UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN

"Estudio sobre herramientas NoSQL"

TRABAJO FIN DE GRADO

Diciembre - 2021

AUTOR: Jorge Soriano Gonzalez
DIRECTOR/ES: Jesús Javier Rodríguez Sala
Miriam Esteve Campello

RESUMEN

Es un proyecto académico vamos a realizar un breve repaso de la historia de las bases de datos y los sistemas de almacenamiento, para centrarnos en el estudio de las bases de datos NoSQL.

También veremos los conceptos básicos para entender las bases de datos NoSQL, donde explicaremos en profundidad las bases de datos clave-valor, las documentales, las columnares, las orientadas a grafos y veremos que son las bases de datos multimodelos.

Además, expondremos diferentes casos de uso para las bases de datos NoSQL clave-valor, documentales, columnares y orientadas a grafos, e implementaremos algunos de ellos.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1
Introducción

1.1.- EVOLUCIÓN DE LOS SISTEMAS DE ALMACENAMIENTO Y LAS BASES DE DATOS

En este epígrafe se va a hacer un repaso cronológico de los diferentes sistemas de almacenamiento y tipos de bases de datos que han existido a lo largo de la historia, desde el siglo XIX hasta la actualidad (para consultar una historia más detallada, tanto sobre los sistemas de almacenamiento, como de las bases de datos, véase el Anexo 1 de la presente memoria).

1.1.1.- Los sistemas de almacenamiento

Un dispositivo de almacenamiento es un conjunto de componentes que tienen la finalidad de servir para leer o grabar datos, en un determinado soporte de almacenamiento de datos, de forma temporal o permanente. Este apartado va a estar dedicado a la evolución de los

sistemas de almacenamiento desde las tarjetas perforadas hasta los sistemas de almacenamiento en la nube.

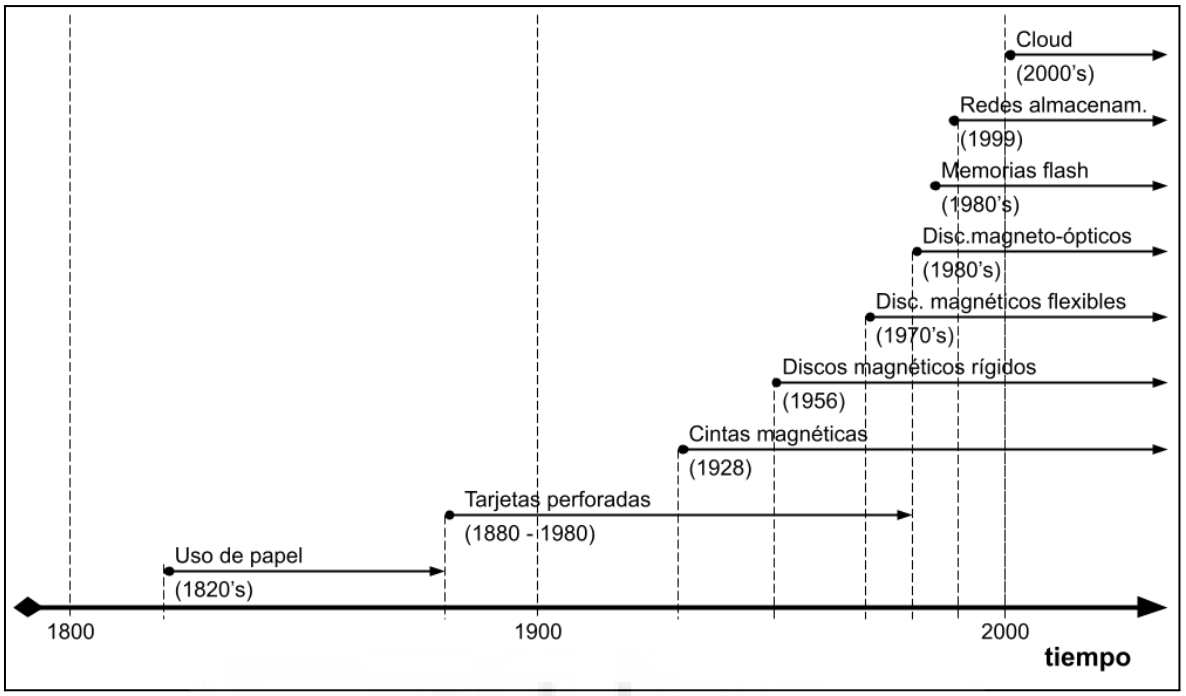


Figura 1.1.- Evolución de los sistemas de almacenamiento

La figura 1.1 muestra esquemáticamente como fueron apareciendo las diversas tecnologías utilizadas como soporte de almacenamiento. A continuación se pasa a dar una breve descripción de cada una de esas tecnologías [1]:

- Soporte Papel.- Los primeros computadores usaban papel para almacenar información. La idea de utilizar papel como medio de almacenamiento de información para computadores se remonta a Charles Babbage, un matemático inglés que inventó una calculadora mecánica denominada Motor de diferencia, diseñada para tabular funciones polinomiales en la década de 1820.

- Tarjetas perforadas.- En 1880 se inventó un mecanismo que podía detectar un agujero en una tarjeta perforada, desde ese momento se convirtieron en el medio más popular de almacenar información, tanto para la transferencia de datos como para el almacenamiento de información. Las tarjetas perforadas funcionan con código binario, esto quiere decir que donde hay un pequeño hueco significa cero (0) para la máquina y cuando no hay hueco significa uno (1). En 1950 fueron reemplazadas por la cinta magnética para el almacenamiento de información persistente, pero se siguieron utilizando hasta mediados de la década de 1980 para transferir datos y programas entre sistemas informáticos.

- Cintas magnéticas.- Las cintas magnéticas tal y como las conocemos hoy en día fueron inventadas en 1928, pero no fue hasta 1950 cuando se empezaron a utilizar como medio de almacenamiento de información auxiliar. La información es accedida de forma secuencial,y el acceso aleatorio de la información requiere el rebobinado de la cinta. La evolución del almacenamiento en cintas magnéticas ha sido notable, ya que en 1950 podía almacenar unos 1.5 Mb por cinta, y en 2014 se consiguió una capacidad de 158 Tb. En 1974 se creó la primera librería de cintas que permitía almacenar 9440 cartuchos de cinta con una capacidad de 472 Gb. Hoy en día las librerías pueden almacenar más de 100000 cartuchos con una capacidad de 2.1 Eb.

- Discos duros.- Un disco duro se compone de uno o más platos, o discos, recubiertos de un material magnético dentro de una caja sellada, sobre cada plato se sitúa un cabezal de lectura y escritura que flota sobre una delgada lámina de aire que creada por la rotación de los discos. Permiten el acceso aleatorio a los datos en un tiempo bajo. El primer disco magnético fue desarrollado en 1956 por IBM, el IBM 350 Disk File tenía una capacidad de 3.75 Mb y media 152 cm de ancho, 172 cm de alto, un grosor de 74cm, tenía discos de 24 pulgadas, pesaba casi una tonelada y alquilarlo costaba 3200$ al mes (equivaldrían a 31135$ en 2021). Durante las décadas siguientes fueron reduciendo su tamaño y coste, y aumentando su capacidad. Actualmente los discos magnéticos están formados por varios discos de 3.5 pulgadas y tiene unas medidas típicas de 101 mm de ancho, 25.4 mm de alto, 146 mm de largo y un peso inferior a 1 kg.

- Discos flexibles.- Los discos flexibles, también conocidos como Floppy Disk Drives (FDD) o disquetes, estaban compuestos por un disco delgado y flexible de plástico recubierto de un material magnético en una caja de plástico, para leerlos era necesario una unidad lectora. Los primeros disquetes salieron al mercado en 1971, eran de 8 pulgadas y solo permitían la lectura, y unos años después se introdujeron los discos que permitían la lectura y escritura. En 1976 salieron los primeros disquetes de 5.25 pulgadas, y en 1980 los discos de 3.5 pulgadas que al principio podían almacenar 438 Kb, después pasaron a 720K (doble densidad) y 1.44Mb (alta densidad).

- Dispositivos magneto-ópticos.- Los discos ópticos y los discos magneto-ópticos son medios de almacenamiento que pueden registrar información cambiando las formas fotofisicas en sus superficies de grabación y leer la información registrada emitiendo rayos de luz contra la superficie y detectando su reflejo. Su origen se remonta al método de grabación de vídeo inventado por David Paul Gregg, pero su uso práctico comenzó con el primer LaserDisc, anunciado en 1980. Los principales productos son CD y sus subsiguientes medios DVD y Blu-ray Disc. Los discos ópticos y magneto-ópticos se utilizan comúnmente para distribuir contenidos

porque son fáciles de usar y pueden fabricarse a bajo costo. El CD ha sido el medio estándar para el contenido de audio y el DVD y el Blu-ray han sido el estándar para el contenido visual.

- Memorias flash.- El uso de la memoria flash, o SCM (storage-class memory), como medio de almacenamiento no se limita a los sistemas recientes, sin embargo, el uso de memoria de clase de almacenamiento solo se hizo popular después de que la memoria flash, inventada por el ingeniero de Toshiba Fujio Masuoka en la década de 1980. Los primeros productos de memoria flash se abrieron camino en la electrónica de consumo, como las tarjetas SD y Compact Flash. En muchos casos la memoria flash se utiliza en unidades de estado sólido (SSD), donde se empaquetan chips de memoria flash y un circuito de control. En los últimos años han salido al mercado nuevos productos de memoria flash como SSD PCI Express o SSD NVMe, proporcionan una latencia de acceso mucho menor.

- Almacenamiento en la nube.-Los servicios de almacenamiento remoto que solían llamarse SSP (Supply-Side Platform) a menudo se brindan como parte de servicios en la nube completos. Actualmente, los principales servicios de almacenamiento basados en la nube incluyen Amazon S3, Windows Azure Storage y Google Cloud Storage, todos diseñados en estrecha coordinación con sus otros servicios en la nube. El almacenamiento basado en la nube no se limita a los sistemas empresariales y se está volviendo más popular para los nuevos tipos de productos electrónicos de consumo, como los reproductores audiovisuales digitales.

1.1.2.- Las bases de datos

Una base de datos se podría definir como una colección de registros integrados, entendiendo por registro una representación de algún objeto físico o conceptual [3]. Las primeras bases de datos no utilizaban lo que hoy denominamos TICs (Tecnologías de la Información y la Comunicación), es decir, no contaban con ordenadores ni nada parecido, sino que eran bibliotecas, ficheros y archivos en papel. A partir de la década de 1960, con la aparición de los primeros soportes para almacenamiento y acceso aleatorio de datos apareció por primera vez el concepto de “Base de Datos”, el cual ha ido evolucionando hasta nuestros días. A continuación se describen algunos conceptos y términos relacionados con la evolución de las bases de datos:

- Acceso aleatorio a los datos.- En 1956 con la salida al mercado del IBM 305 RAMAC (Random Access Method of Accounting and Control) aparecen los primeros discos duros magnéticos, que permitían un acceso aleatorio a los datos en tiempos bajos.

- Bases de datos.- En la década de los 60’s s apareció el término “base de datos”, que representó un contraste con los sistemas basados en cintas del pasado, permitiendo un uso interactivo compartido en lugar del procesamiento por lotes diario. Se desarrollaron dos modelos de datos principales: el modelo de red CODASYL y el modelo jerárquico IMS.

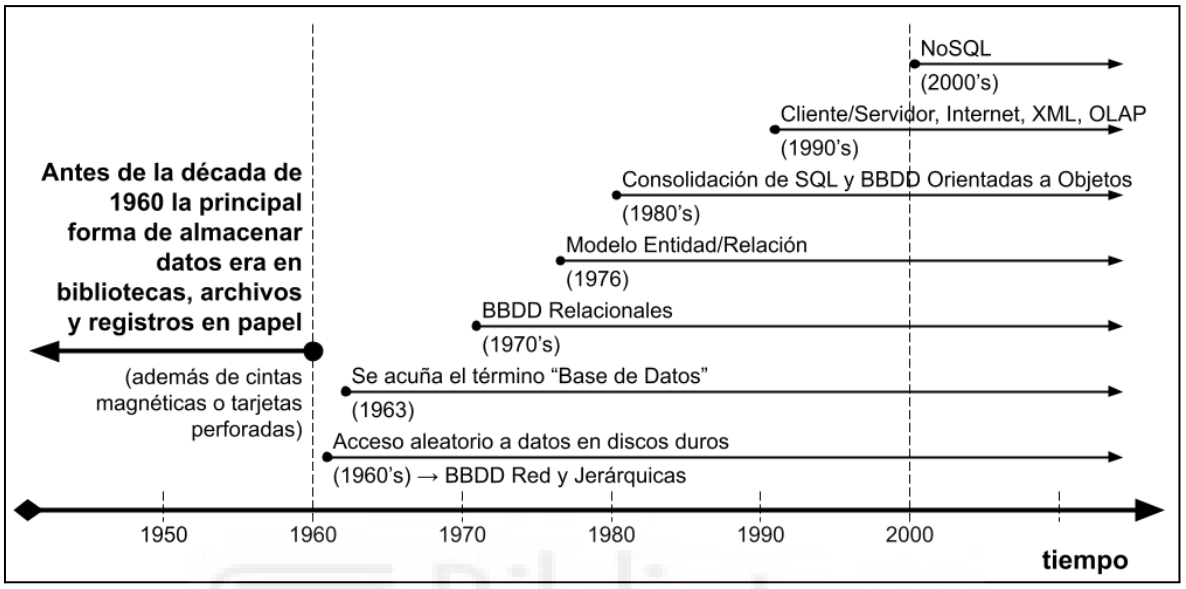


Figura 1.2.- Evolución de las bases de datos

- BBDD relacionales.-Al inicio de la década de los 70’s el modelo de base de datos relacional fue concebido por E.F. Codd. Este modelo se apartó de la tradición al insistir en que las aplicaciones deberían buscar datos por contenido en lugar de seguir enlaces.[6] Durante el período 1974 a 1977, se crearon dos prototipos principales de sistemas de bases de datos relacionales: INGRES y SystemR [8].

- Modelo Entidad/Relación.- En 1976 un nuevo modelo de base de datos llamado Entidad-Relación, o ER, fue propuesto por P. Chen. Este modelo hizo posible que los diseñadores pudieran centrarse en la aplicación de datos en lugar de la estructura lógica de la tabla [6].

- Consolidación SQL y BD Orientadas a Objetos.- En la década de los 80 la comercialización de sistemas relacionales comienza cuando un auge en la compra de computadores impulsa el mercado de bases de datos (DB) para los negocios. SQL (Structured Query Language) se convirtió en el estándar [7]. El término 'sistema de base de datos orientado a objetos' apareció por primera vez alrededor de 1985. Una base de datos de objetos, también sistema de gestión de bases de datos orientado a objetos, es aquella en la que la información se representa en forma de objetos tal como se utiliza en la programación orientada a objetos [9].

- Cliente/Servidor, Internet, XML, OLAP.- En la década de los 90 el modelo cliente-servidor para la informática se convirtió en la norma para futuras decisiones comerciales. Durante la década una gran inversión en empresas de Internet impulsó el auge del mercado de herramientas para conectores Web, Internet y DB, también se popularizaron para su uso comercial el procesamiento de transacciones en línea (OLTP) y el procesamiento analítico en línea (OLAP). A finales de la década se introdujo el Lenguaje de marcado extensible (XML).

- NoSQL.- El término NoSQL se utilizó por primera vez en 1998 por Carlo Strozzi para denominar a su base de datos relacional liviana y de código abierto que no disponía de una interfaz SQL estándar[13]. Actualmente el término NoSQL etiqueta ciertos tipos de bases de datos de reciente aparición, que presentaban cualidades más características no relacionales, y que no siempre proporcionaban garantías ACID (atomicidad, consistencia, aislamiento, durabilidad), que son los atributos clave de los sistemas de bases de datos relacionales clásicos.

1.2.- JUSTIFICACIÓN DEL PROYECTO

Con la llegada de las redes sociales y la necesidad de gestionar grandes volúmenes de datos de manera distribuida y escalable, ha aparecido un nuevo paradigma de bases de datos denominado NoSQL que, en realidad, se podría decir que es más de uno, ya que no hace referencia a un nuevo tipo de base de datos sino a diversos tipos que se salen del modelo clásico de base de datos relacional que, hasta no hace mucho, era prácticamente el único estándar considerado en cuanto al modo de diseñar y gestionar bases de datos.

Con respecto a NoSQL, ha día de hoy no existe un estándar claro, son muchos los productos que hay a disposición de los desarrolladores pero, en general, si bien hay algunos que destacan sobre otros, no hay aún una herramienta o grupo de herramientas que puedan considerarse definitivas o “estándares de facto”, por lo que resultaría de interés disponer de una documentación donde se estudien estas herramientas y se muestre el funcionamiento de algunas de ellas para poder dar una opinión formada acerca de sus características, ventajas e inconvenientes.

1.3.- OBJETIVOS

El presente proyecto tiene una finalidad puramente académica. No se busca desarrollar una aplicación o crear un producto final funcional con una utilidad práctica, sino que se pretende abordar el estudio de un tema, las bases de datos NoSQL, del cual, a día de hoy,

no hay una estandarización que permita afirmar con rotundidad que productos o herramientas son los más idóneos.

Los objetivos que se pretenden conseguir con este proyecto son:

1. Presentar las tecnologías de bases de datos, su evolución hasta la actualidad y realizar una comparativa entre bases de datos relacionales y no relacionales.

2. Estudiar y explicar las tecnologías de bases de datos NoSQL, sus diferentes tipos y casos de uso según sus características.

3. Realizar una comparativa de los diferentes tipos de gestores de bases de datos NoSQL y documentar el funcionamiento de cada una de las herramientas estudiadas.

4. Practicar y adquirir experiencia con estas bases de datos.

1.4.- LÍMITES DEL PROYECTO

El paradigma NoSQL es tremendamente amplio, tanto que hacer un estudio exhaustivo sería tan arduo que excedería por mucho el tiempo razonable que debería ocupar un trabajo fin de grado (véase la referencia [11] con un listado de más de 200 bases de datos NoSQL). Este proyecto pretende sentar una primera aproximación al paradigma NoSQL para disponer de una visión general sobre el mismo, y en un futuro, si se considera oportuno, se puedan realizar otros trabajos que amplíen y profundicen en ciertos aspectos de este paradigma.

Capítulo 2 Antecedentes y estado de la cuestión

NoSQL es una clase de sistemas de administración de bases de datos no relacionales con un esquema flexible. Se utilizan con datos distribuidos, especialmente con aplicaciones que involucran big data y datos en tiempo real, como registros web.

2.1.- CONCEPTOS BÁSICOS DE LA BASES DE DATOS NoSQL

A continuación veremos algunas ideas básicas para poder comparar los modelos de bases de datos no relacionales y entender sus diferencias con las bases de datos relacionales.

- Persistencia: es la capacidad de almacenar los datos en un estado permanente, no significa que los datos no se puedan eliminar sino que los datos no se pueden o no se deben perder.

- Replicación: es la capacidad de almacenar varias réplicas de las bases de datos en varios nodos para garantizar la disponibilidad de los datos en todo momento, esto significa que si algún nodo no se encuentra disponible siempre habrá otro de respaldo con una copia de la base de datos. La presencia de estas réplicas con la capacidad de modificar o leer datos sin editar los demás pueden generar un problema de inconsistencia.

- Fragmentación: es la capacidad de la base de datos para dividir los datos en varios nodos permitiendo un análisis y procesamiento rápidos. Esta característica no está disponible en todas las bases de datos, porque en ese caso los usuarios tendrían que encargarse de la partición de los datos manualmente o esperar una cantidad considerable de tiempo hasta que el sistema procese todos los datos.

- Consistencia: las bases de datos NoSQL permiten el acceso múltiple para escribir y leer los datos al mismo tiempo, esta capacidad de tener los mismos datos entre todas las réplicas y para todos los usuarios se consigue con la consistencia eventual (ver BASE más adelante), lo que significa que una vez que se introducen o modifican datos, eventualmente aparecerán para su lectura.

- Métodos de consulta: Cada tipo de base de datos emplea una técnica diferente para la consulta de datos, algunos ejemplos serían MapReduce, get o consultas basadas en Java, entre otros.

- Teorema del CAP: Estas iniciales significan consistencia (Consistency), disponibilidad (Availability) y tolerancia a la partición (Partition tolerance), tres criterios a considerar al implementar una solución de base de datos no relacional. Se ha demostrado que ninguna arquitectura de sistema distribuido puede cumplir con más de dos de estos criterios [12].

    - Consistencia: esta propiedad implica que las actualizaciones de datos deben transmitirse a través de los nodos de datos (fragmentos), de modo que todos los usuarios puedan recuperar y consultar los datos modificados.

    - Disponibilidad: la necesidad de al menos un nodo con una copia de seguridad disponible en caso de fallo.

    - Tolerancia de partición: el sistema permanece activo incluso si hay un fallo en la red o se corta la conexión entre los nodos.

  A partir de estas tres características, se definen tres grupos de bases de datos que surgen de combinar dichas características de dos en dos:

- Bases de datos CP (Consistent & Partition tolerance): ofrecen consistencia y tolerancia a partición a expensas de la disponibilidad. Cuando se produce una partición entre dos nodos, el sistema tiene que cerrar el nodo que no esté disponible hasta que se resuelva la partición.

- Bases de datos AP (Availability & Partition tolerance): mantienen tanto la disponibilidad como la tolerancia a la partición, pero no la consistencia. Cuando se produce una partición, todos los nodos permanecen disponibles, pero los que se encuentran al otro lado de la partición pueden tener una versión de datos más antigua que otras. Cuando se resuelve la partición se resincronizan los nodos para reparar las inconsistencias en el sistema.

- Bases de datos CA (Consistent & Availability): proporcionan consistencia y disponibilidad en todos los nodos. Sin embargo, no puede hacer esto si hay una partición entre dos nodos en el sistema y, por lo tanto, no puede ofrecer tolerancia a fallos.

A continuación se van a describir los conceptos "ACID" y "BASE", el primero representa una serie de características que tienen las bases de datos relacionales tradicionales, el segundo describe ciertas propiedades que el paradigma NoSQL ofrece, en contraposición al anterior:

- ACID: es un acrónimo para "Atomicity, Consistency, Isolation, Durability" que traducido es Atomicidad, Consistencia, Aislamiento y Durabilidad, y son un conjunto de propiedades de las transacciones de bases de datos destinadas a garantizar la validez de los datos a pesar de errores, cortes de energía y otros contratiempos:

  - Atomicity: La transacción se ejecuta completa, o no se ejecuta.
  - Consistency: La base de datos pasa de un estado válido a otro estado válido.
  - Isolation: Las transacciones son independientes, y no se afectan entre sí.
  - Durability: Los cambios perduran en el tiempo, son persistentes.

- BASE: es otro acrónimo (algo forzado) para "Basically Available, Soft state, Eventually consistent" que traducido significa es básicamente disponible, estado suave y eventualmente consistente:

  - Basically Available: las operaciones de lectura y escritura siempre están disponibles, pero como la consistencia no está garantizada los datos que se escriben o leen pueden no estar actualizados.

- Soft state: sin poder garantizar la consistencia de los datos, solo tenemos cierta probabilidad de saber el estado de la base de datos, ya que podría no haberse actualizado la información.

- Eventually consistent: pasado un tiempo todos los nodos de la base de datos estarán actualizados y la base de datos, eventualmente, será consistente.

2.2.- SITUACIÓN ACTUAL DEL PARADIGMA NoSQL

Las bases de datos NoSQL admiten diferentes tipos de datos, como los estructurados, los semiestructurados y los no estructurados. Están diseñadas para cubrir algunos de los inconvenientes del sistema de administración de bases de datos relacionales, específicamente cuando se trata de big data de alta variabilidad, donde los datos pueden ser de múltiples tipos y de fuentes diferentes.

NoSQL se puede describir como una solución de bases de datos que permite la escalabilidad horizontalmente, y que además, es adecuada para aplicaciones con bases de datos jerárquicas, que almacena los datos enlazando los registros con una estructura de árbol, donde un nodo padre puede tener varios nodos hijos y así sucesivamente.

Las bases de datos NoSQL se usan principalmente en aplicaciones donde no son necesarias las transacciones de tipo ACID, con un esquema flexible, y sin restricciones o validaciones que deban ser ejecutadas por el motor de la base de datos, agilizando así todas las operaciones, con datos temporales, con un alto volumen de datos y con datos de diferente naturaleza (registros, documentos, etc...) [12].

2.3.- BASES DE DATOS CLAVE-VALOR

Una base de datos clave-valor es una simple tabla hash, es de utilidad principalmente cuando todos los accesos a los datos almacenados se realizan a través de una clave primaria.

Las bases de datos clave-valor son las bases de datos NoSQL más sencillas de usar desde el punto de vista de la API. El cliente puede obtener un valor de una clave, asignar un valor para una clave o eliminar una clave de la base de datos. El valor es almacenado en un "*blob*" (Binary Large OBject) que almacena los datos sin importarle ni saber que hay dentro, es responsabilidad de la aplicación comprender los que hay almacenado. Dado que

las bases de datos clave-valor utilizan un acceso de clave principal generalmente tienen un gran rendimiento y pueden ser escaladas fácilmente [13].

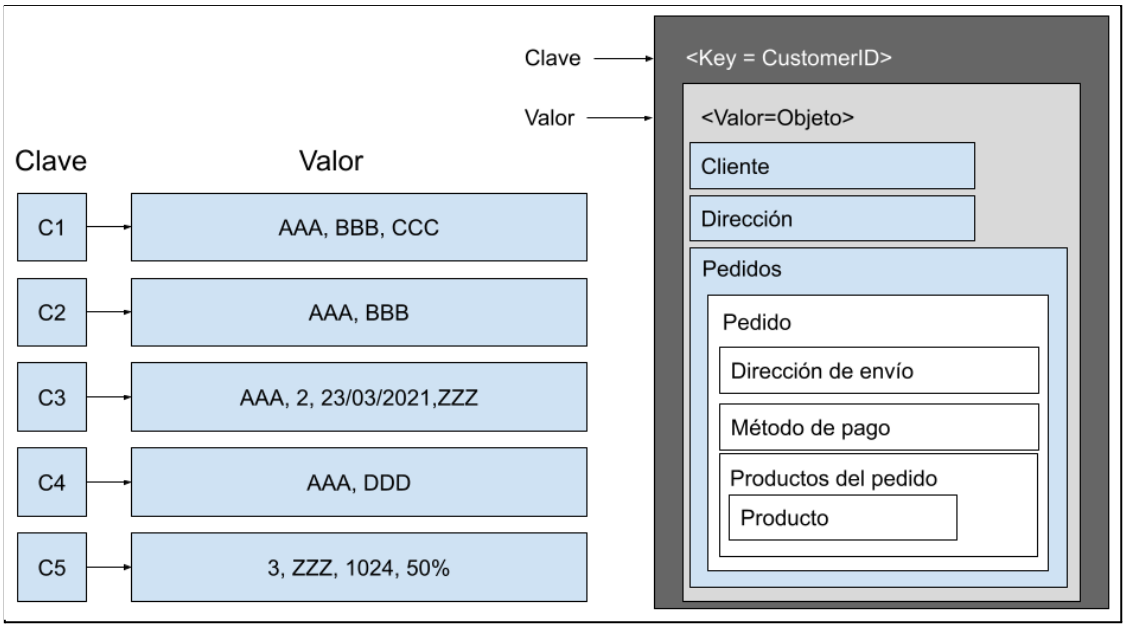


Figura 2.1.- Base de datos clave-valor

A continuación se enumeran una serie de bases de datos clave-valor que hay disponibles en el mercado comentando algunas de sus características:

- Riak: Una base de datos NoSQL clave-valor que ofrece alta disponibilidad, tolerancia a fallos, escalabilidad y simplicidad operacional. Ofrece tanto una versión de código abierto como una versión comercial.

- Redis: Una base de datos NoSQL clave-valor de código abierto que soporta una variedad de estructuras de datos, tales como listas, mapas, juegos, conjuntos ordenados, y cadenas de texto.

- Voldemort: Una base de datos NoSQL clave-valor que ofrece una alta escalabilidad.

- Infinispan: Una base de datos NoSQL clave-valor desarrollada por Red Hat. Es compatible con la transacción, MapReduce. Infinispan es capaz de asegurar la persistencia de datos para el sistema de ficheros a través de arquitectura conectable [12].

En la tabla 2.1 se muestra una comparativa de estas cinco bases de datos, donde se indica qué características (de las descritas en apartados anteriores) cumplen cada una de ellas.

Tabla 2.1: Tabla comparativa de bases de datos clave-valor

	RIAK	REDIS	VOLDEMORT	INFINISPAN
Persistencia	Bitcask, LevelDB, en memoria y multi-backend	Memoria volátil, Sistema de archivos.	BerkeleyDB, MySQL, en memoria	Sistema de archivos, BerkeleyDB, JDBM, JDBC
Replicación	Ring (N-1)	Maestro-Esclavo	Ring (N-1)	Ring (N-1)
Fragmentación	Hash consistente	No soportado	Hash consistente	Hash consistente
Consistencia	Se puede elegir entre eventual o fuerte	Eventual	Eventual	Se puede elegir entre eventual o fuerte
Métodos de consulta	RESTful API, Apache Solr, MapReduce	Get	Get	Get, MapReduce
Lenguaje de implementación	Erlang	C	Java	Java
CAP	AP	AP	AP	AP / AC / CP

2.4.- BASES DE DATOS COLUMNARES

Las bases de datos columnares permiten almacenar los datos con claves asignadas a valores, y los valores agrupados en varias familia de columnas, es decir, que almacenan los datos por columnas en lugar de por filas y cada fila está asociada con uno o más atributos (columnas).

La utilidad de estas bases de datos se pone de manifiesto en aquellas aplicaciones en las que las familias de columnas contienen grupos de datos que están fuertemente relacionados y que es necesario acceder a ellos a la vez de manera habitual, bien para su consulta, o bien para realizar con ellos algún tipo de cálculo o análisis, ya que estas bases de datos optimizan el acceso a los datos por columnas (en vez de por filas como es habitual en la mayoría de sistemas de bases de datos) [13].

Algunos ejemplos de bases de datos columnares que es posible encontrara en el mercado serían los siguientes:

- HBase: Es una base de datos columnar de código abierto, desarrollada por la fundación Apache y creada sobre la arquitectura HDFS. Se caracteriza por tener baja latencia y permitir el acceso aleatorio. Es compatible con el procesamiento de datos en tiempo real y está diseñada para ser escalada linealmente y para su uso con grandes volúmenes de datos. Proporciona características avanzadas tales como la compresión y el procesamiento de la memoria.

- Hypertable: Una base de datos columnar de código abierto creada por Google y modelada por el "Big Table" de Google, que es escalable y compatible con múltiples sistemas de almacenamiento distribuidos. Proporciona un alto rendimiento y escalabilidad.

- Cassandra: Una base de datos columnar de código abierto fundada inicialmente por Facebook, pero en la actualidad desarrollada y mantenida por la fundación Apache. Proporciona una base de datos escalable capaz de manejar una gran cantidad de datos de consistencia ajustable, haciendo una base de datos eventualmente consistente.

- Accumulo: Es otro proyecto desarrollado y mantenido por la fundación Apache, se trata de una base de datos columnar de código abierto del tipo "Big Table", que se puede ejecutar sobre "Hadoop" y "Apache Zookeeper", y proporciona características adicionales como la compresión, y la gestión de acceso a nivel de campo [12].

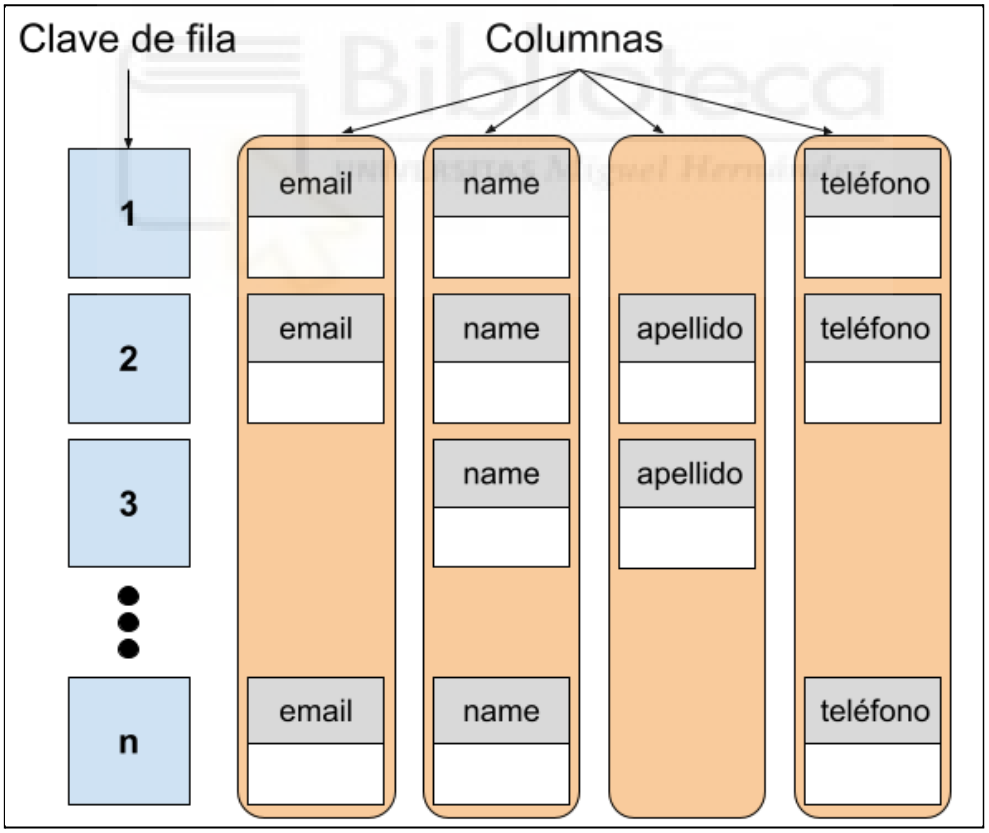


Figura 2.2 .- Representación familia de columnas

En la tabla 2.2 se muestra una comparativa de estas cinco bases de datos, donde se indica qué características (de las descritas en apartados anteriores) cumplen cada una de ellas.

Tabla 2.2: Tabla comparativa de bases de datos columnares

	HBASE	HYPERTABLE	CASSANDRA	ACCUMULO
Persistencia	Construido sobre HDFS	HDFS (Hadoop Distributed File System)	Sistema propio	HDFS (Hadoop Distributed File System)
Replicación	Se maneja en HDFS	HDFS (Hadoop Distributed File System)	Partición automática sin pérdidas	Replicación multi maestro por filas
Fragmentación	Por clave-valor, fragmentación automática	Clave-valor, permite la división	Por hash	Por fila
Consistencia	Alta consistencia	Alta consistencia	Consistencia eventual	Consistencia eventual
Métodos de consulta	MapReduce	MapReduce, HQL (Hipertable query language)	MapReduce	MapReduce
Lenguaje de implementación	Java	C++	Java	Java
CAP	CP	CP	AP	CP

2.5.- BASES DE DATOS DOCUMENTALES

Las bases de datos documentales almacenan y recuperan documentos, los cuales pueden ser XML, JSON, BSON, etc. Estos documentos están organizados en estructuras de árbol jerárquicas autodescriptivas que pueden consistir en mapas, colecciones o valores escalares. Los documentos almacenados normalmente serán similares entre sí, pero no es necesario que sean exactamente iguales.

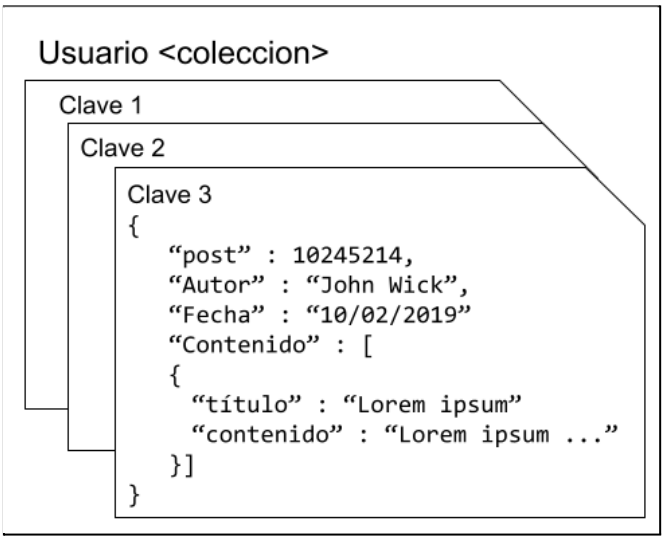


Figura 2.3 .- Representación colección de documentos

Las bases de datos documentales se pueden ver como bases de datos clave-valor donde almacenamos al documento donde iría el valor, y donde el valor almacenado, ahora sí, puede ser explorado [13]. Algunos ejemplos de bases de datos documentales serían:

- CouchDB: Una base de datos NoSQL documental de código abierto que puede transferir, almacenar y procesar los datos a través de múltiples formatos.

- MongoDB: Es una base de datos NoSQL documental multiplataforma, que utiliza documentos tipo JSON como esquema. Cuenta con consultas ad hoc, indexación, replicación, balance de carga, agregación y transacciones.

- RavenDB: Una base de datos NoSQL documental totalmente transaccional. También es una base de datos multimodelo que ofrece una variedad de modelos de datos, tales como: documental , grafos, clave-valor, y de serie de tiempo.

- Terrastore: Una base de datos NoSQL documental que garantiza la consistencia por documento [12].

La tabla 2.3 comparan las características de las bases de datos mencionadas anteriormente:

Tabla 2.3: Tabla comparativa de bases de datos documentales

	COUCHDB	MONGODB	RAVENDB	TERRASTORE
Persistencia	Arbol-B	Objetos Bson GRIDFS (para grandes archivos)	Motor de almacenamiento extensible	Almacenamiento de Terrastore
Replicación	Maestro-Maestro	Maestro-Esclavo	Maestro-Esclavo	Maestro-Esclavo
Fragmentación	Disponible con extensiones	Por campo	Definida por el usuario	Hash consistente
Consistencia	Eventual	Estricta o eventual	Eventual	Eventual
Métodos de consulta	MapReduce	MapReduce, por campo y cursores	MapReduce	Consultas condicionales
Lenguaje de implementación	Erlang	C++	C#, Java, Python, Node.js	Java
CAP	AP	AP / CP	AP	AP

2.6.- BASES DE DATOS ORIENTADAS A GRAFOS

Las bases de datos orientadas a grafos permiten almacenar entidades y relaciones entre estas entidades. Las entidades también se conocen como nodos, y las relaciones se

denominan aristas. Las relaciones, o aristas, tienen significado direccional, es decir que solo se puede recorrer en una dirección. Las entidades, o nodos, están organizados por relaciones, lo que permite encontrar patrones interesantes entre los nodos. La organización de los grafos permite que los datos se almacenen una vez pero puedan ser interpretados de maneras diferentes en función de sus relaciones [13].

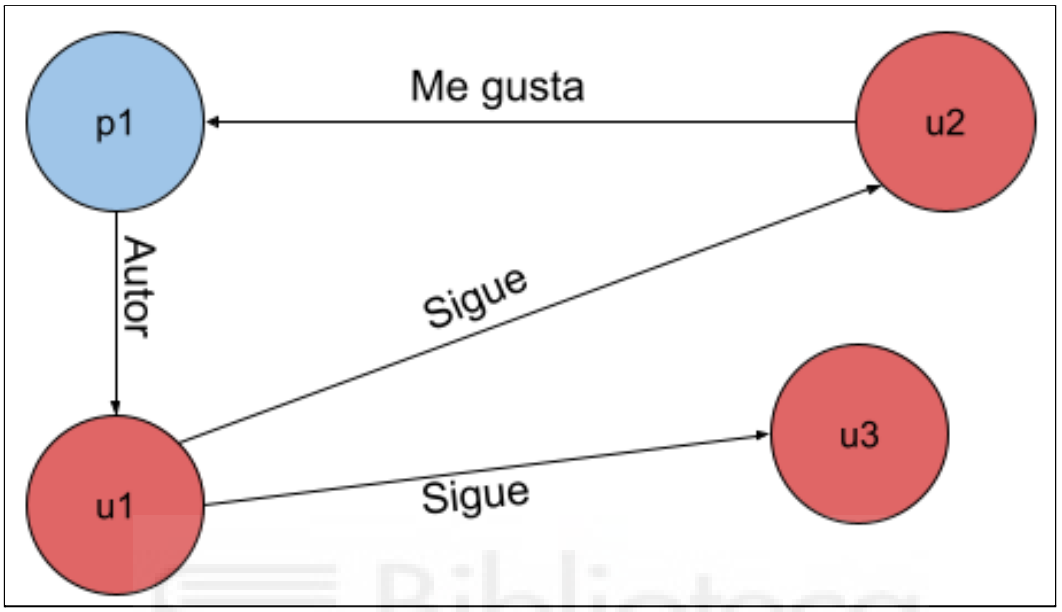


Figura 2.4 .- Representación de bases de datos orientadas a grafos

Bases de datos orientadas a grafo serían las siguientes:

- Neo4J: Una base de datos NoSQL orientada a grafos compatible con transacciones ACID.

- InfiniteGraph: Una base de datos NoSQL orientada a grafos que permite concurrencia, consistencia, procesamiento multi-hilo, y tiene soporte en la nube.

- InfoGrid: Una base de datos NoSQL orientada a grafos basada en web con componentes que permiten el desarrollo de aplicaciones web REST-ful.

- AllegroGraph: Una base de datos NoSQL orientada a grafos de código cerrado, que se utiliza principalmente para la recuperación y gestión de la información orientada a documentos.[12]

En la tabla 2.4 se vuelve a mostrar una comparativa de las cuatro bases de datos mencionadas, indicando qué características cumplen cada una de ellas y a qué clase de base de datos pertenecen según el teorema CAP.

Tabla 2.4: Tabla comparativa de bases de datos documentales

	NEO4J	INFINITEGRAPH	INFOGRID	ALLEGROGRAPH
Persistencia	Apache Lucene	Objectivity BD	Índices y caché de relación	Índices
Replicación	Maestro-Esclavo	P2P (Peer to peer)	P2P	Maestro-Esclavo
Fragmentación	Manual	Basado en reglas	Manual	Manual
Consistencia	Eventual	"Tunable consistency"	Eventual	Eventual
Métodos de consulta	API java	API de grafos transversal	SPARKQL, RDFS++	SPARQL, API de grafos transversal
Lenguaje de implementación	Java	C++	Java, Plantillas HTML y Viewlets	LISP
CAP	AP	AP / CP	AP	AP

2.7.- BASES DE DATOS MULTIMODELOS

La mitad de la década de 2000 estuvo marcada por el entusiasmo de una parte de la industria que esperaba reemplazar por completo o al menos reducir drásticamente el dominio de las bases de datos relacionales y de su lenguaje universal SQL. Este entusiasmo se vio amplificado por la decepción anterior causada por las bases de datos orientadas a objetos a mediados de la década de 1990, que no lograron convertirse en una alternativa a las bases de datos relacionales, como se esperaba [14]. Durante estos años, se han desarrollado una multitud de sistemas dedicados para un cierto tipo de modelo de datos NoSQL, tales como: Memcached para bases de datos clave-valor (2003), CouchDB bases de datos de documentos (2005), Neo4J para bases de datos de grafos (2007) o Cassandra para bases de datos columnares (2008). Pero muy pronto la industria se dio cuenta de que esta limitación a un solo modelo de datos restringe en gran medida su alcance porque las aplicaciones complejas de hoy requieren rendimiento, escalabilidad y flexibilidad a gran escala, y que no se pueden lograr mediante el uso de un solo modelo de datos, por lo que la tendencia ha sido convertir cada uno de estos modelos en un sistema unitario.

Es cierto que, desde la llegada de estos modelos de datos NoSQL, ha habido preocupaciones sobre el desarrollo de bases de datos que integren más modelos de este tipo. Al analizar la evolución de los sistemas de gestión de bases de datos, se pueden delinear tres operaciones sobre la convergencia de modelos de datos en bases de datos únicas:

1. Primera tendencia: los sistemas de bases de datos relacionales establecidos han hecho esfuerzos para integrar modelos de datos NoSQL.

2. Segunda tendencia: las bases de datos NoSQL, inicialmente construidas sobre un solo modelo, integraron otros modelos NoSQL y/o el modelo relacional.

3. Tercera tendencia: representada por el desarrollo de sistemas que se construyen desde el principio para integrar múltiples modelos NoSQL y/o el modelo relacional [14].

2.8.- BASES DE DATOS ELEGIDAS PARA EL ESTUDIO

Elegir una base de datos NoSQL entre las muchas que hay en el mercado no es una tarea sencilla, normalmente a la hora de elegir una base de datos para un proyecto tendríamos en cuenta criterios como el modelo de datos que vamos a implementar, métodos de consulta, las API's disponibles, lenguaje de programación, el tipo de licencia del software y qué tipo de consistencia, disponibilidad y tolerancia a fallos queremos para la base de datos. Para este estudio elegiremos una base de datos para los modelos más comunes, que son el clave-valor, columnaria, documental, y orientada a grafos, y el principal criterio que tendré en cuenta es que tengan licencia de código abierto y que se pueda instalar localmente. Con esto elegimos Redis como base de datos clave-valor, Cassandra como base de datos columnar, MongoDB como base de datos documental y Neo4j como base de datos de grafos. En los siguientes capítulos se describe con más detalle su funcionamiento y se mostrará algún caso de uso práctico con cada una de ellas.

Capítulo 3 Hipótesis de trabajo

Las bases de datos NoSQL han experimentado un importante crecimiento en su aplicación en los últimos años. Su gran flexibilidad, escalabilidad, estructura distribuida y las posibilidades que brindan desde el punto de vista de la optimización a la hora de tratar problemas con grandes cantidades de datos las convierten en una atractiva variante a tener en cuenta. En este capítulo se van a mostrar algunos de los casos de uso más comunes para las bases de datos NoSQL de tipo clave-valor, columnares, documentales y orientadas a grafos.

3.1.- CASOS DE USO DE LAS BASES DE DATOS CLAVE-VALOR

Como se ha visto en el capítulo anterior las bases de datos clave-valor son muy simples, ofrecen un alto rendimiento, y se utilizan cuando se necesita una alta velocidad con un gran

volumen de datos. Se suelen usar en aplicaciones de ingesta rápida de datos, mensajería, inventario en tiempo real, detección de fraudes, almacenamiento de caché, gestión de sesiones, etc. A continuación veremos algunos casos de cómo se utilizan las bases de datos clave-valor.

3.1.1.- Gestión de sesiones de usuarios

Los datos de sesión son aquellos creados de la interacción del usuario con aplicaciones como una página web o un videojuego. Cuando se inicia una sesión se produce un acceso a la base de datos para recuperar datos de sesión guardados previamente. Una aplicación web típica mantiene una sesión para cada usuario mientras esté conectado. La lectura y escritura de los datos de sesión se realizan en segundo plano en caché. Mientras la sesión esté activa no deberían ser necesarios accesos a la base de datos. El último paso del ciclo de vida de los datos de sesión ocurre cuando el usuario se desconecta, donde algunos datos se guardan para su uso futuro y otros se descartan cuando finaliza la sesión.

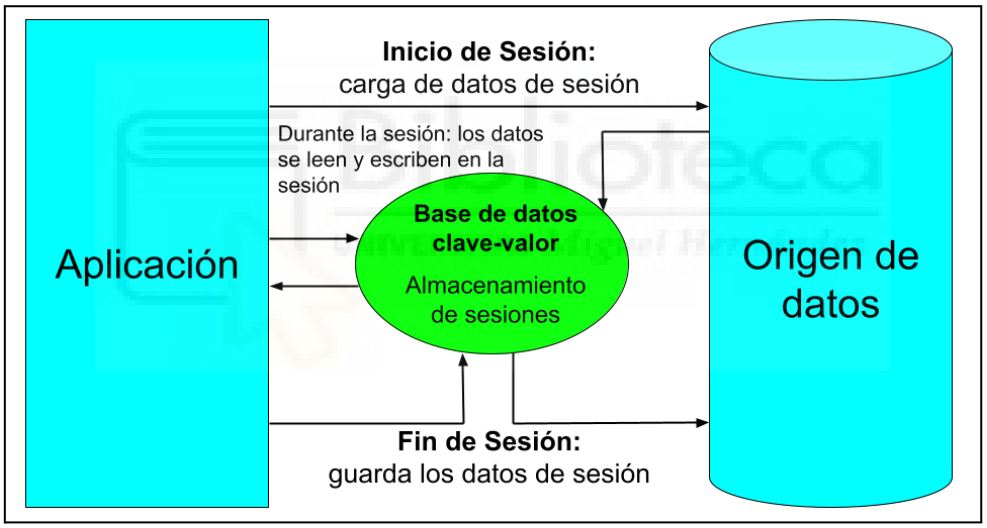


Figura 3.1.- Representación gestión de sesiones

Los datos de sesión pueden ser volátiles o permanentes, lo que significa que los datos pueden descartarse o conservarse en el almacenamiento en disco cuando finaliza la sesión del usuario. Un ejemplo de datos de sesión volátiles podría ser el historial de navegación de páginas en una intranet corporativa, hay poca necesidad de conservarlo. Por el contrario, un carrito de la compra duradero en una página web de comercio electrónico es fundamental para el negocio y debe guardarse en la base de datos.

Los datos se sesión se guardan en una base de datos clave-valor, donde la clave es el identificador y el valor los datos. Esto asegura que los datos de sesión no accedan a datos de otras.[15]

3.1.2.- Anuncios en páginas web

Si bien los anuncios son fundamentales para las empresas que comercializan sus productos o servicios en la web, no son esenciales para la experiencia de navegación web de muchos usuarios. Sin embargo, el tiempo de carga de las páginas web es importante para ellos, y tan pronto como un anuncio con un tiempo de carga lento comienza a aumentar el tiempo de carga de una página, los usuarios comienzan a moverse a sitios web alternativos y más rápidos. Por lo tanto, ofrecer anuncios rápidamente es una preocupación clave. Sin embargo, hacerlo no es sencillo, ya que el anuncio que se muestra al usuario depende de una gran cantidad de factores, como la actividad rastreada del usuario en línea, el idioma y la ubicación.

Uno de los principales casos de uso de las bases de datos clave-valor es la publicidad web. Las empresas de publicidad web utilizan un software propietario donde usan una combinación de factores para determinar lo que un usuario quiere o en lo que está interesado para poder dirigir anuncios a ese usuario. Para esto en el valor estaría almacenado el anuncio dirigido al cliente, y la clave estaría compuesta por una combinación de factores, como: el país de origen, el idioma, una categoría de compras en la que esté interesado, etc., que apunten al anuncio más atractivo para el cliente, un ejemplo de esto podría ser el uso de la clave "*ES-spanish-ssd*" para una persona de españa, hispanohablante, que está buscando componentes para ordenador.[16]

3.1.3.- Caché en memoria

Otro de los usos más comunes de las bases de datos clave-valor es el caching, o caché en memoria. Al implementar una caché en memoria de alta disponibilidad se consigue reducir la latencia de acceso a los datos, incrementar la capacidad de procesamiento y aliviar la carga de la aplicación y de la base de datos relacional o NoSQL.

Esto permite que la base de datos pueda suministrar elementos solicitados con frecuencia en tiempos de respuesta inferiores a un milisegundo, y permite escalar el sistema con facilidad en caso de cargas de trabajo mayores. Algunos de los ejemplos más comunes del almacenamiento en caché que se pueden realizar con las bases de datos clave-valor son:

- **Almacenamiento de datos de caché de la base de datos**, que se utiliza a menudo para almacenar copias de tablas de búsqueda y las respuestas a consultas computacionalmente costosas a la base de datos, para mejorar el rendimiento de la aplicación y reducir la carga en la fuente de datos.

- **Almacenamiento de datos de la sesión de usuario**, debido a que cada interacción del usuario requiere acceso a los datos de la sesión, mantener esos datos en la caché acelera el tiempo de respuesta al usuario de la aplicación.

- **Almacenamiento de la respuesta de la API en caché**, las aplicaciones modernas se crean utilizando componentes poco acoplados que se comunican a través de API, guardar las respuestas en caché, aunque sea brevemente, mejora el rendimiento de la aplicación al evitar esta comunicación entre procesos.[17]

3.1.4.- Personalización de servicios

El concepto de personalización del servicio de usuario es similar a la gestión de sesiones de usuario, pero de mayor duración. Aquí es donde los usuarios configuran su parte front-end de la aplicación para sus necesidades específicas.

Un ejemplo de esto sería una base de datos principal que muestra los niveles de trabajo de un equipo, los archivos de casos actuales en los que están trabajando y todos los datos relacionados. Estos serían los datos principales de la aplicación que podrían estar almacenados en una base de datos relacional o en una base de datos NoSQL.

El uso de los datos puede variar, ya que un usuario puede querer ver un resumen de su carga de trabajo, mientras que un gerente puede querer hacer un seguimiento del trabajo de los empleados de un equipo. Estos usuarios recibirán diferentes vistas personalizadas de los mismos datos.

Estas preferencias pueden guardarse en una base de datos clave-valor, con una clave compuesta que contiene el identificador del usuario y el nombre del servicio, y las opciones de configuración como valor. Esto evita la sobrecarga de la base de datos con los datos de personalización, y aumenta la velocidad de las búsquedas.[16]

3.2.- CASOS DE USO DE LAS BASES DE DATOS COLUMNARES

Como ya se ha visto, las bases de datos columnares son muy escalables y ofrecen un alto rendimiento reduciendo los accesos al disco duro. Se suelen usar cuando se deben realizar operaciones de evaluación de grandes volúmenes de datos, Big Data, ya que el acceso al disco duro suele ser un cuello de botella en la lectura de cualquier base de datos. A continuación se muestran algunos ejemplos de cómo se utilizan estas bases de datos.

3.2.1.- Manejo de datos dispersos

A veces, un sistema de administración de bases de datos relacionales puede tener un esquema en el que las columnas tienen muchos valores nulos. Los valores nulos en las bases de datos relacionales suelen consumir un par de bytes. Este par de bytes pueden parecer poco, pero en algunas situaciones esos dos bytes suponen una cantidad significativa de espacio desperdiciado. Consideremos el ejemplo de una aplicación de contactos que admita nombres de usuario, números de teléfono (fijo y móvil) y redes sociales como Twitter o Facebook, con una columna para cada una de las centenares de opciones y direcciones. Esto significa que se desperdician cientos de bytes por registro (como se ilustra en la tabla 3.1.). Si la aplicación contempla 100 campos y tiene solo 2 campos no nulos, se estarán almacenando 98 campos nulos. Escalando esto a millones de registros (millones de usuarios), tendremos terabytes de espacio desperdiciado.

Tabla 3.1: datos dispersos en una base de datos relacional

ID	Email1	Email2	Movil	Teléfono	Twitter
1234	a@umh.es	NULL	NULL	966586455	NULL
5428	NULL	NULL	NULL	NULL	@alguien
2353	d@umh.es	j@epse.es	NULL	NULL	NULL
9724	NULL	NULL	NULL	NULL	NULL

El uso de una base de datos columnar para administrar datos dispersos en lugar de una base de datos relacional resuelve este problema de almacenamiento. En una base de datos columnar, se puede modelar la misma aplicación de contactos con una familia de columnas para cada medio de comunicación (teléfono, redes sociales, correo electrónico, dirección) y una columna para cada dato definido (teléfono fijo y móvil). Las bases de datos columnares almacenan solo las columnas que se indican en cada instancia de registro. Si se indican dos columnas (correo electrónico y teléfono), solo se almacenarán los valores de tres columnas, sin nulos ni espacio desperdiciado, como se ilustra en la tabla 3.2.[16]

Tabla 3.2: datos dispersos en una base de datos columnar

Clave de fila	Valores de columna	
1234	email:1=a@umh.es	telefono:fijo=966586455
5428	social:twitter=@alguien	
2353	email:1=d@umh.es	email:2=j@epse.es
9724		

3.2.2.- Analizar logs (archivos de registro)

Los archivos de registro son muy comunes en una variedad de sistemas y aplicaciones. Poder grabar estos archivos para su posterior análisis es una característica valiosa. Las bases de datos columnares pueden ser útiles a la hora de almacenar y analizar logs. Un ejemplo es el almacenamiento y análisis de los logs generados por servidores que se analizan casi en tiempo real para detectar problemas antes de que falle el equipo y evitar que se interrumpa el servicio que ofrece.

El enfoque tradicional de base de datos relacional para analizar datos es almacenarlos durante el día y luego migrarlos a intervalos regulares (normalmente durante la noche), para crear una estructura diferente de esos datos en un centro de procesamiento de datos para analizarlos al día siguiente. Pero hay casos en los que no tiene sentido tener una vista que esté desactualizada 24 horas o más.

Las bases de datos columnares son excelentes para recopilar esta información tan crítica y urgente de analizar. Pueden tener columnas flexibles, por lo que si una entrada de registro en particular no tiene datos para un campo, simplemente no se almacena (en la tabla 3.3 se puede ver un ejemplo de una entrada de registro en una base de datos relacional y otro en una columnar).

Tabla 3.3: logs en bases de datos

Logs de una base de datos relacional

Host	Proceso	Timestamp	Tipo	Funcion	Respuesta	Duracion
servidor 1	java	2021-05-21T09:04:32	Java RPC	getcreditscore	NULL	12.5

Logs de una base de datos columnar

Clave de fila	Valores de columna				
34BD7E44	host:name= servidor 1	host:proceso= java	evento:time= 2021-05-21T09:04:32	java :funcion= getcreditscore	evento:duracion = 12.5

Para crear informes de los mismos datos en intervalos de cinco minutos, una hora o un día, se pueden almacenar los mismos datos en estructuras alternativas al mismo tiempo. Esta es una aplicación del patrón de desnormalización que se usa comúnmente en las bases de datos NoSQL, como se muestra en la tabla 3.4.

El enfoque de desnormalización permite una búsqueda rápida de datos en diferentes sistemas y en diferentes intervalos de tiempo, lo cual es útil para consultas ad-hoc y también para construir resúmenes de datos. [16]

Tabla 3.4: tablas de logs desnormalizados

Log de 5 minutos

Clave de fila	Familia de columna
2021-05-21T09:05:00	Evento-34BD7E44:duracion 12.5

Log de 1 hora

Clave de fila	Familia de columnas	Familia de columnas
2021-05-21T09:05:00	Evento-34BD7E44:duracion 12.5	Evento-78B45E44:duracion 81.7

3.3.- CASOS DE USO DE LAS BASES DE DATOS DOCUMENTALES

Como hemos visto en el capítulo anterior el punto fuerte de las bases de datos documentales es su flexibilidad, ya que la estructura de los documentos no tienen por qué ser coherentes, le permiten almacenar datos no estructurados muy grandes en una sola base de datos. Se suelen usar en aplicaciones móviles, videojuegos, gestión de contenidos, internet de las cosas (IoT),etc. A continuación veremos algunos casos de cómo se utilizan las bases de datos documentales.

3.3.1.- Blockchain

Blockchain es esencialmente una base de datos de registros distribuida o un libro de contabilidad público de todas las transacciones o eventos digitales que se han ejecutado y compartido entre las partes participantes. Cada transacción se verifica por consenso de la mayoría de los participantes en el sistema. Una vez ingresada, la información nunca se puede borrar. Blockchain contiene un registro seguro y verificable de cada transacción realizada. Bitcoin es el ejemplo más popular y está intrínsecamente ligado a la tecnología blockchain pero la tecnología blockchain se puede aplicar tanto a aplicaciones financieras como no financieras.[18]

Como ejemplo de uso vamos a ver su aplicación en un sistema de gestión de identidad, usando la estructura y los conceptos de blockchain para guardar y publicar identidades digitales para una red bancaria ficticia. Los nodos de la red son administrados por los distintos socios de la red, impulsando y gobernando la producción de bloques dentro de la cadena.

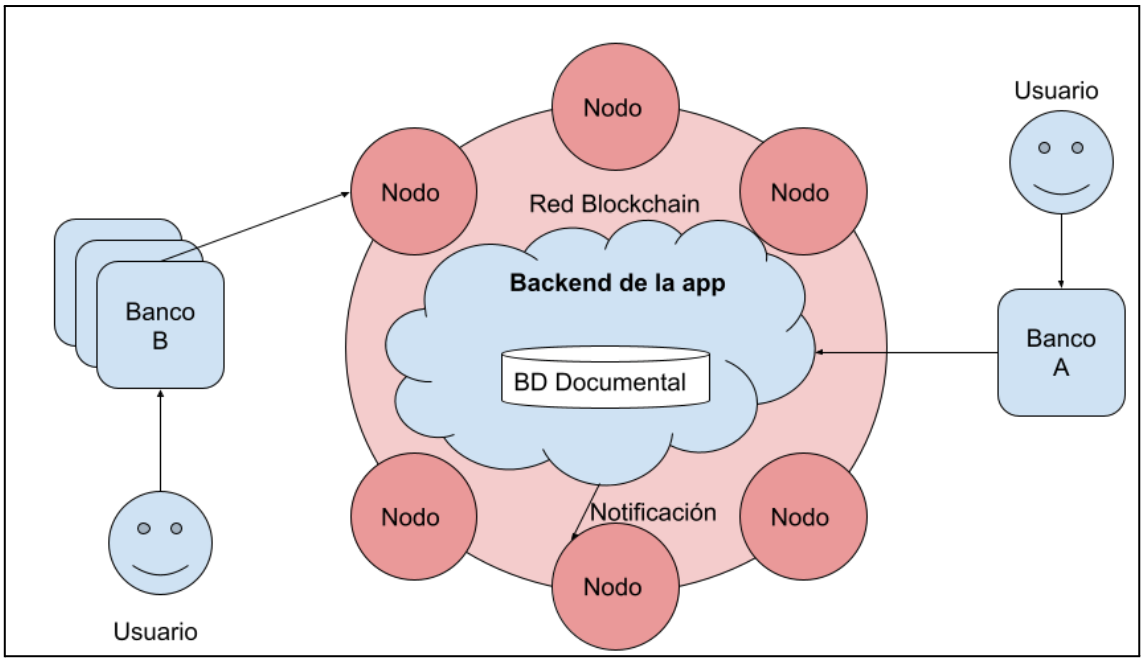


Figura 3.2 La información de identidad se publica en la cadena de bloques.

La idea principal es que los datos pueden ser enviados a la red blockchain por el Banco A (figura 3.2), consumidos y aprobados por la mayoría de la red, y aparecer como información confiable para el Banco B o C (figura 3.3). Si la información está encriptada o no depende de la Entidad A. Con esta información, el Banco B (o C) puede impulsar ofertas y promociones para esos clientes basadas en la información de la red blockchain sin la necesidad de una autoridad de registro centralizada. [19]

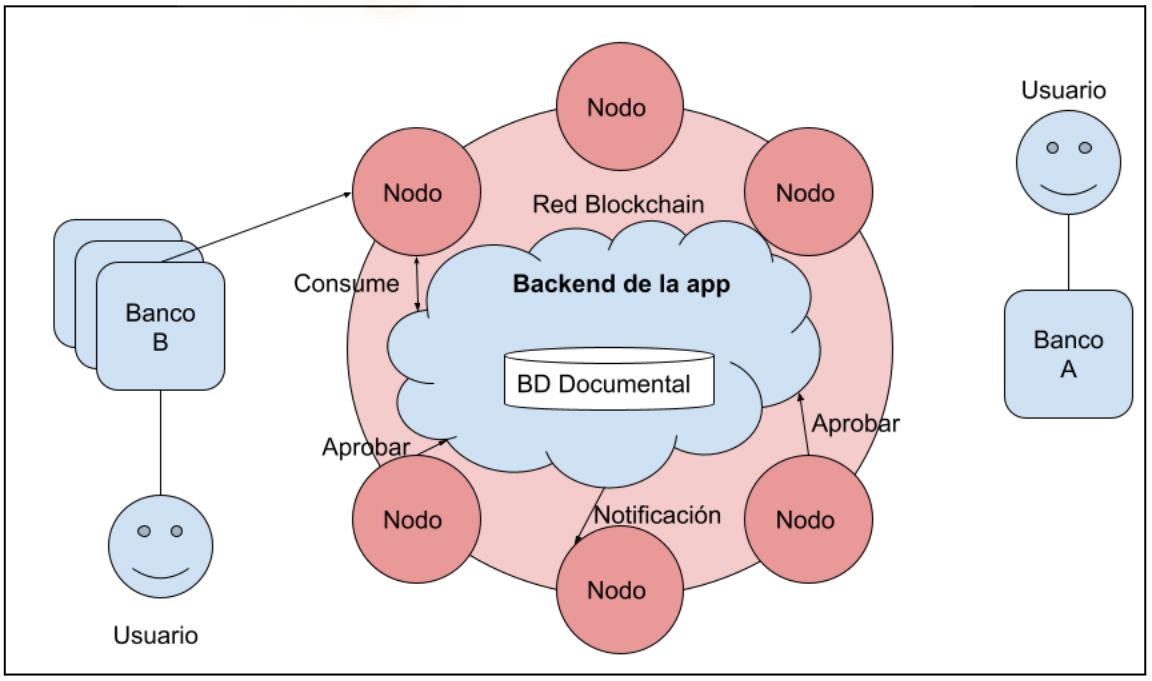


Figura 3.3 Los nodos de la red blockchain aprueban la información.

3.3.2.- Análisis en tiempo real

El procesamiento en tiempo real implica la entrada, el procesamiento y la salida continua de datos. Algunos de los usos más frecuentes son cajeros automáticos de bancos, servicios al cliente, sistemas de radar y sistemas de punto de venta (POS). Cada transacción se refleja directamente en el archivo maestro, con este proceso de datos, para que siempre esté actualizado. Se pueden introducir datos en herramientas de análisis, mediante la creación de flujos de datos, tan pronto como se generan, obteniendo resultados analíticos casi instantáneos mediante el uso de plataformas como Spark Streaming.

Los casos de uso del análisis en tiempo real son muchos, como en el análisis de datos de vehículos para transporte, logística y gestión inteligente del tráfico que ayuda a manejar la congestión del tráfico en una de las principales carreteras de Tel Aviv. Controlando los ingresos del peaje constantemente, donde para evitar la congestión durante las horas pico aumentan los precios del peaje. Este es un factor de disuasión para los usuarios, una vez que la congestión disminuye durante las horas no pico, se reducen las tarifas de peaje.

El análisis en tiempo real también está muy extendido en el sector financiero, donde el volumen de datos es enorme y cambia rápidamente; el impacto del análisis y sus resultados se reduce al aspecto monetario. Este sector necesita instrumentos en tiempo real para un análisis de datos rápido y preciso para los datos de las bolsas de valores, varias instituciones financieras, precios y fluctuaciones del mercado, etc. [20]

3.3.3.- Internet Of Things (IoT) (Internet de las cosas)

El volumen masivo y la naturaleza cada vez más desestructurada de los datos de IoT han impuesto nuevas demandas a la industria. Los modelos de datos RDBMS rígidamente definidos tienen un uso limitado en IoT. Carecen de la flexibilidad, la escala y los análisis en tiempo real necesarios para capturar, compartir, procesar y analizar rápidamente los datos de IoT. Las bases de datos documentales con su modelo flexible, que permite la posibilidad de incorporar modelos nuevos de análisis de datos, estructura rápida, que permite asegurar que las escrituras de datos siempre serán rápidas sin importar la existencia de un fallo de hardware o de red, y alto rendimiento, que permite a las búsquedas recuperar documentos rápidamente.

Un ejemplo de las aplicaciones del IoT es el sector automotriz, con una aplicación IoT es posible capturar datos del vehículo, como el sistema de frenos, niveles de los fluidos del coche, velocidad, sensores, consumo, etc. Estos datos se pueden usar para mejorar los

diagnósticos para realizar mantenimiento preventivo, así como para analizar el rendimiento de los componentes para mejorar los procesos de diseño de productos o de mantenimiento.

Otro ejemplo de aplicación de la tecnología IoT sería en la aeronáutica, donde se puede controlar, con cierto tipo de herramientas, que disponen de sensores y conectividad wireless, como de apretados están los 6 millones de tornillos en un avión. La aplicación IoT capturaría todos los datos transmitidos, desde nivel de batería hasta el par de torsión con que están apretados los tornillos, y usando el análisis en tiempo real podría saber si un tornillo esta muy apretado o algo suelto y avisar al operario en el acto para corregirlo. [21]

3.3.4.- Desarrollo de videojuegos

Los datos siempre han sido una parte esencial de los videojuegos. Desde los perfiles de jugadores a la telemetría, el emparejamiento y las tablas de puntuaciones, los datos son cruciales para que los juegos funcionen y mejoren.

A continuación se enumeran algunos ejemplos de como las bases de datos documentales se aplican en el sector de los videojuegos son:

- Perfiles de jugador extensibles, permite agregar y asociar nuevas funciones al perfil del jugador, como logros, desbloqueos basados en la progresión, moneda del juego, nuevas clases de equipo y más.

- Emparejamiento de jugadores, o matchmaking, las bases de datos documentales permite realizar análisis en tiempo real de las métricas de los jugadores para ayudarlos a asegurarse de que se están reuniendo las personas adecuadas en una partida o sesión.

- Acciones e inventario, para comprobar si un jugador puede realizar una acción, necesita lecturas de baja latencia a escala. Los retrasos en los niveles de carga o la apertura de menús hacen que los jugadores no estén contentos. Cuando se trata de inventario, necesita lecturas y escrituras rápidas con integridad transaccional.

- Captura y análisis de telemetría, capturar eventos de juegos y ejecutar análisis en tiempo real para optimizar las experiencias de los jugadores sobre la marcha con el marco de agregación integrado de muchas bases de datos documentales, como MongoDB.

- Servicios auxiliares, como mensajes e interacciones en tiempo real, incluidas las comunicaciones entre juegos. [22]

3.4.- CASOS DE USO DE LAS BASES DE DATOS DE GRAFOS

Como se ha visto en el capítulo anterior, las bases de datos orientadas a grafos permiten representar datos interconectados, así como las relaciones entre estos, y facilitan su análisis y evaluación. Algunos de sus casos de uso son las recomendaciones personalizadas, ya sea en compras o en redes sociales, la detección de fraudes, el análisis de inventarios, recomendaciones en tiempo real, análisis de causa-raíz, gestión de contenido, grafos de conocimiento, análisis de impacto, etc. A continuación veremos algunos casos de cómo se utilizan las bases de datos de grafos.

3.4.1.- Grafos sociales

Los grafos sociales son importantes en varios casos, incluidos los siguientes:

- Redes sociales: incluye los “me gusta” de Facebook, Twitter y LinkedIn.
- Organizaciones profesionales: Identificar individuos o grupos en grandes organizaciones con la experiencia requerida para un trabajo. También se puede utilizar en organigramas.
- Detección del crimen organizado: la policía y las organizaciones de seguridad realizan investigaciones multianuales de millones de dólares de las redes del crimen organizado con muchos puntos de contacto y relaciones.
- Árbol genealógico: incluye muchas relaciones de padres, abuelos, tíos y otras relaciones. Cuando se trabaja con árboles genealógicos hay que tener en cuenta que no siempre se puede asegurar que todos los datos históricos son precisos.

La gestión de esta información tiene sus desafíos. Se pueden adoptar diferentes enfoques, dependiendo de si se busca describir las relaciones entre personas o si se quiere describir las personas y sus actividades.

Uno de los modelos es el FOAF, “Friend of a Friend”, o amigo de un amigo. El modelo FOAF RDF brinda a las personas la capacidad de describirse a sí mismas, sus intereses, sus datos de contacto, así como una lista de personas que conocen. Este modelo incluye enlaces de datos abiertos vinculados a las descripciones FOAF de sus contactos cuando se conocen. Estos datos pueden descubrirse y extraerse para producir esferas de influencia en torno a áreas temáticas particulares.

Al describir árboles genealógicos, se suele utilizar el formato GEDCOM, “GEnealogical Data COMmunication”, o comunicación de datos genealógicos. GEDCOM usa un modelo de enlaces de datos. Este modelo está basado en el núcleo familiar y el individuo. Contrasta con modelos de evidencia, en el que la información está estructurada reflejando la evidencia descubierta o de soporte. En el modelo de enlaces del GEDCOM, toda la información está estructurada para presentar la realidad asumida, es decir, los núcleos familiares o individuos reales (o hipotetizados).

Otro aspecto importante a tener en cuenta en estos grafos son las búsquedas que pueden ser útiles en muchos casos, como por ejemplo:

- Sugerir un producto a los usuarios basado en artículos similares comprados por otras personas.

- Sugerir nuevos amigos para agregar nuevos amigos en función de que sean amigos de sus amigos existentes.

- Sugerir grupos de los que los usuarios pueden ser miembros en función de las membresías de grupos existentes, amigos con membresías de grupos similares u otros con gustos similares que no están en su red.

- Encontrar clientes que comparten lo mismo o que tienen compras de productos similares.

- Encontrar productos que compran personas con gustos similares y que el usuario no ha comprado. [16]

3.4.2.- Knowledge graph (Grafo de conocimiento)

La definición de un gráfico de conocimiento es una estructura de datos (generalmente contenida en una base de datos de gráficos) que representa las cosas y cómo se relacionan entre sí. Son los datos los que describen una red, donde las relaciones entre las cosas (u objetos comerciales) son al menos tan importantes como las cosas en sí mismas.

Los grafos de conocimiento se suelen usar para enlazar entidades (objetos reales o abstractos) entre sí mediante propiedades semánticas. Algunos de sus usos más comunes son:

- En periodismo, organizaciones como BBC y Reuters, utilizan grafos para seleccionar y organizar su contenido, mejorando el periodismo, la búsqueda y la alimentación de datos personalizados.

- En el sector farmacéutico, los gráficos ayudan a hacer elegir las terapias para los pacientes y a encontrar curas para muchas enfermedades graves.

- Búsqueda y redes sociales, el grafo de conocimiento de Google está generando resultados de búsqueda más significativos e inteligentes. Las plataformas de redes sociales (por ejemplo, LinkedIn) utilizan grafos de conocimiento para seleccionar publicaciones y alimentar sus algoritmos publicitarios. Todos utilizan grafos por su capacidad para representar y analizar redes de usuarios para identificar personas influyentes y clústeres para impulsar su poder de marketing y sus productos. [23]

3.4.3.- Detección de fraude

En este apartado veremos como el uso de una base de datos orientada a grafos permite a la aseguradora Allianz Benelux, filial del grupo Allianz, detectar posibles fraudes.

Históricamente, crear visualizaciones internas de comportamientos sospechosos con tecnología relacional había sido demasiado exigente. Las medidas contra el fraude, como el rastreo de redes, eran simplemente demasiado difíciles de crear en una base de datos relacional. Sudaman, jefe de análisis de datos de Allianz, llama a este ineficiente proceso un enfoque "2 por 2", en el que las tablas del estilo de la base de datos SQL con filas y columnas no ofrecen intrínsecamente las profundas conexiones de datos contextuales que requiere la detección y prevención del fraude. No les permite extraer información contextual y relacional.

Las tecnologías de grafos permiten detectar actividades potencialmente fraudulentas ya que revelan visualmente las conexiones ilícitas ocultas del defraudador. La incorporación de todos los datos de los clientes en una base de datos de grafos también permite desvelar las verdaderas exposiciones al riesgo y detectar los riesgos no cubiertos o las coberturas superpuestas, en particular en el contexto del automóvil o del hogar.[24]

3.4.4.- Recomendaciones de carro de la compra en tiempo real

En este apartado veremos como una plataforma de comercio electrónico como eBay usa una base de datos orientada a grafos en su aplicación para el asistente de google, un altavoz inteligente que conversa con los usuarios a través de la voz.

Un ejemplo de lo que quiere conseguir ebay con la aplicación sería la frase: "*Mi esposa y yo vamos a acampar en Lake Tahoe la próxima semana, necesitamos una tienda de campaña*", la mayoría de los motores de búsqueda reaccionarían a la palabra "tienda". Pero el contexto adicional con respecto a la ubicación, la temperatura, el tamaño de la tienda, el paisaje, etc., normalmente se pierden. Sin embargo, este tipo de información específica es en realidad lo que informa muchas decisiones de compra. La transmisión o el mantenimiento de este contexto es a menudo una carga dejada al usuario y se necesitaba una nueva solución para eliminar el arduo trabajo asociado con las compras.

El objetivo de eBay era construir un motor de recomendaciones en tiempo real que comprendiera y aprendiera del lenguaje contextual proporcionado por el comprador y se centrara rápidamente en recomendaciones de productos específicos. eBay llama a este ejercicio de aprovechar la intención humana como el "santo grial" del comercio conversacional. Para lograr esto se requiere una combinación de procesamiento de lenguaje natural, aprendizaje automático, modelado predictivo y un motor de almacenamiento y procesamiento distribuido y en tiempo real que opera a través de Internet mientras se escalan para contener todo su catálogo de productos.

Para crear esta aplicación, eBay uso una bases de datos orientada a grafos, y un grafo de conocimientos que combinaría la comprensión del lenguaje natural y la inteligencia artificial para almacenar, recordar y aprender de las interacciones pasadas con los compradores.[25]

Capítulo 4
Metodología y resultados

En este capítulo se van a implementar algunos de los casos de usos mencionados en el capítulo anterior, uno por cada tipo de base de datos NoSQL, mostrando todos los pasos desde su instalación, hasta la configuración y uso. Para la implementación de los casos se va a usar una máquina virtual con la última versión de Ubuntu, así como diverso software libre y open-source complementario; en el anexo 2 se indica como instalar y configurar todo este software complementario.

4.1.- ALMACENAMIENTO DE CACHÉ EN MEMORIA CON REDIS

En este apartado veremos cómo almacenando el caché de una página web en una base de datos clave-valor como Redis, se consigue, entre otras cosas, reducir el tiempo de carga, mejorando así la experiencia del usuario al navegar por la web. Para esto se va a crear una

página web WordPress y, mediante un plugin disponible en este CMS para ello, usar Redis para almacenar la página en caché, posteriormente se medirán los tiempos de carga sin el caché en memoria y con él.

Para implementar de forma completa este caso de uso es necesario, además de la propia base de datos Redis, el siguiente software (ver anexo 2 para detalles sobre su instalación y configuración):

- Máquina virtual con instalación del sistema operativo Ubuntu (A2.1).
- XAMPP: Infraestructura LAMP para desarrollar e instalar aplicaciones web (A2.2).
- WordPress: Gestor de contenidos (A2.3).
- Load Time: Extensión de Firefox, obtiene el tiempo de carga de una web (A2.4).

4.1.1.- Instalación de Redis

La instalación de la base de datos Redis en Linux se puede realizar mediante línea de comandos, los pasos serían los siguientes:

1) Instalar el servidor Redis.

```
sudo apt update
sudo apt install redis-server
```

2) Una vez instalado el servidor, conviene verificar que está funcionando correctamente entrando a Redis desde el terminal y haciendo un ping (si todo funciona, Redis responderá "PONG") [X].

```
redis-cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379>exit
```

3) Instalar la aplicación de escritorio Redis Desktop Manager (RDM) usando el comando "snap" que también hay que instalar previamente [X].

```
sudo apt update
sudo apt install snapd
sudo snap install redis-desktop-manager
```

4) Ejecutar la aplicación RDM desde el administrador de programas, conectar con el servidor local de Redis, poniendo un nombre a la conexión y dejamos el resto por defecto.

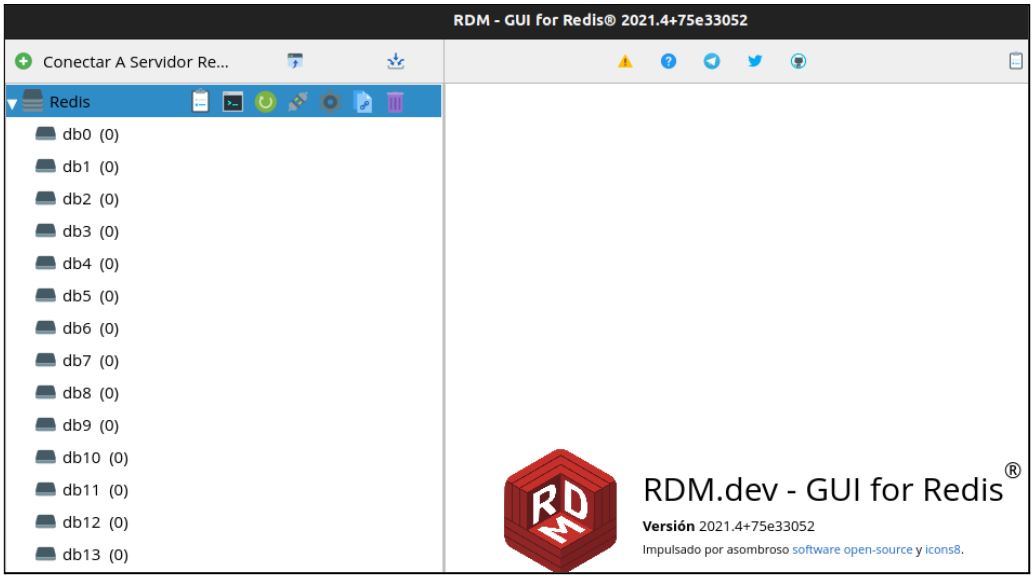


Figura 4.1.- Interfaz visual RDM, Redis Desktop Manager

4.1.2.- Configuración de WordPress y Redis

Ya con todos los componentes necesarios instalados, se accede a WordPress como administrador, desde el siguiente enlace.

```
http://localhost/wordpress/wp-login.php
```

Tras el login se accede al escritorio y en la pestaña de plugins, se localiza e instala el plugin de Redis, llamado “Redis Object Cache” (ver figura 4.2).

Figura 4.2.- Instalación del plugin de Redis

Para probar el funcionamiento del plugin, antes de activarlo, es necesario crear algún contenido en WordPress para luego cargarlo y ver el resultado, una forma rápida de conseguirlo es incorporando un tema. En la pestaña "Apariencia", opción "Temas", se ha optado por localizar e instalar el tema "ColorMag", que permite importar una demo con datos (ver figura 4.3).

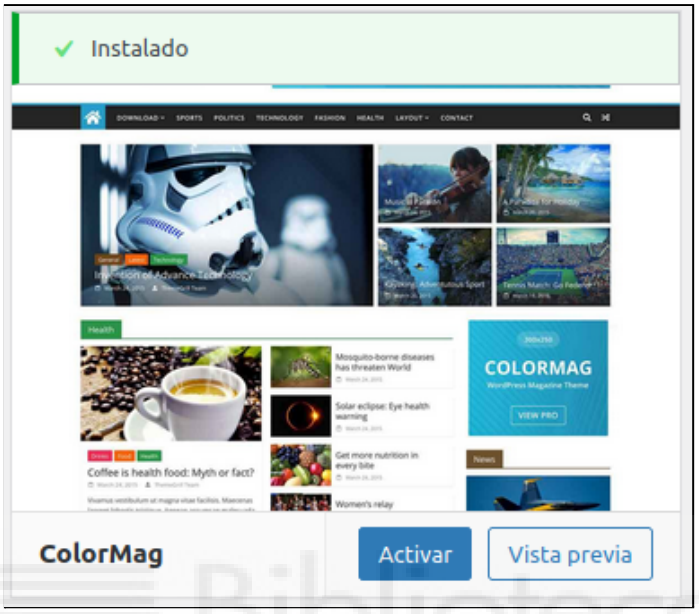


Figura 4.3.- Instalación de un tema en WordPress

Tras instalar el tema, se nos redirigirá a la página "Demo Importer", donde es posible elegir e importar una demo con datos. Una vez importados dichos datos entramos a la página "*http://localhost/wordpress/money*", que se acaba de importar y se comprueba el tiempo de carga sin tener Redis habilitado, en la figura 4.4 se observa que el tiempo para carga página sin Redis es de 1,53 segundos.

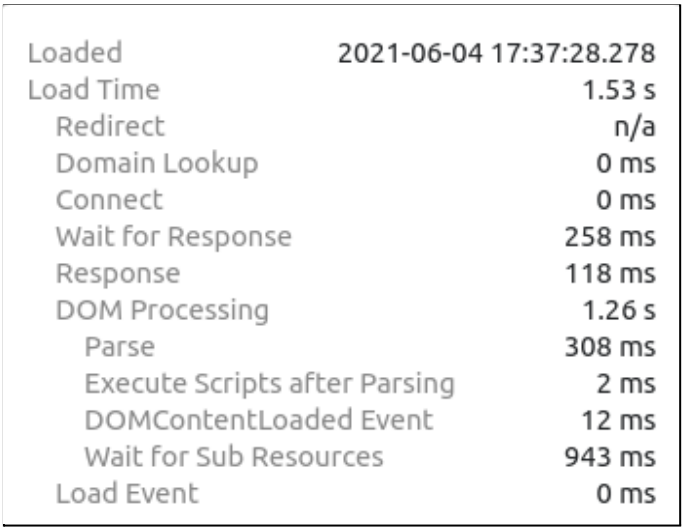


Figura 4.4.- Tiempo de carga sin Redis

Para habilitar Redis en la página volvemos a la vista de administrador de WordPress y en la pestaña de “Ajustes”, opción “Redis” se clica el botón “Activar caché de objetos”.

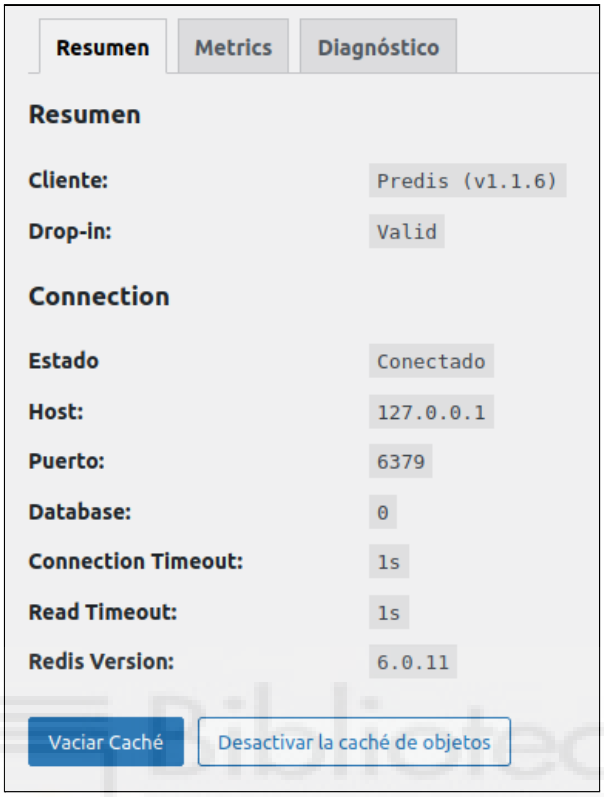


Figura 4.5.- Plugin de Redis activado tras pulsar “Activar caché de objetos”

Una vez activado el plugin, como se muestra en la figura 4.5, los datos de caché de la página se guardarán en el servidor Redis (figura 4.6).

Figura 4.6.- Datos de caché en Redis

Volviendo a la página donde se realizó el test anterior, al recargar dicha página, se observa que, en este caso, el tiempo de carga ahora es de 634 milisegundos, casi un 60% menos que en el caso anterior (ver figura 4.7).

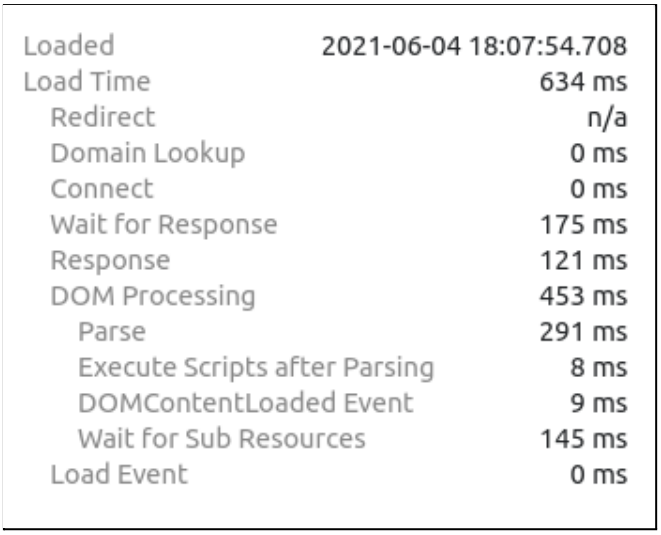


Figura 4.7.- Tiempo de carga de con Redis

Como se ha comentado anteriormente, esta reducción del tiempo de carga del caché de Redis en vez de cargar la página completa desde el servidor, y antes de cargar la página comprueba que tiene datos guardados en cache y que esos datos no hayan sufrido cambios, y tras comprobar que no se han realizado cambios carga los datos y metadatos de la página, si se hubiesen realizado algún cambio en la página o no existiesen datos almacenados la página se cargará desde el servidor y Redis guardará los datos nuevos en caché eliminando los desactualizados, si los hubiese. En la figura 4.8 podemos ver como Redis al no encontrar la página en caché la almacena. Para eso crea el "Post 1578", donde posteriormente guarda la información.

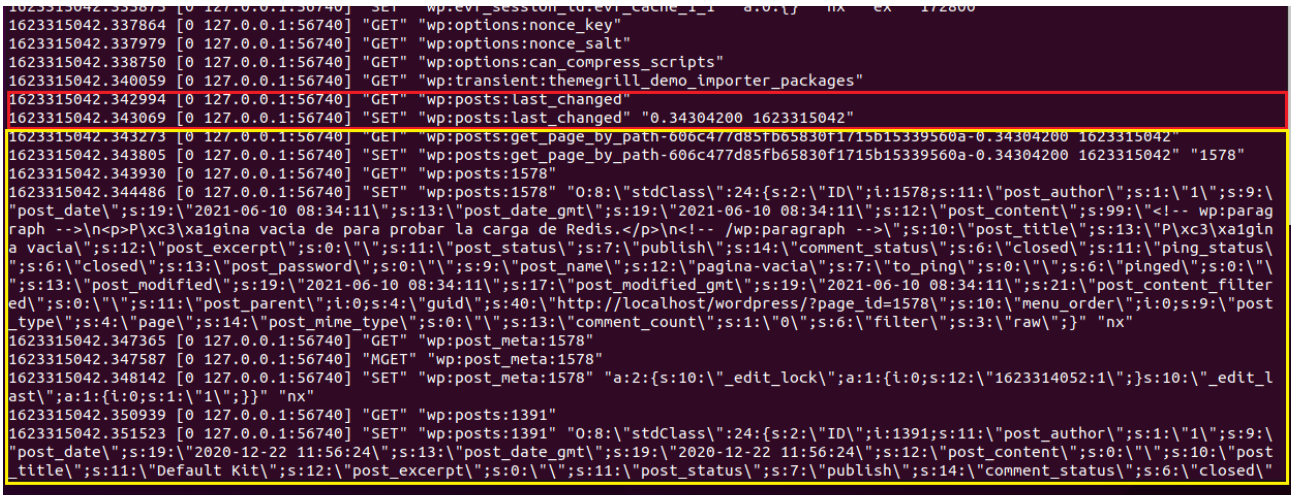


Figura 4.8.- Recuperación de datos desde Redis

4.2.- APLICACIÓN IoT CON MongoDB

En este apartado se va a mostrar un ejemplo de aplicación IoT, donde los datos de varias estaciones meteorológicas simuladas se actualizan en tiempo real en la base de datos documental MongoDB.

En la aplicación IoT se usará el protocolo de transporte de mensajes cliente/servidor MQTT (Message Queue Telemetry Transport), basado en publicaciones y suscripciones a los unos "tópicos". Cada vez que un mensaje es publicado en un tópico será recibido por el resto de dispositivos suscritos a dicho tópico del protocolo. Como se muestra en la figura 4.9, la aplicación IoT consta de varios emisores que publicaran y un único suscriptor que será MongoDB. Para simplificar el proyecto se usará Node-RED para publicar los mensajes JSON simulando las estaciones meteorológicas y para insertarlos en MongoDB.

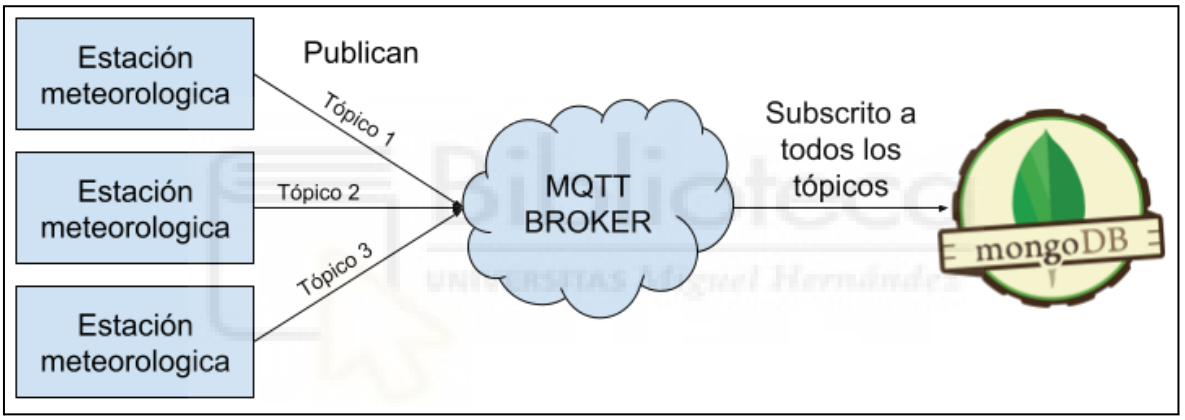


Figura 4.9.- Diagrama aplicacion IoT

Para implementar este caso de uso es necesario, además de la propia base de datos MongoDB, el siguiente software (ver anexo 2 para detalles sobre su instalación y configuración):

- Máquina virtual con instalación del sistema operativo Ubuntu (A2.1).
- Node-Red: Una herramienta de programación visual (A2.5).
- Mosquitto: Broker MQTT (A2.6).
- Metabase: Interfaz de usuario para los datos de MongoDB (A2.7)

4.2.1.- Instalación de MongoDB

La instalación de la base de datos MongoDB requiere los siguientes pasos:

1) Antes de instalar el software propiamente dicho es necesario importar la clave pública de GPG de MongoDB

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc |
sudo apt-key add -
```

2) Se crea el archivo "list" para que el comando apt reconozca el comando de instalación de mongoDB.

```
echo "deb [ arch=amd64,arm64 ]
https://repo.mongodb.org/apt/ubuntufocal/mongodb-org/4.4
multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.4.list
```

3) Se actualiza la lista de paquetes y se instala mongoDB.

```
sudo apt-get update
sudo apt-get install -y mongodb-org
```

4) Una vez instalado se le pueden dar permisos para que se inicie al arrancar el ordenador y, finalmente, lo ejecutamos.

```
sudo systemctl enable mongod
sudo systemctl start mongod
```


4.2.2.- Simulación de estaciones meteorológicas con Node-Red y vista de los datos con Metabase

Para simular el origen de los datos creamos en Node-Red una red con 3 estaciones meteorológicas que envían sus datos a través de una red MQTT, y son insertados en la base de datos, como se muestra en la figura 4.10 (en el anexo 2.5.1 se puede ver en más detalle los nodos de Node-Red).

Los datos que se insertan en la base de datos tienen formato JSON, la estructura del documento JSON contiene los siguientes elementos:

- topic: El tópico MQTT con el que identificar de qué estación provienen los datos, es una cadena de texto.
- Temperatura y Humedad: Los valores simulados de las estaciones.
- Fecha: Fecha y hora en que se produce el documento MQTT.
- _msgid: Un identificador que añade Node-Red

A continuación podemos ver 3 ejemplos de documentos JSON que se van a insertar en la base de datos.

```
{ topic: "Elche/EstacionA", Temperatura: 4, Humedad: 4, Fecha:
"2021-06-16T13:59:46.998Z", _msgid: "1e4948e0.88bb77" }

{ topic: "Elche/EstacionB", Temperatura: 9, Humedad: 3, Fecha:
"2021-06-16T13:59:48.826Z", _msgid: "56b6972b.6e3418" }

{topic:"Alicante/EstacionA", Temperatura: 6, Humedad: 4, Fecha:
"2021-06-16T13:59:49.989Z", _msgid: "8cbaf988.fd4928" }
```

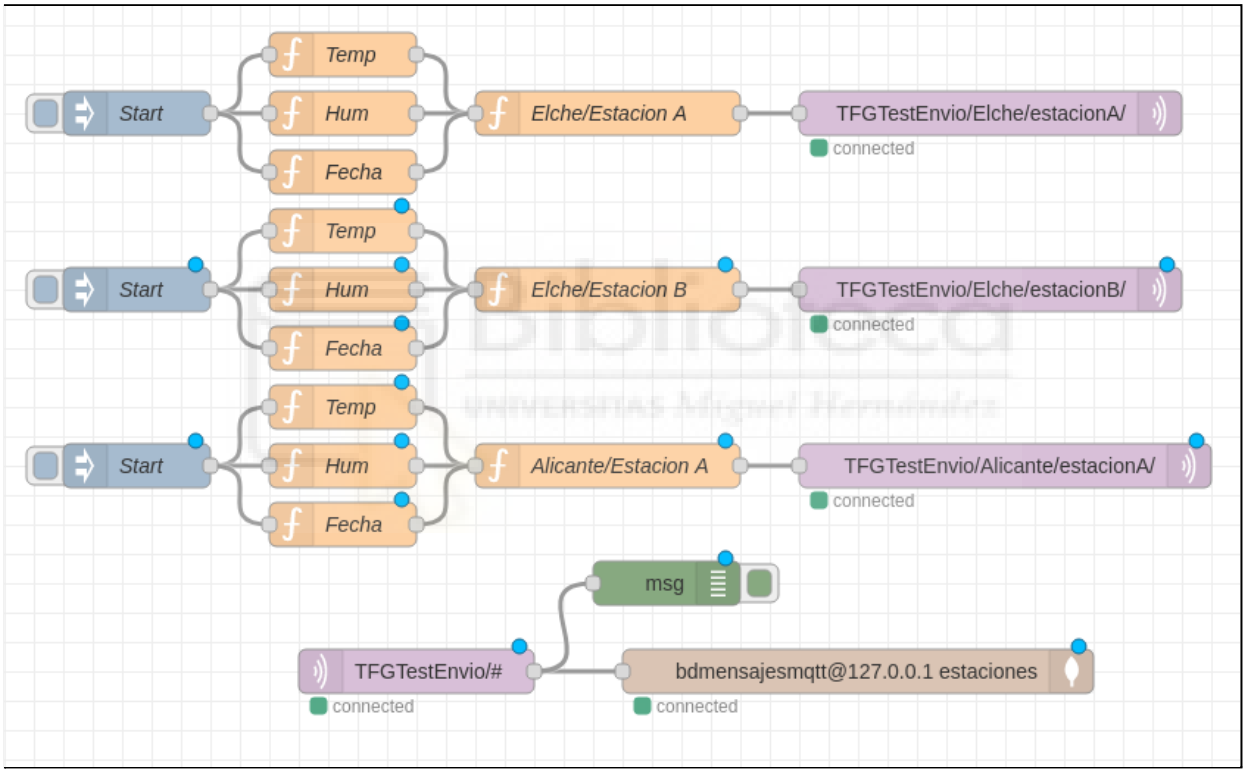


Figura 4.10.- Simulación de estaciones meteorológicas con Node-Red

Una vez insertados algunos datos vamos a la web de Metabase (anexo A2.7), en la dirección http://localhost:3000, que hemos iniciado y configurado anteriormente. Aquí es posible acceder a los datos almacenados en MongoDB, y con estos generar un gráfico. En la figura 4.11 se muestra un ejemplo de gráfico de área con los datos de una de las estaciones meteorológicas, donde en el eje X está la temperatura y humedad, y en eje Y la fecha.

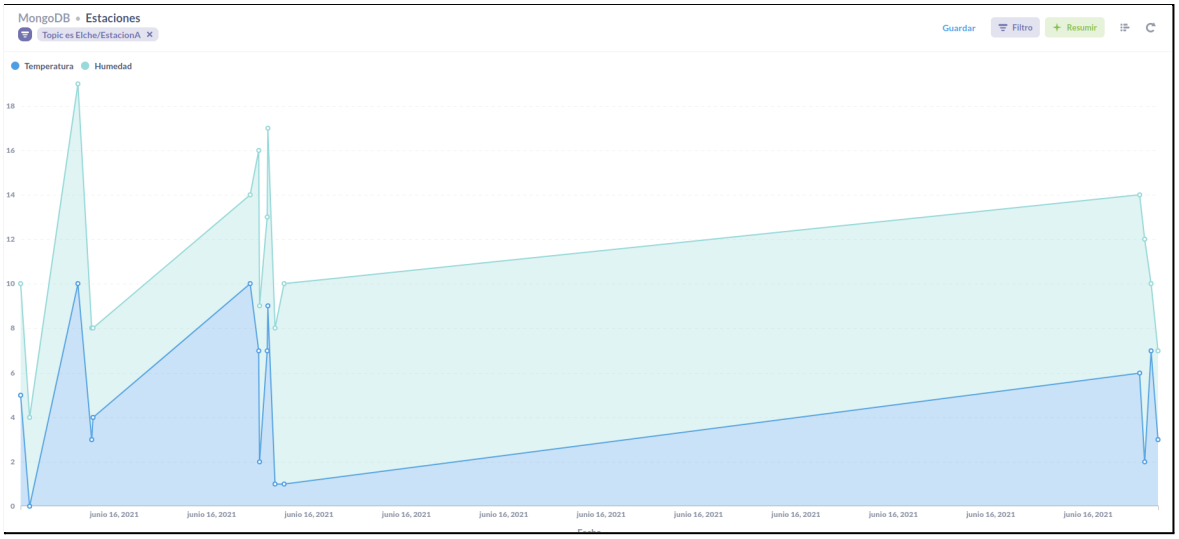


Figura 4.11.- Gráfica en Metabase con los datos de MongoDB

4.3.- SISTEMA DE RECOMENDACIÓN CON Neo4j

En este apartado se muestra un ejemplo de aplicación de una base de datos orientada a grafos, la base de datos usada es Neo4j. El ejemplo consiste en una red social de libros con un sistema de recomendación de libros y usuarios. Los usuarios indican que libros han leído y puede seguir o tener una relación de amigos con otros usuarios, y usando esa información se realizarán unas consultas con las que conseguiremos unas recomendaciones basadas en los libros que ha leído y en los usuarios que sigue.

4.3.1.- Instalación de Neo4J

La instalación de la base de datos Neo4J en Linux requiere de los siguientes pasos:

1) Importar la clave pública de GPG de Neo4J

```
curl -fsSL https://debian.neo4j.com/neotechnology.gpg.key | sudo
apt-key add -
```

2) Añadir el repositorio de Neo4J a la lista de fuentes.

```
sudo add-apt-repository "deb https://debian.neo4j.com stable 4.1"
```

3) Instalación de Neo4J

```
sudo apt install neo4j
```

4) Establecer que Neo4J se ejecute al arrancar el equipo.

```
sudo systemctl enable neo4j.service
```

4.3.2.- Creación de la base de datos en Neo4J

Una vez instalado podemos acceder a la dirección "http://localhost:7474/", donde podemos acceder a la interfaz de Neo4J. Inicialmente, el usuario y contraseña para acceder es "*neo4j*", una vez accedemos hay que cambiar la contraseña.

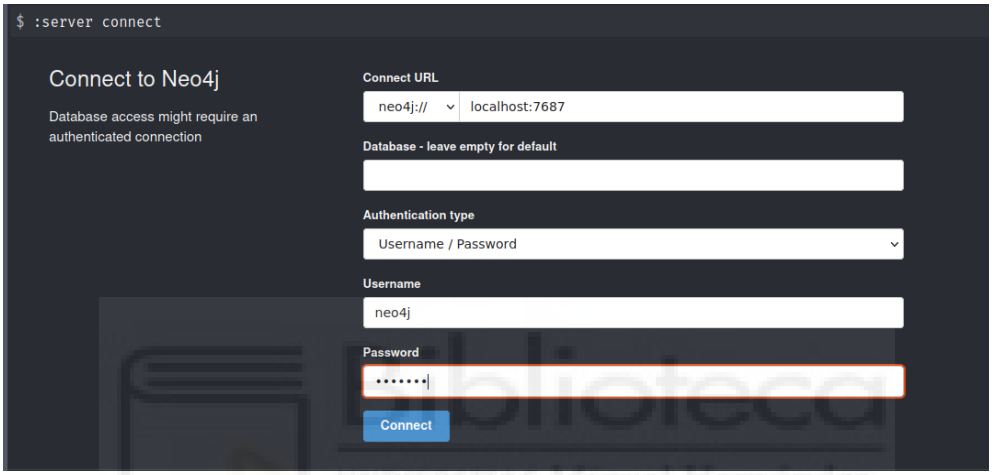


Figura 4.12.- Inicio de sesión Neo4J

Tras cambiar la contraseña, se inicia sesión y se crea una base de datos con el siguiente comando:

```
CREATE DATABASE neo4jtfg
```

Una vez creada la base de datos introducimos datos para simular a los usuarios de la red social de libros. La base de datos va a tener 3 tipos de nodos:

- personas
- libros
- géneros

También va a tener 3 tipos de relaciones (aristas entre nodos):

- BELONG_TO
- FOLLOW
- READ

El siguiente script generará el contenido de ejemplo en la base de datos:

```
CREATE
      (ii:Person {name:"Ian"}), (al:Person {name:"Albert"}),
      (an:Person {name:"Ann"}), (ab:Person {name:"Amber"}),

      (hh:Book {title:"Heima es hogar en islandes"}),
      (nv:Book {title:"El nombre del viento"}),
      (aa:Book {title:"La alianza"}),
      (hp:Book {title:"Harry Potter"}),
      (mi:Book {title:"Memorias de Idhun"}),
      (ce:Book {title:"La cripta embrujada"}),
      (pa:Book {title:"Pupila de águila"}),
      (ep:Book {title:"Escrito sobre la piel"}),
      (ee:Book {title:"Eragon"}),
      (dq:Book {title:"Desde que te fuiste"}),
      (cp:Book {title:"Ciudades de papel"}),
      (ma:Book {title:"Marina"}),

      (mm:Genre {name:"Misterio"}),
      (ff:Genre {name:"Fantasia"}),
      (jj:Genre {name:"Juvenil"}),

      (hh)-[:BELONG_TO]->(jj), (nv)-[:BELONG_TO]->(ff),
      (aa)-[:BELONG_TO]->(mm), (hp)-[:BELONG_TO]->(ff),
      (mi)-[:BELONG_TO]->(ff), (ce)-[:BELONG_TO]->(mm),
      (pa)-[:BELONG_TO]->(mm), (ep)-[:BELONG_TO]->(mm),
      (ee)-[:BELONG_TO]->(ff), (dq)-[:BELONG_TO]->(jj),
      (cp)-[:BELONG_TO]->(jj), (ma)-[:BELONG_TO]->(jj),

      (ii)-[:READ]->(hh), (ii)-[:READ]->(nv), (ii)-[:READ]->(aa),
      (ii)-[:READ]->(mi), (ii)-[:READ]->(ee), (ii)-[:READ]->(dq),
      (al)-[:READ]->(hp), (al)-[:READ]->(mi), (al)-[:READ]->(ce),
      (al)-[:READ]->(pa), (al)-[:READ]->(ep), (an)-[:READ]->(mi),
      (an)-[:READ]->(ce), (an)-[:READ]->(ma), (an)-[:READ]->(dq),
      (an)-[:READ]->(cp), (ab)-[:READ]->(aa), (ab)-[:READ]->(hp),
      (ab)-[:READ]->(pa), (ab)-[:READ]->(ep), (ab)-[:READ]->(dq),

      (ii)-[:FOLLOW]->(al), (ii)-[:FOLLOW]->(an),
      (an)-[:FOLLOW]->(ab);
```

La figura 4.13 muestra una vista del grafo resultante tras ejecutar el script anterior.

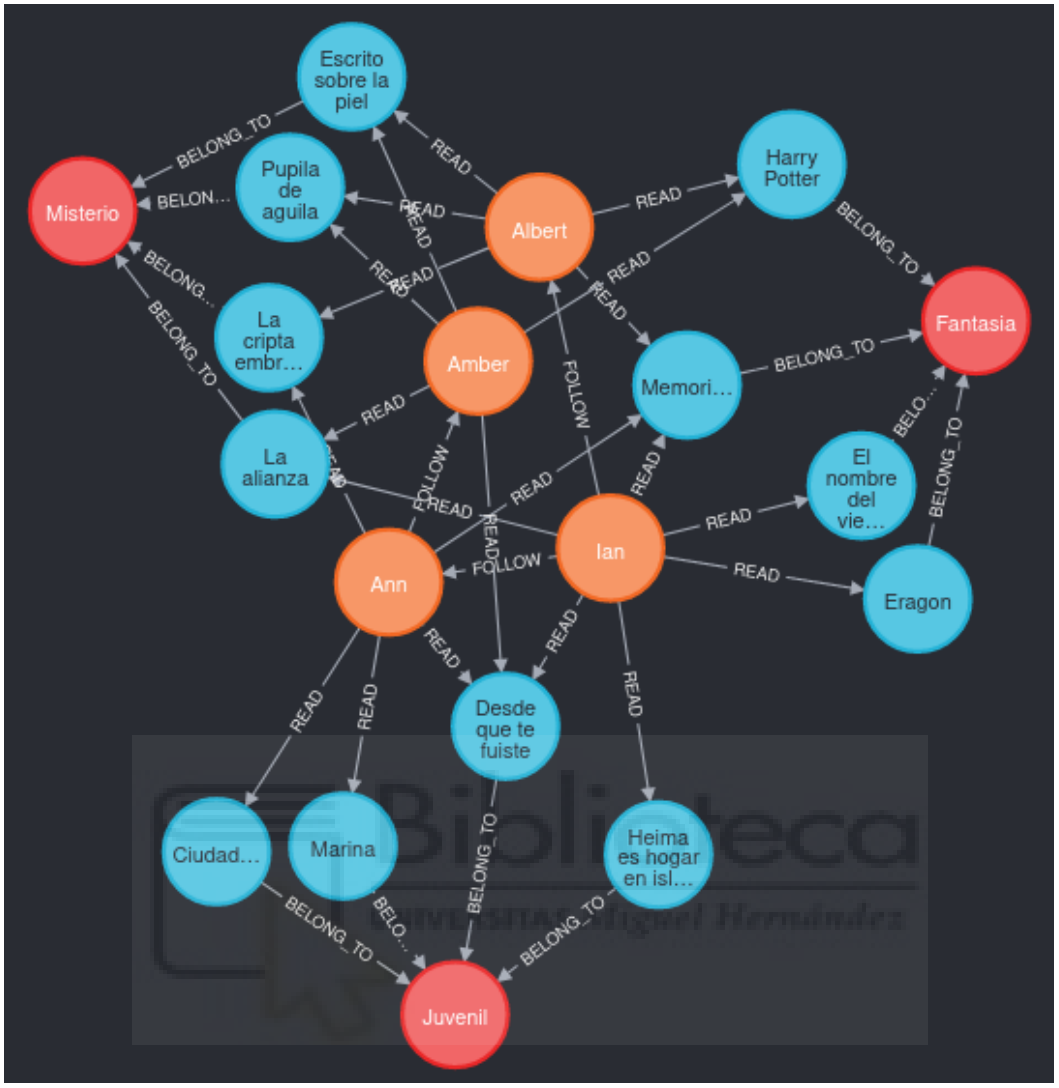


Figura 4.13.- Grafo de la base de datos Neo4J

4.3.3.- Recomendaciones de la base de datos Neo4J

Una vez creada la base de datos con los datos de prueba ya se pueden crear las recomendaciones. En este ejemplo se van a considerar dos tipos de recomendaciones, las de usuarios y las de libros.

4.3.3.1.- Recomendaciones de usuarios

La primera consulta que se va a plantear devolverá una recomendación de personas (usuarios) a los que seguir en función de los libros en común que tiene con otros usuarios, teniendo en cuenta que el usuario recomendado no esté siendo seguido previamente. A continuación se muestra la consulta que debe ejecutarse en Neo4j.

```
MATCH (p1:Person)-[:READ]->(b1:Book)<-[:READ]-(p2:Person)
WITH p1, p2, collect(b1) as LibrosComunes
WHERE not ((p1)-[:FOLLOW]->(p2))
RETURN p1 as Usuario1,
       [x in LibrosComunes | x.title] as LibrosComunes,
       p2 as UsuarioRecomendado;
```

En la tabla 4.1 vemos el resultado de la consulta donde "Usuario1" recibirá las recomendaciones de seguimiento de otros usuarios que tienen algún libro en común.

Tabla 4.1.- Resultado de la consulta de recomendación de usuarios.

Usuario1	LibrosComunes	UsuarioRecomendado
{"name":"Amber"}	["Desde que te fuiste","La alianza"]	{"name":"Ian"}
{"name":"Ann"}	["Desde que te fuiste","Memorias de Idhun"]	{"name":"Ian"}
{"name":"Albert"}	["Memorias de Idhun"]	{"name":"Ian"}
{"name":"Amber"}	["Escrito sobre la piel","Pupila de aguila","Harry Potter"]	{"name":"Albert"}
{"name":"Ann"}	["La cripta embrujada","Memorias de Idhun"]	{"name":"Albert"}
{"name":"Amber"}	["Desde que te fuiste"]	{"name":"Ann"}
{"name":"Albert"}	["La cripta embrujada","Memorias de Idhun"]	{"name":"Ann"}
{"name":"Ian"}	["Desde que te fuiste","La alianza"]	{"name":"Amber"}
{"name":"Albert"}	["Escrito sobre la piel","Pupila de aguila","Harry Potter"]	{"name":"Amber"}

Otra forma de recomendar usuarios es la recomendación de "*amigos de amigos*", es decir, recomendar seguir a los usuarios que están siguiendo los usuarios a los que ya sigues. Esta recomendación es la más utilizada en redes sociales, como facebook o instagram. En la tabla 4.2 podemos ver el resultado de la consulta de recomendación de amigos de amigos que vemos a continuación.

```
MATCH (p1:Person)-[:FOLLOW]->(p2:Person)-[:FOLLOW]->(p3:Person)
WHERE not ((p1)-[:FOLLOW]->(p3))
RETURN p1 as Usuario, p2 as AmigoComun, p3 as UsuarioRecomendado;
```

Tabla 4.2.- Resultado recomendación de amigos de amigos

Usuario	AmigoComun	UsuarioRecomendado
{"name":"Ian"}	{"name":"Ann"}	{"name":"Amber"}

4.3.3.2.- Recomendaciones de libros

A la hora de recomendar libros se busca que los libros recomendados a un usuario no los haya leído, y que sean una buena recomendación, para conseguir esto vamos a poner un mínimo de 2 libros leídos en común entre los usuarios para que la recomendación sea más útil. Esta recomendación se hace con la consulta que se muestra a continuación. El resultado es el que se muestra en la tabla 4.3.

```
MATCH
(p1:Person)-[:READ]->(b1:Book)<-[:READ]-(p2:Person)-[:READ]->(b2:Book)
WITH p1, p2, count(b1) as NLibros, collect(b1) as LibrosComunes, b2
WHERE not ((p1)-[:READ]->(b2)) and NLibros >=2
RETURN p1.name as Persona1,
       p2.name as Persona2,
       [x in LibrosComunes | x.title] as LibrosComunes,
       b2 as LibroRecomendado;
```

Tabla 4.3.- Recomendación de libros basada en libros en común para un usuario.

Persona1	Persona2	LibrosComunes	LibroRecomendado
Albert	Ann	["La cripta embrujada","Memorias de Idhun"]	{"title":"Ciudades de papel"}
Albert	Ann	["La cripta embrujada","Memorias de Idhun"]	{"title":"Desde que te fuiste"}
Albert	Ann	["La cripta embrujada","Memorias de Idhun"]	{"title":"Marina"}
Albert	Amber	["Escrito sobre la piel","Pupila de aguila","Harry Potter"]	{"title":"Desde que te fuiste"}
Albert	Amber	["Escrito sobre la piel","Pupila de aguila","Harry Potter"]	{"title":"La alianza"}

Otra forma de recomendar libros a los usuarios es por medio del género favorito del usuario, para eso con la siguiente consulta se identifica el género que más se repite entre los libros de un usuario y se le recomiendan libros del mismo género, que no haya leído. Se puede ver el resultado de la consulta en la tabla 4.4.

```
MATCH
(p:Person)-[:READ]->(b1:Book)-[:BELONG_TO]->(g:Genre)<-[:BELONG_TO]-(b2:Book)
WITH p,g,count(b1) as Generos,b2
WHERE NOT ((p)-[:READ]->(b2)) AND Generos>1
RETURN p.name as Persona,
       g.name as Genero,
       collect(b2.title) as LibrosRecomendados;
```

Tabla 4.4.- Recomendación de libros por género.

Persona	Género	LibrosRecomendados
Amber	Misterio	["La cripta embrujada"]
Albert	Misterio	["La alianza"]
Albert	Fantasía	["Eragon","El nombre del viento"]
Ian	Fantasía	["Harry Potter"]
Ian	Juvenil	["Marina","Ciudades de papel"]
Ann	Juvenil	["Heima es hogar en islandés"]

4.4.- EJEMPLO DE USO DE APACHE CASSANDRA

Apache Cassandra es una base de datos distribuida con una alta escalabilidad y de alto rendimiento diseñada para manejar grandes cantidades de datos en muchos servidores, proporcionando alta disponibilidad sin puntos de falla.

En Cassandra, uno o más de los nodos de un clúster actúan como réplicas de un dato determinado, usando el protocolo Gossip, un protocolo de comunicación peer to peer que se usa para replicar la información en una red de nodos. Esto permite que los nodos se comuniquen entre sí en segundo plano y detecten cualquier nodo defectuoso en el clúster. En la figura 4.14 se muestra una vista esquemática de cómo Cassandra usa la replicación de datos entre los nodos de un clúster para garantizar que no haya un solo punto de falla.

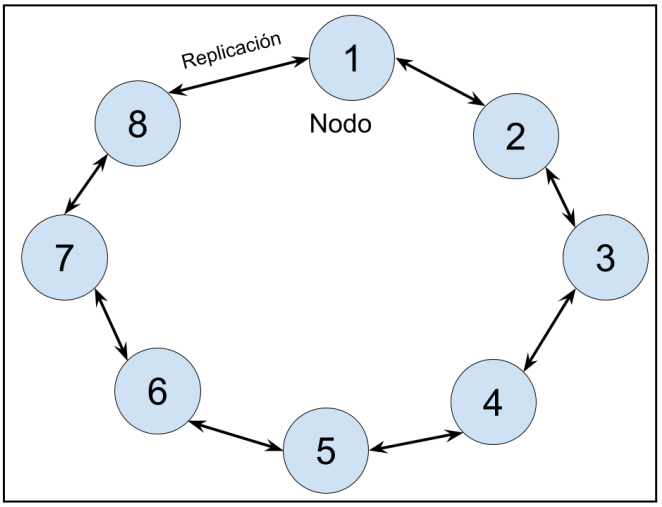


Figura 4.14.- Nodos en una arquitectura distribuida en un cluster de apache cassandra

4.4.1.- Instalación de Apache Cassandra

A continuación se muestra cómo instalar Apache Cassandra y un ejemplo simple de uso. Para instalar Apache Cassandra en ubuntu seguimos los siguientes pasos:

1) Lo primero antes de instalar Cassandra es instalar Java, en este caso instalamos la versión de Java 17.

```
sudo apt update
sudo apt install openjdk-17-jre-headless
java -version
```

2) Tras instalar Java, el primer paso para instalar Apache Cassandra consiste en obtener la clave GPG del repositorio.

```
wget -q -O - https://www.apache.org/dist/cassandra/KEYS | sudo apt-key
add --
```

3) Una vez tenemos la clave del repositorio lo añadimos a nuestra lista de fuentes.

```
sudo sh -c 'echo "deb http://www.apache.org/dist/cassandra/debian 311x
main" \
> /etc/apt/sources.list.d/cassandra.sources.list'
```

4) Una vez completados los pasos anteriores instalamos Apache Cassandra.

```
sudo apt update
sudo apt install cassandra
```

5) Por último, se inicia el servicio y se indica que este debe arrancar cada vez que arranque el equipo.

```
sudo service cassandra start
sudo systemctl enable cassandra.service
```

4.4.2.- Ejemplo de uso Apache Cassandra

A continuación se expondrá un ejemplo simple de como crear una base de datos en Apache Cassandra usando un único nodo. Para sacar el máximo provecho a Apache Cassandra lo ideal sería tener un cluster de varios servidores dentro de uno o en varios centros de datos, pero para disminuir la complejidad del ejemplo usaremos un único nodo. El ejemplo consiste en construir una base de datos en Cassandra para un concesionario de coches, donde veremos como crear la base de datos, como realizar operaciones CRUD usando

CQL (Cassandra Query Language). El primer paso para crear la base de datos en Cassandra es crear un Keyspace. El Keyspace es el contenedor de datos más externo en una base de datos columnar, el concepto de Keyspace sería análogo al de la base de datos en un sistema gestor de bases de datos relacionales convencional. El Keyspace contiene familias de columnas, índices, tipos definidos por el usuario. En la figura 4.15 podemos ver un esquema de cómo estaría un Keyspace en Cassandra.

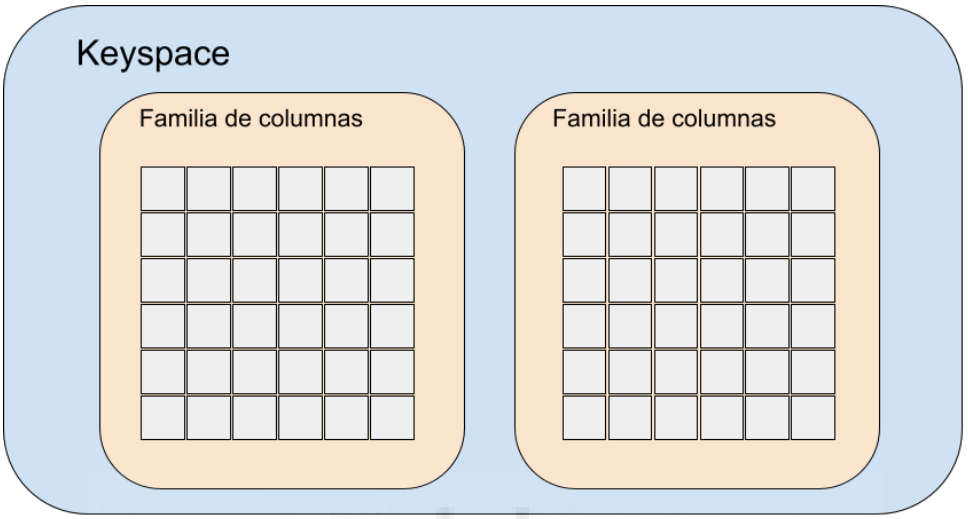


Figura 4.15.- Esquema de un keyspace de Cassandra

A continuación vamos a ver la sintaxis para crear un keyspace en Apache Cassandra. Para entrar a Cassandra abrimos un terminal y usamos el comando "cqlsh" (CQL Shell).

```
CREATE KEYSPACE concesionario
WITH REPLICATION = {'class':'SimpleStrategy', 'replication_factor':1};
```

En el comando anterior, el factor de replicación ('*replication_factor*') es la cantidad de nodos del cluster que recibirán una copia de los datos. Por otra parte, el tipo de estrategia ('*class*') es para indicar como colocar las réplicas en el clúster. La estrategia puede ser simple ('*Simple Strategy*') para cuando hay un único centro de datos, y de topología de red ('*Network Topology Strategy*'), para cuando hay más de un centro de datos. En este caso particular, con un solo nodo, se usará una estrategia simple y el factor de replicación a 1.

Una vez creado el keyspace se puede modificar en cualquier momento para cambiar la estrategia o el factor de replicación. También es posible eliminarlo.

```
ALTER KEYSPACE concesionario
WITH replication = {'class': 'Strategy name', 'replication_factor' :
'No.Of  replicas'};

DROP KEYSPACE concesionario
```

Una familia de columnas puede ser vista como una colección de filas, y cada fila es una colección de columnas. Es análoga a una tabla en RDBMS pero tiene algunas diferencias. Las familias de columnas están definidas, pero no es necesario para cada fila tener todas las columnas, y las columnas pueden ser agregadas o eliminadas de una fila. A continuación se muestra cómo crear una familia de columnas en un keyspace de Cassandra.

```
USE concesionario;

CREATE TABLE coche(
    nserie text,
    marca text,
    modelo text,
    anyo int,
    color text,
    matricula text,
    combustible text,
    descuento int,
    valor int,
    propietarios int,
    estado text,
    PRIMARY KEY (nserie));
```

Una vez creada la familia de columnas podemos añadir o eliminar una columna dentro de la familia de columnas de la siguiente forma. Podemos añadir o eliminar columnas cuando queramos pero hay que tener en cuenta que al eliminar una columna todos los datos que pudiese haber en la columna se eliminaran.

```
ALTER TABLE coche
ADD motor text;

ALTER TABLE coche
DROP estado;
```

Una vez tenemos creada la familia de columnas podríamos eliminarla con el siguiente comando.

```
DROP TABLE coche
```

Una vez realizados los cambios podemos ver el esquema resultante de la familia de columnas con el siguiente comando.

```
DESC SCHEMA;
```

```
cqlsh:concesionario> DESC SCHEMA;

CREATE KEYSPACE concesionario WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1
'}  AND durable_writes = true;

CREATE TABLE concesionario.coche (
    nserie text PRIMARY KEY,
    anyo int,
    color text,
    combustible text,
    descuento int,
    marca text,
    matricula text,
    modelo text,
    motor text,
    propietarios int,
    valor int
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'ma
x_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Comp
ressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

Figura 4.16.- Esquema del Keyspace

Una vez tenemos el esquema de la familia de columnas pasamos a introducir los datos.

```
INSERT INTO coche (nserie, marca, modelo, anyo, color, matricula,
combustible, descuento, valor, propietarios) VALUES ('VW1001',
'Volkswagen', 'Golf', 2020, 'Negro', '0001AAA', 'Diesel', 10, 32000,
0);

INSERT INTO coche (nserie, marca, modelo, anyo, color, matricula,
combustible, descuento, valor) VALUES ('T10041', 'Toyota', 'Yaris',
2020, 'Blanco', '0002AAA', 'Hibrido', 15, 42000);

INSERT INTO coche (nserie, marca, anyo, color, matricula, combustible,
propietarios) VALUES ('VW0001', 'Volkswagen', 1965, 'Negro', 'A0001',
'Gasolina', 6);

INSERT INTO coche (nserie, marca, modelo, anyo, color, matricula,
combustible, propietarios) VALUES ('AF0081', 'Alfa Romeo', 'Alfetta',
1982, 'Rojo', 'A0010AA', 'Gasolina', 4);

INSERT INTO coche (nserie, marca, modelo, anyo, color, matricula,
combustible, motor) VALUES ('AU2854', 'Audi', 'A3', 2012, 'Blanco',
'0045AAA', 'Diesel','1998cc');

INSERT INTO coche (nserie, marca, modelo, anyo, color, matricula,
combustible, motor) VALUES ('S1274', 'SEAT', 'LEON', 2018, 'Rojo',
```

```
'0285AAA', 'Gasolina','1600cc');
```

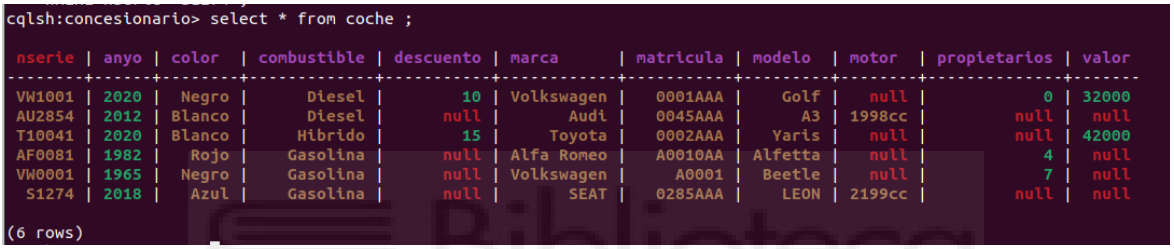
Una vez tenemos introducidos los datos podemos modificarlos con el comando UPDATE.

```
UPDATE coche SET modelo='Beetle', propietarios=7
      WHERE nserie='VW0001';

UPDATE coche SET motor='2199cc', color='Azul'
      WHERE nserie='S1274';
```

También podemos leer los datos introducidos en la familia de columnas con el siguiente comando.

```
SELECT * FROM coche;
```



```
cqlsh:concesionario> select * from coche ;

 nserie | anyo | color  | combustible | descuento | marca      | matricula | modelo  | motor  | propietarios | valor
--------+------+--------+-------------+-----------+------------+-----------+---------+--------+--------------+-------
 VW1001 | 2020 |  Negro |      Diesel |        10 | Volkswagen |   0001AAA |    Golf |   null |            0 | 32000
 AU2854 | 2012 | Blanco |      Diesel |      null |       Audi |   0045AAA |      A3 | 1998cc |         null |  null
 T10041 | 2020 | Blanco |     Hibrido |        15 |     Toyota |   0002AAA |   Yaris |   null |         null | 42000
 AF0081 | 1982 |   Rojo |    Gasolina |      null | Alfa Romeo |   A0010AA | Alfetta |   null |            4 |  null
 VW0001 | 1965 |  Negro |    Gasolina |      null | Volkswagen |     A0001 |  Beetle |   null |            7 |  null
  S1274 | 2018 |   Azul |    Gasolina |      null |       SEAT |   0285AAA |    LEON | 2199cc |         null |  null

(6 rows)
```

Figura 4.17.- Resultado del select en el Keyspace

Y si queremos eliminar un registro de la familia de columnas lo podemos hacer con el siguiente comando.

```
DELETE FROM coche WHERE  nserie='S1274';
```

Capítulo 5
Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

Al inicio de este trabajo de fin de grado nos marcamos como objetivo crear un estudio comparativo de todas las bases de datos no relacionales (NoSQL), pero ante el gran número de tipos de bases de datos NoSQL diferentes preferimos centrarnos en los cuatro tipos más comunes: las clave-valor, las documentales, las columnares y las orientadas a grafos.

Una vez que pasamos a explicar y comparar las bases de datos NoSQL vemos que la mayoría de ellas comparten las características de disponibilidad y tolerancia a la partición, ya que estas bases de datos NoSQL suelen estar diseñadas para maximizar la disponibilidad a expensas de la consistencia que se consigue eventualmente con el paso del tiempo.

Tal y como se planteó en los objetivos del trabajo, se han presentado las tecnologías de almacenamiento y bases de datos, viendo como han ido evolucionando a lo largo del tiempo hasta llegar al actual paradigma NoSQL, se han estudiado los diferentes planteamientos dentro de este paradigma, se han presentado diversos casos de uso, poniendo en práctica algunos de ellos, adquiriendo experiencia en el manejo de este tipo de tecnologías.

5.2.- POSIBLES DESARROLLOS FUTUROS

El presente trabajo es una primera toma de contacto con el paradigma NoSQL, el estudio realizado representa una base a partir de la cual poder abordar nuevos proyectos sobre este paradigma, sus casos de uso y determinadas aplicaciones prácticas concretas.

Dada la característica de escalado horizontal que presentan estas bases de datos, son ideales para desarrollar nuevos trabajos en los que se necesite operar con un volumen de datos muy grande, es decir, pueden ser aplicadas en problemas de Big Data, Internet de las Cosas (IoT) u otros casos donde se necesite analizar gran cantidad de datos en tiempo real.

Adicionalmente, gracias a su naturaleza "flexible", permiten trabajar con datos de naturaleza cambiante, es decir, un registros en una base de datos relacional siempre va a tener la misma estructura y, por ejemplo, las bases de datos documentales, almacenan objetos JSON (o XML o de otro tipo) que pueden ser diferentes unos de otros, pero que es posible almacenar en la misma estructura de datos. En otras palabras, permiten trabajar con datos semiestructurados o, en el caso de las bases de datos clave-valor, donde el valor puede ser cualquier estructura binaria, incluso se pueden almacenar datos no estructurados.

Otra posible línea de trabajo futuro puede ser investigar el paradigma denominado NewSQL. Como se vio en el teorema CAP (epígrafe 2.1) las bases de datos NoSQL buscan sobre todo tener sistemas altamente escalables y tolerantes a la partición. El paradigma NewSQL quiere construir sistemas consistentes a la vez que escalables, sacrificando para ello la tolerancia a la partición.

Bibliografía

[1] The History of Storage Systems
Kazuo Goda, Masaru Kitsuregawa
https://ieeexplore.ieee.org/abstract/document/6182574
Publicado: Abril 2012 / Consultado: Octubre 2020

[2] History Of Databases
Kristi L. Berg, Tom Seymour, Richa Goel
https://clutejournals.com/index.php/IJMIS/article/view/7587
International Journal of Management & Information Systems, Volume 17, Num. 1
2013

[3] Database Concepts. 3rd ed.
Kroenke, D. & Auer, D.
Prentice (New York) 2007

[4] 25 years of database history (starting in 1955)
DuCharme, B.
http://www.snee.com/bobdc.blog/2005/12/25-years-of-database-history-s-1.html
Publicado: Diciembre 2005 / Consultado: Octubre 2020

[5] The Evolution of the Computerized Database
Bercich, N.
http://concept.journals.villanova.edu/article/view/311/274
2002

[6] Database Development for Dummies.
Taylor, A.
John Wiley & Sons, Inc (New York) 2000

[7] SQL for Dummies.
Taylor, A.
John Wiley & Sons, Inc (New York). 2007

[8] Databases DeMYSTiFieD—Hard Stuff made easy.
Oppel, A.
McGraw-Hill (New York). 2011

[9] The Evolution of Database
Haadi, M.
http://mhaadi.wordpress.com/2010/10/18/the-evolution-of-database/
Publicado: Octubre 2010 / Consultado: Octubre 2020

[10] Databases: Past, Present, and Future
Paragon Corporation
http://www.paragoncorporation.com/ArticleDetail.aspx?ArticleID=20
Publicado: Junio 2003 / Consultado: Octubre 2020

[11] List of NoSQL database management systems
https://hostingdata.co.uk/nosql-database/
Consultado: Marzo 2021

[12] SQL, NewSQL, and NoSQL Databases: A Comparative Survey
T. N. Khasawneh, M. H. AL-Sahlee and A. A. Safia
https://ieeexplore.ieee.org/abstract/document/9078970
Publicado: April 2020 / Consultado: Marzo 2021

[13] NoSQL Distilled - A Brief Guide to the Emerging World of Polyglot Persistence
P.J. Sadalage, M. Fowler
Addison Wesley (Indiana). 2012

[14] Multi-Model Database Systems: The State of Affairs
Mihai, G.
http://www.eia.feaa.ugal.ro/images/eia/2020_2/Mihai_Gianina.pdf
Publicado: Agosto 2020 / Consultado: Abril 2021

[15] Session Management
RedisLabs
https://redislabs.com/solutions/use-cases/session-management/
Consultado: Mayo 2021

[16] NoSQL for Dummies
Fowler, A.
John Wiley & Sons, Inc (New Jersey). 2015

[17] Redis Enterprise, The High-Performance Caching Solution For Your Applications
RedisLabs
https://redislabs.com/solutions/use-cases/caching/
Consultado: Mayo 2021

[18] BlockChain Technology: Beyond Bitcoin
M.Crosby, Nachiappan, P.Pattanayak, S.Verma, V.Kalyanaraman
https://j2-capital.com/wp-content/uploads/2017/11/AIR-2016-Blockchain.pdf
AIR - Applied Innovation Review, Issue No. 2 June 2016

[19] Identity management application using Blockchain, MongoDB Stitch & MongoDB Atlas
https://www.mongodb.com/blog/post/identity-management-application-using-blockchain-mongodb-stitch--mongodb-atlas--part-1
MongoDB Blog
Consultado: Mayo 2021

[20] Real-Time Big Data Analytics, Design, process, and analyze large sets of complex data in real time
S.Gupta, S. Saxena
Packt Publishing, February 2016

[21] Bosch Leads Charge into Internet of Things
https://www.mongodb.com/customers/bosch
MongoDB Blog
Consultado: Mayo 2021

[22] MongoDB for Gaming
https://www.mongodb.com/use-cases/gaming
MongoDB Blog
Consultado: Mayo 2021

[23] Knowledge graphs: building smarter financial services
M.Kitson, C.Probert
CAPCO, 2020

[24] Allianz Benelux Case Study
For major insurer Allianz Benelux, graphs are at the core of its data strategy, positioning them for the future and allowing them to be truly customer-centric
https://neo4j.com/case-studies/allianz-benelux/
Consultado: Mayo 2021

[25] eBay Case Study
Neo4j Powers Intelligent Commerce for eBay App on Google Assistant
https://neo4j.com/case-studies/ebay/
Consultado: Mayo 2021

[26] Running Node-RED locally
https://nodered.org/docs/getting-started/local
Consultado: Junio 2021

Anexo 1 Historia de los sistemas de almacenamiento y bases de datos

A1.1.- HISTORIA DE LOS SISTEMAS DE ALMACENAMIENTO

Un dispositivo de almacenamiento es un conjunto de componentes utilizados con el fin de leer o grabar datos en el soporte de almacenamiento de datos de forma temporal o permanente. En este apartado se describe la evolución de los sistemas de almacenamiento, desde las tarjetas perforadas hasta los sistemas de almacenamiento en la nube.

A1.1.1.- Medios de almacenamiento en los primeros ordenadores

Los primeros computadores usaban papel para almacenar información. La idea de utilizar papel como medio de almacenamiento de información para computadores se remonta a Charles Babbage, un matemático inglés que inventó una calculadora mecánica denominada

Motor de diferencia, diseñada para tabular funciones polinomiales en la década de 1820. Pero no fue hasta la década de 1880 cuando Herman Hollerith, un estadístico estadounidense, inventó un mecanismo que podía detectar eléctricamente un agujero en una tarjeta perforada y creó el prototipo de una máquina que podía tabular estadísticas a partir de una serie de tarjetas perforadas.

En 1890 la oficina del censo de los Estados Unidos se enfrentaba a un grave problema provocado por el gran aumento de población inmigrante. El censo se realizaba cada diez años, y el censo de 1880 tardó casi ocho años en completar la tabulación. En ese momento parecía que el censo de 1890 no se podría completar antes del 1900. Para resolver este problema la oficina del censo de EE.UU. realizó un concurso público y tras analizar varias propuestas finalmente decidieron adoptar la idea de Hollerith. La máquina tabuladora de Hollerith fue usada para tabular el censo de 1890 completando la tarea y verificando dos veces los datos en 18 meses. Hollerith comenzó un negocio y continuó desarrollando sus máquinas y alquilándolas a oficinas de censos y compañías de seguros de todo el mundo.

La empresa de máquinas tabuladoras que fundó formó la base de IBM a través de una fusión empresarial. Después de la tarjeta original de Hollerith con 24 columnas y 12 filas, otras empresas diseñaron y fabricaron muchos tipos de tarjetas perforadas. El formato de tarjeta principal fue la tarjeta IBM con 80 columnas por 24 filas.

Hasta la década de 1950, las tarjetas perforadas eran el medio más popular tanto para la transferencia de datos como para el almacenamiento de información. Poco después de que se inventara el computador electrónico y se hiciera popular, las tarjetas perforadas fueron reemplazadas por cinta magnética para el almacenamiento de información persistente, pero debido a su fácil manejo y compatibilidad heredada, las tarjetas perforadas continuaron siendo ampliamente utilizadas para transferir datos y programas entre sistemas informáticos hasta mediados de la década de 1980 [1].

A1.1.2.- Cintas magnéticas y librerías de cintas

La idea original de usar impresiones magnéticas para registrar información fue presentada por Oberlin Smith, un ingeniero mecánico estadounidense. Smith sugirió usar hilo de algodón o seda, en el que se podría suspender el polvo de acero o recortes cortos de alambre fino. Estas partículas debían magnetizarse de acuerdo con la corriente alterna de una fuente de micrófono. Finalmente publicó su idea en Electrical World, una revista británica de ingeniería, en 1888. Las ideas de Smith fueron desarrolladas posteriormente por el ingeniero danés Valdemar Poulsen. En 1894 Poulsen concibió el principio de la grabación magnética y en 1898, llevó a la práctica por primera vez la utilización de un cable magnetizable como medio físico para grabar el sonido, en un dispositivo de su

invención al que denominó Telegraphone, siendo éste el primer aparato práctico para la grabación y reproducción de sonido.

La cinta magnética que es común hoy en día fue inventada por el alemán Fritz Pfleumer en 1928. Su cinta original estaba hecha de papel recubierto de polvo de óxido férrico diseñado para grabación de sonido. A partir de su idea, AEG (Allgemeine Elektrizitats-Gesellschaft, Compañía General de Electricidad), inició un negocio vendiendo el magnetofón, la primera grabadora de sonido que usaba una cinta magnética. Las cintas magnéticas inicialmente eran de mala calidad, pero esto se mejoró con el esfuerzo de muchos, en particular mediante el uso de cinta plástica de acetato creada por la BASF (Badische Anilin-und Soda-Fabrik, Fábrica de anilinas y bicarbonato de sodio de Baden), y la invención de la polarización de corriente alterna. Junto con los discos de vinilo, la cinta magnética fue el medio más popular para la grabación de sonido hasta que fue desbancado por los medios ópticos como el CD. Las primeras cintas magnéticas usaban la tecnología de grabación de exploración lineal, que colocaba las pistas en paralelo al borde de la cinta magnética. En 1950 se desarrolló la exploración helicoidal que colocaba pistas de grabación en diagonal en un rango hasta el borde de la cinta, mejorando drásticamente la densidad de grabación.

En la década de 1950 la cinta magnética empezó a utilizarse para el almacenamiento de información auxiliar en computadores. El primer computador que usó cinta magnética para el almacenamiento de información fue el UNIVAC I (Universal Automatic Computer I) en 1951. Utilizaba una cinta de media pulgada con ocho pistas, de las cuales usaba 6 para datos, 1 para paridad y otra para el cronometraje, de modo que a una velocidad del carrete de 100 pulgadas por segundo produzca un rendimiento de acceso de 7200 caracteres por segundo. Estos computadores usaban un carrete de cinta de 10.5 pulgadas de diámetro, que se convirtieron en un estándar para el almacenamiento de información en grandes sistemas de computadores. Las primeras cintas magnéticas tenían un formato de carrete a carrete, los usuarios debían cargar la cinta magnética en la unidad de lectura directamente, no fue hasta 1970 que se empezaron a comercializar los cartuchos de cinta magnética. Dado que la cinta magnética se diseñó originalmente para la grabación de sonido se asume que la información es accedida de forma secuencial. El acceso aleatorio de la información requiere el rebobinado de la cinta lo que supone grandes tiempos de espera. En contraste, los primeros discos magnéticos, comercializados en la década de 1950, permitían el acceso aleatorio a la información con tiempos de espera mucho menores. Eventualmente los discos magnéticos sustituyeron a las cintas como medios de almacenamiento de información. Sin embargo, la cinta magnética todavía presentaba ventajas significativas en la relación costo-capacidad en esos días, y la cinta magnética siguió usándose para el almacenamiento de información. Un ejemplo de esto es el archivo y backup de la información. A medida que el almacenamiento de información en cintas magnéticas se hizo popular surgió el problema de la gestión de los cartuchos de cintas magnéticas. Para resolver este problema se crearon las librerías de cintas, también llamadas *robots* o *jukeboxes*, para la automatización de la custodia y la carga de los cartuchos de cinta

magnética en las unidades lectoras. Una de las primeras librerías de cintas fue la IBM 3850 Mass Storage System lanzada al mercado en 1974. Podía almacenar 9440 cartuchos de cintas magnéticas con un total de 472 GB de espacio de almacenamiento. Curiosamente, el IBM 3850 ya tenía la capacidad de gestión automática de almacenamiento jerárquico entre sus propios cartuchos de cinta magnética y los discos magnéticos IBM 3330, esto significa que automáticamente migraba los datos que tenían poco o ningún uso desde las unidades de discos magnéticos, de alto costo, a las cintas magnéticas, de bajo costo.

En la actualidad, las librerías de cintas y el almacenamiento en cintas magnéticas están cayendo en desuso debido a que las soluciones de almacenamiento modernas son cada vez más baratas y de mejor calidad. Hay que tener en cuenta que el archivado en cintas magnéticas o en librerías de cintas están diseñadas para el almacenamiento de datos sin modificaciones hasta el final de vida útil de estos, y que uno de los principales usos de las cintas magnéticas son las copias de seguridad, y en caso de un fallo de hardware, un error de software o cualquier otro problema que resulte en una pérdida de datos, se espera tener la copia de seguridad accesibles de forma rápida y sin previo aviso [1].

A1.1.3.- Discos magnéticos y arrays de discos

Los discos magnéticos son los componentes principales en los sistemas de almacenamiento moderno. El primer disco magnético fue el IBM 350 Disk File, desarrollado por en un equipo liderado por Reynold B. Johnson en 1956. El IBM 350 se incorporó al computador IBM 305 RAMAC, siglas para *Random Access Method of Accounting and Control*, lanzado por IBM en 1956. El IBM 350 Disk File podía almacenar 5 millones de caracteres de 6 bits (3.75 MB) y estaba formado por 52 discos de 24 pulgadas de diámetro recubiertos de material magnético, con lo que conseguía 100 superficies de grabación con 100 pistas por superficie, contaba con dos cabezales de acceso independientes que se movían horizontal y verticalmente para seleccionar la pista de grabación. Los discos giraban a una velocidad de 1200 rpm, impulsados por un motor de husillo, y tenía una tasa de transferencia de datos de 8800 caracteres por segundo. Con todo esto el disco medía 152 cm de ancho, 172 cm de alto y tenía un grosor de 74cm con un peso cercano a la tonelada.

A lo largo de las décadas de los 60 y 70 la capacidad de los discos magnéticos y la tasa de transferencia aumentaron, y se redujeron el tamaño, el peso y el tiempo de acceso aleatorio a los datos. En 1961 se presentó el IBM 1301 Disk storage, que contaba con 25 discos de 24 pulgadas de diámetro y 40 superficies de grabación con 250 pistas por superficie, podía almacenar 28 millones de caracteres (21MB), los discos giran a 1800 rpm y tenían una tasa de transferencia de datos de 90.000 caracteres por segundo, también contaban con brazo y un cabezal separados para cada superficie de grabación, con todos los brazos moviéndose hacia adentro y hacia afuera juntos como un gran peine, lo que redujo el tiempo de acceso a los datos con respecto a modelos anteriores. La siguiente gran evolución fue el IBM 3340

Direct Access Storage Facility, apodado Winchester. Usaba módulos de datos extraíbles que incluían el cabezal y el brazo ensamblados, y una puerta de acceso al módulo de datos que se abría o cerraba durante los procesos de carga y descarga, y fue el primero en contar con discos sellados para protegerlos del entorno. El tiempo de acceso era de 25 milisegundos y la tasa de transferencia de datos era de 885 Kb/s. De este modelo se hicieron 3 versiones de módulos extraíbles, una con 35 Mb de capacidad, otra con 70 Mb, y una tercera, también con 70Mb, pero con 500Kb accesibles con cabezales fijos para un acceso más rápido. Un aspecto significativo del IBM 3340, y la razón por la que las unidades de disco en general se conocieron como "tecnología Winchester", fue que este diseño de cabezal era de muy bajo costo y no requería que los cabezales se descargasen del soporte. La tecnología Winchester permitió que el cabezal aterrizase y despegase del soporte del disco mientras el disco giraba hacia arriba y hacia abajo. Esto resultó en ahorros muy significativos y una gran reducción de la complejidad del mecanismo de accionamiento del cabezal y el brazo. Este diseño de cabezal se convirtió rápidamente en un diseño estándar dentro de la comunidad de fabricantes de unidades de disco. Hasta principios de la década de 1990, el término Winchester se usó para las unidades de disco duro en general, pero ya no es de uso común en la mayor parte del mundo.

Los discos magnéticos en los años 70 tenían grandes platos de entre 14 y 8 pulgadas que requieren grandes armarios y fuentes de alimentación de gran capacidad, el uso de estos discos se limitaban a grandes sistemas de computadores. No fue hasta los años 80 cuando Seagate Technology presentó el primer disco duro pensado para el mercado de microcomputadores, el ST-506 tenía un disco de 5.25 pulgadas y 5 Mb de capacidad. El disco se conectaba al microcomputador mediante un controlador de disco y la interfaz SA1000 desarrollada por Seagate, que se basaba en la interfaz de los floppy disk.

En los años siguientes entraron en el mercado nuevos discos duros más pequeños y baratos, y las empresas empezaron a desarrollar unidades de almacenamiento externo, llamados arrays de discos, con varias unidades de disco pequeñas y baratas para lograr una gran cantidad de almacenamiento, un alto rendimiento y alta disponibilidad a precios mucho más bajos que los grandes discos duros.

Otros discos magnéticos que hay que tener en cuenta y que tuvieron una gran popularidad son los Floppy Disk Drives (FDD) o disquetes, que estaban compuestos por un disco delgado y flexible de plástico recubierto de un material magnético en una caja de plástico cuadrada o casi cuadrada forrada con una tela que elimina partículas de polvo del disco giratorio. Para leer los floppy disk es necesario una unidad lectora.

En la década de 1960 IBM, buscaba un sistema simple y económico para cargar microcódigo en sus mainframes System 370 en un proceso llamado Carga de Programa de Control Inicial (ICPL). Normalmente, esta tarea se realizaría con unidades de cinta que incluían casi todos los 370 sistemas, pero estas cintas eran grandes y lentas. IBM quería

algo más rápido y ligero que también pudiera enviarse a los clientes con actualizaciones de software por $5. Desde finales de los años 60 IBM experimentó con diferentes productos de almacenamiento como cartuchos de cinta o discos RCA de 45 rpm, pero finalmente se decantaron por crear su propio sistema de almacenamiento el disquete, que en un principio llamaron disco de memoria. Estos primeros disquetes salieron al mercado en 1971, eran un disco de plástico de 8 pulgadas de diámetro, 1,5 mm de grosor, recubiertos por un lado con óxido de hierro, unido a una almohadilla de espuma. Estos discos eran de solo lectura y tenían una capacidad de 80 Kb, y para leerlos se necesitaba la unidad de disco 23FD, que era una parte estándar de las unidades de procesamiento System 370 y otros productos de IBM. Esta unidad de disco usaba un cabezal magnético de lectura que se movía sobre el disco por medio de solenoides, y leían los datos pregrabados en el disco a una densidad de 1100 bits por pulgada. Estos discos tenían una "sectorización dura" con 8 agujeros alrededor del centro para marcar el inicio de los sectores de datos.

A lo largo de la década de los 70 IBM mejoró la capacidad de los discos doblando el grosor, cambiando los 8 agujeros de la sectorización dura por un único abujero de índice para "sectores blandos" o "sectores IBM" a través de 77 pistas y finalmente con la introducción de los disquetes de doble densidad, que podían almacenar datos en sus dos caras, llegando a una capacidad de 1.2Mb. Durante esta década la empresa Memorex lanzó al mercado los primeros disquetes de 8 pulgadas que permitían la lectura y escritura de datos, inicialmente tenían una capacidad de 175 Kb y a lo largo de la década llegaron a una capacidad de 800 Kb. En diciembre de 1976 Shugart Associates sacó al mercado los primeros disquetes de 5.25 pulgadas y 110 Kb de capacidad en un disco, junto con la primera unidad lectora, la SA-400. Los primeros discos de 3.5 pulgadas fueron desarrollados por Sony en 1980. Los primeros discos podían almacenar 438 Kb, después pasaron a 720K (doble densidad) y 1.44Mb (alta densidad). Durante esta década salieron varios tipos de disquetes de diferentes tamaños, que eran incompatibles entre sí, como el Hitachi de 3 pulgadas, el canon de 3.8 pulgadas y el IBM de 4 pulgadas, pero al final fue el disco de 3.5 de Sony el que se impuso y se convirtió en un estándar para el almacenamiento de datos, y durante años el tipo de almacenamiento más común en los ordenadores personales [1].

A1.1.4.- Discos ópticos y magneto-ópticos

Los discos ópticos y los discos magneto-ópticos son medios de almacenamiento que pueden registrar información cambiando las formas fotofísicas en sus superficies de grabación y leer la información registrada emitiendo rayos de luz contra la superficie y detectando su reflejo. Su origen se remonta al método de grabación de vídeo inventado por David Paul Gregg, pero su uso práctico comenzó con el primer LaserDisc, anunciado en 1980. Los discos ópticos y magneto-ópticos se convirtieron más tarde en medios muy populares para grabar contenido de audio y visual. Se han anunciado varios productos

comerciales, pero se pueden clasificar aproximadamente en cuatro grupos en términos de sus métodos de registro. La primera categoría son los discos ópticos de solo lectura, en los que los fabricantes proporcionan orificios muy pequeños, llamados bits, en una superficie de grabación para registrar información al presionar los medios. Los principales productos son CD y sus subsiguientes medios DVD y Blu-ray Disc. La segunda categoría son los discos ópticos de escritura única, lectura múltiple (WORM), en los que una unidad de disco puede registrar información solo una vez mediante la emisión de un rayo láser para quemar pigmentos irreversiblemente en una superficie de grabación de metal. Los productos principales son CD-R, DVD-R y BD-R. La tercera categoría son los discos ópticos regrabables, en los que una unidad de disco puede grabar varias veces emitiendo un rayo láser para cambiar las formas cristalinas de una superficie de grabación amorfa. CD-RW, DVD-RW / RAM y BD-RE pertenecen a esta categoría. La última categoría son los discos magneto-ópticos regrabables, en los que una unidad de disco emite un rayo láser y un campo magnético para cambiar las propiedades ópticas en una cara de grabación.

Los discos ópticos y magneto-ópticos se utilizan comúnmente para distribuir contenidos porque son fáciles de usar y pueden fabricarse a bajo costo. El CD ha sido el medio estándar para el contenido de audio y el DVD y el Blu-ray han sido el estándar para el contenido visual. Los CD y DVD también se han utilizado ampliamente para distribuir programas informáticos. Sin embargo, recientemente, las redes de banda ancha permiten cada vez más la distribución en línea. En la actualidad, los programas informáticos se distribuyen con mucha frecuencia en línea y están surgiendo varios distribuidores en línea de contenidos audiovisuales. Por otro lado, también se utilizaron discos ópticos y magneto-ópticos para almacenamiento masivo en grandes sistemas informáticos. Los proveedores desarrollaron varias máquinas de biblioteca, a veces llamadas máquinas de discos ópticos, que podían administrar varios discos ópticos y magneto-ópticos en un solo lugar. Sin embargo, la densidad de grabación de los discos ópticos y magneto-ópticos todavía está muy por detrás de la cinta magnética y su implementación en el almacenamiento masivo aún no está muy extendida. Un nuevo tipo de medio óptico que se está investigando es la memoria holográfica, que permite la grabación 3D en material cristalino. Si se desarrolla con éxito, la memoria holográfica puede mejorar significativamente la densidad de grabación en comparación con los medios de disco convencionales [1].

A1.1.5.- SCM (storage-class memory)

Los medios de almacenamiento no mecánicos, como la memoria flash, se implementan actualmente para almacenamiento secundario en sistemas informáticos. Estos medios de almacenamiento se han denominado recientemente memoria de clase de almacenamiento (SCM). Pero el uso de tales medios de almacenamiento no se limita a los sistemas recientes. Las computadoras de tubo de vacío de la década de 1950 permitían conectar

unidades de memoria externas hechas de memoria de núcleo magnético no volátil. Algunos sistemas posteriores también utilizaron memoria de burbuja magnética y memoria con respaldo de batería. Sin embargo, el uso de memoria de clase de almacenamiento solo se hizo popular para el almacenamiento secundario después de que la memoria flash, inventada por el ingeniero de Toshiba Fujio Masuoka en la década de 1980, saliese al mercado.

La memoria flash es una especie de memoria de sólo lectura programable y borrable eléctricamente (EEPROM). El controlador de memoria puede programar/borrar información colocando o quitando un electrón a/desde una celda de estado sólido hecha de una puerta flotante y leer información midiendo el voltaje de la puerta flotante. Para manipular un electrón a la puerta flotante se requiere una transferencia de electrones de túnel llamada efecto Fowler-Nordheim; a medida que se realizan más manipulaciones de electrones, una capa de óxido en la puerta flotante eventualmente se degrada. La vida útil del producto está limitada por este fenómeno. Se conocen dos opciones de diseño para conectar múltiples puertas flotantes, NOR y NAND; de estas, el diseño NAND es el más común porque mejora la densidad de integración. La información se puede programar y leer en una unidad de página (a menudo de varios kilobytes de longitud). Una vez que se ha escrito una página, no se puede volver a programar y debe borrarse con antelación. Por lo general, se permite borrar en una unidad de bloques (a menudo de cientos de kilobytes de longitud), lo que consume relativamente más tiempo que la programación y lectura. Los primeros productos implementaron solo un diseño de celda de un solo nivel (SLC), que podía registrar solo un poco en cada celda. La mejora posterior permitió un diseño de celda de varios niveles (MLC), que podía registrar más de un bit en cada celda. La memoria SLC a menudo se implementa en sistemas que requieren alto rendimiento y alta confiabilidad, mientras que MLC generalmente se encuentra en sistemas que requieren gran capacidad. A diferencia de los discos magnéticos que toman latencias de varios milisegundos incluso en productos de alta gama, la memoria flash ha reducido drásticamente las latencias entre decenas y cientos de microsegundos y mantiene un equilibrio mucho mejor entre el consumo de energía y el rendimiento del acceso.

Los primeros productos de memoria flash se abrieron camino en la electrónica de consumo, como las tarjetas SD y Compact Flash. A partir de la década de 2000, los productos de gran capacidad estuvieron disponibles en el mercado y se empezaron a utilizar como almacenamiento secundario en sistemas informáticos, aunque en estos momentos la memoria flash está por detrás de los discos magnéticos en términos de costo por capacidad. En muchos casos, la memoria flash se utiliza en unidades de estado sólido (SSD), donde se empaquetan chips de memoria flash y un circuito de control. El circuito de control tiene la capacidad emuladora de hacer que la memoria flash adjunta funcione como un disco magnético. Esta emulación permite que los sistemas informáticos utilicen SSD a través de interfaces existentes, como SATA. Los primeros productos SSD presentaban capacidades de control limitadas, pero algunos productos recientes tienen

funciones sofisticadas como nivelación de desgaste para extender la vida útil del producto al equilibrar los tiempos de programación en las celdas y la programación de escritura diferida para absorber latencias de borrado prolongadas mediante el uso de una gran memoria intermedia. En otros usos, las interconexiones existentes se diseñaron para conectar discos magnéticos, pero no siempre se optimizaron para la memoria flash. El mercado está presenciando nuevos tipos de productos de memoria flash como los SSD PCI Express o SSD NVMe, que proporcionan una latencia de acceso mucho menor, y matrices de memoria flash externas compuestas por varios chips de memoria flash y potentes circuitos de control. La memoria flash es actualmente una tecnología dominante para la memoria de clase de almacenamiento. Se está realizando una investigación activa para mejorar aún más el rendimiento, la confiabilidad y la densidad de grabación [1].

A1.1.6.- Redes de almacenamiento (Storage networking)

En los sistemas de almacenamiento convencionales cada dispositivo de almacenamiento se conecta a un ordenador por medio de una tecnología de bus, como puede ser la SCSI. En contraste, las redes de almacenamiento pueden conectar arbitrariamente ordenadores y sistemas de almacenamiento. Al estar conectados en red, los recursos de almacenamiento se comparten y consolidan más fácilmente en un solo lugar, donde se han construido una serie de funciones sofisticadas sobre la infraestructura de virtualización del almacenamiento. Algunas de estas tecnologías son:

A1.1.6.1.- Storage Area Network y Network Attached Storage (SAN y NAS)

Las Storage Area Network o redes de área de almacenamiento (SAN) y Network Attached Storage o almacenamiento conectado en red (NAS) son los dos principales tipos de arquitecturas de redes de almacenamiento.

El término SAN originalmente se refería a cualquier tipo de red diseñada para conectar dispositivos de almacenamiento, después pasó a referirse a una red de almacenamiento para proporcionar un servicio de acceso a nivel de bloque. Cuando el término SAN se usa sin modificaciones, el término a menudo se refiere al canal de fibra, desarrollado a fines de la década de 1990 para transferir cantidades masivas de datos para cálculos científicos y procesamiento de imágenes. Su diseño sofisticado permite que las tramas de canal de fibra encapsule de manera eficiente los comandos SCSI y los transfieran en una red. El canal de fibra pronto se implementó en los centros de datos para conectar dispositivos de almacenamiento y luego se reconoció como la tecnología SAN estándar desde principios de la década de 2000.

A medida que el canal de fibra se hizo común en los centros de datos, se centraron en dos cuestiones. La primera, la distancia de transmisión usando el canal de fibra era limitada y se tuvo que desarrollar tecnologías de red para unir las dos redes de canales de fibra. La segunda cuestión fue una de costo, la tecnología de canal de fibra era cara en comparación con la red ethernet. Se propusieron varias tecnologías para resolver estos problemas utilizando la tecnología IP, que podría permitir comunicaciones distantes con relativa facilidad y podría construir una infraestructura de red a costo más bajo. Las principales propuestas fueron iFCP, FCIP e iSCSI.

iFCP y FCIP eran tecnologías de puerta de enlace de red especialmente diseñadas para encapsular tramas de canal de fibra en paquetes IP, y así conectar fácilmente dos redes de canal de fibra remotas a través de una red IP. La tecnología iSCSI está basada en la tecnología IP y está diseñada para redes de almacenamiento más generales. iSCSI tiene la capacidad de encapsular comandos SCSI en paquetes IP, lo que permite la comunicación SCSI de un extremo a otro a través de una red IP, así como el puente SAN remoto.

El término NAS se puede usar para referirse al dispositivo de almacenamiento en red que proporciona un servicio de acceso a nivel de archivo, y a la red de almacenamiento que proporciona un servicio de acceso a nivel de archivos y su arquitectura de red de almacenamiento. Los principales protocolos NAS son NFS y CIFS que fueron diseñados originalmente para compartir archivos entre ordenadores en red y todavía se usan en centros de datos recientes. Similar a iSCSI, la tecnología NAS generalmente se implementa a través de una red IP.

A1.1.6.2.- Consolidación y virtualización del almacenamiento

Antes de que surgiera la tecnología de redes de almacenamiento, los recursos de almacenamiento se distribuían entre los ordenadores y se administraban por separado. Suponga que el ordenador “X” ha consumido la mayor parte de su espacio de almacenamiento, pero otro ordenador “Y” tiene mucho espacio sin usar. No era fácil para el ordenador “X” acceder al espacio de almacenamiento no utilizado del ordenador “Y”. Una vez que se han desarrollado las redes de almacenamiento, los recursos de almacenamiento se pueden compartir entre ordenadores y estos recursos se pueden asignar de manera flexible. La consolidación de los recursos de almacenamiento en una red de almacenamiento en un solo lugar mejoró la eficiencia de la gestión de los recursos de almacenamiento. Las redes de almacenamiento pronto se hicieron populares en muchos centros de datos.

La virtualización del almacenamiento es una tecnología fundamental en los entornos de almacenamiento en red, que puede crear un grupo de recursos a partir de dispositivos de almacenamiento físicos, organizar los dispositivos de almacenamiento lógicos del grupo de

recursos y proporcionar a las computadoras los dispositivos lógicos organizados. Hoy en día, muchos centros de datos implementan la virtualización del almacenamiento para permitir una gestión flexible de los recursos de almacenamiento. Los ejemplos más populares de virtualización del almacenamiento son RAID y aprovisionamiento (asignación de espacio bajo demanda) que a menudo se implementan en controladores de matriz de discos y administradores de volumen lógico (partes de sistemas operativos). Otros ejemplos son la biblioteca de cintas virtuales y los sistemas de gestión de almacenamiento jerárquico. Cuando nació la red de almacenamiento, solo había un número limitado de expertos. Storage Networking Industry Association (SNIA), una organización sin fines de lucro establecida en 1997, comenzó a promover esta tecnología y ha estado componiendo documentación técnica, promoviendo la estandarización y operando programas educativos. Sus continuos esfuerzos son responsables del amplio uso actual de las redes de almacenamiento.

A1.1.6.3.- Aplicaciones de almacenamiento sofisticadas

Las redes de almacenamiento han impulsado la tendencia tecnológica de los sistemas de almacenamiento sofisticados. El concepto de ejecutar código de aplicación dentro de una unidad de disco magnético se remonta a la investigación en máquinas de bases de datos entre las décadas de 1970 y 1980. Pero a principios de la década de 1990, los principales proveedores de bases de datos se alejaron de las máquinas especializadas y se inclinaron hacia soluciones basadas en software para máquinas de uso general. A fines de la década de 1990, una idea similar ganó la atención en el mercado. Los proveedores de almacenamiento comenzaron a incorporar funciones sofisticadas en sus productos de almacenamiento. Las redes de almacenamiento se convirtieron en una tecnología popular en muchos centros de datos, donde los recursos de almacenamiento se consolidaron y se virtualizaron más. Las políticas de diseño para administrar recursos de almacenamiento dentro de un sistema de almacenamiento se volvieron naturales y aceptables. Además, la rápida evolución de la tecnología del procesador proporcionó a los sistemas de almacenamiento una mayor potencia de procesamiento, lo que permitió tal sofisticación. Curiosamente, ideas similares se discutieron nuevamente en el mundo académico aproximadamente al mismo tiempo.

Una función popular implementada en muchos sistemas de almacenamiento fue la copia de terceros para generar directamente una copia dentro de un dispositivo de almacenamiento o entre diferentes dispositivos de almacenamiento. En los sistemas convencionales, a menudo se construía una computadora especializada, llamada servidor de respaldo, para generar y administrar copias. Pero, en muchos casos, la generación y gestión de copias se pueden desvincular de las aplicaciones que se ejecutan en las computadoras. Ejecutarlos dentro de un sistema de almacenamiento parece bastante natural y esta solución es beneficiosa para simplificar la gestión del sistema. De hecho, esto se implementa en

muchos centros de datos. Otros ejemplos que son populares en los centros de datos recientes son la copia en un momento determinado (PiT) para generar una imagen de copia coherente, la replicación remota para permitir soluciones de recuperación de desastres, el intercambio de datos entre mainframes y sistemas abiertos, WORM para soluciones de seguimiento de auditoría, deduplicación para economizar capacidad de almacenamiento y protección continua de datos (CDP) para guardar automáticamente cada versión de los datos almacenados. En la actualidad se está investigando activamente para mejorar la sofisticación de los sistemas de almacenamiento [1].

A1.1.7.- Almacenamiento en la nube y el futuro del almacenamiento de datos

Poco después del nacimiento de las redes de almacenamiento, varios proveedores, originalmente llamados proveedores de servicios de almacenamiento (SSP), comenzaron a administrar los sistemas de almacenamiento de los clientes en sus centros de datos, donde los clientes podían acceder a sus datos comerciales a través de redes de banda ancha. Desde el punto de vista de los clientes, esta tendencia se consideró con razón como subcontratación de la gestión del almacenamiento, que fue posible gracias a la tecnología de virtualización del almacenamiento emergente. Estos proveedores SSP dedicaban grandes cantidades de recursos de almacenamiento agrupados para ayudar a sus clientes a ampliar o reducir la capacidad de almacenamiento y el ancho de banda rápidamente bajo demanda. Esta agilidad ha resultado muy beneficiosa para controlar las operaciones comerciales en el dinámico mercado actual. Sin embargo, los proveedores SSP no fueron ampliamente aceptados a principios de la década de 2000, cuando comenzaron a ofrecer sus servicios. En esa época, la virtualización del almacenamiento ya era popular en muchos centros de datos, mientras que la tecnología de virtualización de servidores para virtualizar entornos de ejecución de aplicaciones se encontraba en sus primeras etapas de desarrollo. Colocar datos y aplicaciones comerciales en centros de datos remotos se convirtió en una solución realista para muchos clientes cuando ambas tecnologías de virtualización estuvieron disponibles. Posteriormente, estas soluciones se denominaron computación en la nube o cloud computing.

En contextos recientes de computación en la nube, los servicios de almacenamiento remoto que solían llamarse SSP a menudo se brindan como parte de servicios en la nube completos. Actualmente, los principales servicios de almacenamiento basados en la nube incluyen Amazon S3, Windows Azure Storage y Google Cloud Storage, todos diseñados e implementados perfectamente integrados con sus otros servicios en la nube.

El almacenamiento basado en la nube no se limita a los sistemas empresariales y se está volviendo más popular para los nuevos tipos de productos electrónicos de consumo, como los reproductores audiovisuales digitales y los lectores de libros electrónicos. Apple iCloud

y Amazon Cloud son servicios importantes que permiten a los clientes almacenar y administrar sus contenidos comprados en nubes remotas. La computación en la nube es una tecnología emergente. En el momento actual, los proveedores de servicios cloud están tratando de resolver las quejas y preocupaciones sobre problemas de rendimiento y seguridad [1].

A1.2.- HISTORIA DE LAS BASES DE DATOS

Una base de datos se puede definir como una colección autodescriptiva de registros integrados. Un registro es una representación de algún objeto físico o conceptual. Una base de datos es autodescriptiva porque contiene una descripción de su propia estructura. Esta autodescripción recibe el nombre de metadatos (o datos sobre los datos). La base de datos está integrada en el sentido de que incluye las relaciones entre elementos de datos, así como también los elementos de datos en sí mismos [3].

Tabla A1.1: Conceptos de bases de datos según paradigma [2].

Conceptos\Modelos	Red	Jerárquico	Relacional
Artículo	Tipo de elemento de datos	Elemento / campo	Nombre de rol / dominio
Valor artículo	Ocurrencia de elementos de datos	Valor	Componente
Grupo	No permitido	Grupo	No permitido
Entidad (tipo)	Tipo de registro	Tipo de entrada / segmento	Relación
Entidad (instancia)	Registro de ocurrencia	Ocurrencia de entrada / segmento	Tupla
Relación	Tipo de conjunto	Jerárquico	Clave externa comparable dominios subyacentes
Relación (instancia)	Conjunto de ocurrencias	Assembly	
Vista del administrador de datos	Estructura lógica	Estructura lógica	Modelo de datos
Definición de la vista del administrador de datos	Esquema	Esquema	Definición del modelo de datos
Vista de usuario			Submodelo de datos
Definición de la vista de usuario	Subesquema	Subesquema	Definición del submodelo de datos
Subdivisión de la base de datos	Área		
Puntos de entrada	Conjuntos singulares de registros CALC	Grupo raíz	Clave primaria
Identificadores únicos	Clave	Secuencia de segmento raíz (única)	Clave candidata

Los orígenes de la base de datos se remontan a los tiempos de las primeras bibliotecas y los registros gubernamentales, comerciales y médicos, antes de que se inventaran las computadoras. Una vez que las personas se dieron cuenta de que necesitaban los medios para almacenar datos y mantener los archivos de datos para su posterior recuperación, estaban tratando de encontrar formas de almacenar, indexar y recuperar datos. Con la aparición de las computadoras, el mundo de la base de datos cambió rápidamente, por lo que recopilar y mantener bases de datos fue una tarea cada vez más fácil, rentable y que ocupaba menos espacio.

Desde la década de 1960 con el modelo de red, hasta principios de la década de 1980, las bases de datos experimentaron grandes cambios. Durante este tiempo se desarrollaron tres modelos para organizar los datos: el modelo de red, el modelo jerárquico y el modelo relacional (ver tabla A1.1). Una vez que el modelo relacional se convirtió en el paradigma predominante, la historia de las bases de datos dio un giro brusco para encaminarse a mejorar la gestión de los datos.

A1.2.1- Década de 1960

Durante la década de los 60's los computadores ganaron popularidad para su uso por empresas privadas para aumentar su capacidad de almacenamiento de datos. Los años 60 y mediados de los 60 pueden considerarse como la nueva dirección en el campo de la base de datos [4]. La introducción del término base de datos coincidió con la disponibilidad de almacenamiento de acceso directo (discos y tambores) desde mediados de la década de 1960 en adelante. El término representó un contraste con los sistemas basados en cintas del pasado, permitiendo un uso interactivo compartido en lugar del procesamiento por lotes diario. Se desarrollaron dos modelos de datos principales: el modelo de red CODASYL (Conferencia sobre lenguaje del sistema de datos) y el modelo jerárquico IMS (Sistema de gestión de la información).

La primera generación de sistemas de bases de datos fue de navegación. Las aplicaciones generalmente acceden a los datos siguiendo los indicadores de un registro a otro. Los detalles de almacenamiento dependían del tipo de datos que se almacenarán. Por lo tanto, agregar un campo adicional a la base de datos requería reescribir el esquema de acceso / modificación subyacente. Se hizo hincapié en los registros a procesar, no en la estructura general del sistema. Un usuario necesitaría conocer la estructura física de la base de datos para poder consultar información. Una base de datos que resultó ser un éxito comercial fue el sistema SABRE que fue utilizado por IBM para ayudar a American Airlines a administrar sus datos de reservas [5] (los principales servicios de viajes todavía utilizan este sistema para sus reservas).

A1.2.2- Década de 1970

Al inicio de la década de los 70's el modelo de base de datos relacional fue concebido por E.F. Codd. Este modelo se apartó de la tradición al insistir en que las aplicaciones deberían buscar datos por contenido en lugar de seguir enlaces. Su sistema se puede definir utilizando dos terminologías [6]. Instancia, una tabla con filas y columnas, y esquema, especifica la estructura, incluido el nombre de la relación, el nombre y el tipo de cada columna.

El modelo de Codd se basó en ramas de las matemáticas denominadas teoría de conjuntos y lógica de predicados. Sus ideas cambiaron la forma en que la gente pensaba sobre las bases de datos. En su modelo, el esquema de la base de datos, u organización lógica, está desconectado del almacenamiento de información física, y esto se convirtió en el principio estándar para los sistemas de bases de datos. Durante el período 1974 a 1977, se crearon dos prototipos principales de sistemas de bases de datos relacionales:

1. INGRES, que se desarrolló en la Universidad de California-Berkeley y se convirtió en comercial y dio seguimiento a POSTGRES que se incorporó a Informix. Esto finalmente llevó a Ingres Corp., Sybase, MS SQL Server, Britton-Lee y Wang's PACE. Este sistema utilizó QUEL como lenguaje de consulta.

2. System R, que se desarrolló en IBM en San José y dirigió SQL / DS & DB2 de IBM, ORACLE, Allbase de HP y SQL Non-Stop de Tandem. DB2 se convirtió en uno de los primeros productos DBMS (Database Management System) basado en el modelo relacional [8].

En 1976 un nuevo modelo de base de datos llamado Entidad-Relación, o ER, fue propuesto por P. Chen. Este modelo hizo posible que los diseñadores pudieran centrarse en la aplicación de datos en lugar de la estructura lógica de la tabla [6]. El término Sistema de gestión de bases de datos relacionales (RDBMS) se acuñó durante este período.

A1.2.3- Década de 1980

En la década de los 80's la comercialización de sistemas relacionales comienza cuando un auge en la compra de computadores impulsa el mercado de bases de datos (DB) para los negocios. SQL (Structured Query Language) se convirtió en el estándar [7]. DB2 se convirtió en el principal producto de IBM. Los modelos jerárquicos y de red perdieron su encanto sin un mayor desarrollo de estos sistemas, pero algunos sistemas heredados todavía se utilizan hoy en día. El desarrollo del ordenador personal de IBM dio lugar a muchas empresas y productos de bases de datos, como RIM, RBASE 5000, PARADOX,

OS / 2 Database Manager, Dbase III, IV (más tarde FoxBASE y Visual FoxPRO) y Watcom SQL. Todos estos sistemas se introdujeron durante la década de 1980 y se basaron en el modelo relacional.

Durante este período, se desarrolló el concepto de base de datos orientada a objetos. El término 'sistema de base de datos orientado a objetos' apareció por primera vez alrededor de 1985. Una base de datos de objetos, también sistema de gestión de bases de datos orientado a objetos, es aquella en la que la información se representa en forma de objetos tal como se utiliza en la programación orientada a objetos [9]. Las bases de datos de objetos son diferentes de las bases de datos relacionales y pertenecen juntas al sistema de administración de bases de datos más amplio. El uso principal de la base de datos de objetos es en áreas orientadas a objetos. Algunas bases de datos orientadas a objetos están diseñadas para funcionar bien con lenguajes de programación orientados a objetos, como Delphi, Ruby, Python, Perl, Java, C#, Visual Basic, .NET, C++, Objective-C y Smalltalk. Otros tienen sus propios lenguajes de programación.

OODBMS (Sistema de gestión de bases de datos orientado a objetos) utiliza exactamente el mismo modelo que los lenguajes de programación orientados a objetos. OODBMS admite el modelado y la creación de datos como objetos. OODBMS podría administrar de manera eficiente una gran cantidad de tipos de datos diferentes. Los objetos con comportamientos complejos eran fáciles de manejar mediante la herencia y el polimorfismo. Esto también ayudó a reducir la gran cantidad de relaciones al crear objetos. El mayor problema con OODBMS fue cambiar una base de datos existente a OODBMS, ya que la transición requiere un cambio completo desde cero y generalmente está vinculada a un lenguaje de programación específico y una API (Interfaz de programación de aplicaciones) que reduce la flexibilidad de la base de datos. Para superar los problemas de OODBMS y aprovechar al máximo el modelo relacional y el modelo orientado a objetos, el modelo de base de datos relacional de objetos se desarrolló a principios de la década de 1990.

A1.2.4- Década de 1990

A principios de la década de 1990, una reorganización de la industria resultó en que menos empresas sobrevivientes ofrecieran productos cada vez más complejos a precios más altos. Gran parte del desarrollo durante este período se centró en herramientas cliente para el desarrollo de aplicaciones, como PowerBuilder (Sybase), Oracle Developer y VB (Microsoft). El modelo cliente-servidor para la informática se convirtió en la norma para futuras decisiones comerciales [4]. Este período de tiempo también marcó el desarrollo de herramientas de productividad personal, como Excel / Access (MS) y ODBC (Open Database Connectivity).

La gran inversión en empresas de Internet impulsó el auge del mercado de herramientas para conectores Web / Internet / DB. Active Server Pages (ASP), Front Page, Java Servlets, JDBC, Enterprise Java Beans, ColdFusion, Dream Weaver y Oracle Developer 2000 fueron ejemplos de tales ofertas. Las soluciones de código abierto se pusieron en línea con el uso generalizado de GCC (GNU Complier Collection), Apache y MySQL. El procesamiento de transacciones en línea (OLTP) y el procesamiento analítico en línea (OLAP) maduraron con muchos comerciantes que utilizan la tecnología de punto de venta (POS) a diario [4].

Como se mencionó anteriormente, para superar los problemas de OODBMS y explorar completamente las posibilidades del modelo relacional, ORDBMS (Object Relational Database Management System) también surgió durante la década de 1990. El objetivo básico de la base de datos relacional de objetos era cerrar la brecha entre las bases de datos relacionales y las técnicas de modelado orientadas a objetos utilizadas en lenguajes de programación, como Java, C++, Visual Basic .NET o C#. Sin embargo, una alternativa más popular para lograr dicho puente es utilizar sistemas de bases de datos relacionales estándar con algún tipo de software de mapeo relacional de objetos (ORM) [8]. Mientras que los productos RDBMS o SQLDBMS tradicionales se centraban en la gestión eficiente de datos extraídos de un conjunto limitado de tipos de datos (definidos por los estándares de lenguaje pertinentes), un DBMS relacional de objetos permitía a los desarrolladores de software integrar sus propios tipos y los métodos que se aplican a ellos en el DBMS.

Otro desarrollo importante que tuvo lugar durante 1997 fue la introducción del Lenguaje de marcado extensible (XML) [4]. XML es un lenguaje de marcado que define un conjunto de reglas para codificar documentos en un formato que es legible por humanos y legible por máquina. Está definido en la Especificación XML 1.0 producida por el W3C - y varias otras especificaciones relacionadas - todos estándares abiertos gratuitos. Los objetivos de diseño de XML enfatizan la simplicidad, la generalidad y la facilidad de uso en Internet. Es un formato de datos textuales con un fuerte soporte a través de Unicode para los idiomas del mundo. Aunque el diseño de XML se centra en documentos, se usa ampliamente para la representación de estructuras de datos arbitrarias, como servicios web. Se han desarrollado muchas interfaces de programación de aplicaciones (API) para que los desarrolladores de software las utilicen para procesar datos XML, y existen varios sistemas de esquemas XML para ayudar en la definición de lenguajes basados en XML. Los principales proveedores comienzan a integrar XML en productos DBMS.

A1.2.5- Principios del siglo XXI

El año 2000, el miedo al Y2K, fue la razón principal de los rápidos cambios tecnológicos que tuvieron lugar a finales de los noventa. Una vez que llegó la década de 2000, hubo una fuerte caída en la industria de Internet en su conjunto, pero continuó el sólido crecimiento

de las aplicaciones de bases de datos. Aparecieron aplicaciones más interactivas con el uso de PDA (asistentes de datos personales), transacciones de punto de venta y consolidación de proveedores. Tres empresas principales dominaban la mayor cuota de mercado de bases de datos: IBM, Microsoft y Oracle.

Las bases de datos hacen un buen trabajo almacenando datos y contando datos, pero a menudo dependen de la interacción humana o de un programa externo para hacer correlaciones y tomar decisiones basadas en los datos. El paso en esta dirección comenzó a principios del siglo XXI y aún está en su infancia. Actualmente existe una tendencia creciente a proporcionar una lógica de programación más sofisticada dentro de la estructura de la base de datos [10]. Inicialmente, las bases de datos solamente controlaban qué tipo de datos podían colocarse en un campo en particular. Por ejemplo, si el campo definido es para ingresar la fecha / hora, desencadenaría un mensaje de error si se inserta algún otro valor en el campo. Las bases de datos se han vuelto más sofisticadas y tienen funciones como activadores, actualización en cascada y eliminación. Esto evitó que los actualizadores crearan inconsistencias entre las tablas. Las bases de datos también desarrollaron un lenguaje de procedimiento simplificado que contenía SQL incorporado con algunas estructuras de control y bucles; por ejemplo, Transact SQL de los servidores Sybase / SQL, PL / SQL de Oracle y PostgresSQL.

También están apareciendo enormes sistemas de terabytes que requerirán medios novedosos para manejar y analizar datos, como grandes bases de datos científicas como el proyecto del genoma, los datos geológicos, de seguridad nacional y de exploración espacial. En esta época, la minería de datos, el almacenamiento de datos y los mercados de datos son técnicas de uso común.

A1.2.6- NoSQL

El término NoSQL se utilizó por primera vez en 1998 por Carlo Strozzi para denominar a su base de datos relacional liviana y de código abierto que no disponía de una interfaz SQL estándar. Eric Evans, un empleado de la empresa de computación en la nube Rackspace, reintrodujo el término NoSQL durante la preparación de un encuentro para tratar sobre bases de datos distribuidas de código abierto [13]. Este término intentaba etiquetar a ciertos tipos de almacenes de datos de reciente aparición, que presentaban cualidades más características no relacionales, y que no siempre proporcionaban garantías ACID (atomicidad, consistencia, aislamiento, durabilidad), que son los atributos clave de los sistemas de bases de datos relacionales clásicos, como como Sybase, IBM DB2, MySQL, Microsoft SQL Server, Postgres SQL, Oracle RDBMS, Informix y Oracle RDB.

A menudo, las bases de datos NoSQL se clasifican de acuerdo con la forma en que almacenan los datos y se clasifican en categorías como bases de datos clave/valor,

implementaciones de BigTable o columnares, bases de datos documentales y bases de datos de grafos. Los sistemas de bases de datos NoSQL crecieron junto con las principales empresas de Internet, como Google, Amazon, Twitter y Facebook, que tenían desafíos significativamente diferentes al tratar con datos que las soluciones RDBMS tradicionales no podían hacer frente. Las bases de datos NoSQL suelen estar altamente optimizadas para operaciones de recuperación y anexión y, a menudo, ofrecen poca funcionalidad más allá del almacenamiento de registros. La flexibilidad del tiempo de ejecución reducido, en comparación con los sistemas SQL completos, se compensa con ganancias significativas en escalabilidad y rendimiento para ciertos modelos de datos.

En 2011, los desarrolladores de bases de datos comenzaron a trabajar en UnQL (lenguaje de consulta no estructurado), una especificación para un lenguaje de consulta para bases de datos NoSQL. Está diseñado para consultar colecciones de documentos con campos poco definidos frente a tablas con filas y columnas [8]. Se afirma que UnQL es un superconjunto de SQL dentro del cual SQL es un tipo muy restringido de UnQL para el cual las consultas siempre devuelven los mismos campos, el mismo número, nombres y tipos. Sin embargo, UnQL no cubre las sentencias SQL del lenguaje de definición de datos (DDL) como CREATE TABLE o CREATE INDEX.

Anexo 2 Instalación y configuración de componentes necesarios

A2.1.- MÁQUINA VIRTUAL UBUNTU

Para montar las máquinas virtuales usaremos Hyper-V y una imagen de la última versión de Ubuntu, que se puede descargar aquí: https://ubuntu.com/download/desktop.

Hyper-V es una característica de Windows 10 Pro, que se puede activar desde el Panel de Control, en la pestaña de "Activar o desactivar las características de Windows", marcando la pestaña de Hyper-V y dando a "Aceptar", como se puede ver en la figura A2.1.

Las configuraciones de las máquinas virtuales que usaremos serán 6 procesadores virtuales, 8 Gb de RAM y un disco duro virtual con un máximo de 20 Gb, como vemos en la figura A2.2.

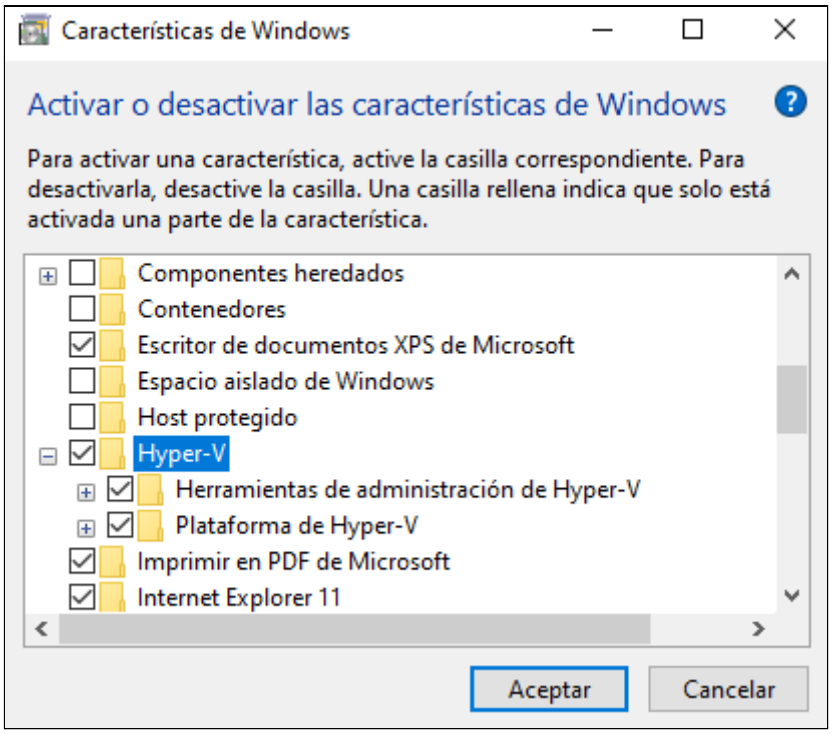


Figura A2.1.- Activar característica Hyper-V.

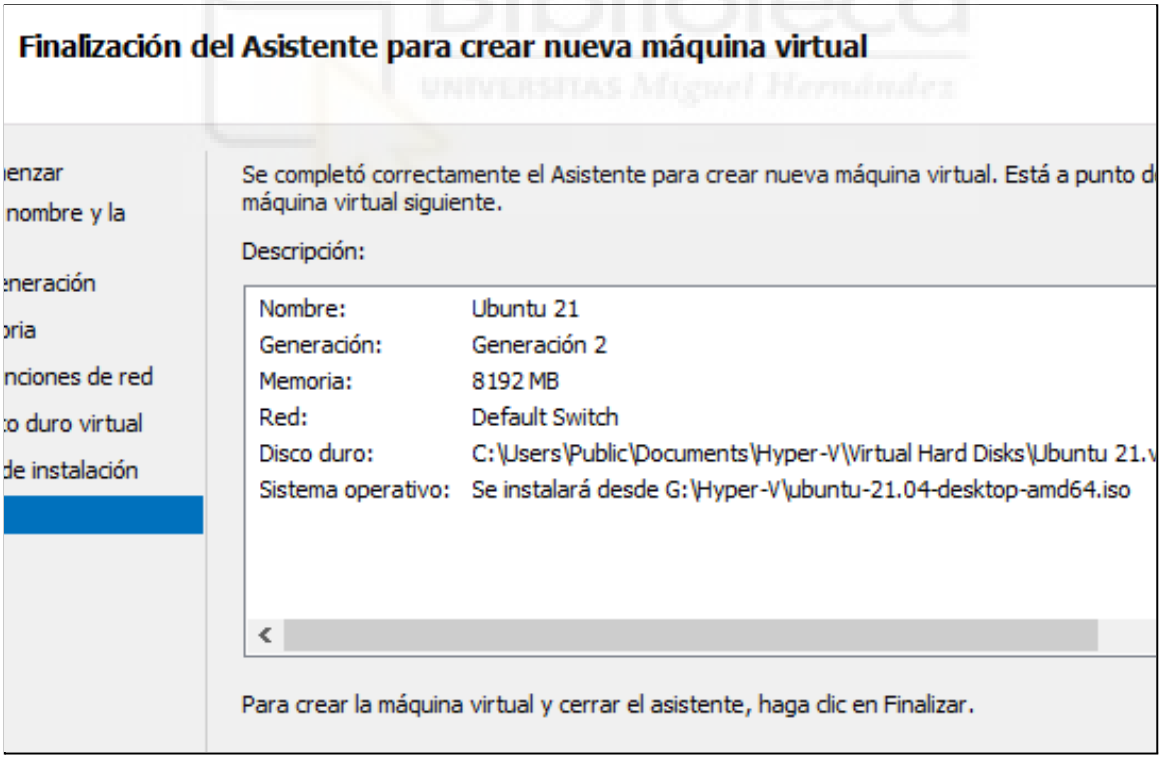


Figura A2.2.- Configuración de las máquinas virtuales

Una vez configuradas las máquinas virtuales la iniciaremos e instalaremos la versión de Ubuntu descargada.

A2.2.- XAMPP PARA LINUX

XAMPP es un software disponible en https://www.apachefriends.org, que permite instalar una infraestructura LAMP (Linux, Apache, MariaDB y PHP) para desarrollar y/o instalar aplicaciones web. En la figura A2.3 se muestra la página de inicio de la web y, marcado en rojo, el link sobre el que hay que clicar para descargarse la versión para Linux, que es la que se ha utilizado en este proyecto.

Figura A2.3.- Descarga de XAMPP para linux

A continuación se indican los pasos que hay que seguir para instalar y ejecutar este paquete en nuestro equipo:

1) Una vez descargado el instalador le damos permisos de ejecución e instalamos.

```
cd ~/Descargas/
chmod 755 xampp-linux-x64-8.0.6-0-installer.run
sudo ./xampp-linux-x64-8.0.6-0-installer.run
```

2) Tras la instalación se ejecuta XAMPP.

```
sudo /opt/lampp/lampp start
```

3) Ahora ya se puede abrir en el navegador la página inicial.

```
http://localhost/dashboard/
```

Si aparece la página de bienvenida de XAMPP, como se muestra en la figura A2.4, se ha instalado correctamente y está en ejecución.

Figura A2.4.- Página de bienvenida de XAMPP

A2.3.- WORDPRESS

El gestor de contenidos WordPress debe instalarse sobre una arquitectura LAMP o WAMP (en nuestro caso, XAMPP, apartado anterior) que esté previamente instalada y en ejecución. Una vez satisfecho este prerrequisito, los pasos para instalar y configurar WordPress serían los siguientes:

1) Entrar en la aplicación de phpMyAdmin (incluida con la instalación de XAMPP), para crear una base de datos y un usuario para WordPress.

```
http://localhost/phpmyadmin/
```

2) Ejecutar el siguiente script SQL (en la pestaña SQL de phpMyAdmin) para crear la base de datos y el usuario:

```
CREATE DATABASE wordpress;
CREATE USER 'redis'@'localhost' IDENTIFIED BY 'admin123';
GRANT ALL PRIVILEGES ON wordpress.* TO 'redis'@'localhost' WITH
GRANT OPTION;
FLUSH PRIVILEGES;
```

Si la base de datos y el usuario se han creado correctamente, phpMyAdmin mostrará la nueva base de datos vacía con el nombre de “wordpress”. Seleccionando dicha base de datos y clicando en la pestaña de “Privilegios” también se mostrará el usuario “redis” que se ha creado para la base de datos (ver figura A2.5).

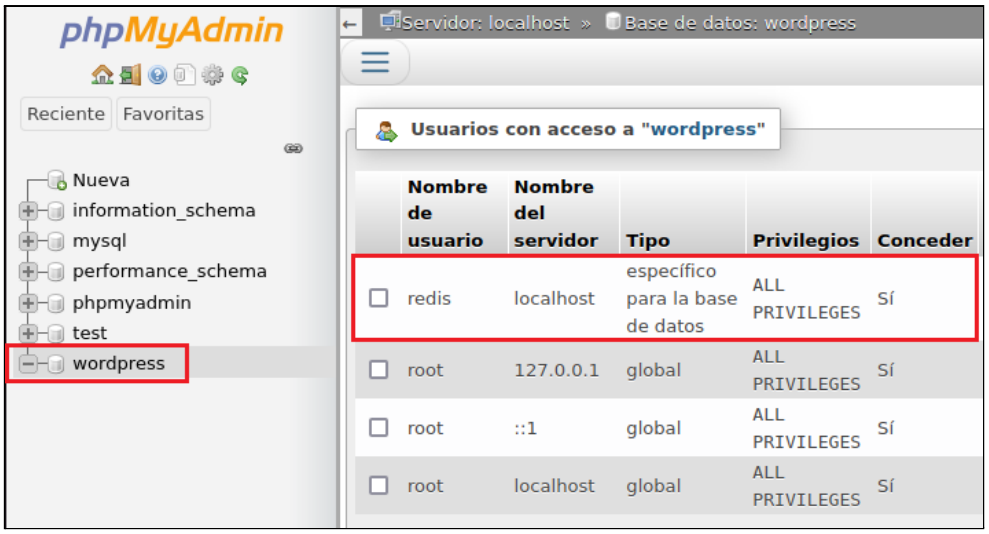


Figura A2.5.- Base de datos y usuario para WordPress

3) Descargar (con el comando "curl") la última versión de WordPress en una carpeta temporal y descomprimir los ficheros, esta acción crea la carpeta "wordpress" con casi todos los archivos necesarios para la instalación.

```
cd /tmp
curl -O https://wordpress.org/latest.tar.gz
tar xzvf latest.tar.gz
```

4) Crear el archivo de configuración de WordPress, copiando el que ya existe de muestra, con el nombre "wp-config.php".

```
cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php
```

5) Modificar el archivo de configuración (con cualquier editor de texto) para añadir los datos de la base de datos, y una función para poder instalar complementos de WordPress directamente desde un fichero.

```
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );
/** MySQL database username */
define( 'DB_USER', 'redis' );
/** MySQL database password */
define( 'DB_PASSWORD', 'admin123' );

...

/* Para instalar paquetes sin necesidad de configurar un FTP */
define( 'FS_METHOD', 'direct' );
```

6) Copiar la carpeta “wordpress” de la carpeta temporal a la carpeta de XAMPP, y dar permisos para que pueda escribir y modificar archivos.

```
sudo cp -a /tmp/wordpress/ /opt/lampp/htdocs/
sudo chmod -R 777 /opt/lampp/htdocs/wordpress/
```

7) Ejecutar el asistente de instalación y configuración de WordPress desde el navegador, en la siguiente dirección.

```
http://localhost/wordpress/wp-admin/install.php
```

El asistente solicitará del usuario la siguiente información:

- Idioma por defecto
- Nombre para la web
- Usuario y contraseña para acceder WordPress como administrador
- Correo electrónico.

8) Ya se puede acceder al gestor de contenidos WordPress en la siguiente dirección, ver figura A2.6.

```
http://localhost/wordpress/wp-login.php
```

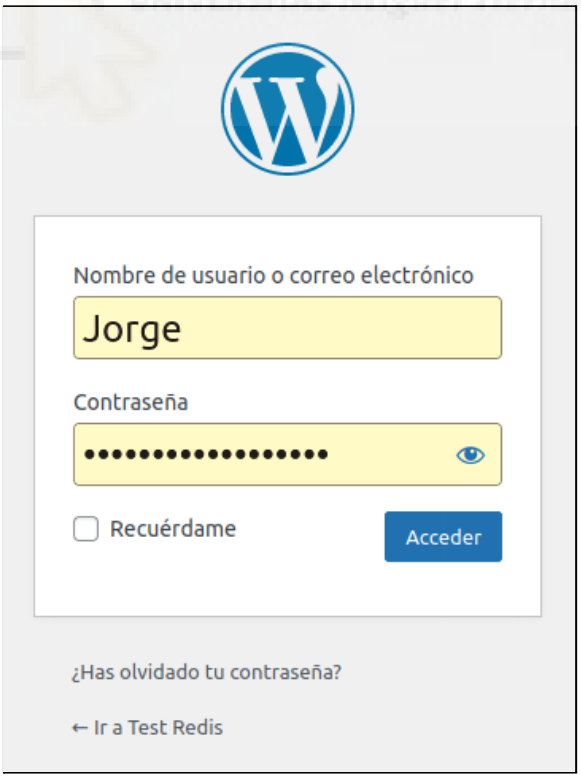


Figura A2.6.- Página de identificación de usuario de WordPress

A2.4.- LOAD TIME

Load Time es una extensión de Firefox que permite calcular los tiempos de carga de una página web, su instalación es muy sencilla, basta con acceder a la siguiente URL con el navegador Firefox y pulsar el botón “Añadir a Firefox”.

```
https://addons.mozilla.org/es/firefox/addon/load-time/
```

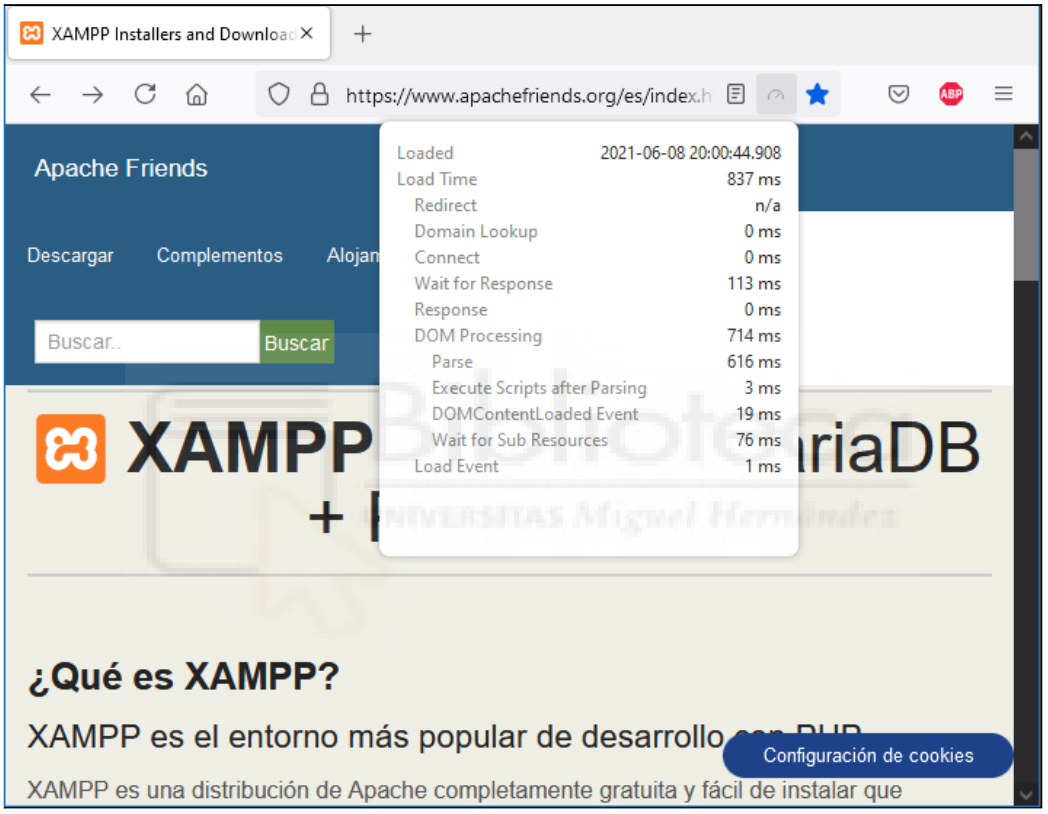


Figura A2.7.- Ejemplo de utilización de Load Time

A2.5.- NODE-RED

Node-RED es una herramienta de programación visual basada en flujo, desarrollada originalmente por el equipo de Servicios de tecnología emergente de IBM y que ahora forma parte de JS Foundation. Node-RED está basado en Node.js, y podemos acceder a él mediante el navegador web.

1) Para poder instalar Node-Red primero tenemos que instalar Node.JS, para eso instalamos curl si no lo tenemos instalado, después descargamos de nodesource y lo ejecutamos y tras eso lo instalamos.

```
sudo apt  install curl
curl -fsSL https://deb.nodesource.com/setup_current.x | sudo -E bash -
sudo apt-get install -y nodejs
```

2) Tras instalar Node.JS se instala Node-Red con el siguiente comando [26].

```
sudo npm install -g --unsafe-perm node-red
```

3) Una vez instalado, ya se puede ejecutar y acceder por el navegador.

```
sudo node-red
http://127.0.0.1:1880
```

4) Adicionalmente, para el caso de uso de este TFG se necesita el paquete de MongoDB para node red Node-Red, se instala con el siguiente comando.

```
npm install node-red-node-mongodb
```

A2.5.1.- Configuración de los Nodos

A continuación se van a describir los diferentes nodos que se ven en la figura 4.13. de la memoria. El nodo "Start" es un nodo de inyección que funciona como disparador.

Figura A2.8.- Nodo inyección de Node-Red.

Los nodos de función permiten insertar código javascript. A continuación veremos el código de los nodos, que es similar para las 3 estaciones, solo cambia el valor del tópico en las estaciones.

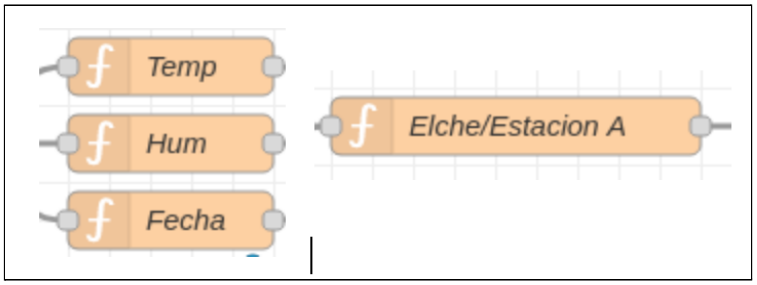


Figura A2.9.- Nodos de función de Node-Red.

1) Código del nodo de temperatura, crea un valor aleatorio entre 0 y 31 para la temperatura.

```
msg.topic="Temp";
temp=Math.round(Math.random()*30)+1;
msg.payload= temp;
return msg;
```

2) Código del nodo de humedad, crea un valor aleatorio entre 0 y 31 para la humedad.

```
msg.topic="Hum";
hum=Math.round(Math.random()*30)+1;
msg.payload=hum;
return msg;
```

3) Código del nodo de fecha, inserta la fecha actual con el formato ISO 8601.

```
msg.topic="fecha";
msg.payload = new Date().toISOString();
return msg;
```

4) Código de las estaciones, recibe los valores de temperatura, humedad y fecha, y crea un documento JSON.

```
context.data = context.data || new Object();
topico = "Elche/EstacionA";
context.data.topic = topico;

switch (msg.topic) {
    case "Temp":
        context.data.Temperatura = msg.payload;
        msg = null;
        break;
    case "Hum":
        context.data.Humedad = msg.payload;
        msg = null;
        break;
    case "fecha":
        context.data.Fecha = msg.payload;
        msg = null;
        break;
    default:
        msg = null;
        break;
}
```

```
if(context.data.Temperatura != null &&
   context.data.Humedad != null &&
   context.data.Fecha != null) {
    msg2 = new Object();
    msg2 = context.data;
    context.data=null;
    msg2.topic = topico;
    return msg2;
}
else return msg;
```

5) Los nodos de envío y recepción de MQTT se configuran con los datos del broker que usamos, en nuestro caso el Eclipse mosquitto.

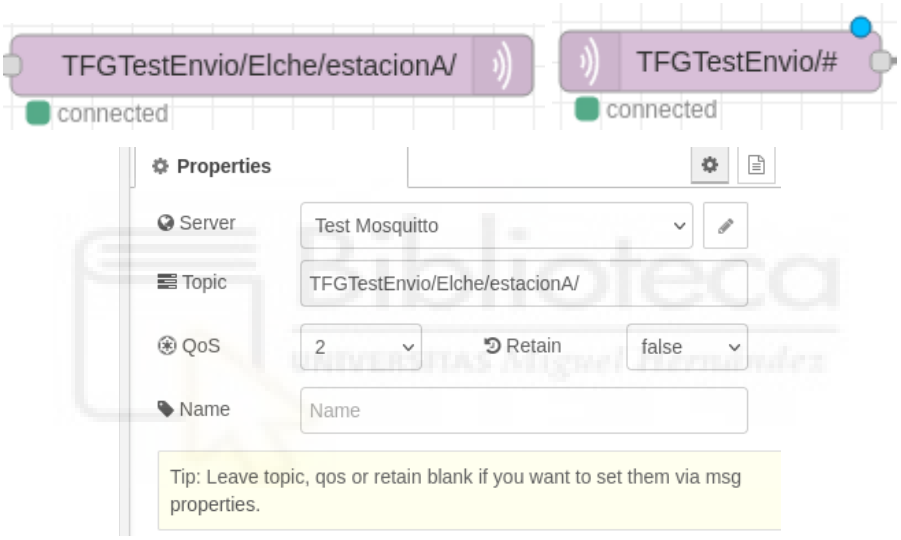


Figura A2.10.- Nodos de envío y recepción MQTT y su configuración.

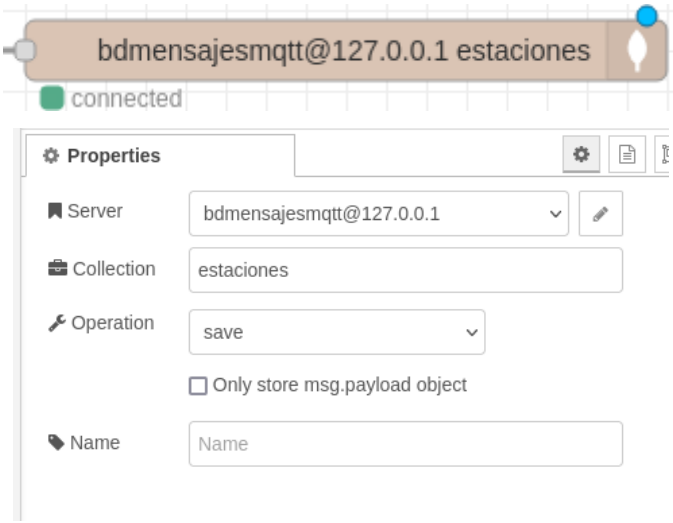


Figura A2.11.- Nodo de conexión con MongoDB y su configuración.

A2.6.- ECLIPSE MOSQUITTO

Eclipse Mosquitto es un broker MQTT de código abierto (con licencia EPL / EDL). El protocolo MQTT proporciona un método ligero para enviar y recibir mensajes mediante un modelo de publicación / suscripción. Esto lo hace adecuado para la mensajería IoT (Internet de las Cosas).

1) Eclipse Mosquitto se instala ejecutamos el siguiente comando:

```
sudo apt update -y && sudo apt install mosquitto mosquitto-clients -y
```

2) Una vez instalado habilitamos la ejecución al inicio.

```
sudo systemctl enable mosquitto
```

A2.7.- METABASE

Metabase es una herramienta de código abierto y fácil instalación que permite realizar gráficos detallados sobre los datos proporcionados.

1) En MongoDB, por línea de comando, se crea un usuario para Metabase

```
mongo
db.createUser(
  {
    user: "Metabase",
    pwd: "admin123",
    roles: [ { role: "readAnyDatabase", db: "admin" } ]
  }
)
```

2) Después se accede a la web de metabase y se descarga el archivo .jar con Metabase, y se ejecuta con el siguiente comando. Una vez instalado accedemos a la página de configuración en el navegador.

```
https://www.metabase.com/start/oss/jar.html
sudo apt install default-jre
http://localhost:3000/setup/
```

En la página de configuración rellenamos los datos y conectamos con la base de datos de MongoDB. Una vez finalizada la configuración, como vemos en la figura A2.8, ya podemos crear gráficos con los datos de la base de datos de MongoDB.

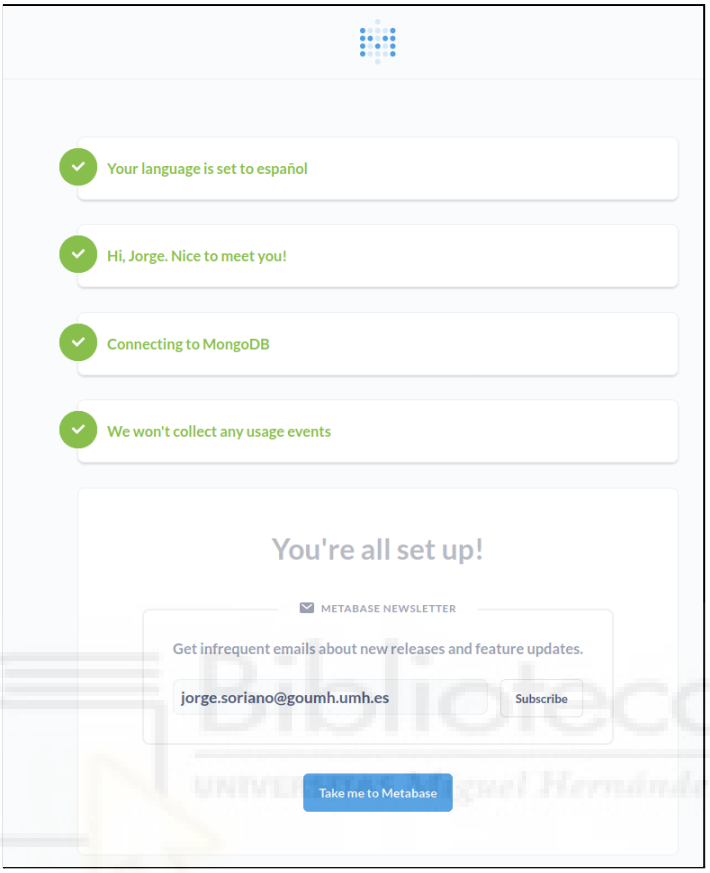


Figura A2.8.- Configuración de Metabase