UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

"Trabajo de grado previo a la obtención del

Título de Ingeniero en Sistemas y Computación"

TRABAJO DE GRADUACIÓN

TÍTULO DEL PROYECTO

ESTUDIO COMPARATIVO DE MODELOS DE BASES DE DATOS RELACIONALES Y NO RELACIONALES Y SU INCIDENCIA DE LOS PROCESOS EN SISTEMAS INFORMÁTICOS.

CASO PRÁCTICO: SISTEMA DE GESTIÓN VEHICULAR.

AUTORES:

PAULINA ALEXANDRA GAONA GUTIERREZ

EDISON PATRICIO SANDOVAL NARVAEZ

Directora: Ing. Anita Congacha

Riobamba – Ecuador

2013

TRABAJO DE GRADUACIÓN

Título del proyecto

ESTUDIO COMPARATIVO DE MODELOS DE BASES DE DATOS RELACIONALES Y NO RELACIONALES Y SU INCIDENCIA DE LOS PROCESOS EN SISTEMAS INFORMÁTICOS.

CASO PRÁCTICO: SISTEMA DE GESTIÓN VEHICULAR.

AUTORES:

PAULINA ALEXANDRA GAONA GUTIERREZ

EDISON PATRICIO SANDOVAL NARVAEZ

Directora: Ing. Anita Congacha

Riobamba – Ecuador

2013

Los miembros del Tribunal de Graduación del proyecto de investigación de título: Estudio Comparativo de Modelos De Bases De Datos Relacionales y No Relacionales y su incidencia de los procesos en Sistemas Informáticos. **Caso Práctico:** Sistema de Gestión Vehicular.

Presentado por: Paulina Alexandra Gaona Gutiérrez, Edison Patricio Sandoval Narváez y dirigida por: Ing. Anita Elizabeth Congacha.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Anita Congacha ----------

Directora de la Tesis **Firma**

Ing. Fernando Molina ----------

Presidente del Tribunal **Firma**

Ing. Jorge Delgado ----------

Miembro del Tribunal **Firma**

AUTORÍA DE LA INVESTIGACIÓN

"La responsabilidad del contenido de este Proyecto de Graduación, corresponde exclusivamente a: Paulina Alexandra Gaona Gutiérrez, Edison Patricio Sandoval Narváez autores del proyecto, a la Ing. Anita Congacha Tutora, y al Departamento de Transportes, el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

AGRADECIMIENTO

Agradezco ante todo a Dios por permitirme estar donde estoy, a la Universidad Nacional de Chimborazo a la Facultad de Ingeniería y por ende a todos los docentes que además de impartirme sus conocimientos en la ciencia también lo hicieron para formarme como persona a todos mis compañer@s de clases por que aun sin saberlo me enseñaron muchas cosas para enfrentar los problemas, a mi familia principalmente a mis padres, ya que en cada uno de mis pasos para la culminación de mi meta estuvieron brindándome su apoyo, y no dejaron de soltarme aun en los momentos más difíciles sin ustedes no hubiera podido hacerlo. A Edison por acompañarme en este trayecto de cumplir una meta más de mi vida.

Paulina A. Gaona G.

Agradezco a Dios y a mis padres por apoyarme moral y económicamente en todo este tiempo, por estar juntos hasta en los peores momentos, gracias por sus consejos y reproches que me enseñaron a ser la persona que ahora soy. A mis compañer@s que compartiendo conocimientos sacamos a flote nuestros valores como personas. A la Universidad Nacional de Chimborazo por darme la oportunidad de formarme en sus aulas, a todos mis profesores y todas las personas que me impartieron su tiempo y sus conocimientos. A Paulina por todo el apoyo y comprensión que me ha brindado para cumplir mi sueño.

Edison P. Sandoval N.

DEDICATORIA

Dedico la presente tesis a mi familia en especial a mis padres, a mi mami por ser mi ejemplo de lucha y perseverancia de no dejarme rendir ante cualquier obstáculo que se interpuso en mi camino por su comprensión ante los errores que cometí, gracias por ser mi fuerza en mis momentos de debilidad mi guía en mis momentos de oscuridad gracias por ser ese ejemplo de mujer a seguir, a mis hermanos por estar junto a mi apoyándome en todo momento, y a todas esas personas que creyeron en mí y a las que no también.

Paulina A. Gaona G.

Dedico este trabajo a mis padres, a mi mami que compartió la mayor parte de mi vida junto a mí, con esto no quiero despreciar a mi padre que por su trabajo permaneció distanciado, gracias viejitos por todo ese apoyo tenaz que me brindaron durante toda mi carrera, sin su apoyo no hubiera podido culminar esta etapa. Dedico de manera especial a mi hermana, tómalo como ejemplo a seguir y nunca mal interpretes los consejos de mis padres. Estoy consciente cuanto han invertido en mí, pero en pago a todo eso, este es el resultado de su inversión.

Edison P. Sandoval N.

INDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

El estudio de esta tesis es el análisis comparativo de los modelos de bases de datos relacionales y no relacionales y su incidencia en los procesos de los sistemas informáticos, en este caso particular en el sistema del Departamento de transportes de la UNACH.

Esta investigación se realizó con el fin de conocer las bases de datos relacionales y NoSQL, estableciendo las diferencias de cada una para determinar cuál de las dos bases de datos es factible implementar en el departamento de transportes.

Después del análisis experimental se eligió el modelo relacional, porque en el departamento no existe un gran número de usuarios, volumen de datos e información, y no es muy escalable.

Para llegar a la determinación de que Modelo es factible se utilizó la herramienta Jmeter el cual es un proyecto de Apache que se puede utilizar como una herramienta de prueba de carga para analizar y medir el rendimiento de una variedad de servicios.

La información acerca de los modelos de base de datos relacionales y NoSQL, se detalla en nuestro trabajo, sus respectivas arquitecturas y procesos y también se describen las ventajas y desventajas de su uso.

Este prototipo se basa en las leyes y los principios colectivos porque somos dos estudiantes que diseñamos el sistema del Departamento de Transportes del la UNACH.

SUMMARY

The study of this thesis is a comparative analysis of relational and non-relational databases models and its impact on computer systems process, in particular case in the transport system department for UNACH.

This research was performed in order to know the relational and NoSQL, database establishing the differences of each one to determine which of the two databases is feasible to implement in the transport system department

After experimental analysis the relational model was chosen because in the department does not exist a large number of users, volume of data and information used daily that it is not very scalable.

In reaching the determination that Model Database feasible, Jmeter tool was used which is an Apache project that can be used as a load testing tool to analyze and measure the performance of a variety of services.

The information about Relational and NoSQL database models are detailed in our work and their respective architectures and processes, also describes the advantages and disadvantages of their use.

This prototype was based on laws and principles collective because we are two students who designed the Department transport system for UNACH.

INTRODUCCIÓN

No cabe duda que la información es la base de nuestra sociedad, recibimos y manejamos volúmenes enormes de datos y el ordenador es la herramienta que nos permite almacenar y manipular dicha información. Para guardar y recuperar la información necesitamos de un sistema de almacenamiento que sea fiable, fácil de manejar, eficiente, y de aplicaciones capaces de llevar a cabo el proceso de almacenar datos y obtener resultados a partir de la información almacenada.

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.

Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado un amplio conjunto de conceptos y técnicas para la gestión de los datos.

En una computadora existen diferentes formas de almacenar información. Esto da lugar a distintos modelos de organización de la base de datos: jerárquico, red, relacional y orientada a objetos.

Los sistemas relacionales son importantes porque ofrecen muchos tipos de procesos de datos, como: simplicidad y generalidad, facilidad de uso para el usuario final, períodos cortos de aprendizaje y las consultas de información se especifican de forma sencilla.

Pero llegó la web, el software como servicio, los servicios en la nube y las startups de éxito con millones de usuarios, y junto a ello llegaron los problemas de alta escalabilidad**.** Si bien los modelos relacionales se pueden adaptar para hacerlos escalar incluso en los entornos más difíciles, pero a menudo, se hacen cada vez menos intuitivos a medida que aumenta la complejidad.

Triples y cuádruples JOINs en consultas SQL muy complejas, a veces poco eficientes, y sistemas de almacenamiento de resultados en cachés para acelerar la resolución de las peticiones y evitar ejecutar cada vez estas pesadas operaciones, son el pan de cada día en muchos de estos proyectos de software.

Los sistemas NoSQL intentan atacar este problema proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas (también llamado JOIN) teniendo que recurrir a la desnormalización de datos.

Algunas implementaciones bien conocidas de NoSQL son: CouchDB, MongoDB, RavenDB, Neo4j, Cassandra, BigTable, Dynamo, Riak, Hadoop, entre otras.

Este documento se centrará en el estudio comparativo de modelos de bases de datos relacionales y no relacionales y su incidencia en los procesos del sistema informático del departamento de Transportes de la UNACH. En las siguientes secciones se analizarán todos los aspectos relacionados así como características, ventajas y otros aspectos de los dos tipos de bases de datos relacionales y no relacionales.

FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1

1 ARQUITECTURAS DE BASES DE DATOS

1.1 BASES DE DATOS RELACIONALES

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo.

Una base de datos relacionales 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. Permiten establecer relaciones entre los datos, y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: "Modelo Relacional".

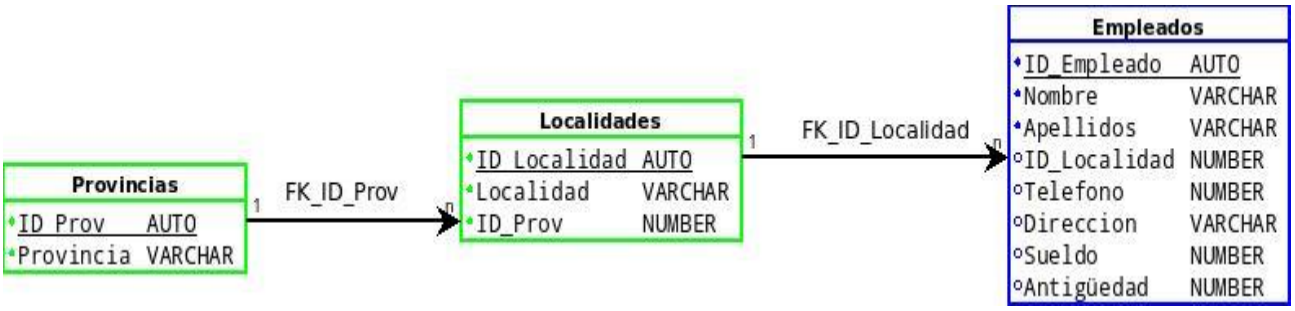


Figura 1: Ejemplo de Modelo Relacional

1.1.1 CARACTERÍSTICAS DE BASES DE DATOS RELACIONALES

Las bases de datos relacionales tienen algunas características que las distinguen de otros tipos de bases de datos:

- ✓ Una base de datos relacional consta de una o más tablas de dos dimensiones de valores de datos, y la construcción de dichas tablas viene definida por reglas

muy simples. Cada tabla corresponde a un tipo de entidad conceptual, que no contenga grupos de repetición.

- ✓ Las relaciones entre dos o más tablas se establecen en virtud de los valores de datos comunes contenidos en las tablas correspondientes.
- ✓ Se ha desarrollado una metodología sistemática para la transformación de un conjunto de tipos de entidad (que representan un diseño conceptual) en un conjunto correspondiente de tablas. El objetivo de esta metodología es la generación de un grupo de tablas en el cual se minimice la duplicación de datos, y en el que se hayan eliminado determinados problemas asociados con el mantenimiento de las tablas.
- ✓ Los sistemas de bases de datos relacionales suelen tener asociados lenguajes de consulta de alto nivel, que facilitan la realización de búsquedas y de actualizaciones en la base de datos, de la manera más flexible posible.

1.1.2 ELEMENTOS DE BASES DE DATOS RELACIONALES

1.1.2.1 Relaciones base y derivadas

En una base de datos relacional, todos los datos se almacenan y se accede a ellos por medio de relaciones. Las relaciones que almacenan datos son llamadas "relaciones base" y su implementación es llamada "tabla". Otras relaciones no almacenan datos, pero son calculadas al aplicar operaciones relacionales. Estas relaciones son llamadas "relaciones derivadas" y su implementación es llamada "vista" o "consulta". Las relaciones derivadas son convenientes ya que expresan información de varias relaciones actuando como si fuera una sola.

1.1.2.2 Restricciones

Una restricción es una limitación que obliga el cumplimiento de ciertas condiciones en la base de datos. Algunas no son determinadas por los usuarios, sino que son inherentemente definidas por el simple hecho de que la base de datos sea relacional. Algunas otras restricciones las puede definir el usuario, por ejemplo, usar un campo con valores enteros entre 1 y 10.

Las restricciones no son parte formal del modelo relacional, pero son incluidas porque juegan el rol de organizar mejor los datos.

1.1.2.3 Dominios

Un dominio describe un conjunto de posibles valores para cierto atributo. Como un dominio restringe los valores del atributo, puede ser considerado como una restricción.

1.1.2.4 Clave única

Cada tabla puede tener uno o más campos cuyos valores identifican de forma única cada registro de dicha tabla, es decir, no pueden existir dos o más registros diferentes cuyos valores en dichos campos sean idénticos.
Pueden existir varias claves únicas en una determinada tabla, y a cada una de éstas suele llamársele candidata a clave primaria.

1.1.2.5 Clave primaria

Una clave primaria es una clave única elegida entre todas las candidatas que define unívocamente a todos los demás atributos de la tabla, para especificar los datos que serán relacionados con las demás tablas. La forma de hacer esto es por medio de claves foráneas.
Sólo puede existir una clave primaria por tabla y ningún campo de dicha clave puede contener valores NULL.

1.1.2.6 Clave foránea

Una clave foránea es una referencia a una clave primaria en otra tabla, determina la relación existente en dos tablas. Las claves foráneas no necesitan ser claves únicas en la tabla donde están y sí a donde están referenciadas.

1.1.2.7 Clave índice

La clave índice surge con la necesidad de tener un acceso más rápido a los datos. Los índices pueden ser creados con cualquier combinación de campos de una tabla. Las consultas que filtran registros por medio de estos campos, pueden encontrar los registros de forma no secuencial usando la clave índice.
Las bases de datos relacionales incluyen múltiples técnicas de ordenamiento, cada una de ellas es óptima para cierta distribución de datos y tamaño de la relación.

1.1.2.8 Procedimientos Almacenados

Un procedimiento almacenado es código ejecutable que se asocia y se almacena con la base de datos. Los procedimientos almacenados usualmente recogen y personalizan operaciones comunes, como insertar un registro dentro de una tabla, recopilar información estadística, o encapsular cálculos complejos. Son frecuentemente usados por un API por seguridad o simplicidad.

Los procedimientos almacenados no son parte del modelo relacional, pero todas las implementaciones comerciales los incluyen.

1.1.3 VENTAJAS DEL USO DE BASES DE DATOS RELACIONALES

- ✓ **Obtener más información de la misma cantidad de data**: La base de datos facilita al usuario obtener más información debido a la facilidad que provee esta estructura para proveer datos a los usuarios.
- ✓ **Compartir los Datos**: Usuarios de distintas oficinas pueden compartir datos si están autorizados. Esto implica que si un dato cambia de contenido como por ejemplo la dirección de un cliente, todos los usuarios que pueden acceder ese dato, verán inmediatamente el cambio efectuado.
- ✓ **Balance de Requerimientos Conflictivos**: Para que la Base de Datos trabaje apropiadamente, necesita de una persona o grupo que se encargue de su funcionamiento. El título para esa posición es Administrador de Base de Datos y provee la ventaja de que Diseña el sistema tomando en mente la necesidad de cada departamento de la empresa. Por lo tanto se beneficia mayormente la empresa aunque algunos departamentos podrían tener leves desventajas. Tradicionalmente se diseñaba y programa según la necesidad de cada departamento por separado.
- ✓ **Redundancia controlada**: Debido al sistema tradicional de archivos independientes, los datos se duplicaban constantemente lo cual creaba mucha duplicidad de datos y creaba un problema de sincronización cuando se actualizaba un dato en un archivo en particular.
- ✓ **Consistencia**: Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. La propiedad de

consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido.

- ✓ **Integridad**: La base de datos tiene la capacidad de validar ciertas condiciones cuando los usuarios ingresan datos y rechazan entradas que no cumplan con esas condiciones. El DBA (Data Base Administrator) es responsable de establecer esas validaciones.
- ✓ **Seguridad**: El DBA al tener control central de los Datos, la Base de Datos le provee mecanismos que le permiten crear niveles de seguridad para distintos tipos de Usuarios. Al permitir restringir el acceso a los usuarios, cada tipo de éstos tendrá la posibilidad de acceder a ciertos elementos.
- ✓ **Aumenta la productividad de los programadores**: Debido a que los programadores no se tienen que preocupar por la organización de los datos ni de su validación, se pueden concentrar en resolver otros problemas inmediatos, mejorando de ese modo su productividad.
- ✓ **Mejora el mantenimiento de los programas**: Los datos son independientes de los programas, si ocurre un cambio en la estructura de una tabla, el código no se afecta.
- ✓ **Independencia de los Datos**: Los datos pueden modificarse para mejorar el "performance" de la Base de Datos y como consecuencia, no se tiene que modificar los programas.

1.1.4 DESVENTAJAS DEL USO DE BASES DE DATOS RELACIONALES

- ✓ **Tamaño**: Al proveer todas las ventajas anteriormente nombradas, el Sistema de Manejo de Base de Datos requiere de mucho espacio en disco duro y también requiere de mucha memoria principal (RAM) para poder correr adecuadamente.
- ✓ **Complejidad:** Debido a la cantidad de operaciones y a las capacidades del DBMS, se convierte en un producto complejo de entender.
- ✓ **Costo:** Los productos de Bases de Datos son productos costosos. Esto sin contar los adiestramientos del personal del centro de cómputos y de los usuarios. También existen sistemas gestores de bases de datos open source, que son de código libre, pero el costo aquí seria la capacitación.
- ✓ **Requerimientos adicionales de Equipo** - El adquirir un producto de Base de Datos, requiere a su vez adquirir equipo adicional para poder implementar ese producto como por ejemplo, servidores, memoria, discos duros, etc.

1.1.5 PROPIEDADES ACID

El término ACID expresa la función que las transacciones desarrollan en aplicaciones críticas para una misión. Acuñado por los pioneros en el procesamiento de transacciones, el acrónimo ACID responde a los términos atomicidad, coherencia, aislamiento y permanencia.

Estas propiedades garantizan un comportamiento predecible, reforzando la función de las transacciones como proposiciones de todo o nada diseñadas para reducir la carga de administración cuando hay muchas variables.

Una breve descripción de las cuatro propiedades ACID:

1.1.5.1 ATOMICITY

Garantiza que las transacciones sean una consulta, o grupos de sentencias SQL, no se puedan subdividir, es decir, se ejecutaran enteramente, o no se ejecutaran. Esto implica que en caso de fallo de hardware, fallo de la base de datos, o fallo de la aplicación, se actualizarán todos los datos o ninguno. Esto impide que la base de datos se corrompa o pierda el sincronismo lógico entre los datos.

1.1.5.2 CONSISTENCY

Garantiza que la base de datos siempre estará en un estado consistente. De hecho, garantiza que cada transacción lleve a la base de datos de un estado consistente a otro estado consistente. En este caso, consistencia se refiere a la consistencia interna de relación entre tablas, y la consistencia en los datos almacenados. La propiedad de consistencia no permitiría guardar un entero en un campo float, o no permitiría borrar una fila que es referenciada por otra.

Esta última forma de consistencia se le llama integridad referencial, y hay tres formas de resolver los conflictos de integridad referencial. Se puede anular la transacción, se puede eliminar todos los registros que referencian el registro a eliminar, o se pueden poner a nulos todas las referencias al registro a eliminar. La mayoría de las bases de datos te permiten elegir en cada caso que acción tomar.

1.1.5.3 ISOLATION

Una transacción es una unidad de aislamiento, permitiendo que transacciones concurrentes se comporten como si cada una fuera la única transacción que se ejecuta en el sistema.

El aislamiento requiere que parezca que cada transacción sea la única que manipula el almacén de datos, aunque se puedan estar ejecutando otras transacciones al mismo tiempo. Una transacción nunca debe ver las fases intermedias de otra transacción.
Las transacciones alcanzan el nivel máximo de aislamiento cuando se pueden serializar. En este nivel, los resultados obtenidos de un conjunto de transacciones concurrentes son idénticos a los obtenidos mediante la ejecución en serie de las transacciones. Como un alto grado de aislamiento puede limitar el número de transacciones concurrentes, algunas aplicaciones reducen el nivel de aislamiento en el intercambio para mejorar el rendimiento.

1.1.5.4 DURABILITY

Garantiza que una vez que la transacción se haya completado, siempre se podrá recuperar independientemente de cualquier fallo de hardware o software.
Una vez la base de datos manda la señal de que la transacción ha sido ejecutada correctamente, se puede tener la certeza de que esa transacción esta aplicada correctamente a los datos y se va a poder recuperar. La mayoría de las bases de datos utilizan un log de transacciones, y no consideran que una transacción esta completada hasta que no esté escrita en el log. Este log secuencial permite recuperar los datos de la transacción en caso de un fallo del sistema, y de esta forma asegurar la consistencia de los datos.

1.1.6 SISTEMAS GESTORES DE BASES DE DATOS RELACIONALES

El sistema de gestión de base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, además de proporcionar un acceso controlado a la misma. Se denomina sistema de bases de datos al conjunto formado por la base de datos, el SGBD y los programas de aplicación que dan servicio a la empresa u organización.
El modelo seguido con los sistemas de bases de datos es muy similar al modelo que se sigue en la actualidad para el desarrollo de programas con lenguajes orientados a objetos, en donde se da una implementación interna de un objeto y una especificación externa separada. Los usuarios del objeto sólo ven la especificación externa y no se deben preocupar de cómo se implementa internamente el objeto. Una ventaja de este modelo, conocido como abstracción de datos, es que se puede cambiar la implementación interna de un objeto sin afectar a sus usuarios ya que la especificación externa no se ve alterada. Del mismo modo, los sistemas de bases de datos separan la

definición de la estructura física de los datos de su estructura lógica, y almacenan esta definición en la base de datos. Todo esto es gracias a la existencia del SGBD, que se sitúa entre la base de datos y los programas de aplicación.

Generalmente, un SGBD proporciona los servicios que se citan a continuación:

- ✓ Permite la definición de la base de datos mediante un lenguaje de definición de datos. Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos.
- ✓ Permite la inserción, actualización, eliminación y consulta de datos mediante un lenguaje de manejo de datos. Los lenguajes procedurales manipulan la base de datos registro a registro, mientras que los no procedurales operan sobre conjuntos de registros. En los lenguajes procedurales se especifica qué operaciones se debe realizar para obtener los datos resultados, mientras que en los lenguajes no procedurales se especifica qué datos deben obtenerse sin decir cómo hacerlo. El lenguaje no procedural más utilizado es el SQL, que de hecho, es un estándar y es el lenguaje de los SGBD relacionales.
- ✓ Proporciona un acceso controlado a la base de datos mediante:
    - Un sistema de seguridad, de modo que los usuarios no autorizados no puedan acceder a la base de datos.
    - Un sistema de integridad que mantiene la integridad y la consistencia de los datos.
    - Un sistema de control de concurrencia que permite el acceso compartido a la base de datos.
    - Un sistema de control de recuperación que restablece la base de datos después de que se produzca un fallo del hardware o software.
    - Un diccionario de datos o catálogo, accesible por el usuario, que contiene la descripción de los datos de la base de datos.

A diferencia de los sistemas de ficheros, en los que los programas de aplicación trabajan directamente sobre los ficheros de datos, el SGBD se ocupa de la estructura física de los datos y de su almacenamiento. Con esta funcionalidad, el SGBD se convierte en una herramienta de gran utilidad. Sin embargo, desde el punto de vista del usuario, se podría discutir que los SGBD han hecho las cosas más complicadas, ya que ahora los usuarios ven más datos de los que realmente quieren o necesitan, puesto que ven la base de datos completa. Conscientes de este problema, los SGBD proporcionan un mecanismo de vistas que permite que cada usuario tenga su propia vista o visión de la base de datos. El

lenguaje de definición de datos permite definir vistas como subconjuntos de la base de datos.

Todos los SGBD no presentan la misma funcionalidad, depende de cada producto. En general, los grandes SGBD multiusuario ofrecen todas las funciones que se acaban de citar e incluso más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes. Lo que se pretende es proporcionar un sistema que permita gestionar cualquier tipo de requisitos y que tenga un 100 % de fiabilidad ante cualquier tipo de fallo.

Los SGBD están en continua evolución, tratando de satisfacer los requisitos de todo tipo de usuarios. Por ejemplo, muchas aplicaciones de hoy en día necesitan almacenar imágenes, vídeo, sonido, etc. Para satisfacer a este mercado, los SGBD deben evolucionar. Conforme vaya pasando el tiempo, irán surgiendo nuevos requisitos, por lo que los SGBD nunca permanecerán estáticos.

1.1.7 TIPOS DE BASES DE DATOS

Existen diferentes modelos de base de datos, es decir, diferentes formas de organizar la información. Cada uno de estos modelos tiene ventajas e inconvenientes y ninguno representa un modelo perfecto. Por ello, es fundamental realizar un estudio previo de la información que se ha de manejar para poder elegir uno de los tipos posibles como el que mejor se ajuste a los requisitos previamente indicado.

Otro factor fundamental en la elección del tipo de base de datos es su costo. El costo de una base de datos se fundamenta, en gran medida, en los requisitos necesarios para su manejo, así como en el entorno informático en que debe incluirse.

La base de datos jerárquica y en red, así como las documentales se instalan generalmente, en grandes sistemas de computadores. Las razones para que estos tipos de base de datos necesiten grandes sistemas son, en primer lugar, su complejidad y, en segundo, el hecho de que sus diseños originales se realizaron, fundamentalmente, antes de la proliferación de una microinformática lo suficientemente potente como para manejar enormes volúmenes de datos.

Las bases de datos relacionales, si bien se desarrollaron en su origen para funcionar en grandes sistemas, han experimentado un considerable auge dentro del campo de la microinformática. Una de las razones de este auge es que ha sido más sencilla la

creación de sistemas gestores de bases de datos que soporten el modelo relacional en el entorno microinformático.

1.1.7.1 BASES DE DATOS JERARQUICAS

Este tipo de base de datos tiene su fundamento en la creación de una estructura de almacenamiento de datos en forma de árbol invertido.

En esta estructura los datos completos de un determinado registro se almacenan en diferentes niveles. Al diseñar esta estructura deben tenerse en cuenta los diferentes accesos que van a necesitar los usuarios para consultar la información que contiene la base de datos.

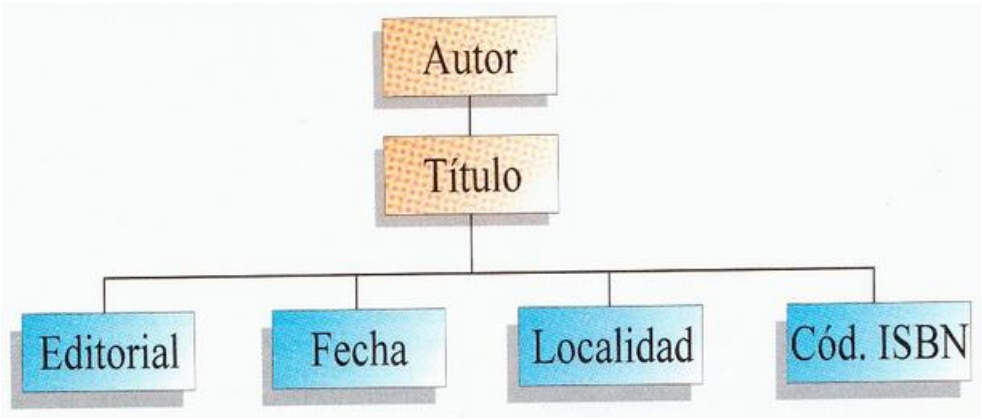


Figura 2: Bases De Datos Jerárquicas

La principal ventaja que presenta este tipo de base de datos es la rapidez en las consultas de información ya que la propia estructura piramidal de los datos permite un rápido acceso a ella.

Las desventajas son importantes, entre las principales tenemos:

- ✓ Se debe realizar un diseño muy robusto, esto es, estable en el tiempo, de la estructura de la información siendo muy complicadas las posteriores modificaciones, así como las labores de mantenimiento de la base de datos.
- ✓ Los accesos a la base de datos también presentan problemas, ya que estos se ven limitados a los registros situados en los niveles superiores de información, con lo cual se restringen las posibilidades de acceder a la información por una gran cantidad de elementos.
- ✓ Es más complicados establece comparaciones entre informaciones situadas a un mismo nivel de la estructura, ya que, al no tener una conexión lógica directa entre ellas, si se intenta realizar consultas entre estas se debe retorcer en el árbol

a través de niveles superiores para llegar a uno que permita acceder a la información solicitada.

1.1.7.2 BASE DE DATOS EN RED

En este tipo de base de datos, la información se almacena también en diferentes niveles pero tiene la ventaja que si se puede acceder a datos situados en el mismo nivel.
La principal ventaja de este modelo es que los accesos a la información son más flexibles en comparación con los de las bases de datos jerárquicas, que son más restringidos.
Las desventajas con relación a las bases de datos jerárquicas son que la velocidad de acceso a la información es más lenta y que aumenta la complejidad de diseño de la estructura de información almacenada en la base de datos.

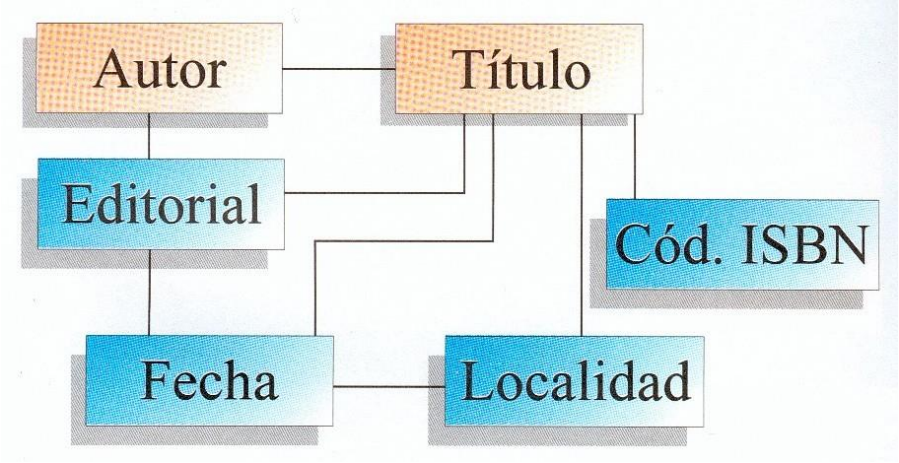


Figura 3: Bases De Datos en Red

1.1.7.3 BASES DE DATOS DOCUMENTALES

Las bases de datos documentales es otro tipo especial de bases de datos que almacenan información en forma de texto.
La estructura lógica de ese tipo de información es muy complicada de diseñar, puesto que los diferentes documentos contenidos en la base de datos están almacenados en registros de longitud variable.
Los accesos a la información también presentan problemas de diseño y programación, puesto que los documentos han de tratarse como cadenas de caracteres, debiendo buscarse el término deseado a través de todo el texto almacenado.
Para agilizar y mejorar el proceso de búsqueda a lo largo de los diferentes textos que componen la base de datos se deben seguir diversas estrategias (índices, búsquedas

complejas, etc.) que provocan una gran complicación al programar la recuperación de la información deseada por los usuarios.

1.1.7.4 BASES DE DATOS SEMÁNTICAS Y ORIENTADAS A OBJETOS

No existe una caracterización universal aceptada del término "orientación a objetos", por lo que cualquier intento de definir el concepto es necesariamente una visión particular. Se va a considerar que son tres las características esenciales que identifican este concepto.

- ✓ **Tipo Abstracto de Dato (TAD):** Es un modelo compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

  El estado de un TAD viene dado por la secuencia de operaciones realizadas sobre él. La definición de las operaciones suele darse mediante axiomas y reglas lógicas.
- ✓ **Herencia:** Es un tipo de jerarquía de abstracciones en la que existen una serie de subclases (abstracciones especializadas) que van a heredar características de una o varias superclases (abstracciones generalizadas).

  El objetivo fundamental de la herencia es, como el de la POO en general, la economía de expresión (reutilización).

  La herencia define una estructura en forma de árbol. En muchos lenguajes existe un nodo raíz común.
- ✓ **Identidad de objetos:** Un sistema de BDOO provee una identidad única a cada objeto independiente almacenado en la base de datos La estructura orientada a objetos impone automáticamente las restricciones relacionales: dominio, llave de integridad de identidad, identidad referencial Propiedades OID Es generado por el sistema

  Propiedades OID. Su valor no es visible para el usuario externo, sino usado por el sistema para identificar el objeto y crear y manejar las referencias entre objetos Es inmutable así preserva la identidad Es preferible que solo se use una vez, aunque se elimine el objeto de la BD La inmutabilidad y el preservar la identidad implican que el OID no dependa de atributo alguno del objeto

1.1.8 ARQUITECTURA DE BASE DE DATOS RELACIONALES

Hay tres características importantes inherentes a los sistemas de bases de datos:

- ✓ La separación entre los programas de aplicación y los datos,

- ✓ El manejo de múltiples vistas por parte de los usuarios
- ✓ El uso de un catálogo para almacenar el esquema de la base de datos.

El comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

La definición de un sistema de información es la descripción detallada de la arquitectura del sistema. Las arquitecturas de bases de datos han evolucionado mucho desde sus comienzos, aunque la considerada estándar hoy en día es la descrita por el comité ANSI/X3/SPARC. Este comité propuso una arquitectura general para DBMS basada en tres niveles o esquemas: el nivel físico, o de máquina, el nivel externo, o de usuario, y el nivel conceptual. Así mismo describió las interacciones entre estos tres niveles y todos los elementos que conforman cada uno de ellos.

1.1.9.1 ARQUITECTURA ANSI

La arquitectura de sistemas de bases de datos de tres esquemas fue aprobado por la ANSI-SPARC para conseguir la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos.

- ✓ **Nivel interno:** Tiene un esquema interno que describe la estructura física de almacenamiento de base de datos. Emplea un modelo físico de datos y los únicos datos que existen están realmente en este nivel.
- ✓ **Nivel conceptual:** Tiene esquema conceptual. Describe la estructura de toda la base de datos para una comunidad de usuarios. Oculta los detalles físicos de almacenamiento y trabaja con elementos lógicos como entidades, atributos y relaciones.
- ✓ **Nivel externo o de vistas:** tiene varios esquemas externos o vistas de usuario. Cada esquema describe la visión que tiene de la base de datos a un grupo de usuarios, ocultando el resto.

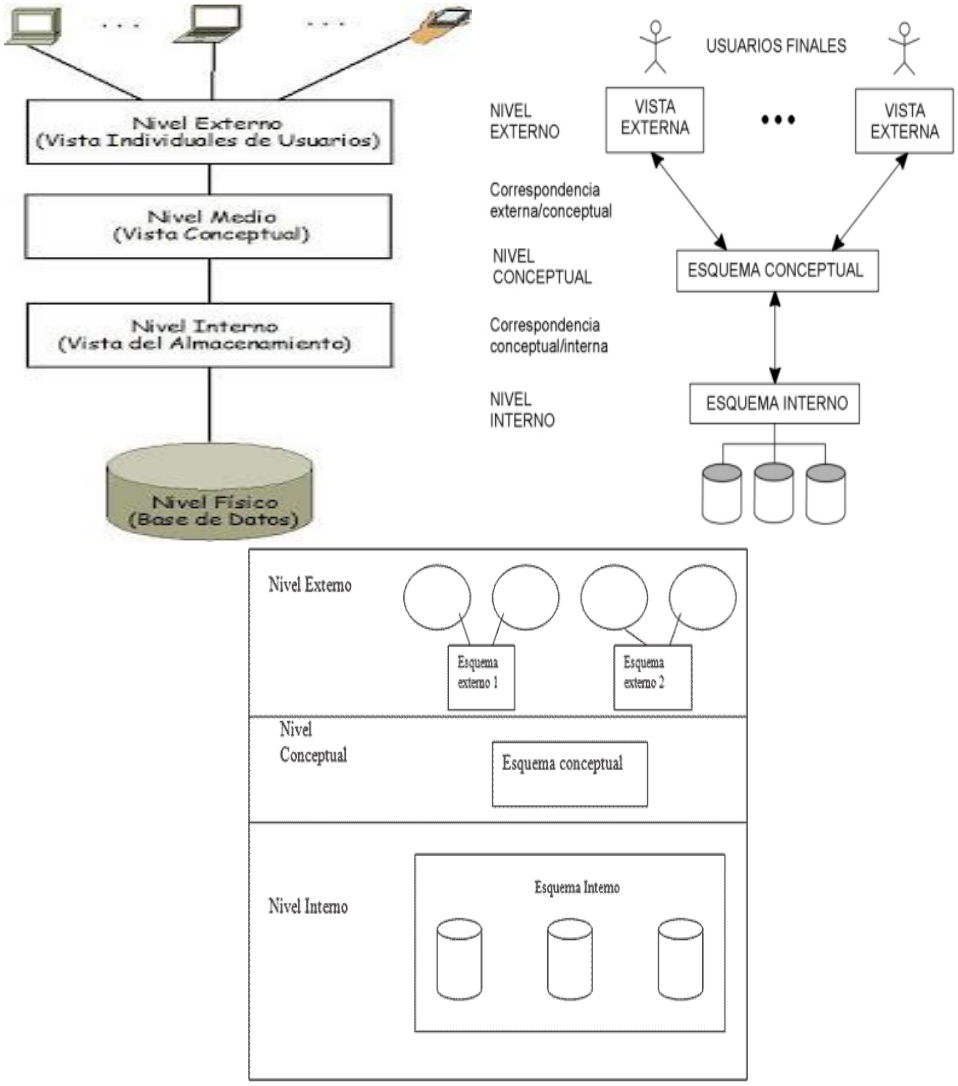


Figura 4: Niveles de Arquitectura de Base de Datos Relacionales

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física.

La mayoría de los SGBD no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. En casi todos los SGBD que se manejan vistas de usuario, los esquemas externos se especifican con el mismo modelo de datos que describe la información a nivel conceptual, aunque en algunos se pueden utilizar diferentes modelos de datos en los niveles conceptuales y externo.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar

cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada. Si la petición es de una obtención (consulta) de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada, para que coincida con la vista externa del usuario. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación. Estas correspondencias pueden requerir bastante tiempo, por lo que algunos SGBD no cuentan con vistas externas.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior.

Se pueden definir dos tipos de independencia de datos:

- ✓ La independencia lógica es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.
- ✓ La independencia física es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

En los SGBD que tienen la arquitectura de varios niveles es necesario ampliar el catálogo o diccionario, de modo que incluya información sobre cómo establecer la correspondencia entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el catálogo. La independencia de datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios, sólo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Por lo tanto, la arquitectura de tres niveles puede facilitar la obtención de la verdadera independencia de datos, tanto física como lógica. Sin embargo, los dos niveles de correspondencia implican un gasto extra durante la ejecución de una consulta o de un programa, lo cual reduce la eficiencia del SGBD. Es por esto que muy pocos SGBD han implementado esta arquitectura completa.

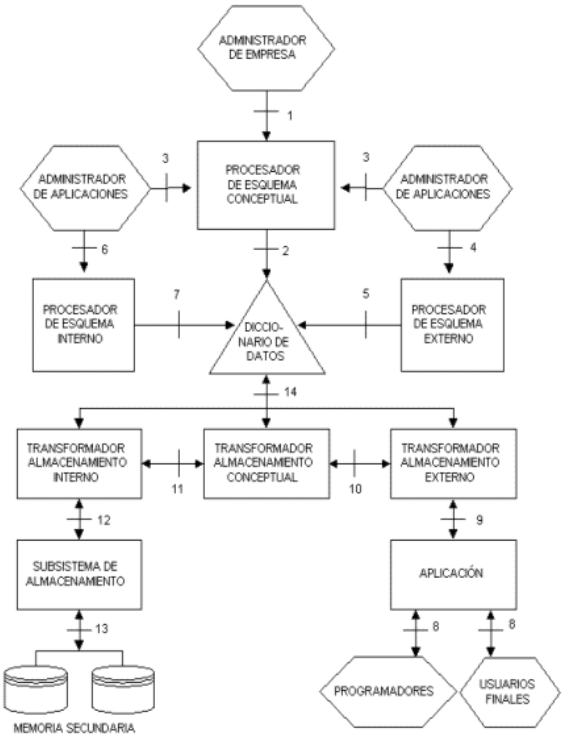


Figura 5: Arquitectura funcional ANSI/X3/SPARC

En esta arquitectura el nivel clave, es el conceptual. Éste contiene la descripción de las entidades, relaciones y propiedades de interés que constituye una plataforma estable desde la cual proyecta los distintos esquemas externos, que describen los datos según los programadores, sobre el esquema interno, que describe los datos según el sistema físico. Las posibles proyecciones de datos quedan resumidas en la gráfica:

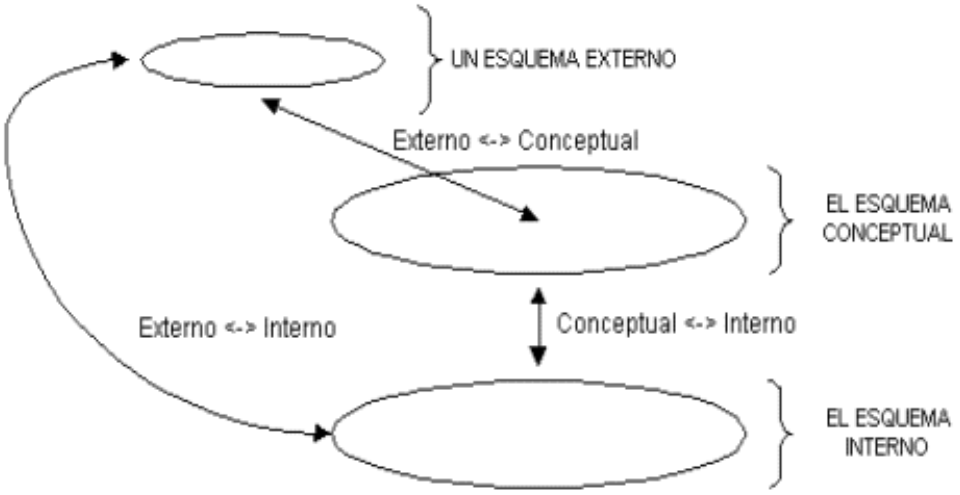


Figura 6: Nivel conceptual de arquitectura relacional

Por lo general, el nivel conceptual se obvia en los productos comerciales, salvo honrosas excepciones. Lo habitual es que el DBA realice el modelado conceptual usando sus propios recursos, o tal vez asistido por alguna aplicación de análisis, ya sea general o específica. El procesador del esquema conceptual, es por tanto el propio DBA. Los DBMS sí suelen ofrecer facilidades para la creación de esquemas externos, pero sin pasar por el nivel conceptual. Por supuesto, un DBMS comercial no está obligado a seguir las recomendaciones de estandarización de arquitecturas de ANSI/X3/SPARC. Por lo que respecta al modelo relacional de bases de datos, que ya existía antes del informe de este comité, los fabricantes de RDBMS se ajustan en mayor o menor medida al modelo teórico y, en cuanto a la arquitectura, han intentado seguir las recomendaciones del grupo RDBTG (Relational Data Base Task Group), parte del comité ANSI/X3/SPARC.

El resultado de este grupo fue restar importancia a las arquitecturas y realzar la de los lenguajes e interfaces. Como consecuencia, el lenguaje SQL, está hoy en día totalmente estandarizado, y en cambio encontramos distintas arquitecturas de RDBMS. Sin embargo se pueden distinguir dos tipos generales de arquitecturas para estos sistemas de bases de datos.

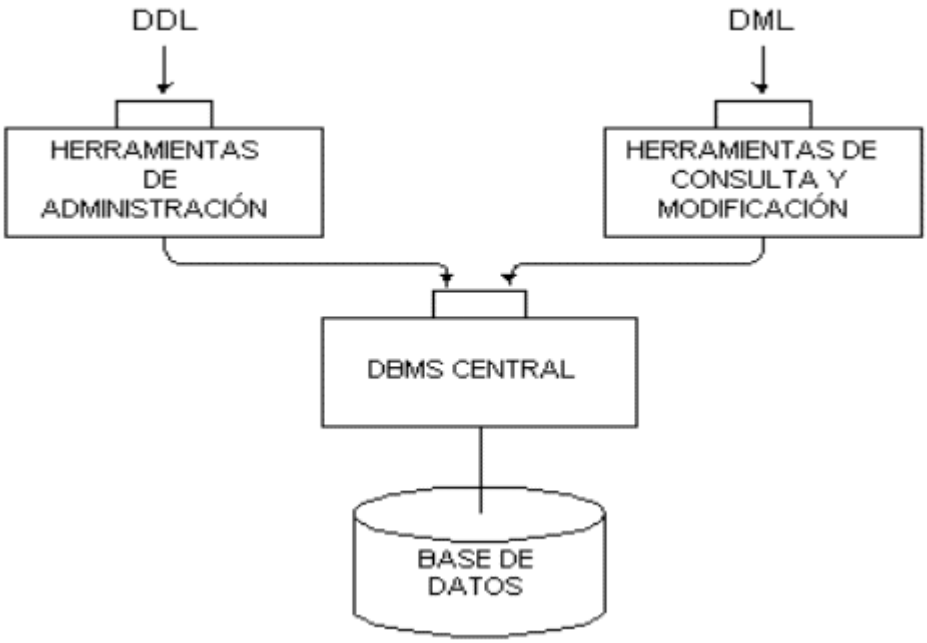


Figura 7: Arquitectura separada de RDBMS

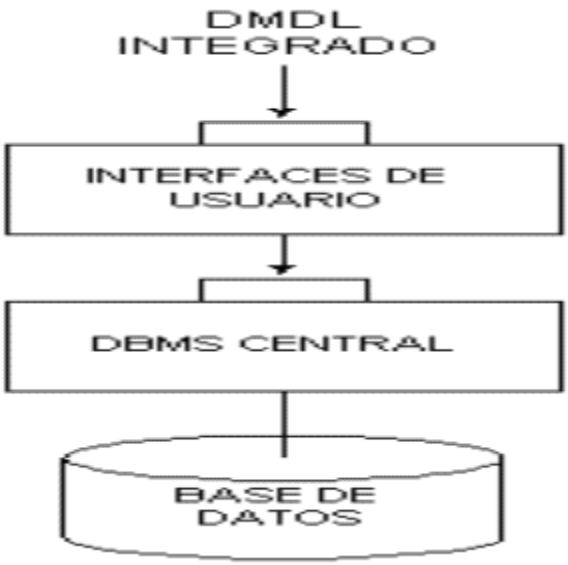


Figura 8: Arquitectura integrada de RDBMS

El tipo de arquitectura integrada es en general preferible a la arquitectura separada y el más común entre los RDBMS comerciales. De todos modos, la consecuencia de una integración de los lenguajes de definición de datos (DDL) y los de manipulación de datos (DML) en un sólo lenguaje (DMDL: Data Manipulation and Description Language), son a nuestro parecer positivas y negativas. Por un lado, esta integración resulta muy cómoda para el DBA, puesto que le basta con aprender un solo lenguaje formal para realizar todas las tareas de creación y mantenimiento de la base de datos. Pero por otro lado, estos sistemas (tanto los separados como los uniformes)

fuerzan una proyección directa desde el nivel externo al interno, haciendo que el nivel conceptual, el fundamental según la arquitectura ANSI/X3/SPARC, desaparezca o se implemente en el nivel externo como una vista global externa. Por esta razón algunos DBA inexpertos tienden a obviar la fase de análisis, cuando de hecho es la vital para la correcta implementación de la base de datos. Insistimos en que un buen modelado conceptual es una condición indispensable para el correcto desarrollo de una base de datos. Pensamos que lo ideal es usar un DBMS que nos permita desarrollar todas las tareas lo más fácilmente posible, pero no sin antes disponer de todas las herramientas necesarias para un correcto modelado conceptual, estén o no incluidas en el DBMS.

1.2 BASES DE DATOS NO RELACIONALES

Las bases de datos no-relacionales son comúnmente llamadas bases de datos No SQL ya que la gran mayoría de ellas comparte el hecho de no utilizar el lenguaje SQL para realizar las consultas. Aunque es una definición controvertida, ya que define estas bases de datos justamente por lo que no hacen, en vez de por lo que hacen, la realidad es que la mayoría de estas bases de datos comparten muy pocas cosas a nivel de arquitectura o diseño.

Las bases de datos No SQL hacen referencia al "almacenamiento estructurado" y este término posiblemente sea más correcto que No SQL, sin embargo, una de las características de las bases de datos no relacionales es que la mayoría de ellas no utilizan esquemas de datos rígidos como las bases de datos relacionales. Esto hace que estas bases de datos también se les llame "*Schema-less*" o "*Schema-free*", lo cual haría más acertado un nombre como "almacenamiento des-estructurado".

Las bases de datos No SQL han surgido en parte por las distintas necesidades de varias empresas que encontraron que la solución de base de datos SQL tradicional no encajaba con las necesidades específicas que tienen. Debido a que las bases de datos relacionales se han convertido en la solución para cualquier necesidad de base de datos, un producto que intenta cubrir todas las necesidades de todos los clientes acabará sin poder cubrir tan bien las necesidades muy específicas de algunos.

Las bases de datos relacionales se utilizan actualmente para casi cualquier problema que necesite un almacenamiento de datos, y aunque son una herramienta muy útil para muchísimos problemas, hay algunas problemáticas para las que no son la solución más idónea, sin embargo, debido a la popularidad de las bases de datos relacionales y la

inercia que se ha creado con ellas como tecnología, se intenta a menudo configurar y estirar las bases de datos al límite de sus posibilidades, cuando otras soluciones, diseñadas desde cero para solucionar el problema específico podrían servir mejor.

1.2.1 DEFINICIÓN

No SQL es, literalmente, una combinación de dos palabras: No y SQL. No SQL quiere decir no RDBMS o No Relacional, a su debido tiempo, algunos han propuesto NonRel como una alternativa a No SQL. Algunos otros han tratado de rescatar el término original, proponiendo que en realidad es No SQL un acrónimo que se expande a "Not Only SQL." Cualquiera que sea el significado literal, No SQL se utiliza hoy en día como un término general para las bases de datos y almacenes de datos que no siguen los principios de RDBMS y con frecuencia se relacionan con grandes conjuntos de datos accedido y manipulado en un sitio Web de gran escala. No SQL no es un producto único o incluso una sola tecnología, representa una clase de productos y una colección de diversos conceptos acerca de almacenamiento y manipulación de grandes cantidades de datos.

1.2.2 CONTEXTO Y UN POCO DE HISTORIA

Antes de empezar con los detalles sobre los tipos No SQL y los conceptos involucrados, es importante establecer el contexto en el que surgió No SQL. Las Bases de Datos No Relacionales no son nuevas, de hecho, las primeras tiendas de bases de datos no relacionales aparecieron cuando se inventaron las computadoras.

Sin embargo, las bases de datos no relacionales son una nueva encarnación, que nacieron en el mundo de forma masiva como aplicaciones escalables de Internet.

Comenzando con Inktomi, considerado como el primer motor de búsqueda verdadero, y culminando con Google, está claro que la gestión de base de datos relacional ampliamente adoptado sistema (RDBMS) tiene su propio conjunto de problemas cuando se aplican a cantidades masivas de datos.

Google, a lo largo de los últimos años, construyó una infraestructura escalable masivamente por su motor de búsqueda y otras aplicaciones, como Google Maps, Google Earth, Gmail, Google Finance y Google Apps. El objetivo era construir una infraestructura escalable para el procesamiento paralelo de grandes cantidades de datos.

La publicación de los documentos de Google para el público estimuló un gran interés entre los desarrolladores de código abierto. Los creadores del motor de búsqueda de

código abierto Lucene, fueron los primeros en desarrollar una primera versión de fuente abierta que replican algunas de las características de la infraestructura de Google.

Esta alternativa de código abierto es Hadoop, sus sub-proyectos, y los proyectos conexos. Sin entrar en la línea de tiempo exacto del desarrollo de Hadoop, en algún lugar hacia el primero de sus comunicados y surgió la idea de No SQL. Es importante tener en cuenta que la aparición de Hadoop sentó las bases para el crecimiento rápido de No SQL. Además, es importante tener en cuenta que el éxito de Google ayudó a impulsar una adopción saludable de los conceptos de la nueva era de computación distribuida, el proyecto Hadoop y No SQL.

Un año después de que los papeles de Google han catalizado el interés en el procesamiento paralelo escalable y de almacenes de datos distribuidos no relacionales, Amazon decidió compartir algo de su propia historia de éxito. En el 2007, Amazon presentó sus ideas de un conjunto de datos distribuidos altamente disponibles y consistentes su producto fue Dynamo.

Con el respaldo No SQL de dos gigantes de Internet más importantes Google y Amazon, surgieron nuevos productos en este espacio. Una gran cantidad de desarrolladores comenzaron a jugar con la idea de utilizar estos métodos en sus aplicaciones y muchas empresas, desde startups hasta grandes corporaciones, se convirtió en susceptibles de aprender más acerca de la tecnología y, posiblemente, el uso de estos métodos. En menos de 5 años, las ideas No SQL y afines para la gestión de grandes volúmenes de datos se han convertido en casos generalizados y gracias a su uso han surgido muchas empresas conocidas, como Facebook, ix Netfl, Yahoo, eBay, Hulu, IBM, y muchos más.

Muchas de estas compañías también han contribuido a abrir la fuente de sus extensiones y nuevos productos al mundo.

1.2.3 SIN ESQUEMA DE DATOS No SQL

La mayoría de las aplicaciones evolucionan con el tiempo. Las distintas necesidades de negocio hacen que se guarden datos distintos, o que haga falta guardar datos adicionales. Los cambios en los esquemas de bases de datos relacionales son procesos extremadamente complicados. En una base de datos tradicional, añadir una columna requiere añadir ese campo a todas las filas existentes, que fácilmente pueden ser cientos de millones de filas. Este proceso puede dejar inoperativa a la base de datos

durante un tiempo considerable, y en algunas bases de datos hasta necesita hacerse cuando la base de datos está apagada.

Hay algunos negocios que pueden permitirse estar inoperativos durante una hora en ciertos horarios, pero hay otros que funcionan 24 horas al día. Algunos ejemplos son los grandes portales web como Facebook, Twitter, o Google, que no se pueden permitir ningún Down time.

Estos problemas para evolucionar el esquema de datos causan una ralentización del desarrollo del sistema, ya que para poder aplicar cambios al esquema de datos, se debe seguir un procedimiento muy trabajoso y en cierta medida hasta peligroso para la empresa. Esta ralentización en el desarrollo es crucial, ya que las empresas deben estar en continua innovación y desarrollo para poder seguir siendo competitivos, y que el esquema de la base de datos sea algo que impida el avance tecnológico es algo que típicamente no se considera razonable.

Muchas bases de datos no relacionales tienen un esquema de datos pre-definido. Cada registro de la tabla puede tener campos distintos, la base de datos no limita ni controla cuantos campos ni de qué tipo se pueden guardar en cada registro.

> **Por ejemplo:** en una base datos no relacionales se guardan registros de tipo "persona". Estos registros tienen los campos "nombre" y "apellido". Después de varios millones de registros, se puede añadir el campo "teléfono" a cualquier registro, sin necesidad de variar el esquema de la base de datos, ya que este no tiene esquema. Algunos registros tendrán el campo "móvil" y otros no.
>
> Sin embargo, la aplicación debe poder manejar todos los casos posibles al recuperar un registro de tipo persona. Habrá casos en los que un registro de tipo persona no tiene el campo móvil, y casos donde si lo tendrá. Es la aplicación la que tiene que tomar las decisiones correctas en cada caso, aunque cause un trabajo extra al programador.

Otra ventaja de tener un almacén de datos sin esquema, es cuando la aplicación que se va a programar no tiene un modelo de datos de fácil normalización. Aunque en general muchas tipologías de datos se pueden normalizar, no todos los datos son de tipo tabular. Esto a veces crea casos donde se intenta a la fuerza normalizar datos que por naturaleza no son de tipo tabular, y se acaban con esquemas de datos grandes y muy complicados cuando se podría haber llegado a una solución mucho más elegante y sencilla utilizando un tipo de almacén de datos más apropiado para el tipo de información que se está almacenando.

Este es un problema mucho más importante de lo que pueda parecer a primera vista. Aunque es muy común organizar datos de forma tabular, hay muchos casos donde esta no es la mejor forma de representar la información.

La normalización excesiva puede fragmentar la información en muchas tablas. Cada tabla requiere una consulta separada para poder acceder a los datos en cuestión. Esto al final significa muchas consultas para retornar una información que de otra manera hubiese sido fácil de representar en algún otro esquema de almacenamiento de información. Las intersecciones entre varias tablas son procedimientos costosos donde se multiplican las filas de cada tabla entre si y posteriormente se seleccionan las filas que cumplan las restricciones. Esto significa que hacer un JOIN sobre 4 tablas de 1,000 filas cada una, genera una tabla con 1,000,000,000 filas y por tanto ralentiza la ejecución de la consulta.

Como ejemplo: se propone una sencilla libreta de direcciones de un sistema de procesamiento de pedidos online. Después de almacenar los campos que siempre estarán rellenos ya que son únicos a la persona (nombre y apellidos) se pueden almacenar los datos de direcciones de entrega, teléfonos y emails. Estos tres campos necesitaran tres tablas separadas, ya que una persona puede tener varias direcciones, teléfonos y emails. Si se siguen añadiendo campos como información de tarjetas de crédito usadas, y otros datos relacionados al usuario, correctamente normalizado se verá que este esquema de datos tiene múltiples tablas, y sin embargo, la gran mayoría de las veces toda esta información será consultada a la vez. Muy pocas veces se iterara por la lista de teléfonos o direcciones, sin embargo cuando accede un cliente a su área personal es muy habitual mostrarles todos los datos que tiene guardados en el sistema.

Con una base de datos relacional, ese acceso del cliente sería muy costoso, ya que requeriría varias consultas, una por cada tabla, y bastante procesamiento para juntar todos los datos y mostrarlos.

Con una base de datos no relacional, todos los campos podrían ser parte del registro tipo cliente, con lo que con un único acceso, y una única lectura a la base de datos, se podría obtener toda la información relacionada con el cliente. Esto es similar a la forma que un documento XML guarda información de manera desestructurada ya que identifica cada campo por su nombre.

Twitter utiliza una base de datos de grafo llamada FlockDB para modelar las relaciones entre sus usuarios. Este tipo de bases de datos No SQL son perfectas para representar

relaciones entre objetos, sin embargo, no son ideales para representar muchos otros tipos de datos.

No todo son ventajas de tener una base de datos sin esquema. Aunque la base de datos no tenga esquema, eso no significa que el programador no tiene que pensar en el modelo de datos. Al contrario, ya que no tiene la herramienta familiar de la normalización, debe tener mucho cuidado y dar mucho pensamiento a cómo organizar los datos de la aplicación.

Utilizando la distintas formas normales, hay un procedimiento fácil y establecido para hacer que los datos de la aplicación se ajusten al modelo, Sin embargo, crear un nuevo modelo que se ajuste a la aplicación, aunque beneficioso en muchos términos es mucho más trabajoso y requiere un trabajo más creativo. Los errores en ese estado de la programación también son más costosos, aunque luego sea más fácil modificar el esquema de datos, cambios a gran escala nunca son fáciles.

Como toda herramienta potente, necesita trabajo para poder ser utilizado correctamente. Hay casos en los que no vale la pena el trabajo de diseñar el propio modelado de datos, cuando una solución en una base de datos relacional SQL es más accesible y más fácil de modelar.

Las bases de datos No Relacionales requieren de un almacenamiento masivo de la información ya que este tipo de base de datos está diseñada para guardar enormes cantidades de datos.

1.2.4 ALMACENAMIENTO MASIVO DE LA INFORMACIÓN Y DATAMINIG

Es extremadamente complejo repartir los datos de una base de datos relacional entre varias máquinas distintas, en gran parte debido a las relaciones entre las distintas tablas de la base de datos. Los sistemas No SQL almacenan grandes cantidades de información, y abstraen la mayoría de las complejidades asociadas con repartir los datos de la aplicación entre varias máquinas, y automatizan de gran forma el añadir o restar maquinas al clúster.

El esquema de datos tabular no es siempre el mejor sistema para guardar cantidades masivas de datos, o datos a los que siguientemente se les va a hacer un tratamiento de Business Intelligence o DataMining. Se han desarrollado varios tipos de esquemas de datos más optimizados para su posterior proceso o agrupación. El modelo Big Table de Google por ejemplo, guarda los datos en columnas en vez de filas, por ello se le llama

base de datos columnas. Esto hace que ciertos procesos como totalizar columnas o hacer operaciones de agrupamiento sean mucho más rápidas que con bases de datos organizadas por filas.

1.2.5 VENTAJAS Y DESVENTAJAS DE BASE DE DATOS No RELACIONALES

1.2.5.1 VENTAJAS

- ✓ Es de código abierto
- ✓ Escalamiento sencillo
- ✓ Diferentes DBs No SQL para diferentes proyectos
- ✓ Las bases de datos No SQL utilizan sobre todo el uso de memoria en vez del disco como la principal ubicación de escritura
- ✓ Ausencia de esquema en los registros de datos
- ✓ Simplicidad al momento de su instalación
- ✓ Son de código libre
- ✓ Pueden manejar enormes cantidades de datos
- ✓ Se ejecutan en clusters de máquinas baratas

1.2.5.2 DESVENTAJAS

- ✓ El código abierto puede significar una "mancha" en el soporte para las empresas
- ✓ No están lo suficientemente maduros para algunas empresas
- ✓ Limitaciones de Inteligencia de Negocios
- ✓ La falta de experiencia

1.2.6 TIPOS DE BASES DE DATOS No SQL

Aunque sean muy difíciles de clasificar debido a las diferencias entre soluciones, hay algunas clasificaciones generales que se pueden hacer a las bases de datos no relacionales. Es posible que productos específicos tengan más de una clasificación, pero en general, estos son los distintos tipos de bases de datos No SQL que existen:

1.2.6.1 ALMACENAMIENTO CLAVE - VALOR

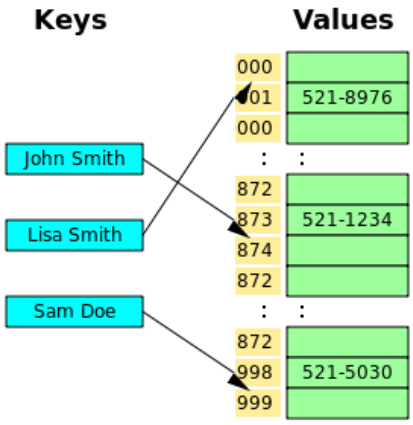


Figura 9: Almacenamiento Clave - Valor

Los almacenamientos clave-valor, o key-value, son uno de los tipos más simples de base de datos no relacional. Los caches de memoria típicamente utilizan el esquema key-valuepara guardar la información en cache. Los almacenamientos key-valueasocian una clave única (key) al valor que se quiere guardar (value).

Varias implementaciones de los almacenamientos key-valuetienen funcionalidad adicional, pero a un nivel básico, el key-valuesolo requiere una clave y un valor.

Este tipo de base de datos suele ser extremadamente rápido y optimizado para una gran cantidad de accesos al mismo tiempo.

Algunos productos que usan base de datos orientadas a almacenamiento clave-valor son:

- ✓ **Amazon SimpleDB:** No es Open Source pero puede usarse gratis siendo usuario en Amazon.com
- ✓ **Azure Table Storage:**Es una colección de entidades de forma libre.
- ✓ **Riak Basho:**Escrita en erlang, es eventualmente consistente.
- ✓ **Tokyo Cabinet:**Es una base de datos que almacena par de valores en árboles y arrays fijos, el principal usuario de esta base de datos es: Scribd.com

1.2.6.2 BASES DE DATOS DE GRAFO

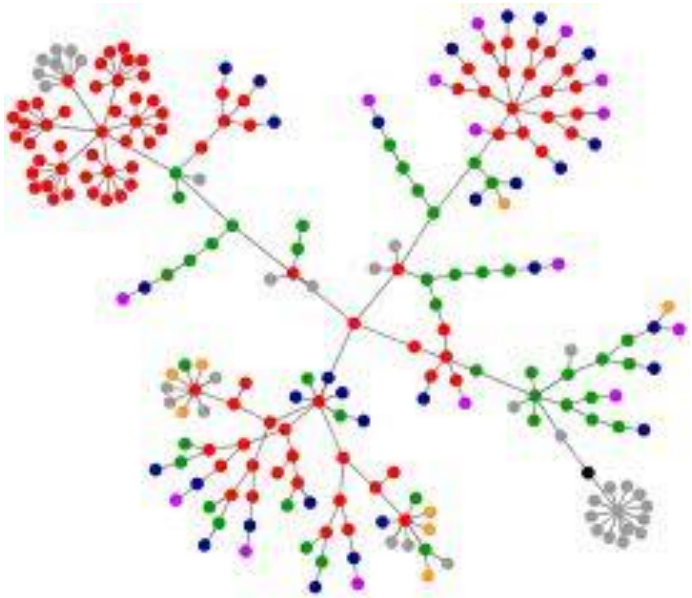


Figura 10: Base de datos de Grafo

Las bases de datos de grafo organizan la información en grafos dirigidos. Están optimizadas para hacer operaciones de consulta de relaciones entre miembros, y para esta tarea en específico son extremadamente rápidas.

Las bases de datos No SQL basadas en grafos están inspiradas en Euler y fundamentadas en la teoría de grafos. La información se guarda en estructuras de nodos, arcos (conexiones) y propiedades (de arcos y nodos).Están orientados a manejos de datos muy complejos y con muchas relaciones entre sí.

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.

Algunos ejemplos de base de datos de grafo son:

- ✓ **VertexBD:** Escrita en Lenguaje C, es una base de datos Open Source.
- ✓ **Neo4j:** Es escrita en Java, posee características ACID.
- ✓ **HyperGraphDB:** Es escrita en Java, es Open Source, es especial para Inteligencia artificial y Web Semántica.

1.2.6.3 BASE DE DATOS ORIENTADA A ALMACENAMIENTO DE DOCUMENTOS

La información se almacena en objetos xml, json, etc. Renuncian a la estructura fija de datos usual en RDBMS. Almacenan información semi-estructurada. Documentos con diferentes estructuras pueden ser adicionados sin afectar la estructura de los documentos ya existentes.

Los almacenes de documentos guardan la información como un listado de documentos desestructurados. Al acceder a un documento, se puede acceder a un número no especificado de campos con sus respectivos valores. Son muy rápidos para recuperar toda la información asociada al documento junto, y tienen un esquema de datos muy flexible. Sin embargo, suelen ser lentos para hacer consultas donde se buscan todos los documentos con un determinado campo, ya que estos no suelen tener índices.

Las bases de datos orientadas a almacenamiento de documentos se destacan por:

- ✓ Modelado de datos natural
- ✓ Fácilmente programables
- ✓ Desarrollo rápido
- ✓ Orientadas a la web

Algunos productos que usan base de datos orientadas a almacenamiento de documentos son:

- ✓ **CouchDB:** Escrita en erlang, sus documentos son .JSON
- ✓ **MongoDB:** escrita en Java, sus documentos son JSON y BSON
- ✓ **TerraStore:** Escrita en Java, tiene soporte para uno o múltiples clústers.
- ✓ **ThruDB:** Escrita en C++ y Java, utiliza Thrift para integrar bases de datos.
- ✓ **RavenDB:** Escrita en .NET, tiene características de una base de datos Par-Valor.
- ✓ **Persevere:**Escrita en Java, Soporta Comet para que los clientes actualicen su sistema por medio de eventos.

1.2.6.4 BASES DE DATOS COLUMNARES

Tabla 1: Base de Datos Columnares

Title	Title	Title	Title	Title
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data
Data	Data	Data	Data	Data

Las bases de datos columnares suelen estar optimizados para guardar grandes cantidades de datos para consultas de agregación y reporte. Estas bases de datos suelen ser muy rápidas en consultas de agregación de datos o data mining, sin embargo, no suelen ser usadas en entornos on-line donde la latencia y tiempo de respuesta de las consultas suele ser crítico.

Las bases de datos orientadas a columnas son aquellas cuyo precursor es Google BigTAble. Su almacenamiento básico es la columna: nombre, valor. Una base de datos orientada a columnas es una clave seguida de un número variable de columnas. Su equivalente relacional sería una tabla.

Las bases de datos orientadas a columnas se destacan por:

- ✓ Una buena gestión de tamaño
- ✓ Cargas de escrituras masivas orientadas al stream
- ✓ Alta disponibilidad
- ✓ MapReduce

Algunos productos que usan base de datos orientadas a columnas son:

- ✓ **Apache Hadoop, H Java, any writer Base:**escrita en Java, tiene 101 usuarios famosos entre los que destacan Google, IBM, Facebook y Twitter.
- ✓ **Apache Cassandra:** Escrita en Java, los usuarios más famosos son: Facebook y Twitter.
- ✓ **Hypertable:** Escrita en C++, el usuario principal es Baidú un buscador chino.
- ✓ **SeiDB:** Escrita en C++, es orientado a arreglos (Arrays),
- ✓ **Open Neptune:** Escrita en Java y Lenguaje C, es una implementación de Big Table de Google.

- ✓ **KDI:** Escrita en C, C++ y Python, es una implementación de Big Table de Google. Trabaja sobre KFS (Kosmos File System, un sistema de archivos distribuido).

Antes de pasar al análisis de bases de datos No SQL es importante establecer el cuadro de usabilidad de bases de datos no relacionales para basarnos en esos datos y decidirnos por cuál base de datos No Relacional implementar en la investigación, Los cuadros son tomados del sitio web addkw.com y google trends, donde indica que en los dos últimos años el incremento de la demanda de los empleadores para especialistas en tecnología No SQL ha sido extraordinario.

- ✓ Fuente: addkw.com

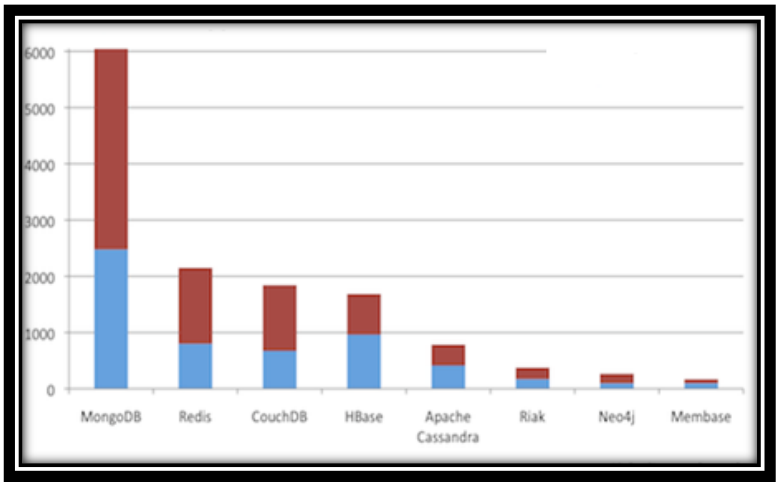


Figura 11: Nivel de Aceptabilidad de bases de datos No SQL

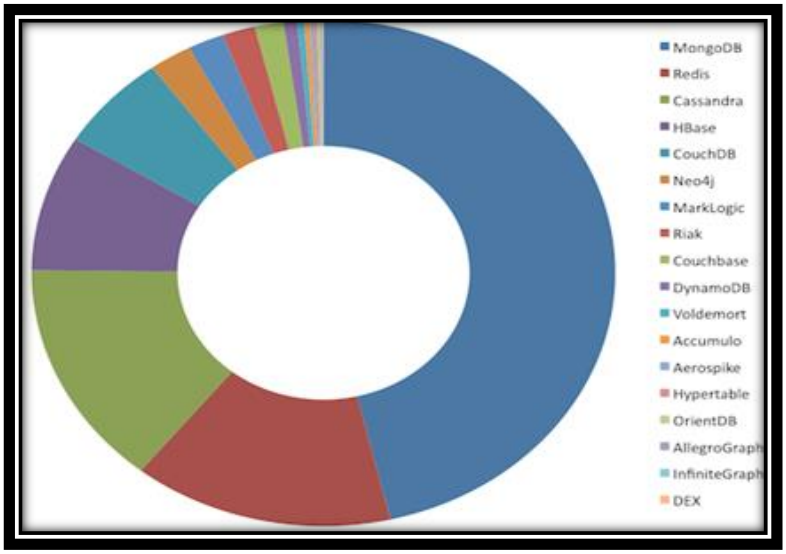


Figura 12: Bases de datos más usadas en los últimos dos años

✓ Fuente: Google Trends

Se debe tener en cuenta que el valor de 100 representa el interés máximo de búsquedas.

✓ Año 2011

Tabla 2: Tendencias de Búsquedas Base de Datos No SQL año 2011

BASE DE DATOS	TENDENCIA DE CRECIMIENTO											
	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
MongoDB	37	38	47	48	51	57	54	59	59	59	61	61
CouchDB	17	19	18	19	18	17	18	17	15	17	17	15
Riak	5	5	7	6	5	7	6	6	7	7	7	7
Redis	24	28	28	28	28	31	32	33	35	34	37	35
Neo4j	2	3	4	5	5	5	5	5	5	6	6	6

✓ Año 2012

Tabla 3: Tendencias de Búsquedas Base de Datos No SQL año 2012

BASE DE DATOS	TENDENCIA DE CRECIMIENTO											
	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
MongoDB	63	72	72	76	79	77	77	83	82	82	84	79
CouchDB	16	14	14	16	15	14	14	13	12	13	11	11
Riak	7	8	8	9	9	8	10	9	8	9	8	7
Redis	34	41	41	43	43	42	42	40	42	41	40	37
Neo4j	6	7	7	7	7	8	7	7	8	8	8	7

✓ Año 2013

Tabla 4: Tendencias de Búsquedas Base de Datos No SQL año 2013

BASE DE DATOS	TENDENCIA DE CRECIMIENTO				
	ENERO	FEBRERO	MARZO	ABRIL	MAYO
MongoDB	86	91	100	98	99
CouchDB	11	11	12	12	11
Riak	8	9	9	9	9
Redis	40	42	45	46	47
Neo4j	9	10	9	10	10

En el caso de MongoDB en los dos últimos años se tiene una cifra extraordinaria de 4.200% entre el 2010 y 2013, por esa razón luego de haber analizado los gráficos se puede apreciar que la base de datos No Relacional más usada actualmente es MongoDB, y será la base de datos No Relacional que usaremos en la investigación.

1.2.7 ANALISIS DE BASES DE DATOS No SQL

Figura 13: Bases de datos No SQL

Mientras las bases de datos tradicionales se forjaron la mayoría como productos de código cerrado, y recientemente han aparecido soluciones de código abierto, las bases de datos No SQL han nacido de los movimientos Open Source, y las mejores y más desarrolladas en estos momentos son las de código abierto. Algunos ejemplos son:

1.2.7.1 CASSANDRA

Cassandra es una mezcla de almacenamiento clave-valor y base de datos columnar creada por Facebook y hecha de código abierto en el 2008. Está escrito en Java con lo que funciona sobre cualquier sistema operativo, y utiliza Thrift (otro

proyecto de Facebook) para serializar y comunicar los datos con programas externos. Esto permite usar Cassandra desde prácticamente cualquier lenguaje de programación. Está diseñada para gestionar cantidades muy grandes de datos distribuidos en una gran cantidad de máquinas comunes.

1.2.7.1.1 MODELO DE DATOS

Cassandra ha sido descrita varias veces por el director del equipo que la desarrollo como una mezcla de Big Table de Google y Dynamo de Amazon, en parte porque utiliza un modelo de datos columnar parecido al de Big Table, mientras que la infraestructura que permite añadir y restar nodos de forma automática y transparente es parecida a la de Dynamo. Uno de los creadores de Dynamo fue contratado por Facebook para desarrollar Cassandra.

Cassandra provee un servicio de clave valor estructurado, ya que cada clave puede ir asociado a valores múltiples (llamados columnas), que a su vez van asociados en familias de columnas. Aunque las familias de columnas son fijas, se pueden añadir y restar columnas a las claves en cualquier momento. Este modelo hibrido es como una mezcla del sistema columnar y el sistema basado por filas y es extremadamente poderoso, aunque a veces puede ser difícil de usar y entender.

Este sistema de claves es muy novedoso, y es una gran parte de la ventaja de trabajar con Cassandra, ya que da la rapidez de un sistema clave valor, pero con la información profunda que puede proveer un sistema columnar. De hecho Cassandra esta descrita como una base de datos OLTP (Online Transaction Processing) con lo que su función es servir datos cambiantes en tiempo real con gran rendimiento y escalabilidad.

El modelado de datos se usa para crear mapas con 4 o 5 dimensiones. Para crear un mapa con 4 dimensiones, considerando el modelo de datos como una pirámide y desde la punta, que llamaremos keyspace, que es donde se guardan todas las claves únicas, hasta el valor final de los datos a guardar en cada columna:

- ✓ Keyspace> column family
- ✓ Column family > column family row
- ✓ Column family row > columns
- ✓ Column > data value

Para un modelo de datos con 5 dimensiones, se añaden Súper Columnas. No está en comendado añadir Súper Columnas al menos que se tenga muy claro el objetivo, ya

que los grupos de Súper Columnas se tienen que guardar en el mismo nodo y por tanto pueden producir una ralentización en los procesamientos de los datos.

- ✓ Keyspace> super column family
- ✓ Super column family > super column family row
- ✓ Super column family row > super columns
- ✓ Súper column > columns
- ✓ Column > data value

Esta flexibilidad en el modelo de datos permite usar a Cassandra para hacer modelos complejos y a la vez muy ricos en información. Cassandra mantiene un tiempo de acceso muy rápido a los datos.

1.2.7.1.2 ESCALABILIDAD

Uno de los puntos fuertes de Cassandra es su escalabilidad. Mientras muchos otros proyectos necesitan código adicional para poder crecer en capacidad, Cassandra ha sido diseñado para ser totalmente descentralizado, y crecer elásticamente añadiendo nodos adicionales. Añadir un nodo a un clúster Cassandra es muy fácil, ya que se auto configura en la información de su nodo más próximo e inmediatamente se pone a copiar información de los demás nodos.

Es igual de sencillo retirar un nodo de un clúster Cassandra. Simplemente apagar o desconectarlo, y los demás nodos se repartirán la carga de las consultas.

La facilidad de Cassandra de crecer y reducir el tamaño del clúster elásticamente es una de los puntos más fuertes de esta base de datos No SQL.

1.2.7.1.3 CONSISTENCIA DE DATOS

Todos los nodos de Cassandra son idénticos, no hay punto único de fallo. La consistencia de los datos es un parámetro que se puede especificar en la configuración del clúster. Una consistencia de datos de 3 es un valor típico. Esto significa que cuando se envía un dato a un clúster Cassandra, no se devolverá un mensaje de haberse escrito correctamente hasta que el dato este replicado en tres máquinas. Esto asegura que siempre haya una copia disponible de los datos aunque se pierdan los datos de máquinas completas. El clúster de Cassandra mantiene automáticamente el nivel de consistencia requerido. Si se retira una máquina del clúster, los datos de los que solamente haya dos copias se distribuirán automáticamente por el clúster hasta que existan 3 copias de los datos.

Cassandra utiliza un modelo llamado “eventualmente consistente”. Esto quiere decir que no garantiza que el dato que se acaba de escribir sea inmediatamente disponible, sino que dado tiempo, estará disponible en todas las maquinas del clúster. Esto es debido a la forma en que se propagan los datos por el clúster de máquinas.

1.2.7.1.4 USUARIOS

Cassandra es utilizado por múltiples empresas importantes. Es un proyecto Apache de máximo nivel. Las empresas que se conoce que usan Cassandra actualmente son:

- ✓ Facebook con más de 200 nodos,
- ✓ Digg,
- ✓ Twitter,
- ✓ Rackspace,
- ✓ Ibm
- ✓ Reddit.

1.2.7.2 HBASE

HBase es una base de datos distribuida de código abierto no relacional. Está inspirada en el Big Table de Google, y es parte del proyecto Hadoop dela Fundación de Apache Software. Está escrito en Java, con lo que es portable a cualquier sistema operativo. Está diseñada para ser una base de datos OLAP (On Line Analitical Processing) con lo que está centrado en el análisis de grandes cantidades de datos, y no en procesar datos rápidamente en tiempo real.

HBase fue desarrollada por la empresa Powerset que fue posteriormente adquirida por Microsoft.

Al hablar de HBase hay que además hacer referencia a Hadoop y HDFS. HDFS es el sistema de archivos de HBase, y se utiliza para guardar los archivos de forma distribuida y redundante y al mismo tiempo tener un acceso rápido a ellos.

Hadoop es un framework para escribir aplicaciones de tratamiento de datos distribuidas, y utiliza a HBase como su base de datos. Hadoop permite hacer trabajos Map Reduce sobre un número ilimitados de nodos HBase, y por tanto hacer un tratamiento masivo de datos. Los tres (Hadoop, HBase y HDFS) son proyectos de código abierto de la Fundación de Apache.

1.2.7.2.1 MODELOS DE DATOS

HBase guarda los datos organizados en columnas al estilo del Big Table de Google.

HBase soporta una variedad muy grande de protocolos para comunicarse con las distintas aplicaciones. Tiene un interfaz Thrift a sus datos, haciéndolos accesibles

desde cualquier lenguaje de programación. También acepta enviar datos en XML, Proto-buffers y datos binarios.

1.2.7.2.2 ESCALABILIDAD

HBase es escalable a un número muy grande de nodos, sin embargo, HBase requiere de software adicional para poder gestionar las maquinas que se van añadiendo al clúster, ya que estas no se auto configuran ni se auto gestionan.

1.2.7.2.3 CONSISTENCIA DE DATOS

HBase no tiene un punto central de fallo, los datos están distribuidos en todos los nodos de la aplicación. Tiene parámetros configurables para regular el nivel de consistencia de los datos.

1.2.7.2.4 USUARIOS

Yahoo es el usuario más importante de HBase, con casi 250 nodos en su datacenter.

1.2.7.3 COUCHDB

CouchDB es una base de datos no relacional de documentos. Está escrita en Erlang, un lenguaje específicamente creado por Ericsson para programar sistemas robustos, distribuidos y multi hilo. CouchDB es un acrónimo para Clúster Of Unreliable Commodity Hardware, o "clúster no fiable de hardware común".

Inicialmente creada por el fundador de Couchio (empresa que da soporte a CouchDB), CouchDB es ahora un proyecto Apache y se está consiguiendo cierta popularidad.

CouchDB fue diseñado como "la base de datos de la web" y el acceso a todos los datos es a través del protocolo HTTP, con un interfaz que cumple el estándar REST (Representation al State Transfer, el estándar de acceso a documentos web).

Los datos se reciben codificados en JSON. Esta elección es algo curiosa, ya que HTTP no es un protocolo especialmente rápido. Sin embargo, es ubicuo, y por tanto es fácil acceder a los datos de CouchDB desde cualquier lenguaje de programación. De hecho, uno de los objetivos de CouchDB es que se accediese a los datos directamente desde JavaScript, obviando la necesidad de pasar a través de un servidor de aplicaciones. Igualmente dispone de librerías para varios lenguajes de programación que hacen de "wrapper" alrededor de la llamada HTTP para facilitar el trabajo al programador.

1.2.7.3.1 MODELO DE DATOS

El modelo de datos de CouchDB es basado en documentos y por tanto muy flexible. A diferencia de muchos otros proyectos No SQL, CouchDB cumple completamente con las propiedades ACID. De hecho, CouchDB va más allá, y mantiene un registro de versiones de cada uno de los documentos en su registro.

Esto permite reconstruir un documento en cualquier momento de su vida. Al nunca sobrescribir ningún dato, el sistema completo mantiene un control de versiones perfecto, parecido a un control de versiones de software como subversión o CVS.

Todas las escrituras son serializadas y las lecturas nunca son bloqueadas, con lo que las lecturas nunca esperaran por ningún concepto. Debido al sistema de escritura, nunca es necesario un chequeo de consistencia al iniciar la base de datos, de hecho, no tiene forma recomendada de terminar la ejecución, simplemente se termina el proceso, ya que nunca se corromperán los datos.

Para presentar los datos, CouchDB utiliza un modelo de presentación de datos llamado "Views". CouchDB no permite hacer consultas adhoc, todas las consultas que se van a requerir deben ser programadas como views. Sin embargo, esto permite que estas consultas se ejecuten rápidamente, ya que cada view es pre computado cuando es creado y luego incrementalmente actualizado cada vez que se hace las escrituras, asegurando un retorno muy rápido de las lecturas de datos.

Los views muestran los datos de distintas formas, seleccionados, agrupados, agregados o combinados. Se guardan en documentos especiales y se replican con el resto de la base de datos.

1.2.7.3.2 ESCALABILIDAD

CouchDB tiene uno de los sistemas más flexibles y configurables de escalabilidad de todos los sistemas revisados. CouchDB utiliza replicación maestro-maestro echa de forma peer to peer entre todos los servidores. Es una funcionalidad extremadamente avanzada que permite tener todas las instancias de CouchDB en sincronismo y leer y escribir desde cualquiera de ellas. Además, no tiene límites de latencia, con lo que una base de datos CouchDB puede estar desconectada del clúster, recibir varios cambios, y propagar estos cambios al clúster cuando se vuelve a conectar.

Utiliza un sistema de resolución de conflictos automático. Se pueden configurar políticas de resolución de conflictos automáticos o se pueden dejar para ser resuelto manualmente.

El sistema de replicación es incremental. Debido a la forma en que CouchDB guarda todas las versiones de cada uno de sus documentos, solamente se envían los cambios hechos a los documentos cuando se hace la replicación.

Debido a la facilidad de replicar y añadir nodos la escalabilidad de un sistema CouchDB es muy sencilla.

1.2.7.3.3 CONSISTENCIA DE DATOS

Los datos son totalmente consistentes debido a la adherencia a los estándares ACID y a la facilidad de crear réplicas de la base de datos que automáticamente pueden ser escritos y leídos.

1.2.7.3.4 USUARIOS

Entre las empresas que usan CouchDB se pueden mencionar:

- ✓ Ubuntu
- ✓ Credit Suisse
- ✓ Meebo

1.2.7.4 MEMCACHE

Memcache es el servidor cache más popular del mundo. Creado por DangaInteractive, posteriormente se lanzó como código abierto.

Su fuerza se basa en la extrema sencillez. Solo soporta guardar un objeto (que debe ser una cadena) o retirar un objeto. Es extremadamente rápido.

Típicamente memcache es usado como una capa entre la base de datos de la aplicación y la propia aplicación. Los resultados pedidos muy a menudo de la aplicación se guardan en memcache y son retirados reduciendo de esta manera la cantidad de impactos a la base de datos.

1.2.7.4.1 MODELO DE DATOS

Memcache soporta guardar claves de 250 bytes y datos de hasta 1 Megabyte. No permite ningún otro tipo de dato.

1.2.7.4.2 ESCALABILIDAD

Los servidores memcache no se comunican entre sí. Cualquier tipo de división de claves o escalabilidad se tiene que realizar en el cliente.

1.2.7.4.3 CONSISTENCIA DE DATOS

Memcache no tiene ningún tipo de consistencia de datos. No está recomendado como única fuente de datos. La mayoría de los datos en memcache tienen un temporizador por lo que expiran después de cierto tiempo. Una vez memcache se queda sin memoria, descarta los valores más viejos.

1.2.7.4.4 USUARIOS

Casi todos los sitios web grandes utilizan memcache para mantener en cache los objetos más utilizados. Algunas empresas que usan memcache son:

- ✓ Facebook
- ✓ Twitter.

1.2.7.5 REDIS

Redis es un almacén clave-valor totalmente en memoria y extremadamente rápido. Es un proyecto de código abierto con licencia BSD, y ha sido adquirido recientemente por Vmware. Redis está escrito en C y puede usarse en sistemas POSIX como Linux, UNIX, o Solaris. A diferencia de otros caches key-value, Redis acepta muchos tipos de datos distintos, y permite hacer varias operaciones sobre los claves y los datos que tiene en memoria. Esta flexibilidad es lo que ha llevado a Redis a definirse como un "servidor de estructuras de datos".

Redis es una base de datos totalmente en memoria, con lo que no puede guardar más información que la memoria disponible en el sistema.

1.2.7.5.1 MODELO DE DATOS

Redis guarda una gran cantidad de datos distintos. Redis permite guardar números enteros, cadenas de caracteres, listas, sets, sets ordenados y hasta hashes de datos.

También soporta contadores con incrementos atómicos. Esta flexibilidad permite hacer muchas operaciones sobre los datos que tiene, mientras que otras bases de datos de clave-valor necesitan que se recupere todos los datos y hacer las operaciones fuera del servidor, mientras que Redis permite hacer muchas operaciones dentro del mismo sistema, con complejidades muy bajas.

Redis se encarga de que no haya ningún otro almacenamiento clave-valor totalmente en memoria que conozca que aparte de poder hacer de cache para cualquier aplicación, permita guardar y hacer operaciones sobre los datos guardados.

1.2.7.5.2 ESCALABILIDAD

Redis es extremadamente rápido, Redis está escrito como un programa de hilo único, con lo que para poder sacar el máximo rendimiento a un servidor moderno con varios procesadores, hay que correr varias instancias de Redis concurrentemente.

Redis soporta particionamiento de datos directamente en la librería de los clientes, pero para poder hacer sistemas más avanzados de particionamiento de datos, como hashing consistente o hashing circular, es necesario hacer la programación.

Por tanto, a Redis todavía le queda trabajo por hacer para poder llegar a ser tan fácilmente escalable como Cassandra.

1.2.7.5.3 CONSISTENCIA DE DATOS

Redis tiene dos modos de funcionamiento: con archivo append-onlyy modo semi-persistente.

En modo append-only, cada cambio hecho en Redis se escribe en un log de disco. Esto afecta dramáticamente la velocidad de escritura de Redis, pero garantiza la durabilidad de los datos.

En modo semi-persistente, Redis escribe todos los cambios a disco tras un periodo configurable de tiempo, típicamente 60 segundos. Esto implica que hay una posibilidad de perder datos, pero Redis nunca será ralentizado ya que la copia se hace en background a través de un fork del propio proceso Redis, asegurando de esta forma que los datos son consistentes y que no se afecta el funcionamiento del proceso principal.

Redis soporta replicación Maestro-Esclavo, pero no replicación Maestro-Maestro. La replicación de un nuevo servidor es muy rápida de configurar e implementar.

1.2.7.5.4 USUARIOS

Entre los principales proyectos y sitios web tenemos:

- ✓ Github,
- ✓ Theguardian,
- ✓ Craigslist (uno de los portales web más grandes de estados unidos).

1.2.7.6 TOKYO CABINET

Tokyo Cabinet es una base de datos No SQL de tipo clave-valor escrita en C y funciona sobre cualquier sistema operativo POSIX. Es de código abierto. Tokyo Cabinet en si es una librería de base de datos, que combinada con TokyoTyrant crea un servidor de base de datos completo. TokyoTyrant es el servidor que gestiona las conexiones a TokyoTyrant y envía por la red la información.

Tokyo Cabinet soporta todas las restricciones ACID, y es extremadamente rápido. Desafortunadamente, es un proyecto algo parado, y no tiene el mismo seguimiento que muchos otros de los proyectos revisados.

Técnicamente sigue siendo una referencia de eficiencia y velocidad.

1.2.7.6.1 MODELO DE DATOS

Tokyo Cabinet puede guardar e indexar los datos de múltiples formas distintas. Es extremadamente modificable, y permite organizar los datos de múltiples maneras, además de especificar qué datos se guardaran en disco y que datos se guardaran en memoria. Los índices también son modificables, pudiendo elegir entre índices hash, extremadamente rápidos para retirar las claves de uno en uno, o índices BTree, ideados para devolver secciones enteras de datos. Tiene un último método de datos que es guardando datos en tuplas, parecido a una base de datos relacional tradicional.

1.2.7.6.2 ESCALABILIDAD

Tokyo Cabinet es extremadamente eficiente, algunas pruebas han determinado velocidades muy parecidas a bases de datos totalmente en memoria.

Tokyo Cabinet soporta replicación en tiempo real Maestro-Maestro, haciéndolo muy sencillo de escalar utilizando particionamiento de claves.

1.2.7.6.3 CONSISTENCIA DE DATOS

Debido a que TokyoCabinet soporta replicación Maestro-Maestro, es fácil conseguir consistencia de datos utilizando varios servidores. Además, TokyoCabinet cumple las restricciones ACID, con lo que los datos siempre estarán disponibles una vez escritos, salvo fallo de hardware.

1.2.7.7 RIAK

Riak es una base de datos No SQL inspirada en Amazon Dynamo de clave-valor.Riak está programado en Erlang y está disponible para sistemas POSIX. Riak utiliza un sistema totalmente descentralizado de nodos, parecido al que usa Cassandra y es resistente a fallos y automáticamente actualizable.

Ambos proyectos se parecen ya que ambos han tomado a Amazon Dynamo como inspiración. Sin embargo Riak ha sido una re implementación más fiel del proyecto Dynamo original, mientras que Cassandra ha añadido nuevas funcionalidades al mismo tiempo que omitió alguna funcionalidad clave de Dynamo, como la utilización de reloj vector en vez de un timestamp para marcar el tiempo en los datos recibidos.Riak tiene un soporte todavía débil para consultas tipo map-reduce.

1.2.7.7.1 MODELO DE DATOS

Riak utiliza un modelo de datos clave-valor. Sin embargo los datos pueden contener referencias a otras claves del clúster Riak.

Riak utiliza el concepto de cubos para organizar las claves, y no tiene concepto de columnas o familias de columnas como Cassandra. Riak permite varios métodos de guardar datos, llamados "back-endstorage". Actualmente utiliza GridFS o Innobase, la versión embebida de InnoDB.

1.2.7.7.2 ESCALABILIDAD

Riak es extremadamente escalable, ya que utiliza un modelo de particionamiento de datos automático dependiendo del número de equipos en el clúster. Permite ajustar el número de nodos de replica que se guardara para los datos y el número de lecturas necesarias para devolver un dato como válido.

1.2.7.7.3 CONSISTENCIA DE DATOS

Riak es eventualmente consistente, con lo que no garantiza que un dato escrito este inmediatamente disponible después de haberlo escrito.

1.2.7.8 PROJECT VOLDEMORT

Project Voldemort es un sistema distribuido de clave valor desarrollado en la web social Linkedin.Utiliza un sistema totalmente descentralizado de nodos que particiona los datos automáticamente a través de todos los nodos disponibles. Maneja transparentemente el fallo de cualquier servidor.

El sistema de socialización es intercambiable, con lo que soporta enviar datos en formato Protocol Buffers de Goolge, Thrift, o hasta socialización Java. Mantiene un versionado de los datos de forma parecida a CouchDB. Voldemort no necesita un nivel de cache independiente, ya que gestiona su propia memoria como cache a través de todos los nodos.

1.2.7.8.1 MODELO DE DATOS

Voldemort es un modelo puramente clave valor. Permite guardar valores de cualquier longitud asociados a una clave.

1.2.7.8.2 ESCALABILIDAD

Voldemort escala tanto lecturas como escrituras simplemente añadiendo más nodos al clúster debido al modelo hash ring de distribución de lecturas y escrituras. Las estrategias de particionamiento son totalmente configurables.

1.2.7.8.3 CONSISTENCIA DE DATOS

Voldemort garantiza la consistencia de los datos a través del clúster configurando el número de réplicas de datos que se desean guardar.

1.2.7.9 MONGODB

MongoDB es una base de datos de documentos, diseñada para reemplazar las bases de datos SQL tradicionales de uso general. Es de código abierto y escrita en C++. Actualmente está disponible para Windows, Linux y otros sistemas operativos POSIX.
MongoDB fue diseñada por la empresa 10gen de Nueva York, que actualmente ofrecen consultoría y soporte técnico para MongoDB. Aunque el desarrollo de MongoDB empezó en Octubre del 2007, MongoDB se hizo público en Febrero del 2009.
La intención de MongoDB es proporcionar más funcionalidad que la típicamente proporcionada por bases de datos No SQL de tipo clave – valor, permitiendo sistemas consultas ad-hoc, indexación de ciertos valores dentro del documento y manteniendo un rendimiento elevado y facilidad para distribuir la carga en varios servidores
Las características que más destacan a MongoDB son: su velocidad y su sencillo sistema de consulta de los contenidos de la base de datos. Se podría decir que alcanza un balance perfecto entre rendimiento y funcionalidad, incorporando muchos de los tipos de consulta que utilizaríamos en nuestro sistema relacional preferido, pero sin sacrificar en rendimiento.

.

1.2.7.9.1 MODELO DE DATOS

El modelo de datos sigue el de una base de datos de documentos. Cada documento tiene su propio id, y un número indeterminado y flexible de campos con información. Sin embargo, a diferencia de otros muchos sistemas de bases de datos No SQL de documentos, MongoDB permite indexar ciertos campos de datos dentro del documento. Esto redunda en una velocidad mucho mayor para las búsquedas sobre esos campos, aunque afecta ligeramente la velocidad de inserción de documentos, ya que hay que escribir al índice además de escribir el documento.

MongoDB guarda cada documento en un formato BSON, que es un acrónimo de Binary JSON. Cada documento es un archivo JSON (un formato de representación des estructurado parecido a XML) que a su vez es codificado en binario. Todas las paginas se guardan en "cubos" que no son más que archivos de 2GB de tamaño reservados de antemano por MongoDB. Estos cubos se van creando secuencialmente a medida que van siendo utilizados. Debido a que MongoDB guarda la relación de las claves que se encuentran en cada cubo, el acceso a los documentos es extremadamente rápido. Estos archivos son cargados en memoria con llamadas al sistema map, y MongoDB tiene su propio sistema para gestionar el uso de la memoria.

A diferencia de otras bases de datos de documentos, MongoDB permite consultas Adhoc parecido a la forma que se pueden hacer con el lenguaje SQL. MongoDB permite buscar documentos por cualquier campo, en vez de únicamente por su clave como es común en el resto de los sistemas, aunque buscar un documento por un campo que no tiene índice puede resultar muy lento.

Para consultas de agregación, MongoDB soporta operaciones MapReduce, aunque no está optimizado específicamente para ellas, es suficientemente rápido para trabajos ocasionales de reportes.

1.2.7.9.2 ESCALABILIDAD

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

El concepto fundamental de sharding es bastante similar a la idea de maestro-trabajador el patrón de la columna de base de datos, donde se distribuye a través de

servidores de rango múltiple. MongoDB permite colecciones ordenadas para ser salvos a través de múltiples máquinas. Cada máquina que ahorra parte de la colección es entonces un fragmento, los fragmentos se replican para permitir la conmutación por error. Así, una gran colección podría ser dividida 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 se encuentran 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 único nodo, mientras que otro en la misma base de datos puede ser fragmentado a cabo con varios nodos.

1.2.7.9.3 CONSISTENCIA DE DATOS

MongoDB soporta replicación de datos. Al combinarlo con soporte de particionamiento de datos, esto permite una redundancia de datos. Sin embargo, le faltan sistemas de recuperación automática del maestro en caso de fallo. Una vez más, se puede conseguir redundancia total de datos en MongoDB, sin embargo, no es sencillo y requiere bastante trabajo de administración de sistemas para que todo el proceso sea relativamente automático.

MongoDB no tiene durabilidad de datos en un solo servidor. Esto quiere decir que MongoDB puede perder datos si no se configura con una réplica. Esto se debe a que MongoDB no cumple con las propiedades ACID, específicamente con la D de durable. Si MongoDB termina inesperadamente puede provocar la pérdida de varias transacciones. Solamente en un ambiente replicado, donde hay dos servidores o más servidores Mongo con los mismos datos es que la consistencia de los datos está garantizada.

1.2.7.9.4 USUARIOS

Los usuarios de MongoDB principales son las redes sociales y navegadores tales como:

- ✓ Facebook
- ✓ Twetter
- ✓ Google.

1.2.7.9.5 TERMINOLOGÍA BÁSICA

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 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.

1.2.7.9.6 CARACTERÍSTICAS PRINCIPALES

- ✓ **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 del 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 elije una llave shard, la cual determina cómo serán distribuidos los datos en una colección. Los datos son divididos en rangos 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 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 se denomina GridFS y 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 se utiliza en MongoDB 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".

1.2.7.9.7 POR QUE USAR MONGODB

El enfoque No SQL presenta enormes ventajas respecto a las bases de datos SQL, ya que permite escalar una aplicación a nuevos niveles. Los nuevos servicios de datos se basan en estructuras verdaderamente escalables y arquitecturas, construido para la nube, construido para su distribución, y son muy atractivos para el desarrollador de aplicaciones. No hay necesidad de DBA, sin necesidad de complicadas consultas SQL y es rápido. Estas son todas las ventajas principales de usar No SQL. Mongodb es una base de datos documental que se almacena mediante colecciones.

1.2.7.9.8 HERRAMIENTAS EXTRAS DE MONGODB

MongoDB necesita de algunas herramientas extras para su total funcionalidad entre ellas cabe mencionar las siguientes:

1.2.7.9.8.1 MAP REDUCE

MapReduce es un marco de software patentado de Google que apoya la computación distribuida en un gran grupo distribuida de computadoras. MapReduce de Google ha inspirado a muchos clones y los marcos de computación distribuida en la comunidad de código abierto. MongoDB es una

de esas. Google y las características de MapReduce MongoDB se inspira también en construcciones similares en el mundo de la programación funcional.
MapReduce a veces puede ser intimidante, pero una vez que entienda su estructura y el flujo, es una poderosa herramienta que le ayuda a realizar los cálculos grandes en las colecciones distribuidas de datos. Así, partiendo de ejemplos simples y luego de graduarse a los más complejos es una buena manera de suavizar la curva de aprendizaje y lograr el dominio del tema.
El ejemplo más simple agregación podría ser un recuento de cada tipo de un elemento en una colección. Para utilizar MapReduce, es necesario definir una función de mapa y una función de reducir y luego ejecute el mapa y reducir funciones contra una colección. Una función map aplica una función a cada miembro de la colección y emite un par clave / valor para cada miembro como resultado de este proceso. La salida de clave / valor de una función de mapa es consumido por la función reduce. La función reduce ejecuta una función de agregación en todos los pares clave / valor y genera una salida a su vez.

1.2.7.9.8.2 APACHE MAVEN

Es una gestión de proyectos de software y una herramienta de comprensión. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar un proyecto de compilación, presentación de informes y la documentación de una pieza central de la información.
Maven, significa acumulador de conocimiento, el resultado es una herramienta que ahora se puede utilizar para crear y gestionar cualquier proyecto basado en Java.
El Objetivo principal de Maven es permitir al desarrollador comprender el estado completo de un esfuerzo de desarrollo en el menor período de tiempo. Para alcanzar este objetivo hay varias áreas de preocupación que Maven intenta tratar:

- ✓ Hacer que el proceso de construcción sea fácil
- ✓ Proporcionar un sistema de construcción uniforme
- ✓ El suministro de información de calidad del proyecto
- ✓ Proporcionar directrices para las mejores prácticas de desarrollo
- ✓ Permitir la migración transparente a nuevas características

1.2.7.9.8.3 SPRING-MVC

Es uno de los módulos del Framework de Spring, y como su propio nombre nos indica implementa una arquitectura Modelo – Vista – Controlador.

La Red Spring modelo-vista-controlador (MVC) está diseñado en torno a un DispatcherServlet que despacha las peticiones a los manipuladores, con asignaciones de controlador configurable, la configuración regional y la resolución de tema, así como soporte para la carga de archivos. El manejador por defecto se basa en las anotaciones @Controller y @RequestMapping, ofreciendo una amplia gama de métodos de manipulación flexibles. Con la introducción de Srping 3.0, el mecanismo @Controller también permite crear sitios Web RESTful y aplicaciones, a través del @Path Variable anotación y otras características.

En Spring Web MVC se puede utilizar cualquier objeto como un objeto de comando o forma de respaldo, no se necesita implementar un marco específico de interfaz o clase base. Spring data es muy flexible, por ejemplo, trata los tipos no coincidentes como los errores de validación que pueden ser evaluados por la aplicación, no como errores del sistema. Por lo tanto no es necesario duplicar las propiedades de los objetos de negocios como cadenas simples, sin tipo de objetos de formulario simple para manejar las comunicaciones no válidas, o para convertir las cadenas correctamente. En su lugar, a menudo es preferible para unirse directamente a los objetos de su negocio.

En Spring Data un Controller es normalmente responsable de la preparación de un modelo de Map con los datos y la selección de un nombre de vista, pero también puede escribir directamente en la secuencia de respuesta y completar la solicitud. El modelo (la M en MVC) es un Map de la interfaz, la cual permite la abstracción completa de la tecnología de vista. Se puede integrar directamente con las tecnologías de la plantilla de representación basados tales como JSP, Velocity y Freemarker, o directamente generar XML, JSON, Atom, y muchos otros tipos de contenido. El modelo de Map simplemente se transforma en un formato apropiado, tal como atributos de la petición JSP, un modelo de plantilla Velocity.

El Módulo de Spring Web MVC incluye muchas características únicas de soporte Web:

- ✓ Una clara separación de roles: Cada función - controlador, validador, objeto de comando, objeto de formulario, objeto modelo, Dispatcher Servlet , asignación de controlador, resolución de vista, se puede cumplir por un objeto especializado.
- ✓ Configuración de gran alcance y directa de ambas clases de aplicación y marco como JavaBeans: esta capacidad de configuración incluye referencias fáciles a través de contextos, como el de los controladores web a los objetos de negocio y validadores.
- ✓ Adaptabilidad, no intrusivo, y la flexibilidad: Defina cualquier firma de método controlador que necesita, posiblemente usando una de las anotaciones de parámetros (como @RequestParam, @RequestHeader, @PathVariable, y más) para un escenario dado.
- ✓ Código reutilizable sin necesidad de duplicación: Utilice los objetos de negocio existentes como objetos de comando o formulario en lugar de reflejar a extender una clase marco básico particular.
- ✓ Encuadernación personalizable y validación: tipos no coincidentes como los errores de validación de nivel de aplicación que mantienen el valor ofender, fecha localizada y vinculante número, y así sucesivamente en vez de objetos String de sólo formulario con el análisis manual y conversión a los objetos de negocio.
- ✓ Asignación de controlador personalizable y la resolución de vista: Asignación de controlador y las estrategias de resolución de vista van desde la simple dirección URL de configuración basada, a las estrategias sofisticadas, resolución de propósito específico. La primavera es más flexible que la web frameworks MVC que exigen una técnica particular.
- ✓ Modelo flexible de transferencia: Modelo con la transferencia de un nombre / valor Map soporta una fácil integración con cualquier tecnología de visión.

1.2.7.9.8.4 SPRING DATA

Es un proyecto de Spring Source cuyo propósito es unificar y facilitar el acceso a distintos tipos de tecnologías de persistencia, tanto a bases de datos relacionales como a las del tipo No SQL.

Spring ya proporcionaba soporte para JDBC, Hibernate, JPA, JDO o MyIbatis, simplificando la implementación de la capa de acceso a datos, unificando la configuración y creando una jerarquía de excepciones común para todas ellas.

Y ahora, Spring Data viene a cubrir el soporte necesario para distintas tecnologías de bases de datos No SQL y, además, integra las tecnologías de acceso a datos tradicionales, simplificando el trabajo a la hora de crear las implementaciones concretas.

Con cada tipo de tecnología de persistencia los DAO (Data Access Objects) ofrecen las funcionalidades típicas de un CRUD (Create, Read, Update, Delete) para objetos de dominio propios, métodos de búsqueda, ordenación y paginación. Spring Data proporciona interfaces genéricas para estos aspectos e implementaciones específicas para cada tipo de tecnología de persistencia.

Hoy en día Spring Data proporciona soporte para las siguientes tecnologías de persistencia:

- ✓ JPA y JDBC
- ✓ Apache Hadoop
- ✓ GemFire
- ✓ Redis
- ✓ MongoDB
- ✓ Neo4j
- ✓ HBase

1.2.7.9.8.5 POR QUE USARSPRING DATA

Por la facilidad de interacción con MongoDB, presenta las siguientes características:

- ✓ Mapeo/Conversión entre documentos MongoDB
- ✓ Plantillas Mongo
- ✓ Implementación automática de repositorios (DAO)
- ✓ DSL basado en Java para consultas y actualizaciones
- ✓ Soporte a persistencia mixta

- ✓ Integración con Geo Espacial de Mongo
- ✓ Integración con Map Reduce de MongoDB
- ✓ Administración y monitorización por JMX

1.2.7.9.9 LATENCIA

Hay aplicaciones en las que es crítica la latencia de la respuesta. Estas aplicaciones típicamente hacen una gran cantidad de consultas sencillas sobre la clave primaria de tabla. Los requisitos son un thoroughput muy elevado de consultas al segundo, y una latencia máxima muy baja. Este tipo de aplicaciones incluyen como ejemplo a caches, donde es muy importante retirar rápidamente la información cacheada o averiguar si no existe.

Los sistemas de almacenamiento No SQL Clave-Valor (o Key/Value) son sistemas sencillos que dados una clave, retornan el valor asociado a la clave.

Son extremadamente rápidos y mantienen latencias muy bajas ya que están especializados en devolver los datos de la forma más rápida posible, y todos los algoritmos internos están optimizados para la búsqueda de claves, típicamente con complejidad. De esta forma tardan lo mismo para devolver una clave cuando hay 100 claves disponibles que cuando hay millones de claves disponibles.

A continuación se muestra un gráfico tomado del sitio web aadkw.com donde se muestra el tiempo que tarda cada base de datos en insertar un dato.

Tabla 5: Tiempo de Inserción de datos en MongoDB

(45 M de documentos)	Tiempo medio de inserción	Tamaño en disco
SolR	0.603(ms)	49 GB
CouchDB	0.297(ms)	43 GB
Cassandra	0.516(ms)	50 GB
MongoDB	0.040(ms)	43 GB

1.2.8 ARQUITECTURA DE BASE DE DATOS No RELACIONALES

Bases de datos documentales y bases de datos orientadas a columnas son los tipos de bases de datos más populares en bases de datos no relacionales. Las arquitecturas de bases de datos no relacionales se pueden aprovechar para mantener grandes cantidades de datos para ser procesados con mucha eficiencia dentro de un plazo razonable.

Existen tres características importantes en las arquitecturas de bases de datos No Relacionales:

- ✓ Los datos tienen que ser almacenados en un sistema en red inalámbrica que puede expandirse para varias máquinas. Los archivos pueden ser muy grandes y se almacenan en múltiples nodos, y cada uno ejecuta en una máquina distinta.
- ✓ Los datos tienen que ser almacenados en una estructura que proporciona más flexibilidad que la tradicional base de datos relacional. El mecanismo de almacenamiento debe permitir efectividad en el almacenamiento de grandes cantidades de conjuntos de datos dispersos. Se necesita acomodar para cambiar esquemas sin la necesidad de alterar las tablas subyacentes.
- ✓ Los datos tienen que ser procesados de una manera que los cálculos que se puedan realizar en subconjuntos aislados se puedan combinar para generar la salida deseada.

Esto implicaría cálculo de eficiencia si los algoritmos se ejecutan en los mismos lugares donde residen los datos.

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 No SQL puede servir gran cantidad de carga de lecturas y escrituras.

Las arquitecturas No SQL 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. Hay dos sistemas que han sido desplegados y que aportan aislamiento snapshot para almacenamientos de columna: El sistema Percolator de Google (basado en el sistema BigTable) y el sistema transaccional de Hbase. 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.

Bastantes sistemas No SQL 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 No SQL 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.

Las bases de datos No SQL son únicas ya que suelen ser independientes del lenguaje de consulta estructurado (SQL) que si condiciona a las bases de datos relacionales. En las bases de datos relacionales es necesario el uso de un lenguaje específico de SQL para las consultas ad hoc que deseemos realizar, mientras que las bases de datos no-relacionales no tienen ningún tipo de lenguaje de estándar de consulta, por lo que se puede usar el que cada uno quiera.

Las bases de datos No SQL están concebidas para obtener una altísima capacidad de volumen de almacenamiento y velocidad de proceso de la información. Para lograr esto, el lenguaje No SQL usa técnicas que pueden asustar a los gestores de bases de datos relacionales, como el que los datos que componen la data, no son coherentes todo el tiempo dentro del sistema.

Las ventajas más significativas de la arquitectura de datos No SQL son:

- ✓ **Escalabilidad:** Se pueden escalar con relativa facilidad ante demandas puntuales de sobre carga de datos.
- ✓ **Rendimiento:** Para obtener un mejor rendimiento en el procesamiento de los datos sólo es necesario añadir más recursos en la plataforma hardware o priorizar cual son los servicios críticos en cada momento.
- ✓ **Estructura:** Los desarrolladores de aplicaciones que trabajan con bases de datos relacionales muchas veces encuentran problemas con la cartografía de los datos y su impedancia. En las bases de datos No SQL, esto no es generalmente un problema, ya que los datos no se almacenan en la misma manera.
- ✓ **Activación/Desactivación:** Debido a la naturaleza distribuida de los datos, los modelos No SQL responden muy bien ante la activación/desactivación de los

servicios en base a las necesidades puntuales de demanda por parte de los usuarios o del mismo sistema.

Las Bases de datos No Relacionales emplean una arquitectura distribuida, donde los datos se guardan de modo redundante en distintos servidores.

1.2.8.1 TOPOLOGÍA DE LA ARQUITECTURA MONGODB

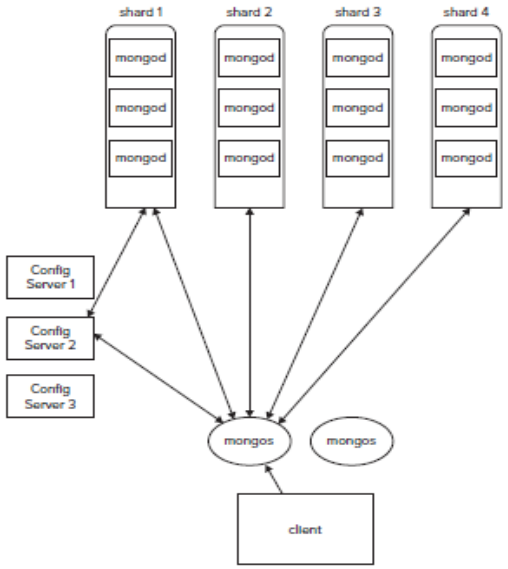


Figura 14: Arquitectura MongoDB

1.2.8.2 CARACTERÍSTICAS DE ARQUITECTURAS DE DATOS SQL Y No SQL

Tabla 6: Características de arquitecturas de datos SQL y No SQL

SQL	No SQL
✓ Cuando el volumen de datos crece, o lo hace poco a poco.	✓ Cuando el volumen de datos crece rápidamente en momentos puntuales.
✓ Cuando las necesidades de proceso se pueden asumir en un solo servidor.	✓ Cuando las necesidades de proceso no se pueden proveer.
✓ Cuando no tenemos problemas de uso del sistema por parte de los usuarios más allá de los previstos.	✓ Cuando tenemos problemas de uso del sistema por parte de los usuarios en múltiples ocasiones.

CAPITULO II

2 DISEÑO DE BASE DE DATOS

2.1 DISEÑO DE BASE DE DATOS RELACIONAL

Esta etapa consta de tres fases:

- ✓ Diseño Conceptual
- ✓ Diseño Lógico
- ✓ Diseño Físico de la base de datos

La primera fase consiste en la producción de un esquema conceptual de los datos, que es independiente de todas las consideraciones físicas. Este modelo se refina después en un esquema lógico eliminando las construcciones que no se pueden representar en el modelo de base de datos escogido. En la tercera fase, el esquema lógico se traduce en un esquema físico para el SGBD escogido.

La fase de diseño físico debe tener en cuenta las estructuras de almacenamiento y los métodos de acceso necesarios para proporcionar un acceso eficiente a la base de datos en memoria secundaria.

Los objetivos del diseño de la base de datos son:

- ✓ Representar los datos que requieren las principales áreas funcionales y los usuarios, y representar las relaciones entre dichos datos.
- ✓ Proporcionar un modelo de los datos que soporte las transacciones que se vayan a realizar sobre los datos.

Especificar un esquema que alcance las prestaciones requeridas para el sistema.

2.1.1 DISEÑO DE LA APLICACIÓN

En esta etapa se diseñan los programas de aplicación que usarán y procesarán la base de datos. Esta etapa y el diseño de la base de datos, son paralelas.

En la mayor parte de los casos no se puede finalizar el diseño de las aplicaciones hasta que se ha terminado con el diseño de la base de datos. Por otro lado, la base de datos existe para dar soporte a las aplicaciones, por lo que habrá una realimentación desde el diseño de las aplicaciones al diseño de la base de datos.

Hay que asegurarse de que toda la funcionalidad especificada en los requisitos de usuario se encuentra en el diseño de la aplicación.

Además, habrá que diseñar las interfaces de usuario, aspecto muy importante que no se debe ignorar. El sistema debe ser fácil de aprender, fácil de usar, directo y estar dispuesto a tolerar ciertos fallos de los usuarios.

2.1.1.1 PROTOTIPADO

Esta etapa, que es opcional, es para construir prototipos de la aplicación que permitan a los diseñadores y a los usuarios probar el sistema. Un prototipo es un modelo de trabajo de las aplicaciones del sistema. El prototipo no tiene toda la funcionalidad del sistema final, pero es suficiente para que los usuarios puedan utilizar el sistema e identificar qué aspectos están bien y cuáles no son adecuados, además de poder sugerir mejoras o la inclusión de nuevos elementos.

Este proceso permite que quienes diseñan e implementan el sistema sepan si han interpretado correctamente los requisitos de los usuarios. Otra ventaja de los prototipos es que se construyen rápidamente.

Esta etapa es imprescindible cuando el sistema que se va a implementar tiene un gran costo, alto riesgo o utiliza nuevas tecnologías.

2.1.1.2 IMPLEMENTACION

En esta etapa se crean las definiciones de la base de datos a nivel conceptual, externo e interno, así como los programas de aplicación. La implementación de la base de datos se realiza mediante las sentencias del lenguaje de definición de datos del SGBD escogido. Estas sentencias se utilizan para crear el esquema físico de la base de datos, en donde se almacenarán los datos de la base y las vistas de los usuarios.

Los programas de aplicación se implementan utilizando lenguajes de tercera o cuarta generación. Parte de estas aplicaciones son transacciones sobre la base de datos, que se implementan mediante el lenguaje de manejo de datos del SGBD. Las sentencias de este lenguaje se pueden embeber en un lenguaje de programación anfitrión como Visual Basic, Delphi, C, C++ o Java, entre otros. En esta etapa también se implementan los menús, los formularios para la introducción de datos y los informes de visualización de datos. Para ello, el SGBD puede disponer de lenguajes que permiten el desarrollo rápido de aplicaciones mediante lenguajes de consultas no procedurales, generadores de informes, generadores de formularios, generadores de gráficos y generadores de aplicaciones.

En esta etapa también se implementan todos los controles de seguridad e integridad. Algunos de estos controles se pueden implementar mediante el lenguaje de definición de datos y otros pueda que haya que implementarlos mediante utilidades del SGBD o mediante los programas de aplicación.

2.1.1.3 CONVERSION Y CARGA DE DATOS

Esta etapa es necesaria cuando se está reemplazando un sistema antiguo por uno nuevo. Los datos se cargan desde el sistema viejo al nuevo directamente o, si es necesario, se convierten al formato que requiera el nuevo SGBD y luego se cargan. Si es posible, los programas de aplicación del sistema antiguo también se convierten para que se puedan utilizar en el sistema nuevo.

2.1.1.4 PRUEBAS

En esta etapa se prueba y valida el sistema con los requisitos especificados por los usuarios. Para ello, se debe diseñar una batería de test con datos reales, que se deben llevar a cabo de manera metódica y rigurosa. Es importante darse cuenta de que la fase de prueba no sirve para demostrar que no hay fallos, sirve para encontrarlos. Si la fase de prueba se lleva a cabo correctamente, descubrirá los errores en los programas de aplicación y en la estructura de la base de datos. Además, demostrará que los programas parecen trabajar tal y como se especificaba en los requisitos y que las prestaciones deseadas parecen obtenerse. Por último, en las pruebas se podrá hacer una medida de la fiabilidad y la calidad del software desarrollado.

2.1.1.5 MANTENIMIENTO

Una vez que el sistema está completamente implementado y probado, se pone en marcha. Se dice que el sistema está ahora en la fase de mantenimiento, en la que se llevan a cabo las siguientes tareas:

Monitorización de las prestaciones del sistema. Si las prestaciones caen por debajo de un determinado nivel, puede ser necesario reorganizar la base de datos.

Mantenimiento y actualización del sistema. Cuando sea necesario, los nuevos requisitos que vayan surgiendo se incorporarán al sistema, siguiendo de nuevo las etapas del ciclo de vida que se acaban de presentar.

2.1.2 DISEÑO CONCEPTUAL

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se

le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren el significado de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- ✓ La perspectiva que cada usuario tiene de los datos.
- ✓ La naturaleza de los datos, independientemente de su representación física.
- ✓ El uso de los datos a través de las áreas funcionales.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación, que se describe en el capítulo dedicado al diseño conceptual.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física.

Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

2.1.3 DISEÑO LÓGICO

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como pueden ser: el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las tablas obtenidas no tienen datos redundantes. Esta técnica se presenta en el capítulo dedicado al diseño lógico de bases de datos.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione después correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos) o mantener la integridad de la base de datos.

También puede ser difícil definir la implementación física o mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños.

Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

2.1.4 DISEÑO FÍSICO

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: determinar las estructuras de almacenamiento y los métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lóg ico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- ✓ Obtener un conjunto de tablas y determinar las restricciones que se debe cumplir sobre ellas.

- ✓ Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
- ✓ Diseñar el modelo de seguridad del sistema.

2.1.5 MODELO ENTIDAD RELACIÓN

Los diagramas o modelos entidad-relación son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

A continuación se muestra el Diagrama Entidad – Relación del Sistema Informático del Departamento de Transportes de la Unach.

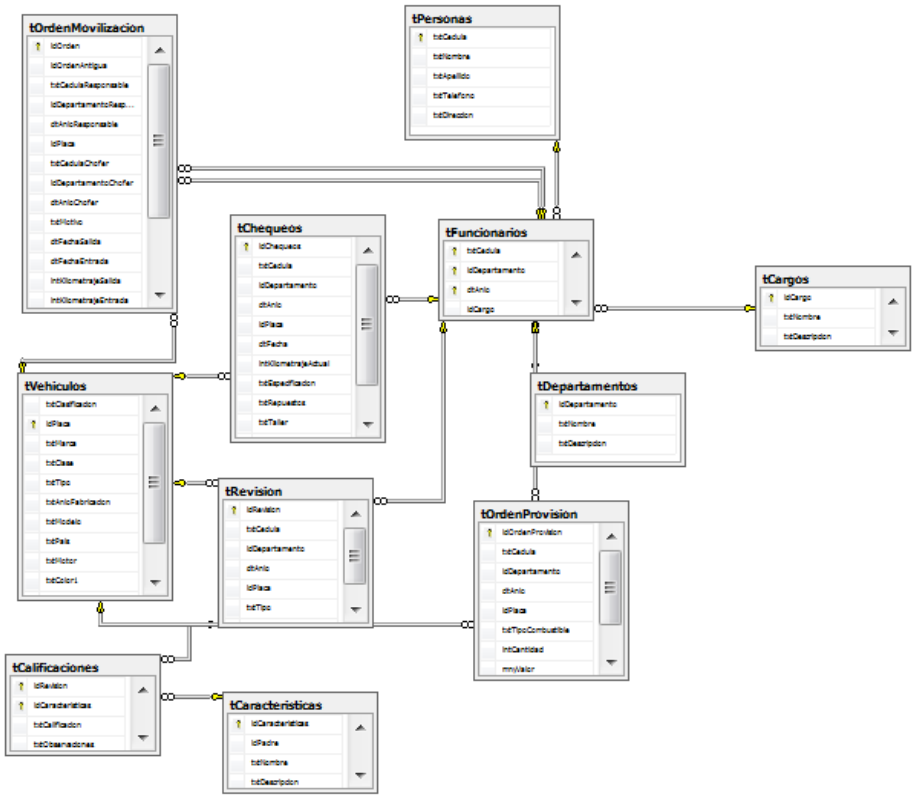


Figura 15: Estructura del modelo Entidad Relación

El lenguaje Microsoft SQL Server es el más universal en los sistemas de base de datos relacionales. Este lenguaje nos permite realizar consultas a nuestras bases de datos para mostrar, insertar, actualizar y borrar datos.

2.2 DISEÑO DE BASE DE DATOS No RELACIONALES

MongoDB forma parte de la nueva familia de sistemas de base de datos No SQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un

esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. MongoDB es una base de datos sin esquema.

De forma rápida y sencilla, un documento JSON no es más que la descripción de un objeto en formato JSON, un formato muy rápido para el intercambio de datos y muy legible para el humano.

Una colección en MongoDB es un grupo de documentos y es el equivalente a una tabla en un modelo relacional.

Para el diseño de una base de datos no relacional se requiere las siguientes fases:

2.2.1 ESTUDIO DE PROBLEMA

En primera instancia es necesario realizar un estudio de necesidades y alcance del proyecto.

Además, evaluar su complejidad y si el dominio es viable para el tiempo que se dispone para su desarrollo.

En la investigación el estudio del problema es realizar un prototipo de base de datos no relacional usando MongoDB para el departamento de Transportes de la Unach.

2.2.2 IDENTIFICACIÓN DE REQUISITOS

Una vez realizado el estudio del problema, se debe realizar una identificación de todos los requisitos funcionales que presentará el sistema y así pasar a la siguiente fase de diseño del meta modelo.

La información que será usada en la base de datos no relacional ha sido extraída directamente de la Unidad de Transportes de la Universidad Nacional de Chimborazo.

El proyecto debe proporcionar una propuesta rápida a las distintas gestiones las cuales se dividen en 4 bloques:

a. Información Administrativa
   - ✓ Director del Departamento
   - ✓ Secretaria del Departamento

b. Información Vehicular
   - ✓ Revisión Vehicular
   - ✓ Provisión de Combustible
   - ✓ Datos de los Vehículos
   - ✓ Mantenimiento Vehicular

c. Información de Movilización:
   - ✓ Órdenes de Movilización (Dentro y Fuera de la ciudad)
   - ✓ Acta Entrega Recepción de los Vehículos

d. Información de Recursos Humanos:
   - ✓ Datos personales de Choferes
   - ✓ Datos Personales de Funcionarios
   - ✓ Cargo
   - ✓ Departamento al que pertenece

Como mencionamos anteriormente se realizará un prototipo de base de datos No Relacional para eso usaremos dos tablas en MongoDB serían dos documentos (Funcionarios y Vehículos).

2.2.3 MODELADO DEL NEGOCIO

El modelado del negocio se basa en dos diagramas principales, el modelo de casos de uso del negocio, el modelo del dominio y los modelos de objetos del negocio.

La Unidad interactúa con distintos elementos externos, entre los que se identifican el usuario externo (persona o entidad que solicita un vehículo a la unidad), personal docente (docentes de las distintas facultades) Personal Administrativo (de las distintas facultades),

2.2.3.1 MODELO DE CASO DE USO

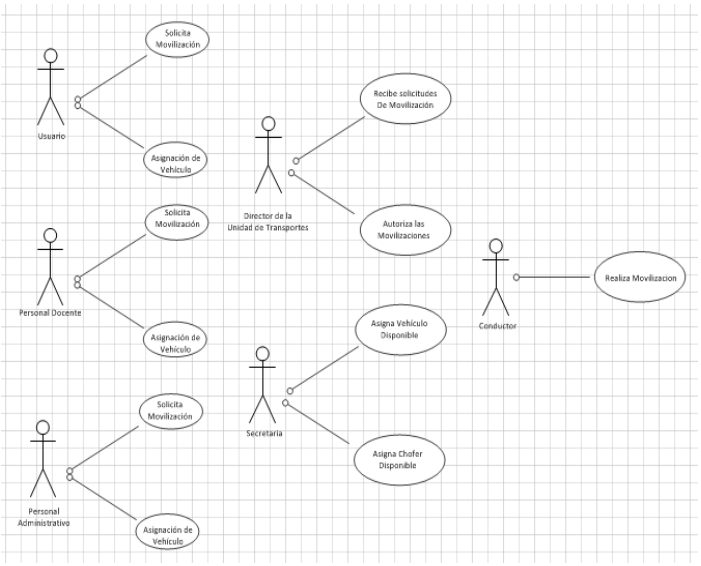


Figura 16: Modelo de Caso de Uso Base de Datos No Relacional

2.2.3.2 MODELO DE DOMINIO

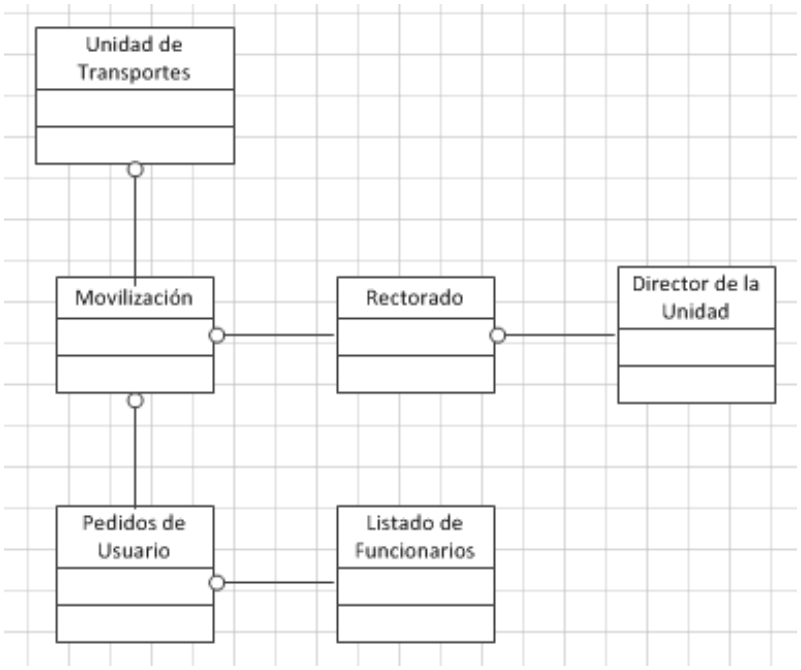


Figura 17: Modelo de Dominio Base de Datos No Relacional

2.2.3.3 MODELO DE OBJETOS PARA SOLICITAR UNA MOVILIZACIÓN

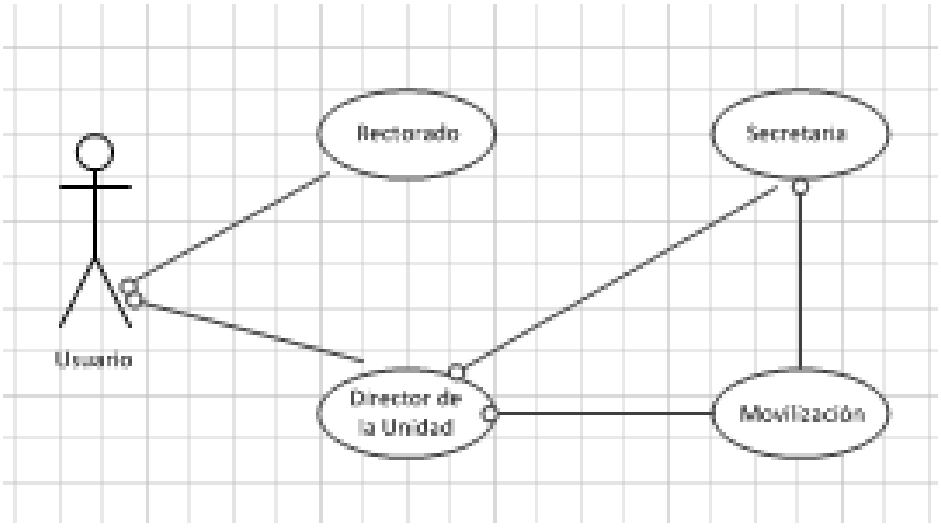


Figura 18: Modelo de Objetos para Solicitar Movilización Base de Datos No Relacional

2.2.3.4 MODELO DE OBJETOS PROVISIÓN DE COMBUSTIBLE

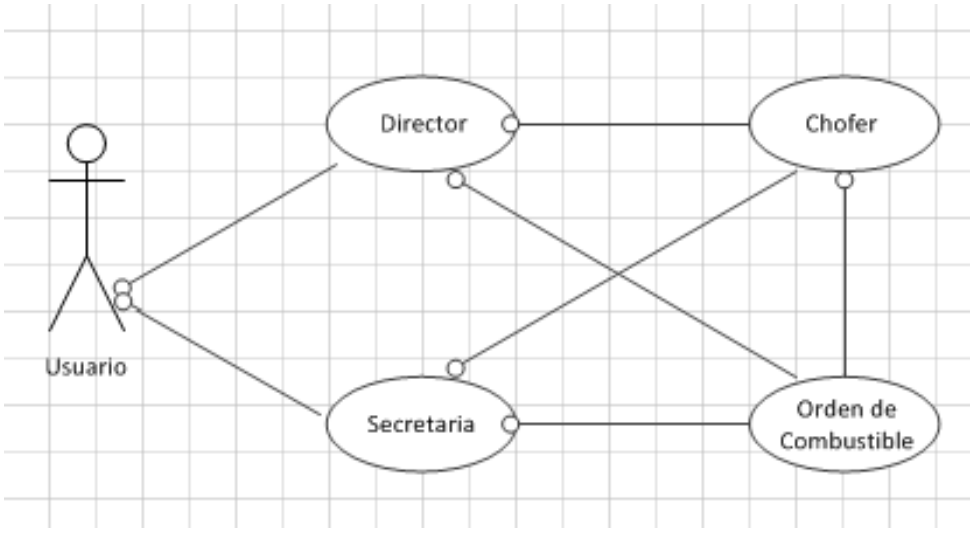


Figura 19: Modelo de Objetos Provisión de Combustible Base de Datos No Relacional

2.2.3.5 MODELO DE OBJETOS MANTENIMIENTO VEHICULAR

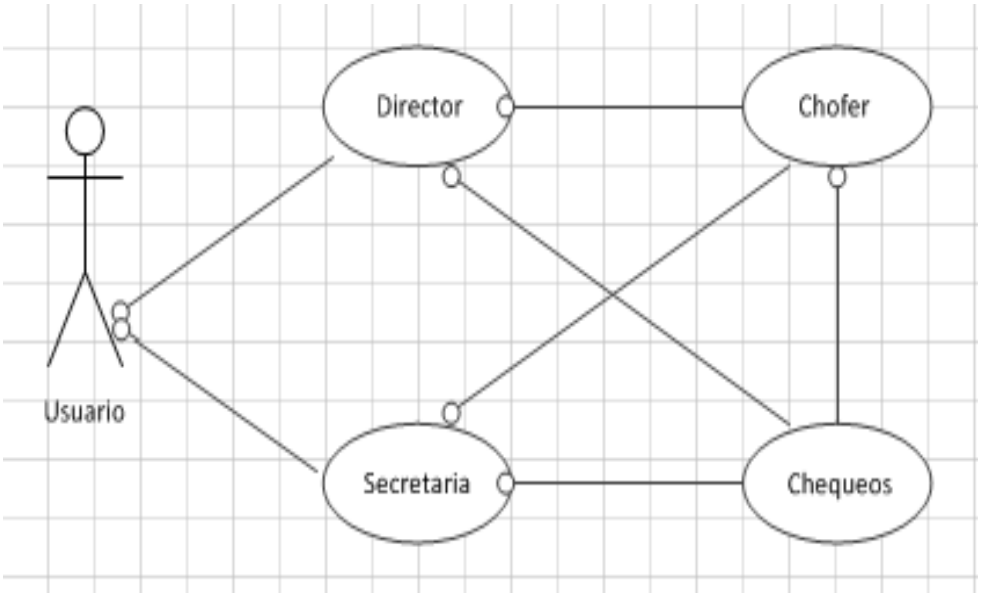


Figura 20: Modelo de Objetos para Mantenimiento Vehicular Base de Datos No Relacional

2.2.3.6 PROCESO PARA EFECTUAR UNA MOVILIZACIÓN

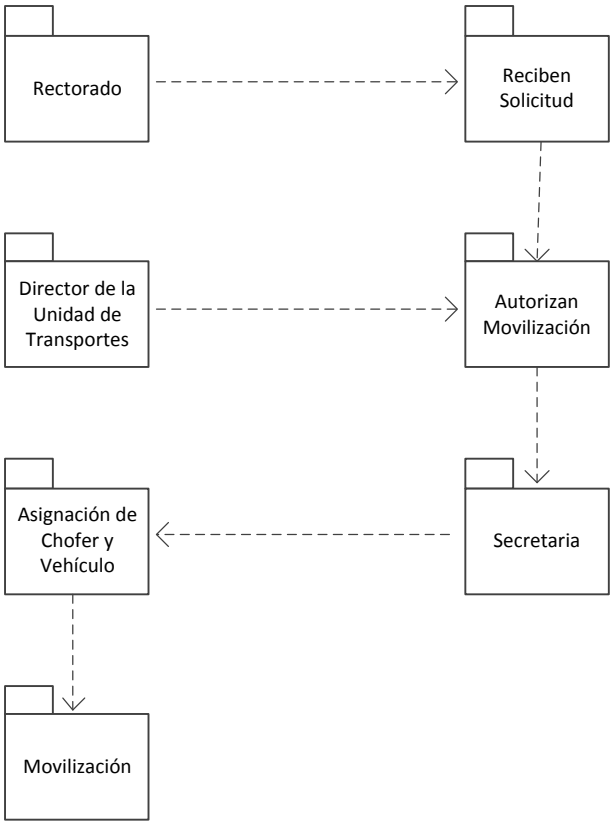


Figura 21: Proceso para Efectuar una Movilización Base de Datos No Relacional

2.2.4 MODELADO DE DATOS No SQL

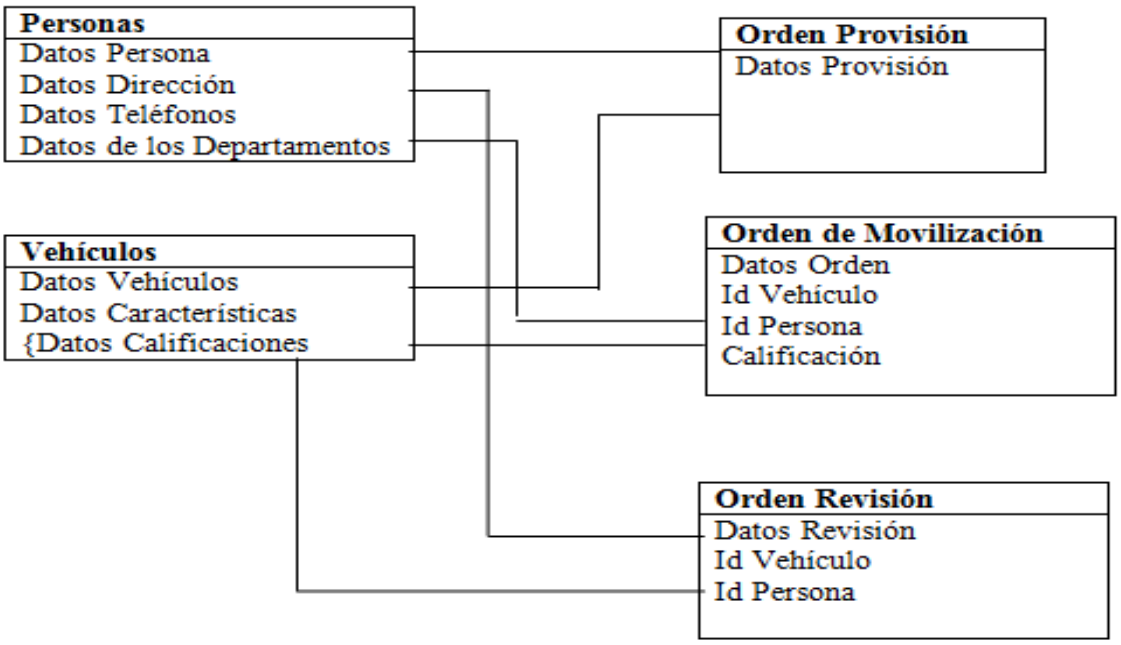


Figura 22: Modelo de Base de Datos No Relacional

CAPITULO III

3 IMPLEMENTACIÓN DE BASE DE DATOS No RELACIONAL

3.1 INSTALACIÓN DE MONGO

Descargar la última versión de MongoDB, dependiendo el Sistema Operativo que se esté utilizando.

Tabla 7: Descarga de MongoDB

	OS X 64-bit	Linux 32-bit note	Linux 64-bit	Windows 32-bit note	Windows 64-bit	Solaris 64-bit	Source
Production Release (Recommended)							
2.4.3 *4/23/2013* Changelog Release Notes	download	download	download	download	download *2008R2+	download	tgz zip
Nightly Changelog	download	download	download	download	download *2008R2+	download	tgz zip
Previous Release							
2.2.4 *4/2/2013* Changelog Release Notes	download	download	download	download	download *2008R2+	download	tgz zip

✓ Una vez descargado se guarda con el nombre (mongodb-win32-i386-2.2.3) este archivo extraemos en el disco C: y cambiamos de nombre dejando simplemente como mongodb. La ubicación en el disco C: y no en otra unidad es por cuestión de permisos de administrador.

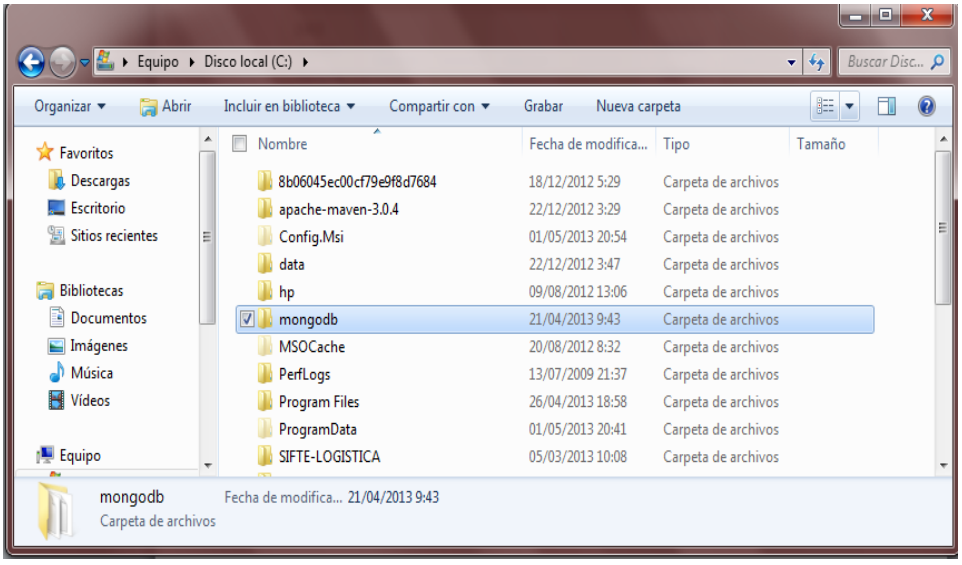


Figura 23: Descarga de MongoDB

✓ Cree una carpeta con el nombre **data** en el disco C,

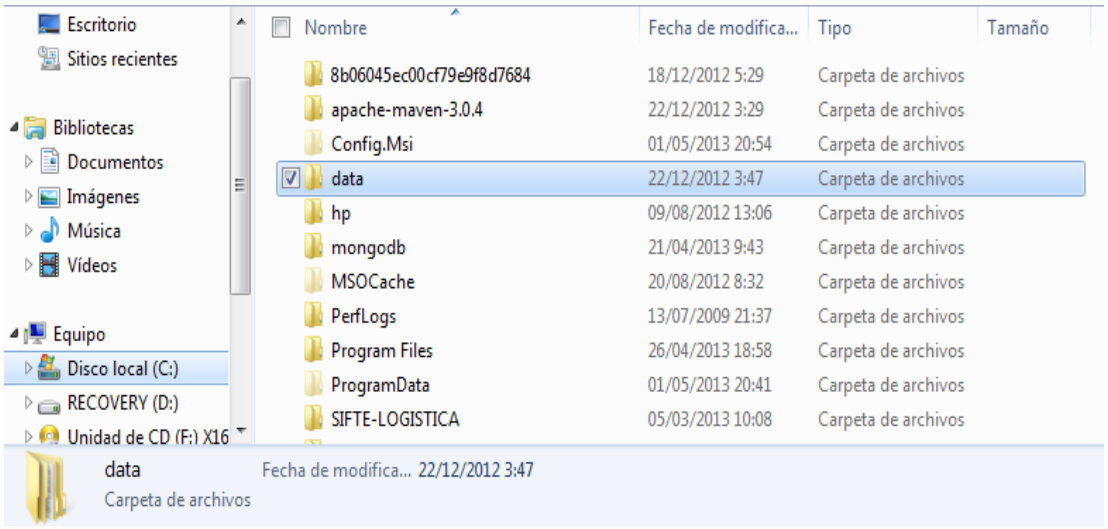


Figura 24: Instalación de MongoDB Carpeta data

✓ Dentro de la misma cree una carpeta con el nombre **db** que será la que permita guardar las bases de datos.

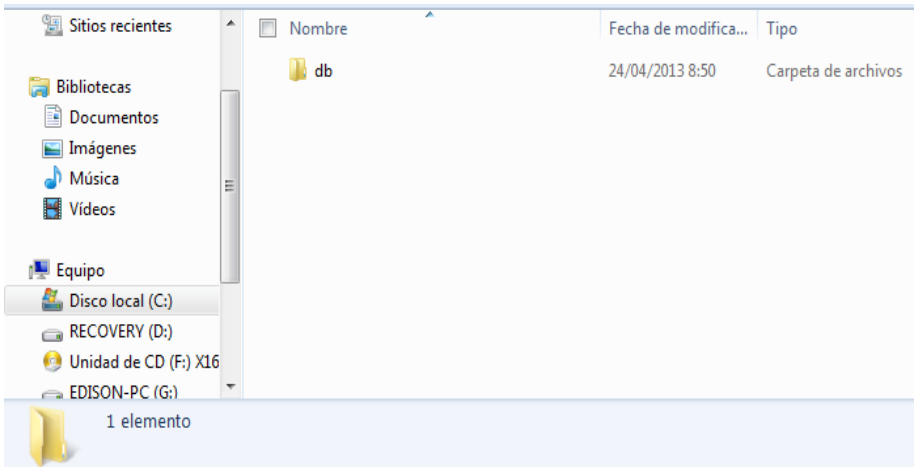


Figura 25: Instalación de MongoDB Carpeta db

✓ En el caso de que no se realice este paso al momento de ejecutar mongodb se cerrará automáticamente.

✓ Probar que la instalación funcione para esto ejecutamos Mongodb/bin/mongod.exe, si los pasos anteriores estuvieron bien generados este mensaje nos muestra que MongoDB está listo para su uso.

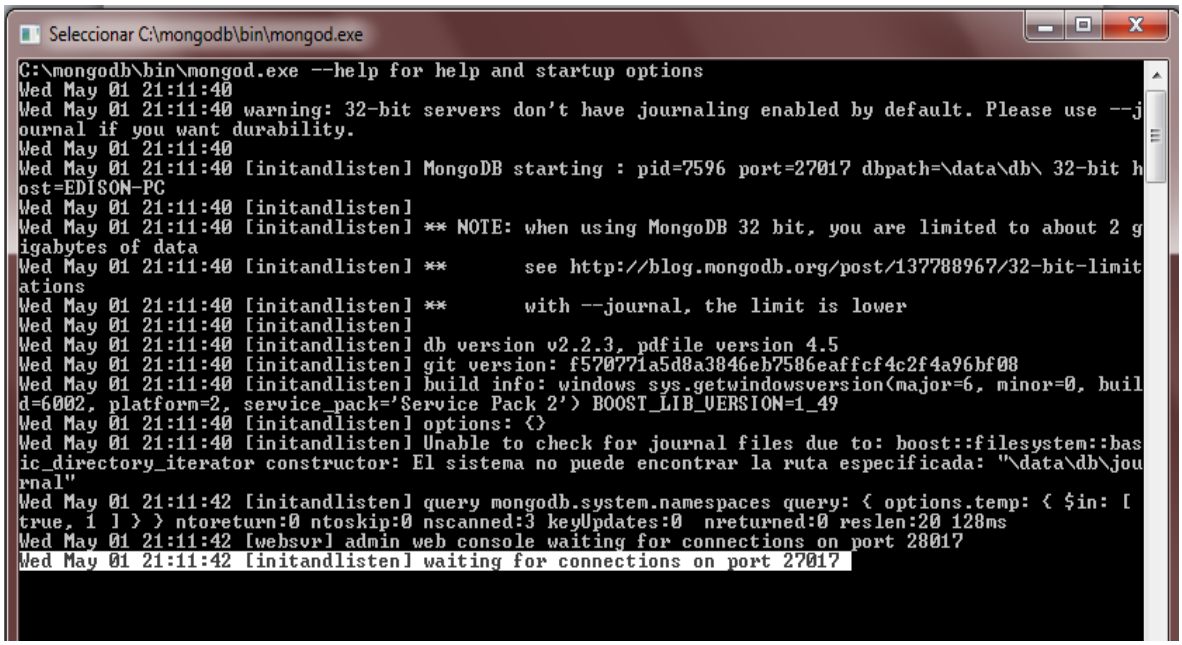


Figura 26: Comprobación del puerto de conexión MongoDB

Una vez que se haya instalado MongoDB correctamente, se debe instalar Spring Tool Suite, ya que proporciona soporte para MongoDB.

3.2 INSTALACION DE SPRING TOOL SUITE

✓ Para descargar Spring Tool Suite se debe tomar en cuenta el sistema operativo que se esté utilizando, y procedemos a la descarga desde el sitio web de Spring Source Tool Suite.

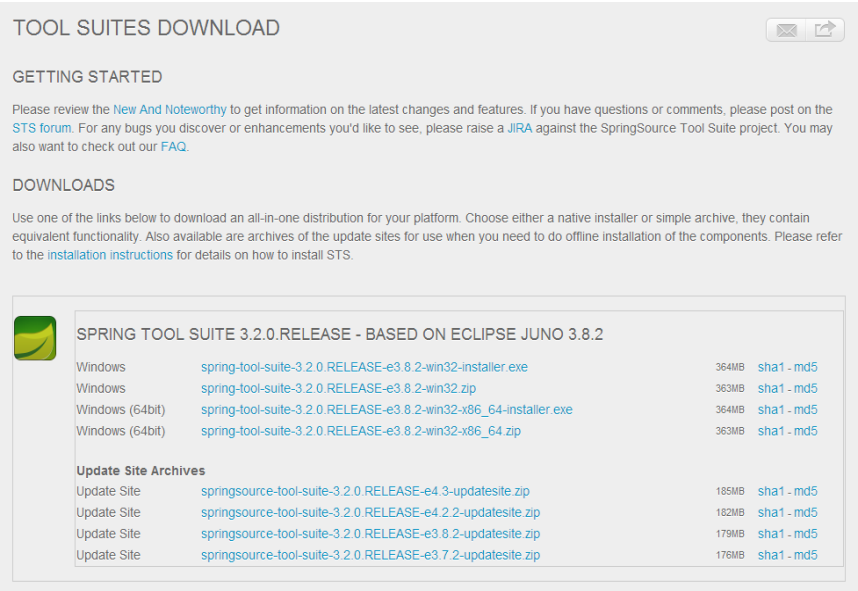


Figura 27: Descarga Spring Source Tool Suite

✓ Una descargado obtenemos el archivo spring-tool-suite-3.2.0 RELEASE-e3.8.2-win32-installer.exe, y procedemos a ejecutar.

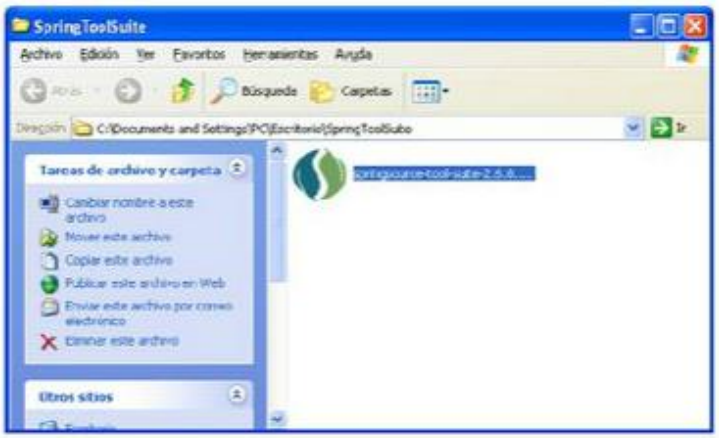


Figura 28: Spring Source Tool Suite .exe

Figura 29: Instalación de Spring Tool Suite Paso 1

✓ Damos click en Next, y se mostrará el acuerdo de licencia

Figura 30: Instalación Spring Tool Suite Paso 2

✓ El siguiente paso es elegir el directorio en donde instalaremos el Spring Source Tool Suite. Por defecto, aparecerá “C:\springsource”. Le damos click en Next.

Figura 31: Instalación de Spring Tool Suite Paso 3

✓ El instalador no solo permitirá instalar el Spring Source Tool Suite, sino también las herramientas Apache Maven, Spring ROO y Spring tc Server (Servidores basados en tomcat desarrollado por Spring Source). La decisión la tiene el usuario sin embargo lo recomendable es instalar las 4 herramientas.

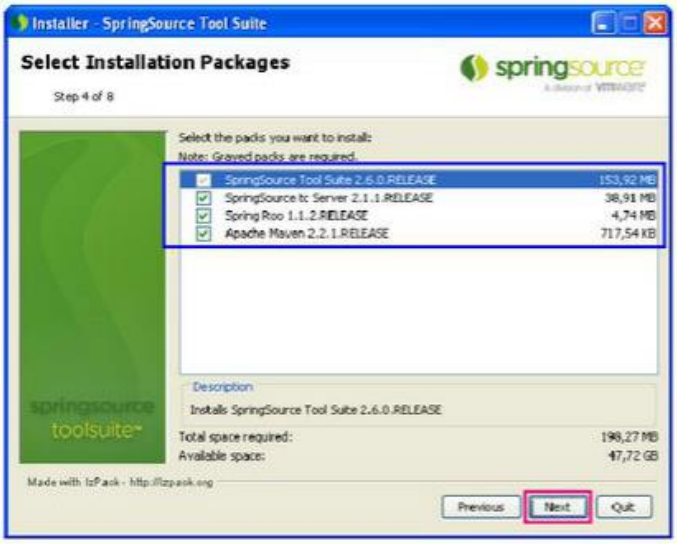


Figura 32: Instalación de Spring Tool Suite Paso 4

✓ Ahora elegimos la ruta en donde se encuentra el JDK, por lo general está en directorio C:\Archivos de programa\Java\JDK y damos click en Next,

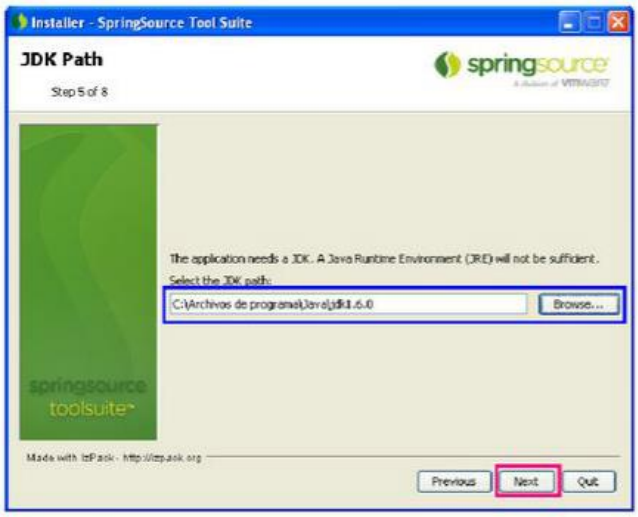


Figura 33: Instalación de Spring Tool Suite Paso 5

- ✓ Dejamos que las herramientas se instalen, esto tarda alrededor de unos 10 minutos, y una vez terminado damos click en Next,

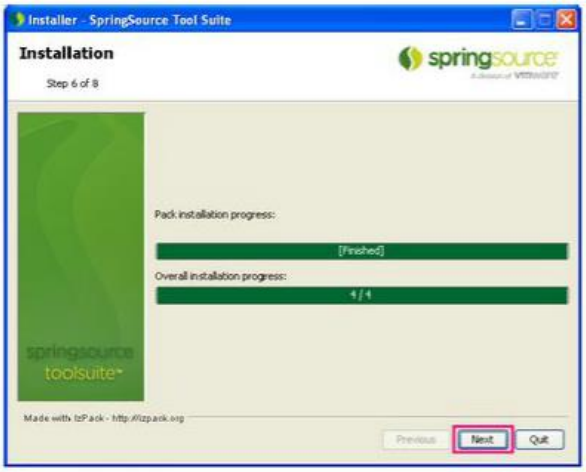


Figura 34: Instalación de Spring Tool Suite Paso 6

- ✓ Lo último que debemos hacer es finalizar la instalación,

Figura 35: Instalación de Spring Tool Suite Paso 7

Una vez instalado MongoDB que es la base de datos No Relacional y Spring Tool Suite que es el entorno de desarrollo para la construcción de aplicaciones procedemos a la creación de la base de datos.

3.3 CREACION Y SELECCIÓN DE LA BASE DE DATOS No RELACIONAL EN LA CONSOLA MONGODB

Para crear una base de datos no relacional se usa el siguiente comando:

use mongodb: crea la base de datos llamada mongodb

El comando **show dbs**muestra la Base de datos y el espacio que se está usando.

Show dbs: mongodb 0.0625GB

El comando **show collections**muestra todos los documentos (tablas) en la Base de Datos actual.

Show collections:

docVehiculo

persona

El comando **db.nombredeldocumento.insert():** se usa para insertar un documento nuevo.

Para insertar un nuevo Vehículo usamos la siguiente sintaxis:

db.docVehiculo.insert({

"Placa": "HBC-0002",

"Modelo": "SUB",

"Marca": "Ford",

"Chasis": "JHSDKHJAHDKAJSD",

"Color": "Azul",

"AnioFabricacion": "2013"

});

Para insertar una nueva Persona usamos la siguiente sintaxis:

```
db.persona.insert({
                        "Cedula": "0604052985",
                        "Nombres": "Francisco Segundo",
                        "Apellidos": "Ruiz Suarez",
        "departamentos": [
                        {"Nombre": "Departamento 1", "CargoDesempenado":
                        "Cargo1", "FuncionesRealizadas": "Funcion1"},
                        {"Nombre": "Departamento 2", "CargoDesempenado":
                        "Cargo2", "FuncionesRealizadas": "Funcion2"},
                        {"Nombre": "Departamento 3", "CargoDesempenado":
                        "Cargo3", "FuncionesRealizadas": "Funcion3"}
                        ],
        "telefonos": [
                        {"Numero": "0992264437", "Tipo": "celular"},
                        {"Numero": "0992264434", "Tipo": "convencional"},
                        {"Numero": "0992264433", "Tipo": "celular 2"}
                      ],
       "direcciones": [
                        {"Nombre": "Direccion 1"},
                        {"Nombre": "Direccion 2"},
                        {"Nombre": "Direccion 3"},
                        {"Nombre": "Direccion 4"}
                       ]
                });
```

Para eliminar documentos usamos la siguiente sintaxis:

```
db.nombre_del_documento.remove();
```

```
db.docVehiculo.remove();
```

3.4 CREACION DE DOCUMENTOS EN SPRING SOURCE TOOL SUITE

Al momento de crear un documento no es necesario establecer el tipo de datos debido a que Mongodb proporciona un completo acceso al lenguaje Java Script y a todas las funciones standars.

3.5 OPERACIONES CRUD

CRUD es el acrónimo de crear, leer, actualizar y eliminar, que son las cuatro operaciones básicas de bases de datos utilizadas en el desarrollo de aplicaciones.

Teniendo en cuenta que la implementación de la base de datos No Relacional es un prototipo usaremos el documento Vehículo para las operaciones CRUD.

3.5.1 CREAR

Para crear un nuevo vehículo editamos el siguiente código:

```
$(".listado").on("click", ".nuevoVehiculo", function(event){
        event.preventDefault();

        $.ajax({
            type: 'POST',
            url: 'vehiculo/nuevo',
            success: function(data){
                $('.nuevo').html(data);
            }
        });
});

$(".nuevo").on("click", ".guardarNuevo", function(event){
    event.preventDefault();
    $.ajax({
        type:'POST',
        url: 'vehiculo/insertar',
        data: 'placa='+ $("#placa").val() +'&modelo='+
$("#modelo").val()
            +'&marca='+ $("#marca").val() +'&chasis='+
            $("#chasis").val() +'&color='+ $("#color").val()
            +'&anio=' + $("#anioFabricacion").val(),
        beforeSend: function(){
            $(".informacion").html("Procesando...");
        },
        success: function(data){
            if (data == true){
                $(".informacion").html("Los datos se han
                guardado!");
                $(".nuevo").html("");
                listadoVehiculos();
            }
            else
            {
                $(".informacion").html("Hubo un error,
                intentelo nuevamente.");

            }

        },
        error: function(){
            $(".informacion").html("Se ha presentado un error.");
        }

    });
});
```

Todas las operaciones de inserción en MongoDB presentan las siguientes propiedades:

- ✓ El tamaño máximo de documento BSON es de 16 megabytes.

✓ Los documentos tienen las siguientes restricciones en los nombres de campo:

- El nombre del campo _id se reserva para su uso como una clave principal, y su valor debe ser único en la colección, es inmutable, y puede ser de cualquier tipo que no sea una matriz.
- Los nombres de los campos no pueden comenzar con el carácter $.
- Los nombres de campo no pueden contener el caracter (.)

3.5.2 ACTUALIZAR

Para actualizar la información de un vehículo insertado editamos el siguiente código:

```
$(".listado").on("click", ".tableVehiculos tbody tr td .editar", function(event){
        event.preventDefault();
        $.ajax({
                type: 'POST',
                url: 'vehiculo/editar',
                data: 'id=' + $(this).data("id"),
                beforeSend: function(){
$(".informacion").html("Cargando... Espere por favor.");
                },
                success: function(data){
                        $(".informacion").html("");
                        $(".divEditar").html(data);
                },
                error: function(data){
                $(".informacion").html("Se ha presentado un error.");
                }
        });
});

$(".divEditar").on("click", ".cerrarEditar", function(event){
        event.preventDefault();
        $(".divEditar").html("");
});

$(".divEditar").on("click", ".guardarEditar", function(event){
        event.preventDefault();
        $.ajax({
                type:'POST',
                url: 'vehiculo/guardar',
                data: 'id='+ $("#id").val() +'&placa='+
                $("#placa").val() +'&modelo='+ $("#modelo").val()
                +'&marca='+ $("#marca").val() +'&chasis='+ $("#chasis").val()
                +'&color='+ $("#color").val() +'&anio=' +
                $("#anioFabricacion").val(),
                beforeSend: function(){
                        $(".informacion").html("Procesando...");
                },
```

```
                success: function(data){
                        if (data == true){
                                $(".informacion").html("Los datos se han
                              guardado!");
                                $(".divEditar").html("");
                                listadoVehiculos();
                        }
                        else
                        {
                        $(".informacion").html("Hubo un error, intentelo
                        nuevamente.");
                                $(".divEditar").html("");
                        }

                        },
                        error: function(){
                        $(".informacion").html("Se ha presentado un error.");
                        }

                });


                $(".divEditar").html("");
});
```

3.5.3 ELIMINAR

Para eliminar un vehículo editamos el siguiente código:

```
$(".listado").on("click", ".tableVehiculos tbody tr td .eliminar", function(event){
        event.preventDefault();

        if (confirm("ADVERTENCIA: ¿Esta seguro eliminar este registro?")){
                $.ajax({
                        type: 'POST',
                        url: 'vehiculo/eliminar',
                        data: 'id=' + $(this).data("id"),
                        beforeSend: function(){
                                $(".informacion").html("Procesando...");
                        },
                        success: function(data){
                                if (data == true){
                                  $(".informacion").html("Los datos se han
                                  eliminado!");
                                listadoVehiculos();

                                }
                                else
                                {
                                $(".informacion").html("Hubo un error,
                                intentelo nuevamente.");

                                }

                                },
```

```
                                        error: function(){
                                          $(".informacion").html("Se ha presentado un
                                          error.");
                        }
        });
```

3.5.4 BUSCAR

Para buscar un documento usamos el siguiente código:

```
$(".listado").on("click", ".buscar", function(event){
        event.preventDefault();

        $.ajax({
                type: 'POST',
                url: 'vehiculo/buscar',
                data: 'query='  + $(".search-query").val(),
                beforeData: function(){
                        $(".informacion").html("Cargando... Espere por
favor.");
                },
                success: function(data){
                        $(".informacion").html("");
                        $(".listado").html(data);
                },
                error: function(){
                        $(".informacion").html("Se ha presentado un error.");
                }
        });
});
```

MongoDB no siempre respeta atomicidad y no define los niveles de integridad transaccional o de aislamiento durante las operaciones simultáneas.

MongoDB define las siguientes operaciones para diversas acciones:

- ✓ **$ inc**: Incrementa el valor de un campo determinado
- ✓ **$ set:** Establece el valor de un campo
- ✓ **$ push:**Añade valor a un campo
- ✓ **$ pushAll**: Anexa cada valor en una matriz a un campo
- ✓ **$ addToSe:**Agrega un valor a un array.
- ✓ **$ pop**: Elimina el último elemento de una matriz

- ✓ **$ pull:**Elimina todas las apariciones de los valores de un campo
- ✓ **$ pullAll**: Elimina todas las ocurrencias de cada valor en una matriz de un campo
- ✓ **$ rename**: Renombra un campo
- ✓ **Count():**Devuelve el número de entidades disponibles.
- ✓ **Delete (ID id):** Elimina la entidad con el id indicado.
- ✓ **Delete(Iterable<? extends T> entities):** Elimina las entidades dadas.
- ✓ **Delete(T entity):** Elimina una determinada entidad.
- ✓ **DeleteAll():**Elimina todas las entidades gestionadas por el repositorio.
- ✓ **Exists(ID id):** Devuelve si una entidad con el id dado existe.
- ✓ **findAll():**Devuelve todas las instancias de este tipo.
- ✓ **findOne (ID id):** Recupera una entidad por su clave primaria.
- ✓ **Save (Iterable<? extends T> entities):** Guarda todas las entidades dadas.
- ✓ **save(T entity):** Guarda una entidad dada.
- ✓ **findAll (Pageable pageable):** Devuelve una página de entidades que cumplen la restricción prevista en la paginación objeto paginable.
- ✓ **findAll(Sort sort):** Devuelve todas las entidades según las opciones dadas.

3.6 COMPARATIVO SQL Y MONGODB

3.6.1 CREATE

Tabla 8: Comparativo SQL y MongoDB Create

SQL	MONGODB
CREATETABLE tVehiculos (idPlaca varchar(8)notnull, txtMarca varchar(30)notnull, txtModelo varchar(50)notnull, txtChasis varchar(17)notnull, txtColor varchar(30), txtAnioFabricacion intnotnulldefaultdatepart(year,getdate()), CONSTRAINT pkVehiculo PRIMARYKEY(idPlaca), CONSTRAINT chkPlaca check(idPlaca like'[A-Z][A-Z][A-Z]-[0-9][0-9][0-9][0-9]'), CONSTRAINT chkAnio check(txtAnioFabricacion <=datepart(year,getdate())))	db.docVehiculo.insert({ "placa": "HBC-0001", "modelo": "CAMIONETA", "marca": "CHEVROLET", "chasis": "122234ABCDD", "color": "PLOMO", "Aniofabricacion": "2013" });

3.6.2 ALTER

Tabla 9: Comparativo SQL y MongoDB Alter

SQL	MONGODB
ALTER TABLE Vehiculo ADD join_date DATETIME	db.docVehiculo.update({}, { $set: {join_date: new Date() } }, {multi: true})

3.6.3 INSERT

Tabla 10: Comparativo SQL y MongoDB Insert

SQL	MONGODB
INSERT INTO Vehiculo(user_id, age, status) VALUES (“bcd001”, 45, “A”)	db.docVehiculo.insert({ user_id: “bcd001”, age: 45, status: “A” })

3.6.4 SELECT

Tabla 11: Comparativo SQL y MongoDB Select

SQL	MONGODB
SELECT * FROM Vehiculo WHERE placa = “abc001” AND Aniofabricacion = 2012	db.docVehiculo.find({ placa: “abc001”, Aniofabricacion: 2012 })

3.6.5 UPDATE

Tabla 12: Comparativo SQL y MongoDB Update

SQL	MONGODB
UPDATE Vehiculo SET placa = “abc001” WHERE Aniofabricacion > 2010	db.docVehiculo.update({ Aniofabricacion:{$gt>2010}}, {$set: {placa:”abc001”}}, {multi: true })

3.6.6 DELETE

Tabla 13: Comparativo SQL y MongoDB Delete

SQL	MONGODB
DELETE FROM Vehiculo WHERE placa = “abc001”	db.docVehiculo.remove({ placa: “abc001})
DELETE FROM Vehiculo	db.docVehiculo.remove()

4 METODOLOGÍA

4.1 TIPO DE ESTUDIO

Para el análisis comparativo de las Bases de Datos Relacionales y NoRelacionales, de la Unidad de Transportes de la UNACH, se opta por la investigación cuasi-experimental, mediante la cual, podemos obtener los resultados de una investigación en ambas situaciones en la que no es posible el control absoluto de las variables.

Para elaborar el prototipo del Sistema Informático de la Unidad de Transportes de la UNACH, basados en los principios, leyes, es de forma colectiva debido que somos dos personas que realizamos la investigación.

4.2 POBLACIÓN Y MUESTRA

Los elementos involucrados en el trabajo de investigación está constituido por las Bases de Datos Relacionales (Microsoft SQL Server) y Bases de Datos No Relacionales (MongoDB) que van hacer el motivo de nuestra investigación.

4.3 OPERACIONALIZACION DE LAS VARIABLES

Tabla 14: Operacionalización de las Variables

Variable	Tipo	Definición Conceptual	Dimensiones	Indicadores
Base de Datos Relacional	Independiente	Una base de datos relacional es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado para implementar bases de datos ya planificadas. Permiten establecer relaciones entre los datos, y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: **"Modelo Relacional"**.	Análisis	Análisis Descriptivo
Base de Datos NoRelacional	Independiente	Las bases de datos No SQL son la nueva generación de bases de datos, frecuentemente no relacionales, distribuidas, de código abierto y escalables horizontalmente. Se han convertido en un requisito principal para los empleadores que buscan desarrollar sus potencialidades en Big Data.	Análisis	Análisis Descriptivo
Eficiencia de los procesos del Sistema Informático de la Unidad de Transportes de la UNACH.	Dependiente	Se define como la capacidad de obtener información del sistema informático de la Unidad de Transportes de la Unach con la utilización mínima de recursos.	Complejidad	✓ Líneas de código ✓ Tiempo de Carga (Rendimiento) ✓ Tamaño en Disco

Los presentes indicadores son tomados directamente del resultado de implementar Bases de Datos Relacionales (Microsoft SQL Server) y Bases de Datos No Relacionales (MongoDB), a continuación se presenta el detalle de cada uno de ellos.

4.3.1 DIMENSION COMPLEJIDAD

4.3.1.1 Indicador: Líneas de Código

Se medirá las líneas de código que se usan al implementar Bases de Datos Relacionales y Bases de Datos No Relacionales.

4.3.1.2 Indicador: Tiempo de Carga (Rendimiento)

En éste indicador mediremos el tiempo de carga que tiene la Base de Datos Relacional y la Base de Datos no Relacional.

4.3.1.3 Indicador: Tamaño en Disco

Se evaluará el tamaño en Disco que usa la Bases de Datos Relacionales y Bases de Datos No Relacionales.

4.4 PROCEDIMIENTOS

- ✓ Recolección de base bibliográfica
- ✓ Recolección de información
- ✓ Estudio comparativo de las Bases de Datos relacional y NoRelacional
- ✓ Adquisición de implementos (tanto hardware y software)
- ✓ Análisis de las Bases de Datos (Base de Datos Relacional y Base de Datos No Relacional)
- ✓ Desarrollo de la Base de Datos
- ✓ Pruebas
- ✓ Análisis de resultados

4.4.1 Bases de Datos a Evaluar

- ✓ Bases de datos Relacionales (Microsoft SQL Server)
- ✓ Bases de datos No Relacionales (MongoDB)

4.4.2 Indicadores a Evaluar

- ✓ Líneas de código
- ✓ Tiempo de Carga
- ✓ Tamaño en Disco

4.4.3 Procesamiento y Análisis

La comparación se usa para determinar y cuantificar las relaciones entre dos o más variables, al observar diferentes grupos que ya sea por elección o circunstancia están expuestos a tratamientos diferentes.

Para la realización de la evaluación se ha decidido utilizar una escala cuantitativa (escala concreta o continua) para evaluar cada una de los indicadores. Siendo 5 el valor máximo y 1 el valor mínimo para el peso de cada indicador

✓ **Líneas de Código**

Escala de Líneas de Código	**Peso**
1-50	1
51-100	2
101-200	3
201-300	4
300-Adelante	5

✓ **Tiempo de Carga(Rendimiento)**(se medirá en milisegundos)

Para la medición de éste indicador se utilizó la herramienta Jmeter (ver Anexos), la misma que nos permite obtener los valores que se reflejan en la siguiente escala.

Tiempo de Carga	**Peso**
0-0.05(ms)	1
0.06-0.10(ms)	2
0.11-0.15(ms)	3
0.16-0.20(ms)	4
0.21-Adelante(ms)	5

✓ **Tamaño en Disco**

Escala de Tamaño en Disco	Peso
1-10 MB	1
11-20 MB	2
21-30 MB	3
31-40 MB	4
40-Adelante MB	5

4.4.4 Evaluación de los Indicadores

Tabla 15: Evaluación de los Indicadores

DIMENSION	INDICADOR	BASE DE DATOS RELACIONALES	BASE DE DATOS NO RELACIONALES	PESO CON BD RELACIONALES	PESO CON BD NO RELACIONALES
COMPLEJIDAD	**Líneas de Código**	318	293	5	4
	Tiempo de Carga (Rendimiento)	25	13	5	3
	Tamaño en Disco	45,1	27,3	2	3
TOTAL				**12**	**10**

5 RESULTADOS

5.1 RESULTADOS FINALES

Hasta este punto, los resultados obtenidos se basan en la investigación teórica de Bases de datos Relacionales (Microsoft SQL Server), y bases de datos No Relacionales (MongoDB) para el desarrollo del sistema informático de la Unidad de Transportes de la Unach. Además, estos resultados son producto de la aplicación de los indicadores descritos anteriormente y que han sido analizados bajo parámetros y herramientas, que permiten determinar, en forma comparativa, el desempeño de estas Bases de Datos.

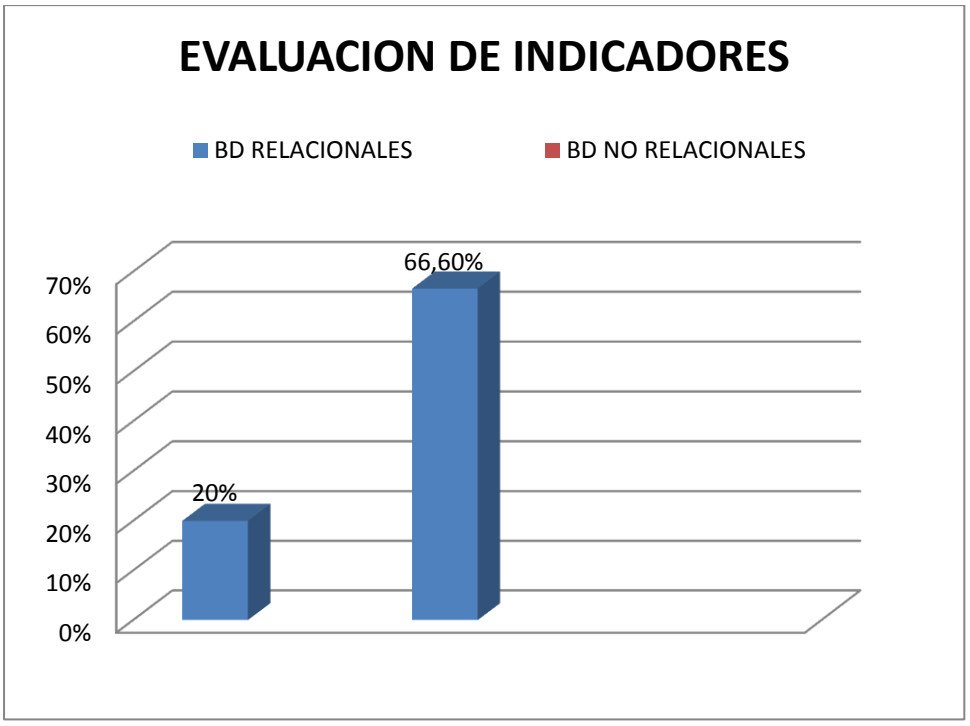


Figura 36: Comparación Base de Datos Relacionales y No Relacionales

Este estudio analítico permite determinar con certeza, que para este escenario (Unidad de Transportes de la Unach) es recomendable la implementación de una Base de Datos Relacional debido a que no existe un volumen extenso de datos, el volumen de datos crece poco a poco y las necesidades de los procesos se pueden asumir en un solo servidor. Dejando para un futuro según crezca la demanda de información la implementación de una Base de Datos No Relacional.

A continuación se presenta la comparación, para ello estableceremos los siguientes rangos:

Si	5
No	1

Tabla 16: Base de Datos SQL escenario Departamento de Transportes de la Unach

PARÁMETROS	SQL	NoSQL
Volumen de datos (crece poco a poco)	5	1
Los procesos se asumen en un solo servidor	5	1
Problemas de uso del sistemas por parte de los usuarios.	1	5
TOTAL	**11**	**7**

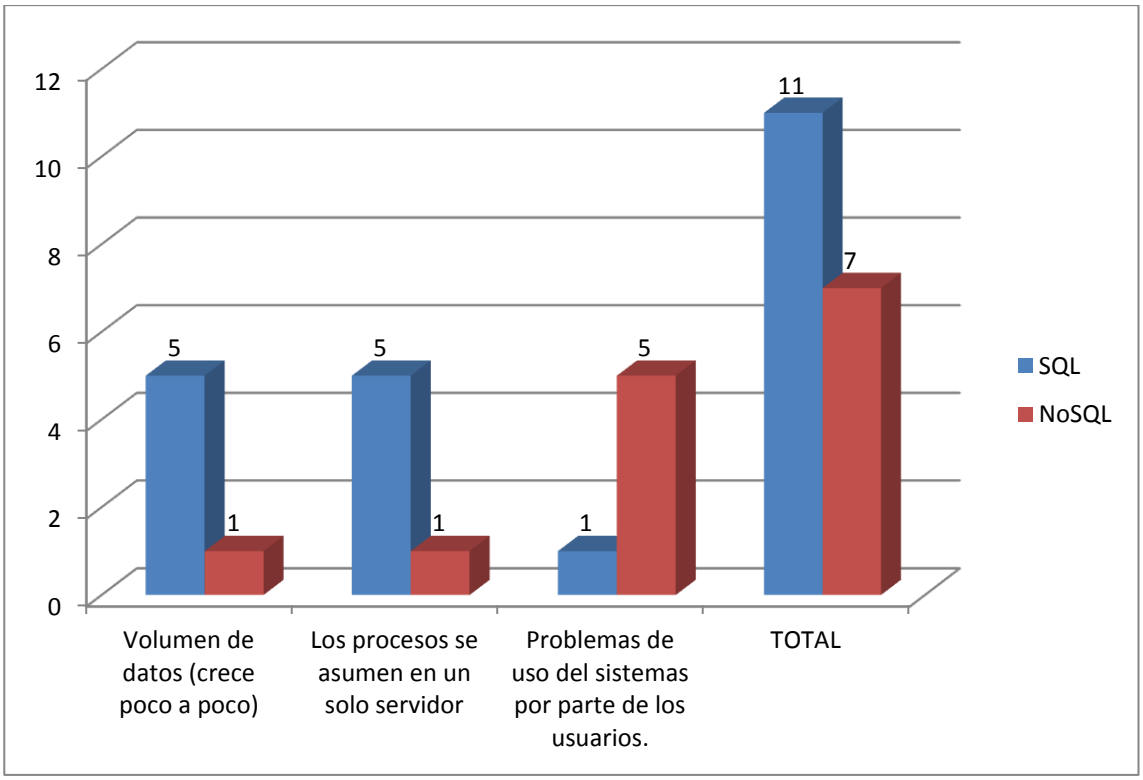


Figura 37: Parámetros SQL escenario del Departamento de Transportes de la Unach

5.2 RESUMEN DE RESULTADOS

Tabla 17: Resultados Indicadores y Parámetros

Dimensión	Indicadores	con BD Relacionales	con BD No Relacionales	Promedio con BD Relacionales	Porcentaje con BD Relacionales	Promedio con BD No Relacionales	Porcentaje con BD No Relacionales
	Líneas de Código	5	4				
	Tiempo de Carga	5	3				
	Tamaño en Disco	2	3				
Resultados				**4**	**20%**	**3,33**	**66,6%**

5 100%

4 X BD Relacionales = 20 %

BD No Relacionales = 66,6 %

5.3 ANALISIS DE RESULTADOS

Los resultados son presentados en términos de porcentajes, de la siguiente manera:

- ✓ Al aplicar Bases de Datos Relacionales en el desarrollo del sistema informático de la Unidad de Transportes de la Unach, se obtiene el 20 %
- ✓ Las Bases de Datos No Relacionales tiene un total de 66,6%.

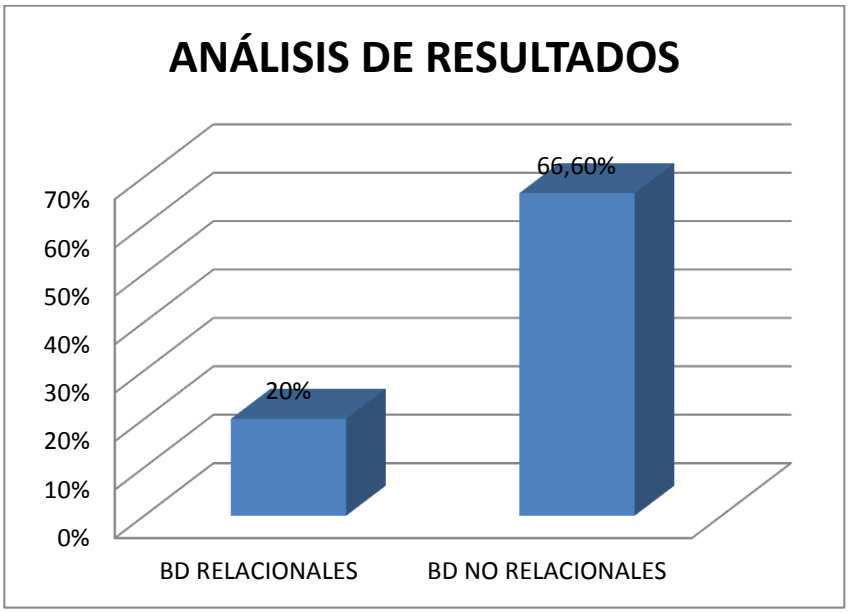


Figura 38: Análisis de Resultados

5.4 COMPROBACION DE LA HIPOTESIS

Con los datos obtenidos hasta el momento, se ha podido identificar que al usar bases de datos no relacionales optimizamos gran cantidad de recursos, teniendo en cuenta que para la Unidad de Transportes, por la cantidad de información que se maneja y el único usuario es la Sra. Secretaria del Departamento es factible el uso de una Base de datos Relacional que agilite los procesos que se realizan diariamente.

5.4.1 HIPOTESIS GENERAL

Las Bases de Datos Relacionales son más eficientes que las Bases de Datos No Relacionales en el Sistema Informático del Departamento de Transportes de la Unach.

Tabla 18: Comparativo de eficiencia

Rango de Eficiencia

RANGO	EQUIVALENCIA
0-1	Poco Eficiente
2-3	Medianamente Eficiente
4-5	Muy Eficiente

DIMENSION	INDICADORES	Peso con BD Relacionales	Eficiente	Peso con BD No Relacionales	Eficiente
COMPLEJIDAD	Líneas de Código	5	Muy Eficiente	4	Muy Eficiente
	Tiempo Medio de Inserción de Datos	5	Muy Eficiente	3	Medianamente Eficiente
	Tiempo en Disco	2	Medianamente Eficiente	3	Medianamente Eficiente
	Suma Total	**12**		**10**	
	Promedio	**20%**		**66,6%**	

5.4.2 CÁLCULOS

El modelo estadístico utilizado corresponde a la diferencia de proporciones y la prueba aplicar será la prueba chi cuadrado.

Las dos variables que usaremos son:

- ✓ Bases de Datos Relacionales
- ✓ Bases de Datos No Relacionales

Las dos variables, tienen dos categorías. Valores observados" y "Valores esperados".

- ✓ **Valores Observados (vo):** son los valores obtenidos luego de haber realizado el análisis a cada uno de los indicadores de las dos bases de datos.
- ✓ **Valores Esperados (ve):** El valor esperado para las bases de datos estimamos un valor de 5 siendo el valor máximo mencionado anteriormente.
- ✓ **Margen de error:** Al aplicar la fórmula del chi cuadrado optamos por un margen de error del 0,05 estipulado por la fórmula.

$$X^2 = \frac{\sum(vo - ve)^2}{ve}$$

ve = 5 – 0,05 = 4,95

Tabla 19: Peso de Indicadores

DIMENSION	INDICADOR	PESO CON BD RELACIONALES	PESO CON BD NO RELACIONALES
COMPLEJIDAD	Líneas de Código	5	4
	Tiempo Medio de Inserción de Datos	5	3
	Tamaño en Disco	2	3
TOTAL		12	9

Aplicamos la fórmula y los resultados son los siguientes:

Tabla 20: Resultados de la fórmula Chi Cuadrado

	BASE DE DATOS RELACIONALES	BASE DE DATOS NO RELACIONALES
Líneas de Código	0,0005	0,1823
Tiempo Medio de Inserción de Datos	0,0005	0,7682
Tamaño en Disco	1,7581	0,7682
TOTAL	**1,7591**	**1,7187**

6 DISCUSIÓN

Luego de haber aplicado la fórmula del chi cuadrado, el valor de parámetros eficientes del sistema informático de Transportes de la Unach usando Bases de Datos Relacionales supera al valor de parámetros eficientes usando Base de Datos No Relacionales se acepta la hipótesis planteada.

7 CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

- ✓ En el Departamento de Transportes de la Unach es factible la implementación de una base de datos Relacional ya que para esto se tomó en cuenta la cantidad de información generada en los procesos de mantenimiento vehicular, ordenes de movilización, provisión de combustible, procesos que no son permanentes durante el día las necesidades del proceso se puede asumir en un solo servidor.
- ✓ El estudio comparativo de los modelos de Base de Datos Relacional y No Sql, sirvió para determinar que las Bases de Datos no Relacionales son la alternativa para aplicaciones sociales y sitios web que necesiten alta escalabilidad, como por ejemplo: google. Twiter, etc (Pág.23).
- ✓ Debido al crecimiento masivo de información en aplicaciones web, se presenta problemas en el modelo relacional como son: recursos limitados, recursos síncronos, límite de escalabilidad, modelo de datos rígido, mayor tiempo para el desarrollo no existen muchos Framework, mayores Recursos, por ello hoy en día se ha buscado una solución la cual es el Modelo de Base de Datos No Sql, son una buena alternativa para el procesamiento de grandes volúmenes de datos, sin embargo éstas no sustituyen a las demás formas de almacenamiento. Los Modelos de Bases de Datos No Sql tienen características importantes como: la escalabilidad horizontal, flexibilidad en su modelo de datos, alta disponibilidad en el servicio, tolerancia a fallos.
- ✓ En la investigación se planteó como hipótesis: Las Bases de Datos Relacionales son más eficientes que las Bases de Datos No Relacionales en el Sistema Informático del Departamento de Transportes de la Unach, aplicando criterios de evaluación como: Líneas de Código, Tiempo de carga de datos y tamaño en disco. Se obtuvieron los siguientes resultados 1.7591 para las Bases de datos Relacionales y 1.7187 para las Bases de Datos No Relacionales, de lo cual se puede concluir que las Bases de Datos Relacionales tienen un valor ponderado superior lo cual evidencia que es la mejor alternativa para aplicar al proyecto del Departamento de transportes de la UNACH.

7.2 RECOMENDACIONES

✓ Es necesario que la Universidad Nacional de Chimborazo actualice la bibliografía para que sirva de apoyo técnico a los posteriores temas de investigación.

✓ Es necesario realizar un estudio de las necesidades de soluciones informáticas en cada departamento de la Universidad Nacional de Chimborazo y su respectiva integración considerando criterios como: cantidad de información que se maneja a diario, número de usuarios que interactúan, requerimientos básicos de Hardware y Software, además es fundamental contar con recursos financieros y humanos.

✓ Se recomienda que a la culminación de un sistema informático, se capacite a los usuarios que van a interactuar con el mismo, en los requerimientos básicos de hardware y software y en el uso en sí del sistema, para ello se deberá adjuntar manual de usuario y manual técnico para así aprovechar al máximo la funcionalidad del mismo.

✓ Los criterios a considerar para la selección de una base de datos son los siguientes: volumen de datos, escalabilidad, modelado de datos, cantidad de recursos, cantidad de usuarios, entre otros.

✓ Se recomienda investigar el uso de Base de Datos No Sql para Data Warehouse y Data Mining.

✓ Es importante para una investigación futura el desarrollo de aplicaciones móviles utilizando las distintas bases de Datos No Sql e implementar en el sistema informático de la Unidad de Transportes.

8 PROPUESTA

8.1 TÍTULO DE LA PROPUESTA

Modelo de Base de Datos No Relacional para EL Sistema Informático del Departamento de Transportes de la UNACH.

8.2 INTRODUCCIÓN

Este capítulo detalla el modelo de Base de Datos No Relacional para el departamento de Transportes de la UNACH, el cual ha sido elaborado tomando en cuenta las necesidades presentes en la Unidad de Transportes.

Con el transcurso del tiempo, se ha ido incrementando la importancia de contar con información confiable, íntegra y oportuna para lograr los objetivos estratégicos del departamento.

Hoy en día las Instituciones de educación superior en el Ecuador requieren automatizar mediante Sistemas Informáticos sus departamentos, con el fin de mejorar los procesos académicos y administrativos; y así lograremos la optimización de los recursos. En la Unidad de Transportes de la UNACH es necesario la existencia de una comunicación interna entre el departamento, además es importante que el flujo de información sea cada vez más rápido con el fin de identificar los problemas en el menor tiempo posible y tomar decisiones oportunas para resolverlos.

En este sentido el desarrollo de un prototipo de una Base de Datos NoRelacional para el departamento de transportes de la Universidad Nacional de Chimborazo es primordial con el fin de conseguir nuestro propósito, el mismo que es la comparación de los dos modelos de Bases De Datos, como son: el Modelo de Base de datos Relacional que ya está implantado y el prototipo creado.

8.3 OBJETIVOS

8.3.1 OBJETIVO GENERAL

- ✓ Mejorar la funcionalidad de la Unidad de Transportes, reduciendo la complejidad en la implementación y proveyendo al sistema de escalabilidad, reusabilidad, seguridad, y facilidad de uso y aprendizaje.

8.3.2 OBJETIVOS ESPECÍFICOS

- ✓ Escoger el Modelo de Base de Datos.
- ✓ Escoger los Documentos (tablas) que se van a usar.
- ✓ Diseñar un prototipo para realizar una Base de Datos NoRelacional.
- ✓ Diseñar interfaces sencillas de uso para el usuario.

8.4 FUNDAMENTACIÓN CIENTÍFICO –TÉCNICA

Contendrá una versión resumida y actualizada del estado del conocimiento en que se encuentra el tema específico de la propuesta.

8.5 DESCRIPCIÓN DE LA PROPUESTA

8.5.1 ANÁLISIS DE REQUISITOS

El análisis de cada uno de los requerimientos relacionados con la Base de Datos No Relacional, Lo que a continuación se detalla es un resumen del análisis de cada uno de los módulos del sistema y cómo cada requerimiento encaja en los módulos a desarrollarse.

Modelo de Base de Datos No Relacional aplicado en el Sistema del Departamento de Transportes de la UNACH

Con la implementación tiene por finalidad automatizar la administración de todos y cada uno de los procesos que se realiza diariamente en el Departamento de Transportes. Este control se está realizando con un sistema informático básico, tales como:

Información Administrativa

- ✓ Director del Departamento.
- ✓ Secretaria de la Departamento.

Información Vehicular

- ✓ Revisión Vehicular
- ✓ Provisión de Combustible
- ✓ Datos de los Vehículos
- ✓ Mantenimiento Vehicular

Información de Movilización:

- ✓ Órdenes de Movilización (Dentro y Fuera de la ciudad)

- ✓ Acta Entrega Recepción de los Vehículos

Información de Recursos Humanos:

- ✓ Datos personales de Choferes
- ✓ Datos Personales de Funcionarios
- ✓ Cargo
- ✓ Departamento al que pertenece

En el siguiente gráfico podremos ver cuáles son los casos de uso que intervienen y quiénes son los actores.

Modelo de Caso de Uso

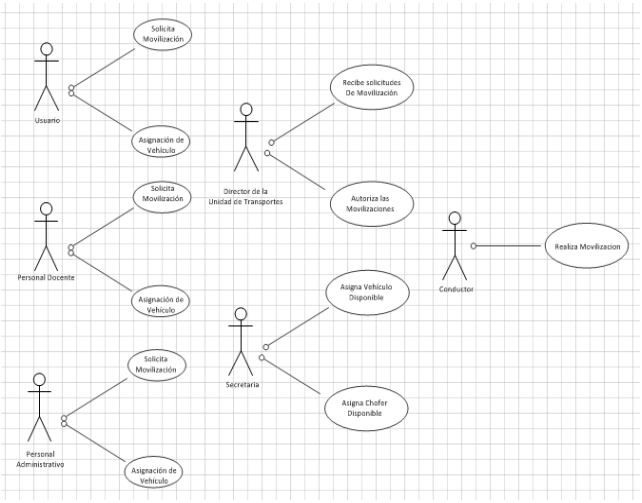


Figura 39: Propuesta Modelo de Caso de Uso

8.5.2 VENTAJA DE LA UTILIZACION DEL MODELO DE BASE DE DATOS No RELACIONAL

La ventaja de utilizar un modelo de Base de Datos No Relacional, es que en el Modelado de Datos se usan documentos (tablas en SQL), el cual nos ayuda a integrar la información llegando a estar en la cuarta forma normal, evitando así las tablas intermedias, llegando a reducir las tablas del Modelo Entidad Relación de la Base de Datos Relacional. Quedando de la siguiente manera:

MODELO DE BASE DE DATOS RELACIONAL (SQL)

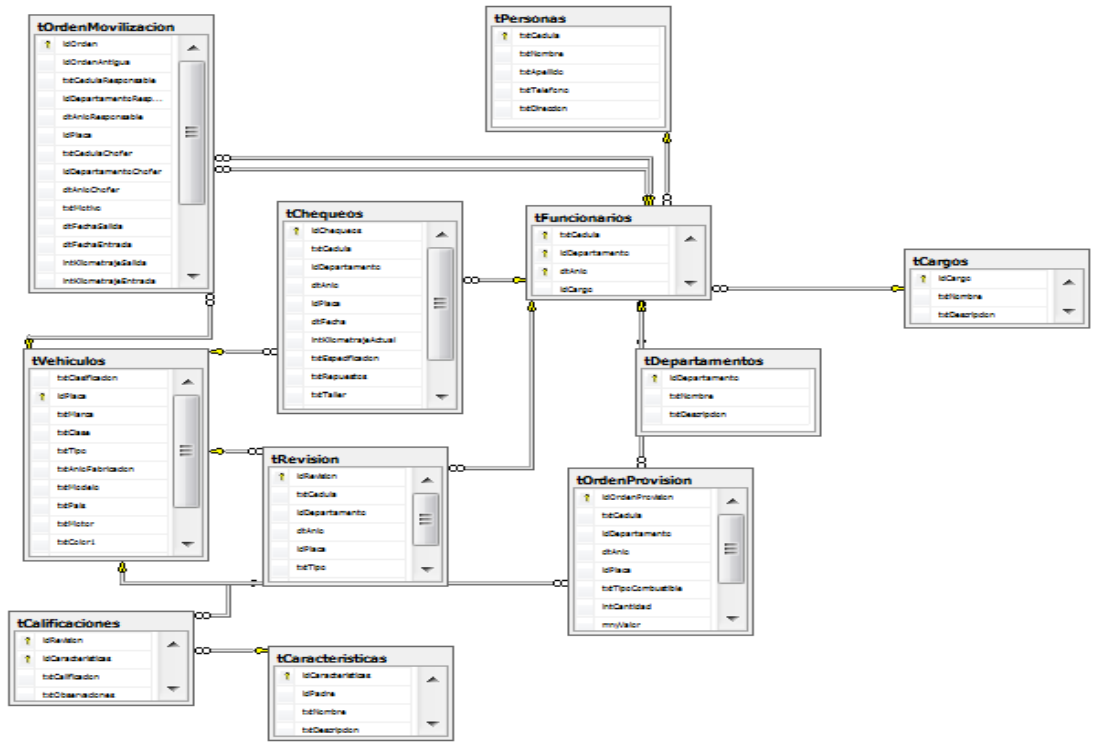


Figura 40: Propuesta Modelo de Base de Datos Relacional

MODELO DE DATOS EN No SQL

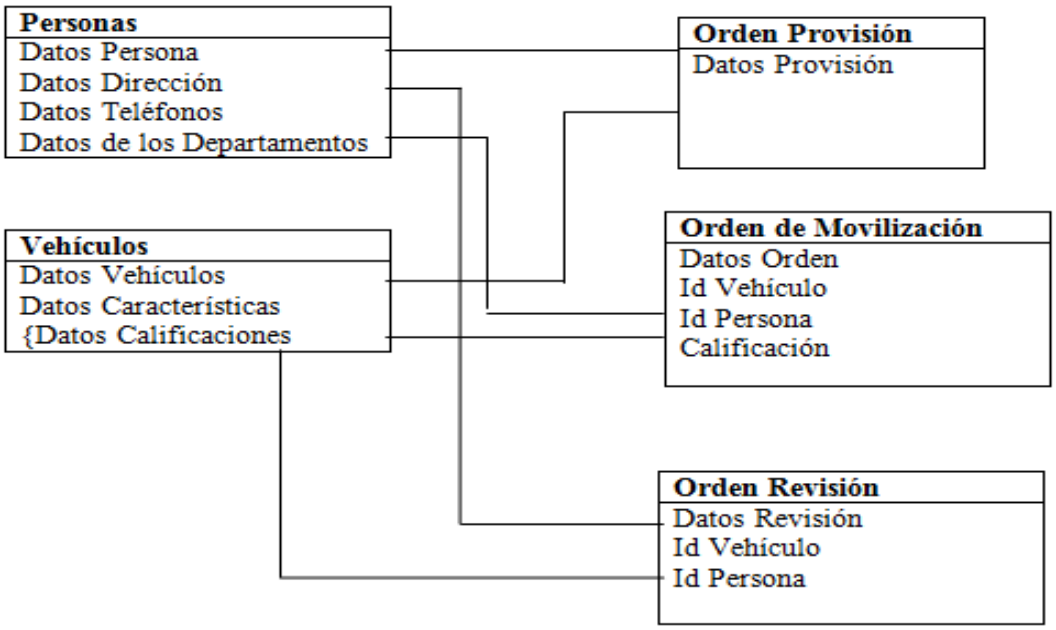


Figura 41: Propuesta Modelo de Base No SQL

8.6 ESTRUCTURA DE LAS APLICACIONES ORIENTADAS A OBJETOS

8.6.1 EL PATRÓN MODELO-VISTA-CONTROLADOR (MVC)

El modelo es el responsable de:

- ✓ Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- ✓ Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- ✓ Lleva un registro de las vistas y controladores del sistema.
- ✓ Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo. (por ejemplo, una inserción).

El controlador es el responsable de:

- ✓ Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- ✓ Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()".

Las vistas son responsables de:

- ✓ Recibir datos del modelo y los muestra al usuario.
- ✓ Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- ✓ Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Qué Ventajas trae utilizar el MVC

- ✓ Es posible tener diferentes vistas para un mismo modelo (eg. representación de un conjunto de datos como una tabla o como un diagrama de barras).
- ✓ Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- ✓ Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

- **Flujo que sigue el control en una implementación general de un MVC**

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

- ✓ El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)
- ✓ El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
- ✓ El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- ✓ El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra
- ✓ La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

8.6.2 CAPA DE ACCESO A DATOS

Con el diseño del modelo de objetos de acceso a datos (DAO) para construir una capa de acceso a datos (DAL)

Una capa de acceso de datos puede ser una parte importante de una aplicación de software. Las aplicaciones de negocios casi siempre necesitan tener acceso a los datos de bases de datos relacionales u objeto y la plataforma Java ofrece muchas técnicas para acceder a estos datos, independientemente de si se utiliza una capa de acceso a datos. La técnica más antigua, y más madura y fiable, es utilizar la conectividad de base de datos de Java - API JDBC, que proporciona la capacidad de ejecutar consultas SQL contra una base de datos y, a continuación buscar a los resultados, una columna a la vez. Aunque esta API proporciona todo lo que un desarrollador necesita para acceder a los datos y para mantener el estado de aplicación, Java Enterprise Edition (JEE) ofrece un marco de persistencia más reciente en forma de beans de entidad, un subconjunto del marco de Empresa JavaBean (EJB). Aunque ha habido muchas mejoras en la más reciente especificación EJB 2.0, muchos desarrolladores están buscando ahora a los marcos de persistencia alternativas, tales como Java Persistence API (JPA).

8.6.3 DAO (Data Access Object)

DAO consiste básicamente en una clase que es la que interactúa con la base de datos. Los métodos de esta clase dependen de la aplicación y de lo que queramos hacer. Pero generalmente se implementan los métodos CRUD para realizar las "4 operaciones básicas" de una base de datos.

8.6.4 DISEÑO

8.6.4.1 Diseño de la interfaz

8.6.4.1.1 Pantalla principal

Figura 42: Pantalla Principal

Hasta el momento de editar este documento el sistema cuenta con los siguientes elementos:

- ✓ Banner
- ✓ Menú Principal

8.6.4.1.2 Banner

Figura 43: Banner

El banner identifica el nombre del sitio web utilizado para el sistema.

8.6.4.1.3 Menú Principal

Figura 44: Menú Principal

En el menú principal va el listado de los procesos que se realizan en el departamento de transportes tales como:

- ✓ Funcionarios
  - o Datos de los Conductores
  - o Datos de uncionarios
  - o Cargo
  - o Departamento al que pertenecen
- ✓ Órdenes de Movilización
  - o Dentro y Fuera de la ciudad (Acta entrega - recepción)
  - o Vehículos
  - o Datos de vehículos
  - o Revisión vehicular

- ✓ Provisión de combustible
  - Datos de vehículos
  - Mantenimiento Vehicular

Funcionarios:

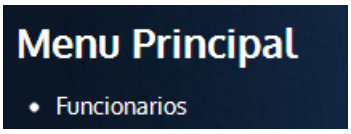


Figura 45: Pantalla Funcionarios

Se direcciona a la pantalla de funcionarios en la podemos realizar las siguientes acciones:

Nuevo:

Figura 46: Ingresar Nuevo Funcionario

Permite ingresar datos de un nuevo funcionario tales como:

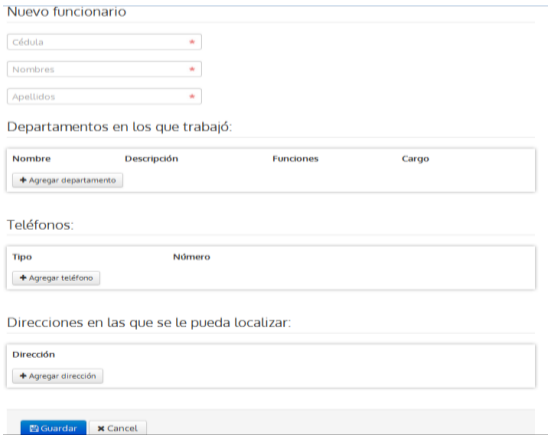


Figura 47: Ingresar Datos de un Funcionario

- ✓ Nombres
- ✓ Apellidos
- ✓ Número de Cédula
- ✓ Departamentos en los que trabaja
- ✓ Teléfonos
- ✓ Direcciones en las que se la puede ubicar

Agregar Departamentos:

Al hacer click en la opción agregar departamentos se despliega la siguiente pantalla:

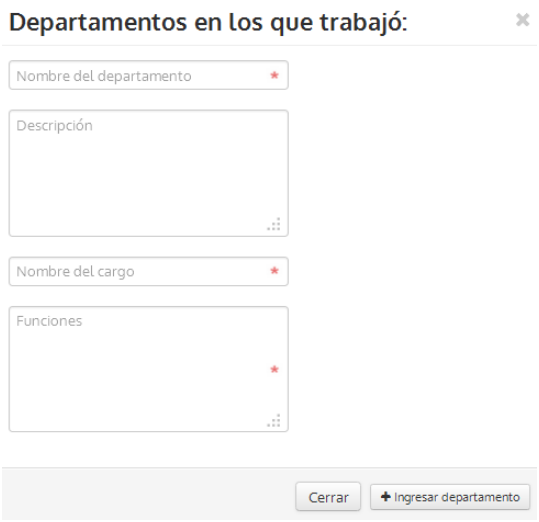


Figura 48: Agregar Departamento

Permite ingresar el departamento y el cargo que desempeña el funcionario.

Teléfonos:

Al hacer click en la opción teléfonos se despliega la siguiente pantalla:

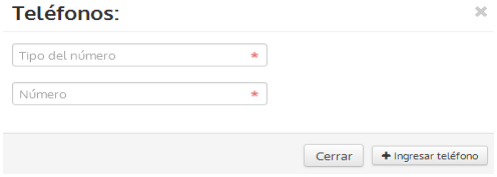


Figura 49: Ingresar Números de Teléfono

Permite ingresar los números de referencia tanto convencional como celular.

Direcciones:

Al hacer click en la opción direcciones se despliega la siguiente pantalla:

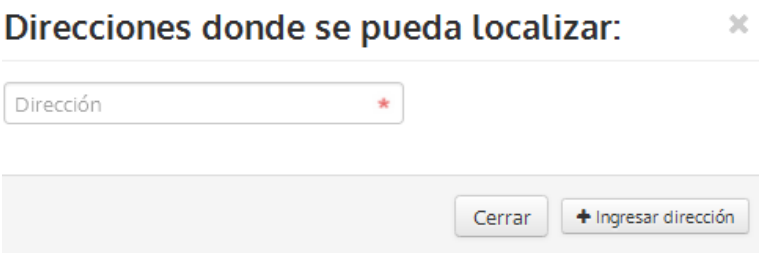


Figura 50: Agregar Direcciones de un Funcionario

El sistema cuenta con una opción de buscar, el cual busca información en la base de datos.

Figura 51: Opción de Búsqueda

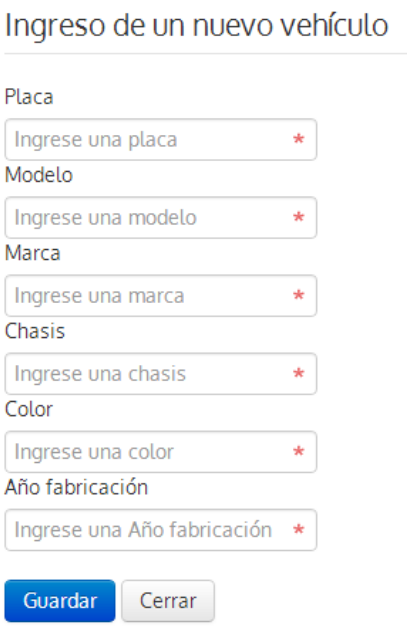
Además en la pantalla principal toda la información tiene las opciones de ingresar, modificar y eliminar.

Figura 52: Ingresar, Modificar, Eliminar

En cada uno de los formularios a llenar los campos con asterisco rojo significa que es un campo obligatorio de llenar.

Vehículos

Se direcciona a la pantalla de vehículos en la que podemos ingresar información de cada vehículo:



Y al igual que en funcionarios toda la información tiene las opciones de ingresar, modificar y eliminar.

9 BIBLIOGRAFÍA

- ✓ Cogoluègnes, A., Templier, T., & Piper, A. (2011). *Spring Dynamics Modules in Action.* United States of America: Manning Publications Co.
- ✓ Corporation, O. (3 de Marzo de 2012). *Project Jigsaw*. Recuperado el 3 de Marzo de 2012, de Open JDK: http://goo.gl/5UJxJ
- ✓ Eclipse. (2010). *Guía de programación de Virgo*. Recuperado el 11 de 5 de 2012, de http://goo.gl/GgEyh
- ✓ Knoernschild, K. (2012). *Java Application Architecture: Modularity Patterns with Examples Using OSGi.* United States of America: Prentice Hall.
- ✓ Martin, R. C. (Diciembre de 1996). *Granularidad*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/dZZEh
- ✓ Martin, R. C. (Mayo de 1996). *Principio de Inversión de Dependencias*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/SA8r9
- ✓ Martin, R. C. (Marzo de 1996). *Principios de la Orientación a Objetos*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/oqExW
- ✓ Moreno, F. G. (01 de Enero de 2011). *Desarrollando una aplicación Spring Framework MVC v3 + JPA paso a paso*. Recuperado el 30 de Enero de 2012, de http://goo.gl/cMS5V
- ✓ Oracle. (1999). *JAR File Specification*. (Oracle) Recuperado el 26 de Marzo de 2012, de JAR File Specification: http://goo.gl/QYbm0
- ✓ Paez, N. (2010). *Utilización de programación orientada a aspectos en aplicaciones empresariales.* Buenos Aires: Anónimo.
- ✓ Palao, D. M. (1 de Febrero de 2010). *Desarrollando una aplicacion Spring Framework MVC paso a paso*. Recuperado el 30 de Enero de 2012, de http://goo.gl/XRFk4

- ✓ Ramos, A. (23 de Febrero de 2010). *Modularización Efectiva en Java*. Recuperado el 19 de Febrero de 2012, de Modularización Efectiva en Java: http://goo.gl/yQIKw
- ✓ Risberg, T., Evans, R., & Tung, P. (1 de Enero de 2010). *Developing a Spring Framework MVC application step-by-step*. Recuperado el 30 de Enero de 2012, de http://goo.gl/kMWUs
- ✓ Völter, M. (10 de Marzo de 2005). *Software Architecture Patterns*. Recuperado el 3 de Marzo de 2012, de A pattern language for building sustainable software architectures: http://goo.gl/jYVHv
- ✓ Walls, C. (2010). *Modular Java, Creating flexible applications with OSGi and Spring.*Dallas, Texas: The Pragmatic Bookshelf
- ✓ Wikilearning. (12 de Enero de 2010). *DEFINICIÓN Y TERMINOLOGÍA DE UN RDBMS*. Recuperado el 6 de Febrero de 2012, de http://goo.gl/CtKl6
- ✓ Pressman*Ingeniería de Software*
- ✓ Oreilly. (12 de Octubre de 2012). *Spring Data*
- ✓ Wrox Professional NoSQL . (Agosto de 2011). *No SQL Profesional.*

10 APÉNDICES O ANEXOS

GLOSARIO

- ✓ **ACTA:** Documento en el que se exponen los trabajos de carácter técnico.
- ✓ **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.
- ✓ **ANSI**: (American National Standards Institute) Instituto Nacional Estadounidense de Estándares es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.
- ✓ **API**: (Application Programming Interface) Interfaz de programación de aplicaciones. Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- ✓ **API JDBC**: Es una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos
- ✓ **Arrays:** Es un medio de guardar un conjunto de objetos de la misma clase. Se accede a cada elemento individual del array mediante un número entero denominado índice.
- ✓ **Background:** Se utiliza para nombrar a todos aquellos procesos o rutinas de ejecución que se realizan en *segundo plano*. Esto implica que el proceso se está llevando a cabo con una prioridad baja y no siempre tiene la CPU (Unidad central de procesamiento) de forma secuencial ejecutando su código.
- ✓ **Big Table:** Es un sistema de gestión de base de datos creado por Google con las características de ser: distribuido, de alta eficiencia y propietario.
- ✓ **Buffers:** Es una ubicación de la memoria en un Disco o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.
- ✓ **Business Intelligence:** Da la información que necesita, cuando lo necesite, en el formato adecuado. Mediante la integración de los datos de toda la empresa y la entrega de informes y análisis de autoservicio, que gasta menos tiempo en responder a las peticiones y usuarios de negocios gastan menos tiempo buscando información

- ✓ **Callback:** es un código ejecutable que es pasado cómo un argumento a otro código. Esto permite al software de la capa de bajo nivel llamar a una subrutina (función) definida en una capa de alto nivel.
- ✓ **Cargo:**El cargo se compone de todas las actividades desempeñadas por una persona, las cuales pueden incluirse en un todo unificado que ocupa una posición formal en el organigrama.
- ✓ **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.
- ✓ **Comet.** En el desarrollo web, Comet es un neologismo para describir un modelo de aplicación web en el que una petición HTTP mantenida abierta permite a un servidor web enviar datos a un navegador por Tecnología Push, sin que el navegador los solicite explícitamente.
- ✓ **CRUD.** Es el acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: **C**reate, **R**ead, **U**pdate and **D**elete). Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.
- ✓ **CVS.** Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones.
- ✓ **Data Mining.** Es el proceso de cálculo de descubrir patrones en grandes conjuntos de datos involucra métodos en la intersección de la inteligencia artificial , aprendizaje automático , las estadísticas y los sistemas de bases de datos.
- ✓ **DB**: Base de Datos.
- ✓ **DAO.** Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.
- ✓ **Departamento:**Es una entidad que forma parte de una empresa.
- ✓ **Dispatcher Servlet.** Despachador central para la solicitud HTTP manipuladores / controladores, por ejemplo, los controladores de la interfaz de usuario web o exportadores de servicios remotos basados en HTTP. Despachos a los controladores registrados para el procesamiento de una solicitud web, proporcionando mapas cómodos e instalaciones de manejo de excepciones.

- ✓ **Down time.** El término tiempo de inactividad se usa para referirse a períodos en los que un sistema no está disponible.
- ✓ **Empresa JavaBean (EJB):** Un "Java Bean" es un componente utilizado en Java que permite agrupar funcionalidades para formar parte de una aplicación, esto puede ser: un "Java Bean" agrupando información personal, datos sobre un pedimento, requerimientos de ordenes
- ✓ **Erlang.** Es un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual y biblioteca.
- ✓ **FlockDB.** Es un código abierto distribuido de alta disponibilidad de base de datos gráficapara la gestión de datos en webscale.
- ✓ **Fork.** Una bifurcación o fork, cuando se aplica en el contexto de un lenguaje de programación o un sistema operativo, hace referencia a la creación de una copia de sí mismo por parte de un programa, que entonces actúa como un "proceso hijo" del proceso originario, ahora llamado "padre". Los procesos resultantes son idénticos.
- ✓ **Framework.** Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.
- ✓ **Freemarker.** Es un "motor de plantillas", una herramienta genérica para generar la salida de texto (nada de HTML de código fuente generado automáticamente) basado en plantillas. Es un paquete de Java, una biblioteca de clases para los programadores de Java. No es una aplicación para los usuarios finales en sí mismo, sino algo que los programadores pueden incrustar en sus productos.
- ✓ **Funcionario:**Es aquel trabajador que desempeña funciones en un organismo, ya sea el legislativo, el ejecutivo o el judicial.
- ✓ **GridFS.** En lugar de almacenar un archivo en un único documento, GridFS divide un archivo en partes.
- ✓ **Hadoop.** Es un framework de software que soporta aplicaciones distribuidasbajo una licencia libre.
- ✓ **Handler**: Es el puente que hay entre un hilo secundario (thread) y el hilo principal (aplicación).
- ✓ **HDFS:** HDFS es el sistema de archivos de HBase.

- ✓ **Hibernate.** Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo deobjetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en losbeans de las entidades que permiten establecer estas relaciones.
- ✓ **HTTP**. Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia dehipertexto) es el protocolo usado en cada transacción de la World Wide Web.
- ✓ **HTTPS**: (Hyper Text Transfer Protocol Secure) Protocolo seguro de transferencia de hipertexto, es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hiper Texto, es decir, es la versión segura de HTTP. Describe cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP.
- ✓ **IBM.** Es una empresamultinacional estadounidense de tecnología y consultoría.
- ✓ **Indices BTree.** Es una estructura de datos de árbol que impide que los datos ordenados y permite búsquedas, el acceso secuencial, inserciones y elecciones en tiempo logarítmico.
- ✓ **Inktomi.** Fue una compañía de California que proporcionaba software para proveedores de servicios de Internet.
- ✓ **JavaBeans.** Son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.
- ✓ Se usan para encapsular varios objetos en un único objeto (la vaina o Bean en inglés), para hacer uso de un solo objeto en lugar de varios más simples.
- ✓ **Java Enterprise Edition (JEE):** es una plataforma de programación parte de la Plataforma Javapara desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.
- ✓ **JAXP**: (Java Api for XML Processing) API de Java que sirve para la manipulación y el tratamiento de archivos XML.

- ✓ **JDBC.** JDBC es un API incluido dentro del lenguaje Java para el acceso a bases de datos. Consiste en un conjunto de clases e interfaces escritos en Java que ofrecen un completo API para la programación de bases de datos.
- ✓ **JDK:** Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red.
- ✓ **JDO.** El Objects (JDO) API de datos de Java es una interfaz basada en Java modelo de abstracción nivel de persistencia, desarrollado bajo los auspicios de la Java Community Process. . Una de sus características es la transparencia de los servicios de persistencia del modelo de dominio . JDO objetos persistentes son ordinarias del lenguaje de programación Java de clases.
- ✓ **JOINS.** Es una combinación de dos o más tablas de una base de datos relacional, es una instrucción de lo más imprescindible si queremos realizar una aplicación que realice un uso correcto e intensivo de un gestor de Bases de datos, podemos leer datos de diferentes bases de datos en una única consulta. Lo que nos permitirá diseñar fácilmente tablas relacionadas entre ellas
- ✓ **JPA.** Más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE.
- ✓ Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard.
- ✓ **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.
- ✓ **JSP.** JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML entre otros tipos de documentos. JSP es similar a PHP pero usa el lenguaje de programación Java.
- ✓ **KFS:** Sistema de archivos distribuido Kosmos (KFS) proporciona un alto rendimiento combinado con la disponibilidad y la fiabilidad.
- ✓ **Licencia BSD.** La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Es unalicencia de software

libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras.

- ✓ **Lucene.** Es una API de código abierto para recuperación de información, originalmente implementada en Java por Doug Cutting.
- ✓ **Mantenimiento:**Son todas aquellas reparaciones que se le realicen a un vehículo automotor con el propósito de prevenir en un futuro fallas del sistema que provoquen un mal funcionamiento.
- ✓ **MapReduce:** Modelo de programación utilizado por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en diversos ordenadores.
- ✓ **Movilización:**Proceso de transportar a una o varias personas de un lugar a otro.
- ✓ **MVC.** El Modelo Vista Controlador (MVC) es un patrón de arquitectura de softwareque separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.
- ✓ **Neologismo.** Puede definirse como una palabra nueva que aparece en una lengua, o la inclusión de un significado nuevo en una palabra ya existente o en una palabra procedente de otra lengua.
- ✓ **NoSQL.** Generación de bases de datos siguientes sobre todo frente a algunos de los puntos: el ser no relacional, distribuido, de código abierto **y** escalable horizontalmente.
- ✓ **OLAP (On Line Analitical Processing).** Es el acrónimo en inglés de procesamiento analítico en línea (On-Line Analytical Processing). Es una solución utilizada en el campo de la llamada Inteligencia empresarial (o Business Intelligence) cuyo objetivo es agilizar la consulta de grandes cantidades de datos.
- ✓ **Open Source.** Código abierto (o fuente abierta) es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de poder acceder al código, que a las cuestiones éticas y morales las cuales se destacan en el software libre.
- ✓ **ORM**: Mapeo Objeto-Relacional (Object-Relational Mapping). Los ORM son herramientas de software que permiten trabajar con los datos persistidos en una base de datos relacional como si fuera parte de una base de datos orientada a objetos. La función del ORM es transformar un registro en objeto y viceversa.

- ✓ **Peer to peer.** 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.
- ✓ **PID:** Es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso.
- ✓ **POJO:** El término "POJO" se utiliza principalmente para referirse a un objeto Java que no sigue ninguna de las principales modelos de objetos de Java, convenciones, o marcos.
- ✓ **POM.** Un modelo de objetos de Proyecto o POM es la unidad fundamental de la labor de Maven. Es un archivo XML que contiene información sobre el proyecto y los detalles de configuración utilizados por Maven para construir el proyecto.
- ✓ **Proto-buffers.** Del idioma neutral independiente de plataforma mecanismo de Google, y extensible para serializar datos estructurados pensar XML, pero más pequeño, más rápido y más sencillo.
- ✓ **Provisión de combustible:**Se genera el tipo de combustible que el vehículo consume.
- ✓ **Ralentiza.** Hacer lenta una actividad o proceso, o disminuir su velocidad.
- ✓ **Ralentización.** Disminución de la velocidad, especialmente referido a un proceso o actividad.
- ✓ **RDBMS.** Relational Database Management System o RDBMS - Sistema de Gestión de Base de Datos Relacional o SGBDR.
- ✓ **REST.** La Transferencia de Estado Representacional (Representational State Transfer) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.
- ✓ **Servlets**: Aplicaciones Java que corren en un entorno de servidor web. Esto es análogo a una aplicación Java que corre en un navegador. Los Java servlets se han vuelto muy populares como alternativas a los programas CGI. La diferencia entre ambos es que los applet de Java son persistentes. Esto significa que una vez que han sido iniciados, se mantienen en memoria y pueden satisfacer múltiples solicitudes. En contraste, los programas CGI desaparecen una vez que han satisfecho una solicitud.

- ✓ **Sharding.** Sharding distribuye un único sistema de base de datos lógica en un clúster de máquinas.
- ✓ **Sistemas POSIX.** Es el estándar de interfaz de sistemas operativos portables de IEEE basado en el sistema operativo UNIX.
- ✓ **Snapshot.** Es una herramienta para instalar en sitios webs, lo que hace esta herramienta o código por así decirlo es crear una visualización o pre visualización de cada vínculo de la página.
- ✓ **SOAP:** (siglas de *Simple Object Access Protocol*) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datosXML.
- ✓ **SQL**: Lenguaje de definición de datos (LDD), un lenguaje de definiciones de vistas (LDV) y un lenguaje de manipulación de datos (LMD), que posee también capacidad para especificar restricciones y evolución de esquemas.
- ✓ **SRS** o **ERS**: (Software Requirement Specifications) Especificación de Requisitos Software, es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales. Además de los casos de uso, la ERS también contiene requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación (Como por ejemplo restricciones en el diseño o estándares de calidad).
- ✓ **Streaming:** Es la distribución de multimedia a través de una red de computadoras de manera que el usuario consume el producto al mismo tiempo que se descarga. La palabra streaming se refiere a que se trata de una corriente continua (sin interrupción). Este tipo de tecnología funciona mediante un búfer de datos que va almacenando lo que se va descargando para luego mostrarse al usuario. Esto se contrapone al mecanismo de descarga de archivos, que requiere que el usuario descargue los archivos por completo para poder acceder a los archivos.
- ✓ **Tomcat**: Contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystems

- ✓ **Tupla.** En informática, o concretamente en el contexto de una base de datos relacional, un registro (también llamado fila o tupla) representa un objeto único de datos implícitamente estructurados en una tabla.
- ✓ **Ubicuo.** Es entendida como la integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados.
- ✓ **UML**: (Unified Modeling Languaje) Lenguaje gráfico para construir, documentar, visualizar y especificar un sistema de software.
- ✓ **Views.** En base de datos , la vista es el conjunto de resultados de un almacenado consulta o mapa y reducir las funciones en los datos , que las bases de datos los usuarios pueden consultar la misma manera que lo haría un objeto de colección de base de datos persistente.
- ✓ **Webscale:** Un sistema que es altamente disponible, confiable, transparente y de alto rendimiento, escalable, accesible, seguro, útil y barato.
- ✓ **Wrapper.** Es un sistema de red ACL que trabaja en terminales y que se usa para filtrar el acceso de red a servicios de protocolos de Internet que corren en sistemas operativos (tipo UNIX), como Linux o BSD.
- ✓ **XML.** Extensible Markup Language (XML) es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos en un formato que sea legible.
- ✓ **Xquery.** Es un lenguaje de consulta diseñado para colecciones de datos XML, proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML como, por ejemplo, bases de datos relacionales o documentos ofimáticos.


APACHE JMETER

JMETER

Esta herramienta se utilizó para comprobar el indicador de tiempo de carga y establecer los valores para la comprobación de la hipótesis.

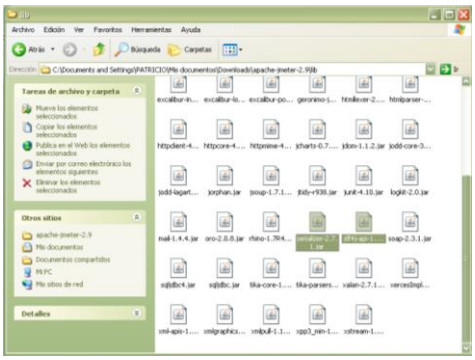
Apache JMeter es un software de código abierto, diseñado para cargar el comportamiento funcional de prueba y medir el rendimiento de sitios web y bases de datos.
JMeter se puede utilizar para simular una carga pesada en el servidor de red o un objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga, algunos servidores de rendimiento son:

- ✓ Web - HTTP, HTTPS
- ✓ JABÓN
- ✓ Base de datos a través de JDBC
- ✓ Correo - SMTP (S), POP3 (S) e IMAP (S)
- ✓ Comandos nativos o scripts de shell

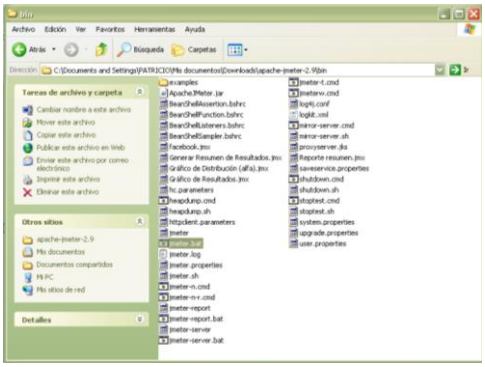
JMeter no es un navegador. En cuanto a los servicios web y servicios a distancia se refiere.
JMeter se parece a un navegador, sin embargo no realiza todas las acciones apoyadas por los navegadores. En particular, no ejecuta el Javascript que se encuentra en las páginas HTML.

PASOS PARA COMPROBAR EL RENDIMIENTO EN LAS BASES DE DATOS

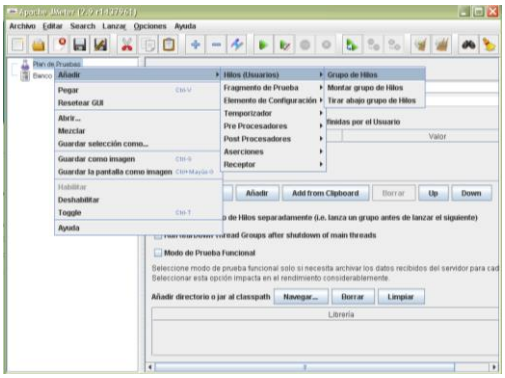
Para realizar un Test de Base de Datos se debe configurar JDBC, ya que este driver contiene el .jar de sql lo que permite una conexión exitosa.
Lo primero que debemos realizar es descargarnos el JDBC, extraer y luego pegar el archivo .jar del controlador en la carpeta Jmeter/lib.



Luego Ejecutamos el JMeter, ingresamos a la carpeta Jmeter luego a la carpeta bin y ejecutamos el archivo jmeter.bat)

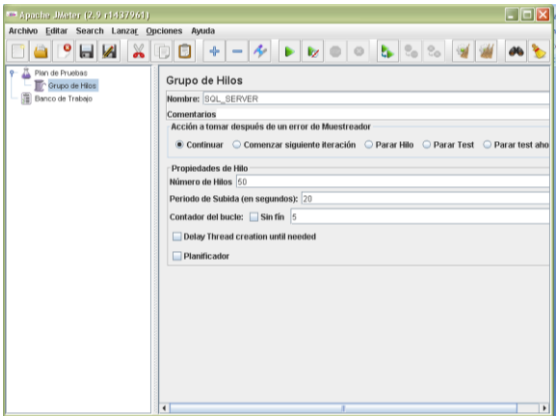


Una vez configurado JDBC y ejecutado JMeter creamos un plan de pruebas, para ello damos click sobre el icono que aparece en el marco de la izquierda denominado “Plan de Pruebas”, y agregamos un nuevo Plan.

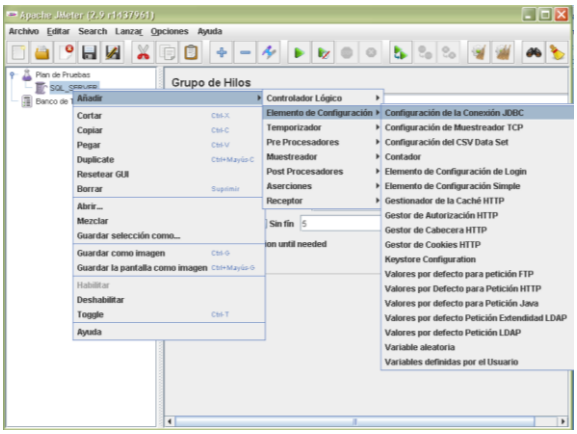


Lo primero que se añade al plan de pruebas es un grupo de hilos de ejecución que permita definir el número de usuarios a simular, donde se puede especificar:

- ✓ **Número de Subprocesos (Usuarios):** 50
- ✓ **Período de Aceleración (Segundos):** 20 es la cantidad a aplazar el inicio de cada usuario.
- ✓ **Loop de Vistas (Número de Iteraciones):** 5

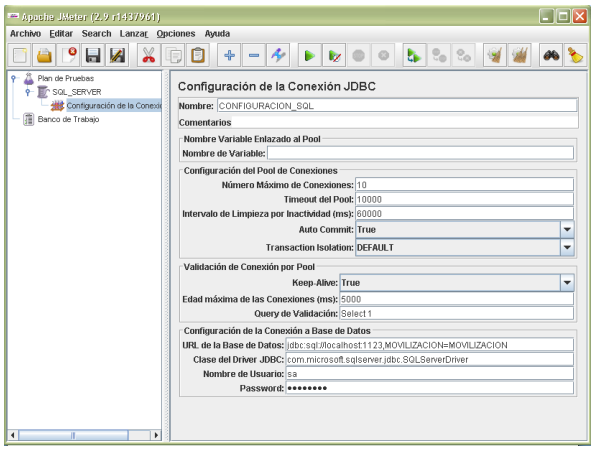


Añadir la conexión JDBC en el mismo grupo, damos click derecho seleccionamos Añadir, Elementos de Configuración y seleccionamos elementos de la conexión JDBC.

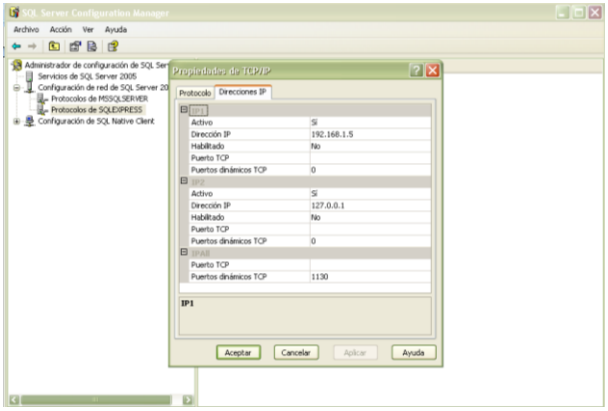


Y proporcionamos los valores solicitados en la configuración de la conexión:

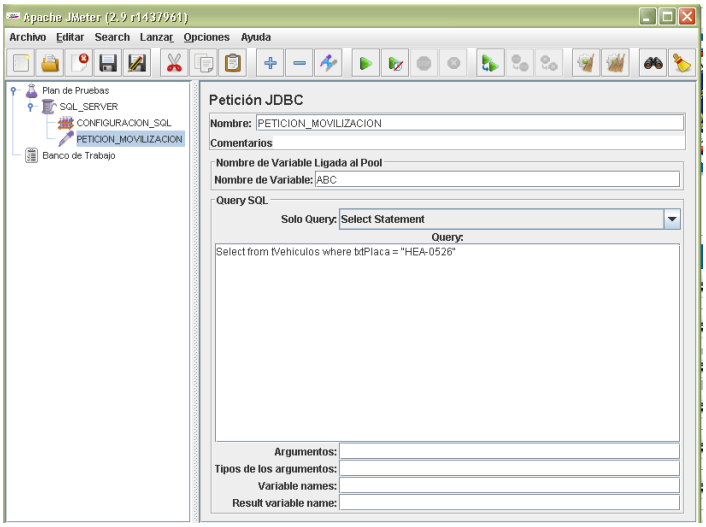
- ✓ Nombre de la Variable
- ✓ Número máximo de conexiones
- ✓ Intervalo
- ✓ URL de la Base de Datos
- ✓ Controlador JDBC
- ✓ Nombre de Usuario de la Base de Datos
- ✓ Contraseña



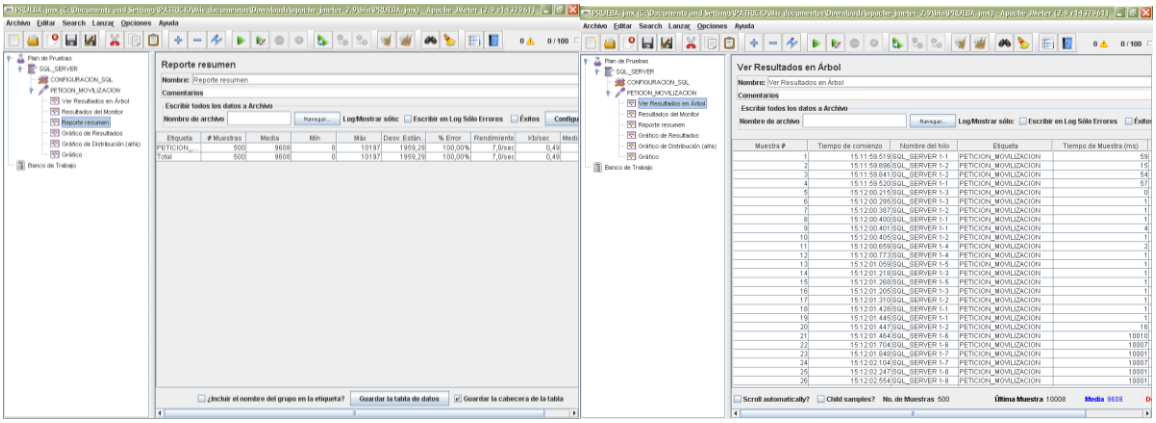
Luego observamos el número de puerto en configuración del servidor SQL.

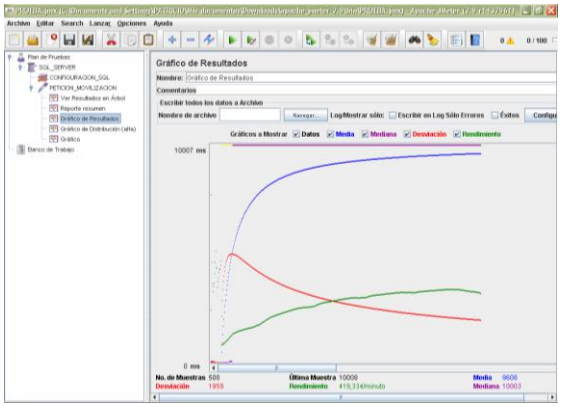


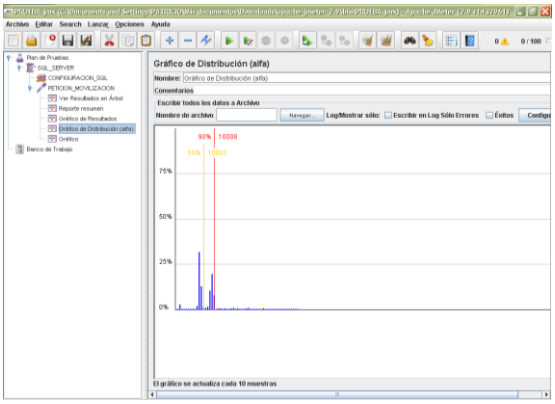
Ahora para realizar una tarea con la base de datos hay que añadir una petición JDBC y especificar la consulta que se requiera realizar.

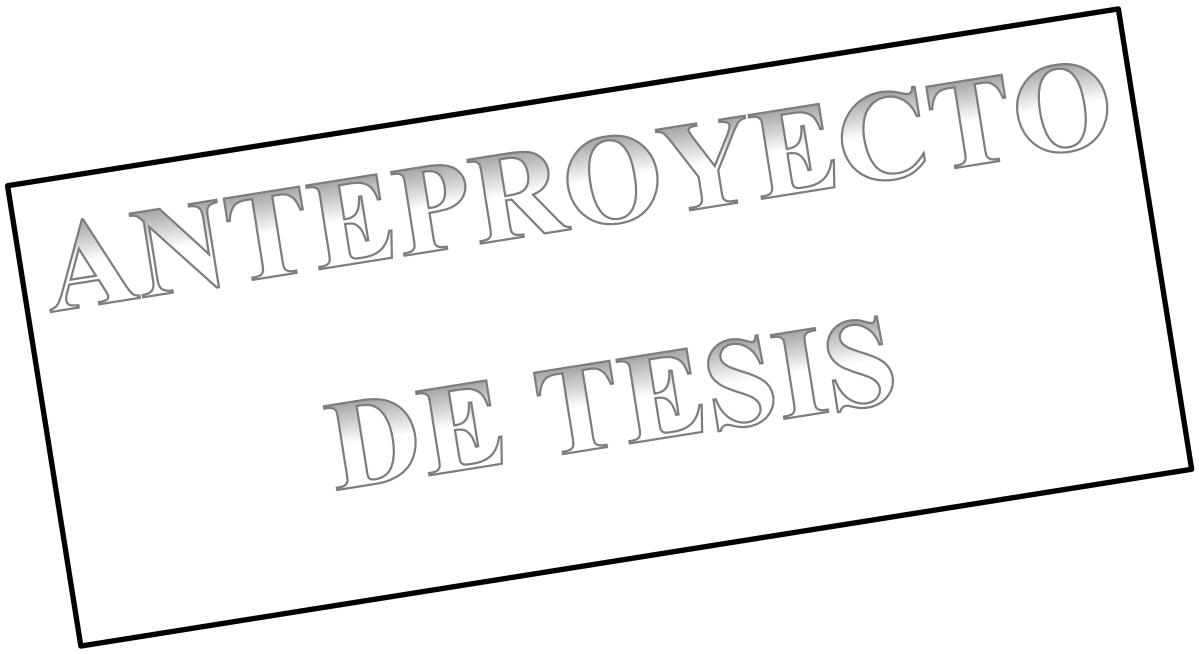


Y por último agregamos todos los resultados, tanto tablas como gráficos que se requiera mostrar.








ANTEPROYECTO
DE TESIS

1. TÍTULO DEL PROYECTO

Estudio comparativo de modelos de bases de datos relacionales y no relacionales y su incidencia de los procesos en sistemas informáticos. **Caso práctico:** Sistema de Gestión Vehicular.

2. PROBLEMATIZACIÓN

2.1. IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA

La Unidad de Transportes de la Universidad Nacional de Chimborazo en la actualidad cuenta con un sistema básico de registro de información de los vehículos, así como también el listado de sus funcionarios, sin embargo es limitado ya que en la Unidad de Transporte se realiza procesos como: Mantenimiento Vehicular, Órdenes de Movilización, Provisión de Combustible, Actas Entrega – Recepción de los diferentes vehículos de la Unidad, lo cual genera gran cantidad de información razón por la cuál es necesario un Sistema de Gestión Vehicular que permita cumplir con los requerimientos básicos de la Unidad.

2.2. ANÁLISIS CRÍTICO

La informática es una ciencia que nunca se deja de crecer por ello día a día hay mejoras con el afán de cubrir las necesidades existentes en este caso específico en la Unidad de Transportes y Riesgos Laborales de la UNACH.

Es necesario para la Unidad de Transportes y Riesgos Laborales de la UNACH, contar con un sistema informático que permita realizar múltiples actividades tales como:

- ✓ El registro de los de los usuarios tales como: el personal que labora en la Unidad que la conforman Choferes, Secretaria.
- ✓ Órdenes de Movilización: dentro de la unidad se clasifican en dos:
- ✓ Órdenes de Movilización dentro de la ciudad: En estas órdenes se debe registrar la información pertinente para la salida de los automóviles.

- ✓ Órdenes Fuera de la Ciudad: En estas órdenes se debe registrar la información pertinente para la salida de los automóviles.
- ✓ Órdenes de Provisión de Combustible: Se debe llevar un registro del consumo de combustible de cada automóvil perteneciente a la Unidad.
- ✓ Mantenimiento: Se debe registrar la fecha del mantenimiento y todos los detalles del mismo.
- ✓ Acta entrega de Recepción: Cada conductor de la unidad es responsable de un vehículo en la realización de los viajes se prestan los vehículos y por ello llenan el acta especificando en qué condiciones reciben el vehículo.

2.3. PROGNOSIS

Hoy en día las Instituciones Educativas automatizan sus procesos con el fin de mejorar su sistema de educación, adaptarse convenientemente a políticas gubernamentales y optimizar recursos. En el ámbito de controlar la información de la Unidad, se debe contar con un Sistema que permita realizar la gestión del departamento.

En este sentido si la Universidad Nacional de Chimborazo emprende un proyecto para la construcción del Sistema de Gestión Vehicular, permitirá integrarse a nivel universitario, administración y de recursos basado en normas internacionales, en la actualidad la Unidad de transporte y Riesgos Laborales cuenta con un sistema básico de registro de información.

El proyecto tiene como finalidad apoyar la gestión de la información del recurso de la Unidad: Ordenes de Salida, Consumo de Combustible, Mantenimiento Vehicular; además de la administración de usuarios como son autoridades, estudiantes, personal docente y administrativo de la Universidad Nacional de Chimborazo.

2.4. DELIMITACIÓN

El campo de las TICS (Tecnologías de Información y Comunicación) es extenso; abarca aspectos muy especializados que van desde la recuperación de información, la aplicación de la tecnología al servicio del usuario, la descripción, los metadatos, las interfaces de usuario o la preservación de los documentos electrónicos, hasta su repercusión social y el mercado y políticas de información.

El proyecto de investigación desarrolla el ámbito de investigar las bases de datos Relacionales y no Relacionales, al Sistema de Gestión Vehicular de la Universidad Nacional de Chimborazo, adopta puntos de vista global de la aplicación de software y la tecnología para la gestión de los procesos técnicos, informativos y de servicio al usuario, es una herramienta que colabora a la eficiencia en los procesos realizados en la Unidad.

2.5. FORMULACIÓN DEL PROBLEMA

¿En qué forma las Bases de Datos Relacionales y no Relacionales incide en la eficiencia de los procesos en sistemas informáticos.

2.6. OBJETIVOS

2.6.1 Objetivo General:

Realizar un estudio comparativo de bases de datos Relacionales y no Relacionales, aplicado al Sistema Vehicular de la Universidad Nacional de Chimborazo.

2.6.2 Objetivos específicos:

- ✓ Análisis de arquitecturas de Bases de datos no Relacionales.
- ✓ Investigar el proceso de desarrollo de las bases de datos no Relacionales
- ✓ Implementación de la Base de Datos no Relacional en el Sistema Vehicular.

2.7.- JUSTIFICACIÓN

Los cambios tecnológicos producidos en los últimos tiempos han sido de gran importancia para la humanidad que busca un medio para facilitar sus necesidades y labores diarias.

Hoy en día las Instituciones de educación superior en el Ecuador requieren automatizar sus departamentos con el fin de mejorar procesos académicos y administrativos; adaptándose convenientemente a políticas gubernamentales y optimización de recursos.

Por esa razón se pretende crear nuevas estrategias enfocadas al crecimiento y perfeccionamiento de sus servicios, analizar sus fortalezas y debilidades, tomar en cuenta sus oportunidades y amenazas para poder así crear ventajas competitivas que las encamine hacia un mejor escenario.

El software de computador es una herramienta indispensable para el trabajo diario en las unidades, permite la administración y gestión del material de forma eficiente gracias a su rápido flujo de información y la confiabilidad al momento de emitir reportes.

En la Unidad de Transportes de la UNACH es vital la existencia de una comunicación interna entre su departamento y externa con sus estudiantes, docentes y comunidad, además es importante que el flujo de información sea cada vez más rápido con el fin de identificar los problemas en el menor tiempo posible y tomar decisiones oportunas para resolverlos.

En este sentido el desarrollo de un Sistema de Gestión Vehicular de la Universidad Nacional de Chimborazo es de vital importancia con el fin de conseguir mayor productividad, proporcionando el máximo rendimiento en la realización de tareas tradicionales; además de potenciar la comunicación entre las diferentes Unidades y la reutilización de información y procesos, garantizando la competitividad con las demás Instituciones.

3.- MARCO TEÓRICO

3.1.- ANTECEDENTES DEL TEMA

3.1.1 Digg

Es un sitio web principalmente sobre noticias de ciencia y tecnología. Combina marcadores sociales, blogging y sindicación con una organización sin jerarquías, con control editorial democrático, lo cual permite que se publiquen artículos sobre una gran variedad de géneros. Se trata de una página web en inglés que permite a los usuarios enviar, votar y jerarquizar las informaciones que crean en sus webs o encuentran en Internet. Es una de las webs precursoras en la valoración social de la información en la Red. Para introducir información hay que estar registrado.

3.1.2 Facebook

Consistente en un sitio web de redes sociales. Originalmente era un sitio para estudiantes de la Universidad de Harvard, pero actualmente está abierto a cualquier persona que tenga una cuenta de correo electrónico. Los usuarios pueden participar en una o más redes sociales, en relación con su situación académica, su lugar de trabajo o región geográfica.

3.1.3 Twitter

Es un servicio de microblogging. La red permite enviar mensajes de texto plano de corta longitud, con un máximo de 140 caracteres, llamados tweets, que se muestran en la página principal del usuario.

3.1.4 Rackspace

Es una empresa de hosting. Rackspace tiene dos principales segmentos de nivel de servicio: Managed e Intensivo. Ambos niveles de servicio reciben apoyo a través de e-mail, teléfono, chat en vivo, y los sistemas de boletos, pero están diseñados para adaptarse a las necesidades de los diferentes negocios.

El nivel de soporte Gestionado consiste en "on-demand" de apoyo, donde se proporcionan servicios proactivos, pero el cliente es ponerse en contacto con Rackspace cuando necesitan ayuda adicional.

El nivel de apoyo intensivo consiste en "proactiva" de apoyo, donde muchos servicios proactivos se proporcionan, y los clientes reciben consultas adicionales sobre su configuración del servidor. Implementaciones altamente personalizadas generalmente caen bajo este nivel de apoyo.

Algunos servicios y productos sólo están disponibles para ciertos niveles de soporte.

3.1.5 SimpleGEO

LOCATION SERVICES EN LA NUBE

Es una “infrastructure play”. Ofrecen una variedad de servicios para startups que tienen que ver con la geolocalización. Estos servicios no sólo son difíciles de desarrollar sino también de operar y mantener.

3.1.6 COUCHDB

QUIÉN USA COUCHDB

HUDORA está usando CouchDB para una pista interna y aplicación de seguimiento más.

Servicios SOTEL IP está usando CouchDB para algunos llaman aplicaciones de enrutamiento dinámico más.

Engine Yard está usando CouchDB para las métricas y seguimiento de la instalación de paquetes.

BerrySki BlackBerry App GPS que ofrecen mapas de esquí de Europa y América del Norte. Utiliza CouchDB para administración de suscripciones.

Credit Suisse Materias primas departamento usar CouchDB para almacenar los detalles de configuración de su pitón marco Marketdata.

3.2.- ENFOQUE TEÓRICO

3.2.1.- ENFOQUE TEÓRICO

3.2.1.1 BASE DE DATOS

Una base de datos o banco de datos (en ocasiones abreviada con la sigla *BD.*) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen programas denominados sistemas gestores de bases de datos, abreviado SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo, en España los datos personales se encuentran protegidos por la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD).

3.2.2 Base de datos relacional

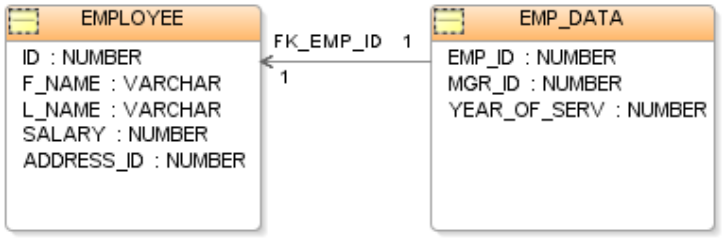


Figura 5. Base de Datos Relacional

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. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: **"Modelo Relacional"**. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

3.2.3 CONCEPTOS BÁSICOS SOBRE LAS BASES DE DATOS RELACIONALES

3.2.3.1 BASE DE DATOS

Una base de datos es un conjunto de información relacionada con un asunto o con una finalidad. En Access, una base de datos es un archivo que puede contener tablas donde se encuentra toda la información sobre un tema específico, consultas, formularios, informes entre otros. Tal como una colección de música, el control de alumnos en un colegio o un directorio telefónico

La información contenida en unas tablas tiene múltiples utilidades.

3.2.3.2 Campo

Un campo es un componente de una tabla que contiene un elemento específico de información, como ejemplo, nombre apellidos, edad, sexo, etc.

3.2.3.3 Registro

Un registro está compuesto por todos los campos de la tabla, de manera que un campo es una parte de un registro.

En una tabla, las filas corresponden a los registros, los cuales son individuales; y las columnas corresponden a los campos, que son una parte única de un registro. De manera tal que la información contenida en una tabla "Directorio Telefónico", podría tener la siguiente estructura:

Nombre del Usuario	Numero Telefónico	Dirección
Ramón Padilla	555-6232	San Salvador
Julio Medina	555-9878	Soyapango
Marvin Zelaya	999-9856	Apopa
Denis Martínez	999-2564	Miralvalle

3.2.4 BASE DE DATOS NO RELACIONALES

Las Bases de Datos no Relacionales intentan atacar este problema proponiendo una estructura de almacenamiento **más versátil**, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas (también llamado JOIN) teniendo que recurrir a la desnormalización de datos.

Algunas implementaciones bien conocidas que podríamos como NoSQL son: CouchDB, MongoDB, RavenDB, Neo4j, Cassandra, BigTable, Dynamo, Riak, Hadoop, y otras muchas.

1.3 ¿En qué se diferencian exactamente?

Si tuviéramos que resumir las características comunes en estos sistemas, yo diría que son principalmente tres:

- ✓ Ausencia de esquema en los registros de datos,
- ✓ Escalabilidad horizontal sencilla,
- ✓ Velocidad endiablada (aunque esto último no siempre es cierto, pues muchos de estos sistemas aún no están suficientemente maduros).

La primera característica significa que los datos no tienen una definición de atributos fija, es decir: Cada registro (o documento, como se les suele llamar en estos casos) 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).

Con escalabilidad horizontal me refiero a 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. Por supuesto, 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.

Por último, muchos de estos 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.

3.2.5 CouchDB

CouchDB es una base de datos orientada a documentos, open source y gratuita. Derivada del almacenamiento clave/valor, utiliza JSON para definir el esquema de un elemento. CouchDB está concebida para tender un puente al hueco que existe entre bases de datos relacionales y bases de datos orientadas a documentos gracias a las "vistas", que pueden ser creadas dinámicamente a través de JavaScript. Estas vistas mapean los datos del documento en estructuras similares a tablas que pueden ser indexadas y consultadas.

3.2.6. Mongo

Mongo es el sistema de base de datos desarrollada en 10gen por Geir Magnusson y Dwight Merriman (recordemos de DoubleClick). Como CouchDB, Mongo es una base de datos orientada a documentos JSON, salvo que está diseñada para ser una verdadera base de datos de objetos, más que para un almacenamiento de clave/valor puro. Originalmente, 10gen enfocó poner juntos una pila completa de servicios web, aunque sin embargo, más recientemente ha tenido que ser reenfocado principalmente en la base de datos Mongo.

3.3.-DEFINICIÓN DE TÉRMINOS BÁSICOS

En esta sección se especifica el significado de término utilizados durante la realización del proyecto de investigación.

- ✓ **Asp.net 2005:** Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.
- ✓ **Automatización de procesos:** La automatización es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.
- ✓ **Baseline:** Es la línea de base de un proyecto, es el plan original más todos los cambios negociados con los patrocinadores y aprobados como parte del proyecto.
- ✓ **Cliente Servidor:**Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es mejor en un sistema operativo multiusuario distribuido a través de una red de computadoras.
- ✓ **C# 2005:** Es un lenguaje de programación diseñado para crear una amplia gama de aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos.
- ✓ **Escalabilidad:** Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para extender el margen de operaciones 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.
- ✓ **Estandarización:** La estandarización forma parte de los seis pasos necesarios para llevar a cabo la limpieza de datos. Esta consiste, en separar la información en diferentes campos, así como unificar ciertos criterios para un mejor manejo y manipulación de los datos.
- ✓ **Etapas Iteradas:** Se refiere a la acción de repetir una serie de etapas, pasos un cierto número de veces.

- ✓ **Flujo de información:** Movimiento de información entre departamentos e individuos dentro de una organización y entre una organización y su entorno.
- ✓ **Frameworks:** Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.
- ✓ **IIS (Internet Information Server):** Los servicios de Internet Information Server (o IIS), son los servicios de software que admiten la creación, configuración y administración de sitios Web, además de otras funciones de Internet. Los servicios de Microsoft Internet Information Server incluyen el Protocolo de transferencia de noticias a través de la red (NNTP), el Protocolo de transferencia de archivos (FTP) y el Protocolo simple de transferencia de correo (SMTP).
- ✓ **Ingeniería de Procesos:** Es diseñar, poner en marcha y ejecutar todo lo necesario para obtener la óptima explotación de los sistemas o procesos a instalar en los departamentos de producción de las empresas.
- ✓ **Integridad de datos:** Se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas formas diferentes.

  Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.

- ✓ **Interfaz:** Es el medio con cual el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.
- ✓ **Microsoft Office Proyect 2007:** Es un software de administración de proyectos diseñado, desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

- ✓ **Microsoft Office Word 2007:** Es un programa que nos ayuda a crear y compartir excelentes documentos combinando un amplio conjunto de herramientas de escritura en una interfaz fácil de utilizar.
- ✓ **Microsoft Office Visio 2007:** Microsoft Visio es un software de dibujo vectorial para Microsoft Windows. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación.
- ✓ **Microsoft Office Excel 2007:**Es una herramienta eficaz que puede usar para crear y aplicar formato a hojas de cálculo, y para analizar y compartir información para tomar decisiones mejor fundadas.
- ✓ **Microsoft Visual Studio 2005:** Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.
- ✓ **Reusabilidad:** Respecto a la calidad de un programa, la reusabilidad hace referencia a poder volver a usar parte de dicho software en otro proyecto.
- ✓ **SaaS (una aplicación en la nube):** Es un modelo de distribución de software en donde la compañía de tecnologías de información y comunicación provee el servicio de mantenimiento, operación diaria, y soporte del software usado por el cliente. Regularmente el software puede ser consultado en cualquier computador, esté presente en la empresa o no.
- ✓ **SQL:** Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas.
- ✓ **SQL Server 2005:** SQL Server 2005 es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la base de datos SQL Server 2005 ofrece almacenamiento más seguro y confiable tanto para datos relacionales como

estructurados, lo que permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento para utilizar en su negocio.

- ✓ **Stakeholders**: Para referirse a quienes pueden afectar o son afectados por las actividades de una empresa.
- ✓ **S+S (Software más Servicios):** Describe la idea de combinar los servicios hospedados con capacidades que se consiguen mejor con ejecución local de software.
- ✓ **Tabla Cutter ´s:** Tabla para la clasificación de autores que proporciona un código y un prefijo.
- ✓ **Tecnología:** Es el conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar y crear bienes o servicios que facilitan la adaptación al medio y satisfacen las necesidades de las personas.
- ✓ **Tecnología .Net:** .NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
- ✓ **Testeabilidad:** Se refiere a la capacidad que tiene una prueba o experimento de ser reproducido o replicado.
- ✓ **Testers:** Un tester técnico se integra más fácilmente con un equipo de programadores, pueden más fácilmente tener un lenguaje común y es más factible que participe en la automatización de las pruebas. Puede aportar en las pruebas técnicas, mejora su participación en pares.
- ✓ **Testing y Mocking:** Con Testing es las pruebas que se realizan al sistema que se está realizando y con mocking es la cadena con que se va a realizar la desencadenación.
- ✓ **UML (UnifiedModelingLanguage):** Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.
- ✓ **Web:** El sistema de documentos (o páginas web) interconectados por enlaces de hipertexto, disponibles en Internet.

- ✓ **WebApps:** Aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.
- ✓ **Web enmarañada:** Entrelazar de manera desordenada y accidental páginas, hipervínculos, enlaces, direcciones, entre otras.
- ✓ **Website:** Conjunto de textos, gráficos, fotografías, sonidos o videos que unidos a otros elementos análogos como pueden ser banners o hipervínculos y que han sido creados para su exposición en la Red para que sean visionados por terceros a través de un navegador.
- ✓ **Windows XP Service Pack 2:** Es el sistema operativo más utilizado contiene un paquete de seguridad denominado Service Pack 2. Está relacionado con la seguridad; y se trata de uno de los Service Pack más importantes publicados hasta el momento y que proporciona una mejor protección contra virus, gusanos y piratas informáticos, e incluye las funciones Firewall de Windows, Bloqueador de elementos emergentes y el nuevo Centro de seguridad de Windows.
- ✓ **Windows 7 Home Premium:** Windows 7 es la versión más reciente de Microsoft Windows, línea de sistemas operativos producida por Microsoft Corporation. Esta versión está diseñada para uso en PC, incluyendo equipos de escritorio en hogares y oficinas, equipos portátiles, tablet PC, netbooks y equipos media center.

3.4.-HIPÓTESIS

Las Bases de Datos Relacionales son más eficientes que las Bases de Datos No Relacionales en el Sistema Informático del Departamento de Transportes de la Unach.

3.5. IDENTIFICACIÓN DE VARIABLES

3.5.1 VARIABLE INDEPENDIENTE

- ✓ Las bases de datos Relacionales y no Relacionales.

3.5.2 VARIABLE DEPENDIENTE

- ✓ Eficiencia de los procesos en sistemas informáticos.

4.- METODOLOGÍA

En esta fase se define los métodos, mecanismos, estrategias o procedimientos a seguirse en la investigación. A continuación se especifica sus componentes:

4.1.- TIPO DE ESTUDIO

La presente investigación de acuerdo al propósito o las finalidades perseguidas se caracteriza por ser aplicada debido a que mediante los aportes teóricos de las Arquitecturas de Software se pretende dar solución al problema del manejo de la
gran cantidad de información en la Unidad de Transportes de la Universidad Nacional de Chimborazo.

El proyecto de investigación según la clase de medios utilizados es documental en su fase de investigación teórica la misma que está sustentada en libros, publicaciones, tesis; además de ser de campo en el proceso de recolección de requisitos de software.

4.2.- POBLACIÓN Y MUESTRA

Los elementos involucrados en el trabajo de investigación está constituido por las Bases de Datos Relacionales (Microsoft SQL Server) y Bases de Datos No Relacionales (MongoDB) que van hacer el motivo de nuestra investigación.

4.3.- OPERACIONALIZACIÓN DE LAS VARIABLES

Tabla 1. Operacionalización de variables

Variable	Tipo	Definición Conceptual	Dimensiones	Indicadores
Base de Datos Relacional	Independiente	Una base de datos relacional es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado para implementar bases de datos ya planificadas. Permiten establecer relaciones entre los datos, y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: **"Modelo Relacional"**.	Análisis	Análisis Descriptivo
Base de Datos No Relacional	Independiente	Las bases de datos No SQL son la nueva generación de bases de datos, frecuentemente no relacionales, distribuidas, de código abierto y escalables horizontalmente. Se han convertido en un requisito principal para los empleadores que buscan desarrollar sus potencialidades en Big Data.	Análisis	Análisis Descriptivo
Eficiencia de los procesos del Sistema Informático de la Unidad de Transportes de la UNACH.	Dependiente	Se define como la capacidad de obtener información del sistema informático de la Unidad de Transportes de la Unach con la utilización un mínima de recursos.	Complejidad	✓ Líneas de código ✓ Tiempo de Carga (Rendimiento) ✓ Tamaño en Disco

4.4.- PROCEDIMIENTOS

4.4.1. TÉCNICA DE INVESTIGACIÓN

Las técnicas e instrumentos que se utilizarán para la recolección de la información y datos; fundamentalmente son las siguientes:

- ✓ Entrevistas
- ✓ Encuestas
- ✓ Observación

La realización del Sistema de Gestión Vehicular involucra descubrir exactamente las fronteras del sistema, sus funciones, quienes son los usuarios y que espera del proyecto cada uno de ellos. Esto se logra mediante las entrevistas y encuestas puesto que nos permitirá recabar información referente al grado de aceptabilidad de los usuarios y requerimientos de software existentes para la creación del Sistema; además la observación nos permitirá sintetizar conocimiento relacionado con investigaciones similares y para el estudio comparativo de las bases de datos Relacionales y no Relacionales. La encuesta y la observación en conjunto nos permitirán comprobar la hipótesis planteada ya sea para aprobarla o rechazarla.

4.4.2 INSTRUMENTOS DE RECOLECCIÓN DE DATOS

Los instrumentos que se emplearán para la recolección de datos en esta investigación serán:

- ✓ Cuestionario de Aceptabilidad del Sistema.
- ✓ Guía de Entrevista de recolección de requerimientos de Software
- ✓ Ficha de eficiencia de procesos por parte de los conductores de la UNACH.

Estas serán aplicadas a los conductores de la Universidad Nacional de Chimborazo con el fin de recolectar la información pertinente para el desarrollo del Sistema; además de ser una pieza fundamental para el análisis de la hipótesis.

4.5.- PROCESAMIENTO Y ANÁLISIS

Para la presente tesis de acuerdo a los objetivos planteados se utilizarán los siguientes modelados y análisis orientados al desarrollo e Ingeniería de Software.

- ✓ Análisis de Riesgos
- ✓ Análisis de Requisitos
- ✓ Modelado de Negocio
- ✓ Modelo de Casos de Uso del Negocio y Modelo de Objetos del Negocio
- ✓ Modelo de Casos de Uso
- ✓ Análisis comparativo de las Bases de Datos Relacionales y no Relacionales enfocada al caso práctico: Migración del Sistema de Gestión Vehicular a software libre.
- ✓ Modelo de Análisis / Diseño Arquitectónico
- ✓ Modelo de Datos
- ✓ Modelo de Implementación
- ✓ Modelo de Pruebas
- ✓ Análisis de los beneficios de la administración automatizada vs manual en la UTRL de la Universidad Nacional de Chimborazo: Análisis de eficiencia a los conductores de la Institución mediante la utilización del Sistema y comparación mediante la gestión manual.

5.- MARCO ADMINISTRATIVO

5.1.1 RECURSO HUMANO

A continuación se detalla el personal encargado de la creación del Sistema los mismos que no poseen un costo directo debido a que desarrollan su rol en beneficio de la Universidad

Nacional de Chimborazo ya sea dentro de su ámbito de trabajo o en el caso de estudiantes vinculado al proceso de graduación.

5.1.1.1 Desarrolladores

La administración del proyecto así como también el análisis, diseño, programación está asignada a los siguientes desarrolladores estudiantes de la Escuela de Ingeniería en Sistema y Computación de la UNACH:

- Paulina Alexandra Gaona Gutiérrez.
- Edison Patricio Sandoval Narváez.

8 horas diarias lunes a sábado

5.1.1.2 Coordinación Técnica y Tutoría

La coordinación técnica y tutoría en cada fase es realizada por:
Ing. Anita Congacha: Ingeniera en Sistemas Catedrática en la Universidad Nacional de Chimborazo, con experiencia en Administración de proyectos, Ingeniería de Software y tutoría de tesis.
Ing. Jorge Delgado: Ingeniera en Sistemas Catedrática en la Universidad Nacional de Chimborazo, con experiencia en Administración de proyectos, Ingeniería de Software y tutoría de tesis

5.1.2 HARDWARE

5.1.2.1 Computadores

Tabla 2. Descripción de Costos/hora de Computadores.

Descripción	Costo/Hora
Desktop-Dual Core 1.6 Ghz-512 MB-250 GB	$ 0.30
Desktop-Pentium IV-1 GB-160 GB	$ 0.30
Portátil-AMD TURION TX2 2.30 Ghz-3 GB-320 GB	$ 0.30

5.1.2.2 Impresoras

Tabla 3. Descripción de Costos/hora de Impresoras.

Descripción	Costo/Hora
Lexmark X5270	$ 0.05
Lexmark X1270	$ 0.05

5.1.3 SOFTWARE

El costo correspondientes a licencias de Software no se específica en esta sección debido al convenio existente entre la Universidad Nacional de Chimborazo y Microsoft Corporation. Las demás herramientas de Software son de libre utilización.

5.1.3.1 Tecnologías

- ✓ Cliente Servidor.
- ✓ Google Maps

5.1.3.2 Sistemas Operativos

- ✓ Windows XP Service Pack 2.
- ✓ Windows 7 Home Premium.

5.1.3.3 Gestor de Base de Datos

- ✓ SQL Server 2005.

5.1.3.4 Lenguajes de Programación

- ✓ Microsoft Visual Studio 2010.
- ✓ Asp.net 2010.
- ✓ C# 2010.
- ✓ Java.

5.1.3.5 Aplicaciones

- ✓ Microsoft Office Word 2010.
- ✓ Microsoft Office Proyect 2010.
- ✓ Microsoft Office Visio 2010.

5.1.3.6 Servidor Web

- ✓ IIS (Internet Information Server).

5.1.3.7 SUMINISTROS

Estos suministros son utilizados durante el desarrollo del proyecto principalmente para tareas de documentación.

Tabla 4. Descripción de costo total de suministros.

Descripción	**Costo Total**
1000 hojas de papel para copia e impresión A4	$ 4.00
Cartuchos para impresión	$ 50.00
Fotocopias	$ 60.00
Memory Flash Kingston 2GB	$ 14.00
Kit de útiles de escritorio	$ 15.00

5.1.3.8 SERVICIOS

Tabla 5. Descripción de costo hora y total de servicios.

Descripción	**Costo/Hora**
Internet	$ 0.80
Descripción	**Costo Total**
Consultoría Adicional	$ 50.00
Teléfono, energía eléctrica	$ 90.00
Transportación	$ 50

5.2.- PRESUPUESTO

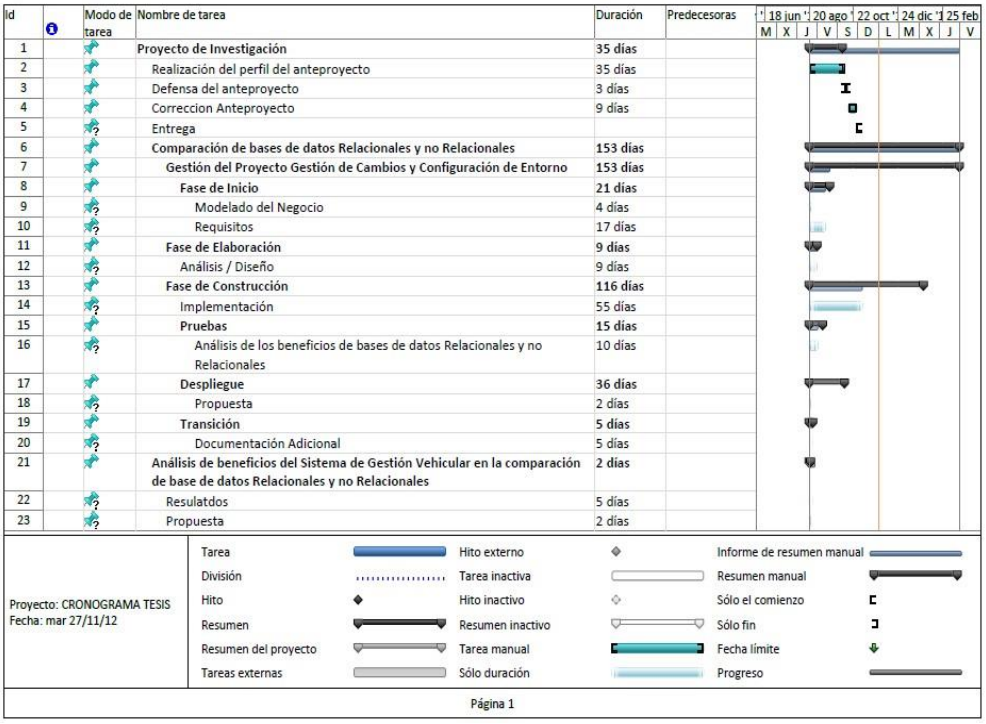
En esta sección se determina la previsión de recursos humanos, materiales y financieros fácilmente deducibles del análisis del cronograma y de la magnitud del proyecto de investigación.

El costo estimado total del proyecto de investigación es de **$ 3899.40** los mismos que serán financiados como aporte personal el mismo que hace un total de **$ 3899.40.** El detalle del presupuesto con sus respectivos costos se detalla a continuación.

Tabla 6. Estimación de costos del proyecto de investigación.

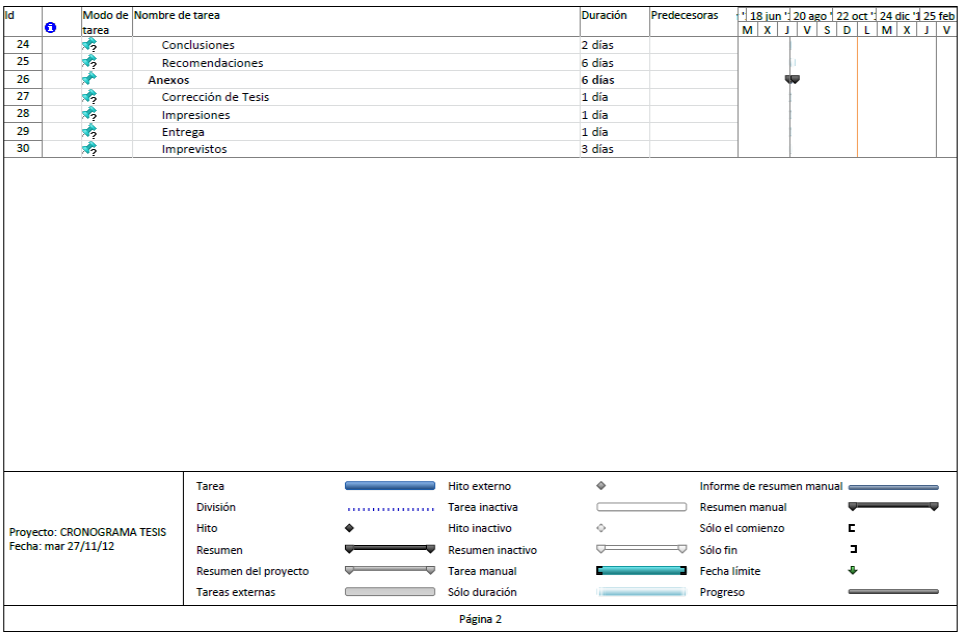
COSTOS		**Aporte UNACH**	**Aporte Personal**
Recurso Humano			
Desarrolladores	$ 2,240.00		**X**
Coordinación Técnica y Tutoría	$ 00.00	**X**	
Subtotal	**$ 2,240.00**		
Hardware			
Computadores	$ 988.80		**X**
Impresoras	$ 8.00		**X**
Subtotal	**$ 996.80**		
Software			
Licencias	$ -	**X**	
Subtotal	**$ -**		
Suministros			
1000 hojas de papel A4 $4	$ 4.00		**X**
Cartuchos para impresión	$ 50.00		**X**
Fotocopias	$ 60.00		**X**
Memory Flash Kingston 2GB	$ 14.00		**X**
Kit de útiles de escritorio	$ 15.00		**X**
Subtotal	**$ 143.00**		
Servicios			
Internet	$ 329.60		**X**
Consultoría Adicional	$ 50.00		**X**
Teléfono, energía eléctrica	$ 90.00		**X**
Transportación	$ 50.00		**X**
Subtotal	**$ 519.60**		
Total	**$ 3899.40**	$ 00.00	$ 3899.40

5.3.- CRONOGRAMA



Id	Modo de tarea	Nombre de tarea	Duración	Predecesoras
1		**Proyecto de Investigación**	**35 días**	
2		Realización del perfil del anteproyecto	35 días	
3		Defensa del anteproyecto	3 días	
4		Correccion Anteproyecto	9 días	
5		Entrega		
6		**Comparación de bases de datos Relacionales y no Relacionales**	**153 días**	
7		**Gestión del Proyecto Gestión de Cambios y Configuración de Entorno**	**153 días**	
8		**Fase de Inicio**	**21 días**	
9		Modelado del Negocio	4 días	
10		Requisitos	17 días	
11		**Fase de Elaboración**	**9 días**	
12		Análisis / Diseño	9 días	
13		**Fase de Construcción**	**116 días**	
14		Implementación	55 días	
15		**Pruebas**	**15 días**	
16		Análisis de los beneficios de bases de datos Relacionales y no Relacionales	10 días	
17		**Despliegue**	**36 días**	
18		Propuesta	2 días	
19		**Transición**	**5 días**	
20		Documentación Adicional	5 días	
21		**Análisis de beneficios del Sistema de Gestión Vehicular en la comparación de base de datos Relacionales y no Relacionales**	**2 días**	
22		Resulatdos	5 días	
23		Propuesta	2 días	

Página 1



Id	Modo de tarea	Nombre de tarea	Duración	Predecesoras
24		Conclusiones	2 días	
25		Recomendaciones	6 días	
26		**Anexos**	**6 días**	
27		Corrección de Tesis	1 día	
28		Impresiones	1 día	
29		Entrega	1 día	
30		Imprevistos	3 días	

Página 2

6.- BIBLIOGRAFÍA

- ✓ Cogoluègnes, A., Templier, T., & Piper, A. (2011). *Spring Dynamics Modules in Action.* United States of America: Manning Publications Co.
- ✓ Corporation, O. (3 de Marzo de 2012). *Project Jigsaw*. Recuperado el 3 de Marzo de 2012, de Open JDK: http://goo.gl/5UJxJ
- ✓ Eclipse. (2010). *Guía de programación de Virgo*. Recuperado el 11 de 5 de 2012, de http://goo.gl/GgEyh
- ✓ Knoernschild, K. (2012). *Java Application Architecture: Modularity Patterns with Examples Using OSGi.* United States of America: Prentice Hall.
- ✓ Martin, R. C. (Diciembre de 1996). *Granularidad*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/dZZEh
- ✓ Martin, R. C. (Mayo de 1996). *Principio de Inversión de Dependencias*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/SA8r9
- ✓ Martin, R. C. (Marzo de 1996). *Principios de la Orientación a Objetos*. Recuperado el 19 de Marzo de 2012, de http://goo.gl/oqExW
- ✓ Moreno, F. G. (01 de Enero de 2011). *Desarrollando una aplicación Spring Framework MVC v3 + JPA paso a paso*. Recuperado el 30 de Enero de 2012, de http://goo.gl/cMS5V
- ✓ Oracle. (1999). *JAR File Specification*. (Oracle) Recuperado el 26 de Marzo de 2012, de JAR File Specification: http://goo.gl/QYbm0
- ✓ Paez, N. (2010). *Utilización de programación orientada a aspectos en aplicaciones empresariales.* Buenos Aires: Anónimo.

- ✓ Palao, D. M. (1 de Febrero de 2010). *Desarrollando una aplicacion Spring Framework MVC paso a paso*. Recuperado el 30 de Enero de 2012, de http://goo.gl/XRFk4
- ✓ Ramos, A. (23 de Febrero de 2010). *Modularización Efectiva en Java*. Recuperado el 19 de Febrero de 2012, de Modularización Efectiva en Java: http://goo.gl/yQIKw
- ✓ Risberg, T., Evans, R., & Tung, P. (1 de Enero de 2010). *Developing a Spring Framework MVC application step-by-step*. Recuperado el 30 de Enero de 2012, de http://goo.gl/kMWUs
- ✓ Völter, M. (10 de Marzo de 2005). *Software Architecture Patterns*. Recuperado el 3 de Marzo de 2012, de A pattern language for building sustainable software architectures: http://goo.gl/jYVHv
- ✓ Walls, C. (2010). *Modular Java, Creating flexible applications with OSGi and Spring*. Dallas, Texas: The Pragmatic Bookshelf
- ✓ Wikilearning. (12 de Enero de 2010). *DEFINICIÓN Y TERMINOLOGÍA DE UN RDBMS*. Recuperado el 6 de Febrero de 2012, de http://goo.gl/CtKl6