Universidad Zaragoza
1542

Trabajo Fin de Grado

Metodologías y herramientas para el desarrollo de un data warehouse

Data warehouse development: methodologies and tools

Autor

Martín Villuendas del Río

Directores

Dra. Dª Piedad Garrido Picazo

Dº Pablo Doñate Navarro

ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL

NOVIEMBRE 2024

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

RESUMEN

Este Trabajo de Fin de Grado (TFG) se centra en encontrar las metodologías y las herramientas más óptimas para implementar un almacén de datos (en inglés, Data Warehouse (DW)) y estudiar si es factible su uso para que la Industria 4.0 pueda tratar grandes volúmenes de información, en tiempo real, en proyectos reales.

En una primera fase, se realizará el estudio de las diversas tecnologías que se pueden usar para diseñar un almacén de datos, seleccionando las más óptimas para trabajar con grandes volúmenes de datos.

Una vez seleccionadas, se desarrollará en cada una de las tecnologías el almacén de datos correspondiente.

Por último, se realizará una batería de pruebas, con el objetivo de recopilar los resultados y analizar cuál de las tecnologías seleccionadas es la más óptima para su uso con grandes cantidades de información.

Palabras clave

Almacen de datos, NoSQL, SGBD, OLAP, ETL

Abstract

This Final Degree Project (TFG) focuses on finding the most optimal methodologies and tools to implement a Data Warehouse (DW) and studying the feasibility of its use for Industry 4.0 to handle large volumes of information in real-time, in real projects.

In the first phase, a study of the various technologies that can be used to create a Data Warehouse will be conducted, selecting the most optimal ones for working with large volumes of data.

Once selected, the corresponding Data Warehouse will be developed using each of the selected technologies.

Finally, a series of tests will be carried out with the aim of collecting results and observing which of the selected technologies is the most optimal for use with large volumes of data.

Keywords

Data Warehouse, NoSQL, DBMS, OLAP, ETL

Capítulo 1

Introducción y Objetivos

La Industria 4.0 representa la cuarta revolución industrial, caracterizada por la convergencia de tecnologías digitales avanzadas y procesos industriales. Esta revolución se fundamenta en la integración de tecnologías como la Internet de las Cosas (en inglés, Internet Of Things (IoT)), la Inteligencia Artificial (IA), el Big Data, la Robótica Avanzada y la Computación en la Nube. Estas innovaciones permiten la creación de fábricas inteligentes donde los sistemas físicos y digitales cooperan de manera autónoma para optimizar la producción y los procesos industriales.

Uno de los elementos centrales de la Industria 4.0 es la capacidad de manejar y analizar grandes volúmenes de datos en tiempo real. La enorme cantidad de datos generados por los dispositivos y sensores IoT, junto con la necesidad de procesar esta información rápidamente, hace que las empresas requieran soluciones avanzadas para el almacenamiento y la gestión de datos. En este contexto, los almacenes de datos (en inglés, Data Warehouses (DW)) se convierten en una herramienta esencial. Un DW permite consolidar datos de diversas fuentes, facilitando el análisis y la futura toma de decisiones.

Este Trabajo de Fin de Grado (TFG) se centra en la implementación de un almacén de datos para la gestión de un gran número de eventos de carga de vehículos eléctricos. La idea surgió de la necesidad de gestionar los datos de la empresa V2C [1], que se dedica a la fabricación y comercialización de cargadores para vehículos eléctricos. V2C maneja una gran cantidad de datos generados por los eventos de carga de sus dispositivos, lo que plantea el desafío de implementar una solución eficiente para el almacenamiento y análisis de estos datos.

El objetivo principal de este TFG es encontrar las metodologías y herramientas más

[1] Página oficial V2C: `https://v2charge.com/es/`

óptimas para implementar un almacén de datos que pueda manejar grandes volúmenes de información en tiempo real, y evaluar la viabilidad de su uso en proyectos reales dentro del marco de la Industria 4.0. Además, de una serie de objetivos específicos:

- Profundizar y entender de manera más amplia el concepto de DW.
- Desarrollar de manera completa y en distintas tecnologías un DW con un gran volumen de datos.
- Conocer nuevas tecnologías y profundizar en el funcionamiento de éstas.
- Realización de pruebas en cada una de las distintas tecnologías y sacar diferencia de rendimiento para probar cual de todas es la más óptima.

Este TFG comienza con una introducción, donde se explica y se hace un repaso de la utilidad y el uso de los Data Warehouse, además de una serie de objetivos principales de este trabajo. Seguidamente se realiza un estado del arte, tratando tanto las tecnologías y las arquitecturas utilizadas como propuestas que se asemejan al trabajo planteado. Como tercer apartado una contextualización, donde se explican los conceptos más importantes de los DW, las arquitecturas existentes y los tipos de tecnologías que se encuentran en el trabajo. Posteriormente se explica el origen y las características de los datos usados, además de las tecnologías que se han utilizado. Por último la implementación de los DW en cada una de las tecnologías, una serie de pruebas, comparaciones y una conclusión.

Destacar que este TFG surge por el trabajo que se está realizando en la Cátedra V2C - Smart Energy de la UZ (Universidad de Zaragoza). Esta cátedra tiene como objetivo fomentar la investigación y estudio en temas relacionados con la gestión inteligente y sostenible de la energía, especialmente para el caso de los hogares y los vehículos eléctricos.

Capítulo 2

Estado del Arte

El tema principal de este Trabajo Fin de Grado (TFG) es la comparación de distintas tecnologías y metodologías que se pueden usar a la hora de implementar un Data Warehouse (DW), con el objetivo de trabajar de la manera más óptima con un gran volumen de datos. Existen varios proyectos que hablan sobre las distintas tecnologías, metodologías e incluso arquitecturas que se pueden utilizar a la hora de diseñar un Data Warehouse. En concreto, se han seleccionado los siguientes cuatro trabajos académicos similares o del mismo campo de estudio que este TFG obtenidos en repositorios oficiales de investigación como son IEEXplore, Google Schoolar, etc.

El primer artículo que se ha encontrado trata del *"Efficient olap query processing across cuboids in distributed data warehousing environmen"* de Santanu Roy, Saikat Raj, Tamal Chakraborty, Anirban Chakrabarty, Agostino Cortesi y Soumya Sen [3] que detalla la eficiencia a la hora de realizar querys mediante cuboids en DW distribuidos y siguiendo la arquitectura de procesamiento analítico en línea (en inglés, Online Analytical Processing (OLAP)). La distribución de los datos se ha hecho a través de varios centros de datos, con la idea de asegurar el procesamiento paralelo con el almacén subyacente y mejorar el rendimiento del sistema. Las pruebas realizadas, con conjuntos reales de datos, muestran la eficacia del método desarrollado, viéndose una reducción del tiempo de ejecución de consultas OLAP, una mejora de la eficiencia del espacio, entre otras muchas mejoras.

Otro de los proyectos seleccionados realizado en la Universidad Oberta de Cataluña (UOC) tiene por título: *"Construcción y explotación de un data warehouse para el análisis de información sobre el tránsito rodado de vehículos"*, de Rubén Sánchez Ginés [4]. Este TFG trata de crear un conjunto de componentes para gestionar los flujos de datos de una empresa, donde se analizará y explotará un Data Warehouse mediante distintos procesos y tecnologías: Pentaho, Microsoft Analysis Services y MySQL. Destacar que este trabajo se centra en la recopilación de información sobre el

tránsito de vehículos, lo que se ajusta bastante al proyecto que se quiere llevar a cabo.

"Evaluating nosql databases for olap workloads: A benchmarking study of MongoDB, Redis, Kudu and ArangoDB" se trata de un artículo realizado por Rishi Kesav Mohan, Risheek Rakshit Sukumar Kanmani, Krishna Anandan Ganesan, and Nisha Ramasubramanian. En éste se trabaja la tendencia NoSQL para cargas de trabajo OLAP [5] donde se investigarán varias opciones y configuraciones de bases de datos NoSQL. Se utiliza Apache Spark como Extracción, transformación y carga (ETL) para la limpieza y varias bases de datos NoSQL como son MongoDB, Redis, Apache Kudu y ArangoDB. Para evaluar el funcionamiento hace uso de un dataset generado por Koalabench. Con Koalabench se generan conjuntos de datos de tamaños variables para el modelo de datos deseado y se realizan una serie de experimentos con datos pertenecientes a dos modelos distintos: planos y nieve.

El último artículo seleccionado trata del *"Data warehouse systems"*, realizado por Alejandro Vaisman y Esteban Zim anyi, el cual cubre de manera extensa todo lo relacionado con los DW, desde las tecnologías más básicas que se pueden usar como las más recientes [6] explicando tanto arquitecturas como tecnologías a utilizar a la hora de desarrollar y diseñar un almacén de datos: OLAP y Microsoft Integration Services, esquemas de Lenguaje Unificado de Modelado (en inglés, Unified Modeling Language (UML)), lenguaje de consulta estructurada (en inglés, Structured Query Language (SQL)).

Artículo	**Fuente**	**Zona Geográfica**	**Año**
Efficient OLAP query processing across cuboids in distributed data warehousing environment [3]	ScienceDirect	Kolkata, India y Venecia, Italia	2024
Construcción y explotación de un data warehouse para el análisis de información sobre el tránsito rodado de vehículos [4]	Universitat Oberta de Catalunya	Barcelona, España	2014
Evaluating NoSQL Databases for OLAP Workloads: A Benchmarking Study of MongoDB, Redis, Kudu and ArangoDB [5]	arXiv y Google Scholar	India	2024
Data warehouse systems [6]	Springer	Panos Vassiliadis, Universidad de Ioannina, Grecia.	2014

Tabla 1. Resumen de los artículos utilizados en el estado del arte

Con esto, se puede concluir que existen una gran variedad de artículos y trabajos que

Artículo	**Fuente**	**Tecnología**
Efficient OLAP query processing across cuboids in distributed data warehousing environment [3]	ScienceDirect	Python, Microsoft SQL Server
Construcción y explotación de un data warehouse para el análisis de información sobre el tránsito rodado de vehículos [4]	Universitat Oberta de Catalunya	Spoon de Pentaho, MySQL y Microsoft Analysis Services
Evaluating NoSQL Databases for OLAP Workloads: A Benchmarking Study of MongoDB, Redis, Kudu and ArangoDB [5]	arXiv y Google Scholar	Apache's Hadoop Distributed File System (HDFS), Apache Spark, MongoDB, ArangoDB, Apache Kudu, Redis, PostgreSQL
Data warehouse systems [6]	Springer	Power BI, Neo4j, Hadoop, Spark, PostgreSQL, SQL Server

Tabla 2. Resumen de las tecnologías encontradas en el estado del arte

abordan el tema de los Data Warehouses, implementándolos con diferentes tecnologías y metodologías. Sin embargo, no se ha encontrado ninguno que realice una comparativa de la implementación de un mismo Data Warehouse utilizando distintas tecnologías y metodologías.

Capítulo 3

Contextualización

Para entrar en contexto, es necesario saber lo que es un data warehouse, las tecnologías necesarias para su implementación y sus posibles arquitecturas [7].

Un almacén de datos (en inglés Data Warehouse (DW)) es un repositorio central que almacena y procesa datos de múltiples fuentes dentro de una organización, facilitando así el análisis y la toma de decisiones posteriores. La esencia de un Data Warehouse (DW) está en su habilidad para integrar datos de fuentes variadas mediante el proceso de Extracción, Transformación y Carga (en inglés, Extraction, Transform and Load (ETL)) [8]. Este proceso se divide en tres etapas. La primera etapa es la extracción de la información, ésta consiste en la lectura de los datos del sistema operacional y fuentes externas, donde se dispone incluso de varios modos de extraer la información [1]:

- Extracción total (en inglés, Full Extract): Esta modalidad consiste en extraer la totalidad de datos, se barren tablas que pueden tener millones de registros.
- Extracción Incremental (en inglés, Incremental Extract): Se va procesando por lotes únicamente lo que fue modificado o agregado.
- Notificador de actualizaciones (en inglés, Update notification): Sólo se van extrayendo los datos a medida que se produce una actualización

La segunda etapa del proceso ETL trata de la de transformación de los datos extraídos de las fuentes operacionales: limpieza y estandarización.

La última etapa en este proceso es la de Carga, ésta consiste en mover los datos desde las fuentes operacionales o de almacenamiento intermedio hasta el almacén de datos y cargar los datos en las correspondientes estructuras de datos.

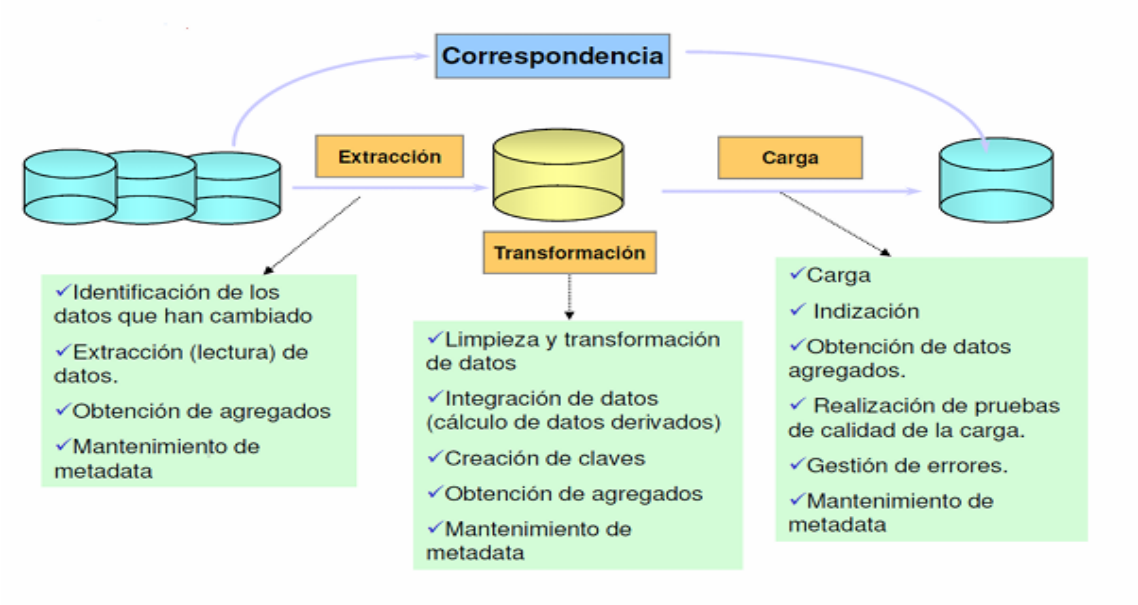

Figura 1: Sistema ETL [1]

En la profundización de estas soluciones, los DW se distinguen por ser bases de datos especializadas en el soporte a la toma de decisiones. A diferencia de las bases de datos tradicionales, que están orientadas al procesamiento de transacciones y al acceso concurrente rápido, los almacenes de datos están diseñados para realizar análisis de negocios. Esto implica la integración de datos procedentes de diversas fuentes operativas y su transformación en estructuras que se adaptan mejor a las tareas de análisis. Estos almacenes de datos se basan en un modelo multidimensional, donde los datos se representan en forma de hipercubos como el que se muestra en la figura 2. En estos cubos [1] se representa una actividad que es objeto de análisis (hechos) y las dimensiones que caracterizan la actividad (dimensiones). La información relevante sobre el hecho (actividad) se representa por un conjunto de indicadores (medidas o atributos de hecho). Por otro lado, la información descriptiva de cada dimensión se representa por un conjunto de atributos (atributos de dimensión)

Además de este enfoque general, existen dos muy conocidos, provenientes de dos autores significativos para este área: Bill Inmon [9] y Ralph Kimball [10]. Bill Inmon define los almacenes de datos en términos de las características del repositorio de datos, mientras que Ralph Kimball define un DW como un almacén de datos que extrae, limpia, conforma y entrega una fuente de datos dimensional para la consulta y el análisis. [11]

Una vez se tiene claro lo que es un DW, se deben conocer también las distintas arquitecturas que los soportan, dependiendo de los requisitos y necesidades que se

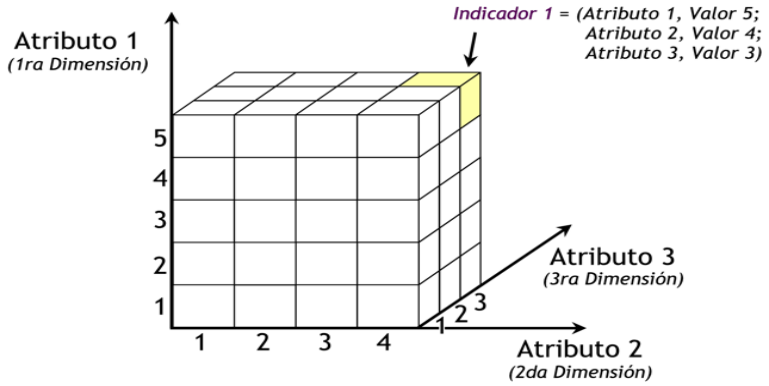

Figura 2: Cubo Multidimensional [2]

tengan.

La arquitectura es un aspecto importante. Además del propio almacén de datos, estos sistemas se componen de herramientas de back-end, que extraen datos de diversas fuentes para alojarlos en el almacén, y herramientas de front-end, que se utilizan para extraer información del almacén y presentarla a los usuarios. Esto destaca la importancia de una arquitectura bien diseñada que almacene los datos y que también permita una recuperación eficiente y una presentación adecuada de la información.

Por otro lado, los sistemas de procesamiento analítico en línea (en inglés, On-Line Analytical Processing (en inglés, OnLine Transaction Processing (OLAP)) son la base de la analítica de información. A diferencia de los sistemas de procesamiento de transacciones en línea (OLTP), que están enfocados en transacciones y operaciones del día a día, OLAP se centra en consultas analíticas. Las bases de datos orientadas a OLAP deben soportar una alta carga de consultas, involucrando agregación de datos y requiriendo técnicas de indexación y optimización de consultas específicas para estos propósitos. Así, OLAP facilita la realización de análisis complejos y es fundamental en la arquitectura de los Data Warehouses. Para efectuar análisis multidimensionales de manera eficiente, se recurre a la tecnología de procesamiento analítico en línea (OLAP), que posibilita una interacción intuitiva de los usuarios con los datos y facilita la realización de análisis complejos. Existen tres enfoques principales dentro de OLAP: MOLAP, ROLAP y HOLAP, cada uno con características y aplicaciones específicas que se adaptan a diferentes necesidades y contextos de análisis [12]:

- **MOLAP (Multi-dimensional Online Analytical Processing):** Este enfoque utiliza estructuras multidimensionales para el análisis de datos. MOLAP almacena los datos en cubos multidimensionales que permiten un rápido rendimiento en consultas complejas. Ejemplos de sistemas MOLAP incluyen

Microsoft Analysis Services y IBM Cognos TM1.

- **ROLAP (Relational Online Analytical Processing):** En contraste con MOLAP, ROLAP almacena los datos en bases de datos relacionales tradicionales. Utiliza técnicas de agregación y consultas SQL complejas para admitir el análisis multidimensional. Algunas soluciones ROLAP incluyen SAP BW y Oracle OLAP.

- **HOLAP (Hybrid Online Analytical Process):** HOLAP combina características de MOLAP y ROLAP, permitiendo a los usuarios aprovechar la velocidad de acceso a los datos en cubos multidimensionales, al tiempo que utiliza bases de datos relacionales para almacenar los detalles de los datos. Ejemplos de sistemas HOLAP son Microsoft SQL Server Analysis Services y SAP HANA.

A continuación se va a entrar más en detalle sobre los tres enfoques comentados en la lista anterior, para entenderlos de manera correcta [12].

En referencia a las tecnologías que se van a utilizar, se encuentran dos tipos diferenciados, los Sistemas Gestores de Bases de Datos Relacionales (SGBDR) y las bases de datos NoSQL.

3.1. Sistemas Gestores de Bases de Datos Relacionales (SGBDR)

Los Sistemas de Gestión de Bases de Datos Relacionales como MySQL, PostgreSQL, Oracle Database, y Microsoft SQL Server, son la columna vertebral de muchos Data Warehouses. Estos sistemas utilizan un modelo de datos basado en relaciones, lo cual facilita la integridad y consistencia de los datos.

Los datos [13] se almacenan en una base de datos relacional en forma de tablas múltiples. Una estructura de base de datos funciona organizando cada tabla en filas (conocidas como registros/tuplas) y columnas (conocidas como campos/atributos). Las tablas, las columnas, las filas y las relaciones son los cuatro componentes principales de una base de datos relacional.

Pero antes de crear las tablas, un SGBDR debe garantizar que cada una de las tablas tiene una clave primaria única, que no tiene valores nulos, la clave ajena, que se utiliza para realizar las relaciones, se conserva en una tabla y se refiere a la clave primaria de otra tabla.

3.2. Tendencia NoSQL

Las bases de datos NoSQL, como MongoDB, Cassandra y Neo4j, ofrecen una alternativa más flexible a los SGBDR tradicionales. Estas bases de datos son ideales para manejar grandes cantidades de datos no estructurados o semi-estructurados.

Dentro de las bases de datos NoSQL se encuentran varios enfoques con distintas características y ventajas a la hora de tratar los datos. Algunos ejemplos [14]:

- Orientadas a clave-valor(en inglés, Key-value): Prometen un rendimiento excelente para volúmenes de datos muy grandes, a cambio de ser muy simples y renunciar a funcionalidades que se tienen en otros sistemas como la verificación intrínseca de la integridad de datos, llaves extranjeras y disparadores. Algunos ejemplos de tecnologías son Redis y DynamoDB.

- Orientadas a columnas(en inglés, Wide Column): Son en realidad lo que se podría suponer, tablas de datos donde las columnas de valores de datos representan el almacenamiento estructural. Los datos son almacenados como secciones de las columnas de datos en lugar de filas de datos, como en la mayoría de los gestores relacionales. Algunos ejemplos son Hadoop / HBase, Cassandra.

- Orientadas a documentos(en inglés, Document stores): son consideradas por muchos como un escalón superior ante los simples gestores de llave-valor, puesto que permiten encapsular pares de llave-valor en estructuras más complejas denominadas documentos. Por otra parte no existe un esquema estricto a seguir para definir estos documentos, lo cual simplifica sustancialmente su uso. Algunos ejemplos son MongoDB, CouchDB o RavenDB.

- Orientadas a grafos(en inglés, Graph databases): representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos ya que ésta puede describir atributos de los nodos (entidades) y las aristas (relaciones). Algunos ejemplos son Neo4j, InfoGrid.

Capítulo 4

Descripción de los datos

A continuación, se realizará un análisis detallado de los datos utilizados en la implementación del Data Warehouse. Se examinará su origen, la estructura que presentan, el volumen manejado, el procesamiento aplicado a dichos datos, y finalmente, se abordarán aspectos clave relacionados con la seguridad y la privacidad de los mismos.

4.1. Origen de los datos

Como se ha indicado anteriormente, se pretende hacer un análisis de tecnologías, con el objetivo de almacenar los datos necesarios para llevar a cabo recargas inteligentes. Para ello, se han utilizado tres fuentes de datos distintas. La primera fuente de datos, la cual ha proporcionado los precios por hora y día de la energía, para la tarifa PVPC. Estos datos son devueltos por la API de la red eléctrica[1] en formato JSON(JavaScript Object Notation), la segunda fuente de datos es Weather API[2]para la obtención de datos meteorológicos en formato JSON y por último se han usado dos conjuntos de datos proporcionados por la empresa de cargadores eléctricos V2C. Cada vez que reciben un evento de cualquier cambio en un cargador, ya sea cargador o aplicación, estos lo publican en formato JSON sobre Kafka. Estos datos se almacenan con el fin de optimizar la carga inteligente de los cargadores eléctricos de V2C, permitiendo ajustar el consumo en función de la demanda energética y las condiciones meteorológicas. Además, son utilizados para desarrollar modelos predictivos que anticipen la demanda de carga, mejorando la eficiencia del sistema y reduciendo el impacto en la red eléctrica. Esto facilita una gestión más sostenible y adaptativa de los recursos energéticos.

Kafka o Apache Kafka[3] es una plataforma distribuida para la transmisión de datos, la cual permite tanto su publicación, como su consumo. En este caso, V2C es el encargado de publicar los datos que son consumidos y tratados para su posterior uso.

[1]API Red Eléctrica`https://www.ree.es/es/apidatos`
[2]Weather API: `https://www.weatherapi.com/`
[3]Apache Kafka: `https://www.redhat.com/es/topics/integration/what-is-apache-kafka`

Esta plataforma está diseñada para administrar los flujos de datos de varias fuentes y enviarlos a distintos usuarios. En Kafka, V2C publica los eventos en dos topics, o conjunto de datos, el primero guarda los eventos de cambios de propiedades y el segundo los eventos de carga, como se explica a continuación.

El primer topic de ellos se denomina change_property y contiene información sobre los cambios de propiedades y configuraciones de cada uno de los cargadores eléctricos. El segundo conjunto de datos proporcionado por V2C incluye información sobre las cargas realizadas en cada cargador.

4.2. Estructura de los datos

En cuanto a la estructura de los datos, se va a analizar cada uno de los campos recibidos, ya sea por parte de Kafka o bien, de una API (Application Program Interface).

4.2.1. Eventos asociados a estaciones de carga de VE

Change_property

Este fichero se compone de 42 campos, como pueden ser el device_id, charge_fv_mode, dynamic... entre otros muchos, pero solo se van a guardar en el DW los verdaderamente relevantes que son:

- device_id: Identificador del cargador.
- fecha: Fecha correspondiente al evento registrado, con formato YYYY-MM-DD HH-MM-SS.
- charge_fv_mode: En caso de tener placas solares instaladas en el hogar, indica si tiene el modo fotovoltaico activado.
- charge_state: Estado de carga, este valor indica si la manguera está conecta, si está cargando el vehículo, si no está cargando, o bien, la manguera está desconectada.
- dynamic: Se guarda, si el usuario tiene el control de pontencia dinámico. El control dinámico de potencia ajusta automáticamente la energía usada por el cargador eléctrico según el consumo de la casa, evitando sobrecargas y optimizando el uso de la red.
- inst_type: Tipo de instalación del cargador, indica si es trifásica o monofásica
- intensity: Intensidad del cargador.

- powerbytram: Indica si el cargador tiene siempre la misma potencia o la tiene definida .por tramos
- powerweekday: Especifica para cada día entre semana la potencia del cargador.
- powerweekend: Similar al campo anterior, pero para el fin de semana.

Charge

Este segundo conjunto de datos es más pequeño que el anterior y está compuesto por un total de 7 campos, de los cuales a continuación se van a mencionar los 6 más importantes, que son los que se guardarán posteriormente:

- device_id: Identificador del cargador.
- fecha: Fecha correspondiente al evento registrado, con formato YYYY-MM-DD HH-MM-SS.
- id_charge: Identificador de la carga(Suele ser la hora de inicio de carga)
- method: Tipo de evento de carga, puede ser inicio de carga (START_CHARGE) o fin de carga (END_CHARGE).
- energy: Energía total de la carga
- energy_by_hour: Energía por cada hora de carga

4.2.2. Datos meteorológicos

Este conjunto de datos, obtenidos a través de Weather API que contiene los datos meteorológicos de cada una de las ubicaciones de los cargadores. De todos los datos que devuelve la API, solo se van a necesitar los siguientes campos:

- device_id: Identificador del cargador.
- latitude: Latitud a la que se encuentra el cargador.
- longitude: Longitud a la que se encuentra el cargador.
- fecha: La fecha en formato YYYY-MM-DD a la que corresponden los datos meteorológicos.
- max_temp_c: Temperatura máxima registrada en Celsius.
- min_temp_c: Temperatura mínima registrada en Celsius.

- avg_temp_c: Temperatura media registrada en Celsius.
- total_precip_mm: Precipitaciones totales en Milimetros.
- avg_humidity: Humedad media.

4.2.3. Datos de tarifa eléctrica

La API de Red Eléctrica proporcionará los datos necesarios del precio de la electricidad por horas y las tarifas que se van a incluir dentro del DW. De esta información que ofrece la API se van a usar los siguientes campos:

- fecha: La fecha en formato YYYY-MM-DD para la cual corresponde el precio.
- precio: El valor del precio por kilovatio hora (kWh), generalmente expresado en €/MWh (euros por megavatio hora).

4.3. Volumen

El volumen inicial con el que se parte antes de realizar el proceso de ETL consta de dos archivos, los cuales contienen los datos de V2C, el primero de los archivos contiene los datos de cambios de propiedades con un total de 3.176.754 registros, mientras que el segundo, contiene los eventos de carga de los cargados y consta de 520.222 registros.

Con esta cantidad de datos de una de las fuentes y los datos recogidos de las otras dos fuentes han sido suficientes para la realización del TFG, sin necesidad de añadir datos extras para la realización de las implementaciones.

4.4. Requisitos de seguridad y privacidad

La Ley Orgánica de Protección de Datos (LOPD) y el Reglamento General de Protección de Datos (RGPD) juegan un papel muy importante a la hora de realizar este TFG, ya que se tratan datos sensibles a la hora de implementar el Data Warehouse. Para cumplir la legislación vigente, es necesario obtener el consentimiento, en este caso de V2C, para usar sus datos.

Otro aspecto fundamental para garantizar la seguridad de los datos en el almacén de datos (DW) es el control de acceso de cada una de las tecnologías utilizadas. Esto asegura que únicamente el usuario administrador pueda gestionar y acceder a los datos, evitando accesos no autorizados.

Capítulo 5

Tecnologías utilizadas

Para este TFG, se han evaluado diversas tecnologías SQL, NoSQL y herramientas especializadas en Data Warehouses, con el objetivo de comparar su rendimiento, escalabilidad y capacidad para manejar grandes volúmenes de datos en tiempo real, identificando la solución más eficiente.

5.1. MySQL

MySQL[1] [15] es un sistema gestor de bases de datos muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características avanzadas disponibles en otros SGBD del mercado, es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento precisamente por su curva de aprendizaje. Esto y su libre distribución en Internet bajo licencia GPL(General Public License) le otorgan como beneficios adicionales contar con un alto grado de estabilidad y un rápido desarrollo.

Figura 3: Logotipo MySQL

[1]Sitio oficial MySQL: `https://www.mysql.com/`

5.1.1. Características

En este apartado se enumeran las prestaciones que caracterizan a este SGBD, así como las deficiencias de diseño, limitaciones o partes del estándar aún no implementadas.

Algunas de sus prestaciones que se encuentran en MySQL son:

- Está optimizado para equipos de múltiples procesadores.
- Es muy destacable su velocidad de respuesta.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Soporta múltiples métodos de almacenamiento de las tablas, con prestaciones y rendimiento diferentes para poder optimizar el SGBD a cada caso concreto
- Su administración se basa en usuarios y privilegios.
- Sus opciones de conectividad abarcan TCP/IP, sockets UNIX y sockets NT, además de soportar completamente ODBC
- Es altamente confiable en cuanto a estabilidad se refiere.

Por otro lado, encontramos varias limitaciones, algunas de ellas son:

- El soporte de transacciones o la integridad referencial (la gestión de claves foráneas) en MySQL está condicionado a un esquema de almacenamiento de tabla concreto de forma, que si el usuario no va a usar transacciones puede usar el motor (MyISAM) y obtendrá mayor rendimiento, mientras que si su aplicación requiere transacciones, deberá usar el motor que lo permite (InnoDB), sin ninguna otra restricción o implicación
- No incluye características de objetos como tipos de datos estructurados definidos por el usuario, herencia etc.

5.2. PostgreSQL

PostgreSQL [2] [16] es un gestor de bases de datos objeto-relacional (ORDBMS en sus siglas en inglés) muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos SGBD comerciales.

[2] Sitio oficial PostgresSQL: `https://www.postgresql.org/`

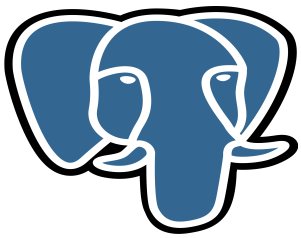

Figura 4: Logotipo PostgreSQL

5.2.1. Características

En este apartado se comentan las características más relevantes de este SGBD con soporte para objetos, tanto sus prestaciones más interesantes o destacadas, como las limitaciones en su diseño o en implementación de los estándares SQL.

Algunas de las prestaciones que se encuentran en PostgreSQL son:

- Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario.
- Su administración se basa en usuarios y privilegios.
- Es altamente confiable en cuanto a estabilidad se refiere.
- Control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados, subconsultas y casi todos los tipos y operadores soportados en SQL92 y SQL99

Por otro lado, también se encuentran una serie de limitaciones:

- Puntos de recuperación dentro de transacciones. Actualmente, las transacciones abortan completamente si se encuentra un fallo durante su ejecución. La definición de puntos de recuperación permitirá recuperar mejor transacciones complejas.
- No soporta tablespaces para definir dónde almacenar la base de datos, el esquema, los índices, etc. (En versiones anteriores de la 9.0)
- El soporte a orientación a objetos es una simple extensión que ofrece prestaciones como la herencia, no un soporte completo.

5.3. MongoDB

MongoDB[3] [17] es un software que permite a los usuarios crear y manipular bases de datos presentados en forma de documentos, los cuales son almacenados en un formato BSON (Binary JSON(JavaScript Object Notation)). La creación de la base de datos no requiere de previa definición de su esquema. Esta característica permite a los desarrolladores crear un sistema más flexible a futuras modificaciones de la estructura de documentos, en comparación con bases de datos relacionales.

Figura 5: Logotipo MongoDB

MongoDB es uno de los sistemas de bases de datos NoSQL más conocidos y debido a esto en muchas ocasiones es contrastado con sistemas relacionales y no relacionales.

Permite expresar consultas en diferentes lenguajes, por ejemplo, PHP, Node.js, Python, Ruby o Java.

5.3.1. Características

Algunas de las caracterísitcas principales que se encuentran en MongoDB son las siguientes [18]:

- Consultas ad hoc. Soporta la búsqueda por campos, consultas de rangos y expresiones regulares.
- Indexación. Cualquier campo que se encuentre en documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios.
- Replicación. MongoDB soporta el tipo de replicación primario-secundario.
- Balanceo de carga. Permite escalar de forma horizontal usando el concepto shard.

[3]Sitio oficial MongoDB: `https://www.mongodb.com/es`

- Almacenamiento de archivos. Puede ser usado como un sistema de archivos, aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos en múltiples servidores.
- Agregación. Proporciona un framework de agregación que permite realizar operaciones similares a "GROUP BY"de SQL.

5.4. Google BigQuery

Google BigQuery[4] [19] es un servicio web en la nube muy atractivo por su facilidad de uso y funcionalidad. Esta plataforma permite almacenar y recuperar grandes cantidades de información en tiempo casi real con un enfoque principal en el análisis de datos

Figura 6: Logotipo Google BigQuery

5.4.1. Características

Las características principales de Google BigQuery son las siguientes:

- Velocidad. BigQuery puede procesar millones de informaciones no indexadas en segundos debido al almacenamiento columnar, y la arquitectura del árbol.
- Escalabilidad. Es la capacidad de gestionar el tamaño de datos enormes con millones de registros, tiene la posibilidad de alcanzar terabytes de información, sin límites de espacio.

[4]Página oficial Google BigQuery: `https://cloud.google.com/bigquery?hl=es`

- Simplicidad. BigQuery proporciona una interfaz sencilla para cargar y ejecutar navegar a través de un lenguaje de consulta similar a SQL.
- Varios Permisos. Es la capacidad de gestionar diferentes permisos de acceso, solo lectura, editor o propietario.
- Seguridad. Para garantizar la seguridad, Google BigQuery utiliza SSL(Secure Sockets Layer).
- Multiples métodos de acceso. Se puede acceder al servicio de distintas maneras. Mediante la herramienta de busqueda proporcionada por BigQuery, por línea de comandos o mediante su API REST.

Capítulo 6

Implementación de los Data Warehouses

Para la realización del TFG se ha diseñado el mismo Data Warehouse, en varias tecnologías, con el objetivo de realizar una comparación más completa y real. Para ello, se han seleccionado tecnologías con características distintas, como pueden ser tendencias relacionales y tendencias NoSQL.

Las tecnologías que se han seleccionado para la implementación y realización del data warehouse son las ya descritas en el apartado anterior: MySQL, PostgresSQL, MongoDB y Google BigQuery.

A la hora de la implementación también se debe tener en cuenta el entorno de trabajo que se va a utilizar, que será el mismo donde se harán pruebas para comparar diferentes sistemas implementados. En este caso se ha utilizado un equipo que consta de un procesador AMD Ryzen 5 5500, lo suficientemente potente como para soportar las tareas que se van a realizar, además el equipo cuenta con 16 GiB de memoria RAM y una GPU RX 6650xt de AMD, la cual permite realizar tareas de procesamiento gráfico.

Como sistema operativo se ha utilizado Windows 10. Este sistema operativo realizado por Microsoft, ofrece una interfaz muy sencilla y fácil de utilizar, lo que ayudará al desarrollo de las distintas tecnologías, además de su seguridad y compatibilidad con un gran número de herramientas

Una vez se han indicado las tecnologías que se van a implementar y el entorno sobre el que se va a trabajar, se va a explicar cómo se ha trabajado con cada una de ellas, así como el procesado de los datos que se ha llevado a cabo.

6.1. Procesado de los datos

En este apartado se va a indicar en qué va a consistir el procesado de los datos que se ha llevado a cabo para todas las tecnologías. Este proceso es un aspecto muy importante a la hora de garantizar el correcto funcionamiento del Data Warehouse y el análisis correcto y específico de los datos utilizados.

Para ello, se ha utilizado Pandas [20], se trata de una librería de Python para la manipulación de datos y análisis de datos.

En este caso, se ha usado Pandas para obtener los datos del servidor de Kafka de V2C. Una vez obtenidos dichos datos se ha procedido a su limpieza. Por una parte, en el subconjunto de datos charge se han descartado todos los eventos 'endcharge' descartando las cargas que no están entre 5 y 150 kw. Para la limpieza del subconjunto de datos change_property, únicamente se han obtenido los eventos que contienen el campo 'charge_state' a 1 o 0. Además de esta limpieza, se han descartado los registros que cuentan con fechas inválidas, como pueden ser, fechas futuras o inexistentes. Tras el proceso de ETL, los datos fueron almacenados en dos archivos CSV, que serán utilizados en la fase de implementación del Data Warehouse.

Además de la limpieza de los datos obtenidos de V2C, se ha utilizado un proceso similar para recolectar y depurar la información de las APIs de Red Eléctrica y Weather API. Los datos de ambas fuentes fueron tratados para simplificar su posterior inserción en el Data Warehouse. Los datos de Weather API se han obtenido en función de la posición del cargador, descartando los cargadores que se encuentran en latitud y longitud 0. Una vez procesados, todos los datos se almacenaron en archivos CSV, siguiendo el mismo enfoque, para facilitar su integración dentro del sistema. [21]

6.2. Implementación en MySQL

Lo primero que se ha realizado para la implementación en MySQL ha sido la instalación del entorno de desarrollo en el equipo.

Para empezar, se ha descargado la versión 8.4.0 proporcionado en la página oficial de MySQL. Una vez ejecutado e instalado, se deberá introducir en las variables de entorno el PATH correspondiente a la instalación para poder acceder a la consola de mysql a través de la PowerShell de Windows.

Una vez se ha instalado y configurado de manera correcta MySQL, se pasará a probar dicha tecnología con un banco de pruebas. Como se ha explicado en el capítulo 4.

6.2.1. Procesado de los datos en MySQL

El procesamiento de datos en MySQL fue sencillo, ya que esta tecnología ofrece gran flexibilidad para introducir datos en el formato deseado. El primer paso fue la importación de los archivos CSV, en la que se trataron las filas innecesarias, como los encabezados, preparando los datos para su inserción en las tablas.

Una vez cargados los datos, se realizó un proceso de normalización para distribuirlos entre las tablas dimensionales. Esta normalización evita la duplicación de información y facilita futuras consultas.

Finalmente, se utilizaron operaciones específicas para realizar agregaciones y unificaciones, combinando registros similares y consolidando datos de distintas tablas. Con los datos correctamente estructurados, se procedió a la implementación.

La implementación en MySQL para el almacenamiento y análisis de eventos relacionados con vehículos eléctricos se realiza mediante un diseño de esquema estrella, que es el más efectivo para almacenes de datos SGBDR. En la figura 7, se representa la estructura de tablas de la implementación. Como se puede observar, se centra alrededor de una tabla de hechos, que en este caso es ‘f_charging_event‘, registrando métricas cuantitativas de eventos de carga, como la identificación del dispositivo, el método de carga y la configuración del dispositivo, además de la fecha y hora del evento. Las tablas de dimensiones proporcionan un contexto descriptivo y se relacionan con la tabla de hechos a través de claves ajenas, lo que permite un análisis multidimensional de los datos.

Las dimensiones incluyen ‘d_device‘, que detalla las características de los dispositivos utilizados en los eventos de carga; ‘d_charge_method‘, que describe los métodos de carga. Las tablas ‘d_pvpc‘ y ‘d_weather‘ contienen información sobre las dos APIs utilizadas, en la primera se guardan la fecha y el precio de la electricidad en dicha fecha, mientras que en la otra se van a guardar los datos relacionados con los datos meteorológicos como son la temperatura máxima, la mínima, la temperatura media o las precipitaciones medias. La razón de esta implementación en MySQL reside en su capacidad para manejar estructuras de datos complejas, proporcionando un esquema bien definido y un fuerte soporte para transacciones, lo que es crucial para mantener la integridad de los datos en un entorno donde la precisión es fundamental. Este soporte transaccional y la gestión avanzada de datos se logran mediante el uso del motor InnoDB, que es conocido por su confiabilidad en la preservación de la integridad referencial y su capacidad para manejar bloqueos a nivel de fila, lo que es especialmente importante para evitar problemas de concurrencia. Además, la naturaleza relacional de MySQL favorece las operaciones de join, que son esenciales en la realización de

consultas que cruzan múltiples dimensiones para el análisis de datos. Esta estructura también está diseñada para adaptarse a la naturaleza cambiante del análisis y del procesamiento de datos en gran escala. Aunque las bases de datos relacionales como MySQL presentan limitaciones en la escalabilidad horizontal y rigidez de esquemas, para el caso de datos estructurados con relaciones bien definidas y requerimientos de consistencia transaccional fuerte, como es el caso de los eventos de vehículos eléctricos, esta implementación sigue siendo una solución efectiva y confiable.

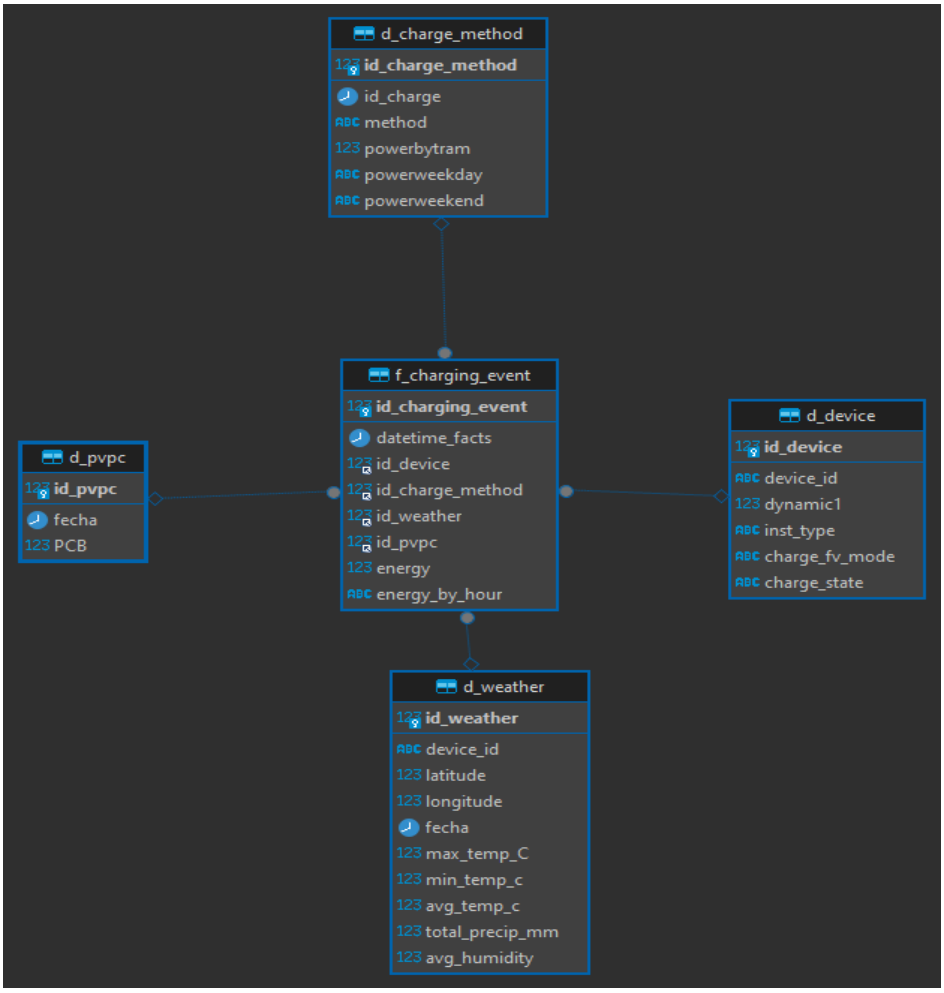

Figura 7: Diseño físico en MySQL

Para importar los datos dentro de la base de datos, se ha realizado un script el cuál recoge los datos de los dos datasets correspondientes y de los datos obtenidos de ambas APIs, crea las tablas correspondientes del data warehouse e introduce los datos de manera correcta mediante consultas SQL en cada una de las tablas.

6.3. Implementación en PostgresSQL

Para la implementación en PostgresSQL, como bien se ha hecho en MySQL, se deberá instalar correctamente la versión que se vaya a utilizar, para ello se deberá seleccionar y descargar el instalador correspondiente, en este caso se ha seleccionado la versión 15.7. Una vez instalado, incluir el PATH de los archivos en las variables de entorno del sistema operativo, al igual que en la instalación anterior.

Una vez que se ha instalado correctamente, se realizará el procesado de los datos y la implementación.

6.3.1. Procesado de los datos en PostgresSQL

El procesado en PostgreSQL es un poco más complejo que MySQL, ya que se trata de una tecnología más restrictiva a la hora de la agregación de los datos.

El primer procesado, al igual que anteriormente, se encuentra a la hora de importar los datos desde los archivos CSV. En esta fase, los datos se encuentran en un formato sin procesar, lo que implica que aún no están estructurados ni normalizados.

A continuación, para evitar los conflictos con los valores nulos o campos vacíos, se convierten los campos en formato texto, una vez se han cargado dichos datos. Posteriormente, se actualizan dichos campos, cambiando esos campos al formato correcto, manejando correctamente su valor.

Finalmente, se insertarán los datos en las tablas de dimensiones de manera estructurada y normalizada.

En este caso, va a resultar similar a la implementación que se encuentra en MySQL ya que PostresSQL también es un SGBDR, por lo tanto la implementación también será realizada en un diagrama en estrella, en el cual se encontraran varias tablas de dimensiones, rodeando una tabla de hechos, que en este caso, y al igual que en la implementación anterior, se trata de la tabla 'f_charging_event' en la cual se guardarán datos cuantitativos de eventos de cargas, como la identificación del dispositivo, el método de carga y la configuración del dispositivo, además de la fecha y hora del evento. Por otro lado, las tablas de dimensiones tienen una mera función descriptiva. Estas tablas serán las mismas que se han definido anteriormente. Estas son 'd_device' en la que se guardarán las características de los dispositivos utilizados, luego se encuentra 'd_charge_method' donde se guardán los métodos de carga. Por otro lado, las tablas 'd_pvpc', 'd_weather' albergan información sobre las APIs que se han utilizado. La primera guarda los datos de la API de Red Eléctrica, y en la segunda los datos meteorológicos. Este esquema se puede observar en la figura 8

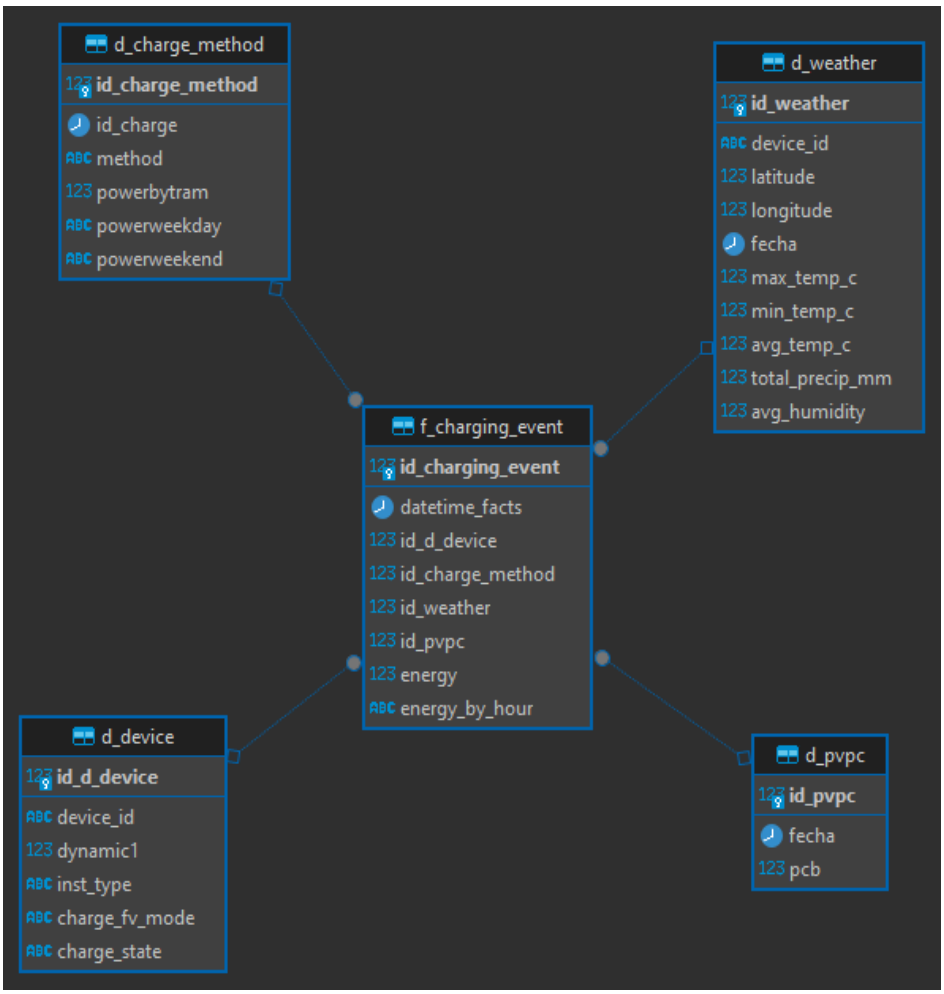

Figura 8: Diseño físico en PostgreSQL

A diferencia de MySQL, PostgreSQL ofrece propiedades ACID en todas sus configuraciones, lo que garantiza un control eficiente de la integridad de los datos. También implementa control de concurrencia multiversión (MVCC), permitiendo el acceso simultáneo de múltiples usuarios sin problemas. Este SGBDR destaca por su flexibilidad en el manejo de tipos de datos, alta escalabilidad y una gestión de la integridad de datos más eficiente.

6.4. Implementación en MongoDB

A diferencia de las implementaciones previas, MongoDB es un sistema gestor de bases de datos NoSQL documental, lo que requiere un enfoque distinto. En lugar de tablas relacionadas entre sí, la base de datos se compone de colecciones que contienen documentos con la información correspondiente.

Para llevar a cabo esta implementación, se ha utilizado la versión 7.0.11 de MongoDB. La instalación en un entorno Windows comienza descargando el instalador MSI (Microsoft Installer) correspondiente desde la página oficial de MongoDB. Una vez descargado, se ejecuta el archivo MSI y se siguen las instrucciones en pantalla, asegurándose de seleccionar la opción para instalar MongoDB como un servicio de Windows, lo que permite que MongoDB se ejecute automáticamente al iniciar el sistema. Durante la instalación, se puede optar por agregar MongoDB al PATH del sistema, lo que facilita el acceso a la línea de comandos de MongoDB desde cualquier ubicación en el sistema. Después de la instalación, se verifica que todo haya sido configurado correctamente abriendo una ventana de terminal y ejecutando el comando mongo –version para confirmar que MongoDB se ha instalado correctamente y que la versión es la 7.0.11.

6.4.1. Procesado de los datos en MongoDB

Este proceso incluye la limpieza, carga y transformación de datos provenientes de archivos CSV, así como la construcción de documentos que se insertarán en una base de datos NoSQL.

El primer paso consiste en importar los datos desde los archivos CSV, asegurando que el delimitador sea el correcto para evitar errores en los datos. Tras la importación, se crea el documento principal, en el cual se indexarán otros documentos y se integrarán los datos necesarios.

Este enfoque prepara los datos para su posterior implementación y análisis en la base de datos NoSQL.

Para esta implementación, se ha utilizado una arquitectura basada en documentos embebidos en MongoDB. En lugar de seguir un enfoque relacional tradicional, donde cada entidad se almacena en una tabla separada, se ha optado por un diseño en el que los documentos contienen toda la información relacionada de manera jerárquica. Este enfoque se basa en la idea de que los datos relacionados deben agruparse en un solo documento siempre que sea posible, lo que mejora el rendimiento de las lecturas y reduce la necesidad de uniones (joins) entre diferentes colecciones. Los documentos en MongoDB tienen una estructura similar a los archivos JSON, como se puede observar en la figura 9. Esta estructura BSON (Binary JSON) permite flexibilidad y la posibilidad de realizar cambios en la estructura de los datos sin tiempo de inactividad.

Por otro lado, MongoDB proporciona una gran escalabilidad horizontal a través de

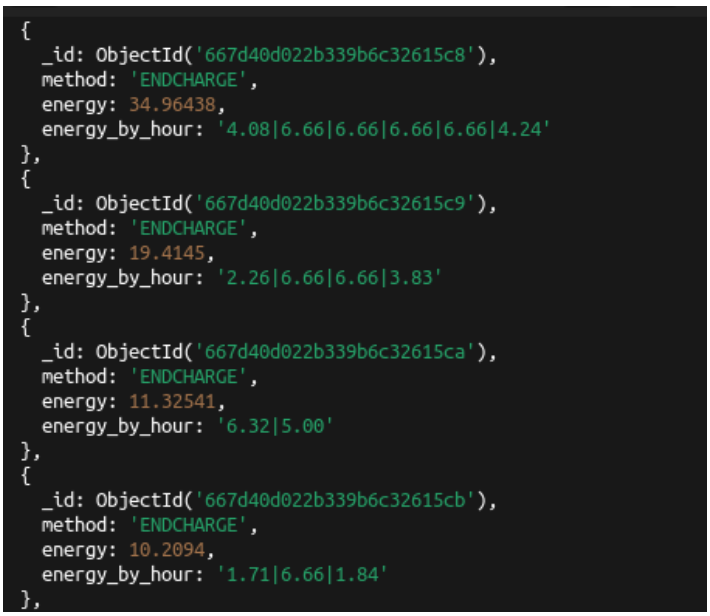

Figura 9: Estructura BSON

un proceso de fragmentación y ofrece un alto rendimiento, ya que almacena los datos en RAM para un acceso más rápido.

En esta implementación, se ha creado una colección principal llamada charging_events como se puede observar en la figura 10, donde se almacenan los eventos de carga de los vehículos eléctricos. Dentro de cada documento, se embeben subdocumentos que contienen los detalles del dispositivo, método de carga y configuración del dispositivo, y otros datos relevantes como los de Red Eléctrica y los meteorológicos. Este diseño reduce la necesidad de realizar consultas complejas y mejora el rendimiento general del sistema. Esta estructura también está diseñada para adaptarse a la naturaleza dinámica y cambiante de los datos de eventos de vehículos eléctricos, permitiendo que la estructura evolucione con el tiempo sin necesidad de realizar migraciones de esquema complejas.

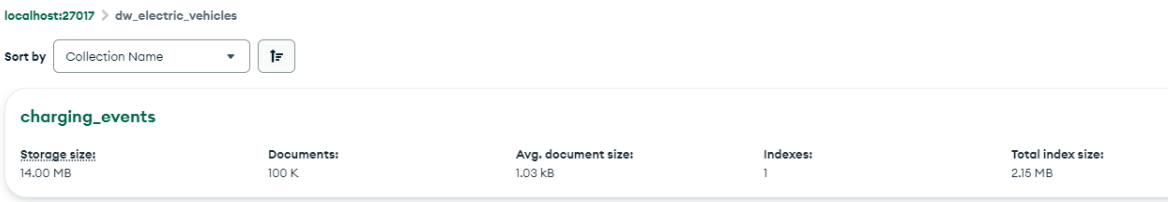

Figura 10: Colección charging_events

6.5. Implementación en Google BigQuery

Por último, se ha implementado el DW en Google BigQuery. Se trata de una tecnología con una gran capacidad para manejar grandes volúmenes de datos, escalabilidad y funcionalidades avanzadas como son el particionamiento y el clustering de tablas, entre otras, explicadas en el apartado de tecnologías.

6.5.1. Procesado de los datos

Respecto al proceso de carga de los datos dentro de de Google BigQuery, se ha creado 1 tabla auxiliar para cada fuente de datos, es decir, una tabla para los datos de PVPC, otra tabla para los datos meteorológicos y dos tablas más para los datos de carga y de cambios de propiedades.

Estos datos, contenidos en archivos CSV se han cargado mediante la opción de carga de archivos locales que ofrece Google BigQuery, haciendo de esto una tarea sencilla y rápida. También se ha utilizado la funcionalidad que ofrece para subir datos desde Google Drive, debido a las limitaciones de tamaño de Google BigQuery.

Por último, destacar que se han limpiado los datos, cambiando los datos de tipo 'TIMESTAMP' a 'DATE' en los casos necesarios y se han eliminado los datos duplicados.

Una vez se ha hablado del procesado de los datos dentro del Google BigQuery, se explicará la estructura y el diseño que se ha seguido para la implementación del DW. Se encuentran cuatro tablas de dimensiones, la cuales son 'd_device', 'd_charge_method', 'd_weather' y 'd_pvpc'. Dichas tablas se encargan de almacenar información como la información del dispositivo, los métodos de carga que se utilizan, los datos meteorológicos que se han obtenido y los precios de la red eléctrica.

Todas estas tablas de dimensiones están relacionadas de forma lógica con una tabla principal, la tabla de hechos, la cual tiene como nombre 'f_charging_event'. En esta tabla se almacenan datos cuantitativos de los eventos de carga. Esta estructura y diseño se puede observar en la Figura 11.

Figura 11: Diseño Físico Google BigQuery

Como se puede ver en la Figura anterior, y se ha explicado, las relaciones entre las tablas de dimensiones y la tabla de hechos es lógica, y se manejan a nivel de consulta mediante JOIN, por lo tanto no son visibles en el diseño físico a diferencia del diseño físico que se encuentra en MySQL o PostgresSQL.

Capítulo 7

Pruebas y Comparación

Una vez se ha implementado el DW con cada una de las tecnologías seleccionadas y explicadas en apartados previos, se realizarán una serie de pruebas para evaluar y comparar el rendimiento de cada implementación. Con el objetivo de comprobar cuál ofrece los mejores resultados a nivel de eficiencia.

Dichas pruebas constarán de las cuatro operaciones CRUD (Create, Read, Update, Delete). Se han realizado para cada una de las implementaciones las mismas consultas y operaciones y se han medido los tiempos de ejecución de cada una de ellas, con el objetivo de compararlos.

A continuación, la figura 12 presenta un gráfico general que aporta una visión rápida y global de las diferencias entre las tecnologías en cada una de las pruebas realizadas.

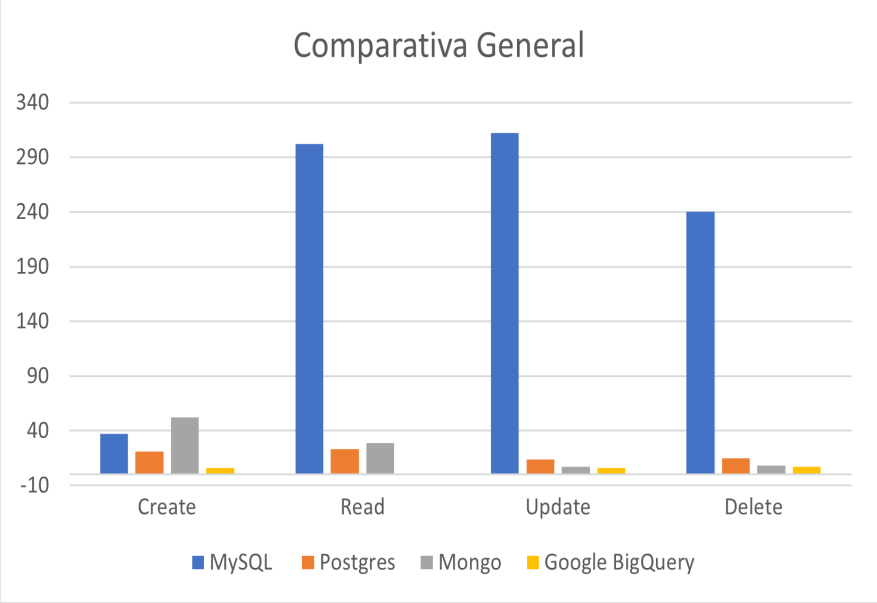

Figura 12: Comparativa General de las Tecnologías

Se presenta también la siguiente Tabla 3, la cual ayudará a visualizar los datos exactos de las pruebas tras la representación gráfica anterior.

Operación	MySQL	PostgreSQL	MongoDB	Google BigQuery
Create	37	21	52	6
Read	302	23	29	1
Update	312	14	7	6
Delete	240	15	8	7

Tabla 3. Comparativa de tiempos de ejecución (s) entre tecnologías en operaciones CRUD

7.0.1. Inserción de datos (Create)

La consulta que se observa en la Figura 13, es la que se ha utilizado para realizar las pruebas de inserción de datos. Esta consulta inserta un total de 1000000 de registros en la tabla 'f_charging_event', con datos que provienen de las distintas tablas de dimensiones comentadas en el apartado de implementación. Cada registro insertado contiene información sobre el dispositivo (id_device), el método de carga (id_charge_method), las condiciones meteorológicas (id_weather), y los precios de PVPC (id_pvpc). La consulta establece una fecha y hora actual mediante NOW(), junto con valores fijos para energy y energy_by_hour (50 y 5, respectivamente).

```
INSERT INTO f_charging_event (id_device, id_charge_method, id_weather, id_pvpc, datetime_facts, energy, energy_by_hour)
SELECT d.id_device, cm.id_charge_method, w.id_weather, pv.id_pvpc, NOW(), 50, 5
FROM d_device d
JOIN d_charge_method cm ON cm.id_charge_method = 1
JOIN d_weather w ON w.device_id = d.device_id
JOIN d_pvpc pv ON MONTH(pv.fecha) = 9
LIMIT 1000000;
--
```

Figura 13: Consulta Insercción MySQL

A su vez, se ha añadido el gráfico 14, el cuál muestra los tiempos de ejecución de la operación de inserción en cada tecnología, medidos en segundos. Los resultados indican variaciones significativas en el rendimiento, destacando a Google BigQuery como la opción más rápida con un tiempo de 6 segundos, mientras que MongoDB presenta el tiempo de inserción más alto con 52 segundos.

Estas gráfica proporciona una representación visual de los tiempos medidos a la hora de realizar la consulta en cada tecnología.

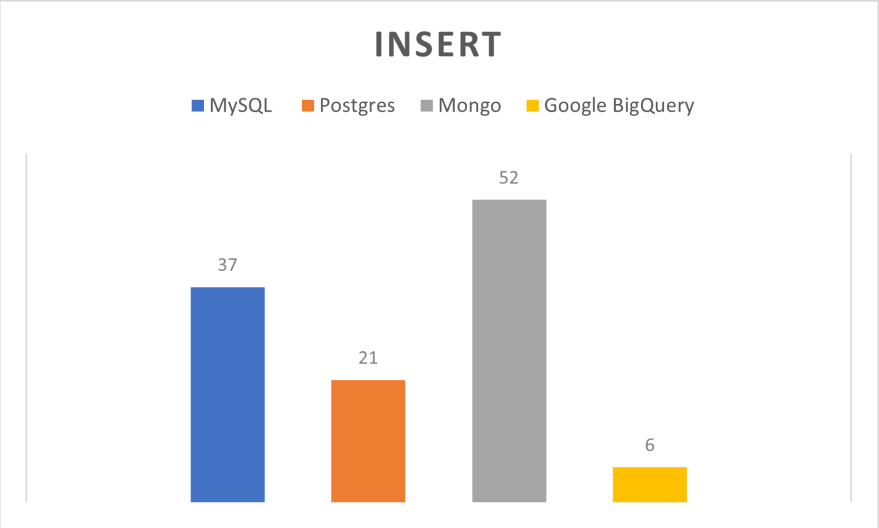

Figura 14: Comparativa Insert de las Tecnologías

7.0.2. Lectura de datos (Read)

Para las pruebas de lectura de datos se ha realizado la consulta que se puede observar en la figura 15. La consulta obtiene un resumen del consumo energético en f_charging_event, combinando datos de dispositivo, método de carga, condiciones meteorológicas y precios de electricidad. Calcula la energía total y promedio por hora, agrupando los datos por dispositivo, tipo de instalación, método, temperatura y precio, y muestra los 1,000 dispositivos con mayor consumo, ordenados de mayor a menor.

```sql
SELECT DISTINCT
    e.id_device,
    d.inst_type,
    cm.method,
    w.avg_temp_c,
    pv.PCB,
    SUM(e.energy) AS total_energy,
    AVG(e.energy_by_hour) AS avg_energy_per_hour
FROM f_charging_event e
JOIN d_device d ON e.id_device = d.device_id
JOIN d_charge_method cm ON e.id_charge_method = cm.id_charge_method
JOIN d_weather w ON e.id_weather = w.id_weather
JOIN d_pvpc pv ON e.id_pvpc = pv.id_pvpc
GROUP BY e.id_device, d.inst_type, cm.method, w.avg_temp_c, pv.PCB
ORDER BY total_energy DESC
LIMIT 1000;
```

Figura 15: Consulta Lectura MySQL

Una vez se tiene clara la consulta que se ha realizado, se ha añadido el gráfico 16, el cual muestra de forma visual los resultados obtenidos. Como se puede observar, Google BigQuery es la más rápida de las tecnologías con mucha diferencia, obteniendo todos los datos en solo 1 segundo. Por otro lado, se observa que MySQL es la más lenta de todas, muy por encima del resto, tardan 302 segundos en realizar la consulta.

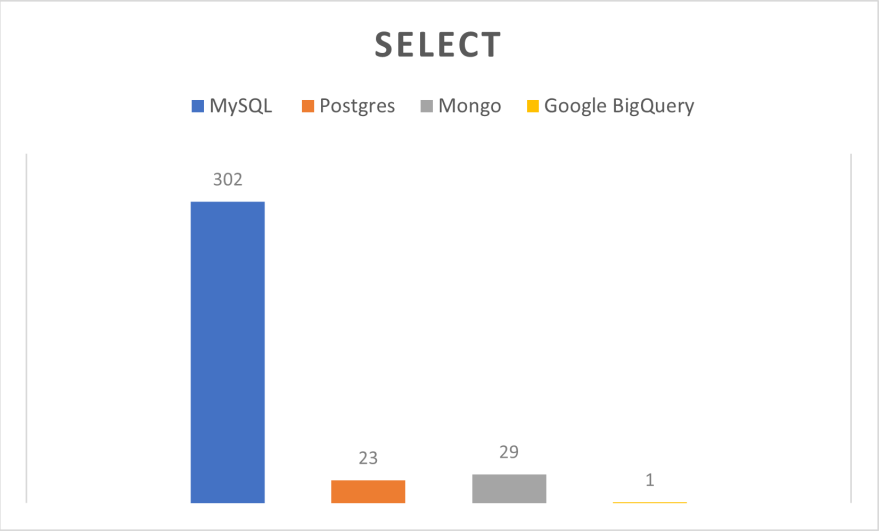

Figura 16: Comparativa Select de las Tecnologías

7.0.3. Actualización de datos (Update)

Como tercera prueba se encuentra la actualización de datos, la consulta que se ha realizado se puede observar en al figura 17. Esta consulta actualiza la tabla f_charging_event incrementando en 10 unidades el campo energy para registros específicos. Filtra los eventos que coinciden con precios de electricidad de los primeros diez días de septiembre y que pertenecen a un conjunto específico de dispositivos (id_device en una lista predefinida). Esto permite ajustar el consumo de energía solo para ciertos dispositivos y fechas.

```
UPDATE f_charging_event e
JOIN d_pvpc pv ON e.id_pvpc = pv.id_pvpc
SET e.energy = e.energy + 10
WHERE MONTH(pv.fecha) = 9
  AND DAY(pv.fecha) <= 10
  AND e.id_device IN (1, 2, 8, 9, 14, 15, 24, 30, 36, 41, 42, 51, 52, 53);
```

Figura 17: Consulta Actualizacion MySQL

En el gráfico 18, se puede observar los resultados de la consulta, donde se observa

que, al igual que en la anterior prueba Google BigQuery es la más rápida y eficiente, muy igualada con MongoDB con un tiempo de 6 y 7 segundos respectivamente, mientras que MySQL vuelve a ser la más lenta de las cuatro tecnologías implementadas.

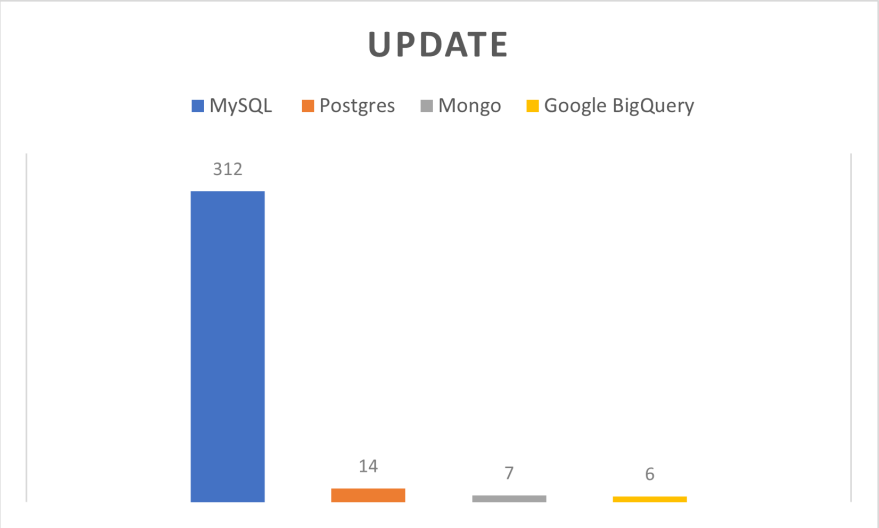

Figura 18: Comparativa Update de las Tecnologías

7.0.4. Eliminación de datos (Delete)

Por último, como se muestra en la figura 19, se ha ejecutado una consulta de eliminación que borra todos los registros de la tabla f_charging_event correspondientes a aquellos con una media de humedad superior a 90 durante el mes de septiembre.

```
DELETE e
FROM f_charging_event e
JOIN d_weather w ON e.id_weather = w.id_weather
WHERE w.avg_humidity > 90
  AND MONTH(e.datetime_facts) = 9
  AND DAY(e.datetime_facts) <= 30;
```

Figura 19: Consulta Eliminación MySQL

Los resultados obtenidos se reflejan en el gráfico 20, donde se puede observar que los resultados de manera visible. Observamos una vez más, que la tecnología más rápida en realizar la consulta es Google BigQuery, una vez seguida de MongoDB al igual que en la prueba anterior, siendo los tiempos de de 7 y 8 respectivamente. Por otro lado PostgresSQL se encuentra con un tiempo más que bueno, tardando 15 segundos

en realizar la consulta y como tecnología más lenta una vez más MySQL, con 240 segundos.

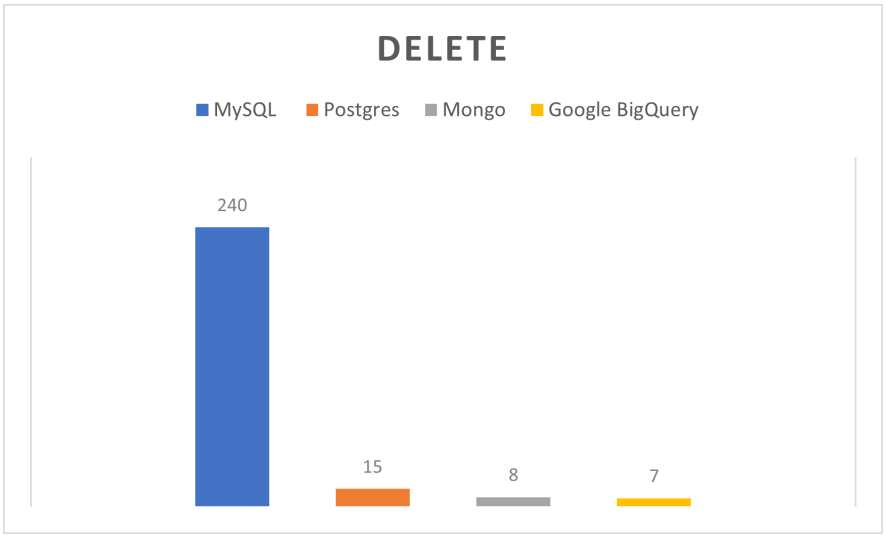

Figura 20: Comparativa Delete de las Tecnologías

En resumen, las pruebas realizadas en este apartado han permitido observar como se comportan cada una de las tecnologías y su rendimiento en las operaciones básicas que se realizan en los DW. Permitiendo observar las diferencias entre cada una de ellas. En el siguiente apartado se comentarán las conclusiones obtenidas a lo largo de la realización del TFG, así como posibles trabajos futuros que se pueden realizar.

Capítulo 8

Conclusiones y Trabajo Futuro

En este trabajo se ha abordado la implementación de un DW en distintas tecnologías. Se han implementado y comparado cuatro tecnologías distintas que son MySQL, PostgreSQL, MongoDB y Google BigQuery. Para la implementación de dicho DW se han utilizado una serie de datasets obtenidos de V2C, otro dataset obtenido de la API de red eléctrica y el último conjunto de datos usado se ha obtenido de WeatherAPI. Se ha evaluado el rendimiento de cada una de la implementaciones realizando operaciones CRUD (Create, Read, Update y Delete).

Tras ello, se ha determinado que el DW implementado en Google BigQuery obtiene mejores resultados a la hora de realizar las operaciones CRUD básicas utilizados en los DWs. Así pues, también se puede determinar que MySQL es la tecnología con peor rendimiento a la hora de implementar un DW.

Por otro lado, se observa que para el uso de DW dentro de la industria 4.0, donde se gestionan grandes volumenes de datos en tiempo real, Google BigQuery y MongoDB se colocan como opciones más óptimas. Siendo Google BigQuery enfocado al análisis de datos estructurados en la nube, mientras que MongoDB es adecuado para datos no estructurados.

Por último, se puede concluir que si se busca una base de datos robusta y que mantenga la integridad de los datos que requieren ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), la mejor opción es PostgreSQL. A pesar de no ser la opción más rápida a la hora de realizar las operaciones CRUD, mantiene unos tiempos bastante constantes y equilibrados.

Este TFG resuelve la hipótesis de partida para la que ha sido necesario disponer de conocimientos en varias tecnologías dentro de las bases de datos. Conocimientos adquiridos a lo largo de la titulación del Grado en Ingeniería Informática (GII). La parte más compleja a la que ha habido que enfrentarse es el proceso ETL que se ha tenido que realizar en cada uno de los conjuntos de datos usados, para poder utilizar los datos de manera correcta y evitar problemas de formatos y datos no válidos o

imposibles.

Una vez implementadas y probadas todas y cada una de las tecnologías para la manipulación de grandes volúmenes de datos, como trabajo futuro se podría implementar el DW en otras tecnologías distintas o probar otras arquitecturas o enfoques.(Ver Anexo I)

Por otro lado, en este TFG se han utilizado datos de un mes para facilitar la implementación de los DW, un trabajo futuro sería la ampliación de los dataset añadiendo datos del año entero, o bien obteniendo los datos en tiempo real. También se podrían tener en cuenta otras tarifas de red eléctrica, ya que se ha usado solamente PVPC.

Por último, cabe mencionar la posibilidad de extender el trabajo añadiéndole alguna técnica de Inteligencia Artifial (IA) para la predicción de datos a partir de los datasets usados.

Capítulo 9

Bibliografía

[1] Universidad de Zaragoza. Almacenes y minería de datos, 2023/24. Curso de la Universidad de Zaragoza.

[2] Dataprix. Data warehouse - conceptos básicos y modelo de datos dimensional, 2024. Artículo en línea; consultado 29-julio-2024.

[3] Santanu Roy, Saikat Raj, Tamal Chakraborty, Anirban Chakrabarty, Agostino Cortesi, and Soumya Sen. Efficient olap query processing across cuboids in distributed data warehousing environment. *Expert Systems with Applications*, 239:122481, 2024. Artículo de revista científica.

[4] Rubén Sánchez Ginés. Construcción y explotación de un data warehouse para el análisis de información sobre el tránsito rodado de vehículos. 2014. Tesis universitaria.

[5] Rishi Kesav Mohan, Risheek Rakshit Sukumar Kanmani, Krishna Anandan Ganesan, and Nisha Ramasubramanian. Evaluating nosql databases for olap workloads: A benchmarking study of mongodb, redis, kudu and arangodb. 2024. Artículo en arXiv.

[6] Alejandro Vaisman and Esteban Zimányi. Data warehouse systems. *Data-Centric Systems and Applications*, 2014. Artículo de libro.

[7] Wikipedia. Almacén de datos — wikipedia, la enciclopedia libre, 2024. Enciclopedia en línea; consultado 1-marzo-2024.

[8] Wikipedia. Extract, transform and load — wikipedia, la enciclopedia libre, 2024. Enciclopedia en línea; consultado 29-mayo-2024.

[9] Wikipedia contributors. Bill inmon — wikipedia, the free encyclopedia, 2024. Enciclopedia en línea; consultado 27-junio-2024.

[10] Wikipedia contributors. Ralph kimball — wikipedia, the free encyclopedia, 2024. Enciclopedia en línea; consultado 27-junio-2024.

[11] GM Faruk Ahmed, Md Shoriful Islam, and Molla Md Rezaul Karim. Comparison between inmon and kilball methodology for the purpose of designing, constructing and testing of a commercial bidw project. *History*, 8(1), 2017.

[12] Marysol Tamayo and Francisco Javier Moreno. Análisis del modelo de almacenamiento molap frente al modelo de almacenamiento rolap. *Ingeniería e Investigación*, 26:135 – 142, 12 2006. Artículo de revista científica.

[13] Astera. Sistemas de gestión de bases de datos relacionales, 2024. Artículo en línea; consultado 1-agosto-2024.

[14] Hansel Gracia del Busto and Osmel Yanes Enríquez. Bases de datos nosql. *Telemática*, 11(3):21–33, 2012. Artículo de revista.

[15] Luis Alberto Casillas Santillán, Marc Gibert Ginestà, and Óscar Pérez Mora. Bases de datos en mysql. *Universitat Oberta de Catalunya*, 2014. Artículo universitario.

[16] Marc Gibert Ginestà and Oscar Pérez Mora. Bases de datos en postgresql. *Sl]:[sn]*, 2012. Artículo universitario.

[17] Kenneth Calvo, Johan Durán, Esteban Quirós, and Elzbieta Malinowski. Mongodb: alternativas de implementar y consultar documentos. In *IX Congreso Internacional de Computación y Telecomunicaciones, COMTEL, Lima*, pages 48–49, 2017. Actas de congreso.

[18] Arimetrics. Mongodb, 2024. Artículo en línea; consultado 1-agosto-2024.

[19] Sérgio Fernandes and Jorge Bernardino. What is bigquery? In *Proceedings of the 19th International Database Engineering & Applications Symposium*, pages 202–203, 2015. Actas de congreso.

[20] pandas development team. pandas user guide. `https://pandas.pydata.org/docs/user_guide/index.html`, 2024. Guía en línea; consultado 4-noviembre-2024.

[21] International Organization for Standardization. ISO/IEC 25010:2011 - Ingeniería de sistemas y software — Requisitos y evaluación de la calidad de sistemas y software (SQuaRE) — Modelos de calidad de sistemas y software, 2011. Último acceso: 21 de noviembre de 2024.

Anexo I. Enfoques dentro de OLAP

Los tres enfoques principales que se encuentran dentro de OLAP son los siguientes:

Un sistema **MOLAP** usa una Base de datos multidimensional, en la que la información se almacena multidimensionalmente. El sistema MOLAP utiliza una arquitectura de dos niveles: la BDMD y el motor analítico. La BDMD es la encargada del manejo, acceso y obtención de los datos. El nivel de aplicación es el responsable de la ejecución de las consultas OLAP. El nivel de presentación se integra con el de aplicación y proporciona una interfaz a través de la cual los usuarios finales visualizan los análisis OLAP. La información procedente de los sistemas transaccionales se carga en el sistema MOLAP. Una vez cargados los datos en la BDMD, se realizan una serie de cálculos para obtener datos agregados a través de las dimensiones del negocio, poblando la estructura de la BDMD. Luego de llenar esta estructura, se generan índices y se emplean algoritmos de tablas hash para mejorar los tiempos de accesos de las consultas. Una vez que el proceso de poblado ha finalizado, la base de datos multidemensinal está lista para su uso. Los usuarios solicitan informes a través de la interfaz y la lógica de aplicación de la BDMD obtiene los datos.

Por otro lado, en **ROLAP** se utiliza una arquitectura de tres niveles. La BD relacional maneja el almacenamiento de datos, el motor OLAP proporciona la funcionalidad analítica, y alguna herramienta especializada es empleada para el nivel de presentación. El nivel de aplicación es el motor OLAP, que ejecuta las consultas de los usuarios. El motor OLAP se integra con el nivel de presentación a través del cual los usuarios realizan los análisis OLAP. Después de que el modelo de datos para el DW se ha definido, los datos se cargan desde los sistemas transaccionales. Los usuarios finales ejecutan sus análisis multidimensionales, a través del motor OLAP, el cual transforma sus datos a consultas en SQL ejecutadas en las bases de datos relacionales y sus resultados son devueltos a los usuarios. La arquitectura ROLAP es capaz de usar datos precalculados (si estos están disponibles), o de generar dinámicamente los resultados desde la información elemental (menos resumida). Esta arquitectura accede

directamente a los datos del DW y soporta técnicas de optimización para acelerar las consultas como tablas particionadas, soporte a la desnormalización, soporte de múltiples reuniones, precalculado de datos, índices etcétera.

El término **HOLAP** hace referencia a soluciones híbridas de OLAP que combinan las arquitecturas ROLAP y MOLAP. En una solución HOLAP, los registros detallados (que suelen tener mayor volumen) se almacenan en la BD relacional, mientras que los agregados se gestionan en un almacén MOLAP independiente (Ibarzábal, 2003).

Finalmente, los cubos de datos en un Data Warehouse facilitan la agregación y el análisis de datos, y permiten explorar los datos a diferentes niveles de detalle gracias a las jerarquías dentro de cada dimensión. Estas jerarquías permiten ver los datos desde diferentes niveles de granularidad, desde los más detallados hasta los más generales, lo que es esencial para obtener conocimientos estratégicos a partir de los datos almacenados.

En resumen, la combinación de un diseño cuidadoso de la arquitectura de Data Warehouse, junto con el uso eficiente de modelos multidimensionales y sistemas OLAP, es clave para el análisis avanzado de datos y la toma de decisiones informadas en la era digital.