UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

"ESTUDIO DE NUEVAS TECNOLOGÍAS DE GESTIÓN DE BASES DE DATOS NOSQL PARA EL DESARROLLO DE APLICACIONES WEB 2.0."

APLICATIVO:

"SISTEMA PROTOTIPO PARA EL CONTROL DE CAMPEONATOS Y JUGADORES DESARROLLADO PARA LA LIGA DEPORTIVA PARROQUIAL SAN PABLO DEL LAGO"

AUTOR: INUCA MORALES MARCO GEOVANY

DIRECTOR: ING. JOSÉ LUIS RODRÍGUEZ

IBARRA – ECUADOR

2015

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1 IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital Institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1002872602		
APELLIDOS Y NOMBRES:	INUCA MORALES MARCO GEOVANY		
DIRECCIÓN:	OTAVALO – SAN PABLO DEL LAGO– CALLE COLON Y AYORA		
EMAIL:	geomarkito@gmail.com		
TELÉFONO FIJO:	062918748	**TELÉFONO MÓVIL:**	0997141779
DATOS DE LA OBRA			
TÍTULO:	“ESTUDIO DE NUEVAS TECNOLOGÍAS DE GESTIÓN DE BASES DE DATOS NOSQL PARA EL DESARROLLO DE APLICACIONES WEB 2.0”		
AUTORA:	INUCA MORALES MARCO GEOVANY		
FECHA:	DICIEMBRE DEL 2015		
PROGRAMA:	☐ PREGRADO	☐ POSTGRADO	
TÍTULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES		
DIRECTOR:	ING. JOSÉ LUIS RODRÍGUEZ		

2AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, INUCA MORALES MARCO GEOVANY, con cedula de identidad Nro. 1002872602, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 144.

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

UNIVERSIDAD TÉCNICA DEL NORTE

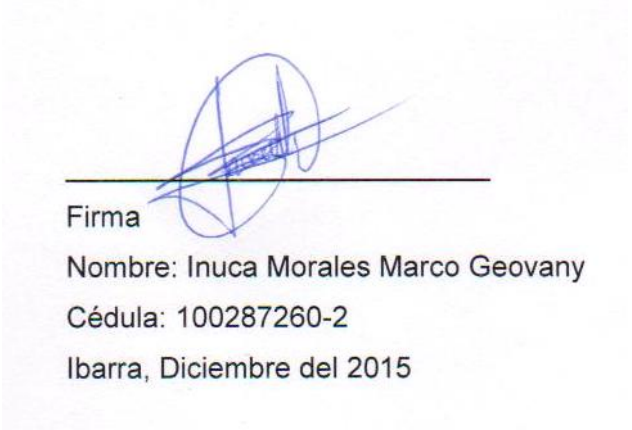
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, **INUCA MORALES MARCO GEOVANY**, con cedula de identidad Nro. 100287260-2, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, articulo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: "**ESTUDIO DE NUEVAS TECNOLOGÍAS DE GESTIÓN DE BASES DE DATOS NOSQL PARA EL DESARROLLO DE APLICACIONES WEB 2.0**" que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte



UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

Certifico que el proyecto de Trabajo de Grado : “ESTUDIO DE NUEVAS TECNOLOGÍAS DE GESTIÓN DE BASES DE DATOS NOSQL PARA EL DESARROLLO DE APLICACIONES WEB 2.0”ha sido realizada en su totalidad por el señor: Inuca Morales Marco Geovanyportador de la cédula de identidad número: 100287260-2

ING. JOSÉ LUIS RODRÍGUEZ
DIRECTOR DE TESIS

CERTIFICACIÓN

Ibarra, 8 de Julio del 2013

Señores

LIGA DEPORTIVA PARROQUIAL SAN PABLO DEL LAGO

Presente

De mis consideraciones.

Siendo auspiciantes del proyecto de tesis del Egresado **INUCA MORALES MARCO GEOVANY** con CI: **1002872602** quien desarrolló su trabajo con el tema "**ESTUDIO DE NUEVAS TECNOLOGÍAS DE GESTIÓN DE BASES DE DATOS NOSQL PARA EL DESARROLLO DE APLICACIONES WEB 2.0.**", con el aplicativo **"SISTEMA PROTOTIPO PARA EL CONTROL DE CAMPEONATOS Y JUGADORES DESARROLLADO PARA LA LIGA DEPORTIVA PARROQUIAL SAN PABLO DEL LAGO"**, me es grato informar que se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte del egresado **INUCA MORALES MARCO GEOVANY**. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa/institución.

El egresado **INUCA MORALES MARCO GEOVANY** puede hacer uso de este documento para los fines pertinentes en la Universidad Técnica del Norte.

Atentamente

Sr. Fernando Chiza
PRESIDENTE DE LIGA

AGRADECIMIENTO

El más profundo y sincero agradecimiento a DIOS que esta sobre todas las cosas, puesto que no me ha desamparado en ningún momento, también agradezco a mi prestigiosa Universidad Técnica Del Norte quien me abrió las puertas a los conocimientos que tanto he añorado, preparándome para un futuro competitivo y formándome con una gran calidad de persona.

A mis maestros por impartirme sus conocimientos y ser grandes consejeros y amigos para ser un excelente profesional, a mis amigos porque siempre conté con una mano amiga cuando más los necesitaba.

Agradezco también a mis padres pues supieron ser grandes amigos y consejeros en este arduo camino para alcanzar mis metas, gracias al apoyo y paciencia que me han sabido brindar para culminar mi carrera.

A todos mi mayor reconocimiento y gratitud.

Marco Geovany Inuca Morales

RESUMEN

La información es el recurso más importante de grandes Empresas e Instituciones en el mundo actual. Avances tecnológicos y científicos se han logrado con la ayuda de un manejo adecuado de la información, combinado con una amplia investigación, logrando resultados fantásticos.

Es por esto, que la presente investigación, propone las mejores alternativas para promover y aplicar una correcta manipulación de la tecnología de gestión de bases de datos NoSQL, al momento de la creación de una aplicación web 2.0 y así poder dar soluciones que permitan a los usuarios y profesionales una correcta utilización de estas herramientas.

NoSQL tiene una amplia variedad de diferentes tecnologías de bases de datos las cuales se desarrollan en base al aumento de gran cantidad de datos ya sean estos de información de los usuarios, productos, la frecuencia con que se acceden a estos datos y en base a las necesidades de procesamiento y rendimiento. Bases de datos relacionales, por otro lado, no fueron diseñadas para hacer frente a la escala y agilidad los retos que enfrentan las aplicaciones modernas, ni fueron construidos para aprovechar el almacenamiento barato y potencia de procesamiento disponible.

En comparación con las bases de datos relacionales, las bases de datos NoSQL son más escalables y ofrecen un rendimiento superior, y su modelo de datos aborda varias cuestiones que el modelo relacional no está diseñado para hacer frente a:

- ✓ Los grandes volúmenes de datos estructurados, semi-estructurados y no estructurados
- ✓ Sprints ágiles, iteración rápida y empujones código frecuente
- ✓ Programación orientada a objetos que es fácil de usar y flexible
- ✓ La arquitectura eficiente de escalado horizontal en lugar de caros, arquitectura monolítica

Las Bases de datos NoSQL se construyen para permitir la inserción de datos sin un esquema predefinido .Eso hace que sea fácil de hacer cambios en las aplicaciones importantes, en tiempo real, sin tener que preocuparse por las interrupciones del servicio lo que significa que el desarrollo es más rápido, integración de código es más fiable, y se necesita menos tiempo del administrador de base de datos.

SUMMARY

Information is the most important resource of large companies and institutions in the world today. Technological and scientific advances have been achieved with the help of proper management of information, combined with extensive research, achieving fantastic results.

It is for this reason that this research proposes the best ways to promote and implement the proper handling of the technology of database management NOSQL data when creating a web 2.0 application and be able to provide solutions that enable professional users and the correct use of these tools.

NoSQL has a wide variety of different database technologies which are developed based on the increasing wealth of data consisting of either user information, products, frequency of access to these data and based on processing and performance needs. Relational databases, on the other hand, were not designed to cope with the scale and agility challenges facing modern applications, and were built to take advantage of the cheap storage and processing power available.

Compared to relational databases, NoSQL data are more scalable and offer superior performance and its data model addresses several issues that the relational model is not designed to deal with:

- ✓ Large volumes of structured data, semi - structured and unstructured
- ✓ Agile Sprints, rapid iteration and shoving frequent code
- ✓ Object-oriented programming that is easy to use and flexible
- ✓ The scale- efficient architecture instead of expensive, monolithic architecture

The NoSQL data bases are constructed to allow insertion without a predefined database schema. That makes it easy to make changes to important applications in real-time, without having to worry about outages which means that the development is faster, code integration is more reliable, and less time database administrator is needed.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

1 INTRODUCCIÓN

En la actualidad se vive en un mundo en el que la tecnología se encuentra muy desarrollada, esto ha cambiado la forma de pensamiento y de vida de muchas personas, gran cantidad de Empresas e Instituciones se han visto en la necesidad de introducirse en el mundo de la informática para brindar un mejor servicio a la comunidad, en vista a la creciente necesidad de representar información, manejarla, explotarla y compartirla.

Ante esto el inconveniente que se presenta es que la información que se manipula en Instituciones, Empresas, Organizaciones que crecen a pasos agigantados y las bases de datos Normales Relacionales, no cumplen con los requerimientos para satisfacer las necesidades de estos, esto se debe a la gran capacidad de acceso a la información.

Es por esta razón, que se han visto en la necesidad de investigar y analizar la tecnología de Gestión de Bases de Datos NoSQL[1] en sus clasificaciones más importantes las cuales permiten a los usuarios trabajar con grandes volúmenes de datos con mayor rapidez.

La taxonomíade las bases de datos NoSQL entre las principales por su implementación que son:

- ✓ Bases de datos key-value.
- ✓ Bases de datos orientadas a columnas.
- ✓ Bases de datos orientadas a documentos.
- ✓ Bases de datos orientadas a grafos.

1.1 ANTECEDENTES

Las bases de datos son conjunto de datos pertenecientes al mismo contexto, cada base de datos se compone de una o más tablas que guarda un conjunto de datos.

[1] **NOSQL:** No Only SQL

NoSQL es un término usado en informática para aglomerar una serie de bases de datos no relacionales que no proporcionan garantías ACID,es decir (permiten el conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción). Normalmente no tienen esquemas fijos de tablas ni sentencias "join", el cual permite combinar registros de dos o más tablas en una base de datos relacional. En el Lenguaje de Consultas Estructurado (SQL) hay tres tipos de JOIN: interno, externo y cruzado.

Es interesante destacar que las NOSQL existen desde hace muchos años (1970), con experimentos en universidades europeas y norteamericanas, sobre sistemas de almacenamiento abiertos. Los almacenamientos de par Clave/Valor, vieron la luz en 1979 con el proyecto DBM, y más tarde BerkeleyDB(1986). Si consideramos que LotusDB es una base de datos orientada a documentos, que la empresa Lotus dio a luz este proyecto en 1989.

Las características principales de una base de datos NoSQL responden a:

a) **Modelos de datos variables y flexibles:** Siempre hay un modelo, una estructura, pero la rigidez del modelo relacional desaparece

b) **Escalabilidad sencilla**, pueden responder a necesidades pequeñas y a altos volúmenes de trabajo

c) **Alta velocidad de respuesta a peticiones**. Se tiene un menor tiempo de respuestas al realizar las consultas a las bases NOSQL

d) **Diferentes lenguajes de consulta (no SQL).** Las diferentes bases de datos NOSQLtienen su propio lenguaje de consulta.

1.2 SITUACIÓN ACTUAL

En los últimos años, principalmente por los incrementos de las redes sociales como Facebook y los grandes sistemas distribuidos como Google App Engine, existen varios problemas con los RDBMS[2]actuales que pueden suponer una seria limitante para la construcción de aplicaciones

[2]**RDBMS**: Un sistema de gestión de bases de datos relacionales es aquel que sigue el modelo relacional. **Fuente**: http://es.wikipedia.org/wiki/RDBMS

Si bien la mayoría de las aplicaciones se centran mucho en las características del modelo relacional, para otras aplicaciones (las cuales se alejen un poco del esquema típico) se empiecen a notar distintos problemas inherentes al modelo. Estos problemas son en gran medida el motivo por el que surgió el NOSQL.

Los problemas que se presenta en la actualidad son:

- **Leer datos es costoso.-** En el modelo relacional los datos se representan mediante conjuntos (tablas) relacionados entre sí. Realizar una consulta al modelo relacional (como todos sabemos), implica juntar grandes conjuntos de datos con operaciones algebraicas (como el producto cartesiano), y luego filtrar todo el conjunto resultante lo cual lleva una gran complejidad computacional**.**

- **Transaccionalidad innecesaria.-** El objetivo de las transacciones es asegurar la integridad de los datos, aunque con este esquema se dejen de lado otros aspectos como la performance.

- **Escalabilidad.-** El gran problema de las bases relacionales es la escalabilidad. Estas fueron pensadas para correr en un solo servidor con mucha potencia, como mucho tener replicaciones y balanceo de carga.

- **Representación del modelo en una RDBMS.-** Si bien es posible representar la mayoría de modelos usando el modelo relacional, no siempre resulta la mejor opción.

Actualmente la mayoría programamos en el paradigma orientado a objetos, lo que ya de entrada involucra el problema de traducir los objetos a un modelo relacional. Existen herramientas que permiten convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia llamadasORM[3]que ayudan a facilitar esta traducción por lo cual es necesario responder estas preguntas.

¿Por qué es necesario siempre un mapeo entre la aplicación y la base?

Así evitarías o disminuirías las funciones que son innecesarias para el buen funcionamiento de los sistemas.

[3]**ORM**: Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping.

¿Por qué no podríamos persistir nuestra aplicación en su paradigma nativo?

En la carrera de Ingeniería en Sistemas actualmente no se topa el tema de base de datos NOSQL dentro del área como parte del pensum académico, es por eso que este tema de tesis es para dar a conocer la existencia de estas bases de datos que desde su origen a la actualidad han venido tomando fuerza en la época tecnológica que hoy en día se vive.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Realizar el estudio de las herramientas para la gestión de base de datos NoSQL y el desarrollo de aplicaciones web 2.0 para determinar sus ventajas y desventajas respecto a las bases de datos relacionales.

1.3.2 OBJETIVOS ESPECÍFICOS

- ✓ Conocer las ventajas y desventajas que ofrecen los gestores bases de datos NoSQL.
- ✓ Aprender los conceptos de la NoSQL.
- ✓ Estudiar la estructura de las bases de datos NoSQL
- ✓ Investigar las herramientas de gestión de Bases de Datos NoSQL.
- ✓ Definir un método para desarrollo de aplicaciones basadas en bases de datos NoSQL.
- ✓ Establecer en qué casos la utilización de las bases de datos NoSQL es ventajoso respecto a las bases de datos relacionales.
- ✓ Diseñar e implementar una aplicación web para la gestión de información de la Liga de San Pablo.

1.4 JUSTIFICACIÓN

El presente trabajo investigativo pretende contribuir a mitigar las necesidades en el manejo de la información y manipulación de los sistemas de gestión de bases de datos NOSQL, con un enfoque totalmente práctico pero dejando un marco teórico que garantice la construcción de modelos más robustos o soluciones alternas.

Hoy en día la tecnología en cuanto a NoSQL a tomado tanta fuerza que se han convertido en una buena opción, pero de ahí a realizar una recomendación de cambiar de una solución de bases de datos relacionales a una base de datos NoSQL con el fin de reducir costos o solucionar problemas de escalabilidad, Transaccionalidad innecesaria, representación del modelo, a los usuarios es de pensar, teniendo en cuenta que el activo más importantes para una organización es la información y se corre el riesgo de perder la credibilidad como profesional, de ahí nace la importancia de diseñar, ejecutar, valorar y analizar pruebas que soporten las posibles recomendaciones que se pueden dar con respecto a ventajas que posee un software de bases de datos NOSQL ya que el crear una aplicación ya depende la políticas y reglas que tenga la empresa o exigencias de los usuarios.

Existen varias posibilidades que ofrece el mercado en cuanto a software libre, pero para este estudio se han seleccionado las herramientas que han tenido mayor difusión, auge, teniendo presente la disponibilidad de utilidades que permitan a los usuarios una mejora de sus procesos diarios, y presentarse como una alternativa viable y de calidad.

En este caso me ha llamado la atención MongoDB. Esta base de datos realizada en C++, que cuenta, entre sus principales características con ser una base de datos de esquema libre, de alto rendimiento y orientada a documentos.

Las herramientas que utilizaremos para el desarrollo de la aplicación son las siguientes:

- ✓ Apache para Servidor de Aplicaciones
- ✓ Netbeans IDE como herramienta de desarrollo.
- ✓ Framework Symfony.
- ✓ HTML, CSS, javascript, estándares abiertos.
- ✓ MongoDB como servidor de base de datos NoSQL.

A medida que se vaya desarrollando la tesis se podrán incorporar herramientas que se crean necesarias para la implementación del aplicativo.

1.5 ARQUITECTURA DEL SISTEMA

Esta es la arquitectura básica de desarrollo y separa sus funcionalidades en tres capas diferenciando claramente la lógica de desarrollo, estas capas son lógica de interfaz de usuario, controlador de interacción, lógica de negocio.

Las capas hacen referencia a la forma como la aplicación es segmentada desde el punto de vista lógico. Los niveles corresponden a la forma en que las capas lógicas se encuentran distribuidas de forma física. Esta lógica se basa en el patrón de diseño MVC (Modelo Vista Controlador).

FIGURA 1: Arquitectura - Modelo Vista Controlador (MVC)

1.6 ALCANCE

1.6.1 ALCANCE DE LA INVESTIGACIÓN

Realizar un estudio de las características, funciones ventajas y desventajas que tienen las bases de datos NoSQL, para poder saber con mayor seguridad el por qué debemos de utilizar estas herramientas como una alternativa de desarrollo para las aplicaciones Web, teniendo en cuenta que las cuatro clasificaciones de bases de datos NOSQLque se investigara.

Lo que se quiere demostrar con esta investigación es la capacidad, funcionamiento, escalabilidad, velocidad de consultas en lectura de datos y compatibilidad que tiene cada una de las clasificaciones antes mencionadas con el fin de dar a conocer a estudiantes de la FICA la existencia de esta herramienta con una alternativa de desarrollo para las Aplicaciones Web 2.0 que hoy en día han tomado fuerza.

Realizaremos un estudio de una base de datos NoSQL por cada tipo para así poder realizar una recomendación de estas tecnologías para el desarrollo de las Aplicaciones web. Con lo cual se demostrara sus características, ventajas y desventajas desarrollando una Aplicación con cada base de datos NoSQL que mejor nos convenga.

1.6.2 ALCANCE DE LA APLICACIÓN

Se realizará una aplicación prototipo para las pruebas y posterior ejecución a los manejadores de datos con el fin de conocer:

- Capacidad para soportar altos volúmenes de datos
- Integridad en los datos
- Concurrencia
- Velocidad
- Sistemas de Backup y Restauración
- En el desarrollo de este trabajo no se incluye análisis de costos con respecto a capacitación, asesoría.
- Se darán a conocer nombres de firmas que prestan soporte y capacitación para la administración del manejador de datos bajo licenciamiento libre

El sistema prototipo se lo realizara para la Liga Deportiva Parroquial San Pablo del Lago, con el fin de mejorar el desempeño de la misma.

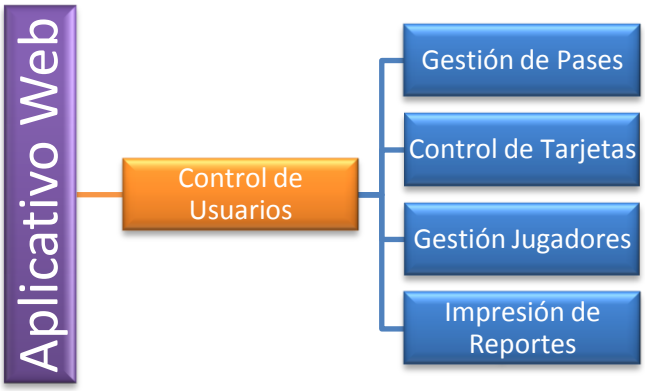


FIGURA 2: Modulo – Módulos de Vista Controlador

Módulo de control de Usuarios: Estará encargado de registrar los datos de cada uno de los usuarios de los equipos pertenecientes a la Liga con el fin de tener un mayor control en la aplicación de sanciones impresión de documentos

Sub Módulo de Gestión de Pases: Estará encargado de realizar el documento respectivo para realizar el pase de cada jugador de equipo a equipo

Sub Módulo De Control de Tarjetas: Este permitirá realizar un control de tarjetas de cada jugador en cada equipo con el fin de poder aplicar las sanciones respectivas de acuerdo con el reglamento de la Liga.

Sub Módulo Gestión Jugadores: Este permitirá realizar un documento tipo carnet para cada usuario registrado en su respectivo equipo.

Sub Módulo Reportes: El generador de reportes será un sub módulo que permitirá saber en tiempo real el comportamiento de cada una de las partes del sistema, actividades realizadas por los usuarios, movimiento de productos, proveedores, clientes, etc. El sistema permitirá la generación de los siguientes informes:

- Listado de Equipos
- Impresión de carnets
- Impresión de pases

CAPÍTULO II

2 DEFINICIÓN DE LAS BASES DE DATOS

(LITH & MATTSSON, 2010).Dice que el término base de datos se utiliza para describir una variedad de sistemas empleados para organizar el almacenamiento de los datos. Existen diferentes tipos de sistemas, así como los modelos que hacen esto de diferentes maneras, y aquí tratamos de examinar las diferencias fundamentales de los más prominentes.

En primer lugar explicaremos los conceptos generales de las bases de datos NoSQL en sus diferentes clasificaciones, que se pueden aplicar en una variedad de escenariosen informática, NoSQL tiene una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado que no usan SQL como el principal lenguaje de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, coherencia, aislamiento y durabilidad), y normalmente estas bases de datos escalan bien horizontalmente.

Los investigadores académicos describen a este tipo de bases de datos como almacenamiento estructurado, término que abarca también las bases de datos relacionales clásicas. A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y Bases de datos orientadas a grafos.

Los sistemas de bases de datos NoSQLempezaron su crecimientojunto con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas tenían que afrontar desafíos con el tratamiento de datos que las tradicionales las bases de datos (RDBMS) que no solucionan. Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares. Estas compañías se dieron cuenta que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.

En ese sentido, a menudo, las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad que almacenar los registros (p.ej. almacenamiento clave-valor). La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

En una breve definición de bases de datos NoSQL, es que es literalmenteuna combinación dedos palabras: No ySQL. La implicación es queNoSQLes una tecnologíao productode los contadores deSQL Server.Los creadoresy pionerosdela palabra de modaNoSQLprobablementequería decir que NORDBMSo No relacional, perose enamoraronpor elmejorsonidoNoSQLypegado a ella.A su debido tiempo, algunos han propuesto una alternativa a laNonRelasNoSQL.

Algunos otroshan tratado derescatarel término original, proponiendo que NOSQLesen realidad un acrónimoque se expande a"No sóloSQL." Cualquiera que sea elsentido literal, NOSQLse utiliza hoy en día comoun término general paratodas lasbases de datos que no siguenlos principiosde RDBMSpopularesbien estableciday, a menudose relacionan congrandes conjuntos de datos, acceso y la manipulacióna escala Web. EstosignificaNOSQLno esun solo producto, o incluso una sola tecnología.Representauna clase de productosy una colección dediversos conceptos relacionados, a veces, sobre el almacenamientoy manipulación de datos.

2.1 HISTORIA DE NOSQL

Antes de empezarcon los detallessobre los tipos deNoSQLy los conceptosinvolucrados, es importante establecer el contexto en quesurgióNoSQL. Las bases de datos no relacionales no son nuevas. De hecho, las primeras bases de datos no relacionales de almacenamiento, aparecen en el tiempocuandoel primer grupo demáquinas de computaciónfueron inventadas.

Las bases de datos no relacionales noprosperaron por la llegada delos mainframesy han existidoen laespecializaday los dominiosespecíficospor ejemplo,directorios jerárquicospara la autenticación dealmacenamiento ylas credenciales de autorización a través de los años.Sin embargo,las bases de datosno relacionalesque han aparecido enel mundo deNoSQLesuna nueva encarnación, que nacieronen el mundo deforma masiva aplicaciones escalablesde Internet.Estas bases de datos no relacionalesNoSQL, en su mayor parte, fueron concebidosen el mundode la computación distribuiday paralela.

Comenzando conInktomi[4], que podría ser considerado verdadera mente y en primer lugar como elmotor de búsqueda,y que culminócon Google, está claro que elampliamente adoptadosistema debase de datos relacional(RDBMS)tiene su propio conjuntode problemascuando se aplica acantidades masivas de datos. Los problemasse refieren aeficienciade procesamientoeficiente, efectivapuesta en paralelo, escalabilidad y costos.

Google, a lo largo de los últimos años, construida conuna infraestructura degran escalabilidadpara sumotor de búsqueday otras aplicaciones, incluyendo Google Maps, Google Earth, Gmail,Google FinanceyGoogle Apps.

El enfoque de Googleera resolverel problemaen todos los nivelesde la pila deaplicación.El objetivoera construiruna infraestructura escalablepara el procesamiento paralelode grandes cantidades dedatos.Por lo tanto,Googlecreó un mecanismocompleto queincluyeun sistema de ficherosdistribuido,una columna-orientado a la familia almacén de datos,un sistema de coordinacióndistribuidos, yun entorno paraleloMapReducealgoritmo basado enla ejecución.

(Carlo Strozzi ,2013). En uno de los comentarios obtenidos de la web dice que:

Se usó el término NoSQL en 1998 para referirse a su base de datos. Era una base de datos open-source, ligera, que no ofrecía un interface SQL, pero sí seguía el modelo relacional (Strozzi sugiere que, ya que el actual movimiento NoSQL, Se sale completamente del modelo relacional, debería, por tanto, haberse llamado 'NoREL', o algo así.)

Eric Evans, un empleado de Rackspace, reintrodujo el término NoSQL cuando Johan Oskarsson de Last.fm quiso organizar un evento para discutir bases de datos distribuidas de código abierto. El nombre intentaba recoger el número creciente de bases de datos no relacionales y distribuidos que no garantizaban ACID, atributo clave en las RDBMS clásicas. Definitivamente, con el término NoSQL nos referimos a una multitud de bases de datos que intentan solventar las limitaciones que el modelo relacional se encuentra en entornos de almacenamiento masivo de datos, y concretamente en las que tiene en el momento de escalar, donde es necesario disponer de servidores muy potentes y de balanceo de carga.

[4]**Inktomi**:Provee servicio de búsqueda a una importante legión de empresas de la web como: HotBot, AOL, ICQ, GeoCities, Search MSN, GoTo, Canada.com, RadarUol, entre otros.

2.2 BASES DE DATOS RELACIONALES

En varias década el modelo de datos de almacenamiento más conocido es el modelo relacional, El término fue definido originalmente por Edgar Codd en IBM Almaden Research Center en 1970, el software de implementación de este modelo se denomina un sistema de gestión de bases de datos Relacional o RDBMS, el cual es capaz de producir, manipular y gestionar bases de datos de tipo relacional, los sistemas de bases de datos relacionales son aquellos que almacenan y administran de manera lógica los datos en forma de tablas. Una tabla es, a su vez un método por el cual podemos representar los datos en filas y columnas.

Cada columna representa un campo único de un registro. Varias de estas columnas o campo componen un registro, proveyendo información significativa yrelacionada. Cada registro es representado en una fila. Una tabla puede consistir en varias columnas. Muchas de las tablas que poseen datos relacionados e interdependientes son agrupadas por medio del establecimiento de relaciones entre ellas. Al administrar las tablas y sus relaciones, encontramos los medios para insertar, borrar, consultar y actualizar la información de un sistema RDBMS. La forma más común que se hace esto es mediante el lenguaje de consulta estructurado, SQL. Estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Un RDBMS proporcionar a los usuarios la capacidad de almacenar los datos en la bases de datos, acceder a ellos y actualizarlos. Esta es la función fundamental de un RDBMS y por supuesto, el RDBMS debe ocultar al usuario la estructura física interna (la organización de los archivos y las estructuras de almacenamiento)

FIGURA 3: Esquema – Representación Modelo Relacional

Existe software exclusivamente dedicado a tratar con bases de datos relacionales. Este software se conoce como SGBD (Sistema de Gestión de Base de Datos relacional) o RDBMS (del inglés Relational Database Management System).

Entre las más principales podemos destacar MySQL, PostgreSQL, Oracle, DB2, INFORMIX, y Microsoft SQL Server.

2.2.1 PRINCIPALES CARACTERÍSTICAS

Como bien se sabe las bases de datos relacionales o RDBMS son el motor principal de las aplicaciones, las RDBMS están compuestas por varias tablas o relaciones, estas no puede tener el mismo nombre en la misma base, cada tabla a su vez es un conjunto de registro (filas y columnas). Cada columna almacena información sobre una propiedad determinada de la tabla también conocida como atributo (nombre, apellido, edad, fecha), cada fila posee una concurrencia o ejemplar de la instancia o relación representada por la tabla (las filas también se las conoce como **tuplas**[5]).

2.2.2 LAS 12 REGLAS DE CODD

Actualmente las bases de datos relacionales son muchas, como también existen bases de datos que lo son, puesto que lo único que hacían es guardar la información en tablas, es por esa razón que (Edgar F. Codd, 2013)propone un sistema de 12 reglas con el fin de definir que se requiere para un sistema de administración de datos.

Un verdadero Sistema relaciona de Bases de datos debería cumplir las siguiente reglas.

1) **Información**. Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.

2) **Acceso garantizado**. Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.

3) **Tratamiento sistemático de los valores nulos**. El DBMS debe permitir el tratamiento adecuado de estos valores.

4) **Catálogo en línea basado en el modelo relacional**. Los metadatos deben de ser accesibles usando un esquema relacional.

5) **Sublenguaje de datos completo.** Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.

[5](Wikipedia, Tupla, 2014)

6) **Actualización de vistas.** El DBMS debe encargarse de que las vistas muestren la última información.

7) **Inserciones, modificaciones y eliminaciones de alto nivel**. Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.

8) **Independencia física.** Los datos deben de ser accesibles desde la lógica de la base de datos aun cuando se modifique el almacenamiento.

9) **Independencia lógica**. Los programas no deben verse afectados por cambios en las tablas.

10) **Independencia de integridad**. Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.

11) **Independencia de la distribución**. El sublenguaje de datos debe permitir que sus instrucciones funciones igualmente en una base de datos distribuida que en una que no lo es.

12) **No subversión**. Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.

2.3 INICIANDO LA PRÁCTICA CON NOSQL

Con el fin de entender completamente las diferencias entre estos sistemas, sus ventajas y desventajasen tema, en que en este caso es NoSQL, que es una abstracción de una clase de bases de datos. NoSQL es un concepto, una clasificación y un punto de vista de almacenamiento de datos de nueva generación. Se incluye una clase de productos y un conjunto de no relacionales opciones las alternativas almacenamiento de datos.

2.3.1 ANÁLISIS DE LA TECNOLOGÍA NOSQL

En la carrera de informática, muchos hemos aprendido que los sistemas de bases de datos se clasifican mayormente en tres tipos: Las bases de datos relacionales, las orientadas a objetos, y las relacionales orientadas a objetos. Sin embargo, pronto medi cuenta que, en la práctica, la mayoría de los motores de bases de datos más populares se basan en la arquitectura relacional, y todos ellos utilizan el lenguaje de consultas SQL (con variaciones) para operar con los datos. Tanto es así, que SQL se convirtió con el paso de los años en un estándar, debido a su uso.

2.3.2 ¿QUÉ ES NOSQL?

Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación al que todos nos acostumbramos, estas bases de datos son no relacionales y no proporcionan garantías ACID, el término hace referencia a un conjunto de características requeridas para que una serie de instrucciones puedan ser consideradas como una transacción, tampoco poseen un esquema fijo de tablas ni sentencias JOIN, con la cual permite la combinación de registros de dos o más tablas en una base de datos relacional,

Pero la principal característica de las bases de datos NoSQL es que están pensadas para manipular enormes cantidades de información de manera muy rápida.

En realidad, un sistema de este tipo no es siquiera una base de datos entendida como tal, sino un sistema de almacenamiento distribuido para gestionar datos dotados de una cierta estructura, estructura que además puede ser enormemente flexible. (Dans, 2011).

(Gonzales,2013) dice en uno de sus artículos.

Los nombres de los proyectos de estas bases de datos son tan diversos como extraños: Hadoop, Voldemort, Dynomite, y otros. Pero suelen estar unificados por algunos puntos en común, incluyendo:

No llamarlos bases de datos. Werner Vogels, CTO de Amazon, se refiere a su sistema Dynamo como un almacenamiento de clave-valor de alta disponibilidad. Google llama a su BigTable, otro de los modelos para muchos simpatizantes de NoSQL, un sistema de almacenamiento distribuido para gestionar datos estructurados.

Pueden manejar enormes cantidades de datos. Hypertable[6], una implementación de código abierto basada en BigTable, se usa dentro del motor de búsqueda Zvents para escribir 1000 millones de celdas de datos por día.A su vez, BigTable, en conjunto con su tecnología hermana MapReduce, procesa hasta 20 petabytes de datos por día.

Definitivamente es notable la cantidad de datos que hoy en día se maneja, es tan grande que las personas están buscando otras tecnologías.

[6]**Hypertable**: Esun sistema de base de datos de código abierto inspirado en las publicaciones sobre el diseño de BigTable de Google. El proyecto se basa en la experiencia de los ingenieros que estaban resolviendo tareas intensivas de datos a gran escala desde hace muchos años. - **Fuente**: http://en.wikipedia.org/wiki/Hypertable

Se ejecutan en clústeres de servidores de PC baratas. Los clústeres[7] de PC se pueden expandir de forma fácil y barata sin la complejidad y el costo del data sharding, que involucra recortar una base de datos en múltiples tablas para ejecutarse en grandes clústeres o grillas.

Google cuenta que uno de sus clúster de BigTable más grande gestiona 6 petabytes de datos sobre miles de servidores.

Superan los cuellos de botella de rendimiento. Al no tener que realizar la traducción de datos hacia un formato amigable para SQL, las arquitecturas NoSQL son mucho más rápidas.

SQL es un enfoque extraño para el código procedural, y casi todo el código es procedural, dice Curt Monash, un analista independiente de bases de datos y blogger. El costo de mapear los datos a SQL puede valer la pena para los casos en que estos datos tengan que manipularse extensivamente... pero cuando la estructura de la base de datos es muy simple, SQL no parece ayudar.

2.3.3 CARACTERÍSTICAS NOSQL

Las bases de datos tienen características únicas y una reputación sólida para la integridad de datos, pero todo esto puede resultar demasiado para nuestras necesidades.

Como menciona (Wiesse, 2011) en un artículo publicado sobre las características principales de una base de datos NOSQL las cuales responden a:

- ✓ **Modelos de datos variables y flexibles**: siempre hay un modelo, una estructura, pero la rigidez del modelo relacional desaparece…
- ✓ **Escalabilidad sencilla**, pueden responder a necesidades pequeñas y a altos volúmenes de trabajo.
- ✓ **Alta velocidad de respuesta a peticiones**. El tiempo a un es un muy alto costo
- ✓ **Diferentes lenguajes de consulta (no SQL).** Lenguajes muy fáciles de entender y comprender.

[7] **Clusters**: Se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora. **Fuente**: http://es.wikipedia.org/wiki/Cl%C3%BAster_%28inform%C3%A1tica%29

Entre otras las características comunes de todas las implementaciones de bases de datos distribuidas no relacionales, propietarias o no, suelen ser las siguientes:

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 periodo de tiempo. Esto se conoce también como BASE (Basically Available Soft-state Eventual Consistency o coherencia eventual flexible básicamente disponible), en contraposición a ACID[8], su analogía en las bases de datos relacionales.

Estructura distribuida: Generalmente se distribuyen los datos mediante mecanismos de tablas de hash[9] distribuidas (DHT) ya que realmente se trata, según las distintas implementaciones, de redes p2p[10].

Escalabilidad horizontal: La implementación típica se realiza en muchos nodos de capacidad de procesado limitado, en vez de utilizar grandes Mainframes.

Tolerancia a fallos y Redundancia: De entre todas las implementaciones de bases de datos NoSQL, hay muchas que no utilizan el lenguaje de consultas SQL (por ejemplo, MongoDB usa JSON[11]), pero hay algunas que siguen usándolo, como por ejemplo BigTable (GQL[12]), que lo ha transformado manteniendo su estructura básica.

Como se puede apreciar NoSQL ofrece una serie de beneficios, sin embargo también tienen sus desventajas:

---

[8]**ACID**: Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.
[9]**HASH**.- Una **tabla hash**, **mapa hash** o **tabla de dispersión** es una estructura de datos que asocia *llaves* o *claves* con *valores*.
[10]**P2P**.- Una **red *peer-to-peer***, **red de pares**, **red entre iguales**, **red entre pares** o **red punto a punto** (*P2P*, por sus siglas en inglés) es una red de computadoras en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí
[11]**JSON**.- acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. **Fuente**: http://es.wikipedia.org/wiki/JSON
[12]**GQL**.- es un lenguaje parecido a SQL que se utiliza para recuperar entidades o claves del almacén de datos escalable de App Engine. Aunque las funciones de GQL son distintas de las de un lenguaje de consulta de una base de datos relacional tradicional
Fuente: https://developers.google.com/appengine/docs/python/datastore/gqlreference?hl=es

1.- El código abierto puede significar una "mancha" en el soporte para las empresas

Mientras que los principales proveedores de RMBMS tales como Oracle, IBM y Sybase ofrecen buenos soportes a pequeñas, medianas y grandes empresas y típicamente a start-ups, los proveedores de código abierto no puedan tener los recursos suficientes para ofrecer un soporte similar -con excepción de un puñado de clientes blue-chip.

Generalmente un vendedor de código abierto no tiene el alcance global, servicios de soporte, y la credibilidad de Oracle o IBM.

2.- No están lo suficientemente maduros para algunas empresas

A pesar de sus implementaciones en algunas grandes empresas, las bases de datos NoSQL aún se enfrentan a un problema de credibilidad importante con muchas empresas. Los críticos señalan la falta de madurez de NoSQL y los posibles problemas de inestabilidad, comparado con el nivel de madurez y estabilidad que presenta las RDBMS.

3.- Limitaciones de Inteligencia de Negocios

Hay una o dos cuestiones acerca de las capacidades de BI de las bases de datos NoSQL. ¿Pueden estas bases de datos proporcionar la clase de minería de datos rigurosos que las empresas utilizan con las RDBMS? ¿Cuántos conocimientos de programación son necesarios para hacer la consulta ad-hoc y análisis?

Las respuestas no son precisamente positivas. Las bases de datos NoSQL no tienen muchos ganchos para el uso general de herramientas de BI, mientras que la más simple consulta ad-hoc y análisis implica unos conocimientos de programación avanzados. Sin embargo, hay soluciones disponibles. Quest Software, por ejemplo, ha creado Toad[13] para bases de datos en la nube, que proporciona capacidades de consulta ad-hoc para algunas bases de datos NOSQL.

4.- La falta de experiencia

La novedad de NOSQL significa que no hay una gran cantidad de desarrolladores y administradores que conocen la tecnología lo que hace difícil a las empresas encontrar personas con los conocimientos técnicos apropiados. Por el contrario, el mundo RDBMS tiene miles de personas muy cualificadas.

[13]**TOAD**:Esuna herramienta de diseñode base de datosque permitea los usuarios crearvisualmente, mantenery documentarlos sistemas debases de datos nuevaso existentes. Fuente: traducida de http://en.wikipedia.org/wiki/Toad_Data_Modeler

5.- 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, las interfaces de consultas son únicas y tienen peculiaridades. Esta falta de normas quiere decir que es imposible cambiar de un proveedor a otro fácilmente, por si el cliente no quedara satisfecho con el servicio dado.Párrafos (12-17)

2.3.4 DIFERENCIAS ENTRE BASES DE DATOS NOSQL

En los últimos años una gran variedad de bases de datos NoSQL se ha desarrollado principalmente por los profesionales y empresas de la web para adaptarse a sus requisitos específicos con respecto al rendimiento escalabilidad, mantenimiento y conjunto de características.

Si resumiéramos las características comunes que estos sistemas presentan, yo diría que son tres principalmente: Ausencia de esquema en los registros de datos, escalabilidad horizontal sencilla, y velocidad endiablada (aunque esto último no siempre es cierto, pues muchos de estos sistemas aún no están suficientemente maduros).

Así que ha habido varios enfoques para clasificar y subsumir bases de datos NoSQL, cada uno con diferentes categorías y subcategorías. Algunos enfoques de clasificación se presentan aquí de la que se llevará a cabo una de las posibilidades para clasificar a los bases de datos NoSQL en los capítulos posteriores.

Ausencia de esquema en los registros de datos

Esta primera característica significa que los datos no tienen una definición de atributos fija, es decir: Cada registro puede contener una información con diferente forma 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 un JOIN para obtener los datos relacionados, pues éstos se encuentran directamente en el mismo documento).

Escalabilidad Horizontal

La escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Un sistema escala horizontalmente si al agregar más nodos al mismo, el rendimiento de este mejora. Por ejemplo, al añadir una computadora nueva a un sistema que balancee la carga entre la antigua y la nueva puede mejorar el rendimiento de todo el sistema.

Existe 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 Map-Reduce, las cuales pueden ejecutarse en todos los nodos a la vez (cada uno operando sobre una porción de los datos) y reunir luego los resultados antes de devolverlos al cliente. La gran mayoría permiten también indicar otras cosas como el número de réplicas en que se hará una operación de escritura, para garantizar la disponibilidad. Y gracias al sharding y a no tener que replicar todos los datos en cada uno de los nodos, la información que se mueve entre las distintas instancias del motor de base de datos no tiene por qué ser demasiado intensiva. Claro, seguiremos encontrándonos con problemas de escalabilidad inherentes al tipo de software que estemos construyendo, pero seguramente podamos resolverlos más fácilmente con la ayuda de estas características.

Algunos de los sistemas realizan operaciones directamente en memoria, y sólo vuelcan los datos a disco cada cierto tiempo. Esto permite que las operaciones de escritura sean realmente rápidas. Por supuesto, trabajar de este modo puede sacrificar fácilmente la durabilidad de los datos, y en caso de cuelgue o apagón se podrían perder operaciones de escritura o perder la consistencia. Normalmente, esto lo resuelven permitiendo que una operación de escritura haya de realizarse en más de un nodo antes de darla por válida, o disminuyendo el tiempo entre volcado y volcado de datos a disco. Pero claro, aun así, existe ese riesgo.(Paramio, 2011)

(López, 2011), en uno de sus comentarios dice que.

Estas bases de datos tienen la ventaja de su rapidez y escalabilidad horizontal y los inconvenientes de que no garantizan las transacciones. Dependiendo del tipo de proyecto, puede que se adapte mejor una base de datos de este tipo o una relacional, todo depende de las necesidades del proyecto.

2.4 SQL VS NOSQL

El SQL (relacional) versus NoSQL escalabilidad es un tema controvertido. En este trabajo se argumenta en contra de los dos extremos. Aquí hay más fondo para apoyar a la posición de NoSQL.

Los argumentos relacionales a favor de NoSQL pueden ser de la siguiente manera:

- ✓ Si los nuevos sistemas relacionales pueden hacer todo lo que un sistema de NoSQL puede, con el rendimiento y la escalabilidad análogo, y con la comodidad de las transacciones y SQL, ¿por qué elegir un sistema de NoSQL?

- ✓ Relational DBMS han tomado y retenido la cuota de mercado mayoritaria sobre otros competidores en los últimos 30 años: los DBMS de red, de objetos y.

- ✓ DBMS relacionales exitosos han sido construidos para manejar otras cargas de aplicaciones específicas en el pasado: sólo lectura o almacenamiento de datos, OLTP en las CPU de varios núcleos múltiples discos , bases de datos en memoria , bases de datos distribuidas , y ahora escalado horizontalmente sobre todo lectura bases de datos.

Si bien no vemos "una talla para todos" en los propios productos SQL, sí vemos una común interfaz con SQL, transacciones, y el esquema relacional que da ventajas en la formación, intercambio continuidad y datos.

Los argumentos en contra de NoSQL:

- ✓ Todavía no hemos visto buenos puntos de referencia que muestran que los RDBMS puedan alcanzar la escala comparable con los sistemasNoSQL como BigTable de Google.

- ✓ Si sólo necesita una búsqueda de objetos en función de una sola clave, y luego una base de almacenamiento de clave-valor es adecuada y probablemente más fácil de entender que un DBMS relacional. Del mismo modo para un almacén de documentos en una aplicación sencilla: usted sólo paga la curva de aprendizaje para el nivel de complejidad que requiere.

- ✓ Algunas aplicaciones requieren un esquema flexible que permite a cada objeto de una colección tener diferentes atributos. Mientras que algunos RDBMS permiten "embalaje" eficiente de tuplas con los atributos que faltan, y algunos permiten agregar nuevos atributos en tiempo de ejecución, esto es poco común.

- ✓ Un DBMS relacional hace "caro" (multi mesa multinodo) operaciones " demasiado fácil". NoSQLsistemas los hacen imposible u obviamente caro para los programadores.

- ✓ Mientras RDBMS han mantenido cuota de mercado mayoritaria en los últimos años, otros productos han establecido mercados más pequeños, pero no triviales en las zonas donde hay una necesidad de capacidades particulares , por ejemplo, objetos indexados con productos como BerkeleyDB , o las operaciones con gráficos de seguimiento con los DBMS orientado a objetos.

2.5 ARQUITECTURA DE LOS SISTEMAS DE GESTIÓN DE BASES DE DATOS NOSQL

Típicamente las bases de datos relacionales modernas han mostrado poca eficiencia en determinadas aplicaciones que usan los datos de forma intensiva, incluyendo el indexado de un gran número de documentos, la presentación de páginas en sitios que tienen gran tráfico, y en sitios de streaming audiovisual. Las implementaciones típicas de RDBMS se han afinado o bien para una cantidad pequeña pero frecuente de lecturas y escrituras o para un gran conjunto de transacciones que tiene pocos accesos de escritura. Por otro lado NoSQL puede servir gran cantidad de carga de lecturas y escrituras.

Las arquitecturas NoSQL frecuentemente aportan escasas garantías de consistencia, tales como consistencia de eventos o transaccional restringida a ítems únicos de datos. Algunos sistemas, sin embargo, aportan todas las garantías de los sistemas ACID en algunas instancias añadiendo una capa intermedia (como por ejemplo, AppScale o CloudTPS). Hay dos sistemas que han sido desplegados y que aportan aislamiento snapshot[14] para almacenamientos de columna: El sistema Percolator de Google (basado en el sistema BigTable) y el sistema transaccional de Hbase desarrollado por la universidad de Waterloo. Estos sistemas, desarrollados de forma independiente, usan conceptos similares para conseguir transacciones ACID distribuidas de múltiples filas con garantías de aislamiento snapshot para el sistema subyacente de almacenamiento en esa columna, sin sobrecarga extra en la gestión de los datos, despliegue en el sistema de middleware[15], ni mantenimiento introducido por la capa de middleware.

[14] **Snapshot**. Es un formato de archivo de Microsoft Access. **Snapshot** o copia instantánea de volumen.
[15] **Middleware:** Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

Bastantes sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores, y el fallo en un servidor puede ser tolerado.

Algunos defensores de NoSQL promueven interfaces simples tales como los arrays asociativos o los pares clave-valor. Otros sistemas, tales como las bases de datos nativas en XML, promueven el soporte del estándar xquery.

(Martin, 2013) En una cita de sus artículos dice que:

Para conseguir cumplir con todo esto, normalmente no se cumplen las garantías ACID, y el sistema tiene una arquitectura distribuida. Además, la interfaz de acceso a los datos pierde capacidad de expresión con respecto a SQL (a esto se debe el nombre NoSQL), dado que la complejidad del esquema de los datos será mucho menor.

Aunque en los sistemas más modernos todas estas características son configurables, cuando se requieren resultados de rendimiento óptimos es necesario renunciar a gran parte de ellas. Por ejemplo, lo más normal es que la consistencia fuerte no esté garantizada, dado que en una arquitectura distribuida los cambios deben propagarse entre las diferentes máquinas. Cumplir con los principios ACID supondría una lacra para el rendimiento que haría el sistema poco óptimo para el escenario más normal de Big Data.

Hoy en día se pueden encontrar diferentes sistemas de bases de datos distribuidos que potencian diferentes atributos. De hecho, existe un teorema sobre sistemas distribuidos, el teorema CAP, que los agrupa de forma clara. Según este teorema, un sistema distribuido no puede satisfacer de manera completa los siguientes atributos al mismo tiempo: Consistencia Fuerte, Alta Disponibilidad y Tolerancia a Particionamiento. Teniendo en cuenta que una base de datos distribuida puede cumplir con dos de estos atributos de manera satisfactoria, se pueden establecer grupos de estos sistemas de acuerdo a estas características. En la siguiente agrupación se pueden observar algunos ejemplos:

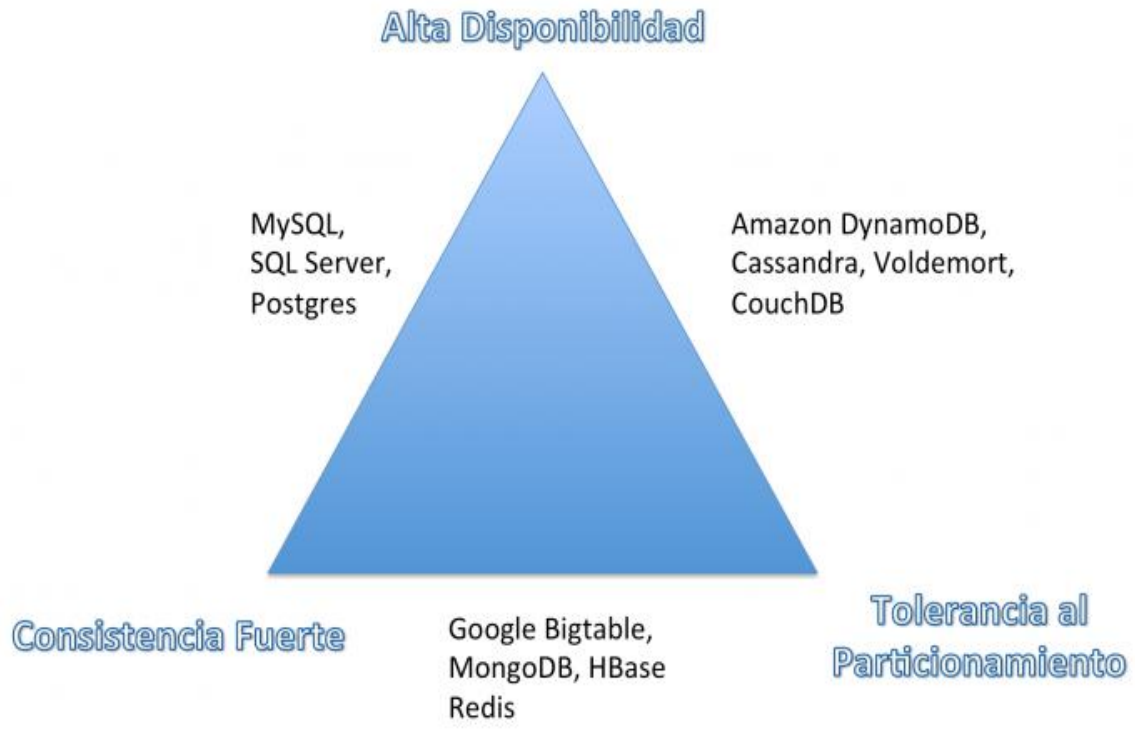


FIGURA 4: Arquitectura- Teorema CAP (Browne, 2011)

La consistenciaes decir, siycómo unsistema está enun estado coherentedespués de la ejecuciónde una operación. Un sistema distribuidopara ser consideradogeneralmente coherentedespués deuna operación de actualizacióndealgún escritor dondetodos loslectores puedan versusactualizacionesde algúnorigen de datos compartido.

La alta disponibilidad que significa que un sistema ha sido diseñado y aplicado de una manera que le permite continuar la operación (es decir, lo que permite leer y escribir operaciones) si nodos en un accidente de clúster o algunas piezas de hardware o de software se han reducido debido a mejoras.

Toleranciapartición se entiende como la capacidaddel sistemapara continuar la operaciónen presenciadeparticiones de red. Esto ocurresi dos o más"islas" denodos de la redquesurgen(temporalmente o permanentemente) no pueden conectarse entre sí. Algunas personastambién entiendenla toleranciapartición como lacapacidad de un sistemapara hacer frente alaadición dinámicay la eliminaciónde nodos.

(Gilbert & Lynch, 2013) Define la tolerancia partición como:

No se permite el conjunto de fallas menores que el fracaso total de la red a la que el sistema responde de forma incorrecta y tomó nota de la observación de Brewer de que una partición de un nodo equivale a una caída del servidor, ya que si no puede conectarse a él, puede ser que también no estar allí.

(Browne, 2011) En uno de sus artículos nos informa sobre la importancia del teorema CAP el cual viene a la vida como una balanza de aplicación. A volúmenes transaccionales bajas, de pequeñas latencias para permitir que las bases de datos conseguir consistencia, no tienen ningún sentido perceptible en cuanto a rendimiento global o la experiencia del usuario. Cualquier distribución de la carga que no se compromete, por lo tanto, es probable que sea por razones de gestión de sistemas.

En cuanto a las bases de datos, Brewer concluye que las actuales "Bases de datos son una mejor consistencia en la disponibilidad" y que "las bases de datos de área extensa no puede tener ambas cosas" (Browne, 2011)una noción que está ampliamente adoptado en la comunidad NoSQL y ha influido en el diseño de almacenes de datos no relacionales.

Ahora, Brewer alega que uno puede elegir dos de estas tres características en un "sistema de datos compartidos" como máximo. En su artículo, se refirió a las compensaciones entre ACID y sistemas de base y ha propuesto como un criterio de decisión para seleccionar uno o el otro para los casos de uso individuales si un sistema o partes de un sistema tienen que ser consistentes y de particiones tolerante, se requiere propiedades ACID, si la disponibilidad y la partición de tolerancia son favorecidos sobre la consistencia, el sistema resultante puede caracterizarse por las propiedades de la base.

El último es el caso para Dynamo de Amazon que está disponible y admite particiones tolerantes, pero no es estrictamente constante, las escrituras de un cliente no se ven inmediatamente después de haber sido comprometido a todos los lectores. Bigtable de Google ni elige ACID ni base, pero tiene la alternativa del tercer teorema de CAP, que es un sistema consistente y disponible y en consecuencia, no puede funcionar completamente en presencia de particiones de red.

En su el articulo Brewer señala rasgos y ejemplos de las tres opciones diferentes que se puede hacer de acuerdo a su teorema CAP.

TABLA 1: Teorema CAP- Opciones Rasgos y Ejemplos

Opción	Rasgos	Ejemplos
Consistencia + Disponibilidad (Pierde Particiones)	2 fase de confirmación de protocolos de validación	Bases de datos de un solo sitiolas bases de datos del clúster LDAP Sistema de archivos XFS
Consistencia + Partición tolerancia (Pierde Disponibilidad)	bloqueo pesimista Hacer particiones minoritarios disponible	bases de datos distribuidas bloqueo Distribuido protocolos Mayoría
Disponibilidad tolerancia + Partición (Pierde consistencia)	Vencimientos / arrendamientos resolución de conflictos optimista	Coda Web Cachinge [sic!] DNS

En cuanto a las bases de datos el teorema CAP es una noción queestá ampliamente adoptandoen la comunidadNoSQLy hainfluido en el diseñodealmacenes de datosno relacionales.

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 llave-valor, como una tabla en la que las columnas son dinámicas, pueden cambiar sin perder la agrupación de la información, así es que puedo tener 'personas' con más atributos que otras, puedo cambiar la estructura de mi información dinámicamente sin tener que rediseñar todo de nuevo.

Una de las gracias de estas bases de datos es que permiten almacenar gran cantidad de datos y escalar linealmente, sin afectar el performance y además son extremadamente rápidas. Es cierto que para una aplicación normal que no almacena muchos datos quizás esto no sea necesario y con MYSQL sea suficiente, pero si se necesita almacenar muchísima información, una base de datos como MYSQL no podrá escalar bien, incluso con clúster, así es que es una buena idea echarle un vistazo a estas soluciones, aunque debo decir que NoSQLno es para todo, cada base de datos

tiene su aplicación y tienes que ver cuál es la que más te sirve antes de implementar tu aplicación.

En la actualidad existen algunos tipos de bases de datos NoSQL, dependiendo de cómo Almacenan la información en los siguientes temas podemos explicar una clasificación según algunos autores.

2.5.1 TAXONOMÍA DE LOS MODELOS DE DATOS

Ha habido varios enfoques para clasificar las bases de datos NoSQL, cada uno con diferentes categorías y subcategorías. Debido a la variedad de enfoques y superposiciones es difícil de conseguir y mantener una visión general de las bases de datos no relacionales. Sin embargo, la clasificación básica de que la mayoría estaría de acuerdo en se basa en el modelo de datos. A algunos de estos y sus prototipos son:

- ✓ **Columna**: HBase, Accumulo, Cassandra
- ✓ **Documento**: MarkLogic, MongoDB, Couchbase
- ✓ **Valor-clave**: Dynamo, Riak, Redis, MemcacheDB, Project Voldemort
- ✓ **Gráfico**: Neo4J, Allegro, Virtuoso

(Yen, 2009) En un artículo publicado, escribe sobre las diferentes categorías de las bases de datos NoSQLcomo se muestra en la siguiente tabla.

TABLA 2: Clasificación.- Taxonomía de las bases de datos según(Yen, 2009)

Categorías	Matching Database
KV Cache	Memcached, Repcached, Coherence, Infinispan, eXtreme Scale, Cache, Velocity, Terracotta, Gigaspaces XAP
KV Store	Keyspace, Flare, SchemaFree, RAMCloud
KV Store - Eventually consistent	Dynamo, Voldemort, Dynomite, SubRecord, MotionDb, DovetailDB
Data-structures server	Redis
KV Store - Ordered	TokyoTyrant, Lightcloud, NMDB, Luxio, MemcacheDB, Actord
Tuple Store	Gigaspaces, Coord, Apache River
Object Database	ZopeDB, DB4O, Shoal, Perst
Document Store	MarkLogic, CouchDB, MongoDB, Jackrabbit, XML-Databases, ThruDB, CloudKit, Persevere, Riak Basho, Scalaris

Wide Columnar Store	BigTable, HBase, Cassandra, Hypertable, KAI, OpenNeptune, Qbase, KDI

Una similar taxonomía sobre las bases de datos NoSQL hace (Strauch, 2012) quien en su documentos menciona que Ken North en uno de sus artículos de "databases in the cloud" que existen distribuciones disponibles en informática en entornos en Cloud computing, en un pequeño resumen en la tabla 2.2

TABLA 3: Clasificación- La categorización por (North, 2013)

Categorías	Bases de datos
Distributed Hash Table, Key-Value Data Stores	memcached MemcacheDB Project Voldemort Scalaris Tokyo Cabinet
Entity-Attribute-Value Datastores	Amazon SimpleDB Google AppEngine datastore Microsoft SQL Data Services Google Bigtable Hadoop HyperTable HBase
Amazon Platform	Amazon SimpleDB
Document Stores, Column Stores	Sybase IQ Vertica Analytic Database Apache CouchDB

Una clasificación similar nos hace mención (Cattell, 2013)donde las diferentes bases de datos NoSQL se caracterizan primariamente por su modelo de datos

TABLA 4: Clasificación- Categorización según (Cattell, 2013)

Categorías	Bases de Datos Por Modelo
Key-value Stores Redis	Scalaris, Tokyo Tyrant, Voldemort, Riak
Document Stores SimpleDB	CouchDB, MongoDB, Terrastore

Extensible Record Stores Bigtable	HBase, HyperTable, Cassandra

2.5.2 COMPARACIÓN POR ESCALABILIDAD, DATOS Y CONSULTAS DE MODELO, PERSISTENCIA DE DISEÑO

En el Blog “NoSQL ecosystem” de Jonathan Ellis discute sobre las bases de datos NoSQL por tres importantes aspectos.

1. Escalabilidad

2. Datos y consulta de Modelos

3. Persistencia del Diseño

✓ **Escalabilidad**

(Ellis, 2013) Argumenta a que es fácil de pensar las operaciones leídas por réplicas de datos y distribuciones de carga entre esas replicas. Por tanto, él sólo investiga las operaciones básicas de escritura de bases de datos que es realmente distribuido y ofrece la partición de datos automáticamente. Cuando el tamaño de datos excede a las capacidades de una sola máquina, los últimos sistemas parecen ser la única opción para él, si siempre hay voluntad de dividir los datos de forma. Para los sistemas distribuidos pertinentes que disponen auto-sharding[16] Ellis considera que hay dos características importantes:

✓ Soporte para múltiplescentros de datos

✓ Posibilidad de añadir máquinas encendidas a un clúster existente, transparente para las aplicaciones que utilizan este grupo

Por estas características Ellis compara una selección de almacenes de datos NoSQL cumplimiento de los requisitos de la distribución de bienes y auto-sharding (Tabla 2.4).

[16]Esel proceso de almacenarregistros de datosa través de múltiplesmáquinasyes el enfoque deMongoDBparasatisfacer las demandas decrecimiento de los datos.

TABLA 5: Clasificación – Características comparativas de Escalabilidad (Ellis, 2013)

Bases de Datos	Agregar Maquinas Encendidas	Soporte Multi-Datacenter
Cassandra	x	X
HBase	x	
Riak	x	
Scalaris	x	
Voldemort		Some code required

✓ **Datos y consulta de Modelos**

La segunda zona que examina es el modelo de datos y la API de consulta que ofrece diferentes bases de almacenamientoNoSQL. En la Tabla 2.5 muestra los resultados de su investigación señalando una gran variedad de modelos de datos y las API de consulta.

TABLA 6: Clasificación - Comparación de los modelos de datos y API de consulta

Nombre	Idioma	Notas
BaseX	Java , XQuery	Base de datos XML
Cloudant	Erlang , Java , Scala , C	JSON almacen (servicio en línea)
Clusterpoint	C + +	XML, orientado para la búsqueda de texto completo
Servidor Couchbase	Erlang , C , C + +	Soporte para JSON y documentos binarios
Apache CouchDB	Erlang	JSON base de datos
djondb [7][8][9]	C + +	JSON , ACID, Almacenamiento Documento.
Elasticsearch	Java	JSON , el motor de búsqueda
eXist	Java , XQuery	Base de datos XML
Jackrabbit	Java	Java Content Repository aplicación
IBM Lotus Notes y Lotus Domino	LotusScript , Java , IBM X Pages, otros	MultiValue
MarkLogic Servidor	XQuery , Java , REST	Base de datos de XML con soporte para

		JSON , texto y binarios
Nombre	**Idioma**	**Notas**
MongoDB	C + + , C # , Go	BSON Almacen (formato binario JSON)
Oracle Database NoSQL	Java , C	
OrientDB	Java	JSON , soporte de SQL
CoreFoundationlista Propiedad	C , C + + , Objective-C	JSON , XML , binario
Sedna	XQuery , C + +	Base de datos XML
SimpleDB	Erlang	servicio en línea
TokuMX	C + + , C # , Go	MongoDB con la indexación del árbol del fractal
OpenLink Virtuoso	C + + , C # , Java , SPARQL	middleware y motor de base de híbridos

✓ **Persistencia del Diseño**

El tercer aspecto en el que (Ellis, 2013) compara su selección de almacenes de datos NoSQL es la forma en la que las bases almacenan los datos (Tabla 2.6)

TABLA 7: Clasificación – Comparación según la persistencia del diseño

Bases de datos	Persistencia del Diseño
Cassandra	Memtable / SSTable
CouchDB	Append-only B-tree
HBase	Memtable / SSTable on HDFS
MongoDB	B-tree
Neo$j	En la lista de discos vinculados
Redis	Dentro de memoria de fondo snapshots
Riak	¿
Scalaris	Dentro de memoria
Tokyo cabinet	Hash or B-tree

Voldemort	Conexión (principalmente BDB MySQL

Considerando la persistencia del diseño como una particular importancia en que estas bases de datos realizan las cargas de trabajo:

Dentro memoria de Bases las bases de datos (In-Memory Databases) son muy rápidos, (por ejemplo Redis puede llegar a más de 100.000 operaciones por segundo en un maquina individual) pero el tamaño de los datos es inherentemente limitados por el tamaño de la memoria RAM. Se considera la persistencia del diseño como importante para estimar las cargas de trabajo para que estas bases de datos tengan un buen rendimiento.

Memtables y SSTables.- Esta estrategia de persistencia tiene características de rendimiento comparables a los de la memoria en bases de datos (una comparación de discos basados en estrategias de búsquedas en el disco se reducen debido a que solo adjunta un registro y el enrojecimiento de Memtables enteras en el disco), pero evita las dificultades de durabilidad en bases de datos de memoria.

B-trees.- se han utilizado en las bases de datos ya que su comienzo para proporcionar un sólido de indización de apoyo. Sus características de rendimiento de los discos giratorios no son muy positivos debido a la cantidad de búsquedas en disco necesario para operaciones de lectura y escritura. La base de datos documental CouchDB usa B-Tree internamente pero evita la sobrecarga de búsquedasolo agregando los B-Tree, lo que implica la desventaja de que se realice una sola operación de escritura en el momento.

2.6 CLASIFICACIÓN BASADA EN LAS NECESIDADES DEL CLIENTE

Se presenta un enfoque diferente para clasificar las bases de datos NoSQL, subsumir las bases de datos según las necesidades del cliente:

Primera Característica.- Es clase de bases de datos proporciona un gran número de características de alto nivel que hacen que el trabajo de el programador sea fácil. En el lado negativo son difíciles de escalar.

Ejemplos: Oracle, Microsoft SQL Server, IBM DB2, MySQL, PostgreSQL, Amazon RDS.

Primera Escala.- Este tipo de bases de datos tiene que escalar desde el principio. En el lado negativo, carecen de características particulares y ponen de nuevo la responsabilidad al programado.

Ejemplos: Project Voldemort, Ringo, Amazon SimpleDB, Kai, Dynomite, Yahoo PNUTS, ThruDB, Hypertable, CouchDB, Cassandra, MemcacheDB.

Simple estructura de almacenamiento.- estas clases de bases de datos subsumen a clave /valor (Key/value) en un énfasis de almacenar y recupera conjuntos de estructura arbitraria. La desventaja es que no tienen las características o escalabilidad de otros sistemas.

Ejemplos: file systems, Cassandra, BerkelyDB, Amazon SimpleDB.

Propósito de almacenamiento optimizado.- Estas son las bases de datos que han sido diseñados y construidos para ser buenos en cosas como almacenamiento de datos o procesamiento de flujo.

Ejemplo: StreamBase,Vertica,VoltDB, Aster Data,Netezza,Greenplum.

2.7 CLASIFICACIÓN EN LA PRESENTE TESIS

- ✓ **Bases de Datos Clave/Valor (Key / value):**Llave-valor es la forma más típica, como un hashmap[17]donde cada elemento está identificado por una llave única, lo que permite la recuperación de la información de manera muy rápida. muchas de ellas están basadas en la publicación de google acerca de su bigtable y de Amazon. Dentro de estas bases de datos podemos encontrar a bigtable de google, simpledb de Amazon, Cassandra, HBase (Hadoop), Riak, Voldemort, tokio-cabbinet y MemcacheDB entre otras.

- ✓ **Bases de Datos Documentales (document based):** Estas almacenan la información como un documento (generalmente con una estructura simple como JSON o bson) y con una llave única. podemos encontrar a MongoDB y CouchDB entre las más importantes de este tipo.

- ✓ **Bases de Datos en Grafos (graph db):** Hay otras bases de datos que almacenan la información como grafos donde las relaciones entre los nodos son lo más

[17] HASHMAP. - En informática, una tabla hash (también de hash mapa) es una estructura de datos utilizada para implementar una matriz asociativa, una estructura que puede asignar claves a los valores. (Wikipedia, Hash Table, 2013)

importante. son muy útiles para representar información de redes sociales. encontramos a Neo4J entre otras.

- ✓ **Bases de datos orientadas a columnas:**Esta es la cuarta clase de bases de datos que se realizara la investigación estos sistemas de bases de datos que tienen la característica de almacenar los datos en forma de columna. Tienen una ventaja principal de este tipo de sistema, es que permite el acceso a grandes volúmenes de datos de forma rápida porque se puede acceder como una unidad a los datos de un atributo particular en una tabla. Un SMBD orientada a columnas es un sistema de gestión de bases de datos que almacena su contenido por columnas (atributos) y no por filas (registros) como lo hacen los SMBD relacionales.

2.8 NOSQL Y LAS APLICACIONES WEB 2.0

Entendemos por el término Web 2.0 que comprende aquellos sitios web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la World Wide Web. Un sitio Web 2.0 permite a los usuarios interactuar y colaborar entre sí como creadores de contenido generado por usuarios en una comunidad virtual, a diferencia de sitios web estáticos donde los usuarios se limitan a la observación pasiva de los contenidos que se han creado para ellos. Ejemplos de la Web 2.0 son las comunidades web, los servicios web, las aplicaciones Web, los servicios de red social, los servicios de alojamiento de videos, las wikis, blogs, mashups y folcsonomías. El término Web 2.0 está asociado estrechamente con Tim O'Reilly, debido a la conferencia sobre la Web 2.0 de O'Reilly Media en 2004

En opinión de Jorge Pérez, a pesar de ser muy diversas las plataformas que conviven bajo la denominación de NoSQL, se pueden identificar ciertas ventajas en algunas líneas de sistemas. Por ejemplo, eficiencia y elasticidad. “Esto se logra en sistemas de datos poco estructurados muy usados en aplicaciones de web 2.0. En éstos no son necesarios los ‘joins’ (una de las tareas más complejas de realizar con una base de datos relacional, que esencialmente consiste en relacionar piezas de datos almacenados en distintas tablas) y muchas de las plataformas NoSQL son capaces de funcionar de manera transparente con muchos servidores dividiendo partes de los datos y, por lo tanto, escalando en eficiencia. Pero, más importante aún, pueden soportar la inclusión de manera dinámica de nuevas máquinas para procesar los datos a medida que la carga va creciendo”

2.9 LISTADO DE ALGUNOS PRODUCTOS NOSQL POPULARES

Los productos más conocidos en cuanto a almacenamiento en bases de datos son varios en lo cual solo estudiaremos cuatro de las más nombradas en estos últimos años.

A continuación,una lista dealgunos de losmás conocidosproductosNoSQLy clasificación en función de su características y atributos.

2.9.1 BASES DE DATOS CLAVE/VALOR (KEY / VALUE)

Como menciona (Tiwari, 2011) en un párrafo de sus libro, acerca de estas bases de datos las cuales utilizan un HashMap o un array asociativo el cual es la más simple estructura de datos que puede contener un conjunto de pares para el almacenamiento clave / valor (Key/value). Estas estructuras de datos son muy populares porque ofrecen un eficiente y gran algoritmo de promedio de tiempo de ejecución para acceder a los datos. La clave de un par clave/valor (Key/value) es un valor único en el conjunto y puede ser fácilmente levantarlo para acceder a los datos.

(David , y otros, 1997) La necesidad de utilizar hashing consistente surgió de limitaciones experimentado durante la ejecución de las colecciones de máquinas de almacenamiento en caché - cachés web, por ejemplo. Si usted tiene una colección de máquinas de caché **n** entonces una forma común de equilibrio de carga a través de ellos es poner en la memoria caché de objetos o número de la máquina de hash (o) mod n. Esto funciona bien hasta que se agrega o quita máquinas de caché (por cualquier razón), por entonces n cambios y cada objeto es ordenado a una nueva ubicación. Esto puede ser catastrófico ya que los servidores de contenido de origen, estén inundados de solicitudes de las máquinas de caché. Es como si la caché se desapareció de repente. Qué lo ha hecho, en cierto sentido. (Esto es por qué usted debe cuidar es necesaria hashing consistente evitar inundar sus servidores).

Sería bueno que, cuando se añadió una máquina de caché, que tomó una buena cantidad de objetos de todas las otras máquinas de caché. Igualmente, cuando se eliminó la máquina de caché, que sería bueno si se compartían sus objetos entre las máquinas restantes. Esto es exactamente lo que hace hashing consistente - mapas constantemente objetos a la misma máquina de caché, en la medida de lo posible, por lo menos.

(White, 2007)Dice en su blogger que la idea básica detrás del algoritmo de hash consistente es para discutir a ambos objetos y cachés utilizando la misma función hash.

La razón de hacer esto es para asignar la memoria caché para un intervalo, que contendrá un número de hashes de objetos. Si la memoria caché se elimina a continuación, su intervalo es tomada por una memoria caché con un intervalo adyacente. Todos los demás cachés se mantienen sin cambios.

DEMOSTRACIÓN

Veamos esto con más detalle. La función hash en realidad asigna los objetos y la memoria caché para un rango de números. Esto debería ser familiar para todos los programadores de Java, el método hashCode en Object devuelve un inti, que se encuentra en el rango de -231 a 231-1. Imagínese la cartografía de este rango en un círculo lo que los valores se envuelven alrededor. Aquí hay una foto del círculo con una serie de objetos (1, 2, 3, 4) y cachés (A, B, C) marcados en los puntos que hash para (sobre la base de un esquema de almacenamiento en caché Web con hashing consistente por David Karger):

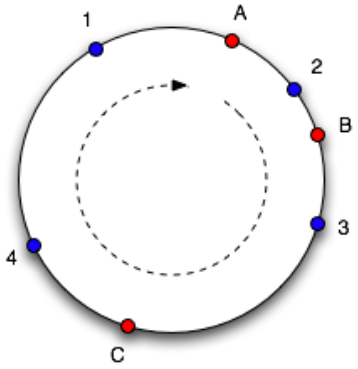


FIGURA 5: HashingConsistente- Situacióninicial (tomada de (White, 2007)

Para encontrar lo que almacena en caché un objeto entra, nos movemos en sentido horario alrededor del círculo hasta que encontremos un punto de caché. Así que en el diagrama de arriba, vemos el objeto 1 y 4 pertenecemos en caché un objeto 2 pertenece en caché B y el objeto 3 va en caché C. Considere lo que sucede si se elimina la memoria caché C: objeto 3 ahora pertenece en caché A, y todo las demás asignaciones de objetos no se han modificado. Si luego otro caché D se añade en la posición marcada tomará objetos 3 y 4, dejando solo objeto 1 que pertenece a A.

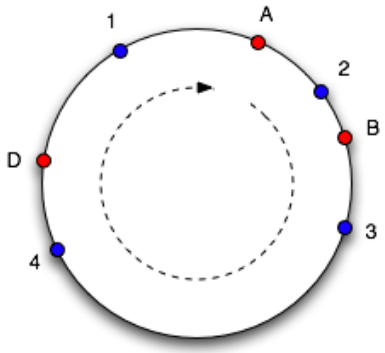


FIGURA 6: Hashingconsistente -Situación trasNododе unión yde salida (tomada de ((White, 2007)

Esto funciona bien, excepto el tamaño de los intervalos asignados a cada caché es bastante inconsistente. Puesto que es esencialmente aleatoria que es posible tener una distribución muy no uniforme de los objetos entre cachés. La solución a este problema es introducir la idea de "nodos virtuales", que son réplicas de los puntos de caché en el círculo. Así que cada vez que añadimos una caché creamos una serie de puntos en el círculo.

Entonces, ¿cómo puede usted usar hashing consistente? Lo más probable es que su encuentro en una biblioteca, en lugar de tener que codificar por sí mismo. Por ejemplo, como se mencionó anteriormente, memcached, un sistema de objetos de caché de memoria distribuida, ahora tiene clientes que admiten hashing consistente. Ketama Last.fm 's por Richard Jones fue el primero, y ahora hay una aplicación Java por Dustin Sallings. Es interesante observar que sólo el cliente que necesita para implementar el algoritmo de hash coherente - el servidor memcached es sin cambios. Otros sistemas que emplean hashing consistente incluyen Chord, que es una implementación de tabla hash distribuida, y Dynamo de Amazon, que es un almacén de claves y valores (no está disponible fuera de Amazon).

Las bases de datos C***lave/Valor son de diversos tipos***: algunos guardan los datos en memoria y algunos proveen la capacidad para conservar los datos en el disco. Pares clave/valor puede ser distribuido y se mantiene en un grupo de nodos.

Un ejemplo sencillo, potente, de estas bases de datos clave/valor es **BerkeleyDB**y **Amazon dynamoDB**.

- ✓ **Base de datos BerkeleyDB**

Es un motor de almacenamiento puro donde la clave y el valor son una matriz de bytes. El motor de almacenamiento central de Berkeley DB puede no atribuir un significado a la clave o el valor. Se necesita pares de matrices de bytes y devuelve la parte de atrás de la misma llamada cliente. BerkeleyDB permite que los datos se almacenen en la memoria y vacíen en el disco a medida que crece. También hay una noción de la indexación de las claves para acelerar la búsqueda y acceso. Berkeley DB ha existido desde mediados de la década de 1990. Se creó para sustituir a AT & T`s NDBM como parte de la migración de BSD 4.3 a 4.4.

Los sistemas operativos, bases de datos, componentes de middleware y aplicaciones de uso de almacenamiento en caché.

Es un software de biblioteca que proporciona un alto rendimiento de base de datos integrada para datos clave / valor. Berkeley DB está escrito en C con fijaciones API para C + +, C #,PHP,Java,Perl,Python,Rubí,Tcl,Smalltalk, y muchos otros lenguajes de programación.(Wikipedia, the free encyclopedia, 2013) BDB almacena pares arbitrarios de clave / datos como matrices de bytes, y soporta múltiples elementos de datos para una sola clave. Berkeley DB no es una base de datos relacional.

BDB puede soportar miles de procesos simultáneos de control de procesos concurrentes o la manipulación de bases de datos tan grandes como 256 terabytes. En una amplia variedad de sistemas operativos. Berkeley DB es también utilizan como el nombre común para tres productos distintos, Oracle Berkeley DB, Berkeley DB Java Edition, y Berkeley DB XML. Estos tres productos todos comparten un ancestro común y se encuentran actualmente en desarrollo activo en Oracle Corporation.

Características

- ✓ Los datos se almacenan en el formato nativo del lenguaje de programación.
- ✓ No tiene modo cliente-servidor.
- ✓ Caché configurable para modificar el rendimiento.
- ✓ Permite crear bloqueos de forma detallada. Esto es especialmente útil para trabajos concurrentes sobre la base de datos de forma que se bloquea una página de

registros durante una transacción para evitar que se modifiquen hasta que termine pero permitiendo actuar sobre el resto de páginas.

- ✓ Posibilidad de realizar copias de seguridad y replicación en caliente.
- ✓ Transacciones y recuperación ante errores ACID. Esto es configurable de forma que se puede ir relajando en función de la aplicación.
- ✓ Es compatible con algunas interfaces históricas para bases de datos en UNIX como dbm, ndbm y hsearch.
- ✓ Permite utilizar la característica de snapshots para poder efectuar varias transacciones sobre los mismos registros de manera simultánea

Posee tres productos asociados a la marca:

Berkeley DB: La base de datos original escrita en C.

Berkeley DB Java Edition: Una versión de la anterior con algunas características menos pero con la ventaja de estar escrita en un lenguaje multiplataforma.

Berkeley XML DB: especialmente ideada para almacenar documentos XML mediante colas XQuery. Esta versión actúa como una capa sobre Berkeley DB y tiene bindings para varios lenguajes (Java, C, PHP, etc.).

- ✓ **Bases de datos Amazon DynamoDB**

Dynamo es una base de datos NoSQL propietaria y de código cerrado, por lo que su uso es exclusivo de la empresa que la implementó: Amazon.

Amazon Dynamo es una de varias bases de datos utilizadas por Amazon para diferentes propósitos (otras son las bases de datos SimpleDB o S3, el Simple Storage Service). Debido a su influencia en una serie de bases de datos NoSQL, especialmente en las bases de datos (key-/value), se realizara un estudio de características ventajas y desventajas.

(DeCandia, y otros, 2007) En uno de sus capítulos se especifica a esta base de datos como una gran tabla de hash distribuida (DHT) y accesible mediante un mecanismo de “Clave-Valor”.

Pese a que esta base de datos no está accesible a los programadores, vale la pena dar a conocer sus principales características ya que DynamoDB es la primera piedra de las bases de datos NoSQL

Es muy importante destacar que uno de los aspectos interesante de Dynamo es el garantizar, en el 99.9% de los accesos, un tiempo de respuesta menor de 300ms, que provee a los clientes una experiencia “siempre en línea”, sin caídas ni restricciones del servicio. Esto quiere decir que la base de datos ha de responder en ese lapso de tiempo aunque un nodo esté caído, se corte el suministro eléctrico en un centro de datos (es decir, que se caigan 300 nodos) o, tal y como menciona el capítulo, “un tornado esté destruyendo centros de datos” (o lo que es lo mismo, que se pierdan esos 300 nodos).

Contexto y Requisitos en Amazon

Dentro del contexto tecnológico estos servicios de almacenamiento operan como se detallará de la siguiente manera según (DeCandia, y otros, 2007):

- ✓ La infraestructura está compuesta por decenas de miles de servidores y componentes de red ubicados en muchos centros de datos de todo el mundo.
- ✓ Se utiliza el hardware de productos básicos.
- ✓ Fallo de un componente es el " modo de operación estándar”.
- ✓ "Amazon utiliza un acoplamiento flexible, arquitectura orientada a servicios altamente descentralizada, que consta de cientos de servicios”.

Además de estos factores tecnológicos, el diseño de Dynamo también está influenciada por consideraciones de negocios.(DeCandia, y otros, 2007)

- ✓ Estricto, los acuerdos de nivel de servicio (SLA internos) con respecto a "rendimiento, fiabilidad y eficiencia" se han de cumplir en el 99,9 por ciento del distribution. Decandia et al. Considerar "la administración del estado" como los ofrecidos por Dynamo y las otras bases de datos como cruciales para cumplir con

estos SLAs en un servicio cuya lógica de negocio se encuentra en la mayoría de los casos bastante ligero en Amazon (DeCandia, y otros, 2007)

- ✓ Uno de los requisitos más importantes en Amazon es la fiabilidad "porque la más mínima interrupción tiene consecuencias financieras importantes e impactos confianza de los clientes".
- ✓ "Para apoyar el crecimiento continuo, la plataforma debe ser altamente escalable".

Otros principios clave abrazado en el diseño son:

Escalabilidad incremental: Dynamo debe ser capaz de escalar un host de almacenamiento (en adelante, denominado " nodo") a la vez, con un impacto mínimo en los dos operadores del sistema y el sistema mismo.

Simetría: Todos los nodos Dynamo debe tener el mismo conjunto de responsabilidades que sus compañeros; no debe haber ningún nodo distinguido o nodos que tienen funciones especiales o juego extra de responsabilidades. En nuestra experiencia, la simetría simplifica el proceso de aprovisionamiento y mantenimiento del sistema.

Descentralización: Una extensión de la simetría, el diseño debe favorecer técnicas descentralizados peer to peer más de un control centralizado. En el pasado, el control centralizado ha dado lugar a cortes y el objetivo es evitar tanto como sea posible. Esto conduce a un sistema más sencillo, más escalable, y más disponible.

Heterogeneidad: El sistema debe ser capaz de explotar la heterogeneidad en la infraestructura que se ejecuta por ejemplo el trabajo distribución debe ser proporcional a las capacidades de los servidores individuales. Esto es esencial en la adición de nuevos nodos de mayor capacidad sin tener que actualizar todos los servidores a la vez.

Ventajas y desventajas

TABLA 8: Amazon's Dynamo – Ventajas y desventajas

Ventajas	**Desventajas**

"No Master" Operaciones "Altamente disponible para escritura" "Tiradores de tuning lee" (al igual que escribe) "Simple"	"Propietario a Amazon" "Los clientes necesitan ser inteligentes" (los relojes de vectores soporteyla resolución de conflictos, el equilibrio racimos) "Sin compresión" (aplicaciones cliente pueden comprimirlos valorespropios, las claves no pueden ser comprimidos) "No es adecuado para cargas de trabajo deforma de columna" "Sólouna base de almacenamiento declave / valor" (por ejemplo,las búsquedas por rangosolas operaciones por lotesno son posibles)

Muchos de estos pares clave / valor tienen APIs que permiten conseguir y establecer mecanismos para obtener y establecer valores. Unos pocos, como Redis[18] (http://redis.io/), proporcionan más ricas abstracciones y las API de gran alcance.

Redis podía ser considerada como una estructura de datos del servidor, ya que proporciona estructuras de datos, como secuencias de caracteres (string), listas y conjuntos, además de los mapas.

✓ **Otras Bases de Datos Clave/Valor (Key/Value)**

Redis

Es relativamente un nuevo almacén de datos que sus desarrolladores se refieren no específicamente como un " almacén de estructura de datos ", sino que comúnmente se subsume bajo bases de datos almacenamiento Clave/Valor (key/value) debido a su mapa / diccionario API. Algo especial sobre Redis es que permite juegos para clave rangos por ejemplo.Adecuación de los rangos numéricos o expresiones regulares. En contraste con otros almacenamientos key/value, Redis no sólo almacena bytes como valores sino que también permite las listas y conjuntos en valores mediante el apoyo a ellos directamente. Una desventaja importante sobre Redis es que la cantidad de

[18] **Redis:** Es un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente. **Fuente:** http://es.wikipedia.org/wiki/Redis

memoria principal limita la cantidad de datos que es posible almacenar. Esto no se puede ampliar por el uso de discos duros. (Ippolito, 2009) Comenta que por esta razón Redis es probablemente una buena opción para una capa de almacenamiento en caché.

Además, Redis proporciona un conjunto muy rico de operaciones para acceder a los datos de estos diferentes tipos de estructuras de datos.

Tokyo Cabinet and Tokyo Tyrant

Este almacén de datos se basa en una colección de componentes de software, de los cuales el Consejo de Ministros de Tokio y Tokyo Tyrant son los más importantes en este contexto. Tokyo Cabinet (Labs, 2010) es la biblioteca central de este almacén de datos persistente de datos y la exposición de una interfaz key/value a los clientes y por lo tanto haciendo abstracción de las estructuras internas de datos como tablas hash o Btree índices. Tokio Tyrant (Labs, 2010) proporciona acceso a esta biblioteca de base de datos a través de la red que es posible a través de un protocolo binario de propietario, HTTP, así como a través del protocolo de memcached. “Tokio Gabinete gestiona el almacenamiento de datos en el disco y en la memoria de una manera similar a la paginación / intercambio. Páginas de memoria se vacían en el disco periódicamente, que deja un agujero abierto pérdida de datos", como comentarios (Hoff, 2009). Por otro lado, Tokio Gabinete permite comprimir páginas por el LZW algoritmo que puede lograr buenas relaciones de compresión (ver por ejemplo (Lin, 2009)).

Tokyo Cabinet tiene datos de la partición de forma automática y por lo tanto tiene que ser replicado con una estrategia similar a MySQL. Además de las búsquedas por clave que puede coincidir con los prefijos y los rangos si se ordenan las llaves. En cuanto a las características transaccionales pena mencionar Tokyo Cabinet proporciona un registro de escritura anticipada y paginación en la sombra. La suite de Tokio se desarrolla activamente, bien documentado y ampliamente considerado como de alto performant: 1.000.000 registros se pueden almacenar en 0,7 segundos utilizando el motor de tabla hash y en 1,6 segundos utilizando el árbol b de acuerdo a (Ken, 2009)

Memcached and MemcacheDB

La solución de la memoria caché de escritura popular y rápida ampliamente utilizado entre los sitios web grandes y ultra gran escala para reducir la base de datos de carga, Servidores Memcached como las almacenamiento de clave / valor se tratan en este capítulo proporcionan un mapa / API diccionario consta de las operaciones get, put y

remove. Aunque no es la intención para el almacenamiento persistenteno es una solución existente denominado MemcacheDB (Steve, 2009) que cumpla con el protocolo de memcached y agrega la persistencia basado en Berkeley DB. Como memcached no proporciona replicación entre nodos y tampoco es tolerante con los fallos de la máquina, simplemente añadiendo un almacenamiento persistente de que no es suficiente, como blogger Richard Jones, de las declaraciones Last.fm. Señala que soluciones como repcached pueden replicar servidores memcached enteros (en una configuración maestro-esclavo), pero sin Particionamiento tolerante a fallos que provocarán los esfuerzos de manejo y mantenimiento (Richard, 2009).

Scalaris

Es un base de almacenamiento key-/value escrito en Erlang aprovechando de este idioma enfoque por ejemplo en la implementación de un protocolo de no bloqueo para cometer transacciones atómicas. Scalaris utiliza una versión adaptada del servicio acorde (Strauch, 2012)para exponer una tabla hash distribuida a los clientes. Ya que almacena las claves en orden lexicográfico, son posibles las búsquedas por rangos de prefijos. En contraste con otras bases de almacenamiento key-/value, Scalaris tiene un modelo de consistencia estricta, proporciona la replicación simétrica y permite consultas complejas (a través de bibliotecas de lenguaje de programación).

Garantiza propiedades ACID también para transacciones simultáneas mediante la implementación de una versión adaptada del protocolo consensuado Paxos (Leslie, 2013) . Como memcached, Scalaris es un puro en la memoria base de almacenamiento key/value (cf.(North, 2013)).

2.9.2 BASES DE DATOS ORIENTADAS A DOCUMENTOS

Bases de datos de documento u Orientados a documentos no son sistemas de gestión de documentos. Los desarrolladores partiendo de administración de base de datos no deben confundir las NoSQL Orientadas a Documentos con un sistema de contenido de documentos. El documento de Word en bases de datos de documento connota vagamente estructurados conjuntos de clave/valor pares en documentos, normalmente JSON (notación de objetos JavaScript) y no los documentos u hojas de cálculo (aunque estos podrían almacenarse demasiado).

Las bases de datos de documento tratan un documento conjunto y evitan dividir un documento en sus constituyentes pares de nombre/valor. A nivel de una colección, esto permite armar un conjunto diverso de documentos en una sola colección. Las Bases de

datos de documento permiten la indización de documentos sobre la base de no sólo de su principal campo de identificación sino también de sus propiedades.

Hoy en día algunas bases de datos de documento disponibles son **MongoDB** y **CouchDB**.

- ✓ **Bases De Datos Couchdb**

Ofrece otra interesante propuesta para el movimiento de NoSQL. La mayoría de las aplicaciones son básicamente offline y necesitan facilidades de replicación transparentes. CouchDB usa una estructura de almacenamiento de Árbol-B, operaciones de sólo agregado con un control de concurrencia basado en el modelo MVCC[19], operaciones sin bloqueos, API REST y operaciones incrementales Map/Reduce. Chris Anderson, uno de los desarrolladores líderes de CouchDB, resume el valor de CouchDB en el mundo web de hoy.

CouchDB es una base de datos documental escrito en Erlang. El nombre CouchDB está hoy en día a veces se refiere a " Clúster de poco fiable hardware commodity " base de datos , que es de hecho una backronym según una de las principales promotoras es (cf. (Pritlove, Lehnardt, & Lang, 2009)) .

CouchDB se puede considerar como un descendiente de Lotus Notes, Una gran cantidad de conceptos de Lotus Notes se puede encontrar en CouchDB: documentos, puntos de vista, la distribución y la replicación entre servidores y clientes.

El enfoque de CouchDB es construir una base de datos de dicho documento a partir de cero con las tecnologías de la zona como la web de transferencia de fase representacional JavaScript Object Notation (JSON) como un formato de intercambio de datos, y la capacidad de integrar con componentes de infraestructura, como balanceadores de carga y servidores proxy de almacenamiento en caché, etc (cf. (Strauch, 2012) (Pritlove, Lehnardt, & Lang, 2009)) .

CouchDB puede caracterizarse brevemente como una base de datos documental que se puede acceder a través de un HTTP interface RESTful, que contiene los documentos de esquema libre en un espacio de dirección plana. Para estos documentos de JavaScript funciones documentos y representaciones de ellas selectas y agregados de manera MapReduce para construir vistas de la base de datos que también obtienen indexación.

[19]**MVCC:** Multiversion concurrency control o MVCC) es un método para control de acceso generalmente usado por SGBDs para proporcionar acceso concurrente a los datos, y en lenguajes de programación para implementar concurrencia. **Fuente**: http://es.wikipedia.org/wiki/Mvcc

CouchDB se distribuye y es capaz de replicar entre nodos de servidor, así como clientes y servidores de forma incremental. Versiones concurrentes múltiples de un mismo documento (MVCC) están permitidos en CouchDB y la base de datos es capaz de detectar los conflictos y gestionar su resolución en la que se delega a las aplicaciones cliente.

(Strauch, 2012)Dice que el uso más notable de CouchDB en la producción es uno Ubuntu el almacenamiento en la nube y el servicio de replicación para Ubuntu Linux. CouchDB es también parte de la nueva plataforma de aplicaciones web de la BBC, Además, algunos blogs (menos prominentes) , wikis , redes sociales, aplicaciones de Facebook y los sitios web más pequeños utilizan CouchDB como su almacén de datos (cf. (Czura, 2010)).

Características Principales

Almacenamiento de documentos: CouchDB almacena los datos como "documentos", esto es, uno o más pares campo/valor expresados en JSON. Los valores de los campos pueden ser datos simples como cadenas de caracteres, números o fechas. Pero también se pueden usar listas ordenadas y vectores asociativos. Todos los documentos en una base de datos CouchDB tienen un identificador único y no requieren un esquema determinado.

Semántica ACID: CouchDB provee una semántica de atomicidad, consistencia, aislamiento y durabilidad. Lo hace implementando una forma de control de concurrencia multiversión, lo que significa que CouchDB puede manejar un gran número de lectores y escritores en paralelo, sin que surjan conflictos.

Vistas e índices Map/Reduce: Los datos almacenados se estructuran por medio de vistas. En CouchDB, cada vista se construye por medio de una función JavaScript que actúa como la mitad Map de una operación map/reduce. La función recibe un documento y lo transforma en un único valor, retornándolo. CouchDB puede indexar vistas y mantener actualizados esos índices a medida de que se agregan, eliminan o actualizan documentos.

Arquitectura distribuida con replicación: CouchDB se diseñó con teniendo en mente la replicación bidireccional (o sincronización) y la operación off-line. Eso significa que múltiples réplicas pueden tener cada una sus propias copias de los mismos datos, modificarlas y luego sincronizar esos cambios en un momento posterior.

Interfaz REST: Todos los ítems tienen una URI única que queda expuesta vía HTTP. REST usa los métodos HTTP POST, GET, PUT y DELETE para las cuatro operaciones básicas CRUD (Create, Read, Update, Delete) con todos los recursos.

Consistencia Eventual: CouchDB garantiza consistencia eventual para poder ofrecer tanto disponibilidad como tolerancia a las particiones.

Hecha para operar offline: CouchDB puede replicar datos a dispositivos (como smartphones) que pueden quedar offline y manejar automáticamente la sincronización de los datos cuando el dispositivo vuelve a estar en línea.

CouchDB también ofrece una interfaz de administración incorporada y accesible vía web llamada Futon.

Futonproporciona acceso completo atodas las funciones deCouchDByhace que seafácil de trabajarcon algunas de lasideas más complejasinvolucradas.ConFutonpodemos creary destruirlas bases de datos; ver y editar documentos, redactar yejecutarvistasMapReduce, y la replicaciónde disparoentre bases de datos.

- ✓ **Base De Datos Mongodb**

Es una más de las bases de datos NOSQL que también ofrece almacenamiento de documentos. No brinda soporte REST[20], pero se basa en un almacenamiento JSON. También tiene Map/Reduce, y ofrece un poderoso conjunto de APIs de consulta muy parecido a SQL.

Este es un punto clave de MongoDB, que resulta muy cómodo a personas que tienen conocimientos de SQL. MongoDB también ofrece replicación maestro-esclavo, y están trabajando en escalabilidad basada en autosharding y en tolerancia a fallos.

MongoDB es una base de datos documental sin esquema escrito en C + + y se desarrolló en un proyecto de código abierto que es impulsado principalmente por la empresa 10gen Inc, que también ofrece servicios profesionales en torno a MongoDB. (Strauch, 2012)

De acuerdo a sus desarrolladores la meta principal de MongoDB es cerrar la brecha entre los almacenamientos key/value rápidos y altamente escalables y sistemas de gestión de múltiples funciones tradicionales RDBMS bases de datos relacionales.

[20] **REST:** Es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Fuente: http://es.wikipedia.org/wiki/REST

Según el artículo publicado por la organización de MongoDB (MongoDB Inc, 2014) Se trata de una base de datos ágil que permite a los esquemas cambiar más rápidamente para que así las aplicaciones evolucionen, sin dejar de ofrecer a los desarrolladores la funcionalidad que esperan de las bases de datos tradicionales, como los índices secundarios, con un lenguaje de consulta de plena y estricta consistencia.

Estructura De Mongodb

Conjunto de colecciones dinámicas, solo existen si hay almacenadas en ellas, colecciones o documentos.

Colecciones: lo que sería una tabla para las bases de datos relacionales. Están compuestas de documentos, estos no se someten a esquemas fijos.

Documentos: son un conjunto de líneas en formato JSON.

Cada documento se almacena en formato BSON. BSON es una representación codificada en binario de un formato de documento de tipo JSON donde la estructura está cerca de un conjunto anidado de clave /valor pares. BSON es un súper-conjunto de JSON y soporta tipos adicionales como expresiones regulares, datos binarios, y la fecha. Cada documento tieneun identificador único, que MongoDBpuede generar, si nose especifica explícitamenteal insertarlos datosen una colección, este genera un id de objeto automáticamente como se muestra en la figura 2-5

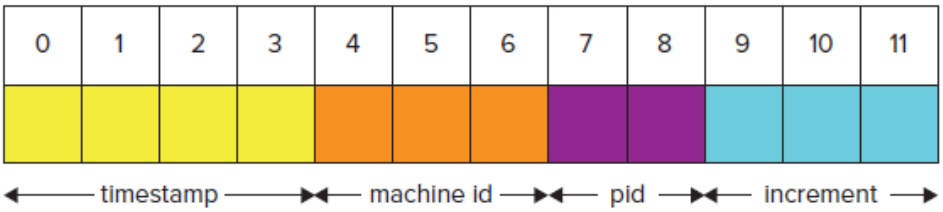


FIGURA 7: Demostración – Generar Id en Colecciones.

Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos.

BSON es un formato de intercambio de datos usado principalmente para su almacenamiento y transferencia en la base de datos MongoDB. Es una representación binaria de estructuras de datos y mapas. El nombre BSON está basado en el término JSON y significa Binary JSON (JSON Binario).

Comparado a JSON, BSON está diseñado para tener un almacenamiento y velocidad más eficiente. Los elementos largos contienen el campo “tamaño” para facilitar su escaneo, lo que provoca que en algunos casos BSON use más espacio en memoria que JSON.(WIKIPEDIA, 2013)

Esta base de datos se construye para la escalabilidad, el rendimiento y la alta disponibilidad, la ampliación de las implementaciones de servidores individuales hasta grandes y complejas arquitecturas multi-sitio. Mediante el aprovechamiento in-memory computing, MongoDB proporciona un alto rendimiento tanto para lecturas y escrituras. Replicación nativa de MongoDB y failover automatizado permiten la confiabilidad de nivel empresarial y flexibilidad operativa.MongoDB es de código binario está disponible para los sistemas operativos Windows, Linux, OS X y Solaris, esta base de datos es adecuada para la el desarrollo de aplicaciones en los siguientes ámbitos:

- ✓ Almacenamiento y registro de eventos
- ✓ Para sistemas de manejo de documentos y contenido
- ✓ Comercio Electrónico
- ✓ Juegos
- ✓ Problemas de alto volumen de lecturas
- ✓ Aplicaciones móviles
- ✓ Almacén de datos operacional de una página Web
- ✓ Manejo de contenido
- ✓ Almacenamiento de comentarios
- ✓ Votaciones
- ✓ Registro de usuarios
- ✓ Perfiles de usuarios
- ✓ Sesiones de datos
- ✓ Proyectos que utilizan metodologías de desarrollo iterativo o ágiles
- ✓ Manejo de estadísticas en tiempo real

MongoDB es utilizado para uno o varios de estos casos por varias empresas

Características Principales

Lo siguiente es una breve descripción de las características principales de MongoDB:

Consultas Ad hoc

MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento pero también puede ser una función JavaScript definida por el usuario.

Indexación

Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.

Replicación

MongoDB soporta el tipo de replicación maestro-esclavo. El maestro puede ejecutar comandos de lectura y escritura. El esclavo puede copiar los datos del maestro y sólo se puede usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. El esclavo tiene la habilidad de poder elegir un nuevo maestro en caso de que se caiga el servicio con el maestro actual.

Balanceo de carga

MongoDB se puede escalar de forma horizontal usando el concepto de “shard”. El desarrollador elige una llave shard, la cual determina cómo serán distribuidos los datos en una colección.

Los datos son divididos en rangos (basado en la llave shard) y distribuidos a través de múltiples shard. Un shard es un maestro con uno o más esclavos. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y nuevas máquinas pueden ser agregadas a MongoDB con el sistema de base de datos corriendo.

Almacenamiento de archivos

MongoDB puede ser utilizado con un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos. Esta función (que es llamada GridFS) está incluida en los drivers de MongoDB y disponible para los lenguajes de programación que soporta MongoDB. Esta base de datos expone

funciones para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiple servidores, los archivos pueden ser distribuidos y copiados entre los mismos varias veces y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.

Agregación

La función MapReduce puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación. Esta función permite que los usuarios puedan obtener el tipo de resultado que se obtiene cuando se utiliza el comando SQL “group-by”.

Ejecución de JavaScript del lado del servidor

MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Ventajas Y Desventajas

TABLA 9: MongoDB – Ventajas y Desventajas

Ventajas	**Desventajas**
No requiere operaciones JOIN Escala horizontalmente Dispone de MapReduce	Se le considera que es no es producto maduro. No escala con herramientas de Business Intelligence.
Uso de memoria en vez de disco Alta frecuencia de escritura y lectura Cambios de esquema de datos frecuente	Requiere conocimientos de programación
Utilizan estructura de datos sencilla, tipo clave/valor AutoSharding	La migración de datos es casi imposible de un proveedor a otro. Restringe el tamaño de datos a un máximo de 2GB en sistemasde 32 bytes

Escalabilidad Horizontal

Una buena razón para utilizar MongoDB es su esquema menos colecciones y el otro es su capacidad inherente para un buen desempeño y la escala. En las versiones más recientes, MongoDB soporta auto - sharding para escalar horizontalmente con facilidad.

El concepto fundamental de sharding es bastante similar a la idea de patrón masterworker de la base de datos de la columna donde los datos se distribuyen a

través de múltiples servidores de alcance. MongoDB permite colecciones ordenadas a través de múltiples máquinas. Cada máquina que guarda parte de la colección es entonces un fragmento. Fragmentos se replican para permitir la conmutación por error. Así, una gran colección podría dividirse en cuatro fragmentos y cada fragmento a su vez puede ser replicado tres veces. Esto crearía 12 unidades de un servidor MongoDB. Las dos copias adicionales de cada fragmento sirven como unidades de conmutación por error.

Los fragmentos están en el nivel de colección y no a nivel de base de datos. Por lo tanto, una colección en una base de datos, puede residir en un solo nodo, mientras que otras en la misma base de datos pueden ser fragmentadas a varios nodos. Cada fragmento almacena conjuntos contiguos de los documentos solicitados. Estos paquetes se denominan trozos en MongoDB. Cada fragmento es identificado por tres atributos, a saber, la clave del primer documento (clave min), la última clave documento (clave MAX), y la colección.

Una colección puede ser fragmentadabasándose en cualquier patrón de clave de fragmento válido. Cualquier campo del documento de una colección o una combinación de dos o más campos del documento en una colección se puede utilizar como la base de una clave de fragmento. Claves Shard también contienen una propiedad de dirección para además del campo para definir una clave de fragmento. La dirección de la orden puede ser de 1, es decir, ascendente o -1, que significa descendente. Esimportante elegir las claves de fragmento con prudencia y asegurarse de que esas claves pueden dividir los datos de una manera equilibrada.

Todas las definiciones acerca de los fragmentos y los trozos que mantienen se mantienen en los catálogos de metadatos en un servidor de configuración. Al igual que los fragmentos, servidores de configuración también se replican en el soporte de failover.

Los procesos de cliente llegan a un clúster MongoDB a través de un proceso de mongo.

Un proceso mongo no tiene un estado persistente y tira de estado de los servidores de configuración. Puede haber uno o más mongos procesos para un clúster de MongoDB. Procesos Mongos tienen la responsabilidad de encaminar las consultas de manera adecuada y la combinación de resultados en los que se requiere. Una consulta a un clúster MongoDB puede ser objetivo o puede ser global. Todas las consultas que pueden aprovechar el clave fragmento en la que los datos suelen ordenar consultas

específicas y aquellas que no pueden aprovechar el índice son globales. Consultas dirigidas son más eficientes que las consultas globales. Piense en las consultas globales como las que implican exploraciones colección completa.

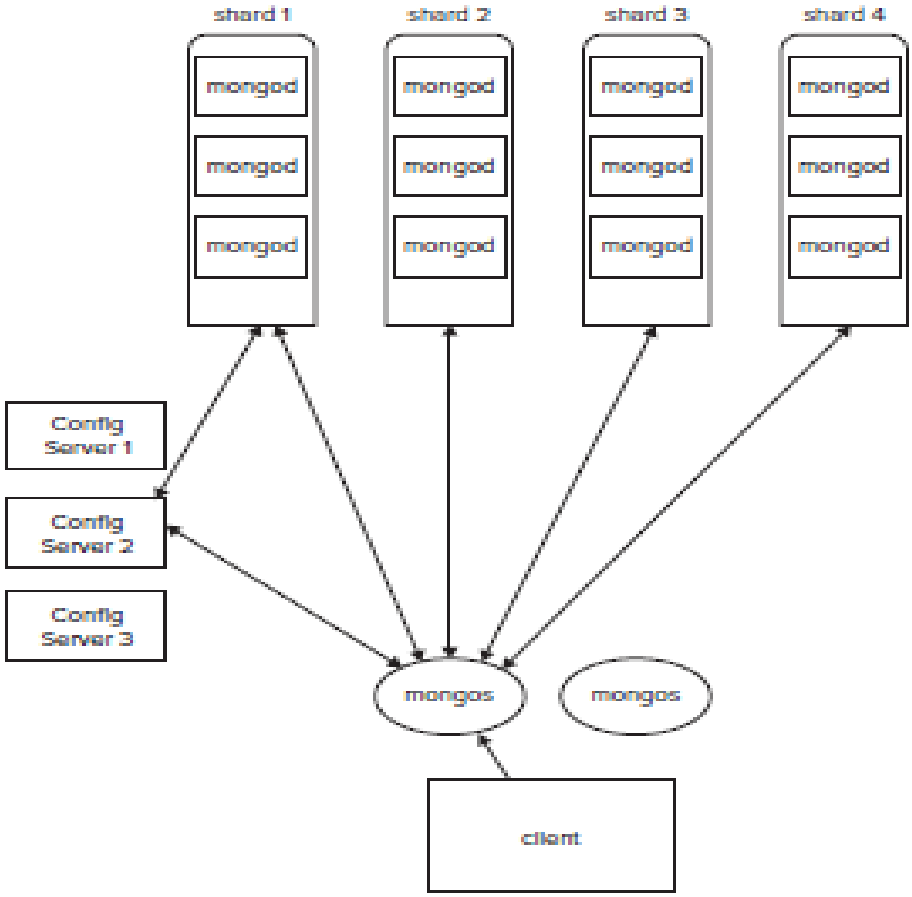


FIGURA 8: MongoDB Escalabilidad Horizontal - topología de arquitectura sharding para MongoDB (Tiwari, 2011)

2.9.3 BASES DE DATOS ORIENTADAS A GRAFOS

Hasta ahora he enumerado la mayoría de los principales productos de NoSQL abierto. Algunos otros productos como Almacenes de datos XML y bases de datos de gráfico también podrían calificar como bases de datos NoSQL.

“Las bases de datos orientadas a grafos (BDOG) representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos ya que esta puede describir atributos de los nodos (entidades) y las aristas (relaciones)” (Wikipedia.org, 2013).

Una base de datos orientada a grafos debe estar absolutamente normalizada, esto quiere decir que cada tabla tendría una sola columna y cada relación tan solo dos, con esto se consigue que cualquier cambio en la estructura de la información tenga un efecto tan solo local.

Básicamente en la teoría de grafos (también llamada teoría de las gráficas) es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos (también llamadas gráficas, que no se debe confundir con las gráficas que tienen una acepción muy amplia) estructuras que constan de dos partes, el conjunto de vértices, nodos o puntos; y el conjunto de aristas, líneas o lados (edges en inglés) que pueden ser orientados o no.

La teoría de grafos es una rama de la Matemática discreta y de las aplicadas, y es un tratado que usa diferentes conceptos de diversas áreas como Análisis combinatorio, Álgebra abstracta, probabilidad, geometría de polígonos, aritmética y topología.

Actualmente ha tenido mayor preponderancia en el campo de la informática, las ciencias de la computación, telecomunicaciones.

La teoría de grafos también ha servido de inspiración para las ciencias sociales, en especial para desarrollar un concepto no metafórico de red social que sustituye los nodos por los actores sociales y verifica la posición, centralidad e importancia de cada actor dentro de la red. Esta medida permite cuantificar y abstraer relaciones complejas, de manera que la estructura social puede representarse gráficamente. Por ejemplo, una red social puede representar la estructura de poder dentro de una sociedad al identificar los vínculos (aristas), su dirección e intensidad y da idea de la manera en que el poder se transmite y a quiénes.

Características

Estas bases de datos pueden resolver fácilmente algunos problemas que se presentan en otros modelos de base de datos, por ejemplo: la combinación de atributos multi-valor y atributos complejos, flexibilidad a cambios de estructura especialmente cuando la fuente de datos es autónoma y dinámica como el Internet, unificación en la representación de los datos, esquemas y consultas.

Algunas BDOG pueden recibir o devolver grafos completos de acuerdo a diferentes criterios de búsqueda.

Consultas más amplias y no demarcadas por tablas.

No hay que definir un número determinado de atributos, esto quiere decir que una persona puede tener relacionados 4 nombres mientras que otra tan solo 2, sin malgastar espacio.

Los registros también son de longitud variable, evitando tener que definir un tamaño y también posibles errores en la base de datos.

Se puede recorrer directamente la base de datos de forma jerárquica, obtener el nodo abuelo del nodo y viceversa.

Isomorfismo: es de vital importancia en las BDOG, ya que este término se refiere a cuando se va a introducir un grafo a la base de datos y este ya se encuentre en la misma, o se pueda obtener de permutar otro grafo. El poder identificar estas situaciones puede ayudar a ahorrar espacio de almacenamiento.

Ventajas

Las también ofrecen servicios nuevos o mejorados como:

Consultas más amplias y no demarcadas por tablas, ejemplo “Muestre todas las tablas que posean un nombre Carlos”.

No hay que definir un número determinado de atributos, esto quiere decir que una persona puede tener relacionados 4 nombres mientras que otra tan solo 2, esto sin desperdiciar espacio.

Los registros también son de longitud variable, evitando tener que definir un tamaño y también posibles fallas en la base de datos.

Se puede recorrer directamente la base de datos de forma jerárquica, obtener el nodo abuelo del nodo y viceversa.

Esta tesis incluirá las bases de datos de gráfico que pueden ser de interés y algo que quiera explorar más allá: Neo4j y FlockDB.

- ✓ **Neo4J**

Es una base de datos de gráfico compatible con ACID. Facilita el rápido recorrido de gráficos.“Los desarrolladores describen a Neo4j como un motor de persistencia

embebido, basado en disco, completamente transaccional Java que almacena datos estructurados en grafos más que en tablas"[21]

Neo4j fue desarrollado por Neo Technology, una startup sueca con base en Malmö y San Francisco Bay Area en Estados Unidos. El Consejo de administración de Neo Technology consta de: Magnus Christerson (Vicepresidente de Intentional Software Corp.), Nikolaj Nyholm (CEO de Rosa polar), Sami Ahvenniemi (socio de Conor Venture Partners) y Johan Svensson (CTO de Neo Technology).

En unos artículo (Solis, 2014) encontrado en internet nos cita un ejemplo el cual describimos en los siguientes reglones.

Pensemos por ejemplo en Facebook, y cómo podría ser su modelo de datos utilizando grafos para una pequeña red de amigos, y en donde la semántica de la relación "amigo de" es fundamental para poder establecer el contexto, esta relación queda establecida por la flecha que une cada una de las cajas siguientes:

FIGURA 9: Redes Sociales – Representación en grafos

En este grafo, Carlos y Sara se consideran amigos mutuos, del mismo modo que Carolina y Sara, y aunque Carlos ha establecido que Carolina es su amiga, ésta aún no ha respondido recíprocamente a esta amistad. Obviamente el modelo de Facebook, es millones de veces más grande que este que acabamos de presentar aquí, pero este pequeño ejemplo permite hacernos una idea de lo que estamos hablando y de su potencia. Esta potencia la vemos enseguida con el siguiente ejemplo, la relación "me gusta" es muy sencilla de implementar dentro del grafo anterior.

[21]Referencia sacada de la Fuente: http://es.wikipedia.org/wiki/Neo4j

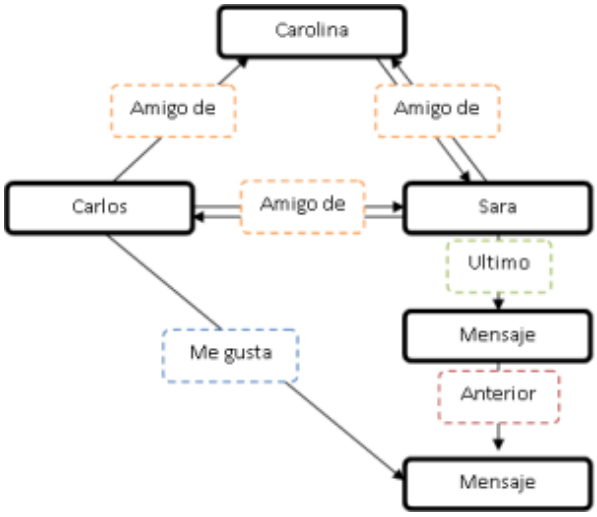


FIGURA 10: Redes Sociales – representación en grafos 2

Si consideramos que Sara ha dado de alta una serie de mensajes que actualizan su estado, y que de estos mensajes el último se enlaza con Sara mediante una relación "último" y los mensajes anteriores mediante una relación "anterior", establecer la relación "me gusta" sobre un mensaje es algo tan sencillo como:

Y en este ejemplo, a Carlos le gusta un mensaje particular de su amiga, pero no ha establecido ninguna opinión sobre el resto de estados emitidos por ella.

Una base de datos basada en grafos debe permitir tanto el modelado de un escenario como el anteriormente establecido, como proporcionar las herramientas necesarias para su consulta y análisis.

Características

Neo4j es una base de datos interesante de investigar ya que al igual que las bases de datos NoSQL, esta resurge con mayores beneficios y prestaciones. Entre las más notables características:

- ✓ Manejo de complejidad
- ✓ Capacidad de almacenamiento de objetos
- ✓ API REST
- ✓ Lenguaje de Consulta más amigables.
- ✓ Neo4j como backend de diferentes frameworks

Entre las características podemos destacar que también utiliza Transaccionalidad ACID.

Manejo De Complejidad

La mayoría de las aplicaciones no sólo tendrán que escalar a una enorme volúmenes, sino también escalar a la complejidad del dominio a la mano. Típicamente, puede haber muchas entidades interconectadas y propiedades opcionales. Incluso los dominios simples pueden ser difíciles de manejar debido a las consultas que desea ejecutar en ellos, por ejemplo, para encontrar caminos. Dos ejemplos de codificación son el ejemplo de red social (aplicación parcial Rubí) y el ejemplo Neo4j IMDB (variación del código Rubí).

Almacenamiento De Objetos

Público de clase Person {

/ / Utilizado por jo4neo

Transitoria nodo NodeID;

/ / Propiedad simple

@ Neo primerNombre String;

/ / Le ayuda a almacenar un java.util.Date a Neo4j

@ Neo fecha;

/ / Jo4neo índice voluntad para usted

@ Neo (index = verdadero) de correo electrónico de cuerda;

/ / Relación de muchos a muchos

@ Neo papeles Colección <role>;

/ * Clase normales orientado

* Cosas de programación va aquí

* /

} </ Role>

Otra forma de persistencia de objetos es utilizando el neo4j.rb envoltorio Neo4j para Ruby. Tiempo para unas pocas líneas de código de ejemplo de nuevo:

requerir "rubygems"

requerir "Neo4j"

```
clase Person

 incluir Neo4j :: NodeMixin

 # Define propiedades Neo4j

 propiedad : Nombre , : salario , : edad , : country

  # Define una relación de un modo a otro nodo

 has_n : amigos

  # Agrega un índice de Lucene en las siguientes propiedades

 índice : nombre , : salario , : edad , : País

end
```

Una característica de gran alcance de la utilización de una base de datos gráficaNeo4J, es que usted puede crear sus propias estructuras en gráfico de datos por ejemplo, una lista enlazada.

Esta estructura de datos utiliza un solo nodo que la referencia de la lista. La referencia tiene una relación de salida de la cabeza de la lista, y una relación de entrada desde el último elemento de la lista. Si la lista está vacía, la referencia que apunte a sí mismo.

Para dejar en claro lo que pasa, vamos a mostrar cómo el gráfico se ve después de cada consulta.

Para inicializar una lista enlazada vacía, simplemente creamos un nodo, y hacerlo enlazar a sí mismo. A diferencia de los elementos reales de la lista, que no tiene un valor de propiedad.

```
CREATE (root {name: 'ROOT'})-[:LINK]->(root)

RETURN root
```

El adición de los valores se realiza mediante la búsqueda de la relación en la que el nuevo valor debe ser colocado en, y su sustitución por un nuevo nodo, y dos relaciones a la misma. También para manejar el hecho de que los antes y después de los nodos podrían ser el mismo que el de la raíz nodo. El caso en el que antes, después y la raíz nodo son todos iguales, hace que sea necesario el uso de CREATE UNIQUE para no crear dos nuevos nodos de valor por error.

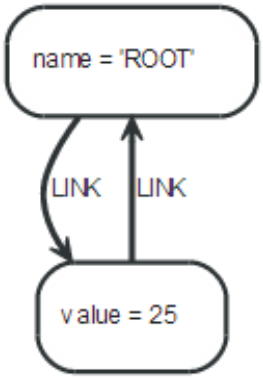
```
MATCH (root)-[:LINK*0..]->(before),(after)-[:LINK*0..]->(root),(before)-[old:LINK]->(after)

WHERE root.name = 'ROOT' AND (before.value < 25 OR before = root) AND (25 <
after.value OR after =

 root)

CREATE UNIQUE (before)-[:LINK]->({ value:25 })-[:LINK]->(after)

DELETE old
```



Vamos a añadir un valor más.

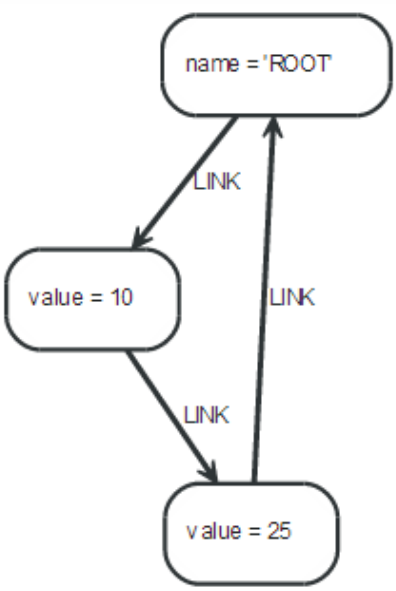
```
MATCH (root)-[:LINK*0..]->(before),(after)-[:LINK*0..]->(root),(before)-[old:LINK]->(after)

WHERE root.name = 'ROOT' AND (before.value < 10 OR before = root) AND (10 <
after.value OR after =

  root)

CREATE UNIQUE (before)-[:LINK]->({ value:10 })-[:LINK]->(after)

DELETE old
```



Eliminación de un valor, por el contrario, se hace al encontrar el nodo con el valor y las dos relaciones que entran y salen de ella, y la sustitución de las relaciones con uno nuevo.

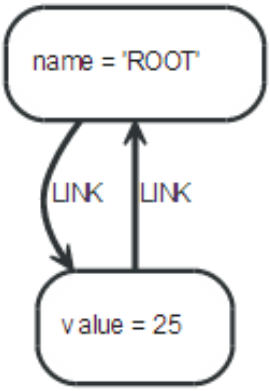
MATCH (root)-[:LINK*0..]->(before),(before)-[delBefore:LINK]->(del)-[delAfter:LINK]->(after),

(after)-[:LINK*0..]->(root)

WHERE root.name = 'ROOT' AND del.value = 10

CREATE UNIQUE (before)-[:LINK]->(after)

DELETE del, delBefore, delAfter



Eliminar el último nodo de valor es lo que nos obliga a utilizar CREATE UNIQUE al sustituir las relaciones.

De lo contrario, nos terminamos con dos relaciones de la raíz nodo a sí mismo, ya que tanto antes y después de los nodos son iguales a la raíz del nodo, es decir, el patrón podría coincidir dos veces.

```
MATCH (root)-[:LINK*0..]->(before),(before)-[delBefore:LINK]->(del)-[delAfter:LINK]-
>(after),

  (after)-[:LINK*0..]->(root)

WHERE root.name = 'ROOT' AND del.value = 25

CREATE UNIQUE (before)-[:LINK]->(after)

DELETE del, delBefore, delAfter
```

API REST

Rest: Utiliza los cuatro métodos HTTP GET, POST, PUT y DELETE para ejecutar diversas operaciones. Esto en contraste con SOAP para ejemplo, que crea nuevos comandos arbitrarios (verbos) como getAccounts () o applyDiscount (). (What is a REST API [duplicate], 2014).

Un API REST es un conjunto de operaciones que se puede invocar por medio de cualquier de los cuatro verbos, utilizando el URI real como parámetros para sus operaciones. Por ejemplo, puede tener un método para consultar todas sus cuentas de lo que se puede llamar desde / accounts / all / esta invoca un HTTP GET y el parámetro 'all' le dice a su aplicación que devolverá todas las cuentas.(Neo4j, 2014)

Lenguajes De Consulta

Neo4j.- utiliza los lenguajes de consulta gremlin, Cypher y SPARQ por el momento.

Gremlin.- Lenguaje script de grafos, es un lenguaje DSL (Domain Specific Languaje) de consulta de grafos con diferentes implementaciones de backend en las obras, así como un conjunto de herramientas de soporte.

Cypher.- Es un lenguaje declarativo "a la SQL", Cypher es el lenguaje de consultas que utiliza Neo4j para actuar sobre los datos que administra. Cypher está actualmente en desarrollo así que, para obtener mayor estabilidad en la utilización de Neo4j es mejor utilizar las API desarrolladas para interactuar con este.

SPARQ.- es un lenguaje de consulta de grafos RDF. Al igual que sucede con SQL, es necesario distinguir entre el lenguaje de consulta y el motor para el almacenamiento y recuperación de los datos. Por este motivo, existen múltiples implementaciones de SPARQL, generalmente ligados a entornos de desarrollo y plataforma tecnológicos.

En un principio SPARQL únicamente incorpora funciones para la recuperación sentencias RDF. Sin embargo, algunas propuestas también incluyen operaciones para el mantenimiento (creación, modificación y borrado) de datos.

Los enlaces de lenguaje que se puede utilizar ya que neo4j tiene su motor gráfico escrito en JAVA. Por lo que puede añadir fácilmente el archivo jar y empezar a utilizar la API simple y minimalista de inmediato.

- ✓ Python - ver PyCon 2010 también
- ✓ Rubí
- ✓ Clojure
- ✓ Scala
- ✓ Mapeo objeto Java

Casos de uso de Neo4j

Principales usos de los Neo4j incluyen redes sociales, la bioinformática, la detección del fraude, gestión de redes, autorización y control de acceso, gestión de contenido, y el enrutamiento de paquetes.

Las industrias que utilizan

- ✓ Telecomunicaciones
- ✓ Finanzas
- ✓ Ciencias de la Vida

- ✓ High Tech
- ✓ Media & Publishing
- ✓ Educación

FlockDB

Es una distribución de código abierto, tolerante a errores de base de datos gráfica para la gestión de las gráficas de la red de ancho pero poco profundo. Fue utilizado inicialmente por Twitter para almacenar las relaciones entre los usuarios, seguidoras y favoritas.FlockDB se diferencia de otras bases de datos del gráfico, por ejemplo, Neo4j, ya que no está diseñado para multi-hop gráfico de recorrido sino más bien para operaciones de conjuntos rápidos, no unlikethe caso de uso primario para sets Redis. Dado que aún se encuentra en proceso de ser embalados para su uso fuera de Twitter, el código sigue siendo muy peligrosa y por lo tanto no hay ninguna versión estable disponible todavía.

FlockDB es una base de datos gráfica distribuida para almacenar listas adyacentes, con las siguientes metas de apoyo:

- ✓ Una alta tasa de Agregar / actualizar / eliminar operaciones
- ✓ Consultas conjunto aritméticas complejas potientially
- ✓ Megafonía a través de conjuntos de resultados de consulta que contienen millones de entradas
- ✓ Capacidad de "archivo" y luego restaurar la forma archivados
- ✓ Escala horizontal incluyendo la replicación
- ✓ Migración de datos en línea

FlockDB es mucho más simple que otras bases de datos de gráficas, tales como Neo4j porque trata de resolver un menor número de problemas. Escala horizontal y está diseñado para baja latencia, entornos de rendimiento altos en línea, tales como sitios web.

Twitter utiliza FlockDB para almacenar gráficos sociales (que va detrás de quién, que bloquea a quien) y los índices secundarios.Segun el artículo publicado en abril de 2010, el almacenamiento Twitter FlockDB en los clúster es 13 + millones de bordes y sostiene los picos de demanda de 20k escribe/segundo y 100 mil lecturas/segundo.

Si, por ejemplo, usted está almacenando un gráfico social (el usuario A continuación el usuario B), y no es necesariamente simétrica (A puede seguir B sin B tras A), entonces FlockDB puede almacenar esa relación como una ventaja: los puntos de nodo de la A en el nodo B.

Almacena este borde con una posición de clase, y en ambas direcciones, de modo que pueda responder a la pregunta "¿Quién sigue a A?" , así como "quién es un siguiente?"

Esto se conoce como un grafo dirigido. (Técnicamente, FlockDB almacena las listas de adyacencia de un grafo dirigido.) Cada arista tiene un identificador de 64 bits de origen, un ID de destino de 64 bits, un estado (normal, eliminado, archivado), y una posición de 32 bits que se utiliza para clasificar. Los bordes se almacenan tanto en una dirección hacia adelante y hacia atrás, lo que significa que un borde se puede consultar basado en el origen o el ID de destino. (Kallen, Pointer, Kalucki, & Ceaser, 2014)

2.9.4 BASES DE DATOS ORIENTADAS A COLUMNAS

Como hemos visto, las bases de datos de almacenamiento de clave-valor y almacenamiento de gráficos tienen estructuras simples que son útiles para resolver una variedad de problemas de negocios. Ahora veamos cómo se puede combinar una fila y columna de una tabla que se utiliza como clave. Las bases de datos orientadas a columna son importantes patrones de arquitectura de las bases de datos NoSQL porque pueden escalar para manejar grandes volúmenes de datos. También son conocidos por estar estrechamente vinculada con muchos sistemas de MapReduce.

MapReduce es un marco para realizar el procesamiento paralelo en grandes conjuntos de datos en varios equipos (nodos). En el marco de MapReduce, la operación de mapa tiene un nodo maestro que rompe una operación en sub-partes y distribuye cada operación a otro nodo para el procesamiento y reducir es el proceso en el que el nodo maestro recoge los resultados de los otros nodos y los combina enla respuestaal problema original. Las bases de datos de familia de Columnas utilizan identificadores de fila y columna como propósitos generales claves para la búsqueda de datos. Algunos casos, se conocen como los almacenes de datos en lugar de bases de datos, ya que carecen de características que usted puede esperar encontrar en las bases de datos tradicionales. Por ejemplo, carecen de columnas con tipo, los índices secundarios, activadores y lenguajes de consulta. Casi todas las bases de almacenamientoorientado a columna han sido fuertemente influenciadas por el trabajo original Google Bigtable.

HBase, Hypertable y Cassandra son buenos ejemplos de sistemas que tienen interfaces Bigtablelike, aunque varía la forma en que se implementan.

Debemos tener en cuenta que las bases de datos orientadas a columna de término son distinto de una base de almacén de la columna.

Un almacén de columnas de datos almacena toda la información dentro de una columna de una tabla en la misma ubicación en el disco de la misma forma una fila de almacenamiento mantiene datos de fila juntos. El almacenamiento de columna se utiliza en muchos sistemas OLAP porque su fuerza es rápida CÁLCULO agregado columna. MonetDB, SybaseIQ y Vertica son ejemplos de sistemas de bases de datos de almacenamiento en columna.

Conceptos básicos de la bases de datos orientadaColumna

El primer ejemplo del uso de las filas y columnas como clave es la hoja de cálculo. Aunque la mayoría de nosotros no pensamos de hojas de cálculo como una tecnología NoSQL, sirven como una forma ideal de visualizar cómo las claves se pueden construir a partir de más de un valor.

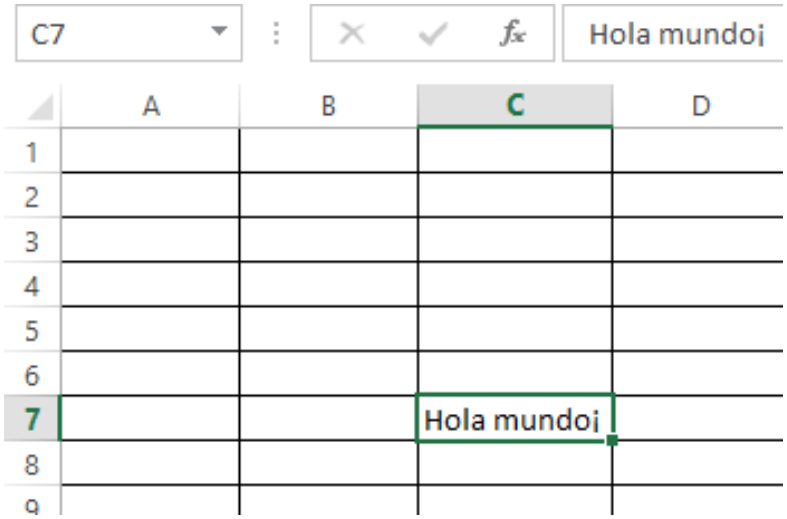


FIGURA 11: El uso de unafila y la columnadeabordaruna célula. La celda tiene una direcciónde**C7**y puedeser considerado comolaclave de búsquedaen unsistema dematriz dispersa.

La figura 11 muestra una hoja de cálculo con una sola celda en la fila 7 y la columna 3 (columna C) que contiene el texto " hola mundo" En una hoja de cálculo, se utiliza la combinación de un número de fila y una letra de la columna como una dirección a "

mirar hacia arriba " el valor de cualquier celda. Por ejemplo, la tercera columna de la séptima fila en la figura se identifica por la Clave. En contraste con el almacén de claves - valor, que tiene una sola clave que identifica el valor, una hoja de cálculo tieneun identificador de fila y columna que forman la clave. Pero al igual que el almacén de clave-valor, puede poner muchas cosas diferentes en una celda. Una celda puede contener datos, una fórmula, o incluso una imagen. El modelo para esto se muestra en la figura 12.

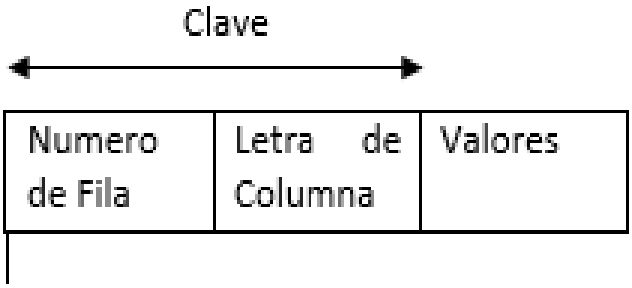


FIGURA 12: Hojas de cálculousan unpar de columnasconsecutivascomo clavepara buscar elvalor de una celda. Esto essimilar al uso deun sistema devalor de clave, donde lallave tienedospartes. Al igual queuna base de almacenamiento devalor de la clave, elvalor de una celdapuede tomaren muchos tipos, tales como cadenas, números o fórmulas

Esto es más o menos el mismo concepto en las bases de datos orientadas a columna. Cada elemento de datos sólo se puede encontrar por conocer información sobre los identificadores de fila y columna. Y, como una hoja de cálculo, puede insertar datos en cualquier celda en cualquier momento. A diferencia de un RDBMS, usted no tiene que insertar todos los datos de la columna de cada fila.

Entendiendo La Clave De Las Bases De Datos Orientas A Columnas

Para mayor comprensión de las claves se va añadir dos campos adicionales a las claves del ejemplo de hoja de cálculo. En la figura 13 se puede ver que hemos añadido una familia de columnas y marca de tiempo a la clave.

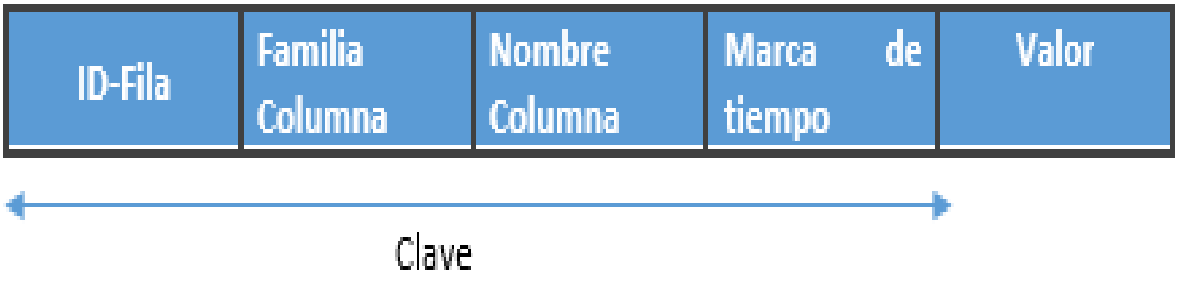


FIGURA 13: Familia de columnas y marca de tiempo a la clave

Laestructura clavede las bases de datos de las bases de datos orientadas a columnaes similar auna hoja de cálculo, pero tiene dosatributosadicionales. Además delnombre de la columna, una familia de columnase utiliza para agruparnombres de columnassimilares juntos. La adición deuna marca de tiempoen la clavetambiénpermite a cadacelda de la tablapara almacenarvarias versiones de unvalor con el tiempo.

La clave en la figura es típico de las bases de datos de columna. A diferencia de la hoja de cálculo típica, que podría tener 100 filas y 100 columnas,Las bases de datos orientadas a columnas están diseñadas para ser muy grandes.

¿De qué tamaño? Los sistemas con miles de millones de filas y cientos o miles de columnas no son desconocidos. Por ejemplo, un Sistema de Información Geográfica (GIS)como Google Earth podría tener un ID de fila de la parte de la longitud de un mapa y utilizar el nombre de columna para la latitud del mapa. Si usted tiene un mapa para cada kilómetro cuadrado de la Tierra, que podría tener 15.000 ID de filas distintas y 15.000 identificadores de columnas distintas.

Lo que es inusual acerca de estas grandes implementaciones es que si usted los ve en una hoja de cálculo, verías que algunas células contienen datos. Esta implementación matriz dispersa es una cuadrícula de valores de la que sólo un pequeño porcentaje de las células contienen valores. Por desgracia, las bases de datos relacionales no son eficientes en el almacenamiento de datos escasos, pero las bases de datos orientadas a columnas están diseñados exactamente para este propósito.

Con una base de datos relacional tradicional, puede utilizar una simple consulta SQL para buscar todas las columnas de cualquier tabla, al consultar los sistemas de matrices dispersas, usted debe buscar todos los elementos de la base de datos para obtener una lista completa de todos los nombres de columna. Un Problema que puede ocurrir con muchas columnas es que la ejecución de informes que enumeran las columnas y las columnas relacionadas puede ser difícil a menos que utilice una familia de columnas (una categoría de alto nivel de datos también conocidos como una ontología de nivel superior). Por ejemplo, es posible que los grupos de columnas que describen una página web, una persona, un lugar geográfico, y los productos para la venta.

Para poder ver estas columnas juntas, los agrupamos en la misma familia de columna para hacer la recuperación más fácil. No todas las bases de datos orientadas a columna utilizan una familia de columnas como parte de su clave. Si lo hacen, usted tendrá que tener esto en cuenta cuando se almacena una clave de tema, ya que la base orientada a columna forma parte de la clave, y la recuperación de datos no puede ocurrir sin él. En la medida que la API es simple, los productos NoSQL pueden escalar para manejar grandes volúmenes de datos, agregando nuevas filas y columnas sin necesidad de modificar un lenguaje de definición de datos.

Ventajas De Las Bases De Datos Orientas A Columna

El enfoque de las bases de datos orientadas a columna usan un identificador de fila y el nombre de la columna como clave de búsqueda es una forma flexible para almacenar datos, te da los beneficios de una mayor escalabilidad y disponibilidad, y le ahorra tiempo y molestias al añadir nuevos datos a su sistema.

Al leer estos beneficios, piense en los datos de su organización recoge para ver si una bases de datos orientada a columnas ayudaría a obtener una ventaja competitiva en su mercado.

Dado que las bases de datos orientados a columna no se basan en combinaciones, que tienden a escalar bien en sistemas distribuidos. Aunque usted puede comenzar su desarrollo en una sola computadora portátil, en las bases de datos orientadas a columnas de producción suelen ser configurados para almacenar datos en tres nodos distintos en posiblemente diferentes regiones geográficas (Diferentes centros de datos geográficos) para garantizar una alta disponibilidad. Las bases de datos orientados a Columna tienen **failover** automático incorporado para detectar nodos que fallan y algoritmos para identificar los datos corruptos. Se aprovechan de hash y de indexación herramientas avanzadas como filtros Bloom para realizar el análisis probabilístico de grandes conjuntos de datos. Cuanto mayor sea el conjunto de datos, estas herramientas mejor realizan su trabajo. Por último, las implementaciones de bases de datos orientadas a columna están diseñados para trabajar con sistemas de archivos distribuidos (como el sistema de archivos distribuidos Hadoop MapReduce) y se transforma para obtener datos dentro o fuera de los sistemas. Así que asegúrese de tener en cuenta estos factores antes de seleccionar una aplicación de las bases de datos orientadas a columna.

De este tipo de bases de datos existen dos bases de datos de las cuales son las más relevantes en cuanto a la estructura que las bases de datos orientadas a columnas presentan una de ellas es la principal fuente de inspiración para las bases de datos orientadas a columnas es Bigtable de Google, también se investiga a Cassandra, que se inspira en Bigtable. Esta base de datos se subsume de forma diferente por distintos autores, por ejemplo como una "base de datos orientada a columna " por (Yen, 2009) , como un “almacén de registros extensible "por (Cattell, 2013) o como un almacén de datos entidad - atributo - valor por (North, 2013). En este trabajo, Cassandra se discute junto con las bases de datos orientadas a columnas.

- ✓ **Google’s Bigtable**

Bigtable se describe como "un sistema de almacenamiento distribuido para gestionar datos estructurados que está diseñado para escalar a un tamaño muy grande: petabytes de datos a través de miles de servidores de productos básicos "(Strauch, 2012)

Según la experiencia de Google demuestra que “Bigtable ha logrado varios objetivos: amplia aplicabilidad, escalabilidad, alto rendimiento y alta disponibilidad.

Sus usuarios, como el rendimiento y alta disponibilidad proporcionada por la aplicación Bigtable, y que puedan escalar la capacidad de sus agrupaciones con sólo añadir más máquinas en el sistema como sus demandas de recursos cambian con el tiempo" (Chang, y otros, 2014).

Para Google como empresa el diseño e implementación de Bigtable ha demostrado ser ventajoso, ya que "ha conseguido una cantidad sustancial de la flexibilidad de diseñar un propio modelo de datos para Bigtable. Además, un control sobre la ejecución de Bigtable, y la otra infraestructura de Google en los que Bigtable depende, significa que podemos eliminar los cuellos de botella e ineficiencias que puedan surgir" (Chang, y otros, 2014).

Bigtable se describe como una base de datos por parte de Google ya que "compartía muchas de las estrategias de implementación de bases de datos", por ejemplo, paralelo y bases de datos de memoria principal. Sin embargo, se diferencia de las bases de datos relacionales, ya que "no es compatible con un modelo completo de datos relacional ", sino uno más simple que se puede controlar de forma dinámica por los clientes. Bigtable además, permite a los " clientes para razonar acerca de las propiedades de localidad de los datos " que se reflejan "en el almacenamiento subyacente " (Chang, y otros, 2014). A diferencia de los RDBMS, los datos pueden ser

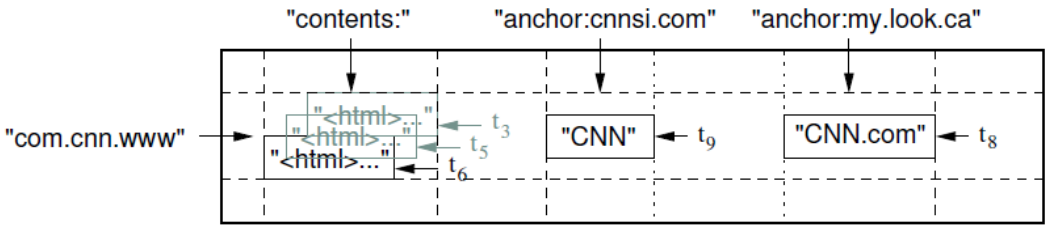
indexados por Bigtable en más de una dimensión, no sólo la fila, sino también de modo de columna. Una proposición distintivo adicional es que Bigtable permite que los datos se entregarán sin memoria o desde se puede especificar a través de la configuración que en el disco.

Modelo De Datos

En una afirmación de (Chang, y otros, 2014) nos dice que "creen que el modelopar clave-valor proporcionado por los árboles Bdistribuidasotablas hashdistribuidases demasiado limitante. Pares de valores clavesonun bloque de construcciónútil,pero no debeser el únicobloque deedificio de una solaproporcionaa los desarrolladores.Por tanto,el modelo de datossediseñan paraBigtabledebería sermás rico quepares clave-valorsimples, y[el apoyo] semiescasodatos estructurados".

Por otra parte, debe permanecer "bastante simple que se presta auna representaciónde archivo planomuy eficiente.

La estructura de datos que nos brinda y procesada por Bigtable de Google se describe como "un sistema distribuido, persistente mapa Ordenado escasa, multidimensional". Los valores se almacenan como matrices de bytes que no se deje interpretadas por el almacén de datos. Están dirigidas por la triple (clave de fila, la columna de clave, fecha y hora (Chang, y otros, 2014).



GoogleBigtable- Ejemplo deResultadosWeb(Chang, y otros, 2014)

En este ejemplo simplificado de una información Bigtable que esta almacenada en un rastreador web podría emitir. El mapa contiene un número no determinado de filas que representan dominios leídos por el rastreador, así como un número no determinado de columnas: la primera de estas columnas, contenidos que contiene los contenidos de la página, mientras que los otros (ancla: <domain-name>) acoplamiento del almacén textos de referencia dominios, de los cuales está representado por una columna dedicada. Cada valor también tiene una marca de tiempo asociada (t3,t5, t6 de los

contenidos de la página, t9 para el texto del enlace de CNN Sports Illustrated, t8 para el texto del enlace de MY- look). Por lo tanto, un valor no es abordado por la triple (de nombre de dominio, nombre-columna, fecha y hora) en este ejemplo (Chang, y otros, 2014).

Las claves de las filas en Bigtable son cadenas de tamaño de hasta 64 KB. Las filas se mantienen en orden lexicográfico y se dividen de forma dinámica por el almacén de datos en las llamadas pastillas ", el de la unidad de distribución y equilibrio de carga " en Bigtable. Las aplicaciones cliente pueden explotar estas propiedades eligiendo sabiamente la claves de fila: como el orden de filas keys influye directamente en la división de las filas en forma de comprimidos, la fila se extiende con una pequeña distancia lexicográfico probablemente están divididos en sólo unas pocas tabletas, por lo que las operaciones de lectura tendrán sólo un pequeño número de servidores de entrega de estas tabletas (Chang, y otros, 2014) .

En el ejemplo antes mencionado de la **figura** los nombres de dominio utilizados como claves de fila se almacenan jerárquicamente descendente (desde el punto de vista DNS) , de modo que los subdominios tienen una distancia lexicográfica menor que si los nombres de dominio se almacenan en sentido inverso (por ejemplo com.cnn.blogs , com.cnn.www en contraste con blogs.cnn.com , www.cnn.com) .

(Chang, y otros, 2014) También se refieren a la propia estructura de datos como Bigtable. El número de columnas por tabla no está limitado.

Las columnas se agrupan por su prefijo de clave en conjuntos llamados familias columna. Familias de columnas son un concepto importante en Bigtable ya que tienen propiedades y consecuencias (Chang, y otros, 2014) Específicos:

Ellos " son la unidad básica de control de acceso ", discerniendo privilegios para enumerar, leer, modificar y añadir las columnas son familias.

Se espera para almacenar el mismo o un tipo similar de datos.

Sus datos se comprimen entre sí por Bigtable.

Tienen que ser especificado antes de los datos se pueden almacenar en una columna contenida en una familia de la columna.

Su nombre tiene que ser imprimible. Por el contrario, los calificadores de las columnas " pueden ser cadenas arbitrarias".

(Chang, y otros, 2014)Sugieren "que el número de familias distintas columnas en una tabla sea pequeño (de cientos a lo sumo), y que las familias rara vez cambian durante la operación".

El ejemplo de la figura muestra dos familias de columnas: de contenido y de anclaje. La familia columna de contenido consiste en sólo una columna cuyo nombre no tiene que ser calificado más. Por el contrario, la familia de columnas de anclaje contiene dos columnas calificados por el nombre de dominio del sitio de referencia.

Las marcas de tiempo (**Timestamps**), representados como enteros de 64 bits, se utilizan en Bigtable discriminar diferente reversión de una celda. El valor de una marca de tiempo se asigna ya sea por el almacén de datos (es decir, la marca de tiempo real de salvar el valor de la celda) o elegido por las aplicaciones de cliente (y requiere ser único).

Bigtable ordena los valores de celda con el fin de su valor de marca de tiempo disminuyendo ", por lo que la versión más reciente se puede leer primero". Para aplicaciones cliente de borrado de revisiones antiguas o irrelevantes de valores de celda, se proporciona una recolección de basura automática y se pueden parametrizar por columna, ya sea especificando el número de revisiones de mantener o de su edad máxima (Chang, y otros, 2014).

Con respecto a las otras definiciones:

Ordenado: Simplemente las “filas” del mapa o vector van a estar ordenadas por un cierto criterio (lexicográficamente por la clave).

Distribuido: Si pensamos en un mapa con 100 “tuplas”, podríamos tener la mitad de un mapa en un servidor, y la otra mitad en otro.

Disperso: Se puede entender como “poco denso” también, puede darnos la sensación de tener pedazos de distintos mapas juntos.

Persistente: Simple, los datos se persisten a una unidad física (discos). O sea, se corta la luz y no perdemos nada.

También conocido como un mapa multidimensional ordenado.

Cassandra

Como último almacén de datos en este documento Apache Cassandra, que adopta las ideas y conceptos de los dos, Dynamo de Amazon, así como Bigtable de Google (entre otros, cf. (Lakshman & Malik, 2014), se discutirán. Fue desarrollado originalmente por Facebook y de código abierto en 2008. Lakshman describe Cassandra como un "sistema de almacenamiento distribuido para gestionar datos estructurados que está diseñado para escalar a un tamaño muy grande." Es " comparte muchas de las estrategias de diseño e implementación con bases de datos ", pero " no es compatible con un modelo de datos relacional completa, sino que ofrece a los clientes con un modelo de datos simple que soporta control dinámico sobre el diseño y el formato de datos "cf. (Cattell, 2013)(Avinash, 2014). Además de Facebook, otras empresas también han adoptado Cassandra como Twitter, Digg y Rackspace (cf.(Strauch, 2012)).

Origen y Requisitos de Concentración

El caso de uso principal para el diseño inicial y el desarrollo de Cassandra era la bandeja de entrada un problema Buscar legitimado en Facebook. El más grande del mundo de la red social Facebook permite a los usuarios intercambiar mensajes personales con sus contactos que aparecen en la bandeja de entrada del destinatario. El problema Bandeja de entrada Buscar se puede describir como la búsqueda de una forma eficiente de almacenar, indexar y buscar estos mensajes.(Strauch, 2012).

Los principales requisitos para el problema de búsqueda de la bandeja de entrada, así como los problemas de la misma naturaleza eran cf.(Avinash, 2014) :

- ✓ Elaboración de un índice de gran cantidad y el alto crecimiento de los datos dado 100 millones de usuarios en junio de 2008 , 250 millones en agosto de 2009 y más de 600 mil millones de usuarios a partir de enero de 2011; . cf.(Strauch, 2012).

- ✓ Alta escalabilidad e incremental

- ✓ Costo –efectividad
- ✓ "La fiabilidad a escala masiva ", ya que "[outages] en el servicio puede tener un impacto negativo significativo "
- ✓ La capacidad de " correr en la parte superior de una infraestructura de cientos de nodos [servidores básicos] (posiblemente repartidos en diferentes centros de datos) "
- ✓ Un " alto rendimiento de escritura sin sacrificar la eficiencia de leer"
- ✓ Ningún punto único de fallo
- ✓ Tratamiento de fracasos " como una norma y no una excepción "

Después de un año de uso productivo de Casandra en Lakshman Facebook resumió que" Cassandra ha logrado varias metas - escalabilidad, alto rendimiento, alta disponibilidad y aplicabilidad " (Avinash, 2014). En agosto de 2009, Lakshman y Malik afirman que " Cassandra ha mantenido la promesa hasta ahora" y el estado que fue también " despliega como el sistema de back-end de almacenamiento para múltiples servicios dentro de Facebook" (cf. (Lakshman & Malik, 2014)).

Modelo de Datos

Una instancia de Cassandra consiste típicamente en una sola tabla que representa un "mapa multidimensional distribuido indexado por la clave". Una tabla está estructurada por las siguientes dimensiones (cf.(Strauch, 2012)): Filas que se identifican por una clave de cadena de longitud arbitraria. Operaciones con filas son " atómica por réplica,no importa cuántas columnas se están leyendo o escribiendo". Las bases de datos orientadas a columna que pueden ocurrir en número arbitrario por fila. Al igual que en Bigtable, las bases orientadas a columnas tienen que ser definidos de antemano, por ejemplo, antes de que se puso en marcha un grupo de servidores que conforman una instancia de Cassandra. El número de bases orientadas a columnas por mesa no se limita, sin embargo, se espera que sólo unos pocos de ellos se especifican.

Una familia de columna compuesta de columnas y supercolumnas que se pueden añadir de forma dinámica (es decir, en tiempo de ejecución) para las columnas son familias y no están restringidas en número (cf. (Avinash, 2014)).

Las columnas tienen un nombre y guardar un número de valores por fila que se identifican por un sello de tiempo (como en Bigtable). Cada fila de una tabla puede tener un número diferente de columnas, por lo que una mesa no puede ser considerada como un rectángulo. Las aplicaciones cliente pueden especificar el orden de las columnas dentro de una familia de la columna y supercolumna que puede ser ya sea por nombre o por fecha y hora.

Supercolumnas tienen un nombre y un número arbitrario de columnas asociadas con ellos. Una vez más, el número de columnas por super columna puede diferir por fila. Por lo tanto, los valores en Cassandra son tratados por la triple (clave de fila, columna clave, fecha y hora) con columnkey como la columnafamilia: columna (por columnas simples contenidos en la familia de la columna) o de la familia de la columna: supercolumna: columna (por columnas subsumido bajo una supercolumna).

API

El API expuesta a clientes- aplicaciones de Cassandra se compone de sólo tres operaciones:

- ✓ _ Get (mesa , llave, columnName)
- ✓ _ Inserción (mesa , llave, rowMutation)
- ✓ _ Borrar (tabla, clave , columnName)

El argumento columnName de la operación de obtener y eliminar identifica una columna o un supercolumn dentro de una familia de la familia de columna o columna como un todo.(Lakshman & Malik, 2014)

Todas las peticiones emitidas por clientes- aplicaciones obtienen la ruta a un servidor arbitraria de un clúster de Cassandra que determina las réplicas de los datos que sirven para la clave solicitada. Para las operaciones de escritura de inserción y actualización de un quórum de nodos de réplica tiene " para reconocer la finalización de las escrituras".

Para las operaciones de lectura, los clientes pueden especificar qué garantía consistencia que ellos desean y en base a esta definición , ya sea el nodo más cercano al cliente atiende la solicitud o de un quórum de las respuestas de los diferentes nodos se esperaba antes de la solicitud rendimientos , por lo que mediante la definición de un cliente de quórum -aplicaciones pueden decidir qué "grado de consistencia eventual " que necesitan (cf. (Lakshman & Malik, 2014)) .

La API de Cassandra se expone a través de Ahorro. Además, las bibliotecas del lenguaje de programación de Java, Ruby, Pyhton, C # y otros están disponibles para la interacción con Cassandra (cf.(Strauch, 2012)).

Arquitectura del Sistema

Particiones: Como se requiere Cassandra ser incrementalmente escalable, máquinas pueden unirse y dejar un clúster (o accidente), por lo que los datos tiene que ser dividido y distribuido entre los nodos de un clúster de una manera que permite volver a particionar y redistribución. Los datos de una tabla de Cassandra, por lo tanto se divida y se distribuyen entre los nodos de una función hash consistente que también preserva el orden de fila -keys. (Lakshman & Malik, 2014)

Replicación: Para lograr una alta escalabilidad y la durabilidad de un clúster de Cassandra, los datos se replican en un número de nodos que pueden ser definidos como un factor de replicación por ejemplo Cassandra. La réplica está gestionada por un nodo coordinador de la clave en particular se está modificando, el nodo coordinador para cualquier clave es el primer nodo en el anillo de hash consistente que es visitado al caminar desde la posición de la llave en el anillo en sentido horario.(Lakshman & Malik, 2014)

Estrategias de replicación múltiple se proporcionan por Cassandra:

- ✓ **Rack Inconsciente** es una estrategia de replicación dentro de un centro de datos donde N - se eligen 115 nodos sucesivos el nodo coordinador en el anillo de hash consistente para replicar los datos a ellos.
- ✓ **Bastidor Aware**(dentro de un centro de datos) y Datacenter Aware son estrategias de replicación, donde por un sistema llamado ZooKeeper un líder es elegido para el grupo que está a cargo de mantener " el invariante que ningún nodo es responsable de más de N- 1 rangos en la ring". Los metadatos acerca de las responsabilidades de los nodos para ranges clave se almacena en caché localmente en cada nodo, así como en el sistema de Zookeeper. Los nodos que se han estrellado y empezar de nuevo, por tanto, pueden determinar qué clave rangos que son responsables.

La elección de los nodos de réplica, así como la asignación de nodos y claves rangos también afecta a la durabilidad: para hacer frente a fallos de nodo, particiones de red y fracasos incluso la totalidad del centro de datos, el " la lista de preferencias de una

clave se construye de tal manera que los nodos de almacenamiento se extienden a través de múltiples centros de datos " (Lakshman & Malik, 2014).

Membresía de Clúster Server y Detección de Fallo

La pertenencia a los servidores de un clúster Cassandra se gestiona a través de un protocolo Gossip-style llamado Scuttlebutt que se ve favorecida por su " utilización de la CPU muy eficiente y la utilización muy eficaz del canal de Gossip", según Lakshman y Malik. Además de la gestión de miembros, Scuttlebutt también se utiliza " para diseminar otro estado de control relacionado con el sistema " en un clúster Cassandra (cf.(Strauch, 2012)).

Los nodos dentro de un grupo Cassandra tratan localmente detectar si otro nodo es hacia arriba o hacia abajo para evitar los intentos de conexión a los nodos inalcanzables. El mecanismo empleado para este propósito de detección de fallos se basa en "una versión modificada de la Acumulación Si no detector". La idea detrás de los detectores de fallo de devengo es que ningún valor booleano es emitida por el detector de fallo sino un " nivel de sospecha para [. . .] Nodos monitoreados " lo que indica una probabilidad acerca de si suben o bajan. Lakshman y Malik Estado que, debido a sus experiencias "Detectores de fallo de devengacion son muy buenas, tanto en su precisión y su velocidad y también se adaptan bien a las condiciones de la red y las condiciones de carga del servidor " (cf. (Strauch, 2012)).

Administración del clúster: De acuerdo con las experiencias en Facebook Lakshman y Malik asumen que " [a] nodo corte rara vez significa un cambio permanente y, por tanto, no debe dar lugar a reequilibrio de la atribución de la partición o la reparación de las réplicas inalcanzables". Por lo tanto, los nodos tienen que ser añadido y retirado de un clúster explícitamente por un administrador (cf. (Avinash, 2014)).

Bootstrapping nodos: Cuando se añade un nodo a un clúster, se calcula una muestra al azar para su posición en el anillo de hash. Esta posición, así como el rango de claves es responsable de se almacena localmente en el nodo, así como en ZooKeeper. El nodo continuación, recupera las direcciones de unos pocos nodos del clúster desde un archivo de configuración o un servicio como ZooKeeper y anuncia su llegada a ellos que a su vez transmite esta información a todo el clúster. Por ese anuncio, la información de suscripción se extendió por todo el clúster para que cada nodo puede recibir solicitudes de cualquier clave y la ruta al servidor apropiado. Como el nodo de llegar divide una gama de claves de otro servidor fue anteriormente responsable de,

esta parte de la gama de claves tiene que ser transferido desde este último al nodo de unión (véase(Strauch, 2012)).

Persistencia: En contraste con Bigtable y sus derivados, Cassandra persiste sus datos a los ficheros locales en lugar de un sistema de archivos distribuido. Sin embargo, la representación de los datos en la memoria y en el disco, así como el procesamiento de lectura y escritura es tomada de Bigtable (cf.(Avinash, 2014)) :

- ✓ Las operaciones de escritura primero van a un persistente cometer registro y luego en una estructura de datos en memoria.

- ✓ Esta estructura de datos en memoria se persistió en el disco como un archivo inmutable si alcanza un cierto umbral de tamaño.

- ✓ Todas las escrituras a disco son secuenciales y se crea un índice " para la búsqueda eficiente basada en clave de fila " (como los bloques -índices de SSTables utilizados en Bigtable).

- ✓ Los archivos de datos en el disco quedan compactados de vez en cuando por un proceso de fondo.

- ✓ Las operaciones de lectura consideran el persistió en memoria de estructura de datos, así como los archivos de datos en el disco".

Con el fin de prevenir que las consultas en archivos que no contienen la clave, un filtro de floración, que resume las claves del archivo, también se almacena en cada archivo de datos y también se mantiene en la memoria". Esta " es la primera consulta para comprobar si la clave consultándose existe de hecho en el archivo dado."

Además, Cassandra mantiene índices para las familias de columnas y columnas "saltar a la porción derecha en el disco para la recuperación de la columna " y evite el escaneo de todas las columnas en el disco. Como las operaciones de escritura van a un sólo anexo cometer registro y como archivos de datos son inmutables, " [el] instancia de servidor de Cassandra es prácticamente sin bloqueo para operaciones de lectura / escritura."

2.10 SERVIDOR WEB

2.10.1 DEFINICIÓN

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa. (wikipedia, 2014)

2.10.2 SERVIDOR WEB APACHE

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés,

2.10.3 ARQUITECTURA

Es el servidor apache es un software que está constituido en módulos. La configuración de cada módulo se hace mediante la configuración de directivas que están contenidas dentro del módulo. Los módulos del apache se pueden clasificar en tres categorías.

- ✓ **Modulo Base**: Modulo con las funciones básicas del servidor Apache
- ✓ **Módulo Multiproceso:** son los responsables de la unión con los puestos de la maquina aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- ✓ **Modulo Adicionales**: Cualquier otro modulo que añada una funcionalidad al servidor

2.10.4 CARACTERÍSTICAS

- ✓ Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado
- ✓ Trabaja sobre múltiples plataformas lo que lo hace prácticamente universal
- ✓ Es posible adaptarse a diferentes entornos y necesidades con los diferentes módulos de apoyo
- ✓ Permite la configuración de servidores virtuales o sea permite alojar más de una web
- ✓ Apache es una tecnología gratuita de código abierto.
- ✓ Soporta los lenguajes de programación: CGI, PERL, PHP
- ✓ Permite la restricion de determinados sitios web
- ✓ Es un servidor de web conforme al protocolo http/1.1
- ✓ Incentiva la realimentación de los usuarios, obteniendo ideas informes de fallos y parches para la solución de los mismos
- ✓ Soporte para Base de datos.
- ✓ Permite una configuración personalizada.
- ✓ Soporte SSL y TSL para transacciones seguras
- ✓ Permite la autentificación de base de datos basada en SGBD

2.11 LENGUAJE DE PROGRAMACIÓN

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación.

También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos, a través de los siguientes pasos:

- ✓ El desarrollo lógico del programa para resolver un problema en particular.
- ✓ Escritura de la lógica del programa empleando un lenguaje de programación específico (codificación del programa).
- ✓ Ensamblaje o compilación del programa hasta convertirlo en lenguaje de máquina.
- ✓ Prueba y depuración del programa.
- ✓ Desarrollo de la documentación.

Permite especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.(Wikipedia, Lenguaje de Programación, 2013).

2.11.1 PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP.2 Este lenguaje forma

parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término PHP.

Teniendo en cuenta las características que PHP nos brinda como lenguaje de programación para el desarrollo de aplicaciones web, PHP es el lenguaje más económico y asequible, nos brinda gran desempeño con Windows, apache para brindarnos un mejor resultado. Además PHP permite un desarrollo en principios más sencillos y rápido, ya que su estructura es fácil lo que permite obtener resultados inmediatos.

2.12 ELECCIÓN DE LA BASE DE DATOS

Como pudimos revisar en los temas anteriores en los cuales se da a conocer sobre las características, funcionalidades, ventajas, desventajas de las bases de datos NoSQL, y en vista que las grandes redes sociales como Facebook, twitter y el propio Google las utilizan como medio fundamental de almacenamiento de información. Solo nos queda poder plantearnos el uso de una base de datos NoSQL, con lo cual nuestra primera decisión será el saber si realmente surge la necesidad de utilizar una base de datos NoSQL.

Para eso se considera los siguientes criterios:

En base a NoSQL.

- ✓ Parte de almacenamiento debe ser capaz de manejar muy alta carga. Se realizan muchas operaciones de escritura.
- ✓ Almacenamiento debe escalar horizontalmente (Clúster) Se prioriza la simplicidad.
- ✓ Lenguaje de consultas simple (normalmente sin JOINS).
- ✓ Cambios de esquemas frecuentes.

Una vez que se ha elegido el uso de una base de datos NoSQL, queda elegir la base de datos NoSQL.

Hay que tener en cuenta que las bases de datos NoSQL se clasifican en cuatro grupos o familias.

- ✓ Stores Key-Value: fundamentalmente se trata de usar una tabla hash donde hay una clave única y un puntero a un elemento de datos en particular.
- ✓ Document databases: similares a stores clave-valor, es el siguiente nivel. El modelo de datos es una colección de documentos que contienen colecciones de clave-valor
- ✓ Graph Databases: en lugar de tablas de filas y columnas y estructura rígida de SQL usan un modelo grafo flexible que puede escalar entre máquinas.
- ✓ Column Family Stores. Todavía hay claves, pero apuntan a varias columnas. Las columnas se organizan por familias de columnas.

Al elegir la base de datos hay que tener en cuenta las siguientes atribuciones de la tabla.

TABLA 10: Taxonomía- Tabla de ventajas y desventajas y alguno usos

Familia (clasificación)	Ventajas	Inconvenientes	Algunos productos /Usos típicos
Stores Key-value	Modelo más simple-Fácil de implementar y usar	Ineficiente cuando se realizan búsquedas, se quiere actualizar parte de un valor,	(Dynamo,Redis,Oracle NoSQL,Voldemort)-Cacheo de datos,Logging
Column Family Stores	Almacenamiento distribuido-Soportan un gran crecimiento Alta disponibilidad Escrituras masivas Soporte MapReduce Mecanismos de replicación	APIs de bajo nivel-Complejidad modelos No hay joins No permite ordenar resultados	(BigTable,Cassandra,HBase,Riak)-Sistemas de ficheros distribuidos MapReduce Estadísticas tiempo real
Document Databases	Modelado de datos natural y amigables-Flexibilidad Desarrollo rápido API de consulta potentes Trabajo con formato JSON Consultas más eficientes que key-value Relación entre documentos Índices Consultas espaciales	Rendimiento frente a Colum-Family Stores-Sintaxis de consulta no estándar Unsafe	(CouchDB,MongoDB)-Aplicaciones web: Wikis, Blogs Gestión documental
Graph Databases	Algoritmos de grafos (camino más corto,…)	No hay lenguaje de consultas estilo SQL-Tiene que recorrerse todo el grafo para obtener la respuesta. Complicado el clustering Problemas para cumplir el teorema CAP	(Neo4J,InfroGrid,AllegoGraph)-Redes sociales: recomendaciones

2.12.1 ELECCIÓN Y JUSTIFICACIÓN

Para el presente trabajo de tesis y el desarrollo de la aplicación web 2.0, se ha elegido la base de datos orientados en documentos ya que el aplicación web para la Liga Deportiva Parroquial San Pablo es de tipo web y almacenara registros de usuarios y documentos de los mismos es por eso que el producto que se utilizada para almacenamiento de la información será MongoDB el cual presta excelentes beneficio, ventajas credibilidad y de entre todos los productos de este tipo es el mejor, teniendo en cuente también la *tabla 10 de la taxonomía Ventajas y Desventajas de las bases de datos* NoSQL.

2.13 INTRODUCCIÓN A MONGODB + PHP

2.13.1 CONOCIENDO MONGODB

Como se ha ido mencionando en el presente documento de tesis MongoDB es una de las tantas bases de datos que NoSQL presenta como una alternativa de desarrollo y almacenamiento de información para las aplicaciones que hoy en día demanda gran cantidad de trasaccionabilidad, rapidez, Además, está licenciado como GNU AGPL 3.0, de modo que se trata de un software de licencia libre. Funciona en sistemas operativos Windows, Linux, OS X y Solaris.

En MongoDB, cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, las cuales se podría decir que son el equivalente a las tablas en una base de datos relacional (sólo que las colecciones pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo). Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos.

Los distintos documentos se almacenan en formato BSON, o Binary JSON, que es una versión modificada de JSON que permite búsquedas rápidas de datos. Para hacernos una idea, BSON guarda de forma explícita las longitudes de los campos, los índices de los arrays, y demás información útil para el escaneo de datos. Es por esto que, en algunos casos, el mismo documento en BSON ocupa un poco más de espacio de lo que ocuparía de estar almacenado directamente en formato JSON. Pero una de las ideas claves en los sistemas NoSQL es que el almacenamiento es barato, y es mejor aprovecharlo si así se introduce un considerable incremento en la velocidad de localización de información dentro de un documento.

Sin embargo, en la práctica, nunca veremos el formato en que verdaderamente se almacenan los datos, y trabajaremos siempre sobre un documento en JSON tanto al almacenar como al consultar información. Un ejemplo de un documento en MongoDB podría ser perfectamente éste:

```
{
  Nombre: "Pedro",
  Apellidos: "Martínez Campo",
  Edad: 22,
  Aficiones: ["fútbol","tenis","ciclismo"],
  Amigos: [   { Nombre:"María", Edad:22  },  { Nombre:"Luis", Edad:28  } ]
}
```

El documento anterior es un clásico documento JSON. Tiene strings, arrays, subdocumentos y números. En la misma colección podríamos guardar un documento como este:

```
{
  Nombre: "Luis",
  Estudios: "Administración y Dirección de Empresas",
  Amigos:12
}
```

Este documento no sigue el mismo esquema que el primero. Tiene menos campos, algún campo nuevo que no existe en el documento anterior e incluso un campo de distinto tipo.

Esto que es algo impensable en una base de datos relacional, es algo totalmente válido en MongoDB.

2.13.2 PRIMEROS PASOS CON MONGODB

En esta parte se dara una breve introducción a las operaciones de base de datos básicos utilizados por mongo shell. Mongo es una distribución de MongoDB y ofrece un entorno de ejecución de comandos en javascript y todas las funciones estándar.

Conectarse a una Base de datos

Nos conectamos al servidor de base de datos que se ejecuta como mongod y comienza a utilizar el mongo Shell.

Desde una ventana de comandos ejecutamos mongo.exe

Por defecto mongo busca un servidor de base de datos que permite conexión por el puerto 27017 en el localhost.

Seleccionamos una base de datos

Para ver las bases de datos en mongo Shell utilizacmos el siguiente comando

Shows dbs

Para seleccionar la base de datos

Use Mydb

Para confirmar la selección de la base de datos

db

Crear una colección e Insertar un documento

En esta sección, insertar documentos en una nueva colección llamada TestData dentro de la nueva base de datos denominada mydb .

MongoDB creará una colección implícitamente en su primer use. No es necesario para crear una colección antes de insertar datos. Además, debido a MongoDB utiliza esquemas dinámicos, también es necesario no especifica la estructura de sus documentos antes de insertarlos en la colección.

Desde el mongo concha, confirme que está en el mydb base de datos emitiendo el siguiente:

db

Si mongo no vuelve mydb para la operación anterior, establecer el contexto de la mydb base de datos, con la siguiente operación:

use mydb

Cree dos documentos con nombre j y k mediante la siguiente secuencia de operaciones de JavaScript:

```
j = { nombre : "mongo" }

k = { x : 3 }
```

Insert los j y k documentos en el TestData colección con la siguiente secuencia de operaciones:

```
db . TestData . insert ( j )

db . TestData . insert ( k )
```

Al insertar el primer documento, el mongod creará tanto el mydb base de datos y la TestData colección.

Confirme que el TestData existe colección. Emita la siguiente operación:

```
show  colecciones
```

El mongo shell devolverá la lista de las colecciones en el actual (es decir mydb) base de datos. En este punto, la única colección con los datos de usuario es TestData .

Compruebe que existen los documentos en el TestData colección mediante la emisión de una consulta en la colección con el find () método:

```
db . TestData . find()
```

Esta operación devuelve los siguientes resultados. Los OBJECTID valores serán únicas:

```
{ "_id" : ObjectId ( "4c2209f9f3924d31102bd84a" ), "name" : "mongo" }

{ "_id" : ObjectId ( "4c2209fef3924d31102bd84b" ), "x" : 3 }
```

Todos los documentos deben tener un MongoDB _id campo con un valor único. Estas operaciones no especifican explícitamente un valor para el _id campo, por lo mongo crea un único ObjectId valor para el campo antes de insertarlo en la colección.

Mongodb Maping Chart

Crear y modificar

La siguiente tabla presenta las diferentes sentencias de acciones a nivel de Colecciones y las declaraciones correspondientes MongoDB.

Declaraciones de esquema MongoDB

Implícitamente creado en la primera pieza de inserción () “insert()” operación. La clave principal _id se agrega automáticamente si _id no se especifica el campo.

```
db . usuarios . inserte ( {

user_id : "abc123" ,

edad : 55 ,

estado : "A"

} )
```

Sin embargo, también puede crear explícitamente una colección:

```
db . createCollection ( "usuarios" )
```

Colecciones no describen ni hacer cumplir la estructura de sus documentos; es decir, no hay alteración estructural en el nivel de la colección.

Sin embargo, a nivel de documento, update () las operaciones se pueden agregar campos a los documentos existentes utilizando el $ set operador.

```
db . usuarios . update (

  { },

  { $ set : { join_date : new date() } },

{ multi : true}

)
```

Colecciones no describen ni hacer cumplir la estructura de sus documentos; es decir, no hay alteración estructural en el nivel de la colección.

Sin embargo, a nivel de documento, update () las operaciones se pueden eliminar campos de documentos utilizando el eliminadas $ operador.

```
db . usuarios . update (
   {  },
   {  $ unset :  {  join_date :  ""  }  },
{  multi :  true}
)
```

Creación de índices

```
db . usuarios . ensureIndex (  {  user_id :  1  }  )
db . usuarios . ensureIndex (  {  user_id :  1 ,  edad :  - 1  }  )
```

Eliminar Colecciones

```
db . usuarios . drop ()
```

Ingresar Datos

Se presenta el esquema de JSON para el ingrso de datos a las colecciones

```
db . usuarios . insert (
   {  user_id :  "bcd001" ,  edad :  45 ,  estado :  "A"  }
)
```

Consultar, Seleccionar o Buscar

Se presenta las diversas formas que se puede realizar una consulta en las colecciones y documentos de la base de datos Mongodb

```
db.users.find()
db.users.find(
   { },
```

```
    { user_id: 1, status: 1 }
)
db.users.find(
    { },
    { user_id: 1, status: 1, _id: 0 }
)
db.users.find(
    { status: "A" }
)
db.users.find(
    { status: "A" },
    { user_id: 1, status: 1, _id: 0 }
)
db.users.find(
    { status: { $ne: "A" } }
)
db.users.find(
    { status: "A",
      age: 50 }
)
db.users.find(
    { $or: [ { status: "A" } ,
```

```
            { age: 50 } ] }
)
db.users.find(
    { age: { $gt: 25 } }
)
db.users.find(
   { age: { $lt: 25 } }
)
db.users.find(
   { age: { $gt: 25, $lte: 50 } }
)
db.users.find( { user_id: /bc/ } )
db.users.find( { user_id: /^bc/ } )
db.users.find( { status: "A" } ).sort( { user_id: 1 } )
db.users.find( { status: "A" } ).sort( { user_id: -1 } )
db.users.count()
db.users.find().count()
db.users.count( { user_id: { $exists: true } } )
db.users.find( { user_id: { $exists: true } } ).count()
db.users.count( { age: { $gt: 30 } } )
db.users.find( { age: { $gt: 30 } } ).count()
db.users.distinct( "status" )
```

```
db.users.findOne()

db.users.find().limit(1)

db.users.find().limit(5).skip(10)

db.users.find( { status: "A" } ).explain()
```

Actualizat registros

Se presenta el esquema para realizar las diferentes formas de actualización de datos en las colecciones y documentos.

MongoDB update() Statements

```
db.users.update(
  { age: { $gt: 25 } },
  { $set: { status: "C" } },
  { multi: true }
)

db.users.update(
  { status: "A" } ,
  { $inc: { age: 3 } },
  { multi: true }
)
```

Eliminar registros

Se presenta el esquema para realizar la eliminación de los registros

```
db . usuarios . remove (  {  status :  "D"  }  )

db . usuarios . remove ({})
```

2.13.3 MODELADO DE DATOS EN MONGODB

Los datos en MongoDB tienen un esquema flexible. A diferencia de las bases de datos SQL, en el que debe determinar y declarar el esquema de una tabla antes de introducir los datos, las colecciones de MongoDB no hacen cumplir la estructura del documento. Esta flexibilidad facilita el mapeo de documentos a una entidad o un objeto. Cada documento puede coincidir con los campos de datos de la entidad representada, incluso si los datos tienen una variación sustancial. En la práctica, sin embargo, los documentos de una colección comparten una estructura similar.

El reto clave en el modelado de datos es equilibrar las necesidades de la aplicación, las características de rendimiento del motor de base de datos, y los patrones de recuperación de datos. En el diseño de modelos de datos, siempre en cuenta el uso de la aplicación de los datos (es decir, consultas, actualizaciones, y el procesamiento de los datos), así como la estructura inherente de los propios datos.

Estructura del documento

La decisión clave en el diseño de modelos de datos para aplicaciones de MongoDB gira en torno a la estructura de los documentos y cómo la aplicación representa las relaciones entre los datos. Hay dos herramientas que permiten a las aplicaciones representan estas relaciones: las referencias y documentos incrustados

Referencias

Referencias almacenan las relaciones entre los datos mediante la inclusión de enlaces o referencias de un documento a otro. Las solicitudes se pueden resolver estas referencias para acceder a los datos relacionados. En términos generales, estos son los modelos de datos normalizados.

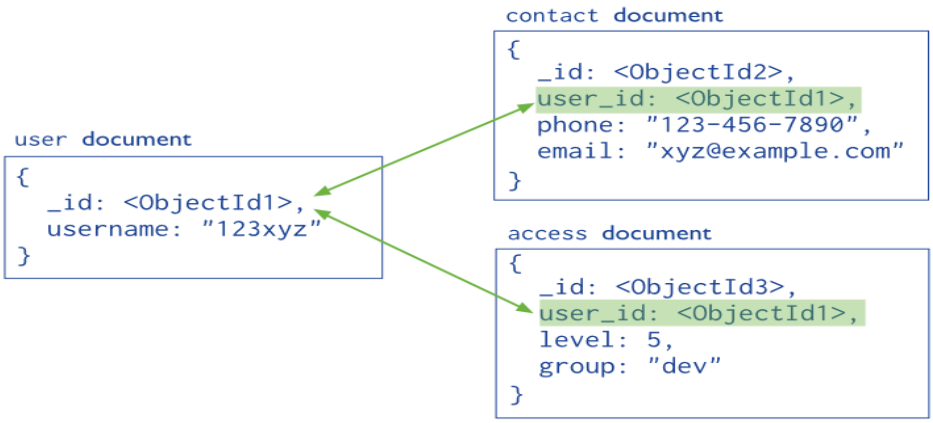


FIGURA 14: Esquema de la base de datos normalizada en mondoDB

Documentos Incrustados O Valores Asociados

Relaciones documentos de captura incrustados entre los datos de almacenamiento de datos relacionados en una sola estructura del documento. MongoDB documents hacen posible embeber estructuras de documentos como sub- documentos en un campo o una matriz dentro de un documento. Estos modelos de datos no normalizados permiten que las aplicaciones para recuperar y manipular datos relacionados en una sola operación de base de datos.

FIGURA 15: Modelos de datos no normalizados o Embedded Data de mongoDB

2.13.4 ESTABLECIMIENTO DE UNA CONEXIÓN

Para conectarse a mongo y seleccionar una base de datos se utiliza, diferente tipos de conexiones por ejemplo:

$connection = new Mongo (); / / se conecta a localhost: 27017

$connection = new Mongo (“sindikos.com”); / / conectarse a un host remoto (puerto por defecto)

$connection = new Mongo (“sindikos.com: 65432″); / / conectarse a un host remoto en un puerto

$db = $connection-> selectDB (“db”); // Trabajar con ‘db’

$db = $connection-> selectDB (“otradb”); //Cambio de BBDD a ‘otradb’

Si la base de datos solicitada no existe, se creará al seleccionarla, debemos tener cuidado pues con la ortografía o podemos crear nuevas bases de datos.

2.13.5 OBTENCIÓN DE UNA COLECCIÓN

Para obtener una colección específica, utilice el selectCollection () de comandos. Al igual que el comando selectDB (), esto crea la colección si no existe ya.

$coleccion = $db-> selectCollection (“articulos”);

2.13.6 INSERCIÓN DE UN DOCUMENTO

Las matrices son el objeto básico que se puede guardar en una colección en la base de datos. Por ejemplo:

$doc = array (“nombre” => “articulo”,

“type” => “Publicada”,

“autor” => “J.P.Aulet”,

“version” => 1,

“texto” => “El texto del artículo”,

“info” => (object) array (“x” => 203,”Y” => 102),

“publicada” => array (“08/11/2011″, “23:00″, “PM”)

Esto crea un documento con los datos especificados, debemos tenga en cuenta que puede tener matrices anidadas y objetos como parte de los datos. Para añadir este nuevo documento, solo debemos hacer lo siguiente:

```
$colection->insert($doc);
```

O de forma más compacta con:

```
$bbdd = new Mongo();

$colection = $bbdd->selectDB(“db”)->selectCollection(“articulos”);

$colection->insert($doc);
```

Encontrar el primer documento en una colección con FindOne ()

Para ver el documento insertado en la anterior sentencia, podemos hacer un findOne(), que devuelve un solo valor:

```
$res = $colección->findOne ();

var_dump ($res);
```

Al hacer esto visualizará el anterior documento, y podrá ver que el campo _id ha sido añadido automáticamente por MongoDB.

2.13.7 USANDO EL CURSOR

Si queremos consultar todos los documentos, MongoDB ofrece un cursor que permite iterar sobre el conjunto de documentos, se invoca al llamar la función find() . Así que para consultar todos los documentos (primero debemos añadir más) e imprimirlos podemos hacer lo siguiente:

```
$cursor = $ colección->find ();

foreach ($cursor como $id => $valor) {

echo “$ id”;

var_dump ($ valor);

}
```

2.13.8 OBTENCIÓN DE UN DOCUMENTO ÚNICO CON UNA CONSULTA

Podemos crear una consulta para pasar al método find() para obtener un subconjunto de los documentos de nuestra colección. Por ejemplo, si queremos encontrar el documento por el cual el valor de la “i” de campo tiene el valor ’71′, haríamos lo siguiente:

```
$query = array (“i” => 71);

$ cursor = $colección->find($query);

while ($cursor-> hasNext ()) {

var_dump ($ cursor-> getNext());

}
```

Fijarnos que podemos usar la función hasNext() o getNext() para iterar sobre el conjunto de documentos devueltos por find(). Hasta aquí este pequeño tutorial de MongoDB. En los últimos tiempos la tecnología NoSQL está en auge, sobretodo por el cambio a esta tecnología de varios de los grandes gigantes de Internet, como son Facebook, Amazon, Google, Foursquares o Yahoo, que han despertado la curiosidad de la comunidad de desarrolladores. Algunos pensarán que no les interesa, pero siempre es útil conocer todas las tecnologías y soluciones disponibles, otros sólo verán que ventajas. Para saber si realmente necesitamos MongoDB.

CAPÍTULO III

3 ARQUITECTURA DEL SISTEMA

3.1 FUNCIONAMIENTO DEL SISTEMA

La aplicación web se desarrollara para la “Liga Deportiva Parroquial De San Pablo Del Lago“, con el fin de demostrar el funcionamiento de una de las bases de datos NoSQL la cual en su taxonomía la base de datos MongoDB de la familia de bases de datos orientadas a columnas es la más eficaz.

El sistema en si permitirá a los usuarios interactuar con la información que este ira almacenando, este sistema permitirá a los usuarios estar al tanto en las novedades, progreso, posiciones en que se encuentre la ejecución de los campeonatos organizados por la Liga Deportiva Parroquial San Pablo Del Lago, como también permitirá el ingreso, actualización de información de los equipos y sus miembros.

Con el fin de demostrar no solo las características y potencial, las debilidades y oportunidades que nos brinda los sistemas al desarrollar sus diferentes módulos:

El esquema de la base de datos que se utilizar es totalmente plano.

3.1.1 DEFINICIÓN DE LOS MÓDULOS

La aplicación web Consta de los siguientes módulos por el momento

- ✓ Módulo de pases
- ✓ Control de Tarjeta
- ✓ Gestión Jugadores
- ✓ Impresión de reportes

Módulo Gestión de Jugadores: Esta arquitectura permitirá administras usuarios administradores y clientes del aplicación.

Módulo de pases: Esta fase de la aplicación estará encargada de generar documentos de pases entre los equipos

Módulo de Control De Tarjetas: Controla las sanciones de los jugadores de cada equipo.

Módulo de Impresión de reportes: Imprime reportes de los documentos.

3.1.2 FUNCIONAMIENTO DE LOS MÓDULOS

Cada uno de los módulos del aplicación web de la liga deportiva parroquia de san pablo del lago realizara la funciones básicas de ingreso, eliminación, búsqueda, y actualización.

Módulo de Gestión de Jugadores

Este módulo realizar el ingreso, modificación, eliminación y consulta de los jugadores de la Liga Deportiva Parroquial San Pablo del lago

Módulo de pases

Este módulo permitirá visualizar un documento en la web con el contenido un informe de realización de pase de los jugadores entres los equipos involucrados

Módulo de Control De Tarjetas

Este módulo permitirá el ingreso y visualizar las sanciones con tarjetas amarillas o rojas que los jugadores acumulen en cada fecha programada.

Módulo de Impresión de reportes

Este módulo permitirá la impresión de los documentos respectivos para cada jugador o usuario ya sea este un carnet, o el documento de pase solicitado como también el reporte de acumulado de tarjetas, así como también un listado de los jugadores inscritos en cada equipo.

3.2 DISEÑO Y DESARROLLO DEL APLICATIVO

3.2.1 FUNCIONAMIENTO GENERAL DE LA APLICACIÓN

Nuestra aplicación consta principal de unas bases de datos, donde se almacena toda la información referente a nuestra aplicación y sus interfaces o páginas web.

En primer lugar al iniciar la aplicación tendremos una presentación de la información general de la Liga San Pablo del Lago, la cual consta de tres accesos.

La primera tendrá acceso a un portal informativo de la Liga Deportiva Parroquial de San pablo del Lago, en esta constara de una animación en JavaScript la cual mejora la presentación dinámica de la página web.

El segundo link de acceso consta de una página la cual es el inicio de sesión, para poder iniciar una sesión de vemos estar registrados en la base de datos de nuestra aplicación, Después en la parte superior derecha de la página, tendremos que poner el nombre de usuario en la casilla “Nombre de Usuario” y luego escribir la contraseña en la casilla correspondiente “Password”.

La contraseña o clave es una forma de autenticación que utiliza información secreta para controlar el acceso hacia algún recurso, en este caso hacia la página que se nos redirige, que dependerá del rol que tengamos, ya seamos administradores, jugadores o directivos.

La contraseña se nos pedirá siempre al inicio de sesión, negándonos el acceso siempre y cuando no la conozcamos o hayamos cometido algún error al introducirla.

Una vez creada la sesión, se le asigna al usuario un nivel de autorización (administrador, jugador, Directivos) del que depende que se nos redirija a una página u otra. Éstas son:

La página del directivo “liga-admin”, donde las opciones que se ofrecen a los administradores son las de crear, modificar y eliminar, con las que podrán gestionar los equipos para depurar las listas de los mismos.

Podrán consultar la lista jugadores en cada uno de los clubs almacenado en la base de datos o aquellos nuevos que se vayan creando así como borrarlos.

La página del Jugadores “liga-jugadores”, permite a los jugadores consultar sus datos almacenados en la base de datos que así ellos puedan modificar o ingresar sus datos personales en la base de datos.

Y por último la página de la administración “liga administrator”, donde el administrador tendrá todas las opciones correspondientes para manejar y controlar el funcionamiento de la aplicación. Opciones de listar, crear y eliminar referentes a cada uno delos bloques de equipos, jugadores, campeonatos y directivos con las que podrán listar, añadir y borrar jugadores de la base de datos, listar, añadir y borrar equipos, al igual que listar, añadir y borrar equipos y campeonatos.

3.2.2 BASE DE DATOS

Nuestra aplicación está sustentada por una base de datos desarrollada en lenguaje C++ MongoDB. La misma sirve para controlar y gestionar los diferentes usuarios del sistema (directivos, jugadores y administradores).

A continuación mostramos un diagrama de clase de la base de datos.

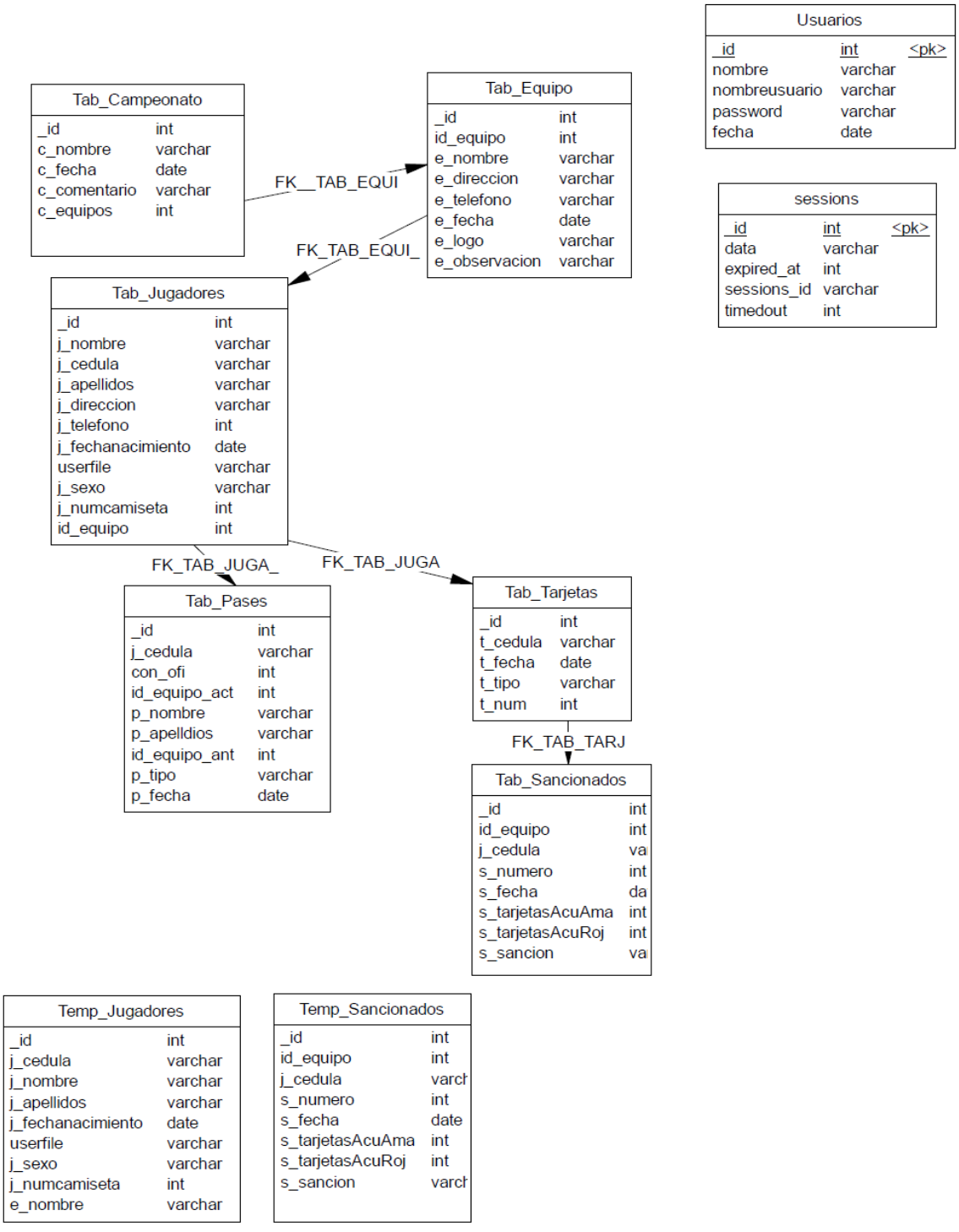


FIGURA 16: Diagrama de la base de datos

Las funcionalidades de cada diferente tipo de usuario pasaremos a comentarlas más adelante cuando estudiemos los casos de uso del sistema.

Destacar que la primera vez que la aplicación arrancó, lo hace con un administrador por defecto, ya que son el administrador los que pueden dar de alta a los directivos y jugadores

3.2.3 ARQUITECTURA DEL APLICACIÓN

El sistema utiliza en sistema del Modelo vista controlador.

- ✓ **Modelo**

El modelo es aquello que soporta los datos (o capa de negocio). La representación específica de la información con la cual el sistema opera.

En este caso el modelo está formado por la base de datos MongoDB

- ✓ **Vista**

La vista presenta un modelo en el cual se pueda interactuar.

Dentro de nuestra aplicación nuestra vista está conformada por los archivos de CSS, JavaScript, PHP y HTML.

CSS.- Los archivos login.css, style.css, index.css

Son las hojas de estilo de las pantallas de index y login respectivamente, esto es, los elementos propios de la aplicación. Además de estas, hay otras hojas de estilo por defecto para los elementos de jQuery y DataTable.

JavaScript.- Archivos indexAdmin.js y login.js

Son archivos de funciones JavaScript de las que se hace uso. Son homólogas a las de las hojas de estilo, salvo que hemos diferenciado en función del tipo de usuario.

En este caso, también hay archivos JavaScript por defecto que marcan el comportamiento de los elementos de jQuery y DataTable.

PHP.- Los archivos index.php, indexAdmin.php, indexJugadores.php y login.php

Son archivos PHP que utiliza nuestra aplicación y que se “visten” con las hojas de estilo y las funciones JavaScript comentadas antes.

De nuevo, son análogos a los mencionados previamente, diferenciamos en virtud del tipo de usuario que usa la aplicación.

- ✓ **Controlador**

El controlador responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista. En el caso corresponde al archivo liga-campeonato

Este es el archivo principal, donde comienza la ejecución de la aplicación. Levanta el servidor de xampp y php, captura los GET y POST que llegan del cliente redirigiendo esas acciones a liga-campeonato. Configura el puerto y el host al que se conectará el cliente, pudiendo estar esta configuración implícita en el archivo o indicándolo como opciones del comando al ser ejecutado.

3.2.4 CASO DE USO DEL SISTEMA

Los diferentes casos de uso de nuestra aplicación los mostraremos y comentaremos con la ayuda de unos diagramas de casos de uso en función del usuario que se encuentre en el sistema y con las pertinentes capturas de la aplicación.

Un diagrama de casos de uso es una especie de diagrama de comportamiento. Define una notación gráfica para representar casos de uso llamada modelo de casos de uso. Esta notación gráfica define la naturaleza de un caso de uso en concreto. Mencionar que, todas las acciones que se van a explicar deben estar en ejecución para que se puedan llevar a cabo.

✓ **Casos De Uso: Administrador Del Sistema**

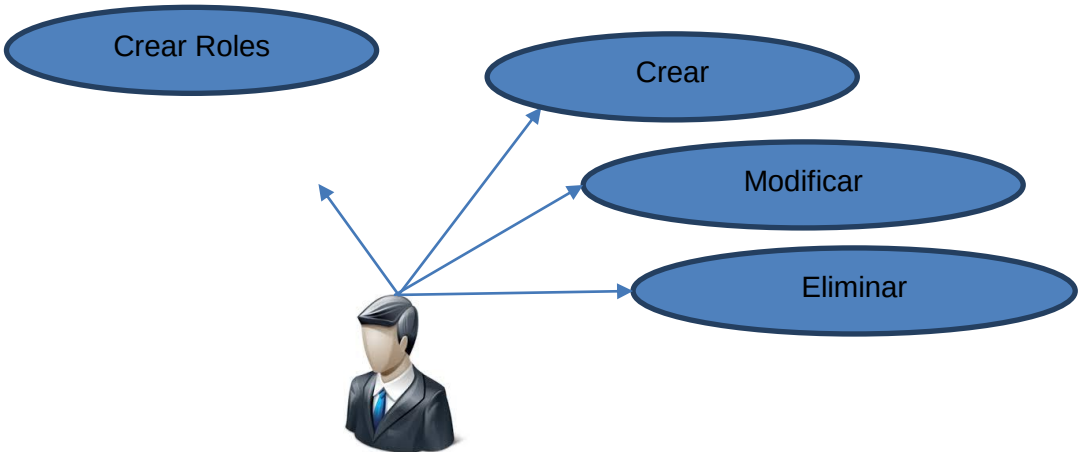


FIGURA 17: Caso de Uso Administrador de Sistemas

➢ **Descripción:**

Este Caso de Uso Describe los procesos que realiza el administrador del sistema

➢ **Flujo Básico del Evento**

El administrador crea usuarios y Roles asignando los permisos necesarios en el Módulo de Seguridad.

➢ **Precondiciones**

- El administrador del sistema se encuentre logeado y tenga en cuenta los permisos de usuario requeridos.
- El administrador del sistema debe conocer cuáles son las responsabilidades que tiene el usuario con el sistema.

➢ **Poscondiciones**

- Dar los permisos de usuarios solicitados.
- Otorgar permisos en el menú del sistema de acuerdo al rol del usuario.
- Crear, Modificar y Eliminar Campeonatos, Equipos, jugadores.

✓ **Casos De Uso: Registro Datos Liga**

FIGURA 18: Caso de Uso Registro Datos Liga

➢ **Descripción:**

Este Caso de Uso Describe los procesos que el presidente y el secretario realizan para realizar el registro de jugadores en cada equipo para la debida participación en el campeonato.

➢ **Flujo Básico del Evento**

- El Presidente realiza la creación del campeonato y los equipos que participarán así también la creación de jugadores.
- El Secretario realizara el creación y registro de los jugadores, asi como también la impresión de listas y carnets de los jugadores y equipos respectivos.

➢ **Precondiciones**

- EL Presidente y secretario deben tener los permisos necesarios para el debido registro de datos.

➢ **Poscondiciones**

- Dar los permisos de usuarios solicitados.

✓ **Caso de Uso: Revisión de Reportes**

FIGURA 19: Caso de Uso Revisión Reportes

➢ **Descripción:**

Este Caso de Uso Describe los procesos que el jugador realiza en el sistema web.

➢ **Flujo Básico del Evento**

- El jugador podrá revisar sus sanciones ingresar datos e imprimir los pases.

➢ **Precondiciones**

- EL jugador deben tener los permisos necesarios para el debido registro de datos.

➢ **Poscondiciones**

- Dar los permisos de usuarios solicitados.

CAPÍTULO IV

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Al final del documento de tesis sobre el "**Estudio De Nuevas Tecnologías De Gestión De Bases De Datos NOSQL, Para El Desarrollo De Aplicaciones Web 2.0"** puedo decir que se ha cumplido los objetivos planteados al inicio del proyecto de la mejor manera.

En la actualidad los desarrolladores de aplicaciones web 2.0 se basan en la confiabilidad, costo, desempeño, rapidez y flexibilidad que las bases de datos NoSQL, que brindan a las grandes empresas como son Facebook, Amazon, twitter, google, por sus beneficios, funciones y alta velocidad.

Las bases de datos NoSQLresulta ser una alternativa potente para el desarrollo de las aplicaciones web que requieren el manejo de gran cantidad de datos, así como también

aplicaciones que requieren menor coste en el tiempo de respuesta, estas se adaptan según las necesidades de los programadores ya que son de fácil aprendizaje.

Teniendo en cuenta las fortalezas y debilidades de las bases de datos tanto relacionales como NoSQL

En este documento se da a conocer 4 de las taxonomías principales de bases de datos NoSQL, que al momento están siendo desarrolladas y utilizadas, estas son las bases de datos Clave/valor, las bases de datos Orientadas a documentos, Bases de datos Orientadas a Grafos, bases de datos Orientadas a Columnas, las cuales presentan diferentes distribuciones como BerkeleyDB y Amazon dynamo bases de datos Clave/valor.

Las bases de datos MongoDB y CouchDB que son bases de datos Documentales, Neo4j y FlockDB que son bases de tipo grafo, las bases de datos de tipo columna como google,s Bigtable, Cassandra, todas estas bases de datos mantienen relaciones en común las cuales les permite ser únicas para cada aplicación.

Por ejemplo BerkeleyDB que tiene un método de table Hash que permite almacenar la información en esquemas organizados por Valores, o MongoDB que almacena la información en esquemas pues su almacenamiento utiliza colecciones , sin una estructura a seguir el cual permite la manipulación de información de una manera rápida. Neo4j una base de datos tipo Grafo la cual permite guardar información en nodos y aristas, estas bases de datos son muy utilizadas por las redes sociales como Facebook, twitter, entre otras.

Las bases de datos tipo columnas que son bases de datos que permite tener una escalabilidad horizontal lo cual permite agregar clúster en cualquier momento.

Teniendo en cuenta la clasificación delas bases de datos que existen hasta el momento hemos tomado la decisión de utilizar una de ellas la cual es relevante para el desarrollo de nuestra aplicación, por sus características e identificación con la aplicación desarrollada para la liga deportiva parroquial de san pablo del lago se ha tomado en cuenta MongoDB una base de datos de tipo documento.

La cual permitirá a la Liga guardar información de sus jugadores y documentos esenciales para la participación en los campeonatos inaugurados en los siguientes años.

Indudablemente el estudio de las nuevas herramientas en este caso las bases de datos NoSQL, significan la exploración y análisis en beneficios una gran alternativa para el desarrollo de aplicaciones web 2.0 en las siguientes generaciones.

4.2 RECOMENDACIONES

Las bases de datos NoSQL fueron diseñadas para realizar tareas específicas, en cuanto al almacenamiento de información, como MongoDB que es un motor de búsqueda en las nubes de gran almacenamiento, Casandra que es un motor de búsqueda de Facebook, que son gestores de bases de datos para un alto coste de información.

En el diseño de un modelo de datos, tenga en cuenta cómo las aplicaciones utilizarán la base de datos. Por ejemplo, si la aplicación sólo utiliza documentos recientemente insertados, considere el uso capsulado de colecciones. O si sus necesidades de aplicación se leen principalmente operaciones a una colección, agregar índices de apoyo a consultas comunes puede mejorar el rendimiento.

NoSQL no es apropiado para el desarrollo de todaslas aplicaciones y proyectos. Los casos típicos de uso son con aplicaciones de Internet con un gran tráfico y muchos datos (hablamos de gigabytes o petabytes). Pocas aplicaciones tienen este tipo de requisitos, por otra parte, las aplicaciones más pequeñas se benefician de los menores requisitos de las base de datos NoSQL (menor infraestructura, fácil replicación, creación de bases de datos automáticas, etc.) pero debe evaluar si se es capaz de sostener la elección NoSQL.

CouchDB Y MongoDb son bases de datos NoSQL que presentan una flexibilidad al momento de modelar los datos, permiten un desarrollo rápido de las aplicaciones y son súper amigables en el código para la creación de bases de datos y en su lenguaje de consulta que es JSON.

Que la Universidad Técnica del Norte fomente acciones que ayuden en la identificación y desarrollo de capacidades – personas, procesos y tecnologías – que puedan apoyar el desarrollo de nuevos productos y servicios utilizando soluciones de bases de datos NoSQL

Se recomienda el estudio de las diferentes clasificaciones de bases de datos NoSQL como Bases de datos Clave/Valor, Bases de datos orientadas a grafos y orientadas a columnas como temas de análisis e investigación para el desarrollo de aplicaciones Web 2.0.

4.3 GLOSARIO DE TERMINOS

ACID.- En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad

BigTable.- BigTable es un sistema de gestión de base de datos creado por Google con las características de ser: distribuido, de alta eficiencia y propietario. Está construido sobre GFS (Google File System), Chubby Lock Service, y algunos otros servicios y programas de Google, y funciona sobre 'commodity hardware' (sencillos y baratos PCs con procesadores Intel).

DBM.- Estructura procesos de cambio planificado para lograr que las personas manejen de manera efectiva la transición y facilita a las empresas y a los individuos la navegación en el cambio organizacional fortaleciendo la visión y los resultados del negocio.

DBMS.- Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

DHT.- Las tablas de hash distribuidas (en inglés, Distributed Hash Tables, DHT) son una clase de sistemas distribuidos descentralizados que proveen un servicio de búsqueda similar al de las tablas de hash, donde pares (clave, valor) son almacenados en el DHT, y cualquier nodo participante puede recuperar de forma eficiente el valor asociado con una clave dada.

DNS.- Domain Name System o DNS (en español: sistema de nombres de dominio) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada.

CSS.- Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL.

CTO.- El director de tecnología (del inglés chief technical officer o chief technology officer, abreviado como CTO) es una posición ejecutiva dentro de una organización en el que la persona que ostenta el título se concentra en asuntos tecnológicos y científicos.

GQL.- Es un lenguaje similar a SQL para recuperar las entidades o las llaves del almacén de datos escalable App Engine. Mientras que las características de GQL son diferentes de las de un lenguaje de consulta de una base de datos relacional tradicional, la sintaxis de GQL es similar a la de SQL.

HTTP.- Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web

HTML.- Siglas de HyperText Markup Language («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web.

Hadoop.- Apache Hadoop es un framework de software que soporta aplicaciones distribuidas bajo una licencia libre.1 Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos. Hadoop se inspiró en los documentos Google para MapReduce y Google File System (GFS).

IDE.- Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación.

JavaScript.- JavaScript (abreviado comunmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.

JOIN.- La sentencia join en SQL permite combinar registros de dos o más tablas en una base de datos relacional. En el Lenguaje de Consultas Estructurado (SQL) hay tres tipos de JOIN: interno, externo y cruzado.

LDAP.- Son las siglas de Lightweight Directory Access Protocol (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red

NoSQL.- En informática, NoSQL (a veces llamado "no sólo SQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado que no usan SQL como el principal lenguaje de consultas.

p2p.- Una red peer-to-peer (P2P) es un tipo de descentralizado y distribuido arquitectura de red en la que cada nodos de la red (llamados "pares") actúan como los proveedores y consumidores de recursos, en contraste con la centralizada cliente-servidor de modelo donde solicitud nodos cliente el acceso a los recursos proporcionados por los servidores centrales.

ORM.- El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia.

RDBMS.- Una Base de Datos Relacional, es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas

SQL.- El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

XML.- Siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible

Xquery.- Es un lenguaje de consulta diseñado para colecciones de datos XML. Es semánticamente similar a SQL, aunque incluye algunas capacidades de programación.

XFS.- Es un sistema de archivos de 64 bits con journaling de alto rendimiento creado por SGI (antiguamente Silicon Graphics Inc.) para su implementación de UNIX llamada **IRIX**. En mayo de 2000, SGI liberó XFS bajo una licencia de código abierto.

4.4 BIBLIOGRAFÍA

10gen, Inc. (2010). *mongoDB. 2010*. Obtenido de http://www.mongodb.org

Apache Software Foundation. (22 de 10 de 2013). *Apache HBASE*. Obtenido de Aticulo : http://hbase.apache.org/

Avinash, L. (07 de 01 de 2014). *Cassandra A structured storage system on a P2P Network.*Obtenido de Blog post: https://www.facebook.com/note.php?note_id=24413138919

Browne, J. (11 de 01 de 2011). *Brewer's CAP Theorem*. Obtenido de :julianbrowne: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

Cattell, R. (12 de 11 de 2013). *Scalable SQL and NoSQL Data Stores.*Obtenido de http://cattell.net: http://cattell.net/datastores/Datastores.pdf

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., . . . Gruber, R. E. (05 de 01 de 2014). *Bigtable:.* Obtenido de articulo: http://static.googleusercontent.com/media/research.google.com/es//archive/bigtable-osdi06.pdf

Czura, M. (28 de 09 de 2010). *CouchDB In The Wild. 2008–2010.*Obtenido de Wiki article, version 97: http://wiki.apache.org/couchdb/CouchDB_in_the_wild

Dans, E. (28 de 11 de 2011). *Entender el futuro: la evolución de las bases de datos*. Obtenido de www.enriquedans.com: http://www.enriquedans.com/2011/11/entender-el-futuro-la-evolucion-de-las-bases-de-datos.html

David , K., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., & Daniel . (27 de 12 de 1997). *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web.*Obtenido de Articulo: http://thor.cs.ucsb.edu/~ravenben/papers/coreos/kll+97.pdf

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., . . . Vogels, W. (1 de 09 de 2007). *Dynamo: Amazon's Highly Available Key-value Store*. Obtenido de articulo: http://www.read.seas.harvard.edu/~kohler/class/cs239-w08/decandia07dynamo.pdf

Edgar F. Codd. (24 de 12 de 2013). *Reglas de Codd.* Obtenido de Documento: http://www.galeon.com/nevifi/Archivos/Codd.pdf

Ellis, J. (14 de 09 de 2013). *NoSQL Ecosytem.* Obtenido de Blog: http://www.rackspacecloud.com/blog/2009/11/09/nosql-ecosystem/

Fundación Wikimedia, Inc. (5 de 11 de 2013). *BERKELEYDB*. Obtenido de Wikipedia.org: http://es.wikipedia.org/wiki/Berkeley_db

Fundación Wikimedia, Inc. (22 de 10 de 2013). *ORM Mapeo Objeto Relacional*. Obtenido de Wikipedia.org: http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

Fundación Wikimedia, Inc. (8 de Diciembre de 2013). *SENTENCIA JOIN*. Obtenido de Wikipedia.org: http://es.wikipedia.org/wiki/Join

Garcete, A. (05 de 01 de 2014). *Base de Datos Orientado a Columnas*. Obtenido de Tesis: http://www.jeuazarru.com/docs/dbco.pdf

Gilbert, S., & Lynch, N. (07 de 12 de 2013). *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*. Obtenido de http://lpd.epfl.ch/sgilbert/pubs/BrewersConjecture-SigAct.pdf

Gonzales, M. P. (11 de 12 de 2013). *¿Qué son las Bases de Datos NoSQL?* Obtenido de BLOGSPOT: http://manuelpereiragonzalez.blogspot.com/2011/05/que-son-las-bases-de-datos-nosql.html

Hoff, T. (28 de 10 de 2009). *MySQL or Memcached or Tokyo Tyrant?*Obtenido de Blog post: http://highscalability.com/blog/2009/10/28/and-the-winner-is-mysql-or-memcached-ortokyo-

Ippolito, B. (28 de 03 de 2009). *Drop ACID and think about Data*. Obtenido de Talk at Pycon: http://blip.tv/file/1949416/

Kallen, N., Pointer, R., Kalucki, J., & Ceaser, E. (08 de 01 de 2014). *FlockDB*. Obtenido de artiuculo: https://github.com/twitter/flockdb

Ken, N. (2009). *Databases in the cloud*. Obtenido de Article in Dr. Drobb's Magazine: http://www.drdobbs.com/database/218900502

Labs, F. (2010). *Tokyo Cabinet: a modern implementation of DBM.*Obtenido de http://1978th.net/tokyocabinet/

Lakshman, A., & Malik, P. (12 de 01 de 2014). *Cassandra - A Decentralized Structured Storage System.*Obtenido de documento: http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf

Leslie, L. (14 de 25 de 2013). *The part-time parliament. In: ACM Transactions on Computer Systems.*Obtenido de Articulos Online: http://research.microsoft.com/en-us/um/people/lamport/pubs/lamport-paxos.pdf

Lin, L. (20 de 04 de 2009). *Some Notes on distributed key value Stores*. Obtenido de Blog post: http://randomfoo.net/2009/04/20/some-notes-on-distributed-key-stores

LITH, A., & MATTSSON, J. (2010). *Investigating storage solutions for large data.* (D. o. Engineering, Ed., & M. G. Inuca , Trad.) Sweden, Sweden.

López, F. A. (09 de 09 de 2011). *Articulo: Bases de Datos NOSQL*. Obtenido de BLOGSPOT: http://federico-wiesse.blogspot.com/2011/09/bases-de-datos-nosql_09.html

Mancuello, J. P. (11 de 12 de 2013). *Bases de datos NOSQ*. Obtenido de http://mistock.lcompras.biz/index.php?option=com_content&view=article&id=2395:httpwwwgenbetadevcombases-de-datosel-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de&catid=60:seginfo2011&Itemid=120: http://mistock.lcompras.biz/

Martin, L. (27 de 06 de 2013). *Principales tecnologías Big Data: NoSQL*. Obtenido de BLOG: http://www.brainsins.com/es/blog/principales-tecnologias-big-data-nosql/107943

MegaSonik. (1 de 04 de 2010). *NOSQL.es.* Obtenido de blog: http://www.nosql.es/blog/nosql/amazon-dynamo.html

MongoDB Inc. (02 de 01 de 2014). *MongoDB Información general*. Obtenido de Articulos: https://www.mongodb.com/products/mongodb

MongoDB.org. (20 de 01 de 2014). *Introduction to MongoDB.*Obtenido de Blogg: http://docs.mongodb.org/manual/core/introduction/

NAWROTH, A. (08 de 01 de 2014). *Las 10 mejores maneras de conocer Neo4j.* Obtenido de blog: http://blog.neo4j.org/2010/02/top-10-ways-to-get-to-know-neo4j.html

Neo4j. (05 de 01 de 2014). *Capítulo 19. API REST.* Obtenido de Documento: http://docs.neo4j.org/chunked/stable/rest-api.html

North, K. (01 de 09 de 2013). *Databases in the cloud.*Obtenido de dR. DROBB´S mAGAZINE: http://www.drdobbs.com/database/218900502

Paramio, C. (26 de 04 de 2011). *El concepto NoSQL, o cómo almacenar tus datos en una base de datos no relacional*. Obtenido de www.genbetadev.com: http://www.genbetadev.com/bases-de-datos/

Perez, M. (25 de 01 de 2014). *Cinco ventajas competitivas para las empresas que usen Big Data*. Obtenido de blogg: http://www.dirigentesdigital.com/tecnologia/38-tecnologia/214227-cinco-ventajas-competitivas-para-las-empresas-que-usen-big-data

Pritlove, T., Lehnardt, J., & Lang, A. (10 de 06 de 2009). *ouchDB – Die moderne Key/Value-Datenbank lädt Entwickler zum Entspannen ein.*Obtenido de Chaosradio Express Episode 125, Podcast published: http://chaosradio.ccc.de/cre125.html

Richard, J. (19 de 01 de 2009). *Anti-RDBMS: A list of distributed key-value stores*. Obtenido de Blog: http://www.metabrew.com/article/anti-rdbms-a-list-of-distributed-key-value-stores/

Sánchez, J. (01 de 01 de 2004). *Principios sobre Bases de Datos Relacionales.* Obtenido de Articulo: http://www.jorgesanchez.net/bd/bdrelacional.pdf

Scofield, B. (12 de 11 de 2013). *NoSQL Death to relational Databases.*Obtenido de Blog: http://www.slideshare.net/bscofield/nosql-codemash-2010

Silvescu, A., Caragea, D., & Atramentov, A. (25 de 12 de 2013). *Graph Databases.* Obtenido de Articulo: http://www.cs.iastate.edu/~silvescu/papers/gdb/report.pdf

Solis, S. M. (04 de 01 de 2014). *Herramientas para Big Data Neo4J .* Obtenido de Blog: http://santiagomarquezsolis.com/herramientas-para-big-data-neo4j-parte-1/

Steve, C. (01 de 2009). *MemcacheDB*. Obtenido de articulo: http://memcachedb.org/

Strauch, C. (2012). *NoSQL Databases.* Hochschule der Medien, Stuttgart.

TechPluto. (12 de 12 de 2013). *Characteristics of Web 2.0 Technology*. Obtenido de blog: http://www.techpluto.com/web-20-services/

Tiwari, S. (2011). *PROFESSIONAL NoSQL.* Indianapolis, Indiana: John Wiley & Sons, Inc.

What is a REST API [duplicate].(09 de 01 de 2014). Obtenido de Articulo: http://stackoverflow.com/questions/2217758/what-is-a-rest-api

White, T. (27 de 11 de 2007). *Consistent Hashing.* Obtenido de Blogger: https://weblogs.java.net/blog/tomwhite/archive/2007/11/consistent_hash.html

Wiesse, F. (09 de 09 de 2011). *Mi universo es una base de datos.* Recuperado el 18 de 12 de 2013, de Blog: http://federico-wiesse.blogspot.com/2011/09/bases-de-datos-nosql_09.html

4.4 LINKOGRAFIA

WIKIPEDIA. (23 de 12 de 2013). *BSON*. Obtenido de articulo: http://es.wikipedia.org/wiki/BSON

Wikipedia. (24 de 11 de 2013). *Hash Table*. Obtenido de Article, Wikipedia: http://en.wikipedia.org/wiki/Hashmap

Wikipedia. (25 de 01 de 2013). *Lenguaje de Programación*. Obtenido de articulos: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n

Wikipedia. (23 de 11 de 2013). *NoSQL*. Obtenido de Article, Wikipedia: http://en.wikipedia.org/wiki/NoSQL#cite_note-6

wikipedia. (28 de 01 de 2014). *Servidor Web*. Obtenido de Articulo: http://es.wikipedia.org/wiki/Servidor_web#Arquitectura

Wikipedia. (02 de 01 de 2014). *Tupla*. Obtenido de Articulo: http://es.wikipedia.org/wiki/Tupla

Wikipedia, the free encyclopedia. (12 de 24 de 2013). *Berkeley DB*. Obtenido de Berkeley DB: http://en.wikipedia.org/wiki/BerkeleyDB

Wikipedia.org. (25 de 12 de 2013). *Base de datos orientada a grafos*. Obtenido de Articulos: http://es.wikipedia.org/wiki/Base_de_datos_orientada_a_grafos

www.imaginanet.com. (05 de 06 de 2011). *www.imaginanet.com*. Obtenido de BLOG: http://www.imaginanet.com/blog/bases-de-datos-nosql-y-php.html

Yen, S. (05 de 11 de 2009). *A Yes for a NoSQL Taxonomy.* Obtenido de Blog: http://highscalability.com/blog/2009/11/5/a-yes-for-a-nosql-taxonomy.html

4.5 ANEXOS

A. DICCIONARIO DE DATOS

A.1 NOMBRES DE LAS TABLAS DEL SISTEMA

- TAB_JUGADORES
- TAB_CAMPEONATO
- TAB_EQUIPO
- TAB_PASES
- TAB_CATEGORIA
- TAB_USUARIOS
- TAB_TARJETAS

A.2 EXPLICACION DE LOS CAMPOS DE CADA COLECCION

- TAB_JUGADORES

Es un esquema en el cual se almacena la información de la persona que pertenece a la LIGA

TABLA 11: Anexo B.2 Atributos del documento Tab_Jugadores.

Atributo	Valor	Tipo de Dato	Descripción
Cedula_id	NULL	Int32	Identifica el Jugador
Nombres	NULL	String	Corresponde al Nombre
Apellidos	NULL	String	Corresponde a el apellido
Direccion	NULL	String	Corresponde a la Direccion
Telefono	NULL	Int32	Corresponde al Telefono
Fecha_Nacimiento	NULL	Date	Corresponde a la Fecha
Residencia	NULL	String	Corresponde a la Residencia
Sexo	NULL	String	Corresponde al sexo
Email	NULL	String	Corresponde al correo
Foto	NULL	DataBinaria	Corresponde a la foto
Categoria	NULL	String	Corresponde a la Disciplina
Numero Camiseta	NULL	Int32	Corresponde a la Numero
Equipo_id	YES	Int32	Corresponde al codigo

- TAB_CAMPEONATO

Es un esquema para registrar los datos del campeonato.

TABLA 12: Anexo B.2 Atributos del documento Tab_Campeonato.

Atributo	Valor	Tipo de Dato	Descripción
Campeonato_id	YES	Int32	Corresponde al Código
Nombre	NULL	String	Corresponde al Nombre
Fecha	NULL	Date	Corresponde a la fecha
Observaciones	NULL	String	Corresponde a las observaciones

- TAB_EQUIPO

Es un esquema para registrar los datos de los equipos que pertenecen a LIGA

TABLA 13: Anexo B.2 Atributos del documento Tab_Equipo

Atributo	Valor	Tipo de Dato	Descripción
Equipo_id	YES	Int32	Código del equipo
Nombre	NULL	String	Corresponde al Nombre
Fecha_fundacion	NULL	Date	Corresponde a la fecha
Observaciones	NULL	String	Corresponde a la Observación

- TAB_PASES

Es un esquema para registrar los documentos generados al realizar los pases de los jugadores

TABLA 14: Anexo B.2 Atributos del Documento de base Tab_Pases

Atributo	Valor	Tipo de Dato	Descripción
Pases_id	YES	Int32	Código Pases
Equipo_id	YES	Int32	Código Equipo
Cedula_id	YES	Int32	Codigo Cedula
Fecha Pase	NULL	date	Corresponde a la Fecha
Observaciones	NULL	String	Corresponde a la observación

- TAB_CATEGORIA

Es un esquema en que tiene la información de categorías de la Liga.

TABLA 15: Anexo B.2 Atributos del documento de base de datos Tab_Categoria

Atributo	Valor	Tipo de Dato	Descripción
Categoría_id	YES	Int32	Codigo Categoria
Tipo	NULL	String	Corresponde al Tipo
Parámetros	NULL	String	Corresponde a los Parametros

- TAB_USUARIOS

Es un esquema se tendrá la información de los usuarios que tendrán acceso al sistema

TABLA 16: Anexo B.2 Atributos del documento de base de datos Tab_Usuarios

Atributo	Valor	Tipo de Dato	Descripción
Usuarios_id	YES	Int32	Codigo Usuarios
Tipo_Usuarios	NULL	String	Corresponde al Tipo
Password	NULL	String	Corresponde a la contraseña
NombreUser	NULL	String	Corresponde al Usuario

- TAB_TARJETAS

Es un esquema en la cual se ingresara las tarjetas que tengan los jugadores

TABLA 17: Anexo B.2 Atributos del documento de base de datos Tab_tarjetas

Atributo	Valor	Tipo de Dato	Descripción
Tarjetas_id	YES	Int32	Código Tarjetas
Equipo_Id	NULL	Int32	Código Equipo
Cedula	NULL	Int32	Corresponde a la cedula
Fecha_tarj	NULL	date	Corresponde a la Fecha
Tipo	NULL	String	Corresponde al Tipo
Numero	NULL	Int32	Corresponde al numero

- TAB_SANCIONES

Es un esquema que tendrá información sobre las sanciones

TABLA 18: Anexo B.2 Atributos del documento de base de datos Tab_Sanaciones

Atributo	Valor	Tipo de Dato	Descripción
Sanciones_id	YES	Int32	Codigo Sanciones
Acumuladas	NULL	Int32	Corresponde al número de tarjetas
Tipo_Sanciones	NULL	String	Corresponde a las sanciones
Observaciones	NULL	String	Corresponde a las observaciones

B. MANUAL DE INSTALACIÓN

B.1 INSTALACIÓN DE MONGODB PARA WINDOWS 64BIT

Mongodb es un Funciona en sistemas operativos Windows, Linux, OS X y Solaris, en este caso se realizar la instalación de este software en Windows.

- Definir la versión de mongodb y plataforma de instalación.

Mongodb V2.4.9 para Windows 64-bit.

- Si no conocemos la plataforma en la que estamos trabajando en el cmd de windows ejecutamos el siguiente comonado:

wmic os get osarchitecture

- Descargamos la versión de mongodb para nuestro sistema operativo del siguiente link

http://www.mongodb.org/downloads?_ga=1.248779843.553799415.142170957

- Una vez descargado abrimos la carpeta de descarga, abrimos el archivo descargado y lo descargamos, esta carpeta la movemos al disco C:\mongodb.

Mongodb no contiene programas adicionales para su ejecución por lo tanto el directorio puede estar en otro lugar C:\Test\Mongodb

MongoDB requiere de un directorio de datos para almacenar todos los datos, mongodb posee una ruta predeterminada es C:\data\db, se debe crear esta dirección.

- Una vez terminada la creación de carpetas iniciamos el servicio de mongo que se encuentra en la carpeta C:\Mongodb\mongod.exe

En la consola de comandos desplegara un mensaje indicándonos que se están esperando conexiones, esto indica que está funcionando correctamente

Nota: MongoDB tiene el puerto predeterminado 27017

C. PROTOTIPO DE INTERFAZ DE USUARIO

Toda obra de diseño conlleva la elección de los elementos básicos que la van a formar. Una composición gráfica está destinada a representar un medio de comunicación entre personas y ordenadores, este es el caso de una interfaz de usuario.

Es decir el diseño gráfico aplicado a la construcción de interfaces Web2.0, para conseguir un medio de interacción entre los usuarios y el conjunto de páginas de un sitio Web y las aplicaciones que corren por debajo de ellas.

Propósito

Dar a conocer a los interesados la plantilla que regirá las aplicaciones que se desean implementar en la aplicación web 2.0, así como también los archivos de configuración, el mismo que servirá de base para las aplicaciones futuras.

Descripción

En este documento presenta al interesado los siguientes aspectos

- Archivos de Configuración necesarios para la presentación gráfica.
- Diseños de plantillas CSS, javascript y HTML.

C.1 ARCHIVOS DE CONFIGURACIÓN

- **Configuración del Servidor**

Para poder visualizar nuestra página Web se configurara el servidor de aplicaciones Xampp en el cual viene incluido los paquetes PHP 5.2 y apache 2.0

Vamos a editar el archivo de configuración para crear nuestro host o dominio.

Abrirmos el archivo Vhost que se encuentra en la dirección C:\xampp\apache\conf\extra\ httpd-vhosts.conf aquí ingresamos la siguiente línea de código.

<VirtualHost *:80>

ServerAdmin sanpablo@deportesp.com

DocumentRoot /htdocs/LigaSP/Index.php

ServerName ligasanpablo.com

ErrorLog @sanpablo@/ligasanpablo.com-error_log

CustomLog@sanpablo@/ligasanpablo.com-access_log common

</VirtualHost>

Una vez ingresada guardamos los cambios realizados y nos dirigimos a re direccionar el tráfico de nuestra pc a la del host configurado en

C:\Windows\System32\drivers\etc\hosts se edita el contenido cambiando el localhost por el de nuestra pagina “ligasanpablo.com”

127.0.0.1 localhost

::1 localhost

Ahora quedaría de la siguiente forma

127.0.0.1 ligasanpablo.com

::1 www.ligasanpablo.com

Con esto configuraríamos nuestro servidor virtual en nuestra maquina local.

C.2 PLANTILLAS DE PERSONALIZACIÓN DE LA APLICACIÓN

En la aplicación existen varios archivos CSS los cuales permiten dar una mejor presentación a la página web estos son los siguientes:

- Style.css Este css o plantilla contiene estilos básicos para la presentación de la primera página de nuestra aplicación, estilos con los cuales mejoramos las vistas de menu, fondos y textos, de las paginas principal, misión y visión, reseña histórica, contactos, como un primer plano de navegación en la Web.
- Table.css.- Mejora la estructura de la tabla de contactos
- StyleP.- Mejora los estilos de las letras y fondos del aplicación Web 2.0
- Animate.css,styles.css y reset.css nos muestran una presentación amigabe al momento de ingresar en el sistema
- Layout.css y menú.css presentan estilos con los cuales mejoramos la interfaz del sistema para ser más amigable con el usuario esto es en la aplicación Web 2.0 para la Liga, estos presentan iconos labels, inputs checkbox, selectbox, text, etc.

También tenemos los javascript que son los que ayuda a mejorar la presentación de los calendarios de nuestra aplicación y movimiento de imágenes.

D. MANUAL DE USUARIO

Atreves de este manual el usuario podrá utilizar el sistema desarrollado de forma correcta para lo cual debe tener en cuenta las siguientes indicaciones.

Ingresar a un navegador puede ser este (Firefox o Google Crome);

Ingresar al portal de la universidad http://www.ligasanpablo.com/ LigaSanPablo/

Aparece nuestra página de portada

Logearse con su cuenta y clave de Usuario.

FIGURA 20: Pantalla principal o portada del Aplicacion

Hay que tener en cuenta que el menú que aparece en la primera página estará presente en las páginas Reseña Histórica, MisiónVisión, Contactos y Equipos, El Enlace Contactos/Login nos dirige al aplicación Web 2.0 en la cual se explicara más adelante.

FIGURA 21: Menú de Portada

Fuente:[Propia]

D.1 INGRESADO AL SISTEMA

En la siguiente pantalla se muestra el login o pantalla de ingreso de usuarios y contraseña. Con la cual se conectara a la base de datos MongoDb.

FIGURA 22: Pantalla de Ingreso al Sistema

Fuente:[Propia]

D.1.1 SECCIÓN LIGA

En esta sección de liga se despliega un menú con dos opciones las cuales permite el ingreso de los datos a los documentos Campeona y Equipos de la liga.

Para poder crear campeonatos debemos dar clic en el enlace Liga\campeonato, de la misma forma para crear los equipos hacemos clic en Liga\Equipos

FIGURA 23: Pantalla del Sistema para Ingreso de datos Campeonato

Fuente:[Propia]

D.1.2 SECCIÓN JUGADORES

En esta sección podemos ingresar a realizar el registro de información de jugadores, tarjetas y pases para lo cual se deberá clic en el acceso correspondiente. En cada formulario nos permitirá realizar la creación de un nuevo documento, modificar, eliminar y guardar.

FIGURA 24: Pantalla del Sistema para Ingreso de datos jugadores

Fuente:[Propia]

D.1.3 SECCIÓN REPORTES

En esta sección podremos visualizar e imprimir los reportes de los jugadores, carnets, nominas, reportes de tarjetas, Documentos de Pases.

FIGURA 25: Pantalla del Sistema para visualizar los reportes

Fuente:[Propia]

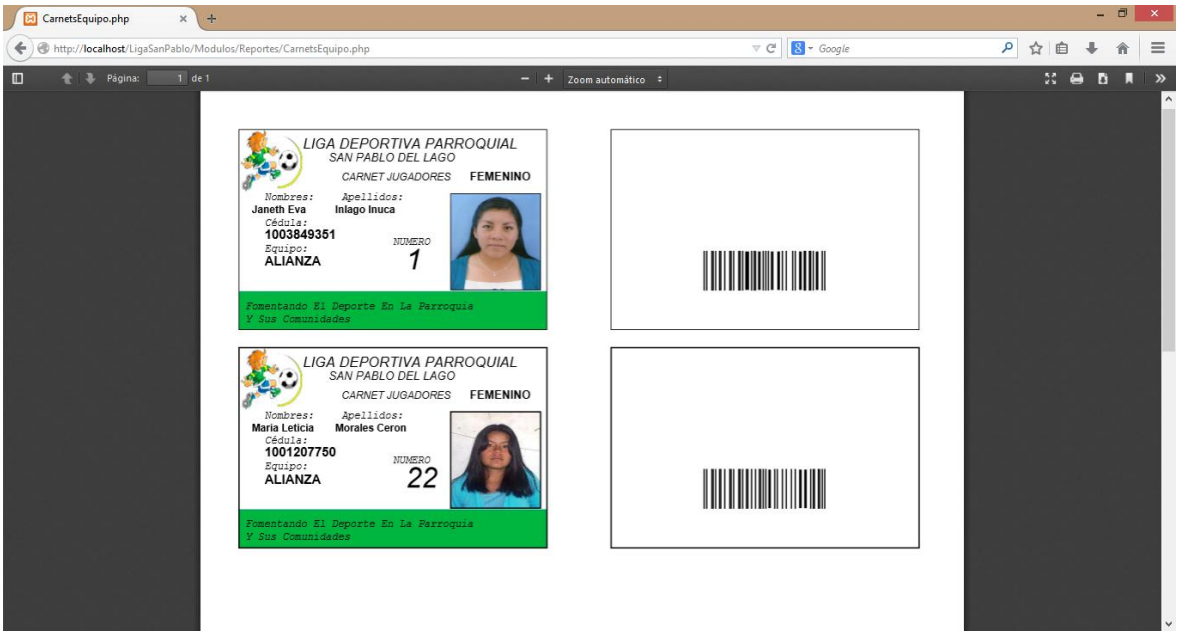


FIGURA 26: Pantalla del Sistema para imprimir los reportes.

Fuente: [Propia.]

FIGURA 27: Pantalla del Sistema para imprimir los reportes Nominas.

Fuente: [Propia.]

FIGURA 28: Pantalla del Sistema para imprimir los reportes Sancionados.

Fuente: [Propia.]

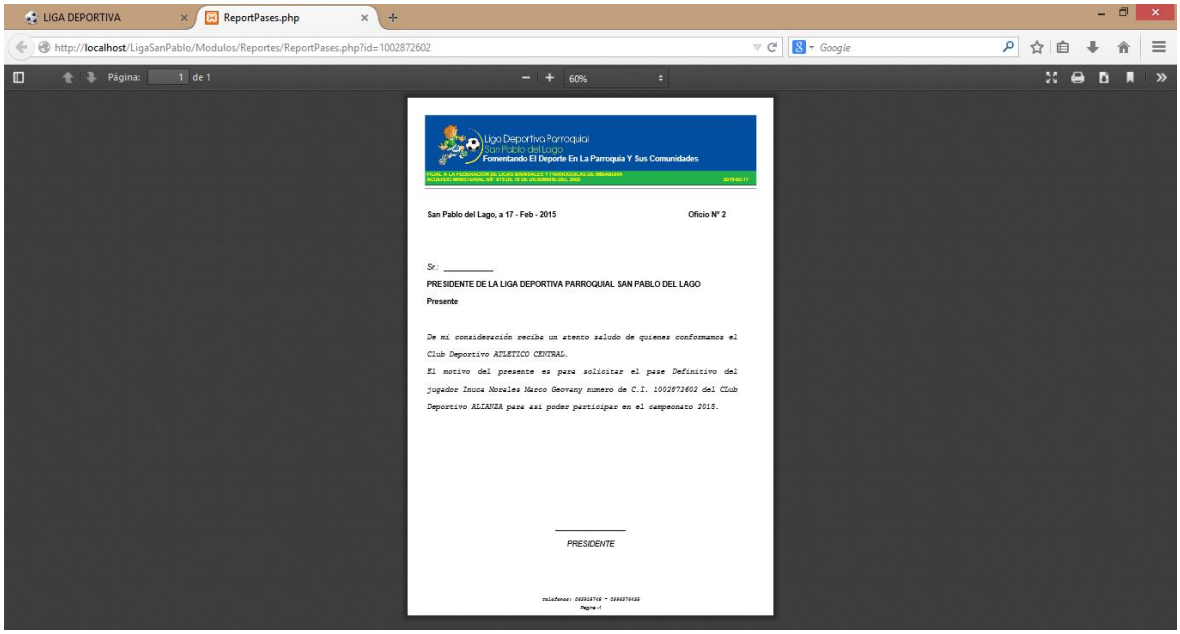


FIGURA 29: Pantalla del Sistema para imprimir los reportes de oficios Pases.

Fuente: [Propia.]