Universidad
Carlos III de Madrid

Departamento de Informática

TRABAJO FIN DE GRADO

ALMACENES DE DATOS NOSQL, ESTUDIO DE LA TECNOLOGÍA

Autor: Ana María Lorente Borox

Tutor: Ana María Iglesias Maqueda

Leganés, septiembre de 2015

Título: ALMACENES DE DATOS NOSQL, ESTUDIO DE LA TECNOLOGÍA
Autor: Ana María Lorente Borox
Director: Ana María Iglesias Maqueda

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día ___ de _____ de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

Se me hace bastante complicado escribir estas líneas ya que seguramente me faltará gente por nombrar, gente que merece todo mi agradecimiento. A todos ellos, gracias.

Con mi familia nunca tendré las palabras suficientes para agradecer todo lo que me han ayudado y apoyado para llegar hasta aquí. A mis padres, que están siempre, incondicionalmente. Tanto en los buenos momentos compartiendo mis alegrías como en los malos aguantándome (tarea bien difícil) y dándome su mejor apoyo y consejo. Por confiar también en mí y darme la libertad de tomar mis propias decisiones y saber que pase lo que pase seguirán al pie del cañón. A mi hermano, también por aguantarme y por recordarme como "esa persona que vive en casa y que no sale de su habitación". A ver si ahora que acabo salgo más y comienzo a ver mundo. También por su "peculiar" sentido del humor que utiliza para animarme siempre que lo necesito.

A mis abuelos, por enseñarme que con una dura vida de trabajo y esfuerzo las cosas siempre salen adelante. A mis tíos y primos, a todos ellos por animarme y darme siempre un buen consejo.

A toda la gente que he conocido en la universidad, desde que comencé los estudios. Me llevo grandes amistades y un montón de experiencias vividas. A Javier, Miguel, Nerea, Epi, David, Álvaro, Marta, Patricia, Geoconda, y un montón de gente más. A todos vosotros gracias por todos los momentos compartidos en la universidad. Me llevo el recuerdo de unos años llenos de alegrías y de mucho trabajo y esfuerzo. De los "maravillosos" días haciendo prácticas en el edificio 4 y los geniales desayunos en la cafetería.

A mis amigas, por animarme siempre que lo he necesitado y estar ahí siempre dispuestas a tomarse unas cañas conmigo y escucharme.

También dar las gracias a Ana Iglesias por guiarme en este proyecto y darme las claves necesarias para poder desarrollarlo.

Gracias.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS

1.1. INTRODUCCIÓN

En los últimos años el volumen de datos que se genera en Internet ha crecido a una velocidad sin precedentes. Cada minuto que pasa se realizan millones de búsquedas, creaciones de miles de webs y envíos de cientos de miles de tuits y de correos electrónicos, entre otros. Todo esto en cuestión de uno o dos minutos. La información almacenada está creciendo exponencialmente, lo que supone un problema en la adaptación de los sistemas de almacenamiento por parte de algunas compañías y organizaciones. Aquellas que no sepan adaptarse a estos cambios estarán condenadas al fracaso.

En la sociedad de hoy en día el cliente o usuario final tiene una gran capacidad de decisión y de influencia en su entorno. Una capacidad que no había tenido nunca antes. Es muy importante pararse a pensar y detectar qué es lo que quiere la sociedad y hacia donde evoluciona su demanda. El cliente es una figura clave; hoy en día además de conocer los datos de carácter general de un usuario ya se quiere predecir qué es lo que desea incluso antes de que él lo sepa. Para ello se evalúan todas las posibilidades de gustos o aficiones del cliente y en función de la utilización de estadísticas, éstas calculan cómo se le puede ofrecer un producto o un servicio ajustado a sus demandas y gustos.

En esto se basa el Big Data, en la gestión y análisis de grandes volúmenes de datos que se generan constantemente para sacar conclusiones de ellos.

Este gran cambio tecnológico, producido entre otros motivos con la llegada de la web 2.0 y por ende el comienzo de la generación y compartición de ingentes cantidades de información, hace que las empresas y organizaciones requieran de sistemas de almacenamiento que respondan a las necesidades del cliente. Pero no hay que olvidar que ya existen sistemas de almacenamiento previos a Big Data que han sabido proporcionar un servicio de almacenamiento para las necesidades que se presentaban en tiempos anteriores.

Es por ello muy importante saber en qué casos una organización o empresa requiere de la utilización de nuevas tecnologías o la reutilización de las tecnologías ya existentes.

Este Trabajo Fin de Grado intenta plasmar algunas de las diferencias entre los Sistemas tradicionales de almacenamiento y los sistemas nuevos que han surgido a partir de Big Data. Como en todo, dependiendo del problema a tratar, se decidirá si un almacén de datos es mejor para unas necesidades específicas que otro. Esto último es algo muy importante a tener en cuenta. Las nuevas tecnologías se crean para solucionar problemas que las antiguas no pueden cubrir. Ello no implica que tengan que sustituir completamente a los sistemas que ya existen. Dependiendo de las necesidades y de la organización de la información se verá la eficiencia de unos sistemas frente a otros.

1.2. OBJETIVOS

Los objetivos que se pretenden conseguir en este proyecto se presentan a continuación:

- Exposición y explicación de los Sistemas de almacenamiento tradicionales. Para ello se recorrerá a lo largo del tiempo el origen de los sistemas según se han ido generando nuevas necesidades.
- Explicación del término "Big Data". Presentación de las características de este fenómeno.
- Presentación de las tecnologías NoSQL, como nuevos sistemas de almacenamiento que intentan resolver algunos de los problemas generados por el concepto "Big Data".
- Comparativa entre un almacén de datos tradicional y un sistema de almacenamiento NoSQL. Ver en qué casos es más propicio utilizar unos sistemas frente a otros. Demostrar que la creación de nuevas tecnologías no implica sustituir las ya existentes; que todo depende del problema a tratar.

Además de estos objetivos generales, este Trabajo Fin de Grado se pretende presentar como un proyecto, para el cuál se ha elaborado una planificación, un análisis de costes y se ha seguido una metodología en la que se lleva a cabo toda la elaboración del proyecto.

1.3. ESTRUCTURA DEL DOCUMENTO

El presente documento se presenta en 5 capítulos, cuyo resumen se comenta a continuación:

En el capítulo 1 "*Introducción y Objetivos*" se realiza una introducción y se establecen los objetivos a lograr a lo largo del proyecto.

En el capítulo 2 "*Gestión del Proyecto"* se realiza una planificación y un análisis de costes, además de explicar las fases de desarrollo del proyecto.

En el capítulo 3 "*Estado del Arte"* se presentan en primer lugar los Sistemas tradicionales de almacenamiento. Después se introduce el concepto de "Big Data" y finalmente se presentan las tecnologías NoSQL.

En el capítulo 4 "*Comparativa"* se lleva a la práctica un dominio de datos escogido que se representa con una base de datos relacional y con una NoSQL. A lo largo de este capítulo se realiza un Análisis, Diseño, Implementación y Pruebas. Al final de este capítulo se muestran las conclusiones obtenidas de este proceso llevado a cabo.

En el capítulo 5 "*Conclusiones y Trabajos Futuros"* se detallan las conclusiones que se han obtenido a partir de la realización de este TFG. Igualmente se comentan algunas líneas futuras que pueden seguirse como continuación a este proyecto.

En último lugar se encuentra el Glosario de términos, seguido por la bibliografía y los Anexos. En el Anexo I se muestra el Diagrama de Gantt de la planificación final, en el Anexo II el detalle del Análisis de Costes. En el Anexo III el Manual de instalación de MongoDB y por último, en el Anexo IV el Manual de uso de MongoDB.

CAPÍTULO 2. GESTIÓN DEL PROYECTO

2.1. FASES DEL DESARROLLO

Este Trabajo Fin de Grado se descompone en dos fases:

- **Primera Fase: definición e introducción de:**
    - Sistemas de Almacenamiento tradicionales.
    - Big Data. Presentación y explicación de problemas.
    - Introducción a las bases de datos NoSQL.
- **Segunda Fase: presentación de un caso práctico. Elección de un dominio de datos**
    - Aproximación y tratamiento del problema con un sistema de almacenamiento tradicional.
        - Análisis
        - Diseño
        - Implementación
        - Pruebas
    - Aproximación y tratamiento del problema con una tecnología NoSQL.
        - Análisis
        - Diseño
        - Implementación
        - Pruebas
    - Conclusiones sobre ambos procesos

El progreso y dependencia de estas fases se muestra en la siguiente figura:

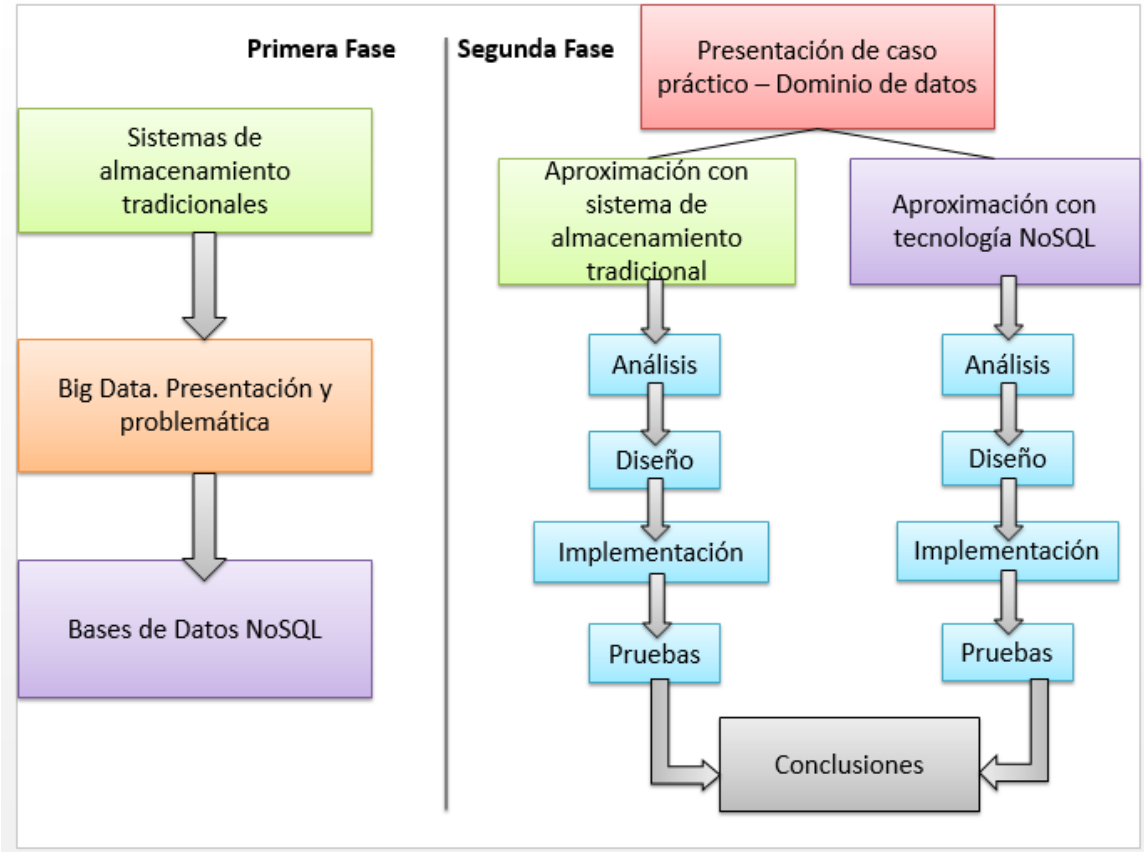

Figura 1. Fases del proyecto

2.2. GESTIÓN SOFTWARE – MEDIOS EMPLEADOS

Los medios empleados para la realización de este Trabajo Fin de Grado han sido los siguientes:

- Editor de textos: Microsoft Word 2013
- Editor de imágenes: Microsoft Power Point 2013
- Herramientas para la generación de diagramas:
    - Gantt Project v2.6.6
- Herramientas para el tratamiento de datos en el caso práctico
    - Hoja de cálculo: Microsoft Excel 2013
    - Notepad ++
- Herramientas de gestión de software – Base de datos relacional
    - Oracle Database 11g Express Edition
- Herramientas de gestión de software – Almacén de datos documental
    - MongoDB

2.3. PLANIFICACIÓN INICIAL

Para realizar la planificación inicial en primer lugar se ha buscado cuál es la duración, según la normativa, que tiene un Trabajo Fin de Grado (TFG). Se ha calculado que el TFG, equivalente a 12 créditos ECTS, tendrá una duración aproximada de 300 horas, ya que cada crédito ECTS supone una dedicación de 25 horas.

Teniendo en cuenta este total de horas, el trabajo se distribuirá a lo largo del tiempo indicado, poniendo como fecha de comienzo el 16 de marzo de 2015. Desde esta fecha hasta la fecha de entrega (23-27 septiembre) hay 28 semanas, por lo que inicialmente se calcula que se deberá trabajar una media de 11 horas semanales.

La **planificación** que se ha elaborado es **adaptativa** en lugar de predictiva. De este modo la planificación inicial podrá variar según avance el proyecto. Estas variaciones pueden ser cambios por posibles complicaciones o nuevas exigencias en el desarrollo del proyecto. Se ha decidido llevar a cabo una planificación adaptativa ya que era muy probable que el proyecto sufriera modificaciones y retrasos sobre la marcha, debido entre otros factores a la falta de experiencia previa en este tipo de proyectos.

A continuación se introducirá la planificación de las tareas a realizar a lo largo del proyecto. Se ha dividido el trabajo a realizar en 3 tareas básicas. En primer lugar **el estudio inicial** (y documentación) sobre la materia, donde se comienza por definir cuál es el problema, qué recursos hay actualmente y qué posibles soluciones se pueden aportar. Después, la realización de la **comparativa entre las dos tecnologías escogidas** ya introducidas en el primer punto y por último la **documentación aportada**, posterior a la comparativa.

De esta forma, para las tres tareas que se han definido, se estima que la duración sea la siguiente:

Tareas	Horas
Estudio inicial	80
Comparativa entre las tecnologías	90
Documentación	130
Total	**300**

Tabla 1. Planificación inicial de tareas

Estas tareas iniciales, mostradas en un diagrama de Gantt, se reflejan en la siguiente figura:

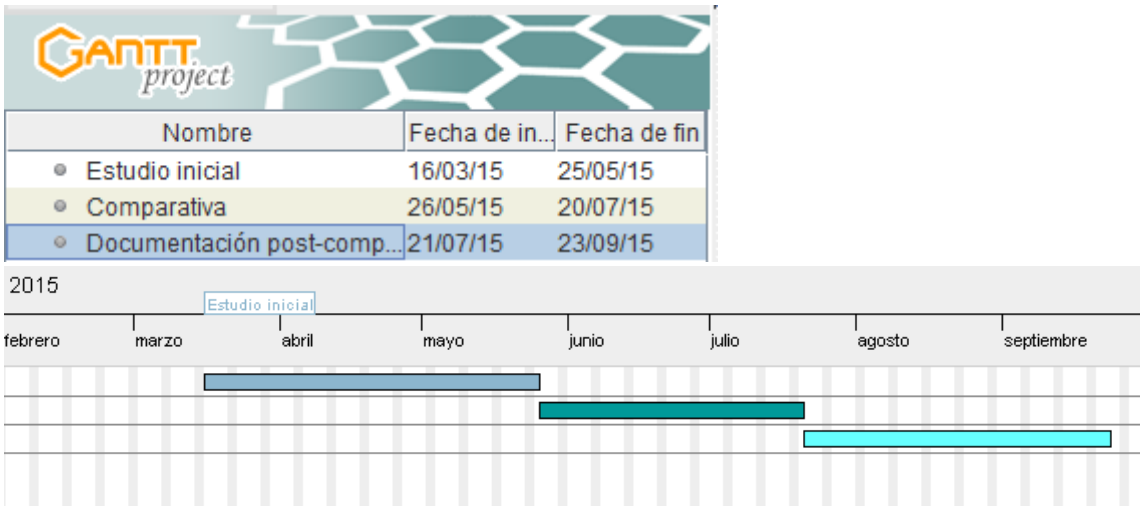

Figura 2. Diagrama de Gantt inicial

2.4. EJECUCIÓN FINAL Y ANÁLISIS DE COSTES

En este apartado se presentará la ejecución final del proyecto; la planificación temporal y las explicaciones de las desviaciones frente a lo planificado inicialmente. También se hará un análisis de costes de este proyecto.

2.4.1. PLANIFICACIÓN FINAL

Finalmente tal y como se indicaba, la planificación inicial se ha ido adaptando a las necesidades y requerimientos que se iban presentando a lo largo del proyecto. Ya que la planificación inicial elaborada anteriormente es muy escueta, a continuación se definen las tareas más específicas que se han ido realizando.

Se ha dividido la planificación en tres fases, igualmente:

- La primera fase, correspondiente al estudio inicial de la situación, la problemática que surge y las tecnologías que existen. Esta primera fase corresponde al "Estado del Arte", y se divide en tareas como:
    - Estudio de los sistemas tradicionales de almacenamiento
    - Estudio de Big Data
    - Estudio de las tecnologías NoSQL
- La segunda fase tiene que ver con la comparativa que se realiza entre las dos tecnologías. En esta fase se incluyen las operaciones realizadas con las dos tecnologías, como:
    - Estudio del dominio de datos
    - Análisis del dominio de datos con herramienta tradicional y NoSQL
    - Diseño del dominio de datos con herramienta tradicional y NoSQL
    - Implementación del dominio de datos con herramienta tradicional y NoSQL
    - Pruebas del dominio de datos con herramienta tradicional y NoSQL
- La tercera fase y última fase está relacionada con la documentación posterior a la comparativa. Una vez realizada la comparativa con las dos tecnologías hay que plasmar los resultados en el documento.

El diagrama de Gantt referente a esta Planificación Final se encuentra en el *Anexo I: Diagrama de Gantt.*

En la figura incluida en el *Anexo I* se puede observar desviaciones frente a la planificación inicial. La desviación más importante es la que tiene que ver con la realización de la comparativa, que ha llevado más tiempo del que se pensaba, lo que ha hecho retrasar considerablemente el tiempo de elaboración de la documentación. A continuación se muestra una tabla con las horas finales dedicadas a cada tarea.

Tareas	Horas finales
Estudio inicial	
Sistemas tradicionales de almacenamiento	25
Estudio de Big Data	35
Estudio NoSQL	35
Total Estudio Inicial	**95**
Comparativa	
Estudio del dominio	15
Análisis	30
Diseño	35
Implementación	50
Pruebas	10
Total Comparativa	**140**
Documentación	90
Total Documentación	**70**
Total	**305**

Tabla 2. Tareas y horas finales dedicadas

2.4.2. ANÁLISIS DE COSTES

En este apartado se calculará el presupuesto necesario para llevar a cabo este proyecto. Se tendrán en cuenta varios parámetros:

- Por un lado se tendrá en cuenta el **gasto de personal**, representado por el coste del tutor y del autor del TFG. Para realizar el cálculo de este coste se utilizarán como guía los salarios anuales de un ingeniero senior y de un ingeniero junior, obtenidos en [1].

Según la referencia, para el ingeniero senior se elegirá el rol de "Administrador de BBDD senior", que cobra unos 45.000€ anuales. Para este Trabajo Fin de Grado se calcula que desde el comienzo hasta el fin se realizará una reunión cada dos semanas, de aproximadamente una hora de duración, por lo que se trabajará un total de 12 horas. La dedicación de las 12 horas por tanto equivale a unos 324,324€.

En cuanto al ingeniero junior, se elegirá el rol de "Administrador de BBDD" este cobra de media unos 24.000€ anuales. Este recurso trabajará según lo establecido en el apartado "*2.3. Planificación inicial*", un total de 300 horas, lo que equivale a 4324,324€.

- Aparte de los gastos de personal se tendrá en cuenta el **coste del uso de las diferentes herramientas** que han sido necesarias para desarrollar el proyecto. Para el cálculo de este coste se tendrá en cuenta el coste amortizado del ordenador de desarrollo y del paquete ofimático para realizar la documentación. En total se contará un total de 99.55€ en estas características.

Finalmente sumado a esto un 20% de tasa de costes indirectos, el total asciende a 5697.84€. Este importe total está desglosado en el *Anexo II: Análisis de Costes.*

CAPÍTULO 3. ESTADO DEL ARTE

En este capítulo se pretende hacer un repaso de la historia de la evolución de las bases de datos y sistemas de almacenamiento para así poder comprender la problemática a la que nos enfrentamos.

Desde que surge la necesidad de recopilar información se han desarrollado diferentes sistemas de almacenamiento que han cambiado a lo largo del tiempo con el objetivo de adaptarse a las nuevas necesidades y exigencias del mercado. Es por ello que el concepto de base de datos no es un invento repentino, es algo que ha evolucionado a través de la experiencia.

El crecimiento del sector tecnológico en los últimos años ha provocado en el entorno socioeconómico un aumento desproporcionado en el volumen de datos e información. Ello requiere la creación e implantación de nuevos sistemas de almacenamiento que se ajusten a las nuevas necesidades. Las grandes empresas y organizaciones demandan cada vez más el uso de nuevas tecnologías para almacenar su información y tratar su volumen e inteligencia de negocio. Se pretende que esta información se gestione de una manera rápida, fiable y eficaz. Esto también se aplica para las organizaciones de usuarios, quienes demandan de manera creciente el uso de nuevas herramientas que resuelvan los problemas derivados del aumento del volumen de información y que éstas se ajusten a las características de las nuevas tecnologías.

A lo largo de este capítulo se hará un resumen de los principales sistemas de almacenamiento desde los años cincuenta hasta la actualidad. El orden que se seguirá en los siguientes apartados atiende a la evolución del tiempo y a las necesidades y problemas que surgían. Se ha clasificado de forma general dos movimientos o corrientes: los sistemas tradicionales de almacenamiento y el Big Data.

A continuación se muestra un eje cronológico (en la *Figura 1. Historia de la evolución de las Bases de Datos 1950-actualidad*) que empieza en la década de 1950 y llega hasta 2015 y que servirá de guía para seguir la evolución de los sistemas de almacenamiento. En los siguientes apartados se explicará en detalle los sistemas de almacenamiento. Mucho antes del comienzo, más concretamente en 1880, ya surgió el primer problema de almacenamiento; cuando Herman Hollerith se halló en la tarea de completar el censo de trece millones de norteamericanos. Fue entonces cuando inventó las tarjetas perforadas como técnica de almacenamiento. Este invento se considera un sistema precursor de las bases de datos, considerada la técnica más moderna de almacenamiento [2].

- **Historia de la evolución de las Bases de Datos 1950-actualidad**

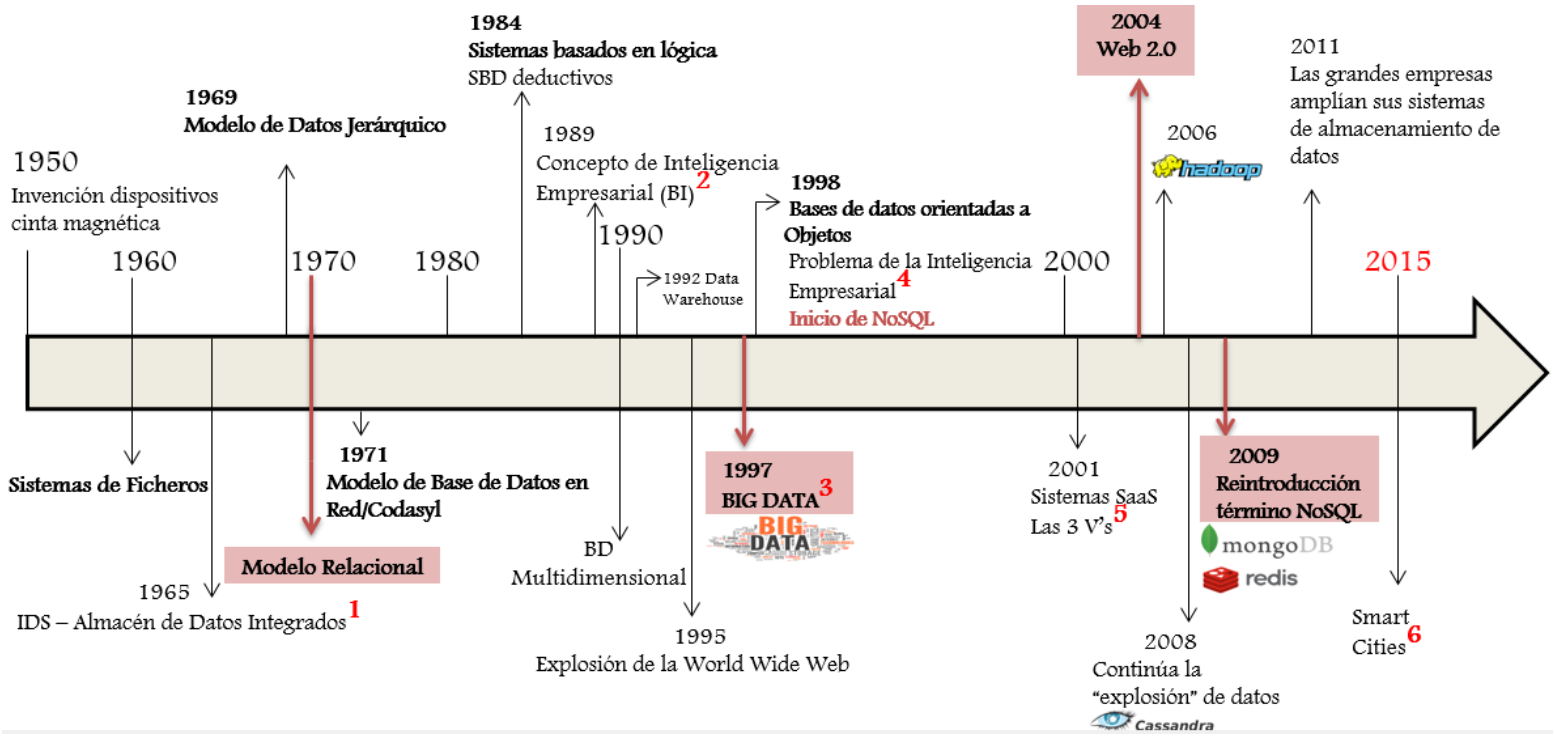

Figura 3. Historia de la evolución de las Bases de Datos 1950-actualidad

1. Integrated Data Store (IDS) o Almacén de Datos Integrados, se utilizó para crear grandes ficheros integrados, que pueden ser compartidos por cierto número de aplicaciones. [2]

2. Inteligencia Empresarial o Business Intelligence (BI). Howard Dresner lo definió como los "conceptos y métodos que mejoran la toma de decisiones de negocio mediante el uso de sistemas de apoyo basados en datos reales". [3]

3. El término "Big Data" se emplea por primera vez en 1997 en un artículo de los investigadores de la NASA Michael Cox y David Ellsworth. Ambos afirman que el ritmo de crecimiento de los datos empezaba a ser un problema para los sistemas informáticos actuales. [4]

4. A finales de los 90, muchas empresas creían que sus sistemas de extracción de datos no funcionaban. Los trabajadores eran incapaces de encontrar respuestas y de acceder a los datos que necesitaban de las búsquedas. Los departamentos informáticos eran responsables del 80% del acceso a BI. [5]

5. Volumen, Variedad y Velocidad. Tres conceptos que definen las dimensiones del Big Data. En el apartado "2.2.2. Características del Big Data" se habla de estas tres características y se añade una cuarta V.

6. Ciudad inteligente o Smart Cities se refiere al uso del análisis de información contextual en tiempo real para mejorar la calidad y el rendimiento de los servicios urbanos, reducir costes, optimizar recursos e interactuar de forma activa con los ciudadanos. Según estimaciones de Gartner habrá más de 1100 millones de dispositivos conectados y en uso en diversas ciudades en 2015, incluyendo sistemas de iluminado LED inteligentes, de monitorización de salud, cerraduras inteligentes y numerosas redes de sensores para detección de movimiento, estudio de contaminación atmosférica, etc. [6]

3.1. SISTEMAS TRADICIONALES DE ALMACENAMIENTO

Con el objetivo de adaptarse a las nuevas necesidades, a lo largo del tiempo han surgido diferentes sistemas de almacenamiento de información. Uno de ellos fue el **Sistema de Ficheros** (décadas 1950 y 1960), consistente en un conjunto de programas que gestionaban ficheros donde se almacenaban los datos. La definición y codificación de los datos se trataba dentro de los programas. Esto suponía un gran inconveniente a la hora de tratar grandes volúmenes de datos, ya que si se añadían nuevos elementos o campos al fichero, los programas que utilizaban ese fichero se tenían que modificar para tratar los nuevos elementos.

Con el paso del tiempo, este sistema de almacenamiento quedó obsoleto por problemas como la redundancia, inconsistencia, dependencia y escaso control y manipulación de los datos. Esto provocó la necesidad de separar los datos contenidos en los ficheros de los programas que los manipulaban, para que así un cambio en la estructura de los datos no afectara en los programas con los que se trabajaban. Surgió la idea general de tratar y estructurar los datos con independencia de los procesos o programas que los gestionaban.

Todo ello impulsó la aparición de las **bases de datos**, concepto utilizado vagamente a mediados de los años sesenta y afianzado a principios de los setenta. Llamadas en adelante BBDD, estas son grandes almacenes de datos cuya descripción y definición se establece al principio y cuyos datos, relacionados entre sí, son independientes de las aplicaciones y pueden ser accedidos de forma simultánea por varios usuarios.

Las BBDD son sistemas que por sus características solucionan las limitaciones de los Sistemas de Ficheros. No obstante, las bases de datos también presentan una serie de inconvenientes, tales como los problemas de instalación, donde el coste puede ser elevado si se necesitan muchos recursos y se definen muchas necesidades, problemas de personal en lo que se refiere al desarrollo o administración de la BBDD, ya que es importante que el personal esté formado en el tratamiento de la BBDD. Problemas de implantación del sistema, que en ocasiones puede ser un proceso lento, y por último, problemas con la estandarización, ya que aunque actualmente hay estándares cuyo uso es muy frecuente, hay grandes diferencias entre gestores de BBDD.

A continuación se muestra una tabla comparativa entre los Sistemas de Ficheros y las bases de datos, con el objetivo de visualizar las principales diferencias [7].

	Sistemas de Ficheros	Bases de Datos
Dependencia Datos-Tratamiento	Sí	No
Orientado a	Procesos	Datos
Redundancia de Datos	Sí, provocando inconsistencias	Poca
Interrelaciones	Escasas	Muchas, refleja el mundo real
Propietario	Uni-usuario	Multi-usuario
Datos y semántica	Separados (no normalizado)	Juntos (Normalizado)
Almacenamiento	Se requiere más espacio	Se necesita menos espacio
Instalación	No requiere	Sí, costosa
Personal	No especializado	Especializado
Implantación	Inmediata	Lenta y costosa
Rentabilidad	Inmediata	Largo plazo

Tabla 3. Comparativa Sistemas de Ficheros y Bases de Datos

Dentro de las bases de datos se pueden distinguir varios modelos. En los siguientes apartados se van a detallar algunos de ellos, que son: el **modelo jerárquico**, el **modelo en red**, el **modelo relacional**, las **basadas en lógica** y **las bases de datos orientadas a objetos**.

3.1.1. BASES DE DATOS JERÁRQUICAS

Las bases de datos jerárquicas estructuran la información mediante jerarquías; la relación entre las entidades se organiza en forma de árbol, siendo ésta del tipo padre/hijo. La representación de los árboles se hace de manera invertida, con la raíz hacia arriba y las hojas y siguientes niveles hacia abajo [8].

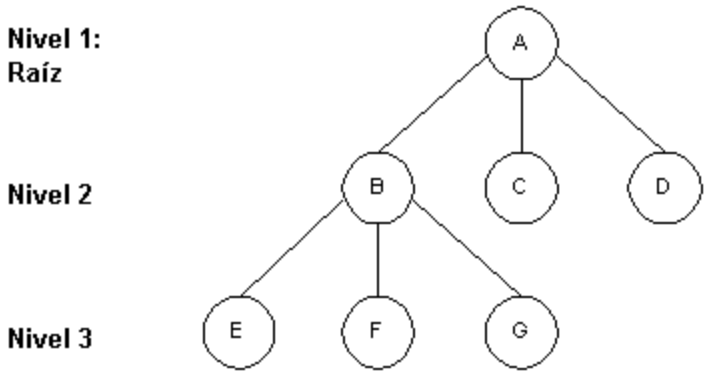

Figura 4. Estructura de árbol jerárquico

Como ejemplo de sistema gestor de bases de datos[1] jerárquicas se puede citar el IMS de IBM Corporation[2].

Este modelo presenta ciertas limitaciones por su estructura de datos, donde un registro hijo tiene un solo padre.

3.1.2. BASES DE DATOS EN RED

El modelo de **bases de datos en red/Codasyl** proporciona una mejora con respecto al modelo jerárquico, ya que permite que un nodo tenga varios padres y varios hijos, por lo que los conjuntos de datos están interconectados entre sí por medio de una red o grafo.

[1] Se explica en qué consiste en el apartado "2.1.5. Sistema Gestor de Bases de Datos – SGBD".

[2] Information Management System (IMS) de IBM es un sistema gestor de bases de datos jerárquicas y gestor transaccional creado en 1969 a partir del Programa Apolo, cuyo objetivo era inventariar la lista de materiales del cohete Saturno V y de la nave Apolo. IMS se sigue utilizando actualmente con nuevas funciones y características que han sido añadidas con el objetivo de responder a la demanda del cliente.

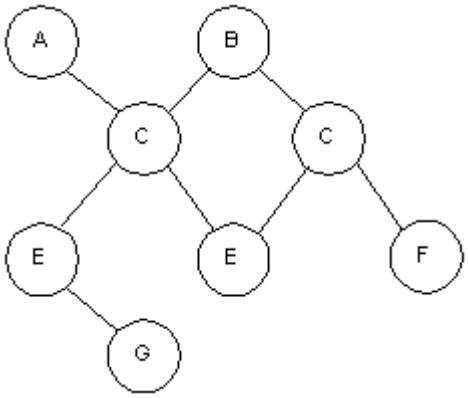

Figura 5. Estructura de datos en red

Un ejemplo de este otro tipo de base de datos es el sistema gestor de bases de datos en red IDMS[3].

3.1.3. BASES DE DATOS RELACIONALES

El modelo **relacional** es uno de los más utilizados en la actualidad y resuelve algunos problemas de los otros modelos:

Está basado en la lógica de predicados y la teoría de conjuntos, lo que implica que se puede utilizar tanto el álgebra como el cálculo relacional para operar con los datos almacenados. El álgebra para describir la forma de realizar una consulta y el cálculo para obtener los valores que se desean devolver.

Una base de datos relacional es un conjunto de una o varias tablas formadas por tuplas o registros que se relacionan entre ellas mediante claves, tanto primarias como ajenas. Los datos están conectados por medio de relaciones entre las tablas que los almacenan.

[3] Integrated Database Management System (IDMS) es un sistema gestor de bases de datos en red desarrollada por una compañía software llamada Cullinet, que ahora forma parte de la compañía Computer Associates. Este sistema fue desarrollado durante la década de 1970 y sigue siendo utilizado actualmente como solución de bases de datos mainframe para grandes empresas.

TABLA DE EMPLEADOS			
ID	NOMBRE	EMPRESA	AÑOS
1	Miguel Díaz	Accenture	24
2	Javier Fernández	BBVA	23
3	Natalia Sanz	Iberia	23
4	Ana Lorente	Santander	24

TABLA DE EMPRESAS		
CÓDIGO	UBICACION	EMPRESA
1841	Valencia	Vodafone
1684	Barcelona	Repsol
2187	Madrid	BBVA
2646	Madrid	Samsung

Tabla 4. Tabla en el modelo de datos relacional

Edgar Frank Codd, científico informático que trabajaba para IBM, fue el que estableció y definió las bases de este modelo en el año 1970. Este científico continuó trabajando sobre las teorías de modelado de datos y definió las tres primeras Formas Normales[4], aplicadas para la normalización de sistemas de bases de datos. La Forma Normal de Boyce-Codd (o abreviada, FNBC) lleva los apellidos de Raymond Boyce y Edgar Frank Codd en su honor. Codd también acuñó las siglas **OLAP** (On-Line Analytical Processing), una solución utilizada en el campo de la inteligencia empresarial (o Business Intelligence) cuyo objetivo es gestionar y agilizar el tratamiento de grandes cantidades de datos [9]. De igual forma Codd redactó 12 reglas, conocidas como *Las 12 reglas de Codd* que definen los requisitos que requiere y debe cumplir un Sistema Gestor de base de datos (SGBD) relacional. [10]

Esta forma de organizar los datos y su posterior manipulación a través del lenguaje matemático del cálculo y álgebra relacional y junto con las Reglas de Integridad y Formas Normales, marca una diferencia muy significativa entre este sistema y los anteriores.

Algunos ejemplos de sistemas gestores de bases de datos relacionales, SGBDR son: Oracle, MySQL o DB2, entre otros.

[4] Las Formas Normales en una base de datos definen y delimitan los problemas que se pueden presentar a la hora de crear las tablas, tales como anomalías lógicas e inconsistencias. Especifican una serie de condiciones que deben cumplir las tablas para evitar estos problemas.

3.1.4. BASES DE DATOS BASADAS EN LA LÓGICA

Este modelo utiliza la lógica matemática para deducir información que no está almacenada de forma explícita en la base de datos. Es muy utilizado en el campo de la inteligencia artificial.

El estímulo principal y surgimiento de este sistema fue a partir de la publicación en 1984 de un artículo de Raymond Reiter. [11]

La base de datos se contempla como un conjunto de axiomas y la ejecución de una consulta se considera como la demostración de que una fórmula es una consecuencia lógica de esos axiomas. [12]

Un ejemplo es Datalog, un lenguaje con el que se pueden formular los axiomas deductivos, utilizado para sistemas de bases de datos deductivas[5].

También se puede citar como ejemplo el lenguaje de programación ABAP.[6]

Una vez se han repasado las bases de datos más importantes hasta el momento (década de 1980), a continuación se explica en qué consiste un Sistema Gestor de Base de Datos. En los ejemplos proporcionados anteriormente de las distintas bases de datos se mencionaba este término. Veamos en qué consiste.

3.1.5. SISTEMA GESTOR DE BASES DE DATOS- SGBD

Para poder almacenar, modificar y obtener la información de una base de datos se necesita un Sistema Gestor de Base de Datos (SGBD), un conjunto de herramientas tales como programas o lenguajes que proporcionan los medios necesarios para modificar, añadir, borrar o consultar datos de la base a todos los niveles (usuarios, programadores, etc.).

Además de poder manipular y obtener la información almacenada en la base a todos los niveles, un Sistema Gestor de Base de datos, en adelante SGBD, también proporciona las herramientas necesarias para administrar el acceso de los usuarios y para recuperar la información en caso de que el sistema falle o sufra algún daño.

[5] Un Sistema de Bases de Datos Deductivo maneja la perspectiva según la teoría de las demostraciones de una base de datos, y en particular es capaz de deducir hechos adicionales a partir de la "base de datos extensional" aplicando a esos hechos *axiomas deductivos* o *reglas de inferencia especificados*. [12]

[6] Advanced Business Applicacion Programming (ABAP), lenguaje de programación propiedad de SAP utilizado para las Bases de Datos Lógicas y para programar muchos de los productos de SAP.

Los servicios que debe presentar un Sistema Gestor de Bases de Datos son:

- **Creación y definición de la base de datos**: en el SGBD se detalla el tipo de datos, su estructura y las relaciones y restricciones entre ellos. Toda esta información se almacena en el diccionario de datos.

- **Manipulación de los datos**: necesario para poder realizar consultas, actualizaciones, inserciones o borrado de los datos.

- **Acceso controlado a los datos**: el SGBD deberá estar dotado de las herramientas necesarias para preservar la seguridad e integridad de los datos. Por ejemplo, estableciendo controles de acceso a los usuarios.

- **Acceso compartido a la base de datos**: es necesario que el SGBD controle el acceso a los datos cuando accedan de forma concurrente varios usuarios, ya que se pueden producir cambios en los resultados de los datos esperados.

- **Mecanismos de recuperación**: en caso de accidente o fallo en el sistema, el SGBD deberá disponer de los mecanismos necesarios para recuperar la información. Por ejemplo realizando copias de seguridad de forma periódica.

Para que el administrador de la base de datos pueda definir los datos que componen la BD, establecer relaciones entre ellos, detallar su estructura, manipularlos y controlar los accesos de los usuarios, entre otros parámetros, los SGBD ofrecen **lenguajes** que permiten estas funciones, y son:

- **Lenguaje de Descripción (LDD)**: el SGBD ha de permitir especificar los elementos de los datos tales como el esquema, las vistas de los usuarios y las estructuras, así como las interrelaciones entre ellos y las reglas de validación semántica[7].

- **Lenguaje de Manipulación (LMD)**: el SGBD ha de posibilitar las operaciones de lectura y actualización de los contenidos de la base. Este lenguaje de manipulación es utilizado por los usuarios para realizar inserciones, borrados, modificaciones o consultas.

- **Lenguaje de Utilización (LCD)**: el SGBD tiene que disponer de un conjunto de herramientas para que el administrador pueda realizar su cometido. Se les llama herramientas de desarrollo. Un ejemplo es el lenguaje PL/SQL[8], que crea procedimientos para manipular los datos almacenados en la base.

[7] Las reglas de validación semántica son un conjunto de restricciones que permiten o deniegan el almacenamiento de determinados valores en una base de datos.

[8] PL/SQL (Procedural Language/Structured Query Language) es un lenguaje de procesamiento procedimental implementado por Oracle que tiene por objetivo interactuar con la base de datos.

Después del repaso de las principales bases de datos y de la explicación de los sistemas gestores de bases de datos se puede concluir que, aunque en ocasiones se considere que el modelo de bases de datos relacional es el más importante (y a su vez el más conocido de todos ellos), éste no es una panacea. Después del modelo relacional han seguido apareciendo nuevos sistemas, ya que para ciertos problemas el modelo relacional y el resto presentan deficiencias, como por ejemplo, problemas en el tratamiento de la información o en el manejo de objetos complejos; problemas producidos por el cambio en los últimos años en el entorno tecnológico, económico y social.

Siguiendo el eje cronológico dibujado anteriormente, la "explosión" de la World Wide Web a mediados de los noventa provocó la necesidad de almacenar de forma masiva grandes volúmenes de información. En un principio se pensó en dos formas de abordar este problema: una de ellas era ampliar el modelo relacional y otra era desecharlo y sustituirlo por sistemas nuevos. Si bien es cierto que al principio se optó por la primera solución, aumentando la capacidad (en hardware) de las máquinas que soportaban las bases de datos relacionales, pronto se vio que esta solución no era suficiente. Se necesitaban crear sistemas nuevos que solucionaran eficazmente los problemas que surgían.

En los siguientes apartados se tratará esta problemática y se hablará de Big Data, en qué consiste, cómo surge y qué soluciones aporta.

3.2. BIG DATA

Como ya se ha comentado anteriormente, el término Big Data fue empleado por primera vez en 1997 ante la creciente cantidad de información a un ritmo sin precedentes y los futuros problemas que esto iba a provocar para los sistemas informáticos de entonces.

"Big Data" es uno de los conceptos que actualmente está más de moda en el mundo de la informática. No es fácil definir en qué consiste, ya que existe mucha información presente en artículos, blogs, periódicos y otros medios que hablan de este tema aportando respuestas muy variadas y diferentes puntos de vista, generando en ocasiones una cierta confusión.

En los siguientes apartados se intentará exponer las ideas y definiciones más importantes referentes a Big Data, con el objetivo de comprender en qué consiste y por qué esta tecnología tiene tanta importancia hoy en día.

3.2.1. QUÉ ES BIG DATA

Crecimiento de los datos: ¿Qué ocurre en un minuto en Internet?

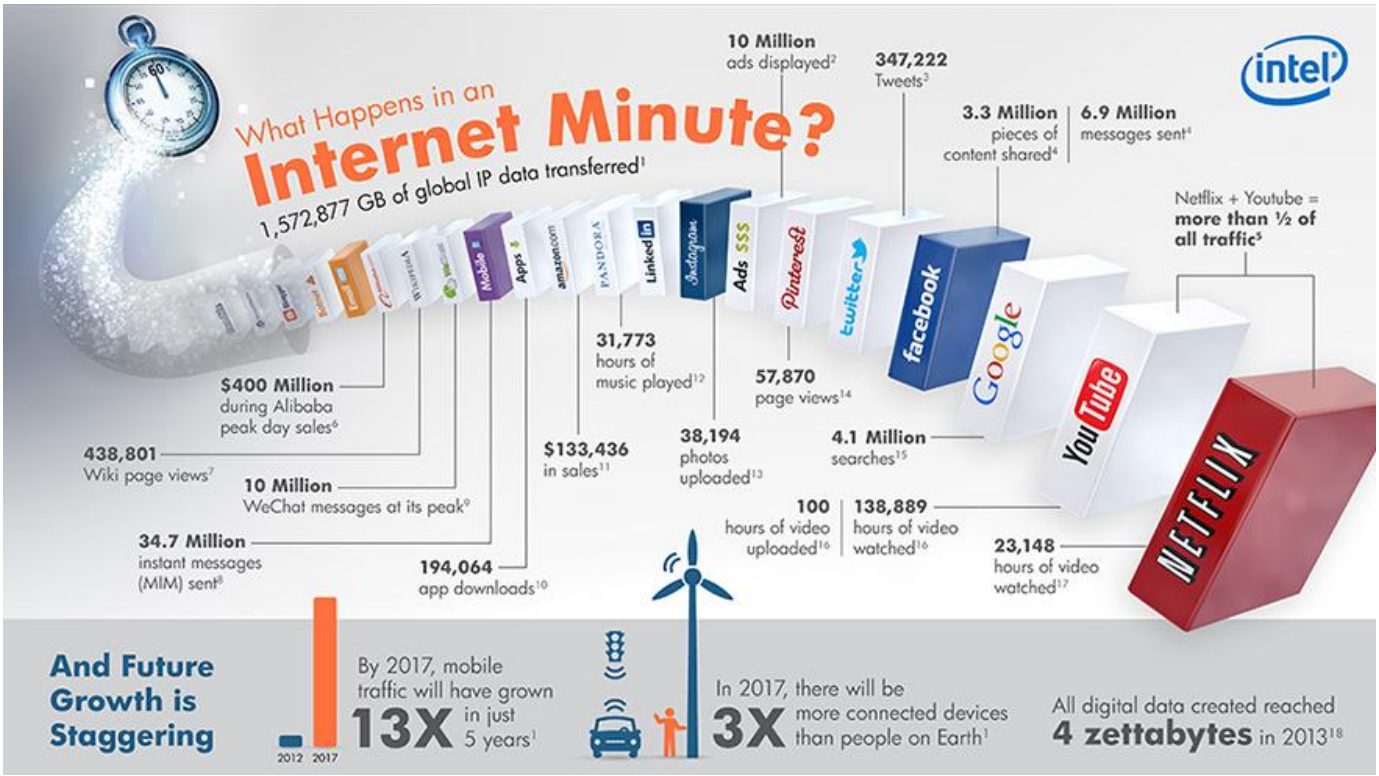

Figura 6. ¿Qué ocurre en un minuto en Internet? (imagen original de [17])

¿Qué es lo que se dice de Big Data? A continuación se plasman diferentes opiniones sobre este término:

Según [13] Big Data es el término que se emplea hoy en día para describir el conjunto de procesos, tecnologías y modelos de negocio que están basados en datos y en capturar el valor que los propios datos encierran. Esto se puede lograr tanto a través de una mejora en la eficiencia gracias al análisis de los datos (una visión más tradicional), como mediante la aparición de nuevos modelos de negocio que supongan un motor de crecimiento. Se habla mucho del aspecto tecnológico, pero hay que tener presente que es crítico encontrar la forma de dar valor a los datos para crear nuevos modelos de negocio o de ayudar a los existentes

Según [14] Big Data es alto volumen, alta velocidad y alta variedad de información que demanda innovadoras y rentables formas de procesamiento de la información para mejorar la comprensión y la toma de decisiones.

Según [15] en términos generales podríamos referirnos como a la tendencia en el avance de la tecnología que ha abierto las puertas hacia un nuevo enfoque de entendimiento y toma de decisiones, la cual es utilizada para describir enormes cantidades de datos (estructurados, no estructurados y semi estructurados) que tomaría demasiado tiempo y sería muy costoso cargarlos a un base de datos relacional para su análisis. De tal manera que, el concepto de Big Data aplica para toda aquella información que no puede ser procesada o analizada utilizando procesos o herramientas tradicionales.

Según [16] Big Data es un término popular usado para describir el crecimiento exponencial y la disponibilidad de los datos, tanto estructurados como no estructurados. Big Data es tan importante para los negocios como para la sociedad, tal y como Internet ha evolucionado. ¿Por qué? Más datos pueden conducir a análisis más precisos.

Estas son algunas de las muchas definiciones que se pueden encontrar referentes a Big Data. A partir de ellas se podría resumir y definir el término Big Data como:

Big Data se refiere a la gran cantidad de información tanto estructurada como no estructurada que se genera por toda la sociedad a través de nuestra actividad en Internet. Esta información puede ser procesada o analizada para, por ejemplo, encontrar patrones y sacar conclusiones a partir de su análisis.

3.2.2. CARACTERÍSTICAS DEL BIG DATA

Como ya se ha mostrado en la imagen del anterior apartado, y a modo de resumen, en **un minuto** en internet se generan:

- 4,1 millones de búsquedas en Google

- 3,3 millones de piezas de contenido compartido en Facebook
- 347.222 tuits en Twitter
- 194.064 descargas de aplicaciones
- 138.889 horas de vídeo vistas en Youtube

Y esto sigue adelante, ya que según [17] en 2013 se crearon cuatro zettabytes[9] de datos por dispositivos digitales. En 2017, se espera que el número de dispositivos conectados alcance tres veces el número de personas en la Tierra.

Esto nos da una idea de la grandísima cantidad de información que se produce continuamente. Las tecnologías Big Data tienen como objetivo sacar el valor que guardan todos estos datos para así extraer conocimiento e información y traducir ese conocimiento e información en nuevos servicios.

La característica que se asocia con mayor frecuencia a Big Data es el volumen, pero hay otros tres elementos más que completan la definición de Big Data: Variedad, Velocidad y Veracidad. Estas cuatro características se conocen como las **4 V del Big Data**.

En la siguiente imagen del informe "Analytics: el uso de big data en el mundo real" de IBM, se muestran estas 4 características:

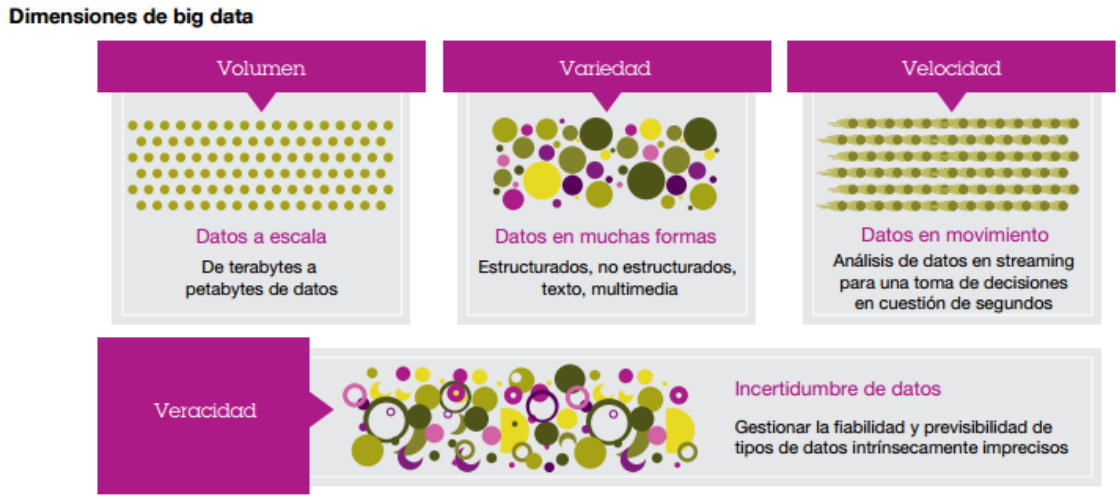

Figura 7. Dimensiones de Big Data (imagen original de [18])

- **Volumen**

  Se refiere a la cantidad de datos. Como ya se ha comentado, el volumen de datos que se genera aumenta continuamente, a un ritmo sin precedentes y lo va a

[9] La unidad de almacenamiento *zettabyte* equivale a 10^{12} GB, 10^{21} bytes.

seguir haciendo en el futuro. Debido a esta cantidad masiva de datos los sistemas tradicionales de bases de datos comienzan a tener problemas de rendimiento al intentar tratar con altos volúmenes de información.

- **Variedad**

  Diferentes tipos de datos provenientes de diferentes fuentes. Este término tiene que ver con cómo se gestionan los múltiples tipos de datos, ya sean estructurados, no estructurados o semiestructurados. Las organizaciones necesitan gestionar y analizar multitud de datos diferentes, provenientes de diversas fuentes. Los datos que se generan y que se van a procesar presentan varios formatos, como por ejemplo, datos de texto, vídeo, audio, datos de sensores, datos web o tuits, entre otros muchos tipos.

- **Velocidad**

  Datos en movimiento. La velocidad se refiere a la rapidez y capacidad en unidad de tiempo de generar, procesar y analizar los datos. La velocidad de procesamiento toma vital importancia, ya que el tiempo de espera entre el momento en el que se crean los datos y el momento en el que se captan y son accesibles puede ser inasumible para modelos de bases de datos antiguos. Big Data intenta solucionar estos problemas donde el tiempo resulta fundamental y donde estos datos deben analizarse en tiempo real, por ejemplo para detecciones de fraudes o procesos en streaming.

- **Veracidad**

  Se refiere a la incertidumbre de los datos. Si son muy fiables o no. El objetivo de Big Data es conseguir unos datos valiosos, pero en algunos casos es muy complicado eliminar la imprevisibilidad de ciertos datos, referentes por ejemplo al tiempo, la economía o datos basados en toma de decisiones. Planificar y reconocer la incertidumbre se encuentra entre los grandes objetivos de Big Data.

En resumen, Big Data se puede explicar a partir de la combinación de estas cuatro características. Big Data surge como remedio, frente a los modelos de bases de datos antiguos, a la hora de tratar grandes volúmenes de datos (Volumen), de diferentes y provenientes de varias fuentes (Variedad), en un tiempo adecuado (Velocidad) y donde la fiabilidad (Veracidad) se convierte en un punto fundamental.

Big Data crea nuevas oportunidades para que las organizaciones puedan manejar los datos dentro del mercado digitalizado en el que vivimos. Permite que se cambie la forma en la que se interactúa con los clientes y cómo se les proporcionan diferentes servicios. Todo ello posibilita la transformación de las organizaciones a la hora de utilizar las nuevas tecnologías.

Un punto importante a tener en cuenta es que Big Data es una tecnología en constante cambio y crecimiento. Según [18], "la opinión generalizada es que nos encontramos en las primeras etapas de la adopción empresarial de Big Data". Lo que quiere decir que las organizaciones, el mercado y sus necesidades están en un cambio constante, por lo que es imprescindible un proceso de progresión y actualización continua para no quedarse obsoleto.

Entonces, ¿cómo se pueden almacenar los datos para posteriormente tratarlos y analizarlos? Esta pregunta conduce directamente al siguiente apartado, el surgimiento del movimiento y bases de datos NoSQL, sistemas que intentan aportar soluciones ante los problemas tecnológicos derivados del fenómeno Big Data.

3.2.3. MARCO REGULADOR

Según el Informe de OBS (Online Business School) [19] se estima que en 2020, más de 30 mil millones de dispositivos estarán conectados a Internet. También se calcula que **en los últimos 10 años se ha creado más información que en toda la historia de la humanidad**. En particular, **el 90% de los datos se ha creado en los últimos dos años** y el ritmo de crecimiento del volumen de datos continúa creciendo.

Según [20] actualmente se están centralizando grandes volúmenes de datos en pocas compañías, algo que puede ser tema de preocupación. Es importante tener en cuenta que la naturaleza de los datos es muy diversa; en ellos puede haber datos de carácter personal que pueden ser utilizados para finalidades muy variadas, lo cual dificulta el cumplimiento de algunos de los principios de protección de datos. El principio de calidad de datos dispone, entre otras cosas, que los datos deberán ser adecuados, pertinentes y no excesivos, además de no utilizarse para fines incompatibles al de su recogida, deberán ser exactos y puestos al día, así como poder cancelarse cuando no sean necesarios pudiéndose conservar únicamente previa disociación. Todo esto es algo que en principio choca con el concepto de Big Data, que intenta recoger los máximos datos posibles de distintas fuentes (a veces no fiables) para generar información.

Todo esto puede ser peligroso, ya que por ejemplo, si no se hace una diferenciación clara entre los datos obtenidos directamente de un individuo y aquellos que han sido creados a partir de inferencias, el origen de los datos será incierto. Se podrían establecer conclusiones injustas. Big Data ha aportado grandes beneficios al sector socio-económico, pero no hay que olvidar el posible impacto que puede tener el manejo de esta gran cantidad de datos en la libertad de las personas. Por ejemplo, en las redes sociales donde el comportamiento de un usuario en el uso de las mismas puede repercutir en su libertad.

3.2.4. SURGIMIENTO DEL MOVIMIENTO Y BASES DE DATOS NOSQL

En los últimos años han salido a la luz numerosas bases de datos que reciben el nombre de NoSQL (Not only SQL - No solo SQL) que han llegado con el objetivo de hacer frente a problemas como la escalabilidad, el rendimiento, la accesibilidad o el mantenimiento de los datos. Problemas que las bases de datos SQL no podían tratar.

El término NoSQL, acuñado por primera vez en 1998 por Carlo Strozzi[10], surge con la llegada de la web 2.0[11] y las aplicaciones como Facebook, Twitter o Youtube, donde cualquier usuario podía subir contenido a la red, provocando así un crecimiento masivo de los datos. Es en ese momento cuando empiezan a surgir los problemas a la hora de gestionar toda esa información generada y almacenada en bases de datos relacionales.

En un primer momento se pensó en utilizar un mayor número de máquinas que alojaran toda esa información, pero el volumen, la variedad y la velocidad con la que se generaban esos datos requerían de nuevas tecnologías que solucionaran estos problemas. De esta forma apareció el movimiento NoSQL como solución a todos los problemas que las bases de datos relacionales no podían tratar. Las bases de datos NoSQL son sistemas de almacenamiento que no cumplen con el esquema entidad-relación, esquema característico en las bases de datos relacionales. Estas nuevas tecnologías hacen uso de nuevas y diferentes estructuras para el almacenamiento de los datos.

A continuación se van a exponer algunas de las diferencias entre las bases de datos relacionales y las bases de datos NoSQL y las principales ventajas de estas últimas [21]:

- **Las BBDD NoSQL no utilizan SQL como lenguaje de consultas**. La mayoría de las BBDD NoSQL evitan utilizar este lenguaje, aunque pueden usarlo. Algunas utilizan lenguajes como CQL[12], JSON[13] o GQL[14].

- **No utilizan estructuras fijas para el almacenamiento de datos**. No hacen uso de las tablas como estructura fija, almacenan la información sin diseñar la estructura por adelantado. NoSQL es libre de schemas[15].

[10] Carlo Strozzi usó este término para referirse a una base de datos desarrollada por él que no ofrecía una interfaz SQL. Más tarde fue Eric Evans quien en el año 2009 recuperó este término.
[11] El término web 2.0 está relacionado con la utilización de diversas aplicaciones en la red que permiten una interacción a nivel social, posibilitando la compartición de información. Este término apareció en el año 2004 y sigue estando vigente.
[12] Cassandra Query Language (CQL), lenguaje de programación para la BBDD NoSQL Apache Cassandra.
[13] JavaScript Object Notation (JSON). Alternativa a XML para el intercambio de datos. Este formato se utiliza, entre otras cosas, para el almacenamiento de documentos en bases de datos documentales.
[14] GQL (Google Cloud Platform), lenguaje parecido a SQL utilizado en la BBDD NoSQL Google BigTable.

- **Tienen arquitectura distribuida.** La información guardada en las BBDD NoSQL puede estar compartida en varias máquinas, mientras que en las BBDD relacionales se suele centralizar toda la información en una máquina.

- **No suelen soportar operaciones JOIN [16] ni garantizan completamente ACID[17].** Es conveniente evitar los JOIN al disponer de un volumen muy grande de datos. La sobrecarga puede ser costosa. A cambio ofrecen escalabilidad horizontal.

- **Escalabilidad horizontal.** Las BBDD NoSQL, al contrario que las BBDD relacionales están pensadas para escalar horizontalmente, de tal forma que se puedan añadir o quitar máquinas de una forma sencilla y con bajo coste.

- **Manejo de gran cantidad de datos.** Debido a su estructura distribuida, las BBDD NoSQL aportan un rendimiento muy superior frente a las BBDD relacionales.

Antes de ver los tipos de bases de datos NoSQL se van a introducir algunos conceptos y características de estos sistemas. En los dos siguientes apartados se hablará del Teorema CAP y de los conceptos ACID y BASE, características muy importantes que definen el comportamiento de las bases de datos que se van a tratar.

3.2.5. TEOREMA CAP

Según el teorema de Brewer [22] "es imposible para un sistema computacional distribuido ofrecer simultáneamente las siguientes tres garantías":

- **Consistencia**: un sistema es consistente si todos los nodos de un sistema ven los mismos datos en lectura después de una operación de escritura.

- **Disponibilidad (Availability)**: se considera que un sistema está disponible si para cada petición que se realiza se recibe una respuesta acerca de si tuvo éxito o no. Es decir, que se puedan seguir realizando operaciones incluso si otras partes del sistema están ocupadas en otros procesos.

---

[15] Schemas, o esquemas, término expresado en un lenguaje formal que describe la estructura de una base de datos, dentro de un sistema gestor de bases de datos.
[16] JOIN es una sentencia en SQL que permite la combinación de registros entre dos o varias tablas en una BBDD.
[17] ACID (Atomicity, Consistency, Isolation (aislamiento) and Durability). Conjunto de propiedades que garantizan que las transacciones en una base de datos son fiables.

- **Tolerancia a Particiones (Partition)**: el sistema es tolerante a particiones si continúa funcionando aunque haya partes que sean inaccesibles.

Este teorema se conoce como "Teorema CAP", haciendo referencia a las siglas de las tres garantías que debe cumplir un sistema. Además de enunciar que es imposible que se cumplan simultáneamente estas tres garantías en un sistema computacional distribuido, este teorema también dice que sí se pueden garantizar dos de estas tres propiedades.

Los SGBD cumplen dos de estas tres características, por lo que se puede realizar una clasificación de diferentes SGBD en función de esto.

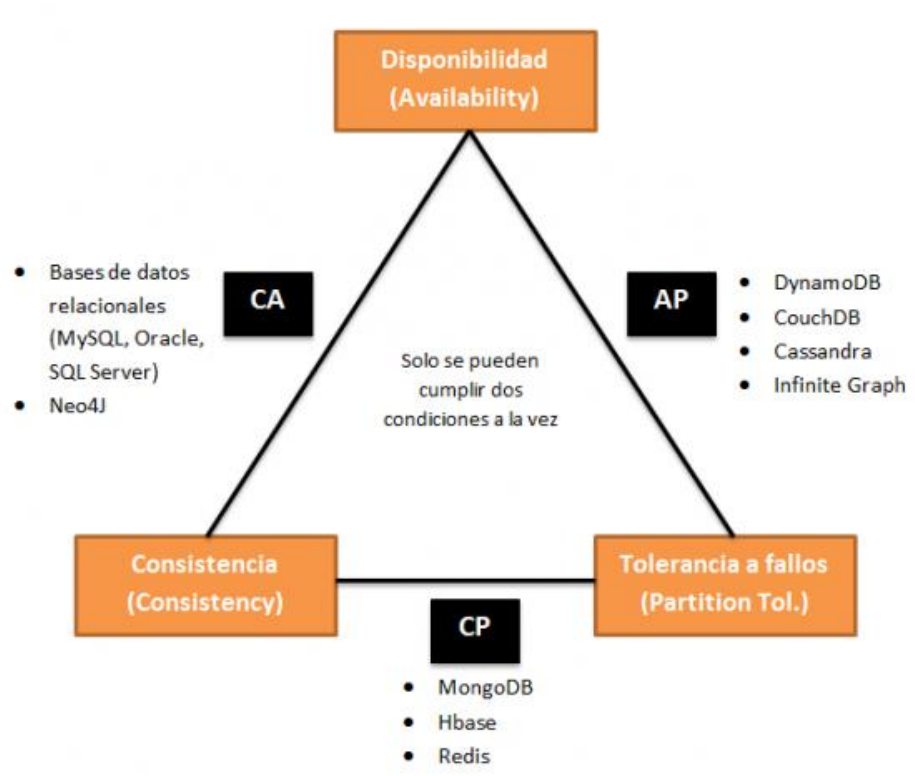

Figura 8. Teorema CAP (imagen original de [23])

Por lo tanto, si hay que elegir dos de las tres características que debe cumplir un sistema de datos compartidos habrá tres tipos de sistemas:

- **Sistemas consistentes y disponibles (CA)**: tendrán dificultades para funcionar si hay muchas particiones.
- **Sistemas consistentes y tolerantes a fallos (CP)**: tendrán carencias en cuestiones de disponibilidad.
- **Sistemas disponibles y tolerantes a fallos (AP)**: con problemas en temas de consistencia.

La elección de un tipo de base de datos u otra dependerá del problema a resolver. Es importante tener en cuenta estas características para conocer las ventajas e

inconvenientes de cada tipo de base de datos, de tal forma que se pueda resolver el problema que se aborde de la mejor manera posible.

3.2.6. ACID VS BASE

El mundo de las bases de datos relacionales está estrechamente familiarizado con las transacciones **ACID** [24] . Este tipo de transacciones se denominan así ya que garantizan la **A**tomicidad, la **C**onsistencia, el aislamiento (**I**solation) y la **D**urabilidad, explicados a continuación:

- **Atomicidad**: las transacciones se ejecutan por completo, lo que significa que las operaciones no pueden quedar a medias, ocurren al completo.
- **Consistencia**: los datos resultantes tras una transacción han de ser válidos de acuerdo a las reglas definidas.
- **Aislamiento (Isolation)**: las transacciones que se llevan a cabo son independientes entre sí, garantizando que una operación no afecte a otras operaciones.
- **Durabilidad**: una vez finalizada una operación esta persistirá en el tiempo.

Las bases de datos NoSQL siguen el modelo BASE:

El modelo BASE aporta un enfoque similar a ACID aunque "relaja" las condiciones perdiendo la consistencia y el aislamiento, y favorece la disponibilidad y el rendimiento.

El término BASE proviene de:

- **Basic Availability** (disponibilidad básica): el sistema funciona la mayor parte del tiempo incluso ante posibles fallos debido al almacenamiento distribuido y replicado.
- **Soft-State** (estado relajado): los sistemas no tienen por qué ser consistentes en todo momento.
- **Eventual Consistency** (consistencia eventual): la consistencia se produce de forma eventual.

Por lo tanto y para concluir, para que un SGBD relacional pueda ser considerado como tal debe cumplir el modelo ACID. Por otro lado, el modelo NoSQL se ajusta más al enfoque BASE, basado en un planteamiento más optimista y relajado y favoreciendo

que las aplicaciones sobre sistemas NoSQL funcionen la mayoría del tiempo y no teniendo que ser consistentes en todo momento.

3.2.7. TIPOS DE BASES DE DATOS NOSQL

A continuación se expondrán diferentes tipos de bases de datos NoSQL [21]. Para cada clasificación se aportarán distintos ejemplos de BBDD. Cabe añadir que mientras se ha investigado en la materia se ha encontrado que una BBDD está clasificada en dos tipos distintos de base de datos NoSQL. Esto se debe a que dependiendo del uso que se haga de la base de datos en cuestión, se puede encontrar que una funcionalidad concreta corresponda con un tipo de base de datos NoSQL y otras configuraciones de esa misma base de datos correspondan con otro tipo de base de datos NoSQL.

3.2.7.1. BASES DE DATOS CLAVE-VALOR

Este tipo de base de datos estructura y almacena la información a modo de diccionario. El diccionario, para este caso particular, está formado por tuplas clave-valor. Cada elemento o "palabra" del diccionario se representa como una clave única. A partir de esta clave un usuario puede añadir la "definición", llamada valor.

Debido a que cada elemento está identificado por una clave única, la recuperación de la información se realiza de forma muy rápida. Este modelo se caracteriza por ser altamente escalable y por tener muy buen rendimiento, debido a la simplicidad de su estructura. Estas bases de datos son muy utilizadas.

A continuación se expone una imagen que muestra la estructura de este tipo de BBDD.

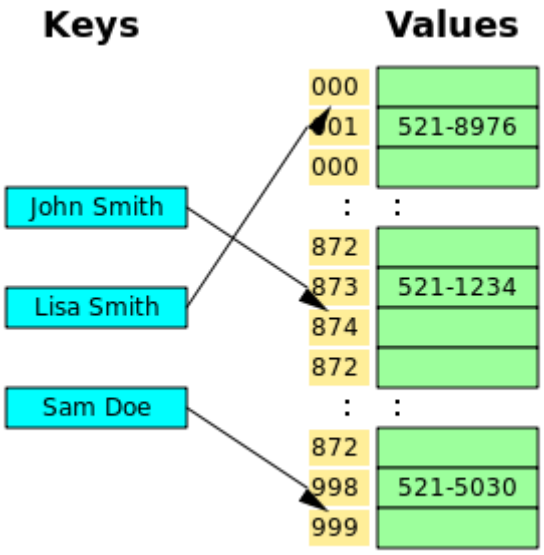

Figura 9. Base de datos clave-valor (imagen original de [21])

EJEMPLOS BBDD CLAVE-VALOR

Algunos ejemplos de Sistemas de Almacenamiento clave-valor comerciales son:

- **DynamoDB:** sistema de almacenamiento de datos NoSQL desarrollado por la famosa compañía de comercio electrónico **Amazon**. Amazon DynamoDB es una base de datos NoSQL gestionada totalmente en la nube y compatible con modelos de almacenamiento de datos de clave-valor y de documentos. Ofrece un rendimiento rápido y constante a todas las aplicaciones, es altamente escalable, flexible, y tiene un acceso de control riguroso, diseñado para gestionar el control de los usuarios dentro de su organización. Además este sistema es totalmente gestionado, el usuario se encarga de crear la BBDD y establecer el rendimiento y la herramienta se encarga de todo lo demás. Esto hace que el usuario no tenga que preocuparse por la gestión de la BBDD en cuanto a hardware o software, ni a su configuración e instalación.

- **Redis:** Base de datos NoSQL que reside en memoria RAM, escrita en ANSI C, apoyada por VMWare. Su funcionamiento se puede comparar a un gran array en memoria para almacenar datos. Estos datos no están limitados a ser de tipo String, se admiten otro tipo de datos, como listas de Strings, sets de Strings (elementos desordenados no repetidos) y hashes o listas. Esta base de datos no permite realizar consultas, solo inserta y permite obtener datos. Soporta varios lenguajes de programación, como Action Script, C, C++, C#, Clojure, Java, Perl y PHP, entre otros.

Redis es utilizada por grandes compañías como **Flickr**, **Instagram**, **Pinterest** o **Github**.

Otras bases de datos clave-valor:

- Riak [25]
- Aerospike [26]
- LevelDB [27]
- Project Voldemort [28]
- Oracle NoSQL Database [29]

3.2.7.2. BASES DE DATOS DOCUMENTALES

Las bases de datos documentales almacenan toda la información en formato de documento. Se entiende como documento un conjunto de datos organizados en una estructura. En el caso por ejemplo de MongoDB, donde, como se verá en el siguiente apartado, los datos se organizan en la estructura BSON (Binary JSON). JSON proviene de JavaScript Object Notation, una alternativa a XML utilizada para el intercambio de datos.

La base de datos gestiona el almacenamiento, recuperación y tratamiento de estos documentos. Para ello utilizan estructuras como JSON, indicada anteriormente, XML o YAML[18], aunque también pueden hacer uso de los formatos Word o PDF. Para cada registro se utiliza una clave única, de tal forma que se puedan realizar búsquedas por clave-valor. Debido a que la unidad de almacenamiento de este tipo de base de datos es el documento, su modelo de datos es más complejo y completo. Se pueden hacer búsquedas por clave-valor como se ha comentado o hacer búsquedas más avanzadas sobre el contenido del documento. Son bases de datos muy versátiles, utilizadas en muchos proyectos.

A continuación se expone una imagen que muestra la estructura de este tipo de base de datos. A la izquierda se puede ver cómo una estructura en JSON almacena el resumen de los datos por clave-valor, resultado de dos documentos. Uno, el documento de información de usuarios y dos, el documento de información de direcciones. El fichero JSON recupera los datos de ambos documentos y los almacena en un solo fichero como valores clave-valor.

[18] YAML es un formato de representación de datos legible e inspirado en lenguajes como XML, C, Python o Perl.

Figura 10. Base de datos documental (imagen original de [21])

EJEMPLOS BBDD DOCUMENTALES

Algunas bases de datos documentales son:

- **MongoDB**: su nombre proviene de *humongous*, traducido como "enorme". Sistema desarrollado bajo el concepto de "código abierto". Esta base de datos guarda colecciones de documentos, el equivalente a almacenar tablas en una base de datos relacional. Con MongoDB no es necesario definir ningún esquema, se puede directamente insertar los datos y acceder a ellos. Esta es una de las características y diferencias más importantes con respecto a las bases de datos relacionales. Los documentos que son almacenados en una misma colección pueden tener esquemas diferentes.

  MongoDB está escrito en C++, aunque las consultas se hacen utilizando objetos JSON, ya que los documentos están almacenados en estructura BSON. Se pueden utilizar otros lenguajes de programación para esta base de datos, ya que existen drivers de forma oficial para lenguajes como C#, Java, Node.js, PHP, Python, Ruby, C y C++ entre otros.

  Debido a su estructura y organización, esta base de datos proporciona una gran escalabilidad, rendimiento y disponibilidad.

  Compañías como **MetLife**, **eBay**, **SourceForge**, **Telefónica** o **CERN** hacen uso de MongoDB.

- **IBM Notes**: sistema software cliente/servidor de colaboración y correo electrónico. La parte del servidor recibe el nombre de Lotus Domino mientras que la parte cliente se denomina **Lotus Notes**. Es un sistema de comunicación que permite el envío de correo electrónico y el manejo de calendarios y agendas.

Y es también un sistema de colaboración ya que permite compartir bases de datos documentales.

Lotus Notes fue desarrollado durante las décadas de los años 1970 y 1980. Este sistema se presentó como una alternativa a la utilización de las bases de datos relacionales, por lo que aunque el movimiento NoSQL se considera de un tiempo más reciente, ya hubo sistemas como IBM Notes que aportaban otro punto de vista al tratamiento de los datos en las bases de datos. Es por ello que se puede considerar precursor de las bases de datos NoSQL.

- Apache CouchDB, conocida como **CouchDB**, es una base de datos documental cuyo desarrollador, Damien Katz, se inspiró en el funcionamiento de Lotus Notes a la hora de crearla.

  Actualmente este sistema es muy utilizado en grandes empresas para la gestión de grandes almacenes de datos, como por ejemplo puede ser el inventario de equipos de una empresa o los datos de personal que la componen.

Otras bases de datos documentales:

- SimpleDB [30]
- Terrastore [31]

3.2.7.3. BASES DE DATOS COLUMNARES

Este tipo de bases de datos estructura la información en columnas, como un modelo tabular, al contrario que el modelo relacional que lo hace por filas. Cada fila puede tener una configuración diferente de columnas. Todos los valores o casos que puede tomar un elemento se almacenan de forma que se pueda acceder como una unidad. Esta característica las hace bastante eficaces a la hora de hacer consultas, ya que el acceso por columnas es rápido.

El modelo de almacenamiento por columnas utiliza el esquema de clave-valor para almacenar la información, pero utiliza otro esquema (columnar) para ordenar y almacenar los datos.

A continuación se muestra una imagen del esquema de tabla en HBase.

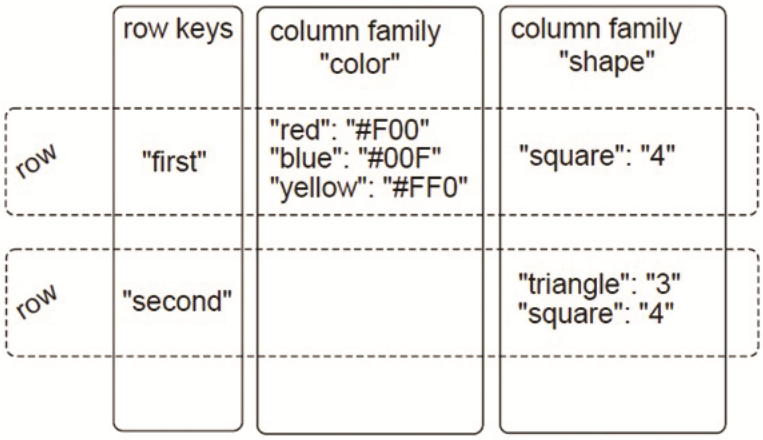

Figura 11. Base de datos columnar (imagen original de [32])

Como se puede observar, el almacenamiento se hace por clave-valor, y hay diferente número de campos (columnas) para un elemento (fila). El acceso a datos es rápido por columnas y lento por filas. Este tipo de base de datos se escala de manera horizontal.

EJEMPLOS BBDD COLUMNARES

- **Cassandra**: Apache Cassandra, base de datos columnar clasificada también como clave-valor por su modelo de almacenamiento. Como se ha comentado en el punto anterior los datos se almacenan bajo el modelo de clave-valor, pero la estructura de la base de datos es columnar. Desarrollada inicialmente por **Facebook**, está implementada en Java.

  Esta base de datos está diseñada para ser desplegada en un servidor distribuido. Aporta una gran escalabilidad, proporciona respuestas rápidas ante el aumento masivo de datos, es muy rápida en escrituras, punto importante para gestionar volúmenes de datos crecientes, es flexible, se puede cambiar su estructura de datos ante la demanda de funcionalidad, es descentralizada.

  Estas características la hacen una de las bases de datos más conocidas del mundo NoSQL y es utilizada para análisis de datos en tiempo real, banca, sectores financieros y otros sectores donde se exige un gran rendimiento y velocidad de escrituras masivas en volúmenes de datos crecientes.

  Apache Cassandra es utilizada por compañías como **Twitter**, **Netflix** o **Digg**.

Otras bases de datos columnares:

- Google BigTable [33]
- HBase [34]
- Hypertable [35]

Debido a que el almacenamiento de los valores se hace por clave-valor, estas bases de datos se pueden encontrar clasificadas también como bases de datos clave-valor.

3.2.7.4. BASES DE DATOS ORIENTADAS A GRAFO

Este tipo de bases de datos representa la información como nodos de un grafo y las relaciones mediante aristas, pudiendo aplicar la Teoría de Grafos para recorrer la base de datos. La navegación entre las relaciones es más eficiente que en el modelo relacional. El modelo de datos es flexible, lo que significa que en ocasiones se permite no declarar nodos o aristas.

El modelo orientado a grafo facilita al desarrollador una gran flexibilidad a la hora de manipular los datos, ya que le permite conectarlos y realizar futuras consultas o actualizaciones de manera sencilla.

La siguiente imagen muestra un ejemplo de la estructura de las bases de datos orientadas a grafo.

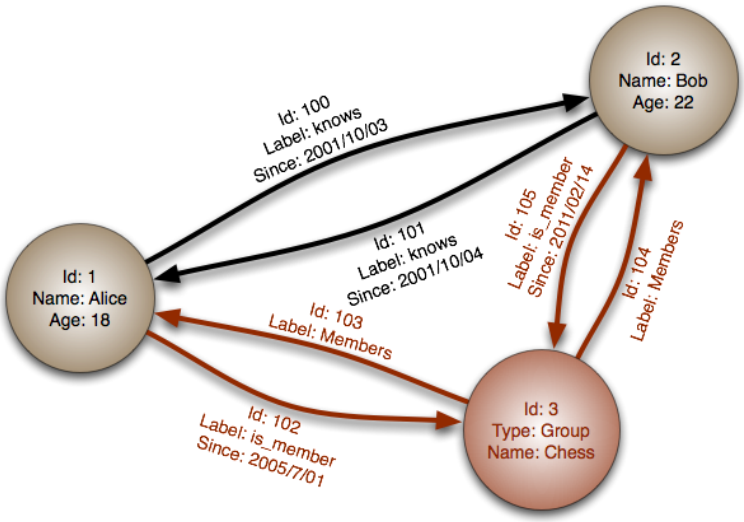

Figura 12. Bases de datos orientadas a grafo (imagen original de [36])

EJEMPLOS BBDD ORIENTADAS A GRAFO

Algunas bases de datos orientadas a grafo son:

- **Neo4j**: implementado en Java, aunque se integra perfectamente con otros lenguajes como PHP, Ruby, .Net, Python o JavaScript. Esta base de datos es especialmente útil para el modelado de datos en redes sociales y sistemas de recomendación, al poder conectar gran variedad de nodos mediante el uso de relaciones.

  Para trabajar con los grafos se hace uso del lenguaje Cypher[19], que permite crear la estructura de un grafo, sus nodos y relaciones y la realización de consultas. Este lenguaje está basado en SQL.

  Neo4j es utilizado por **Infojobs**, entre otros.

Otras bases de datos orientadas a grafo:

- InfoGrid [37]
- InfiniteGraph [38]
- Sparksee [39]

3.2.7.5. BASES DE DATOS ORIENTADAS A OBJETOS

Los sistemas de bases de datos orientados a objetos estructuran la información al igual que lo hacen los lenguajes de programación orientada a objetos (POO), como por ejemplo Java, C# o Visual Basic .NET. La información se representa como objetos; sobre ellos se pueden realizar operaciones para manipularlos.

Si un sistema reúne una base de datos de este tipo y un lenguaje de programación orientado a objetos se puede hablar de un sistema gestor de bases de datos orientado a objetos.

Estos tipos de bases de datos son muy útiles y potentes cuando se utilizan con lenguajes POO, ya que en estos sistemas la capa de persistencia, encargada de conectar la aplicación con la base de datos, está desarrollada específicamente para trabajar con aplicaciones basadas en programación orientada a objetos. La idea fundamental es que el usuario no tenga que batallar con construcciones orientadas al computador tales como

[19] Cypher, lenguaje propio de Neo4j.

registros y campos, sino que pueda manejar objetos que se asemejen y representen sus datos equivalentes en el mundo real.

EJEMPLOS BBDD ORIENTADAS A OBJETOS

- **ZODB**: Zope Object Database. Base de datos orientada a objetos nativa creada a finales de los 90 que almacena de forma persistente objetos. Utiliza el lenguaje de programación Python. ZODB ofrece persistencia de forma transparente para objetos Python, soporte eficiente para almacenar datos de gran tamaño en objetos binarios grandes (BLOBs[20]) y una arquitectura escalable, entre otras ventajas.

Otras bases de datos orientadas a objetos:

- GemStone/S [40]
- Db4o [41]

3.2.8. RESUMEN DE LAS BBDD NOSQL

Vistos todos los tipos de bases de datos NoSQL con ejemplos incluidos, a continuación se muestra una tabla resumen que recoge las principales características de cada una de ellas.

[20] Binary Large Objects (BLOB), son objetos que permiten un almacenamiento binario de la información. Es un tipo de datos utilizado en las bases de datos para almacenar datos de gran tamaño y de todo tipo. No todos los SGBD soportan los BLOB.

Características	Bases de datos NoSQL				
	Clave-Valor	**Documental**	**Columnar**	**Orientada a Grafo**	**Orientado a objetos**
Sistemas de Almacenamiento comerciales	DynamoDB Redis LevelDB Project Voldemort Aerospike Riak	MongoDB IBM Notes CouchDB SimpleDB Terrastore	Cassandra Google BigTable HBase Hypertable	Neo4j Infogrid InfiniteGraph Sparksee	ZODB GemStone/S Db4o
Modo de almacenamiento	Diccionario	Documento y clave-valor	Columnar y clave-valor	Grafo	Objetos
Escalabilidad	Altamente escalable	Altamente escalable	Altamente escalable	Altamente escalable	Altamente escalable
Consistencia	Consistencia eventual	Consistencia eventual	Consistencia eventual	Consistencia eventual	Consistente
Disponibilidad	Alta disponibilidad	Alta disponibilidad	Alta disponibilidad	Alta disponibilidad	Alta disponibilidad
Recomendado para	Aplicaciones que crecen progresivamente Entornos distribuidos	Consultas avanzadas Datos semi estructurados Variedad de datos	Procesamiento y escaneo de grandes cantidades de datos	Datos conectados, modelos con muchas relaciones	Almacenar grandes volúmenes de objetos BLOBs
Casos de uso	Análisis en tiempo real Rankings Clasificaciones	Comercio electrónico Aplicaciones móviles Almacenamiento de comentarios	Escaneo masivo de datos Banca, industria financiera	Redes sociales Medicina Estadística	Sistemas embebidos Aplicaciones móviles Prototipos de sistemas

Tabla 5. Comparativa-resumen bases de datos NoSQL

3.2.9. DISCUSIÓN FINAL

Una vez se han repasado todos los sistemas de almacenamiento, tanto los tradicionales como los NoSQL, en el siguiente capítulo se llevará a cabo una comparativa entre un sistema relacional y uno NoSQL. Se ha decidido escoger de entre los sistemas de almacenamiento tradicionales la base de datos relacional por ser la más conocida y extendida. Esta se comparará con un sistema NoSQL. Para realizar dicha comparativa se elegirá e implementará un caso práctico donde se estudiarán las ventajas y desventajas de escoger uno u otro sistema.

La motivación de esta comparativa es, entre otras causas, que uno de los principales problemas de las empresas hoy en día es saber cuándo utilizar un tipo de base de datos u

otra. Como ya se contará más adelante, dependiendo de las necesidades habrá bases de datos que se ajusten mejor al problema dado que otras. Es importante que las empresas, organizaciones y usuarios comunes sepan el problema al que se enfrentan para así almacenar sus datos de la forma más propicia. Si esto no se hace bien, el resultado será que la base de datos elegida en cuestión no será manejable, será lenta y no responderá adecuadamente al tratamiento y gestión de los datos. En este TFG se busca aclarar cuáles son las principales diferencias entre dos tipos de bases de datos, explicadas en los siguientes apartados, y demostrar mediante un ejemplo cuáles son los problemas, ventajas e inconvenientes que se pueden encontrar en cada una de ellas.

CAPÍTULO 4. COMPARATIVA

4.1. INTRODUCCIÓN Y MOTIVACIÓN

A lo largo de este Trabajo Fin de Grado se realizará una comparativa entre las bases de datos relacionales y las bases de datos NoSQL. Para ello se pondrá un ejemplo que deje ver en qué situaciones resulta más óptimo la utilización de una base de datos NoSQL frente una relacional y viceversa.

Como ya se ha comentado, las bases de datos NoSQL surgen para resolver problemas que las BBDD relacionales no podían tratar. Problemas que derivan de la creciente cantidad de información que se genera constantemente debido al cambio producido hace unos años en la red con la llegada de la web 2.0 y de aplicaciones y tecnologías que creaban, manipulaban y transformaban grandes volúmenes de información. Este fenómeno, conocido como Big Data, es de interés general debido a la importancia que tiene a día de hoy el manejo de datos por parte de usuarios, clientes y organizaciones.

Como ya se ha visto, las bases de datos relacionales no son eficientes cuando se utilizan como modo de almacenamiento para Big Data, ya que la cantidad de datos y la variedad de ellos no permite que sus sistemas de almacenamiento los procesen adecuadamente.

Pese a este problema, las BBDD relacionales siguen siendo hoy en día una de las herramientas más utilizadas, ya que una gran cantidad de organizaciones, empresas y negocios de todo tipo almacenan sus datos de forma relacional. Al igual que el resto de sistemas de almacenamiento, dependiendo del caso de uso para el que se utilice, tendrá sus pros y contras. En este TFG se pretende mostrar un ejemplo donde se realice una comparación entre una BBDD relacional y una BBDD NoSQL, para observar en qué casos es más eficiente y más idónea el uso de una u otra.

En la siguiente figura se muestra un ranking de las bases de datos más populares actualmente. Como se puede observar, las SGBDR siguen siendo los sistemas más populares de almacenamiento de información:

Rank Aug 2015	Rank Jul 2015	Rank Aug 2014	DBMS	Database Model	Score Aug 2015	Score Jul 2015	Score Aug 2014
1.	1.	1.	Oracle	Relational DBMS	1453.02	-3.70	-17.83
2.	2.	2.	MySQL	Relational DBMS	1292.03	+8.69	+10.81
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1108.66	+5.60	-133.84
4.	4.	↑ 5.	MongoDB	Document store	294.65	+7.26	+57.30
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	281.86	+9.04	+32.01
6.	6.	6.	DB2	Relational DBMS	201.23	+3.12	-5.19
7.	7.	7.	Microsoft Access	Relational DBMS	144.20	-0.10	+4.58
8.	8.	↑ 10.	Cassandra	Wide column store	113.99	+1.28	+32.09
9.	9.	↓ 8.	SQLite	Relational DBMS	105.82	-0.05	+16.95
10.	10.	↑ 11.	Redis	Key-value store	98.81	+3.73	+28.01
11.	11.	↓ 9.	SAP Adaptive Server	Relational DBMS	85.11	-2.10	-1.06
12.	12.	12.	Solr	Search engine	81.90	+2.61	+12.87
13.	13.	13.	Teradata	Relational DBMS	73.59	+1.28	+8.21
14.	14.	↑ 16.	Elasticsearch	Search engine	69.64	-0.52	+30.84
15.	15.	15.	HBase	Wide column store	59.95	-0.97	+18.03
16.	↑ 17.	↑ 19.	Hive	Relational DBMS	53.87	+5.67	+23.06
17.	↓ 16.	↓ 14.	FileMaker	Relational DBMS	51.87	+1.05	-0.20
18.	18.	↑ 20.	Splunk	Search engine	42.20	+0.55	+14.59
19.	19.	↑ 23.	SAP HANA	Relational DBMS	38.25	+1.63	+15.92
20.	20.	↓ 17.	Informix	Relational DBMS	36.80	+0.91	+3.64
21.	21.	↓ 18.	Memcached	Key-value store	33.38	+0.25	+2.39
22.	22.	22.	Neo4j	Graph DBMS	33.16	+1.81	+10.25
23.	23.	↓ 21.	CouchDB	Document store	27.28	+0.35	+3.15
24.	24.	24.	Couchbase	Document store	26.16	-0.11	+8.62
25.	↑ 26.	↑ 27.	MariaDB	Relational DBMS	24.12	+0.85	+8.76
26.	↓ 25.	↓ 25.	Firebird	Relational DBMS	22.74	-0.68	+5.95
27.	27.	↓ 26.	Netezza	Relational DBMS	20.90	+1.38	+4.33
28.	↑ 29.	↑ 32.	Amazon DynamoDB	Multi-model	18.45	+1.93	+8.34
29.	↓ 28.	↓ 28.	Microsoft Azure SQL Database	Relational DBMS	17.80	+0.55	+3.34
30.	30.	↓ 29.	Vertica	Relational DBMS	16.26	+0.42	+3.35
31.	31.	↓ 30.	Riak	Key-value store	14.81	+1.16	+3.14

Figura 13. Ranking de las bases de datos más populares (imagen original de [42])

Esta clasificación se ha elaborado a partir de los siguientes parámetros [43]:

- Número de menciones del sistema en la web, en buscadores web.
- Interés general del sistema, medido en la frecuencia de búsquedas en Google Trends[21].
- Número de ofertas de trabajo en las que el sistema es requerido o mencionado.
- Perfiles profesionales, por ejemplo en la red social LinkedIn donde el sistema es mencionado.

[21] Herramienta de Google que muestra los términos más buscados en la red

- Relevancia en redes sociales como Twitter.

A partir de este ranking se puede concluir que las bases de datos relacionales siguen siendo las más utilizadas hoy en día, siendo necesario utilizar estas bases de datos para datos transaccionales. Por otro lado, podemos observar en la tabla que la base de datos NoSQL más utilizada es el almacén documental MongoDB, ocupando la cuarta posición y seguido por la BBDD Columnar Cassandra ocupando la octava posición. Esto hace pensar que la idea de apartar completamente las bases relacionales del mercado por no solucionar ciertos problemas provocados por Big Data no es correcta; dependiendo del problema a tratar y de si una organización o empresa o usuario final dispone de datos estructurados o no le será más conveniente la utilización de una u otra base de datos.

A modo de resumen, a continuación se recogen algunas de las características y diferencias existentes entre las BBDD relacionales y las BBDD NoSQL que se han comentado hasta ahora:

	BBDD Relacionales	BBDD NoSQL				
		Clave-Valor	Documental	Columnar	Orientada Grafo	Orientada Objetos
Sistemas de Almacenamiento comerciales	Oracle MySQL DB2 PostgreSQL SQLite Microsoft SQL Server	DynamoDB	MongoDB	Cassandra	Neo4j	ZODB
Lenguaje	SQL	Java .NET, Ruby, Python, Node.js	JSON, BSON	CQL	Cypher	Python
Modo de almacenamiento	Relacional Almacenamiento por filas	Diccionario	Documento y clave-valor	Columnas y clave-valor	Grafo	Objetos
Patrones	Definidos, estructuras fijas, utilización de schemas	Sin definir, no estructuras fijas				
Arquitectura	Normalmente centralizada	Distribuida o centralizada				
ACID	Garantizan ACID	No garantizan ACID				
Escalabilidad	Difícilmente escalable, datos en un punto concreto	Altamente escalable (horizontal)				

	BBDD Relacionales	BBDD NoSQL
Disponibilidad	En un punto, cuellos de botella	Múltiples réplicas, alta disponibilidad
Consistencia	Altamente consistente	Consistencia eventual
Costes	Requiere de licencias Sensibles a cambios de modelo	"Open source" Menos sensibles a cambios de modelo
Operaciones JOIN	Permite JOIN	No permiten JOIN o de forma muy limitada
Tipo de datos	Estructurados	Estructurados, semi estructurados
Recomendado para	Operaciones y transacciones que impliquen muchas relaciones Datos estructurados	Operaciones sencillas en tiempo real y transacciones en la red
Casos de uso	Informes históricos Informes médicos Bases de datos científicas Gestión empresarial Recursos humanos	Escaneo masivo de datos Aplicaciones móviles Juegos online Análisis de datos y contenidos Rankings Comercio electrónico Social Media Estadísticas Reservas online Servicios de publicidad

Tabla 6. Comparativa BBDD relacionales y BBDD NoSQL

Por tanto y para concluir, no hay que enfocar el tema de la comparativa como un SQL vs NoSQL. Como se ha comentado son dos herramientas que tienen ámbitos diferentes y por ello la utilidad de una frente a otra dependerá del problema que se trate.

Para poner en práctica estas diferencias, en los siguientes apartados se mostrará la elección de un dominio de datos y su aplicación e implementación en las dos plataformas: NoSQL y SQL.

4.2. ELECCIÓN DEL DOMINIO

Para llevar a cabo la comparativa y estudio entre un sistema típicamente relacional (SQL) y uno NoSQL se ha elegido como caso práctico el dominio de artículos médicos, utilizando como fuente de datos la herramienta **PubMed**. A continuación se explica qué es y se indican algunas de sus características:

- **Qué es**

PubMed es un recurso gratuito, desarrollado y mantenido por el Centro Nacional para la Información Biotecnológica[22], en la Biblioteca Nacional de Medicina de los Estados Unidos.[23] Esta herramienta fue creada en 1996. Se puede acceder a PubMed a través del siguiente enlace: http://www.ncbi.nlm.nih.gov/pubmed

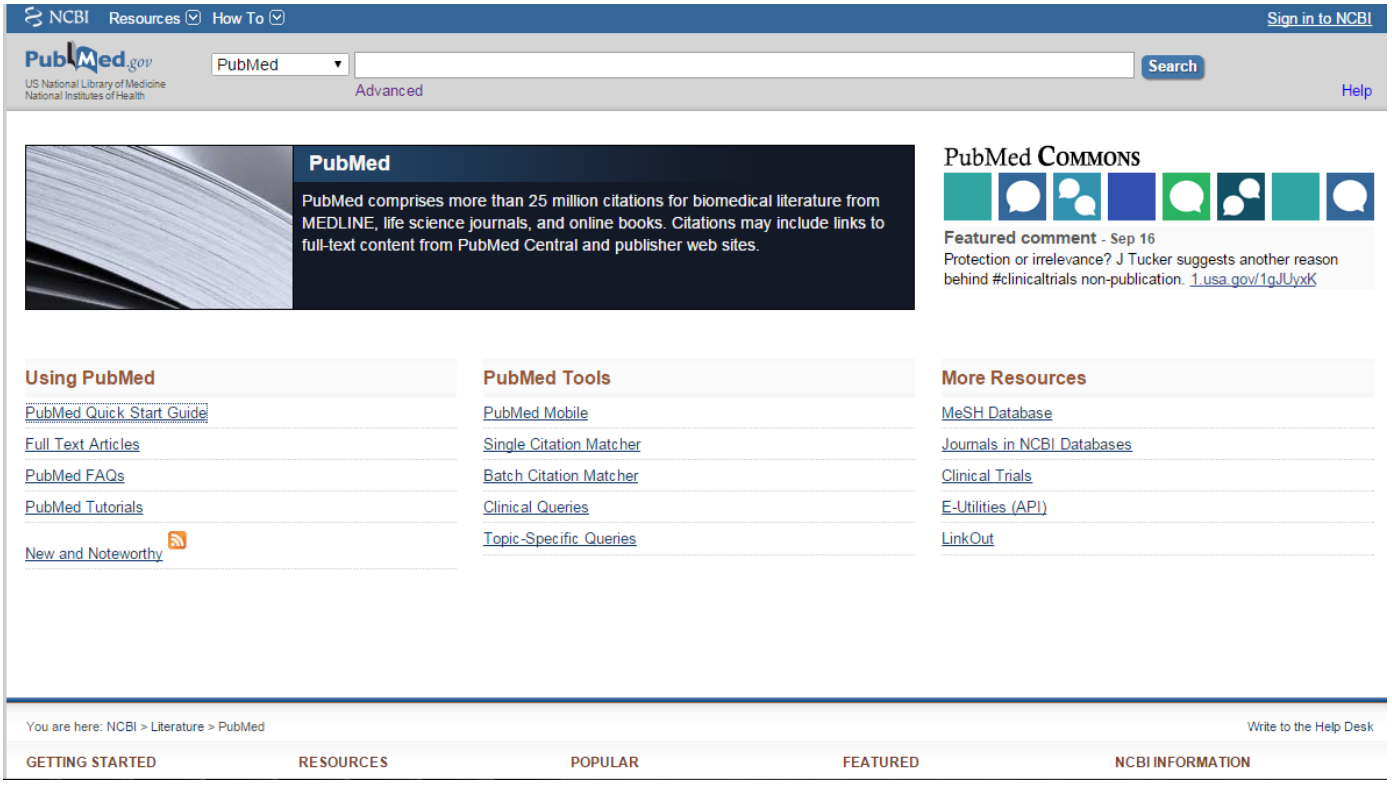

Figura 14. Página principal de PubMed (imagen original de [50])

PubMed actúa como motor de búsqueda de la base de datos **MEDLINE**, una amplísima base de datos bibliográfica médica que incluye citaciones y resúmenes de artículos de investigación biomédica. Esta base de datos cuenta con más de 25 millones de referencias en la que se pueden buscar artículos publicados en más de 5600 publicaciones biomédicas. Puede ser usada gratuitamente e incluye enlaces al texto

[22] National Center for Biotechnology Information (**NCBI**)
[23] U.S. National Library of Medicine (**NLM**)

completo de algunos artículos de revistas médicas por medio de las páginas de Internet del editor. El acceso editorial a los artículos varía según el editor. Algunos proveen acceso gratis, mientras que otros cobran por el artículo o exigen suscripción [44].

- **Iniciar búsquedas en PubMed**

Como motor de búsqueda, en PubMed hay varias opciones a la hora de buscar un artículo. Desde la pantalla de inicio se puede buscar por términos, frases (escritas entre comillas) o autores, entre otros campos.

Se pueden utilizar **operadores booleanos** como AND, OR o NOT para combinar o modificar los criterios de búsqueda.

- El operador AND, que funciona como "intersección"; recupera artículos o citas que contengan todos los términos, combinándolos.
- El operador OR recuperará en su búsqueda los artículos que contengan los dos términos o al menos uno de ellos.
- Por último el operador booleano NOT funciona como excluyente. Localiza artículos que contengan el primer término de búsqueda pero que no contengan el resto de los términos.

Una característica a la hora de realizar búsquedas en PubMed es que también se puede realizar **truncado** de términos. Esto permite recuperar todos aquellos términos que poseen la misma raíz. Colocando un asterisco (*) al final de un término de búsqueda, PubMed busca en todas aquellas palabras que tengan la misma cadena de letras como raíz. Por ejemplo si se busca por neurol* se recuperarán en la búsqueda términos como *neurology* o *neurologic* entre otros.

Otra opción a la hora de realizar búsquedas es utilizar **calificadores de campos o etiquetas**. Cada campo de un registro bibliográfico se identifica mediante una etiqueta. Esta etiqueta puede ser añadida a continuación de cada término entre corchetes. Por ejemplo se puede añadir al término de búsqueda "Darwin" la etiqueta [PS] para buscar aquellos artículos o citas sobre un individuo como tema. A continuación se indican los calificadores de campos que se pueden utilizar.

Nombre del campo	Abreviatura	Descripción
Affiliation - Dirección	[AD, AFFL]	Dirección y filiación institucional del primer autor y número de ayuda económica.
All Fields - Todos los campos	[ALL]	Incluye todos los campos de búsqueda de PubMed.
Author Name - Autores	[AU, AUTH]	Desde el año 2000 Medline no pone límite al número de autores. Hasta 1999 incluía los 25 primeros autores seguido de la abreviatura et al. El formato para buscar un autor es Apellido, seguido de espacio y la/s primeras iniciales sin puntos (ej., Fauci AS). Se pueden omitir las iniciales al buscar.
Corrected and republished from	[CRF]	Versión corregida y publicada nuevamente.
Corrected and republished in	[CRI]	Artículo original que fue corregido y vuelto a publicar
E.C./RN	[RN, ECNO]	Número asignado por la Comisión de Enzimas (E.C.). Número para designar una enzima en particular y el listado RN del Chemical Abstracts Service (CAS) Registry Numbers.
Entrez Date -Fecha de ingreso	[EDAT]	Contiene la fecha que la cita fue agregada a PubMed, en el formato aaaa/mm/dd [edat], ej., 1998/01/10 [edat].
Erratum for	[EFR]	Errata: cita el artículo original que necesita corrección.
Full Author Name	[FAU]	Apellido y nombre completo del autor.
Full Investigator	[FIR]	Apellido completo e inicial del nombre del investigador.
Full Personal Name as Subject	[FPS]	Nombre personal como materia.
General Note	[GN]	Información suplementaria o descriptiva relacionada con el documento.
Grant Number	[GR]	Números de identificación de las agencias de financiación norteamericanas US PHS o Wellcome Trust.
Issue - Número	[IP, ISSUE]	Número del volumen de la revista en la cual el artículo se encuentra publicado.
Investigator	[IR]	NASA-investigador principal.
Investigator Affiliation	[IRAD]	Afiliación NASA del investigador principal.
Journal Name - Nombre de la publicación	[TA, JOUR]	La abreviatura del título de la revista, el título completo o el número ISSN (ej., J Biol Chem, Journal of Biological Chemistry, 0021-9258).
Full Journal Title	[JT]	Nombre completo de la revista procedente de la NLM
Language - Idioma	[LA, LANG]	El idioma en el cual el artículo fue publicado.
MeSH Major Topic - MeSH tópico principal	[MAJR]	Término MeSH que cubre los aspectos más relevantes de un artículo
MeSH Terms - Términos MeSH	[MH,MESH]	El vocabulario controlado de la NLM's (Medical Subject Headings) sobre términos biomédicos que se usan para describir cada artículo de una revista científica en MEDLINE.
Other Abstract	[OAB]	Resumen suministrado por un organismo colaborador de la NLM.
Other Copyright Information	[OCI]	Otra información del Copyright.
Other ID	[OID]	Otros datos de identificación del ID.

Nombre del campo	Abreviatura	Descripción
Original Report In	[ORI]	Informe original asociado al sumario para el paciente.
Other Term	[OT]	Términos no-MeSH adjudicados por otra organización (OTO).
Other Term Owner	[OTO]	Organización que proporcionó otros datos del término.
Owner	[OWN]	Siglas de la organización que proveyeron datos de la citación.
Page - Página	[PG, PAGE]	Página inicial y final del artículo.
Personal Name - Nombre personal	[PS]	Para buscar citas sobre un individuo como tema. Utilice las reglas como para buscar un autor.
Place of Publication	[PL]	País de publicación de la revista.
Publication Date - Fecha de publicación	[DP, PDAT]	La fecha en que el artículo fue publicado en el formato aaaa/mm/dd (ej. 1984/10/06). Un año con solo un mes (ej., 1984/03) mostrará todo lo de ese mes. Las fechas de publicación no están estandarizadas entre las publicaciones.
Publication Type - Tipo de publicación	[PT, PTYP]	Describe el tipo de material que el artículo representa (ej., Review, Clinical Trials, Retracted Publications, Letters).
Publishing Model	[PUBM]	Soporte del artículo de la publicación: impreso (print) o electrónico (electronic).
Space Flight Mission	[SFM]	NASA- datos de la misión espacial.
Subheading - Subencabezado	[SH]	Subencabezamientos utilizados para calificar de forma más precisa la búsqueda con términos MeSH.
Subset - Subgrupos	[SB]	Permite elegir sobre que subbase de la base de datos Medline queremos buscar: Aids, Bioehics, Cancer, etc.
Substance Name - Nombre de substancia	[NM, SUBS]	El nombre de una sustancia química tratada en el artículo (MEDLINE Name of Substance field).
Summary For Patients In	[SPIN]	Sumario para pacientes.
Status Tag	[STAT]	Estatus de la etiqueta en la NLM
Text Words - Palabras del texto	[TW, WORD]	Todas las palabras de los campos del título, resumen, términos MeSH, subencabezamientos, nombres de sustancias químicas, nombre de persona como tema y campos de identificación secundaria.
Title Words - Palabras del título	[TI, TITL]	Palabras que se encuentran en el título de un artículo
Volume - Volumen	[VI, VOL]	El número del volumen de la publicación donde el artículo es publicado.
PubMed Identifier (PMID) & MEDLINE Unique Identifier (UI)		Número de identificación unívoco de cada registro PubMed (PMID) o Medline (UI).

Tabla 7. Calificadores de campos o etiquetas en PubMed

Esta es una introducción sobre algunos tipos de búsqueda que se pueden realizar. Si se desea obtener más información sobre cómo realizar éstas u otras búsquedas se puede consultar el manual de la página web. [45]

- **Tipo de datos en PubMed**

A la hora de descargar una colección de datos resultante de un criterio de búsqueda, PubMed ofrece la opción de descargar varios tipos de ficheros. En la siguiente figura se muestran los tipos, que son:

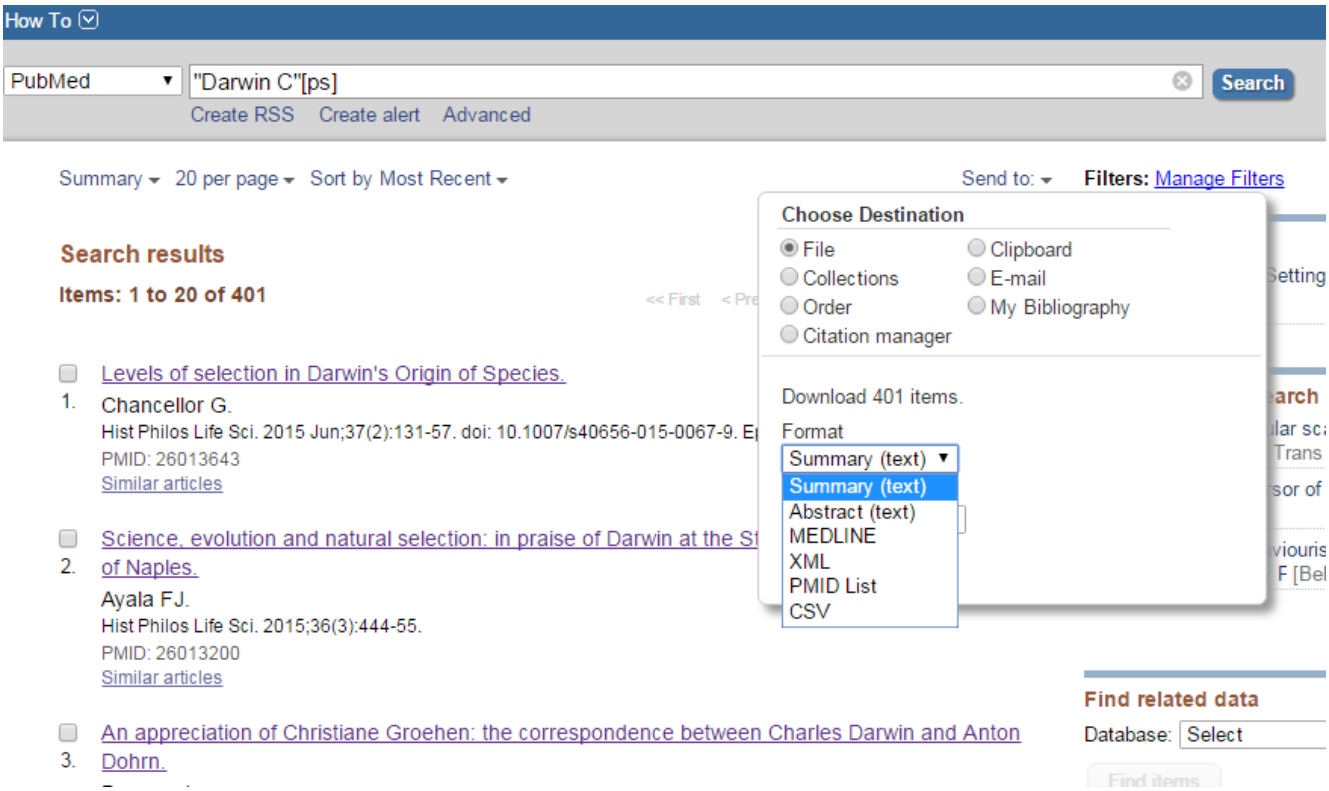

Figura 15. Formatos de fichero para descargar de PubMed (imagen original de [51])

- Summary (text)
- Abstract (text)
- MEDLINE
- XML
- PMID List
- CSV

Las características de cada uno de estos ficheros de datos se detallarán en el apartado, *4.3.2. Análisis de los datos*.

Una vez elegido e introducido el dominio de datos que se va a utilizar para realizar la comparativa entre un sistema NoSQL y uno SQL, en los siguientes apartados se llevará a cabo el Análisis, Diseño, Implementación y Pruebas de un repositorio de datos extraído a partir de PubMed y aplicado a dos tecnologías diferentes (NoSQL y SQL),

con el objetivo de ver qué problemas o ventajas surgen a la hora de representar unos datos en una tecnología u otra.

4.3. ANÁLISIS

Con el fin de facilitar la comprensión de la comparativa que se va a llevar a cabo, en este apartado se van a mostrar los objetivos que se quieren alcanzar.

4.3.1. INTRODUCCIÓN

El problema a tratar es la descarga de un repositorio de datos de la base de datos MEDLINE a través de PubMed y tratar estos datos con dos sistemas de almacenamiento diferentes: como ya se ha comentado previamente, uno NoSQL y uno SQL.

Para ello en primer lugar se realizará una búsqueda en PubMed, se descargarán los ficheros resultantes de esa búsqueda y se tratarán y convertirán los datos contenidos en ellos. Este tratamiento y conversión de datos se hará con el fin de que estos se adapten tanto a la tecnología NoSQL como a la SQL.

4.3.2. ANÁLISIS DE LOS DATOS

- **Estudio inicial**

Como ya se ha explicado en el apartado *4.2. Elección del dominio*, hay 6 tipos de ficheros que se pueden descargar a partir de una búsqueda. (Summary text, Abstract text, MEDLINE, XML, PMID List, y CSV).

Para ver cómo estructuran los datos cada fichero en primer lugar se va a realizar una búsqueda en PubMed. El criterio de búsqueda que se utilizará será "Darwin C"[ps], es decir, se buscarán todos aquellos artículos o citas que incluyan a Darwin como tema.

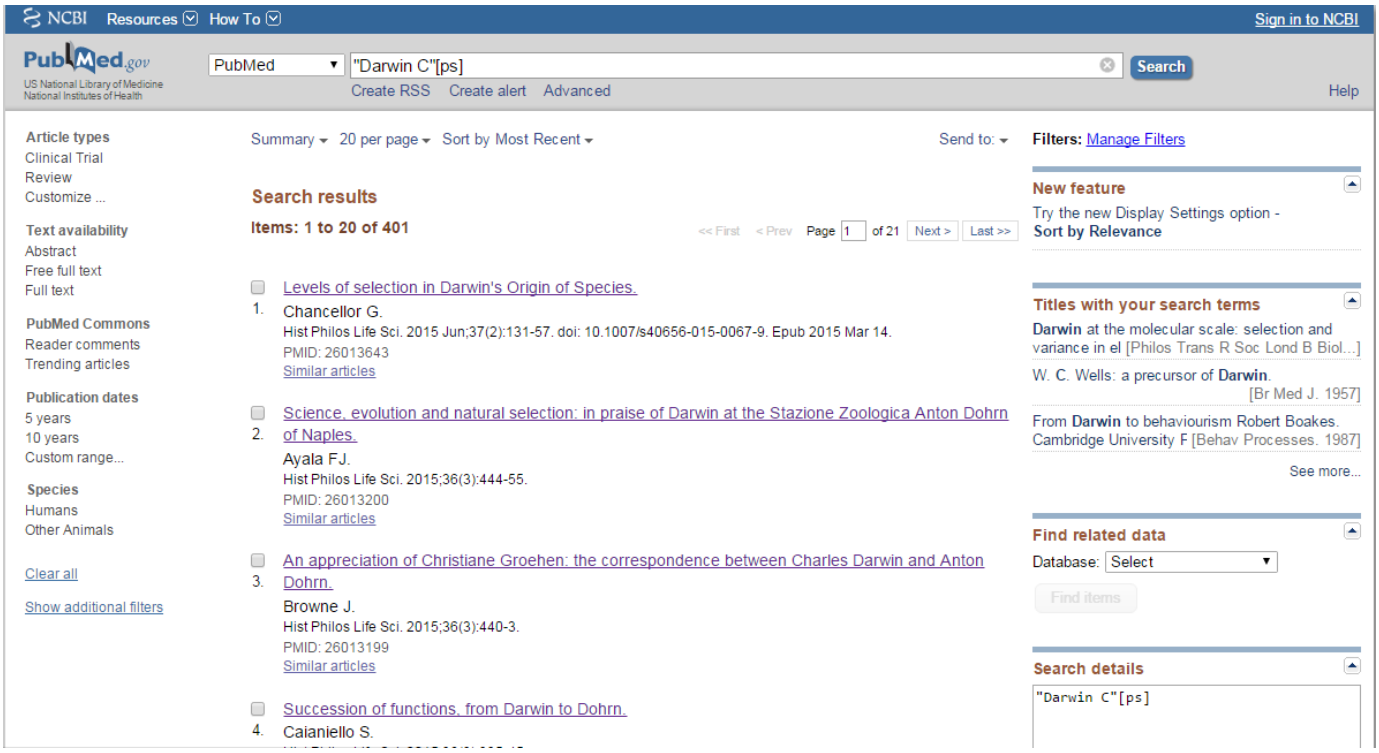

Figura 16. Criterio de búsqueda en PubMed (imagen original de [51])

Como se puede observar en la figura anterior, la búsqueda ha producido 400 resultados.

Una vez mostrados los resultados de la búsqueda podemos seleccionar un artículo[24] o bien descargar una colección o set de artículos. Para el caso práctico que se va a tratar se va a optar por esta segunda opción, la descarga de un fichero que contenga los 400 artículos.

Si se realiza la descarga de los 6 tipos de fichero para esta búsqueda, se observa que cada fichero estructura los datos de formas diferentes. A continuación se muestra cómo lo hacen:

- **Summary (text)**: fichero .txt cuyo formato por cada artículo o citación encontrada es la siguiente (números 1 a 400):

```
1: Chancellor G. Levels of selection in Darwin's Origin of Species. Hist Philos
Life Sci. 2015 Jun;37(2):131-57. doi: 10.1007/s40656-015-0067-9. Epub 2015 Mar
14. PubMed PMID: 26013643.


2: Ayala FJ. Science, evolution and natural selection: in praise of Darwin at the
Stazione Zoologica Anton Dohrn of Naples. Hist Philos Life Sci.
2015;36(3):444-55. PubMed PMID: 26013200.


3: Browne J. An appreciation of Christiane Groehen: the correspondence between
Charles Darwin and Anton Dohrn. Hist Philos Life Sci. 2015;36(3):440-3. PubMed
PMID: 26013199.
```

Figura 17. Formato de fichero Summary (text) en PubMed

[24]Algunos de ellos pueden ser accedidos de forma gratuita y otros tienen coste.

- **Abstract (text)**: fichero .txt cuyo formato de datos es:

```
1. Hist Philos Life Sci. 2015 Jun;37(2):131-57. doi: 10.1007/s40656-015-0067-9. Epub
2015 Mar 14.

Levels of selection in Darwin's Origin of Species.

Chancellor G(1).

Author information:
(1), Bury St Edmunds, UK, gordon.chancellor@btinternet.com.

References in Darwin's Origin of Species to competition between units of
selection at and above the level of individual organisms are enumerated. In many
cases these references clearly speak of natural selection and do not support the
view that Darwin thought selection only occurred at the level of the individual
organism. Darwin did see organismal selection as the main process by which
varieties were created but he also espoused what is here termed community and
varietal selection. He saw no essential difference between varieties and species
and the references show that he also believed that selection could operate at the
species level.

PMID: 26013643  [PubMed - indexed for MEDLINE]
```

Figura 18. Formato de fichero Abstract (text) en PubMed

- **MEDLINE**: fichero .txt que estructura información al estilo clave-valor. Muestra todos los campos encontrados en un artículo y a continuación su valor asociado.

```
PMID- 26013643
OWN - NLM
STAT- MEDLINE
DA  - 20150527
DCOM- 20150901
IS  - 0391-9714 (Print)
IS  - 0391-9714 (Linking)
VI  - 37
IP  - 2
DP  - 2015 Jun
TI  - Levels of selection in Darwin's Origin of Species.
PG  - 131-57
LID - 10.1007/s40656-015-0067-9 [doi]
```

Figura 19. Formato de fichero MEDLINE en PubMed

- **XML**: fichero con extensión xml. que muestra la definición de los campos entre etiquetas y su valor entre ellas.

```
<?xml version="1.0"?>
<!DOCTYPE PubmedArticleSet PUBLIC "-//NLM//DTD PubMedArticle, 1st January 2015//EN" "ht
<PubmedArticleSet>

<PubmedArticle>
    <MedlineCitation Owner="NLM" Status="MEDLINE">
        <PMID Version="1">26013643</PMID>
        <DateCreated>
            <Year>2015</Year>
            <Month>05</Month>
            <Day>27</Day>
        </DateCreated>
        <DateCompleted>
            <Year>2015</Year>
            <Month>09</Month>
            <Day>01</Day>
        </DateCompleted>
        <Article PubModel="Print-Electronic">
```

Figura 20. Formato de fichero XML en PubMed

- **PMID List**: lista de todos los PMID de los artículos encontrados

```
26013643
26013200
26013199
26013192
25827031
25515143
24941738
24921108
24570302
24300650
23705265
23664568
```

Figura 21. Formato de fichero PMID List en PubMed

...

- **CSV**: fichero CSV (Comma Separated Value). Separa los campos y sus valores a través de comas.

```
Title,URL,Description,Details,ShortDetails,Resource,Type,Identifiers,Db,EntrezUID,Properties
"Levels of selection in Darwin's Origin of Species.","/pubmed/26013643","Chancellor G.","Hist
"Science, evolution and natural selection: in praise of Darwin at the Stazione Zoologica Anton
"An appreciation of Christiane Groehen: the correspondence between Charles Darwin and Anton Dol
"Succession of functions, from Darwin to Dohrn.","/pubmed/26013192","Caianiello S.","Hist Phil
"Charles Darwin and the asylum letters.","/pubmed/25827031","King M.","Am J Psychiatry. 2015 Ap
"Darwin: German mystic or French rationalist?","/pubmed/25515143","Ghiselin MT.","Hist Philos
"Unsuitable for ladies?","/pubmed/24941738","Endersby J.","Br J Hist Sci. 2014 Jun;47(173 Pt 2
"Wallace, Darwin and Ternate 1858.","/pubmed/24921108","Smith CH.","Notes Rec R Soc Lond. 2014
```

Figura 22. Formato de fichero CSV en PubMed

Analizando el formato de cada uno de los ficheros a la hora de estructurar los datos para el set de artículos seleccionado, se eligen como mejores opciones los ficheros MEDLINE y XML por la manera en la que estructuran la información. En el caso de MEDLINE al estilo clave-valor y en el caso de XML por la definición de etiquetas y por ser un formato de archivo muy utilizado hoy en día y muy útil a la hora de realizar conversiones de datos. Esta decisión de elegir el mejor fichero de datos es importante para luego hacer la conversión a los ficheros que manejarán las dos bases de datos.

Finalmente se decide que el fichero que se va a elegir para tratar los datos sea MEDLINE txt, por la forma intuitiva de organizar la información, es decir, la presentación del campo y su valor asociado. Se descarta la opción de utilizar XML pese a que al ser la tecnología con mayores facilidades para cambiar el formato de datos, el fichero presenta demasiadas jerarquías y por tanto problemas a la hora de manipular esta información para posteriormente insertarla en las bases de datos. Se ha optado por elegir MEDLINE por una mayor facilidad de operar con los datos. De esta forma, a la hora de hacer asignaciones en una base de datos NoSQL y en una relacional la información estará ordenada al estilo campo-valor y será más fácil e intuitivo.

Los ficheros de datos obtenidos a partir de la búsqueda realizada contienen 400 artículos, cada uno de ellos con un número de campos variable. Algunos de estos campos son obligatorios y otros opcionales. Es importante saber cuántos campos componen un artículo o cita en PubMed. En la tabla que se muestra a continuación se visualizan todos los campos que puede presentar un artículo.

Field	Abbreviation	Field	Abbreviation	Field	Abbreviation
Abstract	(AB)	General Note	(GN)	Pagination	(PG)
Copyright Information	(CI)	Grant Number	(GR)	Personal Name as Subject	(PS)
Affiliation	(AD)	Investigator Name and Full Investigator Name	(IR) (FIR)	Full Personal Name as Subject	(FPS)
Investigator Affiliation	(IRAD)	ISBN	(ISBN)	Place of Publication	(PL)
Article Identifier	(AID)	ISSN	(IS)	Publication History Status	(PHST)
Author	(AU)	Issue	(IP)	Publication Status	(PST)
Author Identifier	(AUID)	Journal Title Abbreviation	(TA)	Publication Type	(PT)
Full Author	(FAU)	Journal Title	(JT)	Publishing Model	(PUBM)
Book Title	(BTI)	Language	(LA)	PubMed Central Identifier	(PMC)
Collection Title	(CTI)	Location Identifier	(LID)	PubMed Central Release	(PMCR)
Comments/Corrections	(atributo compuesto)	Manuscript Identifier	(MID)	PubMed Unique Identifier	(PMID)
Corporate Author	(CN)	MeSH Date	(MHDA)	Registry Number/EC Number	(RN)
Create Date	(CRDT)	MeSH Terms	(MH)	Substance Name	(NM)
Date Completed	(DCOM)	NLM Unique ID	(JID)	Secondary Source ID	(SI)
Date Created	(DA)	Number of References	(RF)	Source	(SO)
Date Last Revised	(LR)	Other Abstract and Other Abstract Language	(OAB)(OABL)	Space Flight Mission	(SFM)
Date of Electronic Publication	(DEP)	Other Copyright Information	(OCI)	Status	(STAT)
Date of Publication	(DP)	Other ID	(OID)	Subset	(SB)
Edition	(EN)	Other Term	(OT)	Title	(TI)
Editor and Full Editor Name	(ED) (FED)	Other Term Owner	(OTO)	Transliterated Title	(TT)
Entrez Date	(EDAT)	Owner	(OWN)	Volume	(VI)
Gene Symbol	(GS)	Volume Title	(VTI)		

Tabla 8. Atributos de un artículo en PubMed

Una vez conocidos todos los atributos o campos que puede contener cada artículo y el tipo de fichero (MEDLINE.txt) que se va a elegir para la transformación de los datos, se va a proceder a la conversión del fichero MEDLINE.txt para que pueda ser interpretado por ambos almacenes de datos, el NoSQL y el SQL.

En la siguiente figura se muestra un esquema de la conversión del fichero de entrada en los ficheros de salida necesarios para ser introducidos en las bases de datos correspondientes.

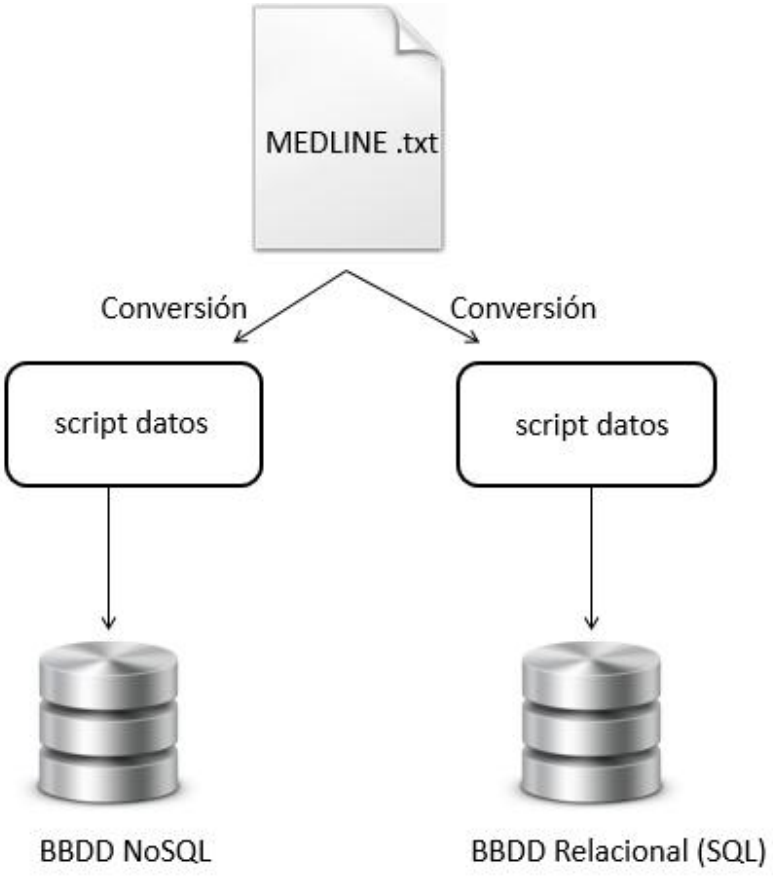

Figura 23. Esquema de conversión de ficheros desde MEDLINE.txt

4.3.3. ELECCIÓN DE LAS TECNOLOGÍAS

Para llevar a cabo este proyecto y una vez estudiados los datos que se van a manejar, se va a realizar la comparativa por parte de la tecnología NoSQL con el almacén de datos documental MongoDB, la cuarta herramienta de almacén de datos más utilizada hoy en día según el ranking mostrado en la *Figura 13* presente en el apartado *4.1 Introducción y motivación*, y por parte de las tecnologías SQL o el modelo de datos relacional se implementará con Oracle.

4.4. DISEÑO

Una vez visto el tipo de datos en el Análisis, en este apartado se especificará el proceso de diseño para preparar los datos.

4.4.1. MONGODB

INTRODUCCIÓN

Como ya se explicó en el *Estado del Arte*, en el apartado *3.2.7.2. Bases de datos Documentales*, MongoDB es un almacén de datos documental. Esto quiere decir que en vez de almacenar los datos como registros los almacena como documentos. Los documentos se almacenan en formato BSON, representación binaria del formato JSON.

Una de las diferencias más significativas con las bases de datos relacionales es que no es necesario seguir un esquema, los documentos incluidos en una colección pueden seguir diferentes esquemas.

¿Pero, qué es una colección? Y, ¿cómo se estructura la información en los documentos?

A continuación se muestra una figura que esquematiza la estructura y almacenamiento de los datos en MongoDB.

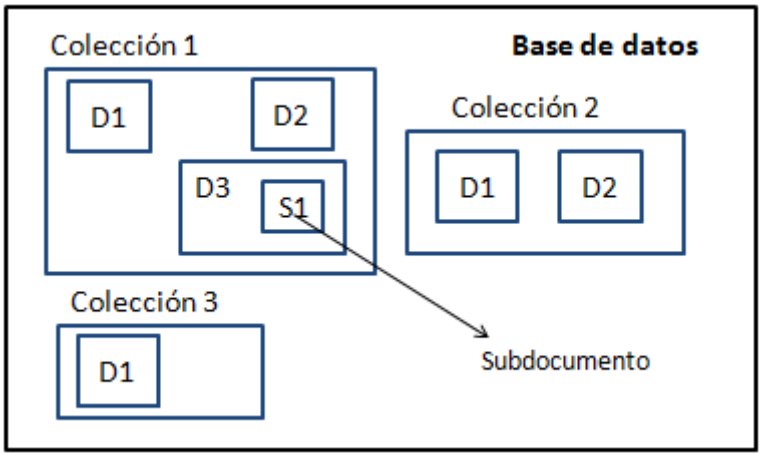

Figura 24. Esquema estructura de datos en MongoDB

Una colección no es más que un conjunto de documentos, lo que tradicionalmente en las bases de datos relacionales es una tabla. Por poner un ejemplo, si se tiene la colección Personas, un documento de esta colección podría almacenarse de la siguiente forma:

```
{
Nombre: “Ana”,
Apellidos: “Lorente Borox”,
Edad: 24,
Aficiones: [“música”, “cine”, “deporte”],
Compañeros: [
    {
     Nombre: “María”,
     Edad: 24
  },
  {
     Nombre: “Javier”,
     Edad: 23
  }
]
}
```

Este podría ser un documento incluido en la colección “Personas”. Como se puede observar, el inicio y fin de documento se declara con corchetes (“{“y “}”). Un documento puede incluir subdocumentos, declarados igualmente con inicio y fin de corchetes. Un subdocumento podría resumirse como el contenido de un campo en un documento JSON, que a su vez podría considerarse como un documento en sí mismo. En este sencillo ejemplo ya mostrado, el subdocumento es el llamado “Compañeros”. Este documento también tiene arrays, en la clave “Aficiones” para así poder insertar varios datos en una misma clave. Este ejemplo, por tanto, es un clásico ejemplo de documento JSON, donde se incluyen strings, números, arrays y subdocumentos.

En la misma colección llamada “Personas” también se podría almacenar un documento como el que sigue:

```
 {
     Nombre: “Pedro”,
     Edad: 23
  }
```

Este último documento no sigue el mismo esquema que el primero que se ha definido. Tiene únicamente dos campos. Esto da una idea de que las bases de datos documentales, en este caso MongoDB, proporcionan flexibilidad a la hora de declarar e insertar datos.

Una vez introducido un ejemplo básico de almacenamiento en MongoDB, a continuación se van a aplicar estas ideas al dominio elegido, los 400 artículos descargados de la herramienta PubMed.

MODELADO DE DATOS

Como se ha visto en el anterior apartado, en MongoDB no es necesario crear ningún esquema inicial donde se piense cómo se van a organizar los datos. Cada documento puede tener un número variable de campos. Únicamente habrá que preocuparse de cómo transformar los datos de MEDLINE a formato JSON.

Sabiendo esto, a la hora de analizar el fichero MEDLINE que contiene el registro de los 400 artículos, se puede observar que en este fichero por cada artículo hay un número variable de campos, lo que no supone ningún problema según lo indicado anteriormente.

Para conseguir el fichero de datos JSON a partir de MEDLINE se ha decidido interpretar el fichero de datos con los 400 artículos como una colección que incluya 400 documentos diferentes.

La apariencia inicial del fichero MEDLINE es la que se muestra en la siguiente figura:

```
PMID- 26013200
OWN - NLM
STAT- MEDLINE
DA  - 20150604
DCOM- 20150707
IS  - 0391-9714 (Print)
IS  - 0391-9714 (Linking)
VI  - 36
IP  - 3
DP  - 2015
TI  - Science, evolution and natural selection: in praise of Darwin at the Stazione
      Zoologica Anton Dohrn of Naples.
PG  - 444-55
AB  - Copernicus, Galileo, Newton and other physical scientists ushered in a conception
      of the universe as matter in motion governed by natural laws. Their discoveries
      brought about a fundamental revolution, namely a commitment to the postulate that
      the universe obeys immanent laws that can account for natural phenomena. The
      workings of the universe were brought into the realm of science: explanation
      through natural laws. Darwin completed the Copernican revolution by extending it
      to the living world. Darwin demonstrated the evolution of organisms. More
      important yet is that he discovered natural selection, the process that explains
      the 'design' of organisms. The adaptations and diversity of organisms, the origin
      of novel and complex species, even the origin of mankind, could now be explained
      by an orderly process of change governed by natural laws. The origin of species
      and the exquisite features of organisms had previously been explained as special
      creations of an omniscient God. Darwin brought them into the domain of science.
FAU - Ayala, Francisco J
AU  - Ayala FJ
LA  - eng
PT  - Biography
PT  - Historical Article
PT  - Journal Article
PL  - England
TA  - Hist Philos Life Sci
JT  - History and philosophy of the life sciences
JID - 8003052
SB  - IM
SB  - QIS
MH  - History, 19th Century
MH  - Italy
MH  - Marine Biology/*history
MH  - Natural History/*history
```

Figura 25. Artículo en formato MEDLINE

Esta figura muestra los campos que contiene el primer artículo. Como se puede ver, hay algunos campos duplicados, como por ejemplo “PT” Publication Type o “MH” MeSH Terms. Esto se interpreta como que estos campos pueden tener varios valores. En el caso de MongoDB los valores de estos campos se almacenarán como un array, resultando, por ejemplo para el caso de la variable “PT”, lo siguiente: PT: [“Biography”, “Historical Article”, “Journal Article”, “England”]. Este procedimiento se aplicará a todos los campos que contengan más de un valor.

Analizando el fichero MEDLINE.txt también se observa que los artículos empiezan siempre con el campo “PMID” y terminan con el campo “SO”. El “PMID” “PubMed Unique Identifier” identifica de forma única cada artículo, por lo que a la hora de hacer búsquedas, esto puede ser de interés.

Que todos los artículos terminen con el campo "SO" ayuda a la hora de modelar los datos. De esta forma se sabe que un documento (artículo) empezará cuando se encuentre el término PMID y terminará cuando se encuentre el término SO. Por tanto, se modelará el fichero de datos para MongoDB como sigue:

```
{
PMID: "123456",
OWN: "NLM",
... : "...",
... : "...",
SO: "TEXTO"
}
```

Este formato se aplicará para los 400 artículos, para que finalmente se disponga de un fichero con 400 artículos cada uno de ellos empezando por "{"y terminando por "}".

4.4.2. ORACLE DATABASE

INTRODUCCIÓN

Oracle Database es un Sistema Gestor de Base de datos Relacional, que por tanto sigue el modelo de datos relacional. Ya en el *Estado del Arte*, en el apartado *3.1.3. Bases de datos Relacionales* se introdujeron las bases de datos relacionales. A continuación se va a hablar más en profundidad de cómo éstas almacenan y estructuran sus datos.

El modelo de datos relacional es uno de los más utilizados actualmente para el diseño de bases de datos. Está basado en la lógica de predicados y la teoría de conjuntos, lo que implica que se puede utilizar tanto el álgebra como el cálculo relacional para operar con los datos almacenados. El álgebra para describir la forma de realizar una consulta y el cálculo para obtener los valores que se desean devolver.

A continuación se describirán las características y los elementos básicos que conforman una base de datos relacional.

Las bases de datos relacionales se componen de tablas. Cada tabla está formada por un conjunto de registros llamados filas o tuplas. Una tupla es una asociación de valores correspondidos con un conjunto de atributos. No existen dos tuplas iguales y cada atributo toma un solo valor del dominio en cada tupla. Las tablas también están

formadas por columnas, subdivisiones lógicas que las conforman. Como norma, no puede haber dos tablas con el mismo nombre.

Una tabla se relaciona con otra a través de una relación. Las relaciones se conforman a partir de claves primarias y ajenas. Ambas claves, primarias y ajenas, deben cumplir la característica de ser unívocas y mínimas. Unívoca quiere decir que a cada valor de la clave le corresponde como mucho una tupla, y mínima, que no existe ningún subconjunto suyo que también sea clave.

La clave primaria es la clave principal dentro de una tabla y además debe cumplir con la norma de Integridad de Entidad, que define que ningún atributo de la clave primaria puede tomar valor nulo.

La clave ajena se coloca en las tablas hijas y contiene el mismo valor que la clave principal de la tabla padre a la que hace referencia. La clave ajena debe cumplir la Integridad Referencial, es decir, que lo referenciado por clave ajena debe existir. Además la clave ajena solo puede adoptar valores que existan en la clave de la relación a la que referencia. Para cumplir con la Integridad Referencial existen una serie de normas llamadas Reglas de Integridad, referidas a acciones que puedan rompen la restricción referencial.

En la siguiente figura se muestra un esquema de almacenamiento en bases de datos relacionales:

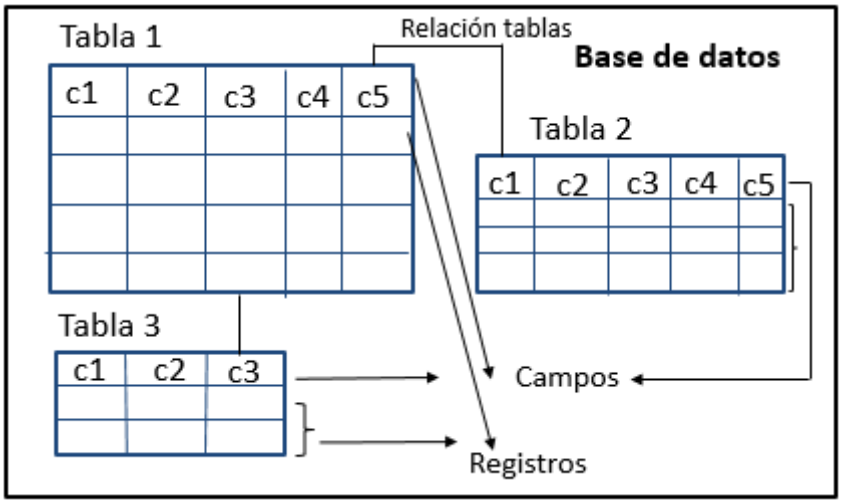

Figura 26. Esquema almacenamiento de datos en BBDD relacional

Como ya se ha comentado, la característica principal de las bases de datos relacionales radica en la relación entre las tablas.

Los datos se separan de tal forma que una colección o un conjunto de registros de similares características se almacenen en una misma tabla. Por poner un ejemplo, si se tiene un registro de Personas, identificadas por un campo “ID”, y registradas con los

campos “Nombre”, “Empresa” y “Años” todas las “personas” o registros que estén incluidos en esta tabla deberán tener estos campos. La codificación y definición de los atributos podrá tomar valores nulos en caso de no tener algún valor de algún atributo, pero aunque esto pase es necesaria e imprescindible la definición de todos los campos que forman la tabla. No puede haber registros en una misma tabla con distinto número de campos. En el ejemplo, la tabla “Personas” está relacionada con la tabla “Empresas”. Gracias a esta relación podremos cruzar campos de ambas tablas y averiguar, por ejemplo qué personas trabajan en qué empresas y en qué lugar.

TABLA DE PERSONAS			
ID	NOMBRE	EMPRESA	AÑOS
1	Miguel Díaz	Accenture	24
2	Javier Fernández	BBVA	23
3	Natalia Sanz	Iberia	23
4	Ana Lorente	Santander	24

TABLA DE EMPRESAS		
CÓDIGO	UBICACION	EMPRESA
1841	Valencia	Vodafone
1684	Barcelona	Repsol
2187	Madrid	BBVA
2646	Madrid	Samsung

Tabla 9. Ejemplo de almacenamiento en base de datos relacional

MODELADO DE DATOS

Una vez explicado en qué consiste el modelo de datos relacional e introducido un ejemplo, se va a proceder a aplicar estos conocimientos al dominio de datos del problema que se está tratando.

Debido a que los datos en una base relacional se estructuran en tablas, es necesario organizar y diseñar un sistema que agrupe los campos que componen un artículo en tablas. Partiendo del fichero MEDLINE que contiene 400 artículos, se puede observar que no todos los artículos tienen el mismo número de campos. Como se ha visto anteriormente, en MongoDB esto no supone ningún problema, pero en una base de datos relacional sí, ya que se sigue un esquema fijo en el que a la hora de insertar y consultar datos se deben tener todos los campos declarados. Para poder estructurar toda la información de un artículo en una base de datos relacional se ha empezado realizando un diseño conceptual entidad/relación en el que se visualizan las futuras tablas (entidades) que compondrán el sistema. Para llevar a cabo este diseño se ha cogido la

totalidad de los campos que pueden estar presentes en un artículo (*Tabla 6*) y se han clasificado en varias categorías[25] según la característica de sus datos.

Antes de mostrar el diagrama entidad-relación que se ha elaborado, se indican unas observaciones previas sobre los campos que componen un artículo. Estas observaciones se han extraído a partir de la información dada en los campos, presentes en [46] :

- En la definición de los campos presente en PubMed se define el término "record" como cada registro que aparece en una colección, independientemente del tipo de registro que sea. Según se analizan los campos, un registro puede ser un "Journal Article", lo que es un artículo, o un "Book" un libro. Por ello, se ha decidido establecer como entidad padre "RECORD", y dentro de ella estarán los tipos.

- Los libros no se incluyen en PubMed; estos están en la base de datos NCBI Books Database. Puede haber alguna referencia a un libro en la colección de registros, por lo que se define algún campo. A la hora de hacer el diagrama entidad-relación se marcará la entidad "BOOK" para indicar conceptualmente que es un tipo de registro que existe, pero que no se relacionará con ninguna otra entidad al no estar como tal en PubMed, que es el dominio que se está tratando.

- Se ha diferenciado entre autor e investigador ya que según se indica, los investigadores no son autores. Los investigadores pertenecen a un grupo en el que pueden colaborar a la hora de elaborar un artículo pero no se consideran autores.

- Definición de claves primarias. El PMID se define como la clave primaria de todo registro que aparece en PubMed. Por ello la entidad "RECORD" tendrá como clave primaria el PMID. En cuanto a las demás entidades, se definen claves únicas e identificativas para cada una de ellas. Por ejemplo, para un artículo, o "Journal Article" se define el "ISSN", que aparece como "IS". Este campo identifica unívocamente un artículo.

Vistas estas consideraciones previas y antes de presentar el diagrama E/R, a continuación se muestra la división de los atributos realizada en sus entidades correspondientes. En rojo están marcados las claves primarias de cada entidad, en naranja las claves alternativas, y en azul oscuro los atributos compuestos:

[25] Para cada "categoría" de datos se define una entidad, conocida así en el esquema entidad-relación

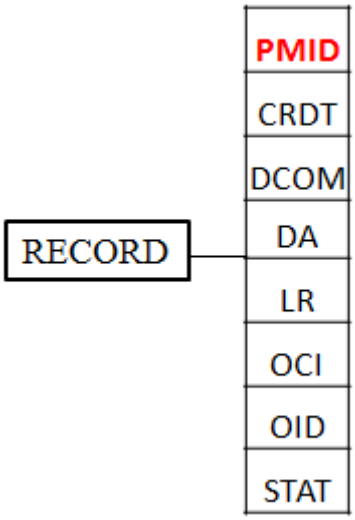

Figura 27. Clasificación de atributos para entidad "RECORD"

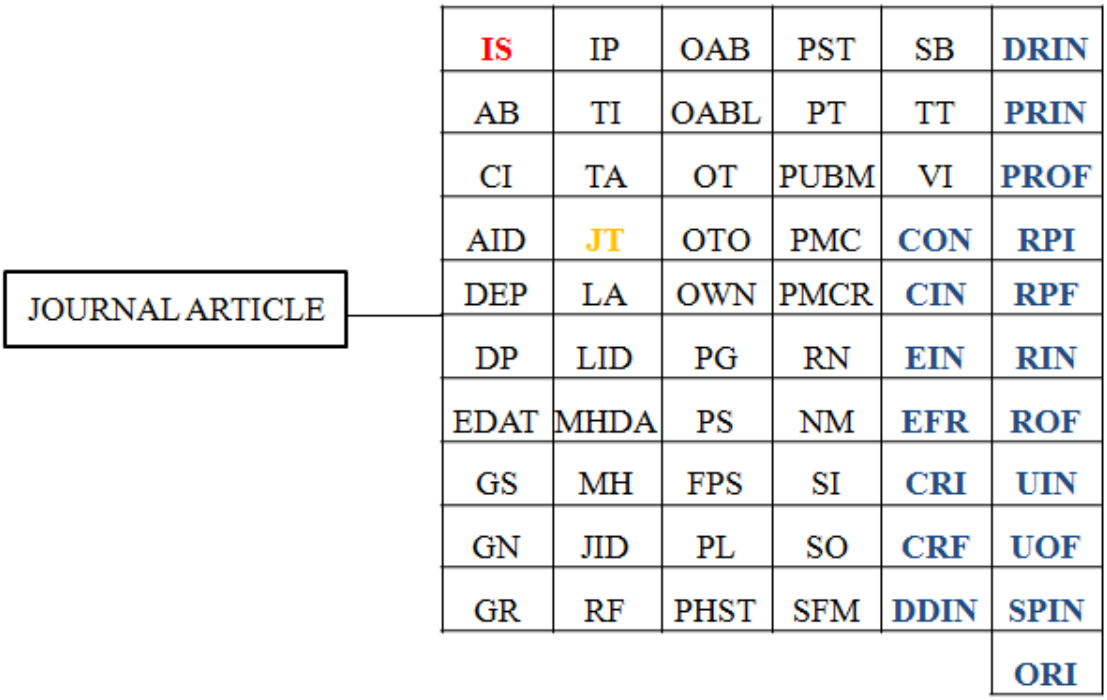

Figura 28. Clasificación de atributos para entidad "JOURNAL ARTICLE"

En este caso, los atributos marcados en azul oscuro señalan un campo "Comment/Corrections" que está compuesto por todos esos atributos. En la *Tabla 6 Atributos de un artículo en PubMed* este campo estaba vacío, aquí se muestran todos los atributos que lo forman.

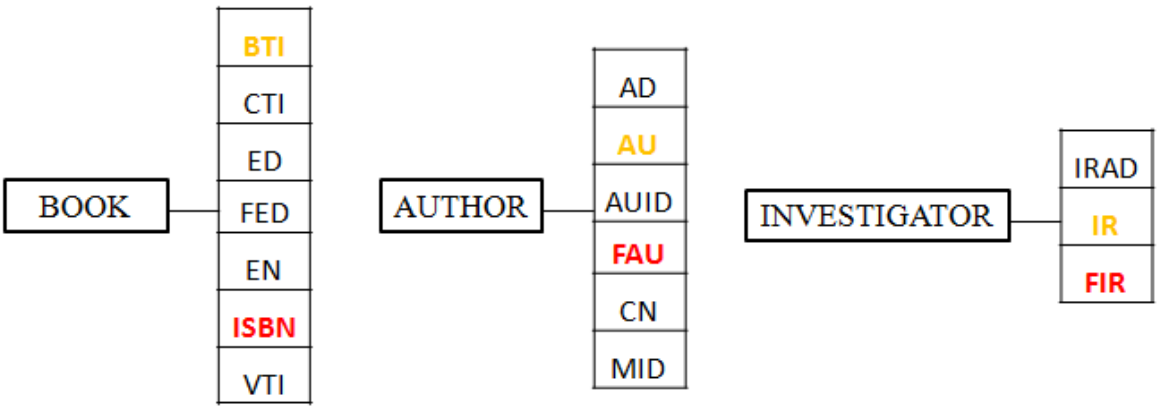

Figura 29. Clasificación de atributos entidades "BOOK", "AUTHOR", "INVESTIGATOR"

Una vez clasificados todos los atributos en sus entidades correspondientes, finalmente se presenta el **diagrama entidad-relación**:

Nota: no se han plasmado todos los campos en este diagrama. Se han puesto los más importantes, las claves principales y secundarias. Con la definición anterior de los campos en cada entidad se rellena este E/R. Para el caso del atributo compuesto "Comment Corrections" se añade un campo en "…" que muestra que faltan los atributos restantes.

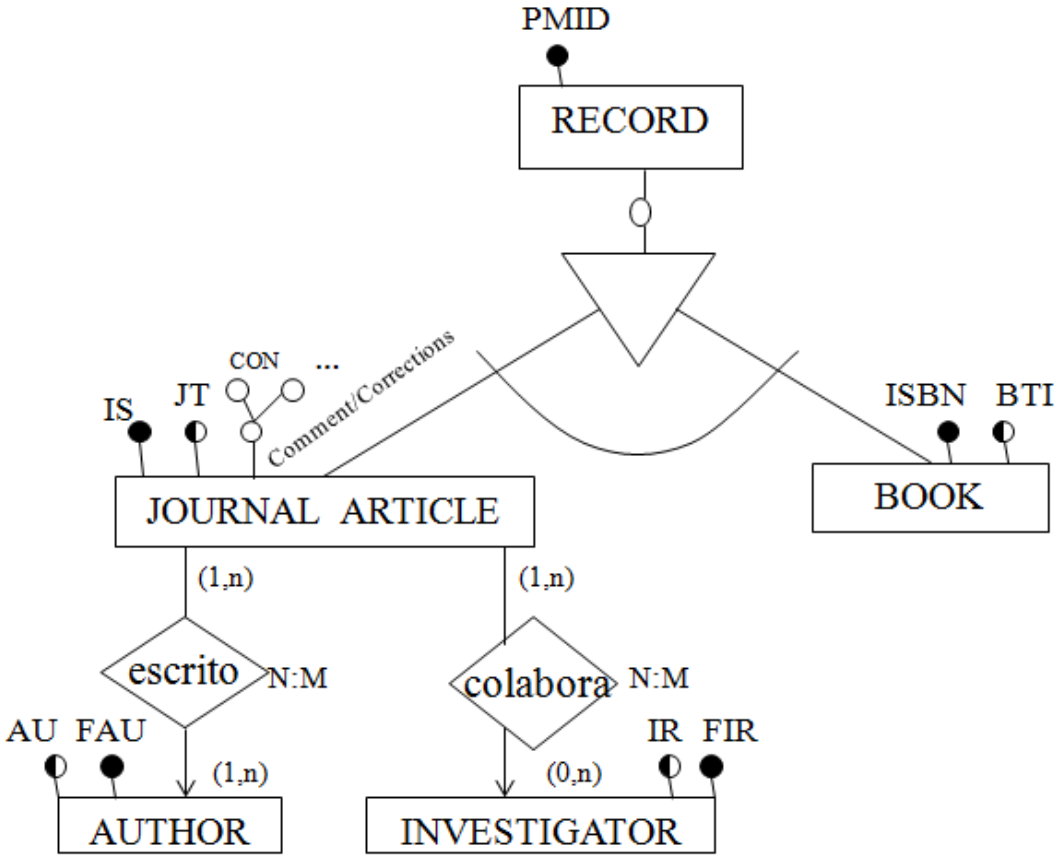

Figura 30. Diagrama entidad-relación

4.5. IMPLEMENTACIÓN

4.5.1. MONGODB

SCRIPT DE DATOS

En el apartado de *Diseño* ya se ha hablado de cómo tiene que ser la estructura de los datos a insertar. Para transformar el fichero MEDLINE en un script de datos interpretado por MongoDB se ha utilizado la herramienta Microsoft Excel. La transformación se ha realizado siguiendo los pasos que se indican:

1. Se han sustituido las comillas dobles por comillas simples. Por ejemplo, si en la descripción de un campo como el "Abstract" (AB) aparecen comillas dobles en alguna parte del comentario para hacer referencia a algún término, esto se tiene que sustituir por comillas simples. Se hace ya que la definición del valor en el par clave-valor empieza y termina por comillas dobles, para que no haya ningún problema a la hora de separar los datos.

2. Como en el fichero MEDLINE.txt cada registro de un artículo comienza por el campo PMID y termina por el campo SO, se han separado los artículos teniendo en cuenta esto. Cada artículo empieza con un "{"y termina con un "}".

3. Cada par clave-valor se separa por comas. Para ello se han concatenado todos los campos de un artículo y se han separado por comas.

Un ejemplo de cómo queda el fichero de script de datos que se insertará en MongoDB se indica en la siguiente figura:

Variable		Valor
{		
PMID	:	"26013643",
OWN	:	"NLM",
STAT	:	"MEDLINE",
DA	:	"20150527",
DCOM	:	"20150901",
IS	:	"0391-9714 (Print) 0391-9714 (Linking)",
VI	:	"37",
IP	:	"2",
DP	:	"2015 Jun",
TI	:	"Levels of selection in Darwin's Origin of Species.",
PG	:	"131-57",
LID	:	"10.1007/s40656-015-0067-9 [doi]",
AB	:	"References in Darwin's Origin of Species to competition between
FAU	:	"Chancellor, Gordon",
AU	:	"Chancellor G",
AD	:	", Bury St Edmunds, UK, gordon.chancellor@btinternet.com.",
LA	:	"eng",
PT	:	"Biography Historical Article Journal Article",
DEP	:	"20150314",
PL	:	"England",

Figura 31. Script de datos versión Excel

Estos datos presentes en el fichero Excel se pasan a un fichero con extensión .JSON, quedando como resultado la figura que se muestra a continuación. Estos datos están ya preparados para ser insertados en el almacén documental MongoDB:

```
{
PMID    :   "26013643",
OWN :   "NLM",
STAT    :   "MEDLINE",
DA  :   "20150527",
DCOM    :   "20150901",
IS  :   "0391-9714 (Print) 0391-9714 (Linking)",
VI  :   "37",
IP  :   "2",
DP  :   "2015 Jun",
TI  :   "Levels of selection in Darwin's Origin of Species.",
PG  :   "131-57",
LID :   "10.1007/s40656-015-0067-9 [doi]",
AB  :   "References in Darwin's Origin of Species to competition between units of selection at and above the
FAU :   "Chancellor, Gordon",
AU  :   "Chancellor G",
AD  :   ", Bury St Edmunds, UK, gordon.chancellor@btinternet.com.",
LA  :   "eng",
PT  :   "Biography Historical Article Journal Article",
DEP :   "20150314",
PL  :   "England",
TA  :   "Hist Philos Life Sci",
JT  :   "History and philosophy of the life sciences",
JID :   "8003052",
SB  :   "IM QIS",
MH  :   "*Biological Evolution History, 19th Century Natural History/*history *Selection, Genetic",
PS  :   "Darwin C",
FPS :   "Darwin, Charles",
EDAT    :   "28/05/2015 06:00",
MHDA    :   "02/09/2015 06:00",
CRDT    :   "28/05/2015 06:00",
PHST    :   "2014/08/29 [received] 2015/03/06 [accepted] 2015/03/14 [aheadofprint]",
AID :   "10.1007/s40656-015-0067-9 [doi]",
PST :   "ppublish",
SO  :   "Hist Philos Life Sci. 2015 Jun;37(2):131-57. doi: 10.1007/s40656-015-0067-9. Epub 2015 Mar 14."
}
{
PMID    :   "26013200",
OWN :   "NLM",
STAT    :   "MEDLINE",
```


Figura 32. Script de datos .JSON

CREACIÓN DE LA BASE DE DATOS Y LA COLECCIÓN

La creación de la base de datos y la colección que incluirá los 400 documentos se define en el *Anexo IV. Manual de uso de MongoDB.* Con sencillos comandos se creará la base de datos y la colección que contendrá los 400 documentos tal y como se ha definido en el apartado de *Diseño.* Para MongoDB no es necesario tener ninguna consideración más en lo que se refiere al tratamiento de datos. Como no se necesita definir un esquema inicial en lo que se refiere a la estructura y organización de los datos y se permite que cada documento tenga un número variable de registros, estos pasos son los suficientes para insertar los datos en el almacén documental.

4.5.2. ORACLE DATABASE

Una vez hecho el diseño relacional y teniendo claro cómo se van a estructurar los datos en las distintas entidades definidas, a continuación se continúa con el siguiente paso, la preparación de los datos y la definición y creación de tablas donde irán almacenados. Para ello en este apartado se hablará en primer lugar de la preparación del script de

datos y en segundo lugar se presentará el esquema relacional donde se verá más concretamente la relación entre tablas. En este segundo paso se mostrarán también los pasos necesarios para la implementación e inserción de tablas y datos, respectivamente.

SCRIPT DE DATOS

Hecho el "Modelado de datos" en la parte de *Diseño*, falta preparar los datos para que puedan ser interpretados por Oracle. Como ya se ha comentado anteriormente, se ha tomado como fichero de inicio MEDLINE.txt, que almacena la información al estilo clave-valor. Las inserciones en las tablas de una base de datos relacional se hacen por filas. Por ejemplo, si se quiere insertar un nuevo registro en una tabla llamada "personas", los diferentes campos se separan por comas[26], resultando lo siguiente:

```
INSERT INTO personas VALUES ('123456A', 'Linda', 'Figueroa');
```

Si se quisiera insertar otro registro se seguiría el formato del ejemplo anterior, con otros datos.

En el dominio que se ha escogido se tienen 400 registros, por lo que habrá 400 líneas de inserción de datos.

Para transformar el fichero MEDLINE en un script de datos interpretado por Oracle se ha utilizado la herramienta Microsoft Excel. La transformación se ha realizado siguiendo los pasos que se indican:

1. Se han sustituido todas las comillas simples por comillas dobles. Por ejemplo, en el caso de un apellido o frase en inglés que las incluya. Se hace esto porque la separación de términos va determinada por comillas simples.

2. Como en el fichero MEDLINE.txt cada registro de un artículo comienza por el campo PMID y termina por SO, se han separado los artículos teniendo en cuenta esto. Para cada uno de ellos hay una línea de inserción de datos como la mostrada en el ejemplo anterior. Se han agrupado los campos de un mismo artículo en una línea y se han separado con comas y añadiendo comillas simples en la definición de cada atributos. Para lograr esta transformación se han transpuesto las columnas en filas y se ha utilizado la función "concatenar".

3. Como ya se ha comentado, no todos los artículos tienen el mismo número de campos, lo que supone un problema en una base de datos relacional, donde todos los registros de una tabla tienen que tener el mismo número de atributos. Para ello, se ha cogido la totalidad de los campos que pueden aparecer en un artículo (como se hizo en el apartado de *Diseño*) y con una

[26] En este ejemplo de inserción en la tabla personas el primer dato podría corresponder al DNI, el segundo al nombre y el tercero al apellido

función se han buscado cuáles faltan. El valor del atributo que falta para cada artículo se ha definido con valores "NULL". De esta forma se declaran todos los campos aunque algunos no aparezcan inicialmente en el fichero MEDLINE.txt

4. Una vez se tiene la totalidad de los campos, tantos los declarados y existentes en el fichero inicial MEDLINE como los que faltan y que han sido buscados, se colocan todos en una misma fila y se hace una trasposición de datos. Hecho esto ya está listo el script de datos para ser interpretado por Oracle.

A continuación se muestra un ejemplo de cómo queda el script de datos:

	AB	AD	AID	AU	AUID	BTI	CI	CN	CRDT
2	('Referenc	. Bury St Edmu	'10.1007/s40656-015-0067-9 [doi]',	'Chancellor G',	NULL,	NULL,	NULL,	NULL,	'28/05/2015 06:00',
3	(Copernic	NULL,	'10.1007/s40656-014-0031-0 [doi]',	'Ayala FJ',	NULL,	NULL,	NULL,	NULL,	'28/05/2015 06:00',
4	(Anton Dc	NULL,	'10.1007/s40656-014-0032-z [doi]',	'Browne J',	NULL,	NULL,	NULL,	NULL,	'28/05/2015 06:00',
5	(By formu	NULL,	'10.1007/s40656-014-0041-y [doi]',	'Caianiello S',	NULL,	NULL,	NULL,	NULL,	'28/05/2015 06:00',
6	(NULL,	From Summit E	'10.1176/appi.ajp.2014.14101260 [doi]',	'King M',	NULL,	NULL,	NULL,	NULL,	'02/04/2015 06:00',
7	(The notic	NULL,	'10.1007/s40656-014-0033-y [doi]',	'Ghiselin MT',	NULL,	NULL,	NULL,	NULL,	'18/12/2014 06:00',

Figura 33. Script de datos versión Excel

5. En último lugar faltaría concatenar estas filas con lo siguiente: "`INSERT INTO RECORD VALUES`" para terminar de definir el script de datos. El fichero resultante tendrá extensión "SQL".

Estos datos presentes en el fichero Excel se pasan a un fichero con extensión .SQL, quedando como resultado la figura que se muestra a continuación. Este script de datos que se muestra ya está preparado para ser interpretado por Oracle:

```
INSERT INTO TOTAL_RECORDS VALUES ('References in Darwin"s Origin of Species to competition betw
INSERT INTO TOTAL_RECORDS VALUES ('Copernicus Galileo. Newton and other physical scientists usl
INSERT INTO TOTAL_RECORDS VALUES ('Anton Dohrn was introduced to Darwinism by Ernst Haeckel du
INSERT INTO TOTAL_RECORDS VALUES ('By formulating in 1875 his major theoretical achievement. tl
```

Figura 34. Script de datos SQL

ESQUEMA RELACIONAL Y CREACIÓN DE TABLAS

Para llevar a cabo la implementación de las tablas se va a realizar un esquema relacional que parte del diagrama entidad-relación elaborado en la parte de *Diseño*. Con este

esquema se visualizarán más concretamente las tablas y los elementos que las conforman[27].

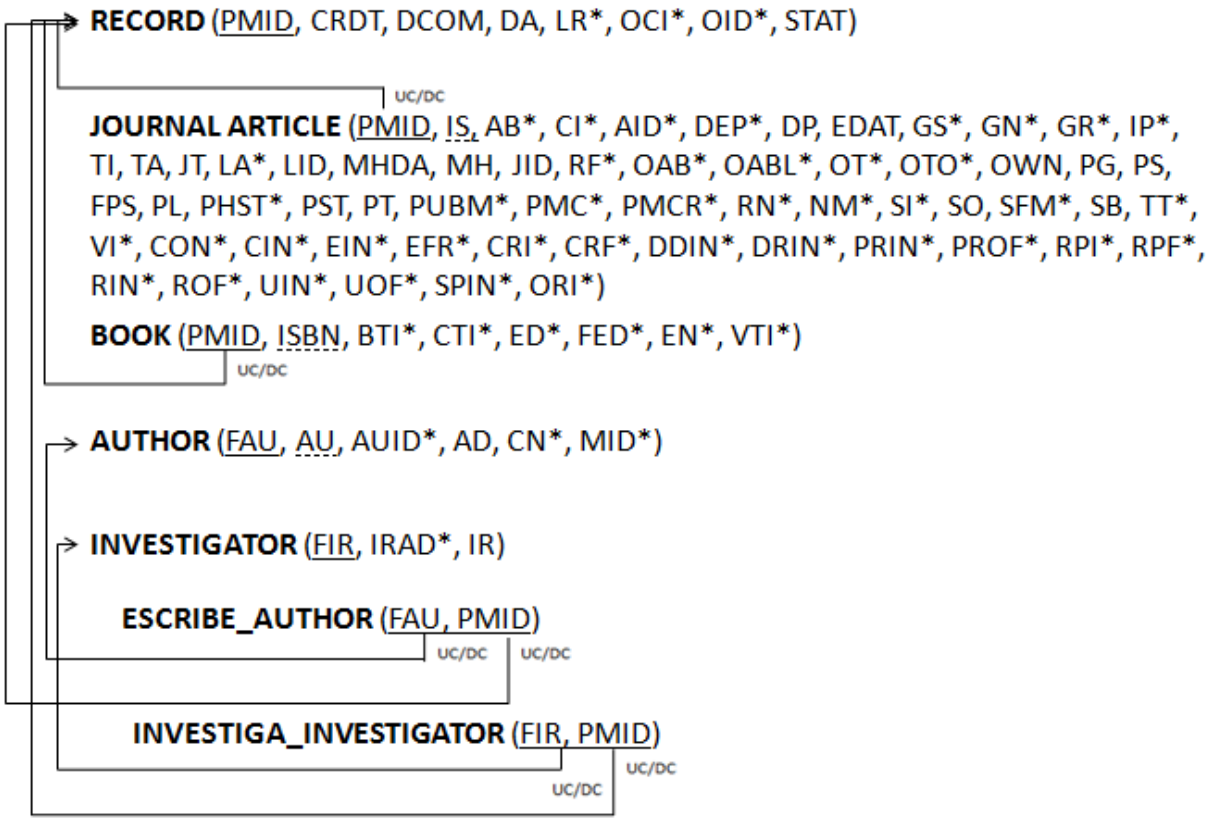

Figura 35. Esquema relacional

Para llevar este esquema a la práctica se va a llevar a cabo un proceso, que se define en la siguiente figura:

[27] La clave primaria es la subrayada en línea continua, la alternativa es la subrayada en línea discontinua y los atributos con "*" son los marcados como opcionales, donde en el caso de que no tengan valor aparecerán como "NULL"

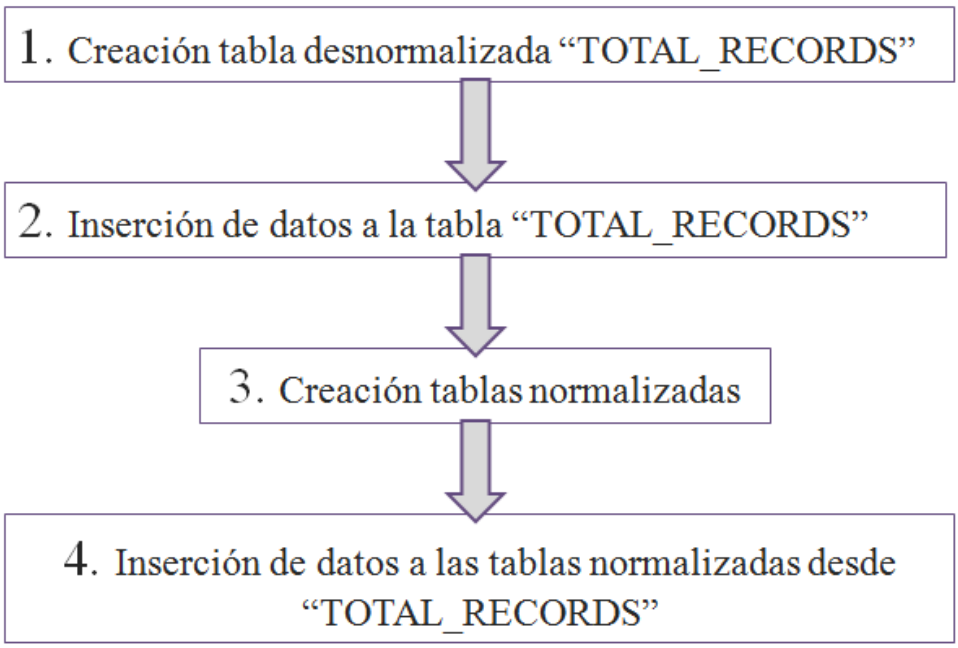

Figura 36. Esquema de implementación: creación de tablas e inserción de datos

A continuación se explica en qué consiste este proceso de implementación:

1. Creación de tabla desnormalizada "TOTAL_RECORDS". Como inicialmente no se sabe qué campos faltan en cada artículo del fichero, se ha decidido empezar la creación de una tabla llamada "TOTAL_RECORDS" que incluya todos los campos que puedan existir en un artículo.

```
CREATE TABLE TOTAL_RECORDS (
    PMID VARCHAR2(8),
    CRDT VARCHAR2(20),
    DCOM VARCHAR2(20),
    DA VARCHAR2(10),
    LR VARCHAR2(10),
    OCI VARCHAR2(20),
    OID VARCHAR2(20),
    STAT VARCHAR2(10),
    AB VARCHAR2(200),
    CI VARCHAR2(20),
    AID VARCHAR2(10),
    DEP VARCHAR2(20),
    DP VARCHAR2(20),
    EDAT VARCHAR2(10),
    GS VARCHAR2(10),
    GN VARCHAR2(10),
    GR VARCHAR2(10),
    ISSN VARCHAR2(10),
    IP VARCHAR2(7),
    TI VARCHAR2(50),
    TA VARCHAR2(7),
    JT VARCHAR2(10),
    LA VARCHAR2(7),
    LID VARCHAR2(7),
    MHDA VARCHAR2(7),
    MH VARCHAR2(50),
    JID VARCHAR2(10),
    RF VARCHAR2(7),
    OAB VARCHAR2(7),
```

Figura 37. Creación de tabla TOTAL_RECORDS[28]

2. Se insertan todos los datos en la tabla "TOTAL_RECORDS". Para ello se utiliza el script de datos obtenido en el apartado anterior (*Figura 32. Script de datos SQL).*

3. Se crean las tablas normalizadas. Estas son las que se han definido en el esquema relacional. La creación de las tablas con los atributos que le corresponde, indicando cuáles son opcionales y cuáles obligatorios (`NOT NULL`). Ejemplo de la tabla de autores:

```
CREATE TABLE AUTHOR (
    FAU VARCHAR2(40) PRIMARY KEY NOT NULL,
    AU  VARCHAR2(40) NOT NULL,
    AD VARCHAR2(40) NOT NULL,
    AUID    VARCHAR2(15),
    CN  VARCHAR2(15),
    MID1    VARCHAR2(20),
    );
    --Tabla que almacena los autores
```

Figura 38. Creación tabla de autores

[28] Esta tabla contiene todos los atributos que puede tener un artículo. En la imagen no se muestran todos

4. Se insertan los datos contenidos en la tabla “TOTAL_RECORDS” (hecho en el paso 2) en las tablas normalizadas.

```
INSERT INTO AUTHOR (FAU_A, AU_A, AD_A, AUID_A, CN_A, MID1_A)(
select distinct FAU,AU, AD, AUID, CN, MIDL from TOTAL_RECORDS
);

--Insertamos,evitando redundancias (distinct) los autores
```

Figura 39. Inserción a tabla normalizada de autor desde TOTAL_RECORDS

4.6. PRUEBAS

Las pruebas son una serie de procesos de evaluación del proyecto creado cuyo objetivo es proporcionar información sobre la calidad del mismo.

En este apartado se mostrarán las pruebas básicas que habrá que realizar para comprobar si las nociones básicas de consulta

4.6.1. MONGODB

En MongoDB se ofrece la posibilidad de consultar los datos almacenados, lo que demuestra que no es un simple sistema de almacenamiento de pares clave-valor.

MongoDB puede utilizar funciones Map y Reduce[29] que están escritas en Javascript. Mediante estas funciones se pueden seleccionar los campos que interesen y unirlos de la manera que se quiera.

En MongoDB también se pueden hacer consultas al valor de un campo específico. Por ejemplo si se quiere obtener un artículo cuyo PMID es 1213456, se escribirá:

```
db.colección.find({PMID : 123456})
```

Con esta instrucción se mostrará el documento cuyo PMID sea el 123456.

Por tanto si se desea buscar información específica dentro de un documento se utilizará el parámetro find (). Dentro de este parámetro se detallarán los criterios de búsqueda.

A continuación se presentan algunos operadores para buscar información específica en un documento:

Operador	Significado
$eq	Igual
$lte	Menor o igual
$lt	Menor que
$gte	Mayor o igual
$gt	Mayor que
$in	Conjunto de valores. Ejemplo DATE:{$in:[2012,2013]}
$and	se deben cumplir 2 o más condiciones Ejemplo $and:[{“PMID”:”123456”},{“LA”:eng}]

[29] MapReduce es un modelo de programación que da soporte a la computación paralela sobre grandes cantidades de datos. Son funciones que se aplican sobre pares de datos clave-valor

Operador	Significado
$or	Por lo menos una condición se debe cumplir. Ejemplo $or:[{“PMID”:”123456”},{“LA”:eng}]
$max	Valor máximo {$max:"$monto"}
$min	Valor mínimo {$min:"$monto"}

Estos son algunos de los operadores más utilizados. También hay operadores de agregación (GROUP, SUM), operadores de agrupamiento (COUNT), operadores para la muestra de valores no repetidos y muchos otros más [47].

Con estos operadores se podrán efectuar las búsquedas que se deseen realizar.

4.6.2. ORACLE DATABASE

Para hacer consultas en Oracle se puede utilizar el álgebra y cálculo relacional para definir y obtener los valores de una consulta realizada.

La representación del dominio en una base de datos relacional permite hacer la operación de JOIN para así combinar columnas de elementos de varias tablas.

En primer lugar, para realizar una consulta sobre una tabla relacional es preciso utilizar la sentencia `SELECT FROM`. Así, por ejemplo si se desea obtener información sobre un artículo definido con PMID: 123456, se escribirá:

```
SELECT * FROM JOURNAL_ARTICLE  WHERE PMID = 123456;
```

En SQL las consultas son muy variadas. En álgebra relacional existen operaciones unarias, operaciones de conjuntos, operaciones de combinación, operaciones de división y operaciones de agregación y agrupamiento (`GROUP BY`, `COUNT`, `SUM`, `AVG`…). Cada una de ellas con operadores específicos.

Con todos estos operadores se podrán efectuar las consultas que se deseen sobre los registros insertados en las tablas.

4.7. RESUMEN COMPARATIVA

Una vez aplicado el caso práctico sobre el almacén de datos documental MongoDB y sobre la base de datos relacional Oracle, se pueden sacar las siguientes conclusiones:

- Para el dominio de datos médico obtenido a partir de la herramienta PubMed, MongoDB proporciona una mayor facilidad a la hora de insertar y manipular los datos, ya que el almacenamiento del fichero MEDLINE.txt es muy similar a cómo se almacenan los documentos en formato .JSON siendo este un formato clave-valor.

- En MongoDB no ha sido necesario tener en cuenta el número de campos que contenía cada artículo. Se podían insertar documentos cuyos artículos tuvieran un número de campos variable. En Oracle esto no se puede hacer. Se ha tenido que crear una tabla con todos los campos existentes en un artículo y para aquellos artículos que no presentaran estos campos poner el valor "NULL" a estos atributos.

- En el caso de Oracle ha sido necesario realizar un diseño conceptual previo donde se ha realizado un diagrama entidad-relación para identificar y estructurar las entidades existentes. Después se ha hecho también un esquema relacional para visualizar más concretamente la estructura y organización de las futuras tablas.

- En MongoDB no han sido necesarios los diseños de estructuras previos. Se ha preparado el fichero de datos para que pudiera ser interpretado por la herramienta y se ha ejecutado.

- En cuanto a las inserciones, en MongoDB se ha hecho la inserción de un documento en formato .JSON en una colección que se ha creado. Para el caso de Oracle, en primer lugar los datos se han insertado en una tabla desnormalizada y después se han pasado estos datos a las tablas normalizadas.

Con estas conclusiones, se puede determinar que para el dominio de datos elegido, MongoDB proporciona una mayor facilidad a la hora de almacenar y tratar los datos. También proporciona escalabilidad al permitir el almacenamiento masivo de información. La elección de Oracle para este caso práctico puede beneficiar a la hora de realizar consultas, donde las bases de datos relacionales son muy eficientes al estructurar muy concretamente la información.

A continuación se muestra una tabla con las principales diferencias entre una base de datos relacional y una base de datos documental.

	Base de datos relacional	Base de datos documental
Terminología	Esquema	Namespace JSON
	Tabla	Colección
	Registro	Documento
	Campo	Campo
	Índice	Índice
	Clave primaria	Clave primaria
Almacenamiento	Documentos	Registros
	No declara schemas	Si declara schemas, diseño E/R y esquema relacional previo
	No tratamiento exhaustivo de datos	Tratamiento de datos exhaustivo, diseño de dependencias, análisis de todos los atributos
	NO JOIN	SI JOIN
	Transformación de datos a JSON	Creación de tablas y declaración de tipos de datos
	Almacenamiento masivo	Eficiente en consultas y almacenamiento de información ordenado
Consultas	SELECT	$project
	WHERE	$match
	DISTINCT	$distinct
	ORDER BY	$sort
	ROWNUM	$count
	GROUP BY	$group
	HAVING	$match
	SUM	$sum
	AVERAGE	$avg
	MIN	$min
	MAX	$max

Tabla 10. Comparativa final BBDD relacional BBDD Documental

CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1. CONCLUSIONES

Este proyecto se inició con el propósito de cumplir una serie de objetivos marcados e indicados al principio del documento, en el apartado *1.2. Objetivos.* Por recordar alguno de ellos, se pretendía exponer los sistemas tradicionales de almacenamiento y el surgimiento de nuevos sistemas con la llegada del fenómeno Big Data. En esta exposición se intentaba mostrar que los nuevos sistemas de almacenamiento surgían a partir de las nuevas necesidades que se iban creando, sin olvidar que los sistemas tradicionales seguían y siguen teniendo un amplio ámbito de uso hoy en día, tal y como se mostró en la *Figura 13. Ranking de las bases de datos más populares*.

A partir de la aplicación de un caso práctico en dos tecnologías diferentes, una relacional y otra NoSQL, se han visto las principales diferencias y problemas de representar la información y estructurar los datos en ambas tecnologías. Se ha concluido que por la manera de organizar los datos desde el dominio escogido, la base de datos NoSQL MongoDB proporciona una mayor facilidad de manipulación y un almacenamiento masivo de datos en entornos que requieren escalabilidad. También se concluye que en el tratamiento de datos MongoDB proporciona mucha más flexibilidad que en Oracle, donde ha sido necesario un estudio exhaustivo previo para el tratamiento de la información. Ahora, si se tiene en cuenta la importancia de la realización de las consultas sobre los datos, hay que indicar que las bases de datos relacionales son muy eficientes al estructurar de forma clara sus datos.

Por tanto y como se ha ido indicando a lo largo de este TFG, la elección de una tecnología u otra variará según el problema. Si se hubiera elegido otro dominio de datos con información estructurada al estilo relacional, probablemente habría sido más sencillo el tratamiento y gestión de datos con una base de datos relacional, aunque también quizá, a la hora de almacenar masivamente los datos, esta tecnología presentaría problemas de escalabilidad.

Este TFG, por ende, puede servir como guía introductoria a la hora de valorar qué parámetros son importantes cuando se pretende realizar un almacenamiento de datos. Se ha mostrado un dominio de datos concreto y se ha aplicado en dos tecnologías diferentes. Los pasos que se han seguido pueden ayudar en la elaboración de otros proyectos a la hora de determinar qué parámetros se quieren cumplir en un sistema para decidir por tanto cuál es el más eficiente.

5.2. TRABAJOS FUTUROS

Una vez terminado el proyecto y mostradas las conclusiones que se han obtenido, se pueden dejar planteadas algunas ideas que continúen con la comparativa NoSQL y SQL.

Un trabajo futuro puede ser realizar una simulación de una base de datos NoSQL (por ejemplo una base de datos columnar) funcionando en una SQL (Oracle) o viceversa.

Otra línea futura puede ser el tratamiento y la representación del dominio de datos también médico en una base de datos orientada a grafo, por ejemplo en Neo4J, en la que por ejemplo se mostraran las relaciones que hay entre distintas enfermedades dados unos síntomas. Para ello se cogerían aquellas publicaciones que tuvieran referencias de varias enfermedades y síntomas presentes y elaborar una relación entre ellas. De esta forma se podría realizar un modelo de datos que representara la relación entre varias enfermedades y que también incluyera los síntomas que la provocan. Por ejemplo sacar una relación de qué enfermedades están relacionadas con el dolor de cabeza.

De igual forma y partiendo de este dominio de datos, se podría hallar la forma de representarlos en el resto de bases de datos NoSQL, como en una base de datos columnar o una base orientada a objetos, para así ver qué ventajas e inconvenientes proporciona la representación de este dominio en todos los tipos de bases de datos vistos en este TFG.

GLOSARIO DE TÉRMINOS

SQL – Structured Query Language

NoSQL - No Structured Query Language

TFG – Trabajo Fin de Grado

ECTS – European Credit Transfer System

IDS – Integrated Data Store

BI – Business Intelligence

BBDD – Bases de datos

SGBD – Sistema Gestor de Bases de Datos

IMS – Information Management System

IDMS – Integrated Database Management System

FNBC – Forma Normal de Boyce Codd

OLAP – Online Analytical Processing

PL/SQL -Procedural Language/Structured Query Language

WWW – World Wide Web

OBS- Online Business School

JSON - JavaScript Object Notation

BSON – Binary JSON

GQL -Google Cloud Platform

CQL -Cassandra Query Language

CAP – Consistency Availavility, Persistence

BASE – Basic Availability Soft Estate

RAM – Random Access Memory

XML - eXtensible Markup Language

POO – Programación Orientada a Objetos

BLOB - Binary Large Objects

NCBI - National Center for Biotechnology Information

NLM - U.S. National Library of Medicine

CSV – Comma Separated Value

ID - Identificador

BIBLIOGRAFÍA

[1] «PagePersonnel,» [En línea]. Available: http://www.pagepersonnel.es/productsApp_pp_es/Estudios%20Remuneracion/er_tecnologia.pdf. [Último acceso: 20 Septiembre 2015].

[2] S.M.Deen, Fundamentos de los sistemas de bases de datos, Gustavo Gili, 1987.

[3] Timothy Chee, Lee-Kwun Chan, Min-Hooi Chuah, Chee-Sok Tan, Siew-Fan Wong,William Yeoh , «BUSINESS INTELLIGENCE SYSTEMS: STATE-OF-THE-ART REVIEW,» 2009. [En línea]. Available: http://spict.utar.edu.my/SPICT-09CD/contents/pdf/SPICT09_A-5_1.pdf. [Último acceso: Agosto 2015].

[4] Michael Cox and David Ellsworth, «Nasa Advanced Supercomputing Division,» 1997. [En línea]. Available: http://www.nas.nasa.gov/assets/pdf/techreports/1997/nas-97-010.pdf.

[5] «Winshuttle,» 2015. [En línea]. Available: http://www.winshuttle.es/big-data-historia-cronologica/. [Último acceso: Agosto 2015].

[6] «Winshuttle,» Agosto 2015. [En línea]. Available: http://www.winshuttle.es/big-data-historia-cronologica/ . [Último acceso: 2015].

[7] Javier Calle, *Ficheros y Bases de Datos,* 2015.

[8] Antonio Moreno Ortiz, «Estudios de Lingüística del Español,» 2000. [En línea]. Available: http://elies.rediris.es/elies9/4-2-1.htm. [Último acceso: 2015].

[9] Codd, Codd and Salley, Providing OLAP (On-line Analytical Processing) to User-analysts: an IT Mandate, 1993.

[10] E.F.Codd, The Relational Model for Database Management, Version 2, 2000.

[11] Raymond Reiter, Towards a Logical Reconstruction of Relational Database Theory. En On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages, Nueva York, 1984.

[12] C.J. Date, Introducción a los Sistemas de bases de datos, Mexico: Addison Wesley Longman, 1998.

[13] «BBVA Innovation Center,» Junio 2013. [En línea]. Available: http://www.centrodeinnovacionbbva.com/innovation-edge/big-data/big-data-en-que-punto-estamos. [Último acceso: 2015 Abril].

[14] «Gartner,» 2014. [En línea]. Available: http://www.gartner.com/it-glossary/big-data/. [Último acceso: Abril 2015].

[15] Ricardo Barranco Fragoso, «IBM developerWorks,» 2013. [En línea]. Available: https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/. [Último acceso: Abril 2015].

[16] «SAS,» Mayo 2013. [En línea]. Available: http://www.sas.com/en_us/insights/big-data/what-is-big-data.html. [Último acceso: Abril 2015].

[17] «Intel,» 2015. [En línea]. Available: http://www.intel.es/content/www/es/es/communications/internet-minute-infographic.html . [Último acceso: Abril 2015].

[18] Michael Schroeck, Rebecca Shockley, Dra. Janet Smart, Dolores Romero-Morales, «IBM Global Business Services,» 2012. [En línea]. Available: http://public.dhe.ibm.com/common/ssi/ecm/gb/es/gbe03519eses/GBE03519ESES.PDF. [Último acceso: Abril 2015].

[19] «OBS Business School,» 2015. [En línea]. Available: http://www.obs-edu.com/noticias/estudio-obs/en-2020-mas-de-30-mil-millones-de-dispositivos-estaran-conectados-internet/. [Último acceso: 10 Septiembre 2015].

[20] «KPMG,» [En línea]. Available: http://www.kpmgblogs.es/tag/big-data/. [Último acceso: 10 Septiembre 2015].

[21] «Acens,» Febrero 2014. [En línea]. Available: http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf. [Último acceso: Abril 2015].

[22] E. Bewer, «CAP Twelve Years Later: How the "rules" have changed,» 2012. [En línea]. Available: http://www.realtechsupport.org/UB/NP/Numeracy_CAP%2B12Years_2012.pdf. . [Último acceso: Agosto 2015].

[23] «genbetadev,» Enero 2014. [En línea]. Available: http://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap. [Último acceso: Agosto 2015].

[24] Lopez de Ipiña, Diego, «Bases de Datos No Relacionales (NOSQL),» 2012. [En línea]. Available: http://www.slideshare.net/dipina/nosql-cassandra-couchdb-mongodb-y-neo4j. [Último acceso: Junio 2015].

[25] «basho,» 2015. [En línea]. Available: http://basho.com/products/#riak. [Último acceso: Mayo 2015].

[26] «Aerospike,» 2015. [En línea]. Available: http://www.aerospike.com/. [Último acceso: Mayo 2015].

[27] «LevelDB,» 2015. [En línea]. Available: http://leveldb.org/. [Último acceso: Mayo 2015].

[28] «Project Voldemort,» 2015. [En línea]. Available: http://www.project-voldemort.com/voldemort/. [Último acceso: Mayo 2015].

[29] «Oracle,» 5 Junio 2015. [En línea]. Available: http://www.oracle.com/technetwork/database/database-technologies/nosqldb/overview/index.html. [Último acceso: Mayo 2015].

[30] amazon, «amazon web services,» 2015. [En línea]. Available: http://aws.amazon.com/es/simpledb/. [Último acceso: Mayo 2015].

[31] «Google Code,» 2011. [En línea]. Available: https://code.google.com/p/terrastore/. [Último acceso: Mayo 2015].

[32] I. Villegas, «SG Buzz,» Abril 2014. [En línea]. Available: http://sg.com.mx/revista/43/nosql-una-nueva-generacion-base-datos#.VcJruvntmko. [Último acceso: Mayo 2015].

[33] «Google Cloud Platform,» 2015. [En línea]. Available: https://cloud.google.com/bigtable/docs/. [Último acceso: Mayo 2015].

[34] «Apache HBase,» 2015. [En línea]. Available: http://hbase.apache.org/. [Último acceso: Mayo 2015].

[35] «Hypertable,» 2014. [En línea]. Available: http://hypertable.org/. [Último acceso: Mayo 2015].

[36] F. S. Caparrini, «Dpto. de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla,» Julio 2015. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=84 . [Último acceso: Julio 2015].

[37] «InfoGrid,» Enero 2015. [En línea]. Available: http://infogrid.org/trac/. [Último acceso: Mayo 2015].

[38] «Objectivity,» 2015. [En línea]. Available: http://www.objectivity.com/products/infinitegraph/. [Último acceso: Mayo 2015].

[39] «Sparsity technologies,» 2015. [En línea]. Available: http://sparsity-technologies.com/. [Último acceso: Junio 2015].

[40] «GemTalk Systems,» 2014. [En línea]. Available: https://gemtalksystems.com/. [Último acceso: Mayo 2015].

[41] «actian,» 2015. [En línea]. Available: http://supportservices.actian.com/versant/default.html. [Último acceso: Mayo 2015].

[42] «DB-Engines,» Agosto 2015. [En línea]. Available: http://db-engines.com/en/ranking . [Último acceso: Agosto 2015].

[43] «DB-Engines,» 2015. [En línea]. Available: http://db-engines.com/en/ranking_definition. [Último acceso: Agosto 2015].

[44] B. N. d. M. d. l. E. U. EE.UU., «MedLinePlus,» [En línea]. Available: https://www.nlm.nih.gov/medlineplus/spanish/faq/difference.html. [Último acceso: 23 Julio 2015].

[45] «Florida International University,» [En línea]. Available: http://libguides.fiu.edu/PubMed/pubmedspanish. [Último acceso: 15 Septiembre 2015].

[46] «U.S National Library of Medicine,» [En línea]. Available: https://www.nlm.nih.gov/bsd/mms/medlineelements.html#au. [Último acceso: 6 Septiembre 2015].

[47] «IBM DeveloperWorks,» [En línea]. Available: http://www.ibm.com/developerworks/ssa/data/library/Gu%C3%ADa_para_JSON_DB2_10.5/index.html. [Último acceso: 10 Septiembre 2015].

[48] «MongoDB,» [En línea]. Available: https://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/. [Último acceso: 15 Agosto 2015].

[49] Antonio Moreno Ortiz, «Estudios de Lingüística del Español,» 2000. [En línea]. Available: http://elies.rediris.es/elies9/4-2-1.htm . [Último acceso: Julio 2015].

[50] «National Center for Biotechnology Information,» [En línea]. Available: http://www.ncbi.nlm.nih.gov/pubmed. [Último acceso: 15 Septiembre 2015].

[51] «National Center for Biotechnology Information,» [En línea]. Available: http://www.ncbi.nlm.nih.gov/pubmed/?term=%22Darwin+C%22%5Bps%5D. [Último acceso: 15 Septiembre 2015].

[52] «HuffingtonPost,» [En línea]. Available: http://www.huffingtonpost.es/2014/10/14/horas-trabajadas-espana_n_5981344.html. [Último acceso: 20 Septiembre 2015].

ANEXO I. Diagrama de Gantt

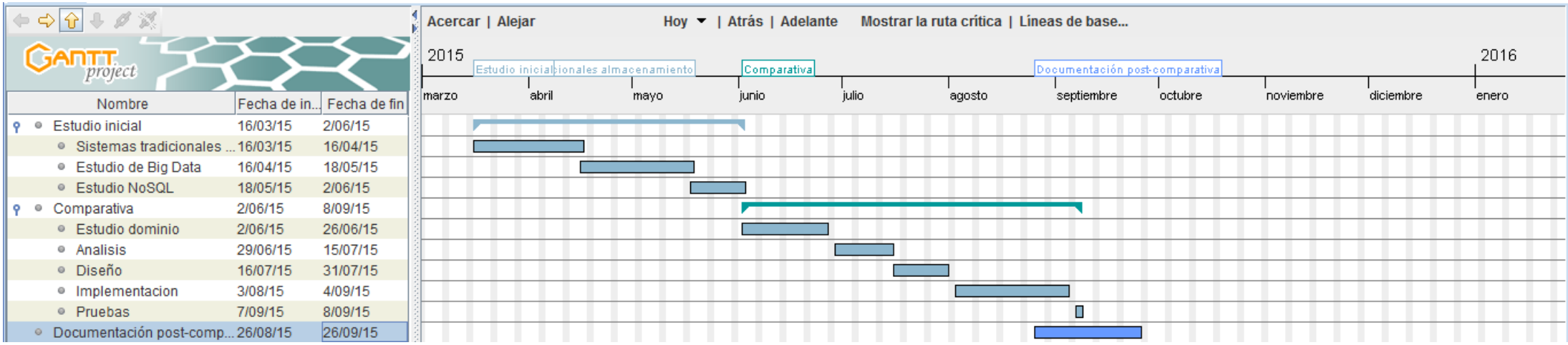

Figura 40. Diagrama de Gantt (final)

ANEXO II. Análisis de Costes

Universidad Carlos III de Madrid

PRESUPUESTO DE PROYECTO

1. Autor: Ana María Lorente Borox

2. Departamento: Informática

3. Descripción del proyecto: Trabajo Fin de Grado
   - Título: **Almacenes de datos NoSQL, Estudio de la tecnología**
   - Duración (meses) : 6

Tasas de costes indirectos 20%

4. Presupuesto total del proyecto (valores en Euros):

5. Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	NIF (no rellenar)	Categoría	Dedicación (hombres/mes)	Coste hombre / mes (euros)	Coste (Euros)	Firma de conformidad
Lorente Borox, Ana María		Administrador BBDD junior	2,17	2000	4324,324	
Iglesias Maqueda, Ana María		Administrador BBDD senior	0,12	3750	324,324	
1 Hombre -mes = 131,25 horas				Total	4648,648	

EQUIPOS

Descripción	Coste (Euros)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador acer TravelMate 5740	645,5	100	6	60	64,55
Microsoft Office Proffesional	350	100	6	60	35
Gantt Project v2.6.6	0	5	6	60	0
Notepad ++	0	50	6	60	0
Oracle Database 11g Express Editio	0	25	6	60	0
MongoDB	0	25	6	60	0
				Total	99,55

$\frac{A \times C \times D}{B}$

A- Numero de meses desde la fecha de facturación en que el equipo es utilizado

B- Periodo de depreciación (60 meses)

C- Coste del equipo (sin IVA)

D- Porcentaje de uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0

OTROS COSTES DIRECTOS DEL PROYECTO

Descripción	Empresa	Coste imputable
	Total	0

5. Resumen de costes

Prespuesto Costes Totales	**Presupuesto Costes Totales**
Personal	4648,648
Amortización	99,55
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes indirectos	949,6396
Total	**5697,8376**

Figura 41. Análisis de costes (final)

ANEXO III. Manual de Instalación MongoDB

En este anexo se va a explicar cómo se ha llevado a cabo la instalación de la herramienta MongoDB. Para ello se ha seguido el manual oficial de la página web [48].

Consideraciones previas: el equipo en el que se va a instalar la herramienta MongoDB cuenta con 4GB de RAM, 500GB de disco duro y sistema operativo Windows 7.

INSTALACIÓN

1. En primer lugar, uno de los requerimientos que se nos indica para instalar MongoDB en Windows 7 es que tenemos que instalarnos un "parche" o revisión de la versión ya que para esta versión del sistema operativo hay un problema a la hora de mapear archivos de memoria en Windows.

   Una vez se ha instalado la actualización, disponible en https://support.microsoft.com/es-es/kb/2731284 **nos descargamos la versión de MongoDB para Windows 64 bits.**

   El fichero de instalación .msi que hay que descargar se indica en la siguiente figura ("Windows 64-bit 2008 R2+"):

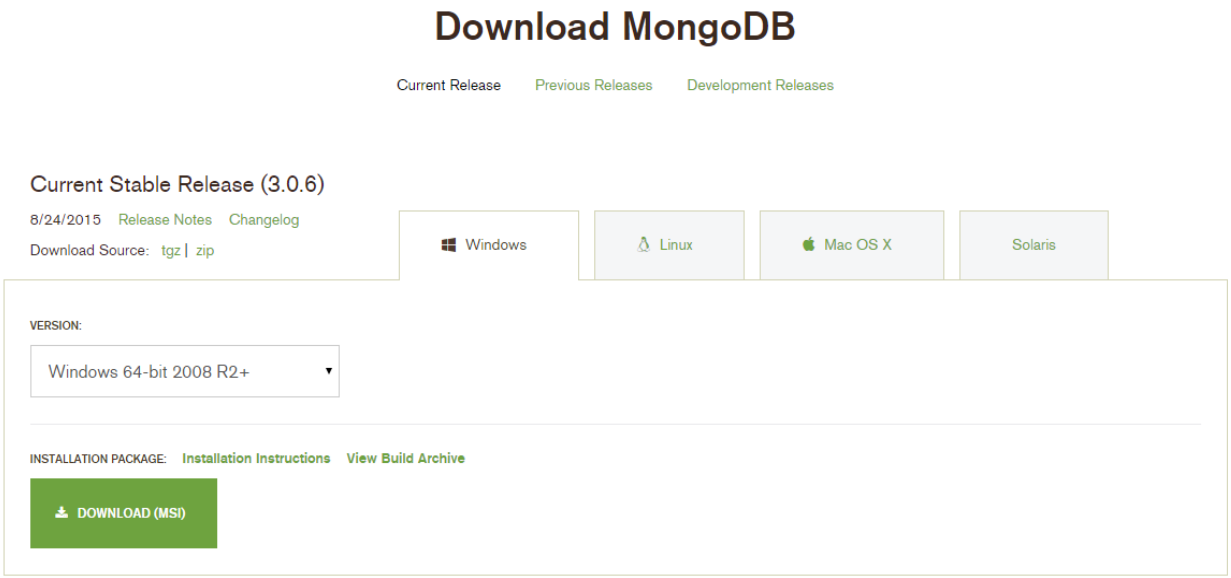

Figura 42. Fichero .msi de instalación MongoDB

1. Instalación de MongoDB para Windows. Una vez descargado el fichero .msi dar al botón "siguiente" en las pantallas de instalación. Hay que especificar el directorio de instalación si se escoge la opción de instalación "Custom". Por defecto, MongoDB se instalará en la ruta "`C:\mongodb`"

2. Abrir consola de comandos en Windows como Administrador. Para ello ir al menú de Windows, teclear cmd.exe y presionar `Ctrl + Shift + Enter` para tener la consola en modo administrador.

3. Después de esto se debe cambiar el directorio que contiene el fichero .msi que se ha descargado anteriormente. Para ello se escribirá en la consola de comandos:

```
msiexec.exe /q /i mongodb-win32-x86_64-2008plus-ssl-3.0.6-
signed.msi ^

      INSTALLLOCATION="C:\mongodb" ^

       ADDLOCAL="all"
```

En la siguiente figura se muestra este paso:

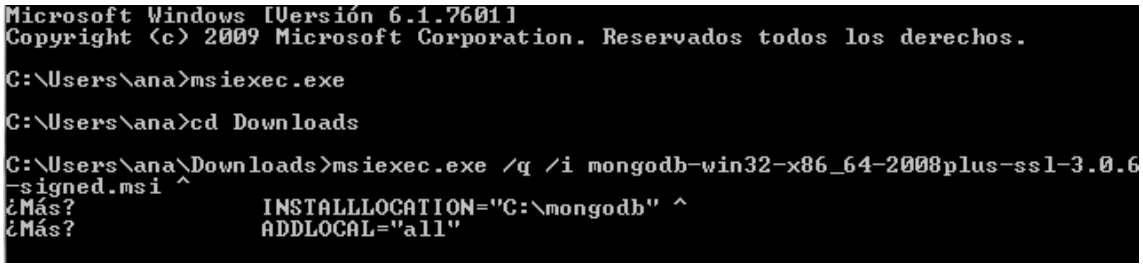

Figura 43. Instalación de MongoDB

El valor de INSTALOCATION se puede modificar si se desea que la instalación se lleve a cabo en otro directorio que no sea el de "C:"

En cuanto al argumento ADDLOCAL se ha indicado que instale todos los ficheros binarios. Si se desea instalar componentes específicos de la herramienta se deben indicar en este parámetro

A continuación se muestra la tabla con los posibles valores del parámetro ADDLOCAL para instalación de componentes [48]:

Component Set	Binaries
Server	mongod.exe
Router	mongos.exe
Client	mongo.exe
MonitoringTools	mongostat.exe, mongotop.exe
ImportExportTools	mongodump.exe, mongorestore.exe, mongoexport.exe, mongoimport.exe
MiscellaneousTools	bsondump.exe, mongofiles.exe, mongooplog.exe, mongoperf.exe

Tabla 11. Tabla con valores de parámetro ADDLOCAL

1. Configurar ruta de datos

Se debe crear un directorio de datos donde se guardarán todos los datos. Por defecto la ruta del directorio de datos es `\data\db`. Se creará esta carpeta usando el comando `md \data\db`

Si se desea se puede especificar un directorio para datos alternativo usando el comando `–dbpath` para mongod.exe

```
C:\mongodb\bin\mongod.exe --dbpath d:\test\mongodb\data
```

Hay que tener en cuenta que si el directorio incluye espacios, se pondrá entre comillas como:

```
C:\mongodb\bin\mongod.exe --dbpath "d:\test\mongo db data"
```

2. Comenzar con MongoDB

2.1. Ejecución de mongod.exe

Para empezar con MongoDB, se ejecutará mongod.exe desde la consola de comandos:

```
C:\mongodb\bin\mongod.exe
```

Esto hace comenzar la ejecución de MongoDB. Los mensajes que aparecen en la consola de “waiting for connections” indican que el proceso mongod.exe se está ejecutando satisfactoriamente.

2.2. Alertas de Seguridad

Dependiendo del nivel de seguridad del equipo, Windows puede mostrar un mensaje de alerta de seguridad alertando sobre el bloqueo de algunas funcionalidades de MongoDB. Para una correcta configuración se debe fijar una conexión de red privada.

2.3. Conexión a MongoDB

Para conectarse a MongoDB se deberá otra ventana de comandos (en la primera ventana de comandos se está ejecutando mongod.exe) y se introducirá el siguiente comando:

```
C:\mongodb\bin\mongo.exe
```

Figura 44. Ejecución de MongoDB

CONFIGURAR UN SERVICIO WINDOWS PARA MONGODB

Adicionalmente, se puede configurar un servicio de Windows para MongoDB. En este apartado se definen algunos ficheros de configuración que se pueden crear para esta herramienta.

1. Abrir una consola de comandos en modo Administrador.
2. Creación de los directorios `data\db` y `data\log`. Si se han seguido los pasos del anterior apartado, el directorio `data\db` no hará falta crearlo ya que estará creado. Si no lo está, y se quiere crear los directorios, escribir en la consola de comandos:

   ```
   mkdir c:\data\db

   mkdir c:\data\log
   ```

3. Crear un fichero de configuración

   Se creará el fichero de configuración que controla los logs y el almacenamiento en MongoDB, llamado `mongod.cfg`. Este fichero estará en la ruta `C:\mongodb\mongod.cfg` e incluirá los siguientes parámetros de configuración:

   ```
   SystemLog:

      Detination: file

      Path: c:\data\log\mongod.log

    Storage:

      dbPath: c:\data\db
   ```

   El fichero mongod.log controla los logs de acceso a la herramienta. En la variable "Storage" se controla el lugar de almacenamiento.

4. Para instalar el servicio MongoDB, finalmente:

```
"C:\mongodb\bin\mongod.exe" -config "C:\mongodb\mongod.cfg"
--install
```

ANEXO IV. Manual de uso de MongoDB

En este anexo se explicará cómo se ha utilizado MongoDB para el caso práctico que se ha expuesto en este TFG. En particular, se explicará cómo crear una base de datos, una colección y cómo se insertan documentos en las colecciones. También se incluirá alguna consulta sobre los datos, para ver su funcionamiento.

CREACIÓN DE UNA BBDD Y UNA COLECCIÓN EN MONGODB

1. **Iniciar en la consola de comandos "cmd" la herramienta**:

Figura 45. Ejecución de MongoDB

2. **Creación de la base de datos "articulos_medicos" con el comando**

```
> use articulos_medicos
```

Figura 46. Creación de base de datos en MongoDB

3. **Creación de la colección e inserción de documentos**. Una vez creada la base de datos, se va a crear la colección que incluirá los 400 documentos. La colección se llamará "coleccion_darwin".

```
db.createCollection ('coleccion_darwin');
```

Para insertar los 400 documentos en la colección se abrirá otra consola de comandos y utilizaremos el parámetro `mongoimport`, de la siguiente manera:

```
mongoimport --db articulos_medicos --collection coleccion_darwin -file "C:\data\db\JSON\articulos_darwin.json"
```

```
C:\mongodb\bin>mongoimport --db articulos_medicos --collection coleccion_darwin
--file "C:\data\db\JSON\articulos_darwin.json"
2015-09-26T22:33:24.318+0200    connected to: localhost
2015-09-26T22:33:24.563+0200    imported 400 documents

C:\mongodb\bin>
```

Figura 47. Importar archivo .JSON en MongoDB

De esta manera se habrán insertado los 400 documentos en la colección `coleccion_darwin`

CONSULTAS SOBRE LOS DOCUMENTOS

1. El comando `find ()`, llamado sobre una colección permite obtener un listado de todos los objetos contenidos en la colección. En este caso, si se ejecuta `db.coleccion_darwin.find({})` se obtendrán los 400 documentos:


```
n, Charles", "EDAT" : "11/10/2012 06:00", "MHDA" : "26/10/2012 06:00", "CRDT" :
"11/10/2012 06:00", "PST" : "ppublish", "SO" : "Notes Rec R Soc Lond. 2012 Jun 2
0;66(2):115-24." }
{ "_id" : ObjectId("56070114ed8d5fe1ee9948d7"), "PMID" : "22834067", "OWN" : "HM
D", "STAT" : "MEDLINE", "DA" : "20120726", "DCOM" : "20121002", "LR" : "20130403
", "IS" : "0008-8994 (Print) 0008-8994 (Linking)", "VI" : "54", "IP" : "2", "DP"
 : "2012", "TI" : "The 'Annie hypothesis': did the death of his daughter cause D
arwin to 'give up Christianity'?", "PG" : "105-23", "AB" : "This article examine
s one of the most widely believed episodes in the life of Charles Darwin, that t
he death of his daughter Annie in 1851 caused the end of Darwin's belief in Chri
stianity, and according to some versions, ended his attendance of church on Sund
ays. This hypothesis, it is argued, is commonly treated as a straightforward tru
e account of Darwin's life, yet there is little or no supporting evidence. Furth
ermore, we argue, there is sufficient evidence that Darwin's loss of faith occur
red before Annie's death.", "FAU" : "Van Wyhe, John Pallen, Mark J", "AU" : "Van
 Wyhe J Pallen MJ", "AD" : "National University of Singapore.", "LA" : "eng", "P
T" : "Biography Historical Article Journal Article", "PL" : "Denmark", "TA" : "C
entaurus", "JT" : "Centaurus; international magazine of the history of science a
nd medicine", "JID" : "372565", "SB" : "Q", "MH" : "*Grief History, 19th Century
 *Parent-Child Relations/ethnology/legislation & jurisprudence *Religion/history
", "PS" : "Darwin C", "FPS" : "Darwin, Charles", "EDAT" : "28/07/2012 06:00", "M
HDA" : "04/10/2012 06:00", "CRDT" : "28/07/2012 06:00", "PST" : "ppublish", "SO"
 : "Centaurus. 2012;54(2):105-23." }
Type "it" for more
>
```


Figura 48. Consulta genérica en MongoDB

2. Se puede utilizar el comando `pretty ()` después de la llamada a `find ()` para que los datos se presenten de forma indexada, ya que llamando solo a `find ()` los datos se devuelven de forma secuencial. Ver los datos de forma indexada facilita la lectura y la interpretación de la información. Si se ejecuta el comando que se indica a continuación, el resultado es el que se muestra en la *Figura 45*.

   ```
   db.coleccion_darwin.find().pretty()
   ```

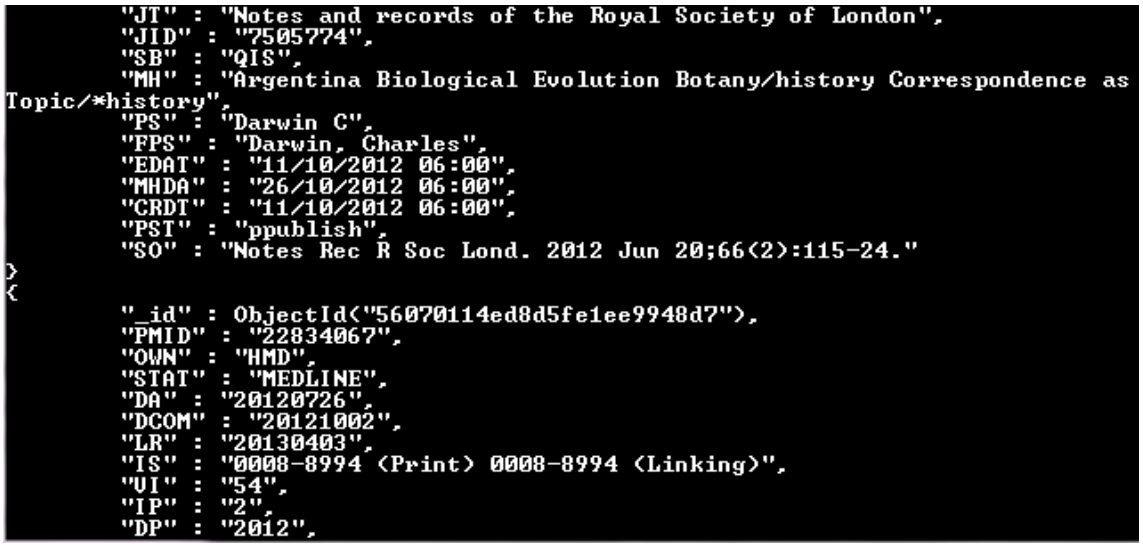

Figura 49. Consulta en MongoDB utilizando el comando pretty