily (addColumn, modifyColumn, removeColumn)*
- *Region (move, assign, unassign)*

Por ejemplo, cuando el método *disableTable* de *HBaseAdmin* se invoca, es atendido por el servidor maestro. El maestro ejecuta varios subprocesos de fondo:

- *LoadBalancerperiodically* reasigna las regiones en el clúster.
- *CatalogJanitorperiodically* revisa y limpia la tabla -**META**-.

RegionServer

HRegionServer es la implementación *RegionServer*. Es responsable de la porción y la gestión de las regiones. Los métodos expuestos por HRegionRegionInterface contienen ambos métodos orientados a datos y región de mantenimiento:

- *Data (get, put, delete, next, etc.)*
- *Region (splitRegion, compactRegion, etc.)*

Por ejemplo, cuando el método *majorCompact* de *HBaseAdmin* se invoca en una tabla, el cliente está realmente iterando a través de todas las regiones de la tabla especificadas y solicitando una mayor compactación directamente a cada región.

El *RegionServer* ejecuta una variedad de temas de fondo:

- *CompactSplitThreadchecks* para divisiones menores y manejar las compactaciones.
- *MajorCompactionCheckerchecks* para compactaciones principales.
- MemStoreFlusherperiodically vuelca en la memoria y escribe en el *MemStore* de *StoreFiles*.
- *LogRollerperiodically* comprueba el *HLog* de la *RegionServer*.

Client

El cliente HBase *HTable* es responsable de encontrar *RegionServer* que están sirviendo a la gama de fila de interés. Para ello consulta la -**META**- y la tabla de catálogo -**ROOT**-. Después de localizar la región deseada, el cliente directamente contacta con el servicio de *RegionServer* de esa región (es decir, que no pasa por el maestro) y las cuestiones de la solicitud de lectura o escritura. Esta información se almacena en caché en el cliente para que las solicitudes posteriores no necesiten pasar por el proceso de búsqueda. En caso de una región ser reasignada, ya sea por el balanceador de carga principal, o porque el

RegionServer ha muerto, el cliente deberá realizar una nueva consulta a las tablas de catálogo para determinar la nueva ubicación de la región del usuario.

ZooKeeper

Un HBase distribuido depende de un clúster *ZooKeeper* en ejecución. Todos los nodos participantes y clientes necesitan ser capaces de tener acceso al conjunto *ZooKeeper* en ejecución. HBase por defecto maneja un empleado del clúster *ZooKeeper*. Se pondrá en marcha y se detendrá el conjunto de *ZooKeeper* como parte del proceso iniciar / detener de HBase. El *ZooKeeper* puede gestionar un conjunto independiente de HBase y HBase solo un punto en el grupo que debe utilizar. Para cambiar la gestión del ZooKeeper de HBase, se utiliza el HBASE_MANAGES_ZK variable que se encuentra en *conf/HBase-env.sh*. Esta variable, que por defecto es true, dice si HBase desea iniciar / detener el conjunto *ZooKeeper* como parte de los servidores de HBase de arranque / parada.

Cuando HBase gestiona el conjunto *ZooKeeper*, puede especificar la configuración *ZooKeeper* utilizando su zoo.cfgfile nativo, o bien, la opción más fácil para especificar sólo las opciones de *ZooKeeper* es directamente en conf / HBase-site.xml. Una opción de configuración *ZooKeeper* se puede establecer como una propiedad en el fichero de configuración de HBase HBase-site.xmlXML anteponer el nombre de la opción *ZooKeeper* con HBase.zookeeper.property.

Region

Físicamente, las tablas se dividen en rangos de filas llamadas regiones. Cada rango de fila contiene las filas de *StartKey* (inclusivo) hasta *Endkey* (exclusivo). Un conjunto de regiones, ordenadas apropiadamente, forma una tabla completa. El rango de fila es identificado por el nombre de tabla y el *StartKey*.

Cada familia de columnas en una región es administrada por un *Store*. Cada *Store* puede tener uno o más *StoreFiles* (un tipo de archivo Hadoop HDFS), que son los elementos básicos de la disponibilidad y la distribución. Los *StoreFiles* son inmutables una vez cerrado. Los StoreFiles se almacenan en el HDFS Hadoop. HBase escala teniendo regiones a través de muchos servidores. El *RegionServer* divide una región, en la región dividida se agregan las regiones hijas de -**META**-, se abren las hijas en la *RegionServer* de alojamiento de los padres y luego informa de la división para el master.

Periódicamente y cuando no hay ninguna región en transición, un equilibrador de carga se ejecutará y se moverá en torno a las regiones para equilibrar la carga del clúster. El período en el que se ejecuta se puede configurar.

Store

Un *Store* alberga una *MemStore* y 0 o más *StoreFiles* (*HFiles*). Un *Store* corresponde a una familia de columnas de una tabla para una región dada.

StoreFile

El formato de archivo *hfile* se basa en el archivo *SSTable* y en el *tfile* de Hadoop.

MemStore

El *MemStore* mantiene en memoria las modificaciones del *Store*. Las modificaciones son *KeyValues*. Cuando se le pregunta para limpiar, el *MemStore* actual se mueve a la memoria instantánea y se borra. HBase continúa sirviendo a las ediciones del *MemStore* nuevo y de respaldo de memoria instantánea.

2.1.5 Seguridad

La seguridad puede identificarse como la necesidad de evitar determinados riesgos en el software, entre los que se encuentran: la incorrecta manipulación de los datos valiosos o sensibles, las interrupciones o los ataques al sistema y la transmisión de virus y otros instrumentos de destrucción de datos. De forma general tiene como objetivo garantizar la disponibilidad, confidencialidad e integridad de la información.

La confidencialidad se refiere a la llamada oculta de los datos reales o información, especialmente en las áreas sensibles, la confidencialidad de los datos sobre los requisitos más estrictos. La integridad de los datos en cualquier estado está sujeta a la necesidad de garantizar que no ocurra supresión, modificación o daño no autorizado. La disponibilidad de datos significa que los usuarios pueden tener las expectativas de la utilización de los datos mediante la utilización de la capacidad.

En el caso particular de HBase, varios son los factores que se deben analizar para tener una idea de cómo se maneja el tema de la seguridad, entre los que se encuentran: la tolerancia a fallos y el respaldo de los datos.

2.1.5.1 Tolerancia a fallos

La tolerancia a fallos es la capacidad de un sistema para seguir funcionando correctamente y no perder los datos, incluso después que algunos componentes de ese sistema han fracasado. Es difícil lograr el cien porciento de la tolerancia a fallos, porque hay muchas circunstancias físicas que no pueden ser planeadas, pero el objetivo de esta es planificar para todas las fallas comunes. En la gestión de la tolerancia a fallos, es importante eliminar los puntos únicos de fallo (punto de fallo), que son los elementos individuales del sistema, que cuando fallan, pueden derribar todo el sistema(EVANS, 2011).

En HBase las características básicas de la tolerancia a fallos son:

- No existe ningún punto único de fallo,
- aislamiento del componente que falla,
- contención del fallo para impedir la propagación del mismo,
- la disponibilidad de los modos de reversión.

HBase utiliza Hadoop para garantizar aspectos esenciales que lo convierten en un sistema tolerante a fallos y la disponibilidad constituye el elemento fundamental. Existen tres formas o mecanismos que se utilizan en HBase para contrarrestar cualquiera de estos:

- Réplica. Proporciona múltiples instancias idénticas del sistema, la dirección de las tareas y solicitudes de todos ellos en paralelo, permitiendo en caso de que ocurra un fallo poder utilizar cualquier instancia del sistema que halla sido replicada.
- Redundancia. Proporciona múltiples instancias idénticas del sistema y cambia a uno de los casos restantes en caso de fallo,
- Diversidad. Proporciona múltiples implementaciones diferentes de la misma especificación y se utiliza como sistema replicado para hacer frente a los errores en una aplicación específica.

En HBase cuando existe un fallo el sistema no se queda colgado sino reparte la carga que existía en ese punto por todo el clúster, utiliza las réplicas que se le hacen al sistema para restaurar el mismo, dándole al usuario la apariencia de que no ha sucedido nada.

2.1.5.2 Respaldo de datos

Hay dos estrategias generales para la realización de copias de seguridad en HBase: realizar copias de seguridad de apagado completo y la copia de seguridad en un clúster en vivo. Cada método tiene sus pros y sus contras.

Copia de seguridad de apagado completo. Algunos ambientes pueden tolerar una interrupción periódica completa de su clúster HBase, por ejemplo, si se está utilizando una capacidad analítica de fondo y no sirve de *front-end* (frontal) para las páginas web. Los beneficios son que el *NameNode* / Maestro está en la parte baja de la *RegionServer*, así que no hay posibilidad de perder todos los cambios durante el vuelo a cualquier *StoreFiles* o metadatos. Es obvio que el clúster se ha reducido. Los pasos incluyen:

- Distcp. Puede ser utilizado para copiar cualquier contenido del directorio de HBase en HDFS ya sea al mismo clúster en otro directorio, o para un clúster diferente. Sin embargo, esto puede dejar

los datos en un estado incoherente, por lo que debe ser evitado. Por otra parte, si no funciona se debe crear una instantánea de la tabla.

- Restaurar (si es necesario). La copia de seguridad del directorio de HBase HDFS se copia en el directorio "real" a través de HBase *distcp*. El acto de copiar estos archivos crea nuevos metadatos HDFS, por lo que una restauración del NameNode desde el momento de la copia de seguridad HBase no se requiere para este tipo de restauración, porque es una operación de restauración (a través de *distcp*) de un directorio específico de HDFS (es decir, la parte HBase), no de todo el sistema de archivos HDFS.

Copia de seguridad en un clúster en vivo. En este caso particular, cuando el clúster se encuentra en pleno funcionamiento se pueden llevar a cabo cuatro métodos para realizar las copias de seguridad, los mismos son:

- Replicación. La replicación es una manera de copiar datos entre las implementaciones de HBase. Puede servir como una solución de recuperación ante desastres y contribuir a proporcionar una mayor disponibilidad en la capa de HBase, además puede, de manera más práctica, servir como una forma de copiar fácilmente ediciones de una red orientada al clúster *MapReduce* que procesará los datos antiguos y nuevos y enviará de vuelta los resultados automáticamente. La replicación se realiza de forma asincrónica, lo que significa que los grupos pueden estar geográficamente distantes, los vínculos entre ellos pueden estar sin conexión durante algún tiempo y las filas insertadas en el clúster principal no estarán disponible al mismo tiempo en los grupos de esclavos (consistencia eventual).
- Replicación HDFS. Se puede simplemente subir el factor de replicación en HDFS y llamar a eso una copia de seguridad. Esto no puede luchar contra la corrupción de datos, pero protege contra ciertos fallos de hardware parcial.
- Copy Table. Es una utilidad sencilla de Apache HBase que se puede utilizar para copiar tablas individuales dentro de un clúster HBase o de un clúster HBase a otro. Utiliza el estándar HBase senda de exploración de lectura de interfaz para leer los registros de una tabla individual y los escribe en otra tabla (posiblemente en un clúster aparte).
- Export. Las tablas en HBase pueden exportarse usando la exportación de *MapReduce* (*org.apache.hadoop.HBase.mapreduce.Export*) a través de la misma se exportan los datos de la tabla en un archivo de secuencias de HDFS. Esta herramienta funciona en una tabla a la vez, en el

caso que se necesite realizar copia de seguridad de varias tablas, se ejecuta en cada tabla. Los datos exportados se pueden volver a importar a HBase por la herramienta Import.

2.2 Instalación y configuración de HBase

Para realizar el proceso de instalación y configuración de Hbase se utilizó la Guía de Referencia que aparece en el sitio oficial de la herramienta. En el **Anexo 4** se encuentra el url del sitio donde aparece la guía.

2.3 Pruebas

Las pruebas permiten ejecutar un sistema determinado bajo condiciones definidas. Permiten observar y registrar los resultados para posteriormente emitir una evaluación con la que se pueda validar dicho sistema. En esta sección se describen las pruebas que se le realizan a la herramienta HBase, propuesta como solución para la arquitectura de datos de la red social universitaria.

Con el fin de garantizar que HBase es una herramienta que ofrece alto rendimiento y gran capacidad de almacenamiento, se plantea como indicador a medir el tiempo de respuesta del sistema ante determinadas condiciones.

2.3.1 Tipos de pruebas

Con el objetivo de evaluar el rendimiento y la capacidad de almacenamiento de la herramienta HBase se decide ejecutar un conjunto de pruebas de desempeño (ver **Anexo 5**), que permiten determinar la respuesta, el rendimiento, la confiabilidad y la escalabilidad de un sistema bajo una carga de trabajo determinada. Entre las pruebas de desempeño se encuentran las pruebas de carga, estrés y volumen.

2.3.1.1 Pruebas de carga

El objetivo de las pruebas de carga es verificar el comportamiento de una aplicación bajo condiciones normales de carga y durante los picos de carga, que es cuando la aplicación llega al límite de su funcionamiento normal. Este tipo de prueba es realizada para verificar que la aplicación cumpla con los criterios de aceptación o compromisos de rendimiento. Las pruebas de carga permiten además medir los tiempos de respuesta, intervalos de rendimiento, los niveles de utilización de los recursos y la identificación del límite de funcionamiento, asumiendo que el punto de ruptura esté por encima del pico de carga.

2.3.1.2 Pruebas de estrés

Se realizan para determinar o validar el comportamiento de una aplicación cuando es sometida a condiciones de carga superiores a los límites aceptables, sobre condiciones normales y sobre los picos de carga. La meta de las pruebas de estrés es revelar los errores que solo aparecen bajo condiciones extremas de carga. Estos errores incluyen cuestiones de sincronización y fallas de memoria. Las pruebas de estrés ayudan a identificar los puntos débiles del sistema y su comportamiento bajo estas condiciones. Dentro de las pruebas de estrés se llevan a cabo las pruebas de picos cuyo objetivo es determinar o validar las características de desempeño del producto bajo prueba sujeto a modelos de carga de trabajo y volúmenes de carga que repetidamente aumentan más allá de las condiciones esperadas en operaciones de producción, se realizan por cortos intervalos de tiempo.

2.3.2 Herramientas

HBase posee un paquete de clases compiladas en Java o archivo *.jar* (*jar: HBase-*-tests.jar*) que permite realizar pruebas de rendimiento (carga, estrés y volumen) a la herramienta, dentro de este paquete se encuentra la clase *PerformanceEvaluation* (*class: org.apache.hadoop.HBase.PerformanceEvaluation*), utilizada para realizar las pruebas descritas en la sección anterior.

A través de la clase *PerformanceEvaluation s*e pueden hacer lecturas / escrituras a HBase. Se utiliza una estación de trabajo *MapReduce* para hacer lecturas / escrituras en paralelo, además se puede utilizar la opción de hacer las operaciones en hilos en vez de utilizar *MapReduce*.

Para ejecutar las pruebas que aparecen en # *HBase org.apache.hadoop.HBase.PerformanceEvaluation* se utiliza la siguiente secuencia de comandos:

```
java org.apache.hadoop.hbase.PerformanceEvaluation \
[--master=host:port] [--miniCluster] [--nomapred] [--rows=ROWS] <command> <nclients>
```

El primer parámetro *master=host:port* permite especificar la dirección *IP*[27] y el puerto de la máquina que se desea para que cumpla las funciones de máster, este parámetro posee carácter opcional y su valor por defecto es *localhost*.

Las opciones que se pueden utilizar son:

- *miniCluster:* permite ejecutar la prueba en un *HBaseMiniCluster.*

[27] Protocolo de Internet.

- *nomapred*: desactiva la opción de *MapReduce* que aparece por defecto permitiendo que se ejecuten las pruebas utilizando hilos.

El parámetro *rows=ROWS* posee carácter opcional, el mismo permite especificar la cantidad de filas que serán escritas o leídas para realizar la prueba, en caso de que no se especifique toma por defecto el valor de un millón.

Los parámetros <*command*> y <*nclients*> son de carácter obligatorio, se utilizan para indicar el tipo de prueba que se va a desarrollar y la cantidad de clientes respectivamente. Los comandos que se utilizan son:

- *randomRead*. Ejecuta la prueba aleatoria de lectura.
- *randomWrite*. Ejecuta la prueba aleatoria de escritura.
- *sequentialRead*. Ejecuta la prueba secuencial de lectura.
- *sequentialWrite*. Ejecuta la prueba secuencial de escritura.

2.3.3 Entorno de prueba

Para la ejecución de las pruebas se definió como ambiente de prueba un clúster compuesto por 4 máquinas virtuales las cuales poseen las características que aparecen en la Tabla **1**.

Tabla 1. **Descripción del entorno de prueba.**

Nombre	Función	RAM	Microprocesador	HDD	Sistema operativo
Nodo 1	Master	512 Mb	2.60 GHz	40 Gb	Ubuntu 10.10
Nodo 2	ZooKeeper	512 Mb	2.60 GHz	40 Gb	Ubuntu 10.10
Nodo 3	HRegion 1	512 Mb	2.60 GHz	40 Gb	Ubuntu 10.10
Nodo 4	HRegion 2	512 Mb	2.60 GHz	40 Gb	Ubuntu 10.10

Las máquinas virtuales se encuentran alojadas en un *host* físico, dicho hardware posee las siguientes características:

- microprocesador Intel Core i3-370 a 2.60 GHz
- memoria RAM de 4Gb
- disco duro de 500 Gb
- sistema operativo Ubuntu 12.04 LTS.

Las pruebas se realizan con varios usuarios concurrentes, simulando distintas cargas en cuanto a transacciones con el fin de analizar el rendimiento del sistema así como para establecer los límites de carga que el sistema es capaz de soportar.

2.3.3.1 Ejecución de pruebas de carga y estrés

Se simularon pruebas para cargas con diferentes configuraciones y valores de concurrencia de usuarios, a continuación se muestran los resultados que fueron obtenidos, teniendo en cuenta las métrica que aparece en la Tabla **2**.

Tabla 2. **Métricas de las pruebas.**

Métrica	
tr (ms)	Tiempo de respuesta en milisegundos.

En la Tabla **3** se describen cada una de las columnas que conforman la tabla donde aparecen los resultados de las pruebas de carga y estrés.

Tabla 3. **Descripción de los títulos de la tabla de resultados de las pruebas de carga y estrés.**

Nombre	Descripción
NUC	Número de usuarios concurrentes.
NR	Número de registros.
***randomRead* tr (ms)**	Tiempo de respuesta expresado en milisegundos de la prueba de tipo "Lectura aleatoria".
***randomWrite* tr (ms)**	Tiempo de respuesta expresado en milisegundos de la prueba de tipo "Escritura aleatoria".
***sequentialRead* tr (ms)**	Tiempo de respuesta expresado en milisegundos de la prueba de tipo "Lectura secuencial".
***sequentialWrite* tr (ms)**	Tiempo de respuesta expresado en milisegundos de la prueba de tipo "Escritura secuencial".

A continuación se muestran los resultados de las pruebas de carga y estrés. Se realizaron cada una de las pruebas descritas en el entorno para tres configuraciones, cambiando por cada una de las pruebas el número de usuarios concurrentes y la cantidad de registros (ver Tabla **4**).

Tabla 4. **Resultados de las pruebas de carga y estrés.**

NUC	NR	*randomRead* tr (ms)	*randomWrite* *tr (ms)*	*sequentialRead* *tr (ms)*	*sequentialWrite* *tr (ms)*
1	10	15	4	265	6
	100	145	11	487	13
	1000	1040	68	3159	60
50	10	2017	472	72422	236
	100	15164	2960	790331	2724
	1000	95327	8944	323566	8708
100	10	3187	941	395115	705
	100	18519	6312	1728719	6076
	1000	171479	18947	19025064	18711

Analizándose los tiempos de respuesta obtenidos en las pruebas se evidencia que, en clústeres de pocos nodos, las operaciones de escritura se realizan más rápido que las de lectura. El uso de los recursos del sistema se comportó de forma diferente en cada uno de los nodos, apreciándose que, en escenarios de alta concurrencia de usuarios: en el Nodo 1, donde se encuentra el *HMaster*, se aprecia un consumo moderado de los recursos; en el Nodo 2, donde se encuentra alojado el *ZooKeeper*, se pudo apreciar un alto consumo de la memoria *RAM*[28] siendo necesario el uso de la memoria *SWAP*[29] lo cual ralentiza los tiempos de respuesta del sistema; en el caso de los nodos 3 y 4, donde están instalados los *HRegion,* se observa un elevado uso de la memoria *RAM* y el microprocesador. Estos resultados permiten tomar decisiones a la hora de diseñar un esquema de despliegue para la herramienta propuesta, asignándole adecuadamente la cantidad de recursos en los nodos según su función. Aún cuando los tiempos de respuestas obtenidos no satisfacen las expectativas, debido a que las condiciones en las que se desarrollan las pruebas no se corresponden con el entorno de despliegue de las aplicaciones en los servidores de la UCI, se puede apreciar que el sistema es capaz de responder de manera estable ante las

[28] Por sus siglas en inglés *Random Access Memory*, memoria de acceso aleatorio.

[29] Memoria virtual ó memoria "de intercambio", no se trata de memoria RAM como tal, sino de una simulación funcional, esto significa que se crea un archivo de grandes dimensiones en el disco duro el cual almacena información simulando ser memoria RAM cuando esta se encuentra parcialmente llena, así se evita que se detengan los servicios de la computadora. http://www.informaticamoderna.com/Memoria_RAM.htm#swap

peticiones de 100 usuarios concurrentes ejecutando 100 operaciones de escritura en un tiempo inferior a 7 segundos.

Casos de éxito

Independientemente de que las pruebas no se desarrollaran en el ambiente adecuado para poder apreciar con claridad el rendimiento que aporta la herramienta HBase a los sistemas que la implementan, existe una amplia gama de compañías muy populares en el mundo que, por la gran cantidad de datos que almacenan y gestionan, utilizan y aprovechan las grandes prestaciones que posee. A continuación se mencionan algunas de estas compañías (en orden alfabético) y se describen las características de los clústeres en los que están desplegadas.

- **Adobe**. Utiliza Hadoop y HBase en varias áreas de los servicios sociales de almacenamiento estructurado de datos y el procesamiento para su uso interno. En la actualidad tiene un clúster con alrededor de 30 nodos HDFS funcionando, Hadoop y HBase en grupos de entre 5 y 14 nodos sobre la producción y el desarrollo. Tienen previsto un despliegue de un clúster de 80 nodos. Constantemente se escriben datos en HBase y se ejecutan trabajos de *MapReduce* para procesar y a continuación guardar de nuevo a los sistemas de HBase o externo. Su clúster de producción ha estado funcionando desde octubre de 2008.
- **Aguja** (análisis de datos de comercio electrónico). Utiliza Hadoop y HBase para analizar registro de búsqueda, ver los datos de productos y analizar todos sus registros. Tienen un clúster de 3 nodos con 48 núcleos en total, 4GB de RAM y 1 TB de almacenamiento cada uno.
- **Detikcom** (portal de noticias más grande de Indonesia). Utiliza Hadoop y HBase para analizar búsquedas de registros, generar más vistas de noticias y analizar todos sus registros. Actualmente utiliza 9 nodos en su clúster.
- **EBay**. Realiza un uso intensivo de HBase y Java *MapReduce* para la optimización de búsqueda y de investigación. Posee un clúster con 532 nodos (8 * 532 núcleos, 5.3Pb).
- **Facebook**. Utiliza Hadoop y HBase para almacenar copias de registros interno y fuentes estadísticas de datos con el fin de utilizarlo como una fuente para la presentación de informes o análisis y el aprendizaje de la máquina. Actualmente posee 2 grandes grupos: un grupo de 1100 máquinas con 8800 núcleos y alrededor del 12 Pb de almacenamiento en crudo y un grupo de 300 máquinas con 2400 núcleos y alrededor de 3 Pb de almacenamiento en crudo. Cada nodo (productos básicos) tiene 8 núcleos y 12 Tb de almacenamiento.

- **WorldLingo**. Posee 44 servidores (cada servidor tiene: 2 CPU de doble núcleo, 2 Tb de almacenamiento, 8 Gb de RAM). Cada servidor se ejecuta en una instancia Hadoop / HBase y en otra instancia con los servidores o aplicaciones web, dando 88 máquinas virtuales utilizables. Se corren dos grupos separados de Hadoop / HBase con 22 nodos cada uno. Hadoop se utiliza principalmente para ejecutar HBase y *MapReduce* sobre las tablas HBase para realizar tareas específicas. HBase se utiliza como almacenamiento de gama escalable y rápido regreso de millones de documentos. Actualmente almacenan 12 millones de documentos, con una meta de 450 millones en un futuro próximo.

Estos ejemplos demuestran las prestaciones que posee HBase en materia de almacenamiento de grandes cantidades de datos y de alto rendimiento. Convirtiéndose en una opción a tener en cuenta de carácter indispensable para la arquitectura de datos de la red social universitaria que se construye.

En el presente capítulo se describieron las particularidades de la herramienta HBase, sus características, arquitectura, instalación y seguridad permitiendo obtener los conocimientos y la documentación necesaria para guiar el proceso de instalación y configuración de la herramienta. Se diseñaron y ejecutaron las pruebas de rendimiento a la herramienta HBase obteniéndose resultados que facilitaron el análisis de las condiciones que debe poseer el ambiente en el que se despliegue la herramienta en la capa de persistencia de datos de la red social universitaria de la Universidad de las Ciencias Informáticas.

Conclusiones

- El análisis realizado a los modelos de base de datos existentes permitió identificar las bases para la posterior selección de la herramienta a utilizar en el almacenamiento de datos en la red social universitaria de la Universidad de las Ciencias Informáticas.
- Con la realización de la comparación entre los sistemas de base de datos de las redes sociales analizadas, se determinó HBase como la herramienta para la arquitectura de datos de la Red Social Universitaria.
- La elaboración de la propuesta de solución permitió caracterizar la herramienta HBase, conocer su arquitectura, mecanismos de manipulación de datos, seguridad y obtener la documentación necesaria para guiar el proceso de instalación y configuración de la herramienta facilitando su mantenimiento.
- Las pruebas de rendimiento aplicadas a la herramienta brindaron la posibilidad de analizar las características que debe tener un entorno de pruebas favorable así como el ambiente de despliegue de la herramienta en la UCI.

Recomendaciones

Se recomienda:

- El estudio de la herramienta Cassandra ya que la misma posee grandes prestaciones que pueden ser aprovechadas en futuros proyectos que formen parte de la red social universitaria de la Universidad de las Ciencias Informáticas.
- Realizar las pruebas de rendimiento a la herramienta en el ambiente de servidores de la UCI.
- Que se ponga en uso el sistema para aquellos servicios de la red social, incluyendo el núcleo que no requieran en su diseño un modelo relacional sino que las operaciones básicas que realice sean de inserción de registros como el caso de los logs.

Bibliografía referenciada

ABREU, L. M. T.; MIRANDA, I. E. R., *et al. Internet Relay Chat* [Consultado el: 05 de marzo de 2012]. Disponible en: http://www.cenco.sld.cu/node/38/#NORMAS%20PARA%20EL%20MEJOR%20USO%20DEL%20IRC.

BESEMBEL, I. y ROBERTS, S. Comparación entre métodos de acceso espaciales y de puntos multidimensionales. *V Jornadas Científico-Técnicas de la Facultad de Ingeniería, ULA. Mérida (Venezuela)*, 1998, nº [Consultado el: 10 de junio de 2012].

BIOGRAFÍADE. *Abraham Silberschatz* [Consultado el: 10 de junio de 2012]. Disponible en: http://www.biografias10.com/s/Abraham-Silberschatz/1/.

BLADE, L. *Tupla* [Consultado el: 10 de junio de 2012]. Disponible en: http://es.diveintopython.net/odbchelper_tuple.html.

CALDERÓN, Y. A. *NoSQL* [Consultado el: 26 de abril de 2012]. Disponible en: http://www.ecured.cu/index.php/NoSQL.

CARRION, G. A. Integracion de esquemas en bases de datos heterogeneas fuertemente acopladas. 1999, nº [Consultado el: 05 de marzo de 2012].

CASSANDRA, A. *Cassandra* [Consultado el: 6 de mayo de 2012]. Disponible en: http://cassandra.apache.org/.

CERI, S. y PELAGATTI, G. *Distributed databases: principles and systems.* McGraw-Hill, 1984. vol. 14,

CORPORATION, M. *¿Qué es SQL Server 2005?* Última actualización: 25 de Mayo de 2006. [Consultado el: 25 de abril de 2012]. Disponible en: http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx.

DATE, C. J. *Introducción a los Sistemas de Bases de Datos.* Pearson Publications Company, 2001. ISBN 9684444192.

DEAN, J. y GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 2008, vol. 51, nº 1, p. 107-113. [Consultado el: 10 de junio de 2012]. ISSN 0001-0782.

ECURED. *Netlog* [Consultado el: 08 de diciembre de 2011]. Disponible en: http://www.ecured.cu/index.php/Netlog.

ELLISON, N.; STEINFIELD, C., *et al. The benefits of Facebook "friends": social capital and college students use of online social network sites.* 2007, vol. 12, 1143-1168 p. Disponible en: http://jcmc.indiana.edu/vol12/issue4/ellison.html.

ENTERPRISEDB. *Postgres Plus Solution Pack v9.1 Release Notes* publicado el: 06 de febrero de 2012 de 2010, última actualización: 06 de febrero de 2012. 7 p.

ESPAÑOLA, R. A. D. L. L. *Base de datos* Madrid: [Consultado el: 23 de noviembre de 2011]. Disponible en: http://buscon.rae.es/draeI/.

ESPAÑOLA, R. A. D. L. L. *Dato* Madrid: [Consultado el: 25 de noviembre de 2011]. Disponible en: http://buscon.rae.es/draeI/.

EVANS, J. Fault Tolerance in Hadoop for Work Migration. 2011, nº [Consultado el: 25 de abril de 2012]. Disponible en: http://salsahpc.indiana.edu/b534projects/sites/default/files/public/0_Fault%20Tolerance%20in%20Hadoop%20for%20Work%20Migration_Evans,%20Jared%20Matthew.pdf.

FIELDING, R. y TAYLOR, R. N. *Representational State Transfer.* Ph. D. Thesis. UCI, 2000, [Consultado el: 11 de junio de 2012].

FOUNDATION, A. S. *Provides HBase Client* [Consultado el: 14 de junio de 2012]. Disponible en: http://HBase.apache.org/apidocs/org/apache/hadoop/HBase/client/package-summary.html.

FUNDATION, A. S. *HBase REST* [Consultado el: 11 de junio de 2012]. Disponible en: file:///C:/Documents20and20Settings/Administrator/Application20Data/Mozilla/Firefox/Profiles/fw6ad1c0.default/ScrapBook/data/20120611092040/index.html.

GARCÍA, K. R. y FERNÁNDEZ, E. E. B. *Sistema de réplica para la Intranet Corporativa y el Sitio en Internet de PDVSA.* Grado, Universidad de las Ciencias Informáticas, 2009.

GUERVÓS, J. J. M. y ESPARCIA, A. I. Usando bases de datos NoSQL para algoritmos evolutivos paralelos. 2012, nº [Consultado el: 25 de abril de 2012].

HBASE, A. *Apache HBase Reference Guide* [Consultado el: 8 de mayo de 2012]. Disponible en: http://HBase.apache.org/book.html.

HESMONDHALGH, D. Indie: The institutional politics and aesthetics of a popular music genre. *Cultural studies*, 1999, vol. 13, nº 1, p. 34-61. [Consultado el: 10 de junio de 2012]. ISSN 0950-2386.

IBM. *What is HBase?* [Consultado el: 18 de mayo de 2012]. Disponible en: http://www-01.ibm.com/software/data/infosphere/hadoop/HBase/.

KIMBALL, R. y ROSS, M. *The data warehouse toolkit: the complete guide to dimensional modeling.* Wiley, 2011. ISBN 1118082141.

LAPUENTE, M. J. L. Hipertexto: el nuevo concepto de documento en la cultura de la imagen. *http://www. hipertexto. info/documentos/web_tecnolog. htm. Consultado em*, 2007, vol. 19, nº p. 06-07.

[Consultado el: 10 de junio de 2012]. Disponible en: http://www.hipertexto.info/documentos/dom.htm.

LAWRENCE, D. R. *Base de datos Móviles* [Consultado el: 26 de marzo de 2012]. Disponible en: http://kuainasi.ciens.ucv.ve/bd_moviles/index.html.

LINOFF, G. S. y BERRY, M. J. *Data mining techniques: for marketing, sales, and customer relationship management.* * Wiley Computer Publishing, 2011. ISBN 111808750X.

MARCOS, E. y VELA, B. El proceso de creación de una base de datos web. *El profesional de la información*, 2002, vol. 11, nº 4, p. 248-255. [Consultado el: 12 de diciembre de 2011]. ISSN 1386-6710.

MEIER, J.; FARRE, C.*, et al. Performance testing guidance for web applications: patterns & practices.* Microsoft Press, 2007. ISBN 0735625700.

MEJıA, O. A. Computación en la nube. *ContactoS*, 2011, vol. 80, nº p. 45-52. [Consultado el: 26 de marzo de 2012].

MÉXICO, D. S. y LATINA, A. Redes sociales y su efecto en la vida diaria. 2011, nº [Consultado el: 05 de marzo de 2012]. Disponible en: http://vinculando.org/articulos/redes_sociales_y_su_efecto_en_la_vida_diaria.htmlutm_source=rss&utm_medium=rss&utm_campaign=redes_sociales_y_su_efecto_en_la_vida_diaria.

MSDN. *Realizar copias de seguridad y restaurar bases de datos en SQL Server.* [Consultado el: 12 de mayo de 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/ms187048%28v=sql.90%29.aspx.

MSDN. *Transacciones (motor de la base de datos).* [Consultado el: 12 de mayo de 2012]. Disponible en: http://msdn.microsoft.com/es-es/library/ms190612%28v=sql.90%29.aspx.

MUELA, T. M.; GÓMEZ, J. M. M.*, et al. Bases de datos deductivas y bases de datos difusas. Modelos Avanzados de Bases de Datos.* publicado el: 29 de noviembre de 2011 de 2009, última actualización: 29 de noviembre de 2011. 18 p.

MUÑOZ, V. A.; SANTAMARÍA, Á. E.*, et al. Modelos de Bases de Datos Orientadas a Objetos y Bases de Datos Objeto-Relacionales.* Universidad de Castilla-la Mancha, 2008, [Consultado el: 26 de noviembre de 2011].

MYSPACE. *MySpace* [Consultado el: 20 de marzo de 2012]. Disponible en: http://developer.myspace.com/.

ORACLE. *Java Management Extensions (JMX) Technology* [Consultado el: 03 de junio de 2012]. Disponible en: http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html.

ORALLO, J. H. *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas. .* Valencia: publicado el: 13 de diciembre de 2011 de 2002, última actualización: 13 de diciembre de 2011. 35 p.

PALACIOS, C. A. *Redes sociales* [Consultado el: 08 de diciembre de 2011]. Disponible en: http://www.ecured.cu/index.php/Redes_sociales.

PÉREZ, A. J. P. *Facebook* [Consultado el: 08 de diciembre de 2011]. Disponible en: http://www.ecured.cu/index.php/Facebook#Servicios_que_ofrece.

RABELO, R. J.; AFSARMANESH, H.*, et al.* Applying Federated Databases to Inter-Organizational Multi-agent Scheduling. En *1999*.

RAGGETT, D.; LE HORS, A.*, et al.* HTML 4.01 Specification. *W3C recommendation*, 1999, vol. 24, nº [Consultado el: 10 de junio de 2012].

ROB, P. y CORONEL, C. *Sistemas de bases de datos: diseño, implementación y administración.* Cengage Learning Mexico, 2003. ISBN 9706862862.

ROBIE, J. *XQL (XML Query Language)* [Consultado el: 10 de junio de 2012]. Disponible en: http://metalab.unc.edu/xql/xql-proposal.xml

ROS-MARTÍN, M. Evolución de los servicios de redes sociales en Internet. *El profesional de la información*, 2009, vol. 18, nº 5, p. 552-558. [Consultado el: 05 de marzo de 2012]. ISSN 1386-6710.

ROWLAND, E. *The Decline of the "Webmaster".* [Consultado el: 07 de febrero de 2012]. Disponible en: http://www.clickfire.com/the-decline-of-the-webmaster/.

RUIZ, M. *Introducción a los Sistemas de Base de Datos* [Consultado el: 07 de febrero de 2012]. Disponible en: http://www.ecured.cu/index.php/Bases_de_datos.

SÁEZ, B. S. *Twitter* Cienfuegos: [Consultado el: 9 de diciembre de 2011]. Disponible en: http://www.ecured.cu/index.php/Twitter#Historia.

SERVER, S. O. S. S. *Sponsored and Custom development* [Consultado el: 05 de febrero de 2012]. Disponible en: http://sphinxsearch.com/services/development/.

SHETH, A. P. y LARSON, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 1990, vol. 22, nº 3, p. 183-236. [Consultado el: 12 de diciembre de 2011]. ISSN 0360-0300.

SILBERSCHATZ, A.; KORTH, H. F.*, et al. Fundamentos de bases de datos.* 4ta ed. Madrid: Concepción Fernández Madrid, Susana Santos Prieto, 2002. 787 p. ISBN 0-07-228363-7.

SOARES, I. O. A Formação do Educomunicador: 15 anos na busca de uma mais profunda relação entre o profissional da comunicação/educação eo mundo das crianças e dos adolescentes. 2005, nº [Consultado el: 13 de diciembre de 2011].

SQL.ORG. *SQL* [Consultado el: 10 de junio de 2012]. Disponible en: http://www.sql.org/.

SUBRAHMANIAN, V. y SCHULZRINNE, H. T1-Principles of Multimedia Database Systems. 1998, nº [Consultado el: 05 de noviembre de 2011].

THRIFT, A. *Thrift* [Consultado el: 04 de junio de 2012]. Disponible en: http://thrift.apache.org/.

TWITTER. *Introducing FlockDB* [Consultado el: 05 de febrero de 2012]. Disponible en: http://engineering.twitter.com/2010/05/introducing-flockdb.html.

WEB, S. Web 2.0. *Sborník příspěvků*, 2007, nº p. 19. [Consultado el: 13 de diciembre de 2012].

WHITE, T. *Hadoop: The definitive guide.* Yahoo Press, 2010. 526 p. ISBN 1449389732.

Bibliografía consultada

BAIGET, T. Aspectos psico-sociológicos del uso de Internet. *El profesional de la información.*, 2011, vol. 20, nº 1, p. 87-93. [Consultado el: 05 de marzo de 2012]. ISSN 1386-6710.

BOJA, C. y POCOVNICU, A. Distributed parallel architecture for storing and processing large datasets. En *2012*. p. 125-130.

CHARTE, F. Sql Server 2005. 2006, nº [Consultado el: 25 de abril de 2012]. ISSN 8441520283.

GUZMÁN FERNÁNDEZ, I.; CASTILLO FIGUEROA, R.*, et al.* Propuesta de arquitectura de una herramienta web para la administración del gestor PostgreSQL. *Revista Cubana de Ciencias Informáticas*, 2011, vol. 5, nº 1, ISSN 1994-1536.

LI, J.; SINGHAL, S.*, et al. Managing Data Retention Policies at Scale* [Consultado el: 10 de junio de 2012]. Disponible en: http://www.hpl.hp.com/techreports/2010/HPL-2010-203.pdf.

MCGLOTHLIN, J. P. y KHAN, L. Scalable queries for large datasets using cloud computing: a case study. En *2011*. p. 8-16.

WIKI, H. PoweredBy- Hadoop [Consultado el: 10 de junio de 2012]. Disponible en: http://wiki.apache.org/hadoop/PoweredBy.

Anexos

Anexo 1

En la siguiente página aparece un ejemplo de lo que puede ser un cliente sencillo utilizando la API de Java, en este ejemplo se asume que se ha creado una tabla llamada " *myTable* " con una familia de columna llamada "*myColumnFamily*".

http://HBase.apache.org/apidocs/org/apache/hadoop/HBase/client/package-summary.html

Anexo 2

En las siguientes páginas aparece la información referente a cómo se utiliza el marco de trabajo Thrift en HBase.

http://wiki.apache.org/hadoop/HBase/ThriftApi

file:///C:/Documents%20and%20Settings/Administrator/Application%20Data/Mozilla/Firefox/Profiles/fw6ad1c0.default/ScrapBook/data/20120611092028/index.html

Anexo 3

Para obtener más información acerca de las particularidades del uso de HBase REST visitar en el sitio de Apache Software Foundation la sección dedicada al tema, la misma se encuentra en la siguiente dirección:

http://wiki.apache.org/hadoop/HBase/HBaseRest

Anexo 4

En el siguiente link aparece el manual de Instalación y configuración de Hbase que ofrece el sitio oficial de la herramienta.

http://hbase.apache.org/book.html

Anexo 5

Las pruebas de desempeño se llevan a cabo para:

- Evaluar la disposición para la producción,
- evaluar a partir de los criterios de desempeño,
- comparar las características de desempeño de múltiples sistemas o configuraciones,
- identificar la fuente de los problemas de desempeño,
- brindar soporte para el ajuste del sistema en función del desempeño,

- encontrar niveles de rendimiento.

Las pruebas de desempeño constan de las siguientes actividades:

Actividad 1. Identificar el entorno de pruebas. Identificar el entorno físico de pruebas y el entorno de producción así como las herramientas y recursos disponibles a usar por el equipo de prueba. El entorno físico incluye hardware, software y las configuraciones de red. Una vez lograda la comprensión total del entorno de pruebas esto permitirá una mayor eficiencia en el diseño y planificación de las pruebas lo que ayudará a identificar los objetivos de las pruebas tempranamente. En algunas situaciones este proceso debe ser revisado periódicamente a través del ciclo de vida del proyecto.

Actividad 2. Identificar criterios de aceptación de desempeño. Identificar el tiempo de respuesta, el rendimiento y los recursos utilizados esperados, así como sus limitaciones. En general, el tiempo de respuesta es una preocupación de los usuarios, el rendimiento es una preocupación del negocio y la utilización de los recursos es una preocupación del sistema. Adicionalmente, identificar criterios de éxito del proyecto que no deben ser capturados por estas metas y limitaciones; por ejemplo, utilizando pruebas de desempeño para evaluar cual combinación de los parámetros de configuración obteniendo las características de desempeño más eficientes.

Actividad 3. Planificar y diseñar las pruebas. Identificar escenarios claves, determinar la variabilidad entre los usuarios representativos y cómo simular dicha variabilidad, definir los datos de las pruebas y establecer las métricas a recolectar. Consolidar esta información en uno o más modelos de uso del sistema a ser implementado, ejecutado y analizado.

Actividad 4. Configurar el entorno de pruebas. Preparar el entorno de pruebas, herramientas y recursos necesarios para ejecutar cada estrategia, así como que sus características y componentes estén disponibles para las pruebas. Asegurar que el entorno de prueba esté preparado para monitorear los recursos necesarios.

Actividad 5. Implementar el diseño de las pruebas. Desarrollar las pruebas de desempeño de acuerdo con el diseño de las pruebas.

Actividad 6. Ejecutar las pruebas. Ejecutar y monitorear las pruebas. Validar las pruebas, los datos de las pruebas y la recolección de los resultados. Ejecutar las pruebas validadas para el análisis mientras se monitorean las pruebas y el entorno de pruebas.

Actividad 7. Analizar resultados, reportes y volver a ejecutar las pruebas. Consolidar y compartir los datos resultantes. Analizar los datos individualmente y en equipo. Darle nueva prioridad a las pruebas no

ejecutadas y ejecutarlas nuevamente cuantas veces se necesite. Cuando los resultados obtenidos expresados en métricas coinciden con los criterios de aceptación, cuando ninguno de los objetivos han sido violados y toda la información deseada ha sido recopilada, entonces se han concluido las pruebas para un escenario particular con una configuración específica.