Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS

Obed Alejandro Espinoza Guevara
Asesorado por el Ing. César Rolando Batz Saquimux

Guatemala, noviembre de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

OBED ALEJANDRO ESPINOZA GUEVARA
ASESORADO POR EL ING. CÉSAR ROLANDO BATZ SAQUIMUX

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton De León Bran
VOCAL IV	Br. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. Marlon Francisco Orellana López
EXAMINADORA	Inga. Claudia Liceth Rojas Morales
EXAMINADOR	Ing. Álvaro Giovanni Longo Morales
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la Ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 23 de febrero de 2021.

Obed Alejandro Espinoza Guevara

Guatemala, 04 de Septiembre de 2021

Ingeniero
Carlos Alfredo Azurdia
Coordinador de Privados y Trabajos de Tesis
Escuela de Ingeniería en Ciencias y Sistemas
Facultad de Ingeniería - USAC

Respetable Ingeniero Azurdia:

Por este medio hago de su conocimiento que en mi rol de asesor del trabajo de investigación realizado por el estudiante **Obed Alejandro Espinoza Guevara** con carné **200610991 y CUI 2515 17152 0101** titulado **"Diseño de una Arquitectura de Datos utilizando las herramientas de AWS",** lo he revisado y luego de corroborar que el mismo se encuentra finalizado y que cumple con los objetivos propuestos en el respectivo protocolo, procedo a la aprobación respectiva.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. César Rolando Batz Saquimux
Colegiado No. 8549

César Rolando Batz Saquimux
Ingeniero en Ciencias y Sistemas
Colegiado No. 8549

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 11 de octubre de 2021

Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **OBED ALEJANDRO ESPINOZA GUEVARA** con carné **200610991 y** CUI **2515 17152 0101** titulado **"DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS",*** *realizado por el estudiante, OBED ALEJANDRO ESPINOZA GUEVARA aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Msc. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala,30 de noviembre de 2021

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

DTG. 724.2021

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **DISEÑO DE UNA ARQUITECTURA DE DATOS UTILIZANDO LAS HERRAMIENTAS DE AWS,** presentado por el estudiante universitario: **Obed Alejandro Espinoza Guevara,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

Guatemala, noviembre de 2021

AACE/cc

ACTO QUE DEDICO A:

Dios	Porque todo lo soy, he conseguido y seré es gracias a él.
Mis padres	Jaime Espinoza y Ruth Guevara, su amor y apoyo serán siempre mi inspiración.
Hermanos	Ivo y Cesar Espinoza, que siempre me han inspirado a superarme en especial Ivo que me instó a nunca darme por vencido.
Amigos	Por apoyarme en todo el proceso y motivarme a nunca darme por vencido.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser mi *alma máter* que me acogió y me inspiró a convertirme en un profesional.
Facultad de Ingeniería	Por darme las herramientas para convertirme en profesional.
Mis amigos de la Facultad	José Cerrato, Jorge Illescas, Christopher Santisteban, Daniel Muñoz y Jorge Ordóñez.
Ivo Espinoza	Que me convenció de nunca darme por vencido y completar mis estudios.
Cesar Espinoza	Por ser mi inspiración para estudiar en la universidad y cumplir mis metas.
Cesar Batz	Por apoyarme en el proceso de completar este Proyecto de Graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
$	Dólar
Tb	Es una unidad de almacenamiento de información equivalente a 1024 Gb
GB	Gigabyte
%	Porcentaje

GLOSARIO

API	Siglas del inglés *Application Programming Interface*, que se refiere a una interfaz de comunicación entre componentes de software.
Amazon	Utilizado para describir la organización, compañía o empresa de nacionalidad estadounidense encargada del comercio electrónico y servicios de *cloud computing* a diferentes niveles.
Amazon S3	Servicio de almacenamiento de objetos que brinda disponibilidad de datos, escalabilidad, seguridad y de igual manera rendimiento líderes en el sector.
Aplicación	Programa de cómputo diseñado como herramienta y preparado para una utilización específica.
Cache	Conjunto de datos copiados de otros originales, con la propiedad de que estos datos son de fácil acceso, a diferencia de los datos originales que son costosos de acceder en cuanto a tiempo.

Cloud computing	Computación en la nube, es un tipo de tecnología orientada a servicios que permite tener distintos de estos sin adquirir equipo físico, sino todo se ejecuta en un ambiente remoto**.**
CPU	*Central Processing Unit*, es el hardware dentro de una computadora u otros dispositivos programables, que interpreta las instrucciones de un programa informático a través de la realización de las operaciones básicas aritméticas, lógicas y de entrada salida del sistema.
Framework	Llamado marco de trabajo define en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
Infraestructura	Conjunto de elementos o servicios que se consideran necesarios para la creación y funcionamiento de una organización cualquiera.
Interfaz	Conexión física y funcional entre dos aparatos o sistemas independientes.

JavaScript	Lenguaje de programación interpretado orientado a objetos. Se usa enfocado a las páginas web.
JSON	Acrónimo de JavaScript *Object Notation*, el cual es un formato ligero de intercambio de datos y usa la sintaxis de JavaScript.
Máquina virtual	Software que emula el funcionamiento real de un equipo de cómputo.
Navegador	Programa que permite ver la información que contiene una página web.
Nodo	Punto de intersección o unión de varios elementos que confluyen en el mismo lugar. Por ejemplo: en una red de ordenadores cada una de las máquinas es un nodo, y si la red es Internet, cada servidor constituye también un nodo.
Plataforma	Sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

RAM	Siglas del inglés *Random Acces Memory*, es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados, es de acceso aleatorio pues se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes.
Redis	Motor de base de datos en memoria, basado en el almacenamiento clave/valor.
Servidor	Computadora que formando parte de una red, provee servicios a otras computadoras denominadas clientes**.**
Sistema operativo	Programa o conjunto de programas que efectúan la gestión de procesos básicos de un sistema informático.
Software	Sistema informático que comprende un conjunto de formas lógicas que hacen posibles tareas específicas.
SQL	Lenguaje estructurado de consultas utilizado para acceder datos en un gestor de almacenamiento.
XML	De sus siglas en inglés extensible *markup language*. Es un lenguaje para representar información estructurada en páginas web.

Diseñado para cualquier lenguaje y alfabeto, basado en texto.

RESUMEN

En el presente trabajo de graduación se dará a conocer los conceptos detrás de los procesos que existen para construir un repositorio de datos, se explica el razonamiento detrás de cada uno, de igual manera se explica el propósito y el contexto en el que se desarrolla, para poder explicar los objetivos de las soluciones que se pretenden obtener al aplicar los conceptos. Se define la importancia para las organizaciones el contar con centros de datos que permiten colectar, almacenar y analizar los datos de las organizaciones con el propósito de asistir en los procesos de toma de decisiones para garantizar que el potencial completo de una organización pueda ser alcanzado.

A partir de este punto se inicia a mencionar los conceptos de Big Data, y el pilar principal de este concepto que es el lago de datos. La manera en que este lago de datos ha venido a revolucionar la colección y almacenamiento de datos, y como este debe estructurarse para poder obtener el mejor provecho de la herramienta, de igual forma, como este se integra con diferentes herramientas de procesamiento de datos y análisis de datos.

También se presentan las diferentes arquitecturas de datos que han surgido en los últimos 15 años, sus objetivos y casos de uso, para concluir con la recomendación según la necesidad. Finalmente, se adjuntan las conclusiones, recomendaciones, así como las referencias bibliográficas que sustentan la investigación.

OBJETIVOS

General

Diseñar una arquitectura de datos utilizando las herramientas de AWS que aplique los conceptos modernos de arquitectura de datos y proveer una comparativa de herramientas para implementar los diferentes bloques conceptuales.

Específicos

1. Definir los conceptos modernos que constituyen una arquitectura de datos.

2. Analizar los diferentes tipos de arquitectura que proponen diferentes autores para determinar cuál es el óptimo según las necesidades.

3. Listar y describir las diferentes herramientas que AWS provee para administrar datos.

4. Describir cómo los bloques conceptuales de una arquitectura se implementan usando las herramientas de AWS.

INTRODUCCIÓN

El presente trabajo de graduación consiste en la recopilación de las ideas principales que motivaron la revolución de datos que inició a mediados de los años 80, con la invención del ya conocido Repositorio de Datos, el cual es reconocido en la actualidad como un pilar base en cualquier arquitectura de datos. Se presentan los conceptos detrás de esta solución y cuál era su caso de uso principal, así como las herramientas que Amazon AWS, provee para poder construir este tipo de soluciones.

A partir de este punto se detallará el catálogo de herramientas que Amazon pone a disposición de todos, así como una comparativa entre cada una de las diferentes herramientas, los casos de uso y esencialmente los costos estimados según soluciones estándar. Se explica cómo estas herramientas se pueden integrar para construir arquitecturas más robustas que permitan resolver problemas modernos, como lo son: el procesamiento, análisis de grandes cantidades de datos y la integración con la ciencia de datos.

Para luego adentrar en las diferentes arquitecturas de datos que existen actualmente para resolver la mayoría de las necesidades de datos de las organizaciones, donde se analizará la arquitectura tradicional, sus ventajas y desventajas, para luego compararla con las arquitecturas Lambda y Kappa, explicando los casos de uso, ventajas, desventajas y brindar una propuesta de implementación de cada una en un entorno en la nube con Amazon AWS. Finalmente se dará una propuesta de arquitectura hibrida que permita reusar las ventajas de las tres arquitecturas anteriores y así obtener una solución mucho más robusta.

1. CONCEPTOS BÁSICOS

1.1. Computación en la nube

No cabe la menor duda que los servicios en la nube que originaron el concepto de Computación en la Nube *Cloud Computing*, como se le conoce en inglés, cambiaron de manera permanente la forma en que las organizaciones utilizan los recursos de informática. Esta idea que comenzó con un concepto simple, el poder arrendar computadoras virtuales, conocidas como servidores, a un proveedor en internet, a un precio mucho más económico que le costaría a la organización, al comprar una computadora física que llenará dichos requisitos.

Así mismo, permitiendo a la organización ahorrar en costos que conlleva colocar una computadora física, para que esta sea el punto de acceso para los servicios propios de la organización hacia la web. Esto permitió a las organizaciones poder saltar al internet a bajo costo y cambió todo para bien.

Esto no solo fue explotado por pequeñas organizaciones, las organizaciones más grandes que deseaban saltar a la internet y que tienen requerimientos mucho más grandes, encontraron estos servicios mucho más económicos, que la inversión inicial en equipo físico que realizaron, a sí mismo como costos de mantenimiento y actualización, ya que estas tareas eran absorbidas por el proveedor de servicios en la nube.

Y así nació y se popularizó la computación en la nube, algo que actualmente ha crecido a niveles gigantescos, ya que ahora no solo existen muchos más

proveedores que en aquel entonces, cuando AWS era el pionero en esta área, que no solo proveen servidores en renta, ahora proveen todo tipo de servicios de informática, en lo que se denomina servicios autoadministrados.

Los servicios autoadministrados fue el siguiente paso en la computación en la nube, que ayuda en gran manera a las organizaciones, ya que simplificó la tarea de instalar, configurar y administrar servicios, como bases de datos, servidores web, almacenamiento de archivos, entre otros. En los cuales ahora los técnicos de informática no necesitan estar administrando y actualizando constantemente, ya que el proveedor se encarga de dicha tarea, bajo el mismo esquema de renta.

Para el caso específico de los servidores de base de datos, representó un gran cambio; ahora estos servicios autoadministrados, proveen una configuración base, actualización constante y copias de respaldo automáticas, lo que permite a las organizaciones reducir sus costos de mantenimiento aún más. Esto permitió crear un nuevo tipo de ecosistema de servicios, el cual consta de tres componentes principales:

- Consumidores de servicios: estos son los que consumen los servicios en la nube constantemente. Estos usuarios finales no necesariamente deben contar con los conocimientos de cómo los servicios funcionan de manera interna, solamente deben ser capaces de interactuar con estos servicios.

- Proveedor de servicios: consta de las organizaciones que prestan los servicios en la nube, estos servicios van desde servidores como tal, hasta servicios mucho más elaborados y específicos, para que los consumidores puedan utilizarlos de manera fácil.

- Diseñadores de servicios: estas corresponden a organizaciones que diseñan y construyen servicios que funcionan sobre los servicios que los proveedores brindan, ampliando sus capacidades, o bien, haciendo más fácil su uso, para que los consumidores finales puedan tener productos aún más específicos a su disposición.

1.1.1. Conceptos de la nube

La computación en la nube consiste en una serie de recursos compartidos, como lo son aplicaciones, almacenamiento, incluso desarrollo de aplicaciones, que se pone a disposición de las organizaciones a través de los recursos de internet. Lo cual permite a las organizaciones poder contratar infraestructura como un servicio, según sus necesidades y presupuesto. Entre las características principales que ofrecen los servicios en la nube, están:

- Estandarización: todos los servicios en la nube se ofrecen con una interfaz consistente, lo cual permite a las organizaciones puedan utilizarlos sin ningún problema.

- Automatización: es un proceso, que, basado en reglas de negocio específicas, permite la creación de recursos, despliegue de aplicaciones, entre otras cosas. La automatización es esencial en los servicios de la nube, ya que permite el uso eficiente de los recursos, así como, reducir la cantidad de personal que las organizaciones requieren para trabajar en sus aplicaciones.

- Elasticidad: es un beneficio grande, que la computación en la nube trajo a las organizaciones, ya que permite incrementar o decrementar los recursos de infraestructura que las organizaciones utilizan de manera automática o

manual. Esto representa un gran ahorro para las organizaciones, dado que les permite adaptar la cantidad de recursos según la demanda del momento, y así poder optimizar de gran manera los costos que estos servicios tienen. Algo muy difícil de lograr con infraestructura propia.

1.1.2. Modelos de servicio

Los modelos de servicio que ofrecen la mayoría de las plataformas en la nube se clasifican en tres, que son:

- Infraestructura como servicio
- Plataforma como servicio
- Software como servicio

1.1.2.1. Infraestructura como servicio

Consiste en proveer a los consumidores, lo equivalente una computadora física, corriendo en la nube, la cual cuenta con un sistema operativo, disco duro, servicios de red, entre otros. Para este servicio, los proveedores usualmente crean una máquina virtual, dentro de la infraestructura física que poseen, brindándole acceso únicamente a esta máquina virtual al consumidor.

Las ventajas de este tipo de servicios, es que el consumidor puede hacer uso del servicio de la forma que considere mejor. Al tratarse de una máquina virtual, se puede instalar y configurar cualquier tipo de software que el consumidor necesite, lo que le brinda mayor flexibilidad.

Este tipo de servicio se han popularizado de gran manera, ya que actualmente, los proveedores ofrecen al consumidor, la posibilidad personalizar

este servicio como deseen, desde incluir el tipo y cantidad de procesamiento deseado, memoria RAM, tipos y tamaños de disco dura e incluso interfaces de red, según la necesidad y el uso que el consumidor desee.

1.1.2.2. Plataforma como servicio

Este tipo de servicio consiste en infraestructura como servicio, pero agrega una capa de servicios intermedios que facilita la instalación y despliegue de cierto tipo de aplicaciones de software.

Lo que les permite a las organizaciones, contar un ecosistema consistente, para desplegar sus aplicaciones en la nube y reducir el tiempo que usualmente este proceso conlleva un ejemplo de plataforma como servicio, se puede utilizar el servicio de AWS, conocido como *"ElasticBeanStalk",* el cual es un servicio que permite agilizar el proceso de despliegue de un sitio web, ya que abstrae de tal manera dicho proceso, que les permite a las organizaciones, únicamente proveer una copia del código fuente del sitio web y el servicio se encarga de la configuración y despliegue del mismo, para ponerlo a disposición por los clientes o usuarios de la organización.

Lo que brinda a las organizaciones un ahorro en tiempos de despliegue y mantenimiento de servicios, ya que, en el modelo de plataforma como servicio, el proveedor se encarga de la administración de todos los servicios internos de los que depende dicho servicio.

Otro ejemplo que se puede mencionar que será de gran importancia en el análisis, son los servicios de AWS llamados RDS *Relational Database Service* por sus siglas en inglés, que ofrece base de datos relaciones como servicio, con el AWS desliga de la administración del sistema operativo y de la base de datos

como tal, o al menos a un nivel de infraestructura. Esto provee un servicio de bases de datos de manera consistente y lista para ser utilizada.

1.1.2.3. Software como servicio

Consiste en aplicaciones de software desarrolladas en un modelo que se conoce como “Multi Inquilino”, en el cual el mismo servicio, mediante una capa de autenticación o acceso, funciona para muchos usuarios de la misma organización, o bien incluso puede soportar múltiples organizaciones.

Un ejemplo popular de este tipo de servicios es el producto de Microsoft, Office 365, el cual provee todas las herramientas del producto Office original, pero estos ya no necesitan ser instalados previamente en la computadora del consumidor, más bien, mediante un acceso, se puede hacer uso de estas herramientas desde un navegador web.

1.1.3. El ciclo de vida de los recursos

Como ya se indicó, la idea detrás de la computación en la nube es que los consumidores utilicen los recursos computacionales que necesitan y cuando los necesitan. Estos servicios son cobrados de la misma manera, por lo tanto, el consumidor renta los recursos computacionales, según los modelos de pago que el proveedor establezca, siendo estos generalmente, por cobros mensuales de servicios consumidos. Dada la naturaleza que estos servicios tienen, los proveedores deben cumplir los siguientes requisitos.

- Los servicios de computación en la nube deben estar disponibles la mayor parte del tiempo, ya que ningún proveedor puede garantizar una disponibilidad del 100 %, pero al menos estos servicios deben tener una

alta disponibilidad.

- Deben ser capaces de responder inmediatamente a cualquier solicitud o cambio que los consumidores soliciten.

- Deben implementar un sistema de rastreo, que les permita saber qué servicios ha contratado el consumidor y qué uso le ha dado, esto para poder cobrar la renta de los servicios.

- Cuando el consumidor desactiva ciertos recursos, o bien, los reduce, el proveedor debe ser capaz de regresar dichos recursos a la fuente, para que estos puedan ser reasignados y utilizados por otros consumidores.

1.2. Arquitectura de datos

Según la RAE la palabra arquitectura se define como “La estructura física y lógica de los componentes de un computador.”[1] Otra definición interesante que provee la RAE es la de “Es el arte de proyectar y construir edificios.”[2]

Estos dos conceptos que dan una idea más clara de lo que se refiere a la Arquitectura de Software, la cual se podría definir, como el arte de diseñar y construir una estructura compleja de software, para cumplir un propósito específico.

[1] ARISER. *Historia del Big Data – Del comienzo del análisis de datos a nuestros días*. https://bigdataparacuriosos.wordpress. com/historia-big-data/. Consulta: enero de 2021.
[2] BARKER, Dan. *The Real Original Source of the Phrase Big Data.* http://barker.co.uk/bigdata Consulta: marzo de 2021.

Si bien la comparativa entre arquitectura y software fue realizada por el investigador Edsger Dijkstra en el año de 1968[3]. Pero el término de arquitectura de software se popularizó hasta la década de los años 1990, cuando se comenzó a trazar cierto paralelismo entre los patrones de diseño y construcción de edificios y casas, con los patrones identificables para diseñar y construir software. De ahí en adelante el término ha sido tan popularizado hasta el punto de no ser solo un concepto, incluso en la actualidad se trata de un rol importante que debe ser ocupado por ingeniero en sistemas con la experiencia necesaria.

El término arquitectura, como se observa, no es nada nuevo y se basa en aplicar planificación y conceptos que han probado ser efectivos, en la aplicación de modelos y bloques que resuelvan o faciliten la resolución de problemas comunes. Los principios básicos que tiene toda arquitectura de software son simpleza, reusabilidad y tolerancia al cambio. Estos principios permiten crear un producto robusto que se adapte a las cambiantes necesidades de una organización, sin dejar de cubrir las necesidades iniciales, con las cuales fue concebida.

1.2.1. Problema de diseño

Si bien todo diseño comienza con un problema en mente, y luego se plantea una solución, esto puede parecer sencillo a primera vista, pero el problema del diseño resulta ser complejo cuando se observa desde el punto de vista de que toda solución debe ser capaz de balancear las cuatro fuerzas principales que son:

- Económica: toda solución debe ser económicamente viable, es decir, que

[3] DIJKSTRA, Edsger. *The Structure of the Multiprogramming System*. https://www.cs.utexas.edu/users/EWD/ewd01xx/EWD196.PDF. Consulta: marzo de 2021.

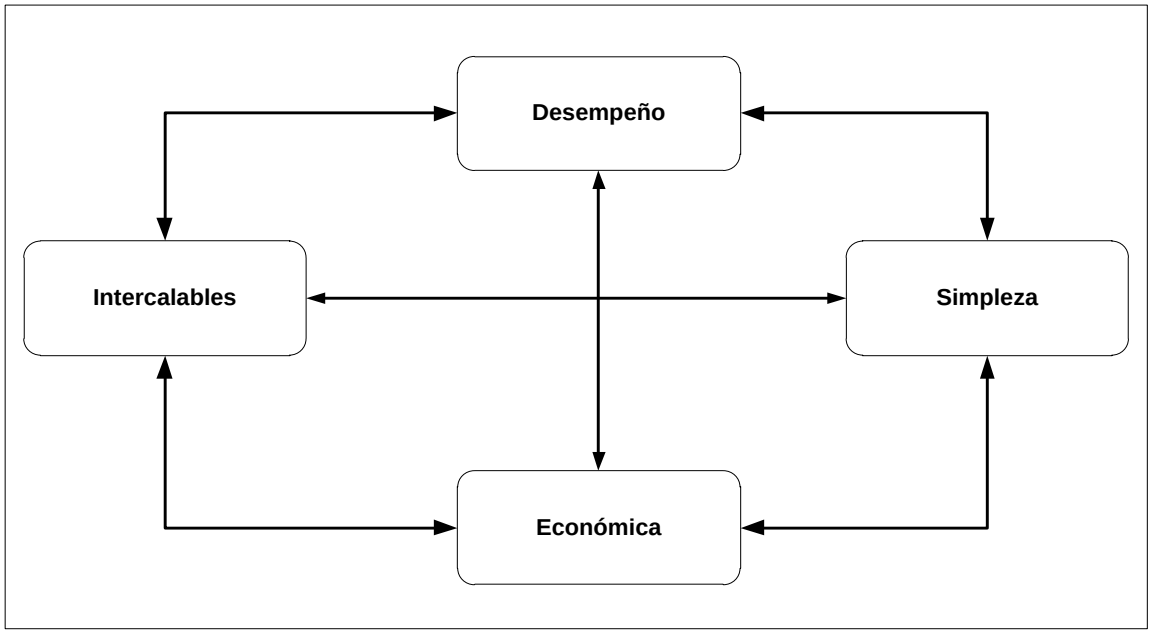
sus costos no deben exceder el presupuesto que la organización tiene para dicho problema, o bien, el costo no debe ser mayor a las ganancias que se pueden tener, producto de resolver el problema.

- Desempeño: se espera que el producto tenga un desempeño aceptable, tanto del producto final, como del proceso de construcción del mismo.

- Simpleza: entre más componentes o bloques conformen la solución, mayor será la complejidad de funcionamiento y construcción de la solución. Por lo que se recomienda incluir únicamente aquellos componentes esenciales para su construcción.

- Intercalables: los componentes deben poder conectarse de manera simple, es decir, que ningún componente debe sentirse forzado para integrarse con otro dentro de la solución.

Figura 1. **Problema de diseño**



Fuente: elaboración propia.

1.2.2. Patrones

Los patrones son conceptos aceptados y reconocidos para resolver un problema común, es decir, que los patrones son soluciones comúnmente aceptadas a problemas generales. Christopher Alexander en su libro *The Timeless Way of Building* publicado en el año de 1979, establecía la primera definición de patrones de arquitectura para la construcción.

Definiendo que cada patrón como “La relación de 3 partes entre un determinado contexto, un problema y una solución.” [4] En esta definición

[4] BBC. *La ciencia de encontrar perlas en el Big Data.* http://www.bbc.com/mundo/noticias/2013/03/130310_tecnologia_big_data_datos_informacion_perla_dp. Consulta: marzo de 2021.

Christopher Alexander, pretende establecer la abstracción que debe tener cada patrón para que pueda ser común a uno o varios problemas.

1.2.3. Arquitectura de la información

Charles Tupper en su libro de Arquitectura de Datos publicado en el año 2011, define la arquitectura de la información como “El diseño y la supervisión de la construcción de conocimiento derivado del estudio, la experiencia, instrucciones, o conocimiento de una situación específica, o bien una colección de hechos o datos.”[5]

Charles luego establece que el proceso de construir una arquitectura de información es sumamente importante para las organizaciones modernas, dado que actualmente el mundo gira en torno a los datos que son generados en todo momento. Estos datos son recolectados, almacenados y transformados, de manera que se puede obtener información acerca de ellos.

Si bien esto no parece ser tan complejo a primera vista, ya que el concepto de capturar, transformar y almacenar datos no es nada nuevo, el reto en la actualidad es poder obtener información, que no siempre se encuentra a simple vista, por lo que es necesario comprender el significado y la naturaleza de los datos, para poder realizar interpretaciones acertadas y de ahí poder obtener información significativa, que permita a las organizaciones entender el comportamiento del entorno en el que operan, para poder tomar decisiones acertadas.

[5] CANTABRIATIC. *NoSQL y el teorema de CAP*. http://www.cantabriatic.com/nosql-y-el-teorema-de-cap/. Consulta: diciembre de 2020.

Para poder transformar los datos en información, es importante saber cómo estructurar los datos, qué datos se están obteniendo y porque se están obteniendo, para poder analizarlos posteriormente, se debe poder proveer una manera sencilla, para los usuarios, de encontrar los datos que estos están buscando para poder obtener la información que buscan.

Finalmente, todos estos datos deben residir en un ecosistema que permita, no solo su almacenamiento, si no que permita que estos datos puedan ser extraídos o minados, según la necesidad, que puedan ser transformados, según reglas particulares de la organización, finalmente deben poder ser compactados para generar indicadores, los cuales son presentados de tableros de control, para que los altos mandos de la organización puedan leer e interpretar la información ya generada.

Ahora bien, los datos ya no solo se necesitan para poder brindar información del estado actual de la organización, ya que, con el crecimiento de la ciencia de datos, los datos se pueden utilizar para generar conocimiento, el cual permite a las organizaciones predecir comportamientos, o bien, anticiparse a futuras necesidades.

Con la creciente competencia, esto ya no es solo una limitante para organizaciones sumamente grandes, es esencial para cualquier organización que desee ser competitiva. Por lo que se observa, los ecosistemas de datos deben poder adaptarse a estas necesidades, es ahí donde los datos necesitan de un proceso de arquitectura, que permita a las organizaciones establecer patrones, que les permitan resolver los problemas de almacenamiento y procesamiento de los datos y así poder generar mucha mejor competencia.

1.3. Repositorio de datos

Desde sus inicios, todas las aplicaciones de software deben interactuar y manipular datos, a eso se le suma la necesidad de persistir dichos datos en algún dispositivo de almacenamiento.

Estos dispositivos de almacenamiento, conocidos como discos duros, eran costosos, y proveían almacenamiento limitado, por lo que los ingenieros de la época idearon estructuras que permitieran optimizar la forma en que se guardaban los datos en los medios de almacenamiento. Es aquí donde nacen las bases de datos y su popular modelo entidad relación o esquema de datos relacionales.

En el año de 1960, Charles W. Bachman, desarrollo lo que se conoce como el primer software de base de datos, el cual consiste en un producto de software que facilitaba a los programadores el manejo de la persistencia de datos, ofreciendo consistencia y confiabilidad en los procesos de almacenarlos, sin necesidad de que ellos explícitamente tuvieran que manejar dichos procesos.

Este nuevo concepto se popularizó a principios de la década de los años 1970, ya existían empresas como IBM, que habían entrado al mercado de los manejadores de bases de datos. Aquí se propuso la implementación del esquema de datos relacionales, enfocados en la optimización del espacio del disco duro, así como brindar a los programadores una herramienta generalizada para crear sus propias estructuras de datos con estos principios.

Es aquí donde se crea la normalización, la cual consiste en una serie de transformaciones que se aplican a los datos, para llevarlos a una estructura relacional.

Este *Framework* permite definir una serie de etapas, que van cambiando los datos de manera incremental, eliminado estructuras complejas y redundantes en favor de estructuras con identificadores, conocidos como llaves primarias, donde es el identificador de los datos el que es replicado a lo largo del modelo, con el objetivo de reemplazar los valores tipo carácter redundante, favoreciendo valores numéricos, los cuales ocupan mucho menos espacio.

Actualmente la normalización consta de hasta 5 etapas de transformación, llamadas formas normales, en la que en cada transformación existen reglas que los datos deben cumplir para poder indicar que los datos cumplen con la etapa o forma normal aplicada.

Desafortunadamente, el modelo relacional y las bases de datos relacionales, no proveía una solución aceptable para la nueva creciente demanda en los productos de software, que era, el análisis de datos, el cual requería poder generar análisis estadísticos, métricas, tendencias, entre otras, para poder brindar información importante a los dirigentes de una organización.

Esto ocasionó que las empresas comenzarán a utilizar estructuras de datos, un poco más amigables para esta necesidad, lo cual generó un dilema aún mayor, ya que esto provocaba que las organizaciones tuvieran que trabajar con estructuras de datos en paralela, es decir, que tuvieran los mismos datos, pero estructurados de manera más amigable, para los procesos de análisis.

Esto fue motivo de análisis de Richard Nolan, en su artículo *Managing the Crises in Data Processing* publicado en el año de 1979, donde él realiza un análisis profundo de esta creciente problemática en las organizaciones.

1.3.1. Toma de decisiones y datos

En el año de 1980, Peter Keen y Michael Scott-Morton[6], desarrollaron los conceptos para la toma de decisiones dentro de una organización, estableciendo que esta se clasifica en tres tipos de decisiones:

- Decisiones estructuradas: son aquellas que se toman basadas exclusivamente en los datos que tiene la organización y después de un análisis estadístico.

- Decisiones semi estructuradas: son decisiones que se toman asistidas por los datos de la organización, pero utilizando la intuición y conocimientos previos de los altos mandos.

- Decisiones no estructuradas: estas se toman únicamente basadas en la intuición y conocimiento de los altos mandos, sin utilizar ningún tipo de datos.

Asimismo, definió los tres componentes que se deben tener en una toma de decisión:

- Cuál es la pregunta que se está tratando de responder.
- Qué información o datos se necesitan para tomar una decisión.
- Qué acción se va a tomar basados en la decisión.

Al centrarse en la primera pregunta, se debe establecer cuáles son las incógnitas que se tienen en la organización, por ejemplo; cuáles son las ventas

[6] KEEN, Peter, MORTON, Michael. *Definition of DSS. Decision Support Systems: A Research Perspective*. Massachusetts: Massachusetts Institute of Technology. p. 50.

del último mes, a las personas le gustan los productos, entre otras. Es muy importante establecer esta primera incógnita ya que en el acto que sigue, se debe plantear qué datos se tienen a disposición que podrían permitir responder a esta pregunta.

Por ejemplo; las estadísticas de ventas del último mes, estadísticas de satisfacción de los clientes, entre otras, que permiten analizar la pregunta y darle una respuesta acertada. Esto conlleva a la toma de una decisión, la cual debe ir acompañada de una o varias acciones que se van a tomar para lograr ejecutar la decisión tomada.

Al comprender el esquema de la toma de decisiones, es claro que las organizaciones necesitan evolucionar, en cuanto al manejo de datos, para poder responder a las preguntas que el equipo administrativo plantea y así poder cambiar las decisiones de no estructuradas, a decisiones semiestructuradas o bien estructuradas. Es acá donde el repositorio de datos entra a resolver este problema, para los departamentos de tecnología.

1.3.2. El repositorio de datos

Es una colección grande de datos que una organización tiene a su disposición, y que se encuentra organizada de tal forma que pueda ayudar en el proceso de análisis y toma de decisiones. Un repositorio de datos se debe construir con la filosofía de que todos los datos que la organización pueda capturar se almacenen aquí, con la única premisa de que estos deben estar organizados de tal forma, que puedan ser accesados en el menor tiempo posible.

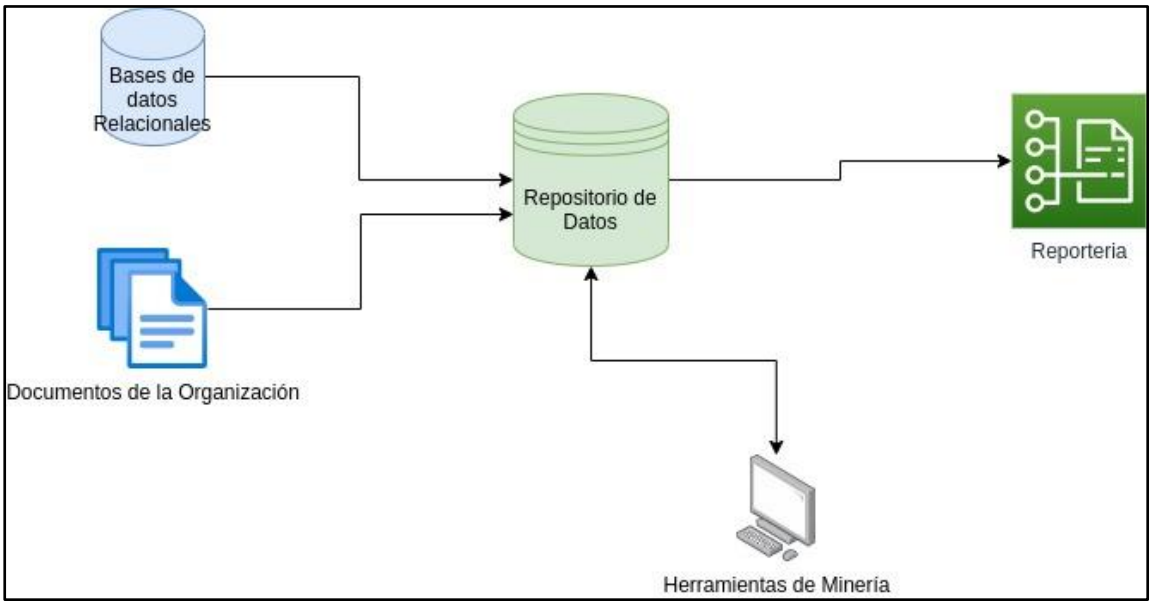
Los repositorios de datos deben tener los datos con diferentes niveles de abstracción, lo cual difiere de esquema de reportería, el cual brinda los datos en

un nivel de abstracción y granularidad específica, con esto los repositorios de datos proveen datos desde varias perspectivas diferentes en un mismo esquema, lo que facilita que este pueda resolver múltiples preguntas a la vez, facilitando así su mantenimiento, no obstante, su construcción no siempre es sencilla.

Charles Tupper, autor del libro *Data Architecture*, define que un repositorio de datos se compone de lo siguiente:

- Fuentes de datos: estas pueden ser, otras bases de datos relacionales, o bien cualquier tipo de datos externos que sean compartidos con la organización y sean relevantes.
- Herramientas de reportería.
- Herramientas de minería de datos.

Figura 2. **Repositorio de datos**



Fuente: elaboración propia.

1.3.3. Características

La primera característica que tiene el repositorio de datos es que difiere del esquema datos relacional, ya que aquí no es problema la redundancia de datos, por el contrario, se dice que un repositorio de datos debe guardar los datos desnormalizados, es decir, deshaciendo los cambios aplicados en el proceso de normalización, ya que esto eficiente el acceso de los datos, a costa de aumentar el almacenamiento.

La siguiente característica es, que los datos deben ser almacenados según el tema, el cual debe ser enfocado en los clientes, productos o cualquier actividad de la organización. Los datos almacenados deben ser no volátiles, lo que significa que los datos no deben cambiar a lo largo del tiempo, por lo que se

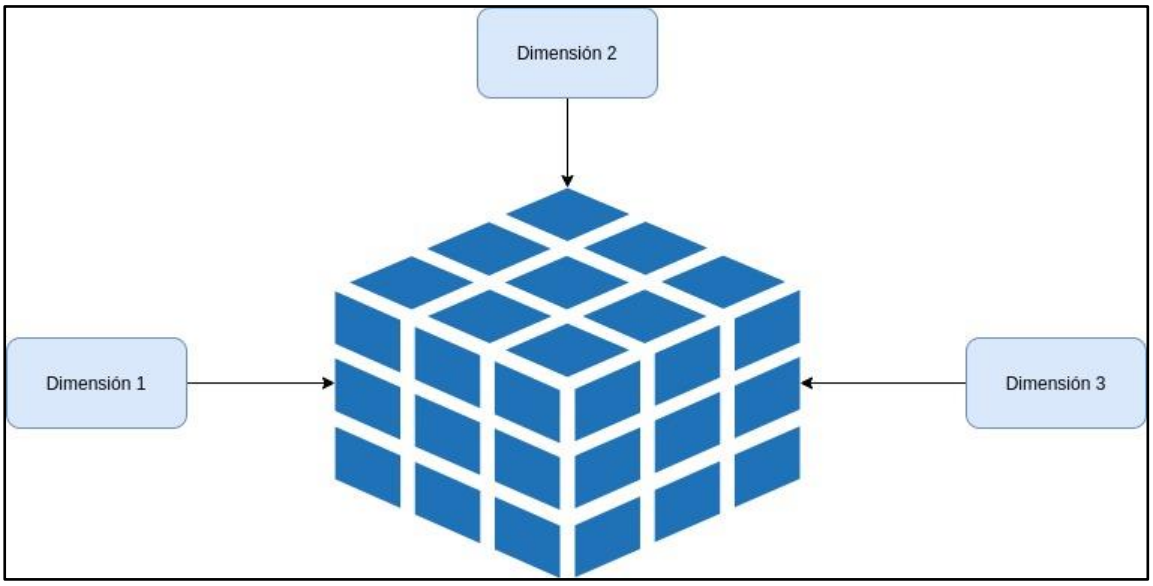
recomienda agregar datos que hagan correcciones, en lugar de modificar los datos previamente ingresados.

1.3.4. Modelo de datos

La primera característica que tiene el repositorio de datos es que difiere del esquema datos relacional, ya que aquí no es problema la redundancia de datos, por el contrario, se dice que un repositorio de datos debe guardar los datos desnormalizados, es decir, deshaciendo los cambios aplicados en el proceso de normalización, ya que esto eficiente el acceso de los datos, a costa de aumentar el almacenamiento.

La siguiente característica es, que los datos deben ser almacenados según el tema, el cual debe ser enfocado en los clientes, productos o cualquier actividad de la organización. Los datos almacenados deben ser no volátiles, lo que significa que los datos no deben cambiar a lo largo del tiempo, por lo que se recomienda agregar datos que hagan correcciones, en lugar de modificar los datos previamente ingresados.

Figura 3. **Modelo dimensional**



Fuente: elaboración propia.

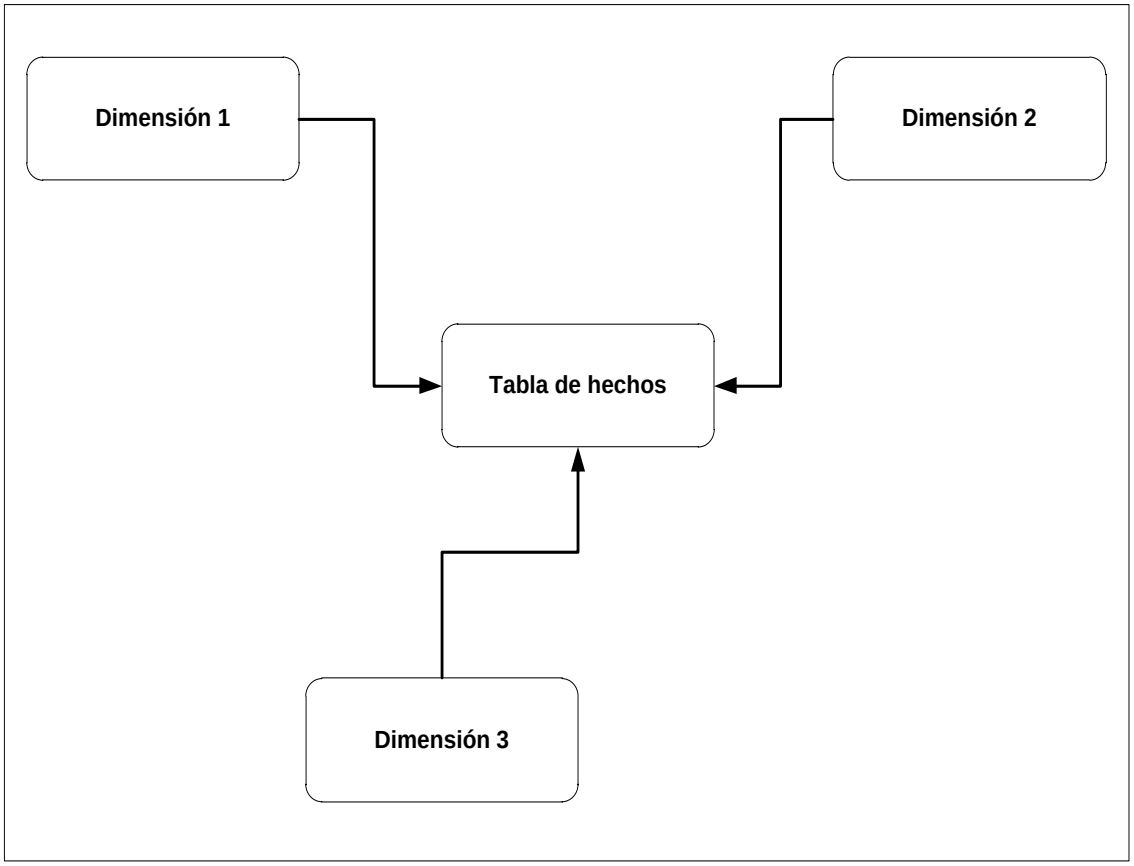
1.3.4.1. Modelo estrella

El modelo estrella es el más común que se utiliza para diseñar un esquema de datos dentro de un repositorio de datos, es el más utilizado dado que es el más sencillo y por ende el que mejores resultados da en términos de velocidad de acceso de los datos.

Para su implementación, se debe centrar en el tema o concepto base que se desea trabajar y todos los datos que se tengan sobre dicho tema se deben agregar en un esquema totalmente desnormalizado, es decir crear una tabla con muchas columnas, a esta tabla se le conoce como tabla de hechos, ya que es la que reúne o centraliza todos los hechos que la organización conoce sobre el tema en particular.

Luego de tener esta tabla de hechos, se procede a identificar aquellas columnas que proveen información que describen los hechos, las cuales para este modelo se le llaman dimensiones y se mueven a una tabla diferente, o tabla de dimensiones. Finalmente se obtiene una tabla de hechos en el centro relacionado con múltiples dimensiones, similar al esquema relacional, pero las relaciones únicamente tienen un solo nivel para optimizar la consulta.

Figura 4. **Ejemplo de modelo estrella**

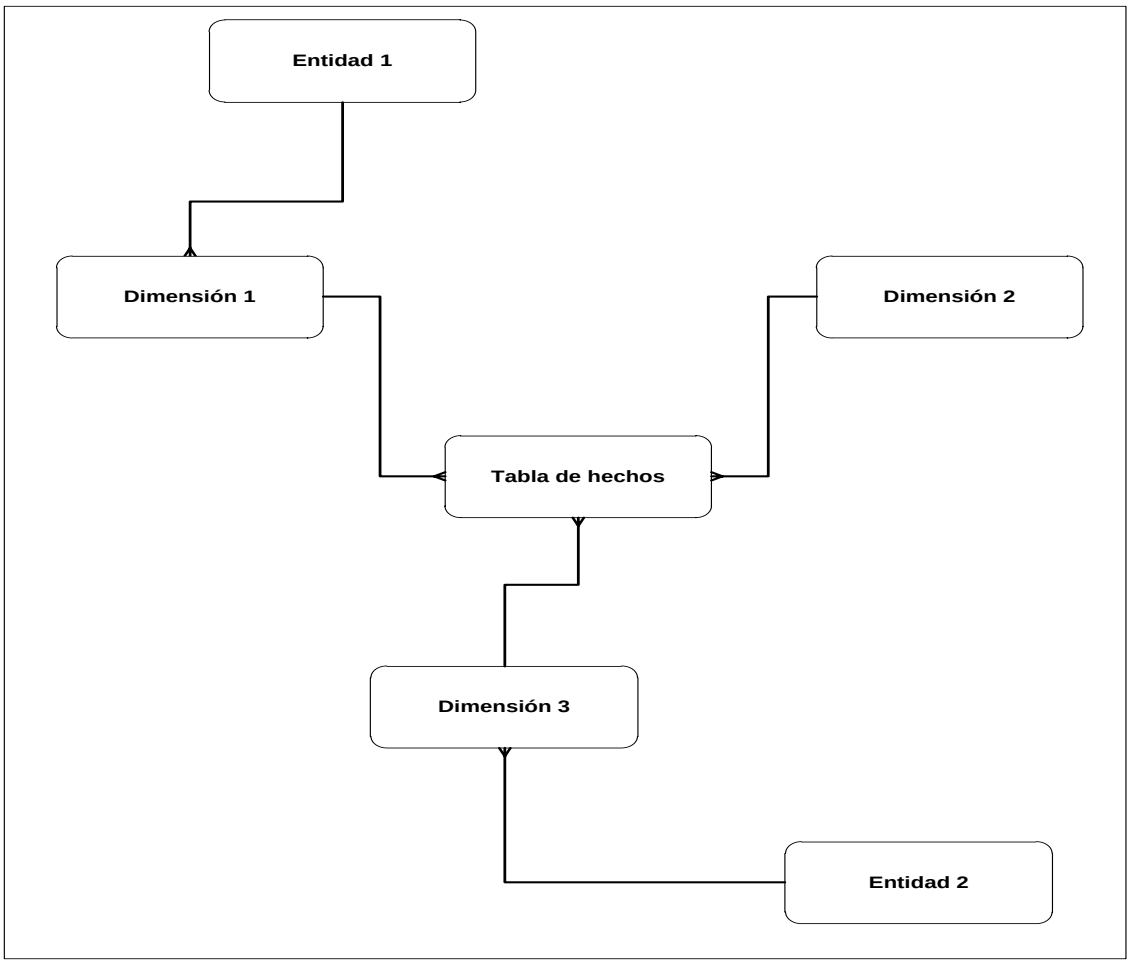


Fuente: elaboración propia.

1.3.4.2. Modelo copo de nieve

El modelo copo de nieve, se considera una variación del modelo estrella, el cual consiste en tener una tabla de hechos y sus respectivas dimensiones, pero este introduce cierta normalización sobre las dimensiones, con lo cual se asemeja un poco más al modelo relacional, no obstante, se considera que las dimensiones solo deben relacionarse con una sola entidad, esto para mantener el modelo no tan complejo en cuestiones de acceso.

Figura 5. **Modelo copo de nieve**



Fuente: elaboración propia.

1.3.5. Data Mart

El modelo copo de nieve, se considera una variación del modelo estrella, el cual consiste en tener una tabla de hechos y sus respectivas dimensiones, pero este introduce cierta normalización sobre las dimensiones, con lo cual se asemeja un poco más al modelo relacional, no obstante, se considera que las dimensiones solo deben relacionarse con una sola entidad, esto para mantener el modelo no tan complejo en cuestiones de acceso.

1.4. Lago de datos

A partir del año 2000, las organizaciones comenzaron a integrarse a las decisiones basadas en datos, probando ser la mejor opción para poder crecer a una escala mayor, con la inclusión de gigantes como lo son Google, Facebook, Microsoft, entre otros, que se percataron de esta revolución de datos y comenzaron a implementar decisiones basados en datos, volviéndose en la actualidad las organizaciones más grandes e importantes.

Actualmente las decisiones basadas en datos ya no son una opción, son una necesidad para cualquier organización que quieren mantenerse a flote y ser competitiva, es por ello que la administración de los datos evolucionó, de tener únicamente repositorios de datos y Data Marts, a tener bloques de datos mucho más grandes, los cuales no siempre es factible organizarlos a nivel que un repositorio de datos necesita. Es aquí donde entra un nuevo concepto, el lago de datos.

En octubre del año 2010, James Dixon el CTO de Pentaho *Hitachi Data Systems*, escribió en un artículo “Si tú piensas que un Data Mart almacena datos equivalentes a lo que es una botella de agua, es decir, purificado embotellado y listo para ser consumido, entonces un lago de datos es un gran cuerpo de datos en un estado natural. El contenido en un lago de datos consiste en una serie de archivos, sin un formato específico los cuales pueden ser examinados, analizados o tomados como ejemplo.”[7]

Dixon fue el primero en acuñar este término y brindar lo que en su opinión debía ser un lago de datos, así mismo, brinda una idea del porqué de su nombre,

[7] COX, Michael; ELLSWORH, David. *Application-Controlled Demand Paging for Out-of-Core Visualization.* p.12.

ya que compara a la Data Mart con una botella de agua y hace hincapié en el contexto de ser un pequeño extracto de un lago de datos mucho más grande.

Este nuevo concepto de lago de datos en un principio parecía ser algo que no era relevante para la mayoría de las organizaciones, pero la idea de Dixon, parecía ser una respuesta al nuevo concepto de Big Data, el cual nadie tenía claro en su momento.

Ahora bien, cómo se puede definir un lago de datos, Keith D. Foot en su artículo “Una breve historia de los lagos de datos”, provee la siguiente definición “Un lago de datos consiste en áreas de almacenamiento consolidado y centralizado, de datos crudos, es decir sin ningún tipo de procesamiento, estos son adquiridos de múltiples fuentes y carecen de un esquema definido.”[8]

Con esta definición, se puede ver uno de los conceptos que el lago de datos promueve que, se almacena ahora, analiza después, el cual cambia totalmente el concepto del repositorio de datos y Data Mart.

1.4.1. Big Data

No se puede referir un lago de datos sin tocar el tema de Big Data, el término fue creado por Roger Mougás, en el año 2005, para describir los volúmenes de datos que ahora se podían capturar con la introducción de la Web 2,0. En su descripción, se refiere a que estos volúmenes de datos eran casi imposibles de analizar con las herramientas de inteligencia de negocios de la época, lo cual requiere de la creación de nuevas herramientas que permitieran hacer dicho análisis.

[8] IBM. *What is cloud computing?* https://www.ibm.com/cloudcomputing/learn-more/what-is-cloud-computing/. Consulta: mayo de 2021.

Es en este mismo año que Yahoo introduce al mercado su popular herramienta llamada Hadoop, el cual consiste en un nuevo e innovador sistema de archivos optimizado para analizar y sumarizar archivos. Lo cual abrió la puerta a la captura de grandes cantidades de datos para luego ser analizados, finalmente en el año 2010 cuando el lago de datos es introducido, se comenzó a proveer un guía de como poder manejar los volúmenes de datos crecientes que cada organización debe adquirir y almacenar.

Doug Lanley definió el término de Big Data en función de lo que él llama las cinco V, que son:

- Volumen: define que Big Data consiste en una gran cantidad de datos que sobrepasan las capacidades tradicionales de análisis.
- Variedad: para referirse a un ecosistema de Big Data, existe una gran diversidad de datos de diferentes fuentes, que se almacenan en un solo lugar.
- Velocidad: define que, a pesar del volumen y la variedad, los datos deben poder analizarse de una manera rápida.
- Veracidad: define que los datos deben ser valores verdaderos, comprobables y finales, por lo que se recomienda que estos no sufran modificaciones.
- Valor: los valores dentro del ecosistema deben ser legibles, es decir, no se puede tener dentro del ecosistema valores o archivos ilegibles.

Como se observa Big Data es el concepto para describir los ecosistemas modernos y sus características, los cuales deben de estar preparados para lidiar con volúmenes de datos mucho mayores, y es aquí donde el lago de datos se vuelve una herramienta clave para toda implementación de Big Data.

No obstante, organizaciones cuyos datos no cuentan con las características para ser considerados Big Data, es recomendable siempre contar con una implementación de lago de datos, ya que este se ha vuelto un pilar importante de cualquier arquitectura de datos, sin importar sus características y objetivos.

1.4.2. Pantano de datos

Si bien la definición de un lago de datos dice que es un repositorio grande de datos en un estado inicial o crudo, este debe tener una organización base, de manera que sea fácil buscar y encontrar los datos que el usuario necesita. Cuando una organización implementa un lago de datos que carece de este requisito básico, se dice que los datos caen en un estanque sucio, del cual es complicado obtener los datos que se buscan, a este ecosistema desordenado se le conoce como “pantano de datos”, lo que significa que es un lago de datos sin documentación o gobernabilidad.

Si bien no es considerada mala práctica que el lago de datos tenga áreas de almacenamiento desordenadas, ya sea que se use como un bloque de análisis preliminar o bien un bloque temporal, cuando estas áreas crecen o se propagan, complicando el análisis de los datos, es síntoma de una problemática que si no se corrige puede provocar que el lago de datos sea difícil o imposible de utilizar.

1.4.3. Componentes de un lago de datos

Si bien construir un lago de datos pareciera, a primera vista, no ser un proceso complicado, la mayoría de los proyectos de construcción terminan fallando, o bien, con un pantano de datos en lugar de un lago de datos robusto y funcional. Es por ello que para construir un lago de datos se debe tener un enfoque en las fuentes, etiquetar los datos y crear una capa semántica que

permita su búsqueda y consumo de datos. Para poder implementar este enfoque un lago de datos debe contener al menos cinco bloques, que son:

- Ingestión e Integración de datos
- Persistencia
- Gobierno
- Análisis e inteligencia de negocios
- Ciencia de datos

1.4.3.1. Ingestión e integración de datos

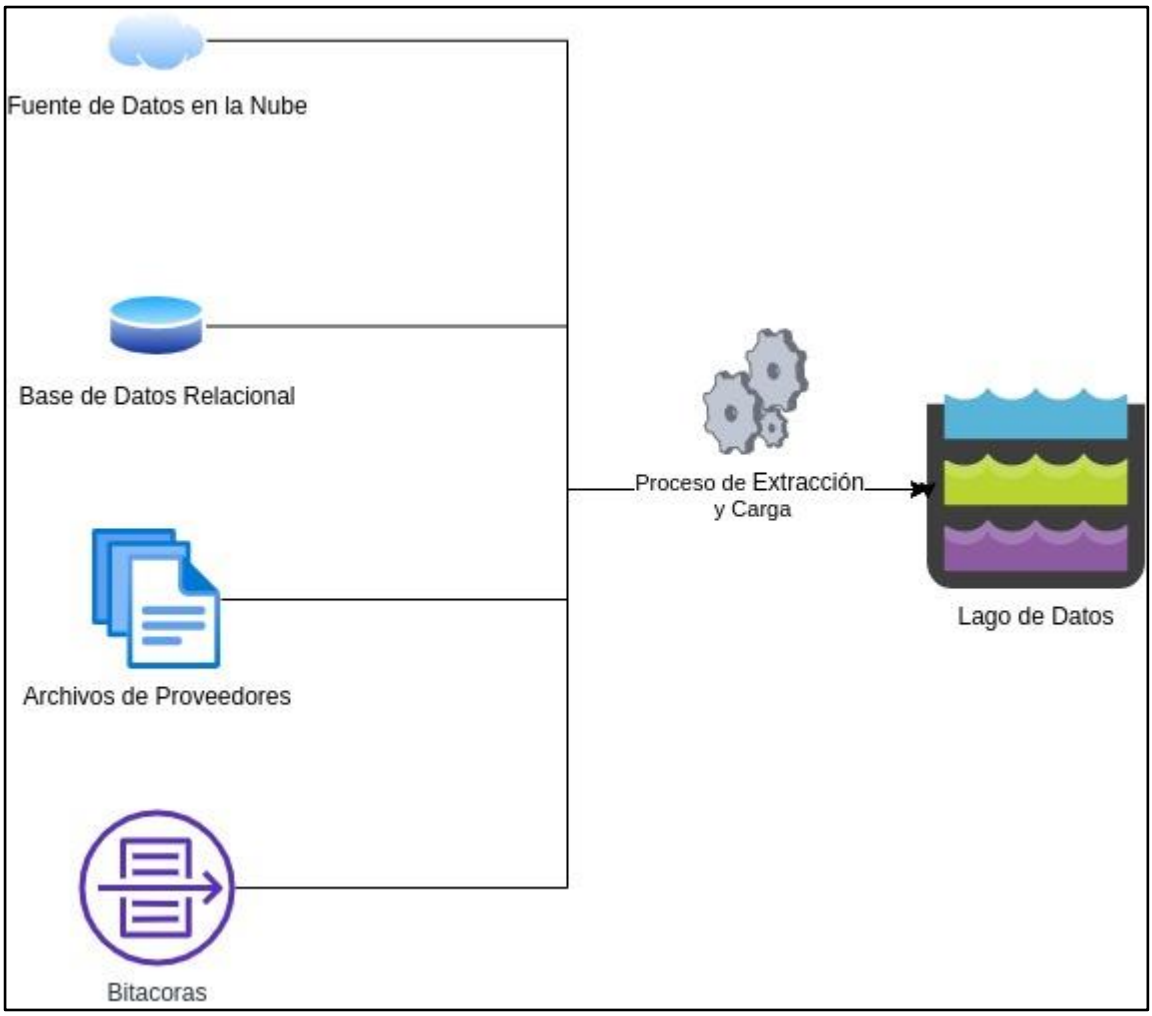
Tradicionalmente en un repositorio de datos, se utilizan las técnicas de extraer, transformar y cargar datos, o bien ETL, por sus siglas en inglés, con lo cual a partir de una fuente de datos construir un proceso que extraiga los datos que se necesitan, realice las transformaciones necesarias, para que estos datos se apeguen al esquema del repositorio de datos o bien el Data Mart, y finalmente estos datos se carguen.

En el lago de datos la filosofía es diferente ya que aquí se busca acumular la mayor cantidad de datos posible, sin importar su estructura, más bien se busca guardar los datos en su estado natural, es por ello que, para la ingestión de datos en el lago de datos, se utilizan procesos híbridos, que muchas consisten únicamente en extracción y carga, sin aplicar ninguna transformación.

Finalmente, los datos ingresados deben poder integrarse unos con otros para complementarse, este proceso es importante ya que, si se cuenta con datos que no se puede cruzar o complementar u otros, se dice que estos datos son aislados y usualmente tienen muy poco provecho. Estos bloques de integración

son usualmente campos de fecha, tipos de producto, o cualquier otro valor que permita asociar las diferentes fuentes entre sí.

Figura 6. **Proceso de extracción y carga**



Fuente: elaboración propia.

1.4.3.2. Persistencia

Una vez definida la forma de ingestar e integrar los datos, la persistencia de los datos debe tener un esquema de organización, que permita saber el estado de los datos, es decir, cuando los datos son recién cargados en el lago de datos, estos deben llegar a una zona donde se etiqueta como una zona de datos crudos, luego se deben construir procesos que, utilizando las reglas de integración, procedan a leer los datos, enriquecerlos y resistirlos en una zona diferente dentro del lago de datos.

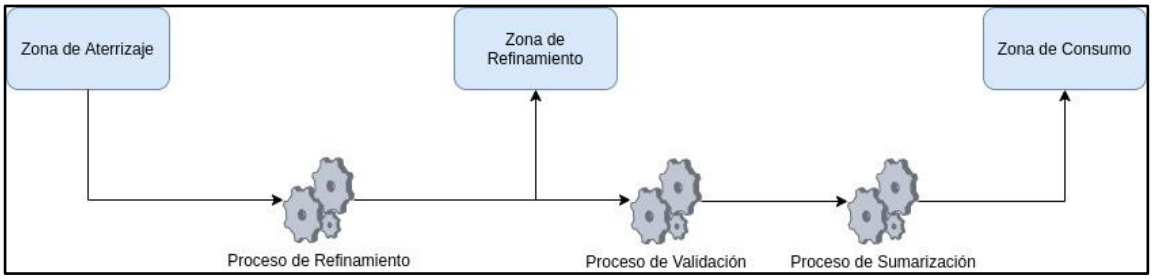
Esto se puede entender con una analogía de una planta procesadora de agua, donde primero se tiene un estanque donde toda el agua sucia llega, de ahí se comienza a mover el agua a diferentes tanques o compartimentos, donde diferentes procesos de limpieza y purificación del agua son aplicados, para que finalmente en la etapa final, se pueda tener agua apta para el consumo. Para un lago de datos se debe tener al menos tres zonas, basadas en el estado y enriquecimiento de los datos, estas son:

- Zona de aterrizaje: es la primera etapa de los datos, y debe ser una zona donde toda la Data Cruda sea etiquetada y almacenada. En distintas arquitecturas a esta se le llama zona de desarrollo o zona bronce y usualmente está destinada para usos de búsqueda y explotación de datos.

- Zona enriquecida: en esta los datos que llegaron a la zona de aterrizaje se extraen y se les aplican transformaciones, así como integraciones con otras fuentes de datos, para cumplir con el proceso de enriquecimiento de los datos. A esta también se le conoce como zona bronce, es usada generalmente por los científicos de datos o bien para realizar análisis sobre datos detallados.

- Zona de consumo: contiene los datos que ya fueron enriquecidos validados y transformados en su totalidad y se dice que se encuentran listos para ser consumidos por los usuarios de inteligencia de negocios para construir indicadores y tableros, para asistir la toma de decisiones, o bien modelos ya fabricados por los científicos de datos. A esta también se le puede llamar zona refinada o zona oro.

Figura 7. **Zonas y procesos en el lago de datos**



Fuente: elaboración propia.

1.4.3.3. Gobierno

Para que un lago de datos no se convierta en un pantano, este debe tener gobierno sobre los datos, es decir, debe contener las etiquetas, accesos, clasificación, entre otros, apropiados, es decir, la Data debe tener el contexto adecuado para favorecer el acceso a los datos de una manera simple y sencilla.

Todo esto se logra con la implementación de meta data, la cual describe los datos que son almacenados y su propósito, básicamente, dota de esquema y estructura los datos que su propósito inicial era no estar amarrados a un esquema en particular, si bien suena contradictorio, este proceso es necesario, ya que el

no realizarlo, convertiría el proceso de analizar y transformar los datos, mucho más complejo.

Si bien no existe una normativa o proceso estándar para aplicar gobierno a los datos, se consideran estos siete procesos básicos, para construir un gobierno sobre el lago de datos exitoso:

- Proveer herramientas de búsqueda y acceso sobre los datos
- Glosarios de negocio, flujo de datos, información y administración
- Políticas de definición y ejecución de gobierno
- Catálogo de transformaciones
- Administración centralizada de datos y su calidad
- Políticas de autoservicio para la preparación de datos
- Linaje sobre los datos y auditoría

1.4.3.4. Análisis e inteligencia de negocios

Una vez los datos ya fueron enriquecidos y validados, es decir, se encuentran en la zona de consumo dentro del lago de datos, estos ya pueden ser analizados por el equipo de inteligencia de negocios, encargado de construir los tableros necesarios para responder a las incógnitas del negocio y asistir la toma de decisiones.

Este bloque pretende que todos los datos que lleguen a esta zona pueden ser analizados, sin necesidad que estos estén en un esquema rígido como el del repositorio de datos, lo que les brinda flexibilidad a los analistas de realizar proceso de análisis e integración de datos en modo de autoservicio.

1.4.3.5. Ciencia de datos

Este bloque es el más reciente del lago de datos, y el que más relevancia está cobrando en la industria, ya que los procesos de toma de decisión son más complejos y demandan estructuras de datos mucho más complejas, lo cual demanda la implementación de procesos de máquinas de aprendizaje, *machine learning*, e Inteligencia artificial.

Los científicos de datos son los más beneficiados de la implementación de un lago de datos, ya que esto les permite poder buscar e integrar más fácilmente, nuevas fuentes de datos, sin estar atados al esquema rígido del repositorio de datos, lo que les brinda Data Cruda para que ellos puedan explorar.

Los científicos de datos, constantemente se encuentran explorando los datos en todas las zonas de la Data Lake, en búsqueda de valores, comportamiento y perspectivas que les permitan refinar los modelos de datos creados. Se dice que los datos cuentan la historia de lo que sucede en una organización, si esto es cierto, entonces los científicos de datos utilizando la historia, tratan de predecir el futuro.

1.5. Bases de datos columnares

Las bases de datos relacionales fueron creadas para resolver y optimizar el problema del almacenamiento, pero, como se ha mencionado, no son la opción ideal cuando se necesita hacer análisis de datos, especialmente análisis de grandes cantidades de datos, esto genera un problema para los motores de bases de datos relaciones, ya que estos almacenan registros completos de los datos.

Lo antes descrito, implica una gran cantidad de operaciones de lectura para las bases de datos, y esto da como resultado, consultas con tiempos de espera muy largos, o bien sobresaturación del uso del CPU, dada la cantidad de bloques de datos que deben mover a memoria. Es aquí donde las bases de datos columnares se convierten en una solución orientada puramente al análisis de datos y consultas muy grandes.

Las bases de datos columnares, como su nombre lo indica cambian el formato de almacenamiento tradicional, es decir, registros completos, por almacenamiento en columnas. Esto permite que el manejador de base de datos almacene bloques de datos de la misma tipificación, en un mismo bloque dentro del disco duro, con lo cual se leer únicamente los bloques de datos de las columnas que el usuario está solicitando en su consulta.

Así mismo, al contar los bloques con los mismos tipos de datos, se pueden introducir compresiones que optimizar la cantidad de datos dentro de un bloque. Estas optimizaciones en su conjunto reducen considerablemente las operaciones de lectura que debe realizar el manejador de base de datos, lo que resulta en consultas mucho más rápidas.

Otra característica que introducen las bases de datos columnares es que las columnas al no estar almacenadas sucesivamente, como en las bases de datos relacionales, se elimina la necesidad de meta Data dentro de los bloques de datos, lo que permite reducir la cantidad de información almacenada. Esto también reduce la necesidad de los índices, ya que cada columna tiene su propio almacenamiento, y esto lo convierte en un índice indirectamente.

Sumado a esto, los motores de bases de datos columnares modernos usualmente incluyen secciones de sumarización en los encabezados de los

bloques de almacenamiento, lo que les permite conocer rangos de valores de cada bloque, totales, valores únicos, entre otros; lo cual agrega una capa aún mayor de optimización sobre las consultas.

1.5.1. Compresiones

Como se indicó anteriormente, la compresión es un componente adicional de las bases de datos columnares que proveen una gran ventaja, sobre las bases de datos relacionales, ya que permiten reducir el tamaño de los bloques de almacenamiento, por lo que se optimiza la lectura de los mismos. Las compresiones más comunes en las bases de datos columnares son:

- Codificación de diccionario: este tipo de compresión asigna un Código, el cual puede ser una letra o número, a cada valor único de la columna, y reemplaza dicho valor por su código, lo que permite reducir considerablemente el tamaño de los bloques de almacenamiento.

- Compresión por recorte de espacios: recorta o elimina los espacios en blanco de los campos de tipo carácter, lo que permite reducir el tamaño del bloque de almacenamiento necesario para dicha columna.

- Compresión por valores de columnas: cuando en una columna existen muchos valores repetidos, esta compresión indica los números de registros en los que dicho valor se aplica, lo que permite únicamente almacenar dicho valor en el encabezado del bloque de almacenamiento y no en cada detalle.

- Compresión delta: cuando una columna guarda valores consecutivos, ya sean secuencias numéricas o de fechas, se puede almacenar únicamente

el primer valor, y los siguientes valores únicamente el cambio o delta del segundo valor con respecto del primero, y así sucesivamente.

Si bien estas cuatro compresiones son las comunes los manejadores de bases de datos columnares pueden tener implementaciones propias de diferentes tipos de compresiones, es importante conocer cómo funcionan, antes de aplicar las compresiones, ya que, según los tipos de datos y sus posibles valores, puede ser que existan compresiones que den mejores resultados, es por ello que se debe analizar con cuidado la naturaleza de los datos de columna para poder determinar la compresión adecuada.

1.5.2. Desventajas

Si bien las bases de datos columnares proveen una solución mucho más robusta para realizar análisis de datos, en bases de datos de tamaños considerablemente grandes, es necesario resaltar las siguientes desventajas que tienen al ser comparadas con las bases de datos relacionales:

- Al almacenar los datos por columnas, operaciones de inserción y actualización tradicionales, suelen demorar más de lo que toman en una base de datos relacional. Por lo que se dice que no deben utilizarse en requerimientos de datos transaccionales, más bien, datos con propósitos analíticos con poca tendencia a cambios.

- Dado el almacenamiento columnar y procesos de compresión, estas bases de datos obtienen un mejor tiempo de respuesta si los datos se encuentran ordenados, lo que implica que se debe hacer un ajuste o mantenimiento sobre las tablas con frecuencia para evitar el degradé de los tiempos de respuesta. Si bien esto también es recomendado en bases de datos

relacionales, el degradé que pueden sufrir estas bases de datos puede ser más frecuente.

- Si se compara los tiempos de respuesta, con datos relativamente pequeños, es decir tablas que cuentan con pocos miles de registros, los tiempos de consulta son similares a los tiempos de respuesta de las bases de datos relacionales, dado que ese no es el propósito de las bases de datos columnares.

1.6. Bases de datos clave valor

Son las bases de datos más sencillas de la familia de las NoSQL, dado que las bases de datos NoSQL, obtienen su nombre del hecho de que no utilizan estructuras basadas en filas y columnas, por lo que no utilizan el lenguaje SQL como una manera de interactuar con dichas bases de datos, aunque ya existen muchas de ellas que implementan su propia versión del lenguaje, para ofrecer a los desarrolladores una transición mucho más sencilla.

Las bases de datos clave valor, provienen de las estructuras homónimas de los lenguajes de programaciones, en el que permite almacenar un objeto complejo y este es accedido mediante el uso de una llave única que lo identifica.

El primer ejemplo que se puede dar sobre este tipo de estructuras son los arreglos, los cuales consisten en un conjunto de valores asociados a una misma variable, donde se puede acceder a un valor en particular utilizando el índice en el que se encuentra. Por ejemplo: si se tiene un arreglo de nombres, se vería de la siguiente manera:

Tabla I. **Ejemplo de almacenamiento clave valor**

Índice	Valor
0	Pedro
1	Juan
2	Manuel
3	María
4	Ana
5	Sofia

Fuente: elaboración propia.

Si este arreglo se encontrara asignado a la variable Nombres, si se quiere obtener el nombre Juan, bastaría con que se seleccione el índice o posición en la se encuentra dicho valor, lo cual se hace indicando el nombre de la variable, seguido de paréntesis o corchetes indicando el índice, lo cual sería Nombres [1]. Es una práctica común en los arreglos en la mayoría de los lenguajes de programación, iniciar los arreglos en la posición 0 en lugar de la posición 1.

La estructura de los arreglos es muy conveniente ya que acceder a posiciones específicas del arreglo es algo que se puede realizar sumamente rápido, la desventaja del uso de los arreglos es que los índices deben ser estrictamente numéricos, y todos los valores del arreglo debe tener el mismo tipo, en el ejemplo anterior se definió Nombres como un arreglo de tipos de datos cadena de caracteres, por lo que todos los valores que se asignen al arreglo deben ser de ese tipo.

Así mismo, los arreglos usualmente tienen tamaños definidos, por lo que, para incrementarlos, se deben crear una variable con un tamaño mayor. Luego se decidió crear una estructura que generaliza el concepto de los arreglos, pero

que brinda flexibilidad en los índices y los valores, es decir, que los índices pudieran ser de cualquier tipo de dato al igual que los valores, con la restricción de que los índices deben ser únicos al igual que en los arreglos, así nacieron los tipos de datos conocidos como Diccionarios.

Dependiendo del lenguaje de programación, los diccionarios brindan la flexibilidad que los arreglos no, y al permitir guardar cualquier tipo de valor, su uso se popularizó. Siendo uno de sus principales usos el de caché. El caché se volvió algo necesario para cualquier aplicación moderna ya que permite guardar valores de acceso recurrente en la memoria de la computadora y al utilizar tipos de datos como los diccionarios, esto se volvió muy fácil de implementar para los desarrolladores.

El problema es que este tipo de estructuras reside únicamente en la memoria de la computadora que creó inicialmente los valores, por lo que, al comenzar a crear aplicaciones distribuidas, es decir, que funcionaran en múltiples computadoras, la necesidad de poder crear y acceder a estas estructuras desde cualquier computadora se dio.

Es aquí cuando las bases de datos Clave Valor nacen, ya que su principal punto de enfoque fue brindar a los desarrolladores la habilidad de persistir las estructuras de tipo diccionario en una especie de base de datos diseñada específicamente para funcionar exactamente como un diccionario.

A partir de ese momento las bases de datos clave valor, existían únicamente como un cache centralizado, por lo que los valores almacenados existían únicamente en memoria y un tiempo definido, pero al crecer su popularidad, estas evolucionaron hasta el punto de persistir la información no solo en memoria.

También en el disco duro, en la actualidad incluso se ofrecen soluciones de alta disponibilidad que ofrecen almacenamiento redundante que garantiza que la información será persistida y no se perderá al servicio ser reiniciado, todo esto sin perder la velocidad con la que usualmente son capaces de retornar a las consultas.

1.6.1. Características

Entre las características de las bases de datos clave valor, se puede mencionar:

- Simpleza: ya que su uso es sumamente sencillo, se debe generar una llave única que identifique que el objeto que se desea al amanecer y listo, para acceder al objeto nuevamente solo basta con solicitarlo.

- Velocidad: al contar con llaves únicas, el acceso a los objetos es sumamente rápido, por lo que el agregar y consultar objetos en estas bases de datos ayuda a acelerar los tiempos de respuesta de las aplicaciones.

- Escalabilidad: al incluir alta disponibilidad y redundancia permite que estas bases de datos pudieran escalar a grandes cantidades de información, sin perder la velocidad y simpleza de su funcionamiento.

- TTL: tiempo de vida, por sus siglas en inglés *Time to Live,* se refiere a que permiten definir un tiempo de expiración para los objetos, con lo cual estos son borrados una vez este tiempo es alcanzado.

1.6.2. Ejemplos

Entre las bases de datos más populares se pueden mencionar:

- Memcache: es una base de datos clave valor, utilizada generalmente como caché, ya que no persiste los datos en disco, manteniéndolos únicamente en memoria.

- Redis: se creó inicialmente como una alternativa a Memcache, y aunque inicialmente no ofrecía persistencia en disco, ha evolucionado hasta ofrecer alta disponibilidad en el almacenamiento de los datos.

- Couchdb: creada como una base de datos especialmente para aplicaciones móviles, y aunque inicialmente basada en documentos, se podría decir que es una base de datos Clave documento, ya que todos deben ser definidos con una clave única.

- DynamoDb: es la base de datos clave valor por defecto de AWS, y aunque ofrece persistencia de alta disponibilidad sus tiempos de respuesta son muy similares a los de Redis y Memcache utilizando únicamente memoria.

1.7. Datos en tiempo real

Los datos en tiempo real, o bien lo que se le conoce como transmisión de datos en tiempo real, es un concepto que ha cobrado mucha relevancia en los tiempos actuales, y es considerado uno de los principios básicos de los datos de gran volumen.

Consiste en poder generar, capturar, procesar, analizar y almacenar datos o bloques de información en cuanto estos son generados. Se diferencian de los procesos tradicionales de batch, mini batch o incluso micro batch, en el que estos datos son generados constantemente, en lo que comúnmente se compara a un flujo constante de agua que pasa por un sistema de tuberías, es por esto que los sistemas que se construyen para procesar datos en tiempo real difieren bastante de los sistemas de procesamiento de datos tradicionales.

Los datos en tiempo real se componen de bloques, que representan un evento en particular, el cual es de interés para el sistema y el negocio, este evento debe contener la suficiente información para poder ser analizado de manera individual y aislada y poder ser capaces de interpretar el contexto del que se generó, es por ello que cada bloque individual se le llama evento de datos.

Los eventos generados en un sistema de transmisión en tiempo real poseen tres características principales que los distingue de cualquier otro sistema, en los cuales radica muchas veces la complejidad que tienen y las decisiones que se deben tomar para poder procesarlos de una u otra manera, estas características son:

- Se generan constantemente: como ya se estableció los datos en tiempo real son similares a los flujos de agua, son constantes, por lo que se debe estar preparado para procesarlos todo el tiempo. Esto se vuelve crítico ya que los procesos de validación o enriquecimiento de datos no pueden demorar más del tiempo que se tarda en producirse la data.

- Estructura variable: dado que los datos son generados por múltiples fuentes, es común que los eventos generados tengan estructuras variables a lo que comúnmente se trabaja con otro tipo de sistemas, aún se definan

estructuras base para todas las fuentes de datos es común que cada bloque se le agregue bloques de contexto variable, es aquí donde la ideología almacenar primero y analizar después resulta más apropiada para este tipo de sistemas.

- Alta cardinalidad en el almacenamiento: la cardinalidad se refiere a la cantidad valores únicos dentro de los datos, por lo que se considera como la cardinalidad que tienen las diferentes dimensiones que componen los eventos. Usualmente a esta cardinalidad alta de dimensiones o combinación de dimensiones se le conoce como rasgos de cola larga, esto basado en la curva estadística del mismo nombre, que representa la variabilidad de los datos. Si bien esto es común en todos los sistemas de datos, en los datos de transmisión en tiempo real es un problema aún más difícil de resolver, dado la complejidad que resulta en homogeneizar los valores en cuanto los eventos son generados.

1.7.1. Arquitectura de datos en tiempo real

Construir una arquitectura de datos en tiempo real, no es una tarea sencilla, ya que se debe tener consideración múltiples componentes que deben ser capaces de interactuar entre sí, con tiempos de respuesta muy bajos, para garantizar que no existan atrasos innecesarios que puedan formar cuellos de botella.

Los cuellos de botella en una arquitectura de datos de este tipo, puede llegar a ser aún más catastrófico que en cualquier otro tipo de sistema, ya que si los datos son procesados y analizados mucho más lento de lo que son generados y consumidos, esto puede provocar colas de procesamiento muy grandes y por ende el colapso del sistema en pocos minutos.

Para ello es importante definir los componentes más comunes que constituyen una arquitectura de datos en tiempo real, tomando en cuenta que, dependiendo de los casos de uso y las necesidades de cada organización, puede ser que existan componentes adicionales o bien componentes que no sean necesarios.

1.7.1.1. Eventos de datos

Los eventos son bloques de datos que tienen significado para una organización, estos eventos son generados desde diversas aplicaciones y son enviados a través de la red, al sistema de procesamiento de datos, para que aquí comience su flujo hacia los sistemas de reportería y almacenamiento.

Dado que estos eventos tienen estructuras variables, como se estableció anteriormente, además de que es común que cada evento sea enriquecido con diferentes tipos de contexto, el formato usual o preferido por la mayoría de los desarrolladores para generar este tipo de eventos, es el formato de notación de objetos en JavaScript o JSON por sus siglas en inglés. Este formato da la flexibilidad de adaptarlo a cualquier estructura, así como la libertad de poder enriquecer fácilmente los eventos con más información.

Este formato se creó a principios de los años 2000, y se popularizó a mediados de esa década, ya que era un formato mucho más liviano que el entonces popular lenguaje de marcado extendido o XML por sus siglas en inglés, esto debido a que el JSON era casi 4 veces más liviano que el XML, lo que resultaba en menor cantidad de datos a transmitir, por ende, transmisiones mucho más rápidas.

Finalmente, en la década de los años 2010 y con la necesidad de mejorar aún el tiempo de transmisión de datos, nacieron dos formatos nuevos el formato Thrift y el Formato ProtoBuf, ambos inventados por Facebook y luego adoptados y popularizados por Google, los cuales era una versión binaria y comprimida del formato JSON, lo cual permite reducir aún más el tamaño de los eventos de datos generados y transmitidos.

1.7.1.2. Flujo de datos

El flujo de datos representa el recorrido que los eventos deben realizar desde que son generados hasta su destino final, este flujo debe ser construido de manera modular, de manera que permita reemplazar u optimizar diferentes componentes para asegurar que los eventos son procesados a la velocidad necesaria, así como detectar de una manera más sencilla los fallos o problemas en el flujo. Los componentes de los flujos de datos se estructuran a su vez de cuatro bloques principales que son:

- Grifo: continuando con la analogía del flujo de agua, el primer componente llamado grifo se refieren a la fuente de eventos constantes, que, al estar activo, funciona como un grifo abierto que deja pasar todos los eventos que son generados.

- Flujo de eventos: se refiere a los datos constantes que pasan por el grifo y que deben ser procesados.

- Procesamiento: esta estación se refiere a una o múltiples operaciones que se deben realizar sobre los eventos antes de que estos sigan su recorrido.

- Fregadero: se refiere a la salida del flujo donde desemboca el flujo de datos

luego de haber sido procesado, este fregadero a su vez puede estar conectado a otro grifo el cual inicia otro flujo de datos que utiliza los eventos que ya fueron procesados por el flujo anterior.

Utilizando estos cuatro bloques se pueden construir diferentes estaciones de procesamiento de datos, cuyo objetivo es recibir el flujo de eventos y llevarlo hasta su destino final, el cual puede ser el almacenamiento o bien las aplicaciones de análisis.

1.7.1.3. Procesamiento

El procesamiento de datos se refiere a las diferentes operaciones que se deben realizar sobre los eventos de datos, para que estos puedan ser analizados posteriormente, entre los procedimientos más comunes están:

- Limpieza y homogeneización: los eventos al no tener estructuras rígidas, debe existir un proceso que permita limpiar y homogeneizar los datos, para garantizar que estos son correctamente interpretados o bien.

- Transformación: consiste en extraer valores específicos de los eventos, los cuales son de interés para un análisis en particular, esto puede implicar que, para un mismo evento, existan múltiples estaciones de transformación, ya que de un mismo evento se pueden analizar múltiples contextos para diferentes propósitos.

- Enriquecimiento: muchas veces los eventos por sí mismo son difíciles de utilizar o interpretar, por lo que es necesario enriquecerlos con más información. Esto se logra al agregar descripciones sobre ciertas dimensiones del evento.

- Selección y reducción: conocido popularmente como Map Reduce, en inglés, se refiere al proceso de extraer campos y sumarizar los, con el objetivo de proveer estadísticas generales sobre los eventos generados, por ejemplo si uno de los objetivos es conocer la cantidad de eventos generados por un determinado grupo de usuarios, los procesos de selección y reducción, deben extraer los campos que identifiquen a dichos usuarios y luego sumarizar los o contabilizarlos de manera que puedan dar estadísticas generales sobre dichos eventos.

Cada evento generado y que es ingresado al flujo de datos puede pasar por uno a múltiples tipos de proceso, esto se define según la necesidad que se tenga de análisis sobre los eventos, esto debido a que de un mismo tipo de evento se pueden generar múltiples estadísticas de interés y que analizarlas todas en un mismo bloque de procesamiento puede resultar en flujos de datos lentos o costosos, por lo que es común segmentarlos en procesos químicos para garantizar la velocidad y no generar cuellos de botella.

1.7.1.4. Almacenamiento

Las opciones para el almacenamiento de los eventos en un sistema de transmisión de datos en tiempo real son diversas, siendo las bases de datos relaciones y columnares las menos utilizadas para ello, esto debido a que estos sistemas buscan tener el mejor tiempo de respuesta posible, por lo que garantizar la consistencia del almacenamiento no es el principal requisito.

Es por ello que es más común utilizar bases de datos NoSQL, además de que la mayoría de ellas utilizan el formato JSON como formato principal de almacenamiento. Esto no quiere decir que las bases de datos relacionales no se puedan utilizar para estos propósitos, en la actualidad han mejorado muchísimo

en la velocidad de inserción de datos, así mismo como la inclusión del soporte del formato JSON, lo cual les permite seguir siendo una opción para tomar en cuenta.

Otra ventaja que proveen las bases de datos NoSQL y en especial las bases de datos Clave Valor es la habilidad de definir un tiempo de vida para los eventos, lo que permite automáticamente expiarlos cuando estos ya no son necesarios, esto es importante ya que un sistema de datos en tiempo real los tiempos de respuesta son lo más importante.

Además, generalmente se le da prioridad a la Data recién llegada, es decir, no es usual realizar análisis de los datos que ya llevan mucho tiempo atrás, únicamente los datos generados durante el día, esto permite también, no sobrecargar el almacenamiento de la base de datos y mantener los tiempos de respuesta.

Finalmente se recomienda que estos lleguen al lago de datos, en sus diferentes estados, es decir, estado inicial y sin transformar, hasta los datos ya sumarizados, ya que aquí donde se pueden realizar análisis de datos históricos, o bien, para que los científicos de datos puedan utilizar los eventos.

1.7.1.5. Presentación de los datos

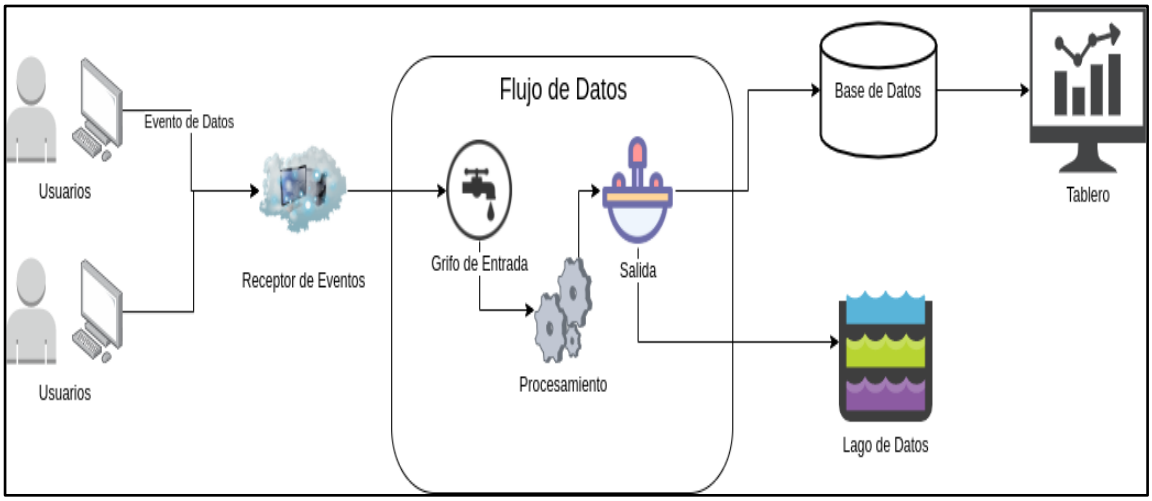
Este componente es uno de los más importantes, ya que refleja como los datos van a ser presentados a los usuarios finales, usualmente se construyen tableros, que muestran diferentes indicadores y los resultados de los eventos sumarizados, que permitan entender el comportamiento del sistema minuto a minuto, por ejemplo, mostrar la cantidad de ventas totales generadas, el monto

de las ventas generadas, la cantidad usuarios en el sistema, la demografía de los usuarios actuales, entre otras estadísticas.

Es importante que estos tableros pueden ser actualizados automáticamente para que sea perceptible el cambio de los datos en cuanto sucede, esto aquí donde la sensación de tener datos en tiempo real es perceptible.

Considerando que, en la realidad, siempre existe un retraso corto desde el momento que se generaron los datos hasta que estos son prestados, es por ello que en sus inicios estos sistemas eran llamados sistemas de tiempo casi real, y se tomaba como aceptable hasta cinco minutos de retraso en el que el evento se generaba, hasta que era prestando en el tablero general.

Figura 8. **Arquitectura de datos en tiempo real**



Fuente: elaboración propia.

1.8. Procesos de Extracción, Transformación y Carga

Los procesos de Extracción, Transformación y Carga o ETL por sus siglas en inglés, son la forma por defecto de integrar fuentes de datos a los Repositorios de datos y los lagos de datos, básicamente consiste en extraer los datos que se necesitan de la fuente de datos, esta puede ser un servicio web, una interfaz de aplicación o API, una base de datos, o inclusive archivos de éxito plano.

Luego de leer la información esta se debe transformar según las reglas del negocio y las estructuras definidas dentro del repositorio de datos, para finalmente ser cargados a este. Con la inclusión del lago de datos, estos procesos se cambiaron por los llamados de extracción y carga, ya que al lago de datos es necesario cargar los datos tal y como estos se generan.

Para luego ser transformados según necesidades diferentes, con lo que comienza la ideología llamada “almacenar primero, analizar después”. Los procesos de carga se pueden clasificar de la siguiente manera:

- Procesos de carga según su frecuencia
- Procesos de carga según su flujo

1.8.1. Procesos de carga según su frecuencia

Se llaman así porque se diferencian en la frecuencia con la que son ejecutados, por ende, la cantidad de datos o eventos que procesan en su ejecución. Estos procesos pueden ser:

- Lote: se llaman proceso en lote, porque se ejecutan con una frecuencia moderada, es decir, una vez por día, una vez por semana, entre otros. En

este periodo de ejecución acumulan en lotes los datos, para que estos sean procesados en ese momento, usualmente sin importar el tiempo que tome ejecutar el lote completo, siempre y cuando no colisione con la siguiente ejecución.

- Mini Lote: se consideran un caso especial de los procesos en lote, dado que se procesan con mayor frecuencia, por ende, los lotes de datos que procesan son menores. Dependiendo de la cantidad de datos que procesa por lote, pueden llamarse también micro lotes, y la frecuencia con que se ejecutan usualmente va desde una hora hasta un máximo de 5 minutos.
  En este tipo de procesos el tiempo de ejecución si es importante, ya que al ser ejecutados con mayor frecuencia es más probable que colisione con la siguiente ejecución.

- En tiempo real: estos van de la mano de un grifo, y procesan datos o eventos uno a uno, conforme van llegando en el flujo de datos, al ser en tiempo real, pueden ser únicamente procesos que involucran extracción y carga, únicamente transformación, entre otros. En estos procesos el tiempo de ejecución es crítico ya que debe demorar al sumo, el mismo tiempo que toma generar nuevos eventos.

1.8.2. Procesos de carga según su flujo

Los procesos para implementar cargas de datos, también se pueden categorizar según su flujo, esto determina el orden y tipo de procesos a implementar, estos pueden ser:

- Extracción, transformación y carga: estos son los procesos más comunes, ya que en un solo flujo implementan los tres bloques, y se asume que son

exitosos si y sólo si, los tres bloques son exitosos. Estos procesos son los utilizados por defecto en la carga de datos al repositorio de datos.

- Extracción y carga: estos procesos son usualmente utilizados más en el lago de datos que el repositorio de datos, ya que los datos llegan en su estado natural sin ningún tipo de transformación. Este tipo de procesos es utilizado, cuando una misma fuente de datos debe ser sometida a diferentes transformaciones en diferentes flujos de datos, lo que permite utilizar un solo proceso de extracción, en lugar de replicarlo en múltiples ocasiones.

- Transformación y Carga: estos procesos son subsecuentes de los procesos de extracción y carga, y se ejecutan una vez la fuente de datos ya se encuentra disponible para ser consumida, por lo que el proceso de transformación es ejecutado, posterior a que este sea cargado en su destino final.

    - Estos procesos son utilizados cuando una misma fuente de datos se utiliza en diferentes Data Marts.

1.9. Tecnologías de Amazon

Amazon comenzó a proveer servicios en la nube en el año 2002, cuando lanzó su servicio web que permite a sitios web de terceros obtener en formato XML el catálogo de productos de Amazon. Esto les dio a los desarrolladores externos, la habilidad de poder realizar implementaciones propias que les permitieran interactuar de mejor manera con sus clientes y mantener una relación con Amazon.

No fue hasta el año 2004, cuando Amazon se percató del potencial que tienen gracias a la infraestructura que habían desarrollado, por lo que decidieron anunciar el desarrollo de la plataforma AWS, la cual fue lanzada oficialmente en marzo 2006. En un comienzo los servicios que se ofrecían eran:

- Servicio de Almacenamiento Simple o S3 por sus siglas en inglés
- Computación en la Nube o EC2
- Servicios de Colas SQS

En la actualidad Amazon a través de sus servicios de AWS provee más de 60 diferentes servicios en la nube, que van desde servidores, bases de datos, hasta servicios personalizados para desarrolladores, contando con disponibilidad en las zonas principales de Estados Unidos, y expandiéndose a los puntos principales de Europa.

1.9.1. Bases de datos relacionales

Conocido como *AWS RDS* por sus siglas en inglés, fue lanzado en octubre del año 2009, con soporte únicamente para MySQL, lo cual consistía en un servidor que incluía la versión más reciente de la base de datos instalada, y esta era completamente administrada por Amazon.

La idea de Amazon era poder cubrir de una manera más sencilla para los desarrolladores, una necesidad bastante recurrente, que era contar con un servidor de base de datos en la nube, por lo que Amazon optó por ofrecer el combo, con la promesa de ahorrar tiempo y necesidad de mantenimiento, ya que ofrecía el calendarizar y ejecutar copias de respaldo automáticamente.

En la actualidad sigue siendo uno de los servicios más populares ya que ofrece compatibilidad con las bases de datos más utilizadas en el mercado, como lo son Oracle y Microsoft SQL Server, para las cuales ofrecen un sistema de pago por hora, que incluye el costo del servicio y el costo de licenciamiento.

1.9.1.1. Amazon aurora

Para el año 2014, Amazon lanzó su propia versión de MySQL llamado Amazon Aurora, el cual consistía en un servicio RDS, pero con una versión optimizada de MySQL que prometía un mejor rendimiento y velocidad de transacciones, que su contraparte original, a un precio menor.

Esta nueva base datos, al ser una implementación propia de Amazon, les permite integrarse con otros servicios propios como S3, así como poder ofrecer almacenamiento Elástico y virtualmente ilimitado, teniendo una capacidad máxima de 64 Terabytes. En el año 2017 Amazon agregó compatibilidad con PostgreSQL, con características similares a su contraparte de MySQL, con transacciones mucho más rápidas y almacenamiento virtualmente ilimitado.

1.9.2. Servicio de almacenamiento simple

Conocido como S3, por sus iniciales en inglés que comienza con la letra S. Es uno de los principales servicios que Amazon ofrece, así como de los más populares, ya que provee almacenamiento ilimitado y según las necesidades que tenga cada organización.

No se tiene ningún tipo de límite en el tipo de archivos o el tamaño de los mismos para ser almacenados en este servicio, así mismo, dependiendo de la frecuencia con que los archivos son accesados, Amazon s3 proveed diferentes

niveles de acceso y costos para dichos niveles, siendo la capa de acceso más frecuente la más costosa y la menos frecuente menos costosa.

En el año 2012 Amazon lanzó Glacier, que consiste en un servicio basado en S3, pero diseñado para soportar copias de respaldo de gran tamaño de acceso esporádico, con lo cual permite reducir los costos a las organizaciones aún más, dando un precio de 10 a 1 versus el tradicional servicio de S3.

Si bien los usos para S3 son diversos, los más comunes consisten en almacenar copias de respaldo, registros del sistema, e incluso HTML estático para servir contenido web, esto último requiere configuración adicional, pero ha permitido a los desarrolladores poder construir sitios web estáticos sin contar con un servidor web o infraestructura costosa.

Con el crecimiento en popularidad del lago de datos, Amazon aprovechó el cambio, para promover S3 como base para construir un lago de datos, por lo que incluye compatibilidad con sistemas distribuidos como los de Hadoop, e incluso implementó su propia versión de Hive y Presto, llamado Amazon Athena.

Con el cual permite crear una base de datos a partir de archivos en el S3 con una estructura definida y jerarquía de carpetas definida. Esto permitió potenciar aún más el uso de Amazon S3 y consolidarse como uno de los recursos más importantes de Amazon en su plataforma AWS.

1.9.3. Redshift

Es una base de datos diseñada por Amazon, tomando como base la arquitectura de PostgreSQL en su versión 8.0.2, Amazon cambió la capa de almacenamiento, para convertirla en una base de datos columnar, contrastando

con el esquema tradicional de PostgreSQL de ser una base de datos relacional, con el objetivo de poder soportar conjuntos de datos masivos y ofrecer tiempos de respuesta mucho mejores que sus contrapartes relacionales.

Así mismo, implementa un esquema de grupo de nodos, en lugar del esquema monolítico de los servidores de base de datos relacionales, lo que le permite escalar de forma horizontal, lo que significa, que cuando el grupo de nodos ha alcanzado su capacidad máxima, se pueden agregar fácilmente más nodos y así incrementar las capacidades, por el contrario los servidores de bases de datos tradicionales que cuando alcanzan su capacidad máxima se debe cambiar a una servidor con el doble de características y doble de precio.

1.9.3.1. Características de Redshift

Redshift posee diversas características que lo hacen una herramienta muy popular y potente, que lo hace destacar por sobre otras alternativas de diferentes proveedores, estas características son:

- Procesamiento paralelo: al estar basado en nodos, redshift toma todas las ventajas de esto al dividir el procesamiento que cada consulta entre la cantidad de nodos disponibles, lo que permite obtener mejores tiempos de ejecución.

- Compresión: al ser una base de datos columnar, le permite a redshift decidir el almacenamiento apropiado para cada columna, en lugar de por cada tabla, esto permite realizar operaciones como compresión, sobre las columnas, lo que permite no solo reducir el tamaño de los bloques de almacenamiento para la columna, también permite mejorar los tiempos de lectura, al reducir la cantidad de bloques que deben ser leídos por consulta.

Gracias a estas características se dice que Redshift tiene la capacidad de realizar operaciones sobre miles de millones de registros al mismo tiempo, por lo que se ha convertido en un servicio esencial para la construcción de arquitecturas de datos masivos.

1.9.4. Kinesis

Es una solución que Amazon provee para el manejo de flujos de datos constantes y en tiempo real, la cual permite centralizar la colección de estos datos, para luego ser procesados según la necesidad de organización.

Kinesis ofrece múltiples operaciones sobre los datos colectados, como pueden ser, almacenados como archivos en S3, almacenarse en cualquier base de datos que ofrece Amazon, o bien realizar cualquier tipo de operación personalizada, con la capacidad de soportar cantidades de datos de cualquier escala, así como ofrecer características de alta disponibilidad, que permiten garantizar que los datos recibidos serán procesados de una u otra forma, lo que lo hace resiliente a posibles imprevistos.

Si bien no es tan flexible como otras soluciones del mercado, como por ejemplo Kafka o RabbitMQ, las cuales son servicios de colas para propósitos específicos, Kinesis puede ser una solución aceptable para problemas comunes de flujos de datos constantes, así mismo, de poseer la ventaja de estar totalmente administrada por Amazon, por lo que reduce el tiempo de instalación y administración requerido.

1.9.5. Lambdas

Es una plataforma para implementar servicios basados en eventos, en un ambiente llamado Serverless o sin servidor. Se dice que está basado en eventos ya que las Lambdas son creadas y configuradas para responder a diferentes tipos de eventos, como lo son, solicitudes web, eventos calendarizados, cambios en diferentes entornos, entre otros.

Se dice que es Serverless, ya que consiste en subir a la nube una porción de código que procese basado en una entrada de datos o evento y produzca una respuesta y Amazon procede a instalarla en un servidor propio, con lo cual únicamente cobra basado en las ejecuciones y tiempo que la Lambda esté en funcionamiento, lo que resulta mucho más económico y fácil de implementar que un servidor tradicional.

Si bien este servicio ha venido a ser revolucionario y muy útil, posee dos desventajas grandes que limitan su usabilidad, las cuales son:

- Tiempo máximo de ejecución: las Lambdas al ser pequeñas funciones, Amazon tiene la limitante de proveer un tiempo máximo de 5 minutos de ejecución, por lo que, si el proceso implementado excede este tiempo, el servicio será detenido al cumplirse el tiempo sin importar si el procesamiento concluyó o no, lo que es una limitante a tener en cuenta para procesos de datos.

- Tiempo de calentamiento: si bien las Lambdas se pueden configurar para responder a eventos web, lo que permite que sean implementadas como servicios web, poseen la desventaja de necesitar un tiempo de calentamiento, que es el tiempo que necesita Amazon para alojar los

recursos de infraestructura para ejecutar la Lambda, lo que puede resultar en tiempos de respuesta largos, al menos para las primeras solicitudes. A partir del año 2019, Amazon anunció que esta limitante se puede configurar, para garantizar que las Lambdas estarán siempre listas, eliminando esta restricción a cambio de un coste mayor.

A pesar de las restricciones que presentan, que podrían ser importantes para servicios web, las Lambdas se han convertido en una herramienta importante en el procesamiento de datos, ya que proveen la habilidad de responder a eventos como, por ejemplo, entrada de datos en Kinesis, entrada de datos en S3, entre otros.

Esto ha permitido el desarrollo de arquitecturas de datos que procesan datos en tiempo real, cuyo foco principal son estos recursos, permitiendo a las organizaciones automatizar procesos de ingesta, transformación y prestación de datos.

1.9.6. DynamoDb

Es un servicio de Base de datos NoSQL de la familia de las Clave Valor, administrado completamente por Amazon. La principal característica de este servicio es que permite crear tablas, con una llave principal y soporte para cualquier tipo de objetos como valor, donde los objetos pueden ser columnas adicionales o bien objetos en formato JSON.

Donde Amazon implementa un esquema de cobros basado en la cantidad de lecturas y escrituras que dicha tabla recibirá por segundo, donde el almacenamiento de la Data originalmente no era sujeto de cobro, actualmente

ofrecen 25 Gigabytes gratuitos de almacenamiento y $0,25 al mes por cada Gigabyte adicional de almacenamiento.

Esta característica de poder configurar la cantidad de lectura y escritura permite modificar a voluntad la velocidad de respuesta que tendrá la tabla, según la demanda que esta tendrá, incluso permitiendo crear reglas para incrementar o decrementar estos valores según horarios de mayor demanda, así mismo, al configurar de manera separada la lectura y escritura, se puede optimizar basado en el uso de la tabla, en donde se puede implementar tablas con poca escritura pero acceso frecuente y viceversa.

Al ser una base de datos clave valor, ofrece tiempos de respuesta muy cortos a solicitudes de claves específicas, no obstante, no se recomienda su uso para consultas abiertas, es decir, consultas donde no se conoce con certeza las claves, ya que este tipo de consultas demora tiempos muy altos.

Otra de las ventajas que posee DynamoDb, es que permite configurar claves principales y particiones, con lo cual facilita el acceso a los datos cuando estos se masifican y de esta manera optimizar la lectura y escritura, siempre y cuando las particiones sean adecuadamente balanceadas.

Otra característica es que provee disparadores que responden a ciertas condiciones y permiten ejecutar Lambdas, se realizarán acciones personalizadas a eventos de escritura, modificación o eliminación de registros dentro de la base de datos.

1.9.7. Redis

Consiste en un servicio de datos en memoria, utilizado tradicionalmente como caché para todo tipo de aplicaciones, aunque ha evolucionado a ser una NoSQL de clave valor, que puede funcionar de manera distribuida para garantizar alta disponibilidad, con la habilidad de persistir los datos a disco.

También provee servicios de colas y disparadores que permiten ejecutar funciones personalizadas basado en condiciones que suceden con la data, provee servicios de contadores que funcionan en ambientes distribuidos y accedidos por múltiples servicios.

Todo esto con tiempos de lectura y escritura muy cortos, la que la convierten en una alternativa útil, por encima de otros servicios y recursos en la web. No hace mucho fue incluida en el paquete de Elasticache que Amazon provee, donde se brinda el servicio de Redis, completamente distribuido y administrado por Amazon.

1.9.8. Glue

Introducido en el año 2018, es un servicio basado en eventos y Serverless, conocido como el hermano de Amazon Lambdas, pero con enfoque a los procesos de Extracción, Transformación y Carga o ETL por sus siglas en inglés. Al ser su enfoque principal para procesos de ETL, la restricción de 5 minutos no está presente, por el contrario, Amazon presenta un modelo de cobro donde el mínimo es de un minuto de tiempo de ejecución.

Así mismo, permite definir la cantidad de procesadores y memoria *RAM* que el proceso necesita, por lo que esto también se toma en cuenta en el proceso de

cobro de la herramienta. Su enfoque principal consiste en ser una solución alternativa para los servicios de EMR *Enterprise Map Reduce*.

EMR consiste en un conjunto de tecnologías como Apache Spark, Apache Hive, Presto, entre otras, que permiten leer, procesar y sumarizar grandes cantidades de datos que residen en un servicio de S3, para poder presentarlos o ser utilizados en otros procesos, se puede seleccionar la cantidad de procesadores y memoria, para cada proceso ejecutado en este servicio.

Si bien este servicio se cobra bajo demanda, resulta ser una solución costosa para procesar datos, es por ello que Amazon introdujo Glue, como una alternativa más económica, que permite realizar casi las mismas operaciones, pero donde se paga únicamente por el tiempo que se ejecuta el proceso.

Por lo que se ha convertido en un recurso vital para procesar grandes cantidades de datos, y gracias a su habilidad de responder a diversos eventos, permite ejecutar procesos, cuando archivos cambian dentro del S3 o bien calendarizados para ejecuciones constantes.

2. PROPUESTA DE ARQUITECTURA

2.1. Análisis de arquitecturas tradicionales

Las arquitecturas tradicionales de datos consisten en implementaciones de repositorios de datos, basado en la propuesta original de Charles Tupper, en la que consistía de crear un lugar en donde se centralizan todos los datos que colecta una organizaciones, este proceso como se mencionó, define un modelo dimensional que representa los datos a un nivel de abstracción alto, y se diseñado específicamente para responder a las preguntas más comunes de la organización y asistir en el proceso de toma de decisiones.

Los modelos de datos más usuales consisten en modelo estrella o modelo copo de nieve, en donde ambos modelos consisten en representar los datos como hechos y los campos que los describen como dimensiones, en una estructura desnormalizada para garantizar que no existan más de dos niveles de relación, lo que permite simplificar las consultas en volúmenes grandes de datos.

Para poder llevar los datos a esta estructura es necesario construir procesos de extracción, transformación y carga que apliquen las operaciones necesarias sobre los datos crudos, para que estos se adapten y encajen en la estructura diseñada. Las fuentes de datos, suelen ser cualquier tipo de datos que son compartidos con la organización como, archivos de texto, bases de datos, servicios, entre otros.

Este proceso es de los más antiguos en el desarrollo de sistemas y arquitecturas de datos, por lo que se considera uno de los más robustos y

documentados hasta el momento, así mismo por el hecho de ser una arquitectura amigable con cualquier sistema de inteligencia de negocios y análisis de datos, es usual que las organizaciones prioricen la construcción de un sistema de repositorio para asistir el proceso de toma de decisiones.

2.1.1. Ventajas

Entre las ventajas cabe resaltar de este tipo de arquitecturas están:

- Existe una metodología definida que puede ser fácilmente adaptada a cualquier necesidad, por lo que implementarla es más sencilla que cualquier otro tipo de arquitectura.

- Son fáciles de integrar con cualquier sistema de inteligencia de negocios, como lo son *Tableau, Power BI, Looker, Quick Sight*, entre otros.

- Si bien su construcción inicial lleva un tiempo considerable, su costo de mantenimiento es relativamente bajo, ya que, una vez implementada la estructura, esto define los lineamientos que todos los procesos de extracción, transformación y carga deben seguir.

- Se pueden implementar en casi cualquier motor de base de datos, relacional o columnar, por lo que son fáciles de entender una vez implementadas, por cualquier tipo de usuario con experiencia manejando sistemas relacionales.

2.1.2. Desventajas

Entre las desventajas de esta arquitectura se pueden mencionar:

- Si bien existen lineamientos estándar, su construcción requiere un alto grado de abstracción y entendimiento de los procesos de la organización, por lo que su diseño, desarrollo e implementación, usualmente toma tiempo.

- Cuando la complejidad de las operaciones de una organización es grande, no es posible resolver las necesidades con un solo modelo de datos, por lo que se deben dividir cada hecho de interés en un Data Mart, por separado. Esto implica que cada hecho debe ser analizado y diseñado por su propia cuenta.

- Al proveer un modelo estándar para los datos, muchas veces adaptar las fuentes de datos al modelo, resulta en procesos complejos, ya que muchas veces las fuentes de datos no son lo suficientemente homogéneas.

- Una vez el modelo se encuentra implementado, los cambios son complejos de implementar, ya que estos pueden romper la homogeneidad del modelo de datos, o bien requerir la construcción de un Data Mart por separado, por lo que se considera que la implementación de cambios puede tomar mucho tiempo.

- Cuando la organización crece demasiado en cuanto a su volumen de datos, puede implicar en tiempos de respuesta altos, ya que, a pesar de implementar un modelo desnormalizado, generalmente se implementan sobre motores de bases de datos relacionales, las cuales sufren un desgaste considerable cuando el volumen de datos crece exponencialmente.

- Se considera que estas arquitecturas son ideales para proyectos de

inteligencia de negocios, pero usualmente no son viables de utilizar para modelos de ciencia de datos, ya que estos procesos usualmente requieren datos crudos y no procesados.

2.1.3. Bases de datos para utilizar

Esta arquitectura puede ser implementada en cualquier motor de base de datos relacional o columnar, por lo que resulta en el modelo más común y fácil de implementar. Para esto Amazon ofrece las siguientes herramientas.

2.1.3.1. RDS y Amazon Aurora

Amazon provee una solución de motores de bases de datos relacionales autoadministrados, lo que facilita mucho su administración e implementación, así mismo al ofrecer los motores de bases de datos más populares como MySQL, PostgreSQL, MariaDb, SQL Server y Oracle, donde para los últimos dos se necesita cancelar una cuota adicional por concepto de licenciamiento.

Adicionalmente Amazon ofrece dos tipos de instancias, una de uso general y otro con optimización de memoria RAM, para estos dos tipos, Amazon ofrece seis diferentes subtipos, lo cuales tienen componentes especiales como GPUs y/o CPUs dedicados o bien de diferentes versiones, finalmente ofrece seis tamaños diferentes que van desde el tamaño micro hasta el 16 xLarge. Esto puede ser complicado en un principio, ya que escoger la instancia con las características adecuadas no es simple.

Para facilitar esto se debe considerar lo siguiente, las instancias optimizadas para uso de memoria RAM, están diseñadas para soportar consultas complejas y grandes que son ejecutadas con mucha frecuencia, por lo que, si bien en un

repositorio de datos este puede ser el escenario más común, estas instancias pueden ser mejor opción.

No obstante, si sobre el modelo de datos no se espera ejecutar consultas complejas, una instancia de uso general podría ser lo más apropiado, además de que son un poco más económicas. Para escoger el tamaño adecuado, esto se basaba enteramente en la cantidad de datos que se tienen y que se espera tener en un lapso de entre tres y cinco años.

Este cálculo se puede hacer basado en la tabla de hechos y el tamaño de cada registro, el cual se calcula sumando el tamaño de cada columna, multiplicando por la cantidad de registros actual y la cantidad proyectada.

Porque el tamaño de los datos es importante, por dos razones, la primera es que según la cantidad de datos que se tienen en la base de datos, esta o similar será la cantidad de datos que se necesitan subir y descargar y Amazon provee diferentes cuotas según el tamaño de la instancia, en donde sí se excede la cuota de la base de datos, esta cuota adicional es cobrada además del costo de la instancia.

Otra característica es que según el tamaño de los datos es la cantidad de espacio en memoria RAM que la base de datos necesitará reservar en consultas complejas, entre menos memoria RAM se tenga disponible para ser utilizada, esto impacta negativamente en el uso del CPU ya que este deberá trabajar más para poder responder a la consulta.

Así mismo se debe considerar la cantidad de conexiones que se van a utilizar, ya que el límite de conexiones permitidas para los motores de base de datos está dado directamente por la cantidad de memoria RAM, por lo que, si las

bases de datos estarán sometida a grandes cantidades de operaciones y consultas, puede saturar fácilmente la instancia y resultar en bajas en el servicio.

2.1.3.2. Redshift

Si bien Redshift es una base de datos columnar, es una opción ideal para la implementación de los diferentes modelos de repositorio de datos, ya que Redshift maneja de mucho mejor manera estructuras desnormalizadas, no obstante, su costo inicialmente puede ser mayor al de un RDS o Aurora tradicional, por lo que se recomienda utilizar Redshift cuando el volumen de datos es lo suficientemente grande y los tiempos de respuesta en una estructura relacional ya no es el adecuado.

Dado que las instancias de Redshift funcionan en clúster, lo que permite agregar o quitar nodos cada vez que se necesita, lo que permite realizar escalamiento horizontal, el cual es más rápido y económico que el escalamiento vertical que se debe realizar en instancias de RDS y Aurora.

2.1.3.3. Comparativa

En la siguiente tabla se describe la comparativa de bases de datos para arquitectura tradicional, detallando los tipos y costos.

Tabla II. **Comparativa de bases de datos para arquitectura tradicional**

Característica	RDS	Aurora	Redshift
Tipo de Motor	Relacional	Relacional	Columnar
Tipo de Instancia	Servidor Monolítico	Servidor Monolítico	Clúster
Optimizado para	Operaciones Transaccionales	Operaciones Transaccionales	Procesos analíticos
Almacenamiento	Dependiendo el Tamaño	Ilimitado	Dependiendo del tamaño (existen instancias de almacenamiento ilimitado)
Costo de Instancia más pequeña	$12,25/mes para MySQL, PostgreSQL y MariaDb. $27,36 para Oracle y $31,68 para SQL Server	$29,52/mes para MySQL y $59,04 para PostgreSQL	$360/mes

Fuente: elaboración propia.

Si bien utilizar un RDS o una instancia de aurora puede ser una opción viable para organizaciones pequeñas, o bien, pruebas de concepto sobre modelos de repositorio de datos o Data Mart, a largo plazo Redshift es una mejor opción ya que permite una mejor escalabilidad y tiempos de respuesta para las consultas, dado que está optimizado para proyectos de inteligencia de negocios.

2.1.4. Herramientas

La opción preferida para construir procesos de extracción, transformación y carga suele ser crearlos desde 0 utilizando un lenguaje de programación y el sistema operativo para calendarizados, Amazon provee una variedad de herramientas que se debe tomar en cuenta para facilitar este proceso.

2.1.4.1. Funciones Lambda

Las funciones Lambda son un servicio de Amazon que permite ejecutar porciones de código sin necesidad de administrar servidores, con lo cual simplemente basta con cargar el código que se desee ejecutar y configurar el tipo de recursos necesarios para que este se ejecute y Amazon procederá a asignar los recursos necesarios para poder ejecutar dicho código.

Si esto provee una gran ventaja para los desarrolladores, ya que es una forma económica de ejecutar pequeñas porciones de código que realizan una tarea pequeña y específica, la cual puede configurarse para ser ejecutada con diversos disparadores, como, por ejemplo, solicitudes web, eventos calendarizados o bien bajo demanda.

La desventaja que poseen las funciones Lambdas es que utilizan recursos limitados, por lo que no se pueden ejecutar funciones que consuman muchos recursos, y tienen un límite de tiempo de ejecución de 5 minutos, por lo que no son una solución apropiada para la ejecución de procesos en bloque que requieran más tiempo y recursos.

Para poder brindar una solución más robusta y aprovechar la ventaja de los servicios sin servidor que ofrecen las Lambdas, Amazon implementa lo que se le

conoce como funciones paso a paso, que permiten ejecutar y coordinar varias funciones Lambdas en un solo proceso, lo que permite ampliar los casos de uso a los que se puede aplicar.

Esto permite que se pueda utilizar para construir para implementar procesos de extracción, transformación y carga, con el único requisito que dichos procesos deben ser divididos en pequeñas porciones de código que requieran pocos recursos y tomen menos de 5 minutos de ejecución.

2.1.4.2. Data Pipelines

Consiste en una implementación específica de las funciones paso a paso, orientado para facilitar la creación de flujos de datos que utilizan los recursos más comunes de Data en Amazon, como lo son servicios de S3, RDS, DynamoDb, entre otros, aprovechando la filosofía de las Lambdas y funciones paso a paso.

2.1.4.3. Glue

Glue es la inclusión más reciente en el catálogo que ofrece Amazon para construir procesos de extracción, transformación y carga, enfocado específicamente para procesos de carga de grandes cantidades de datos, ya que permite alojar recursos dinámicos y de procesamiento paralelo para poder procesar rápidamente bloques de datos.

Esta herramienta no es tan flexible como las funciones, paso a paso o los Data Pipelines, dado que requiere que el usuario implemente los flujos de Data utilizando el framework Spark, ya sea mediante Python o bien mediante Java. Lo que lo restringe a casos de uso particulares.

2.1.4.4. Airflow

Es una plataforma para la construcción de flujos de datos, desarrollado por la empresa Airbnb, que permite calendarizar y ejecutar cualquier tipo de flujo de datos. Se basa en el principio de grafos acíclicos dirigidos, en el que cada flujo tiene un nodo raíz y un nodo final, y no admite ciclos, lo que significa que no puede existir una transición a un nodo previo.

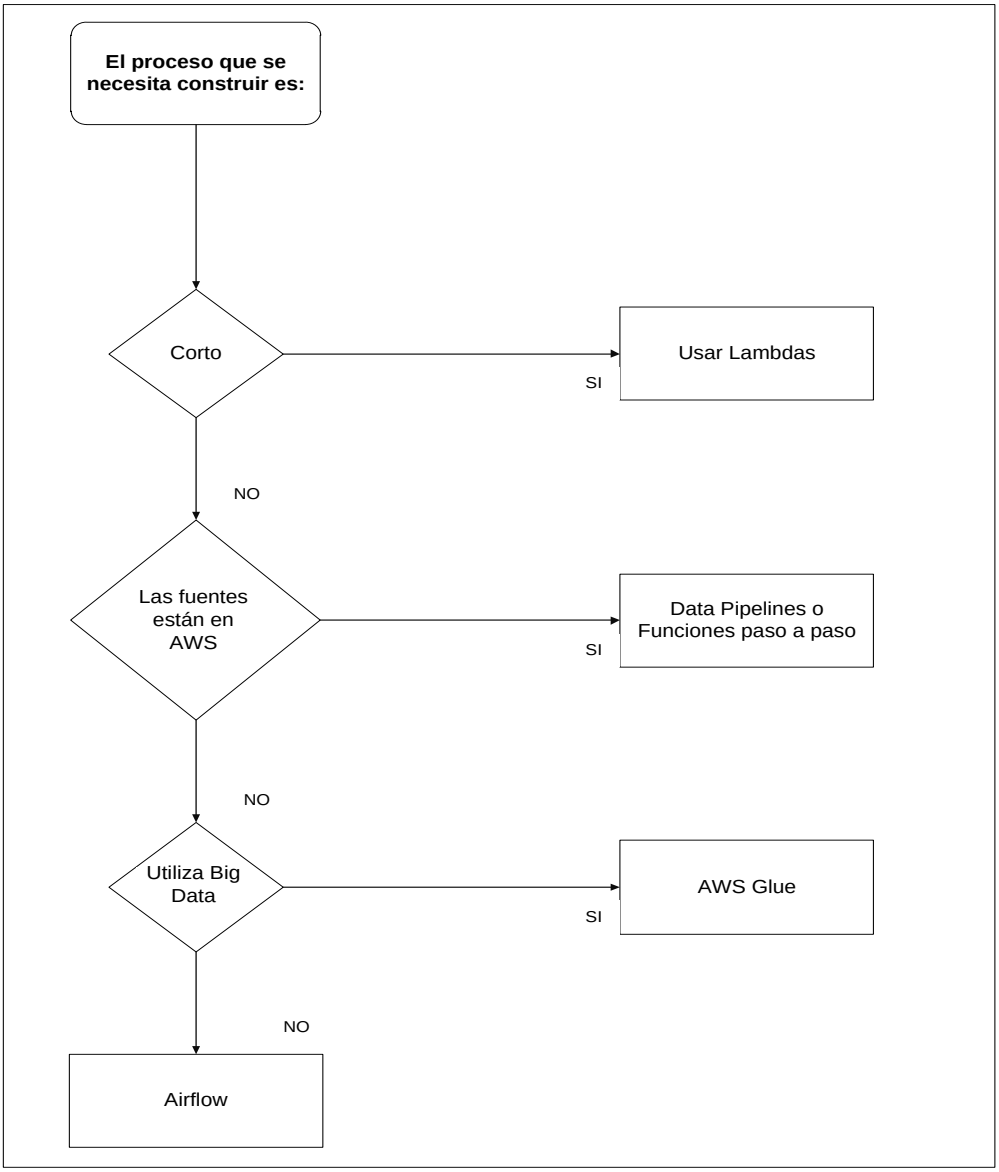
Esta plataforma cuenta con una popularidad por su facilidad de uso y que puede ser adaptada a casi cualquier requerimiento, aunque su concepción inicial fue para construir procesos de extracción, transformación y carga utilizando Python.

Esto motivó a Amazon a incluir Airflow como un servicio autoadministrado en su plataforma desde el año 2019, con lo cual los usuarios pueden utilizar dicha herramienta sin la necesidad de administrar los servidores o bien la configuración que requiere para poder instalarlo, así mismo ofrece un flujo de implementación sencillo que permite automatizar los procesos de despliegue de los flujos implementados.

2.1.4.5. Comparación

Todas las herramientas que sean prestando pueden ser utilizadas para cumplir el mismo propósito de extraer, transformar y cargar datos a un repositorio de datos, por lo que compararlas no resulta adecuado, ya que dependiendo del caso de uso puede ser que una herramienta sea más apropiada que otra, es por ello que se presenta un diagrama de casos de uso y que herramienta podría ser la más apropiada.

Figura 9. **Flujo de decisión para arquitecturas tradicionales**



Fuente: elaboración propia.

2.2. Análisis de arquitectura Lambda

Con la llegada del concepto de Big Data y el constante crecimiento y generación de data, el lograr construir una arquitectura eficiente que pueda resolver la necesidad de inteligencia de negocios, así como la creciente necesidad de análisis y construcción de modelos predictivos. Esto plantea dos retos fundamentales a resolver en esta nueva era:

- La latencia: este es el problema principal que deriva la mayoría de las soluciones, cómo se puede surtir los datos necesarios, de la manera más rápida posible, ya que la mayoría de las organizaciones no pueden esperar un día completo o bien una hora completa para poder analizar la nueva información generada.

- La precisión: se dice que no tener los datos es mejor que tener datos imprecisos, ya que la ausencia de datos es un problema fácil de detectar e intentar resolver, no obstante, los datos imprecisos pueden llevar a tomar decisiones no acertadas o bien a realizar predicciones sesgadas a valores erróneos.

Estos problemas podrían parecer comunes, pero resultan sumamente complejos de resolver para cualquier organización; siempre existirá el problema de poder obtener, transformar y servir datos precisos en el menor tiempo posible, sin importar su volumen y fuentes.

En la actualidad, las fuentes de datos son tan diversas que van desde archivos de texto con reportes hasta datos de ubicación y transacciones generadas por dispositivos de internet de las cosas. Es aquí donde Nathan Marz

en el año de 2011 acuña el término de Arquitectura Lambda[9], para indicar el concepto de una infraestructura inmutable que obtener datos, procesarlos y servirlos.

El concepto detrás de la arquitectura Lambda se basa en que esta no dependa de una tecnología en específico, más bien en la definición de bloques con diferentes tecnologías para cumplir un propósito específico, este concepto no es nada nuevo, ya que es el principio base en el que se construyen la mayoría de las soluciones basadas en Big Data.

El patrón inicial que define la arquitectura Lambda es el de que la Data historia y la Data generada en tiempo real, no se deben manejar de manera aislada, por el contrario, estas deben complementarse la una a la otra, lo cual contrasta desde un principio con los principios de la arquitectura tradicional, la cual especifica lo opuesto. Para lograr esto la arquitectura Lambda propone los siguientes puntos:

- Patrones y guías para que la historia y en tiempo real pueda ser consultada junta, lo que permite combinar diferentes tipos de vistas y reportes.

- Debe ser agnóstica de la tecnología, por lo que obliga a definir bloques intercambiables, generalizables y que mantengan las responsabilidades de cada capa.

- Define tres capas principales de datos con responsabilidades claras, aplicando el principio de separación de conceptos.

[9] TOMCY John, PANKAJ Misra. *Data Lake for Enterprises*. Estados Unidos: Packt Publishing, 2017. https://learning.oreilly.com/library/view/data-lake-for/9781787281349/301fc654-0f78-4055-8bb2-de036b938bee.xhtml. Consulta: abril de 2021

- Se considera independiente del dominio, lo que significa que puede ser implementada en cualquier tipo de organización.

- Tolerante a fallas, la arquitectura debe construirse teniendo en cuenta como cada bloque debe recuperarse ante cualquier tipo de falla.

- La Data es inmutable, este es un principio base en Big Data, donde los datos se deben agregar y complementar, más nunca actualizar o borrar.

- Reprocesable, la Data en este tipo de arquitectura al ser inmutable implica que esta puede ser reprocesada en cualquier momento y el mismo resultado se debe obtener siempre.

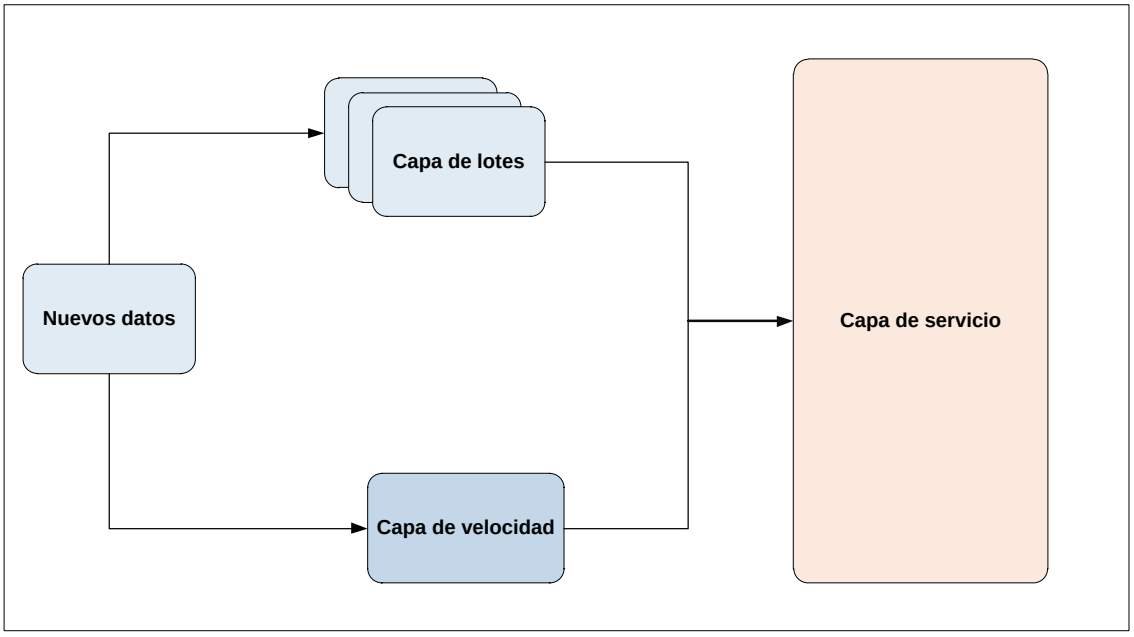
2.2.1. Componentes de una arquitectura Lambda

La arquitectura Lambda se basa en la implementación de tres capas principales con responsabilidades específicas y agnósticas de tecnología, por lo que es vital entender la responsabilidad de cada capa, para poder incorporar las herramientas adecuadas para construirla.

Independientemente de las capas debe existir un lago de datos, ya que este es el requisito principal para cada capa, por lo que es vital que antes de construir una arquitectura Lambda, se cuente con una implementación de lago de datos en la organización. Con esto en mente las tres capas que define la arquitectura son:

- Capa de lotes
- Capa de velocidad
- Capa de servicio

Figura 10. **Bloques conceptuales arquitectura Lambda**



Fuente: elaboración propia.

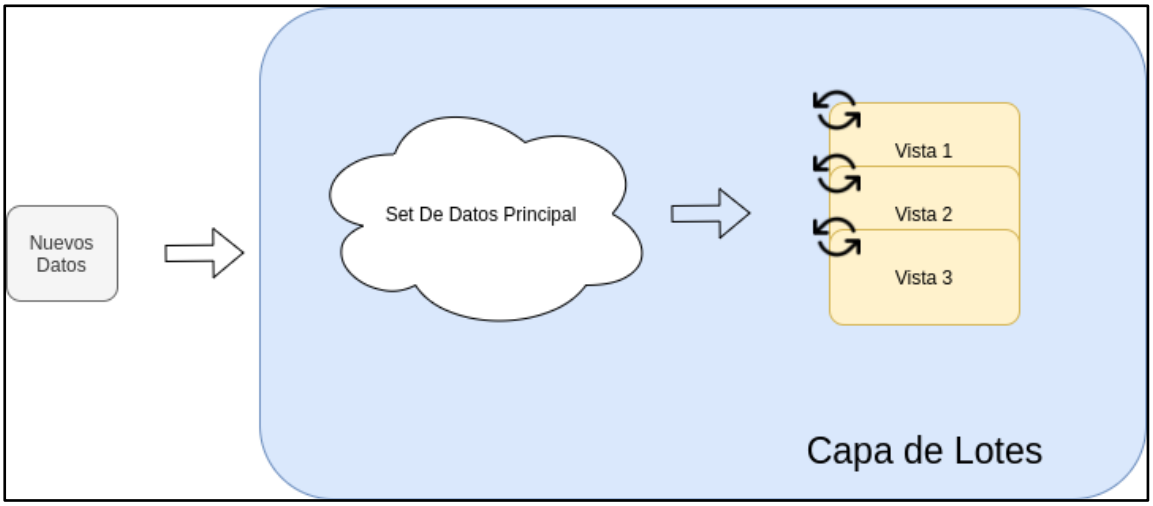
2.2.1.1. Capa de lote

Esta es la capa en la que toda la Data en su forma más cruda es almacenada, en referencia a Data cruda estos son los datos en su forma pura sin ninguna transformación ni cambio, esto permite construir vistas con diferentes puntos de vista, que utilicen diferentes bloques de datos de una misma fuente.

En este bloque los datos deben almacenarse de manera inmutable, por lo que no se permiten operaciones de actualización o borrado, y la Data puede ser únicamente agregada, aquí se recomienda siempre agregar una clave de fecha que permita distinguir la última versión de los datos.

Si bien realizar consultas a los datos en esta capa puede resultar lentos y requerir mucho procesamiento, se recomienda de manera periódica crear y almacenar vistas sobre esta capa que realicen las transformaciones iniciales de cómo la Data será consumida, a estas se les conoce como vistas de lotes. Estas vistas pueden ser regeneradas cada vez que sea necesario y esta sobrescribirá la vista anterior o bien se puede descartar la vista anterior.

Figura 11. **Capa de Lote**



Fuente: elaboración propia.

2.2.1.2. Capa de velocidad

Esta capa también se le conoce como capa de datos en tiempo real, y su objetivo es permitir agregar los datos de la capa de lotes con los datos generados en tiempo real, esto agrega una complejidad, ya que la ventana de procesamiento en lotes se debe reducir lo más que se pueda, pero esto podría tomar más tiempo del que la organización necesita, ya que usualmente debe procesar una cantidad considerable de datos.

Para lograr esto la capa de velocidad debe tomar los datos en tiempo real y agregarlos, para que puedan ser consumidos en lo que se conoce como vistas en tiempo real, las cuales son construidas para que tengan un tiempo de respuesta y reprocesamiento bajo, una vez la capa de lotes es capaz de regenerar las vistas, las vistas en tiempo real se pueden descartar, y proceder a construir nuevas.

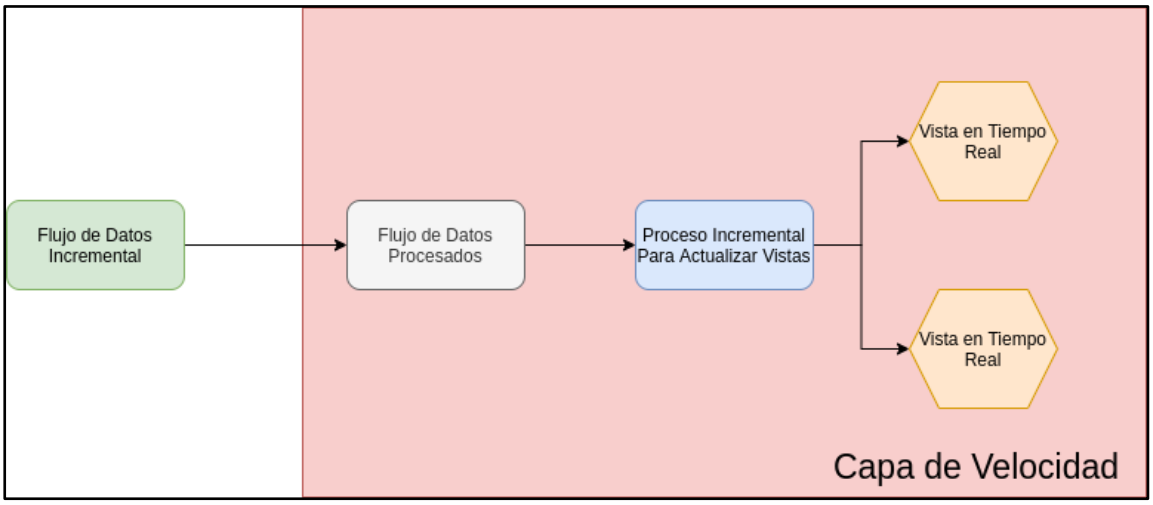
Esto significa que los usuarios deben de ser capaces de consumir las vistas de la capa de lotes y de la capa de velocidad, combinando los resultados de ambas para proveer los datos necesarios. Para lograr estos objetivos la capa de velocidad debe ser capaz de implementar estos conceptos:

- Cálculos incrementales, esto implica que se debe poder crear una nueva vista con los datos en tiempo real, procesando cada pequeño bloque de datos nuevo que se agrega, de manera que esta se pueda actualizar en un tiempo corto de procesamiento.

- Consistencia eventual, este principio se basa en que existen muchos cálculos que es imposible ejecutarlos en tiempo real, por lo que se pueden generar aproximaciones, las cuales se considera que eventualmente serán las correctas.

Uno de los retos más grandes al construir una capa de velocidad, es que, por el contrario de la capa de lotes, esta si admite mutabilidad, lo que implica que la Data puede ser actualizada, pero al lidiar únicamente con un set de datos pequeño, comparado con la capa de lotes, con lo cual el impacto en los tiempos de ejecución no debería ser significativo.

Figura 12. **Capa de velocidad**

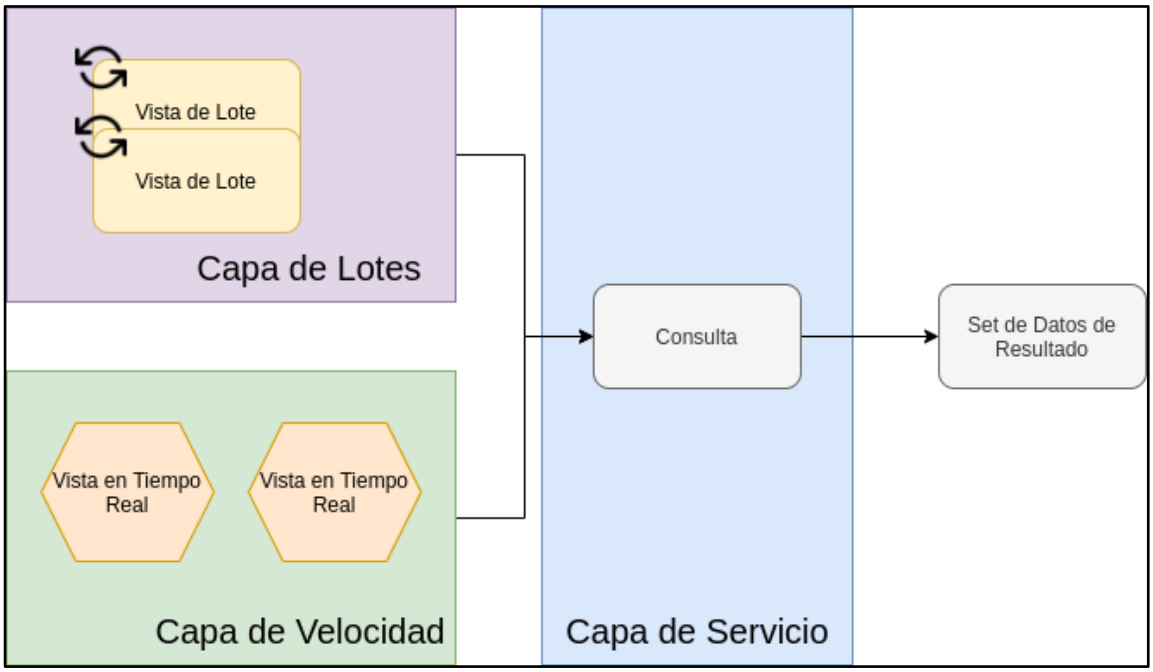


Fuente: elaboración propia.

2.2.1.3. Capa de servicio

Esta capa existe con el propósito para exponer los datos a los usuarios finales o sistemas de reportería. Esto requiere varios procesos de orquestación que permitan combinar los datos de las dos capas anteriores, con lo cual debe determinar cuándo se debe utilizar las vistas de la capa de velocidad y cuando las de la capa de lotes.

Figura 13. **Capa de servicio**



Fuente: elaboración propia.

2.2.2. Ventajas

Entre las ventajas principales que se pueden resaltar de la arquitectura Lambda son:

- Tolerancia a fallos y robustez, ya que la arquitectura Lambda debe ser construida para ser tolerante a fallos, así mismo al buscar enforzar la inmutabilidad de los datos (exceptuando en la capa de velocidad donde esto no es esforzado completamente), lo que permite reducir los fallos en los datos.

- Escalable, una de las ventajas principales de ser construida en bloques y

agnóstica a la tecnología, es que permite escalar fácilmente, al construir los procesos de manera distribuida, lo que permite fácilmente incrementar la cantidad de procesos que colectan y manipulan los datos.

- Generalización, como se observa la arquitectura es definida con principios de generalización que permiten que esta sea adaptada a cualquier necesidad y organización.

- Extensible, ya que cada vez que una nueva fuente de datos debe ser agregada al proceso, bastará con construir o editar las vistas actuales en cada capa, sin afectar los procesos actuales.

- Queries ad hoc, esto implica que de ser necesario la capa de servicio puede ser generada a partir de consultas y no necesariamente depender de vistas previamente creadas, lo cual puede afectar los tiempos de respuesta, pero agrega flexibilidad al proceso.

- Depurable, se considera que, debido a la separación de bloques y capas en la arquitectura, facilita la depuración de los procesos para encontrar fallas.

2.2.3. Desventajas

Entre las desventajas para la arquitectura Lambda existen las siguientes:

- El costo, si bien esta arquitectura resulta útil para un requerimiento de volumen grande datos, puede resultar en una solución relativamente costos, ya que la capacidad de crecimiento consiste en agregar más recursos a la infraestructura, lo que puede incrementar los costos si no se

controla, así mismo, el hecho de que los datos no deben ser borrados (al menos en la capa de lotes), lo cual puede llegar a incrementar los costos de almacenamiento.

- Complejidad, se considera que puede llegar a ser complejo manejar dos tipos de conjuntos de datos, el de la capa de lotes y la capa de velocidad, ya que ambos operan con los mismos datos y deben realizar las mismas operaciones, de lo contrario podrían presentar valores inconsistentes o bien no compatibles.

2.2.4. Capa de almacenamiento

La arquitectura Lambda fue concebida para utilizar un Data Lake como capa de almacenamiento principal, no obstante, esto no implica que no se pudiese utilizar un motor de base de datos, no obstante, si se decide utilizar un motor de base de datos por la facilidad de realizar consultas, no se recomienda utilizar el mismo motor de base de datos para cada capa como se analizará a continuación.

2.2.4.1. Amazon S3

Es un servicio de almacenamiento ofrecido por Amazon, el cual permite almacenar cualquier tipo de archivos, por lo que se ha convertido en la base para la mayoría de las implementaciones de lagos de datos. Entre las ventajas al utilizar Amazon S3 es el precio del almacenamiento ya que este incluye 5Gb gratuitos, luego el precio es de $0,023 por GB, y este se reduce a 0,022 una vez calculados los 50 Tb de almacenamiento.

Entre las desventajas que tiene es que, al ser únicamente una capa de almacenamiento, esto implica que el procesamiento de datos deba ser

implementado con una herramienta distinta, afortunadamente Amazon provee herramientas que pueden permitir generar las vistas necesarias, como lo son Amazon Athena, Amazon Glue o bien Amazon EMR, los cuales se analizarán a detalle más adelante.

2.2.4.2. Motor de base de datos

Como se indicó anteriormente, Amazon S3 es la herramienta por defecto para implementar una arquitectura Lambda, no obstante, si se requiere facilitar la creación de vistas en cada capa, se puede implementar una arquitectura híbrida que utiliza un motor de base de datos particular para cada capa, estos pueden ser:

- Capa de Batch
  - Redshift
- Capa de Velocidad
  - Aurora PostgreSQL
  - Elasticsearch
- Capa de Servicio
  - Redshift Federated Queries
  - Amazon Athena

2.2.4.3. Redshift

Dadas sus capacidades para manejar grandes bloques de datos, convierten a *redshift* en una herramienta para tomar en cuenta en cuanto a la implementación de la capa de lotes.

Entre las ventajas que puede dar utilizar Amazon *Redshift*, para esta capa es la capacidad de generar fácilmente vistas sobre este bloque de datos, a esto se le suma la capacidad del motor de *redshift* de soportar vistas materializadas.

Esto consiste en vistas normales, es decir, que se crean utilizando una consulta, con la diferencia que los datos son pre calculados y almacenados en una tabla, lo que garantiza un tiempo de respuesta mucho mejor al momento de ser consultados, a diferencia de las vistas tradicionales que consisten en consultas precompiladas.

Finalmente, el hecho de que la capa de Batch considera la inmutabilidad en los datos, lo que favorece al motor de redshift, ya que la inserción de datos puede ser optimizada, no obstante, las operaciones de borrado y actualización no son recomendadas en este motor de base de datos.

2.2.4.4. Aurora PostgreSQL

Este motor tiene una capacidad para brindar tiempos de respuesta bajos, lo que lo convierte en un candidato ideal para la capa de velocidad, así como brindar la capacidad crear vistas similares a redshift.

Otro aspecto para tomar en cuenta es su bajo costo, ya que se puede utilizar la versión más pequeña de este motor de base de datos a un costo de $12,25, y dado que esta capa no necesita un bloque de datos extenso.

2.2.4.5. Elasticsearch

Consiste en una NoSQL que permite procesar grandes cantidades de datos en tiempo real, basada en el principio de clusterización, lo que permite tener alta disponibilidad sobre los datos.

Aunque su uso principal es de Clave Valor, permite realizar consultas sobre cada campo dentro los datos que tiene registrados, lo que le da una gran ventaja sobre otras NoSQL como lo son DynamoDb, y con un muchos mejores tiempos de respuesta que MongoDb, lo que la convierte en una opción para la capa de velocidad, dado que se puede contratar el servicio a un precio de $25,92 al mes por instancia, pero sí requiere alta disponibilidad se recomienda tener un clúster con nodos impares, es decir, 3, 5, 7 entre otros.

2.2.4.6. Redshift fedérate queries

Es una opción de configuración que permite acceder a otro motor de base de datos desde una conexión de redshift, sin aplicar costos adicionales a la configuración, únicamente los costos de los motores de base de datos, lamentablemente de momento el único motor de base de datos compatible con esta configuración es Aurora PostgreSQL, lo que limita un poco las capacidades de conectar otras herramientas, como lo son Aurora MySQL o cualquier RDS.

No obstante, si se utiliza Redshift para la capa de lotes y Aurora PostgreSQL para la capa de velocidad, el realizar la configuración de fedérate queries o queries federados, puede resultar en una capa de servicio fácil de utilizar e implementar, sin embargo, se debe tomar en cuenta que esta opción permite acceder a las tablas de Aurora PostgreSQL, pero no soporta la creación

de vistas que utilicen tablas que no sean de Redshift, lo que limita el uso de estas capacidades.

2.2.4.7. Amazon Athena

Consiste en una implementación del motor de consultas desarrollado por Facebook llamado PrestoDb, el cual permite normalizar las estructuras de archivos almacenadas en Amazon S3, y convertirlos en tablas, y brindar la opción de realizar consultas sobre dichas tablas.

Esto permite realizar operaciones de exploración sobre los datos con mucha facilidad. La restricción a tomar en cuenta es que, al crear una tabla, se debe proveer una dirección en el S3 en donde los archivos que se encuentren tengan el mismo formato y la misma estructura, de lo contrario Athena no será capaz de leer los datos dentro de los archivos y normalizarlos.

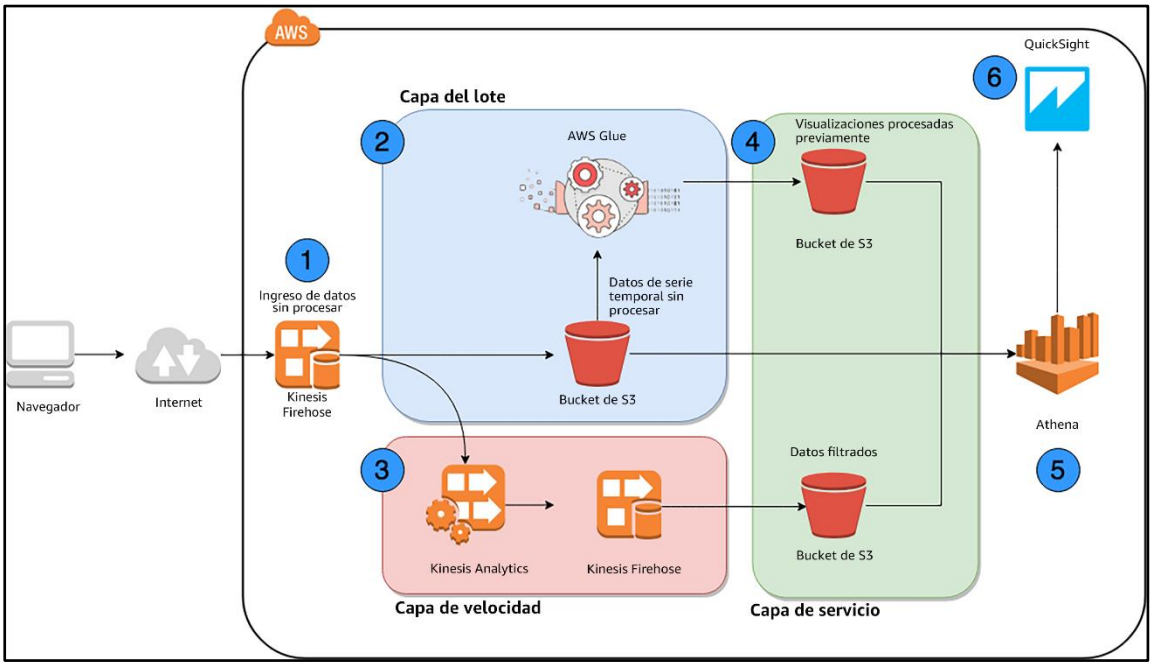
Amazon cobra el servicio por consulta, teniendo un costo de $5,00 por consulta, pero existen técnicas recomendadas para optimizar los costos y el tiempo de respuesta, al particionar y comprimir los archivos adecuadamente. Si en la arquitectura se utiliza únicamente el servicio de S3, Athena es la opción ideal para la construcción de las vistas, dado que permite materializar los resultados de las querencias en una ubicación específica dentro el servicio de S3.

2.2.5. Implementación completa

Amazon AWS provee en su sitio web una propuesta documentada de cómo implementar una arquitectura Lambda, utilizando diferentes servicios para la misma, así mismo, pone a disposición de sus clientes soporte técnico para

implementarla, en caso las organizaciones no cuenten con personal entrenado para realizar dicha implementación. La propuesta es la siguiente.

Figura 14. **Arquitectura Lambda propuesta por Amazon**



Fuente: Amazon AWS. *Arquitectura Lambda*. https://docs.aws.amazon.com/es_es/wellarchitected/latest/analytics-lens/reference-architecture-3.html. Consulta: julio 2021

Como se puede observar esta arquitectura se basa en utilizar Amazon S3 como fuente de almacenamiento principal para las capas de lote y servicio, no obstante, introduce para la capa de velocidad el servicio de Kinesis en sus variantes Firehose y Analytics, que consisten en servicios de colas con la posibilidad de realizar consultas sobre los datos encolados.

2.2.5.1. Capa inicial

Es importante, primero percatarse que Amazon propone implementar una capa inicial previa a ingresar a la capa de lote y la capa de velocidad, en la cual se coloca un servicio Kinesis Firehose, este servicio es una variante del servicio de colas Amazon Kinesis, en el que brinda la comodidad de establecer fuentes y destinos predeterminados, sin necesidad de implementar código a la medida.

Esta capa propone centralizar el ingreso de datos para luego ser trasladado a Amazon S3 para el procesamiento de la capa de lotes y Kinesis Analytics para la capa de velocidad.

2.2.5.2. Capa de lotes

La capa de lotes no presenta una complejidad extensa, ya que, a partir de todos los datos obtenidos de la cola de la capa inicial, plantean utilizar Amazon *Glue*, para poder crear vistas sobre los datos ingresados, y estas vistas luego colocarlas en Amazon S3 para la capa de servicio.

2.2.5.3. Capa de velocidad

Para esta capa Amazon propone la combinación de Kinesis Analytics y Kinesis Firehose. Kinesis Analytics que permite sobre una cola de datos y utilizar un lenguaje de consultas SQL, extraer y transformar la información de las colas, así como complementar con otras fuentes para realizar transformaciones básicas sobre los datos.

A partir de los resultados obtenidos del servicio de Kinesis Analytics se implementa otro servicio de Kinesis Firehose, cuyo único propósito consiste en

llevar los datos ya transformados a un destino dentro de Amazon S3 para la capa de servicio.

2.2.5.4. Capa de servicio

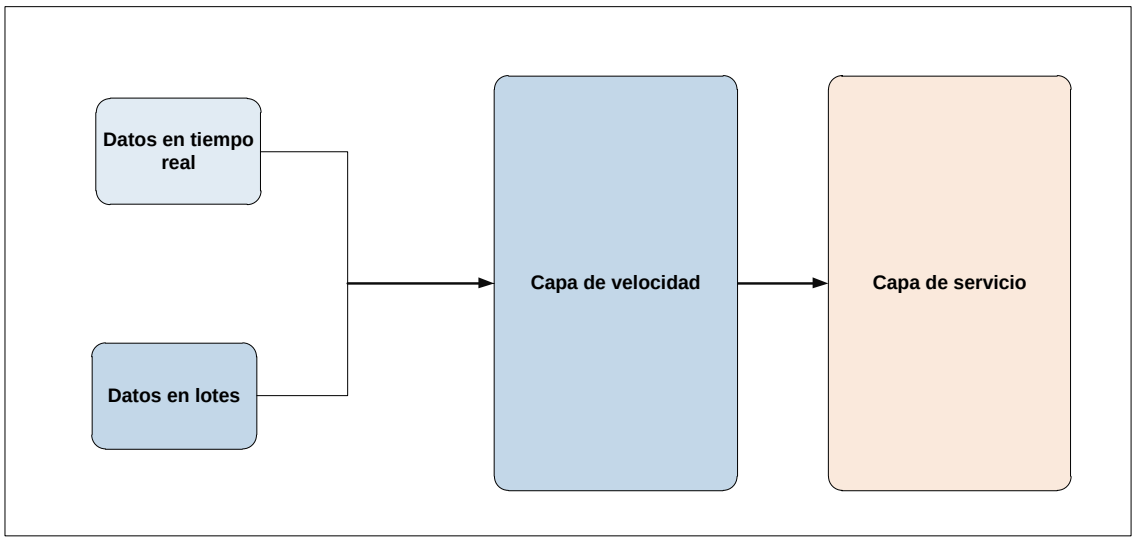
Finalmente, para la capa de servicio la propuesta consiste en utilizar diferentes áreas dentro de un servicio de S3, en donde utilizando el servicio de Amazon Athena, se puede construir vistas que unan los datos que provee la capa de lote y la capa de velocidad, para finalmente estos ser consumidos por cualquier servicio de reportería, en este caso Amazon recomienda el uso de Quick Sight.

2.3. Análisis de arquitectura Kappa

La arquitectura Kappa es una alternativa a la arquitectura Lambda que busca simplificar su implementación, eliminando la capa de lotes, y dejando únicamente la capa de velocidad para procesar datos en tiempo real y datos generados no en tiempo real.

Todos los datos son ingresados a la capa de velocidad implementando una cola de entrada que permita clasificar los datos para que estos puedan ser posteriormente procesados por la capa de velocidad y luego trasladarlos a la capa de servicio. Esto implica que el procesamiento de combinar los datos en tiempo real y los no en tiempo real, se puede realizar directamente en la capa de velocidad ya que estos no residen en otro lugar.

Figura 15. **Arquitectura Kappa**



Fuente: elaboración propia.

Esta arquitectura simplifica mucho los bloques a utilizar para ser implementada en AWS, ya que se necesita definir únicamente el servicio de colas apropiado, que soporte datos en tiempo real y los procesados en lotes. Para la cola de procesamiento, por defecto AWS recomienda utilizar el servicio de Kinesis para procesar todas las fuentes de datos.

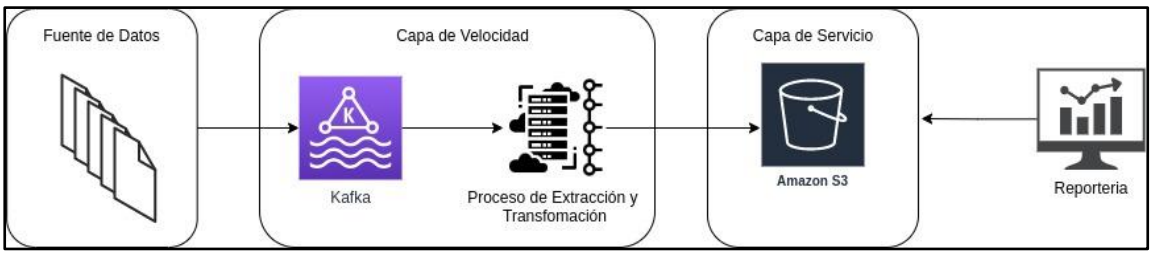
De igual manera, agregando servicio de Lambdas para realizar diferentes transformaciones, pero dado el trabajo que esta cola debe realizar y la importancia que tiene ya que es el único punto de conexión para esta arquitectura, se recomienda utilizar un servicio de colas como Kafka.

Este servicio de colas ofrece dos grandes ventajas sobre cualquier otro servicio, implementa el concepto de temas, lo cual permite utilizar el mismo

servicio y clasificar los datos que se ingresan según su tema, así el servicio que lee la información puede leer los datos generados para un tema en particular.

La otra característica que hace que Kafka destaque por encima de otros servicios es que provee su propio lenguaje de consultas basado en SQL tradicional, lo que permite realizar consultas sobre un tema, e incluso aplicar filtros y transformaciones básica a los datos; lo que facilita el proceso de realizar transformaciones directamente en la capa de velocidad para combinar los datos de diferentes temas en un solo flujo de datos previo a que estos están disponibles en la capa de servicio.

Figura 16. **Arquitectura Kappa en AWS**



Fuente: elaboración propia.

2.3.1. Ventajas

Al ser una variante de la arquitectura Lambda posee las mismas ventajas, agregando la reducción en la infraestructura y la eliminación de la redundancia de datos que se crea entre la capa de lotes y la capa velocidad, lo que hace que sea más compacta y simple de implementar.

2.3.2. Desventajas

Si bien su arquitectura es simple, su implementación puede resultar compleja, ya que el manejo de las colas para distinguir y combinar los datos en lotes y en tiempo real puede resultar complicado.

Si bien esto puede ser mitigado al implementar un servicio de colas como Kafka, se tiene la desventaja que Kafka es un servicio mucho más costoso que Kinesis, siendo casi 4 veces más caro que el anterior, lo que puede volver la implementación de esta arquitectura costosa en implementación o bien monetario.

2.4. Comparativa de herramientas

Al analizar los tres tipos de arquitectura, se han observado diferentes herramientas para realizar la implementación en los servicios de AWS, que van desde almacenamiento crudo como lo pueden ser los servicios de S3, diferentes modelos de bases de datos, herramientas para procesos de transformación y carga.

Finalmente, colas para procesamiento de datos en tiempo real, es por ello que se hará un breve análisis de los bloques propuestos para dar un resumen de que herramientas son las más apropiadas según la necesidad.

2.4.1. Almacenamiento

Para la capa de almacenamiento se ha identificado dos tipos de almacenamiento, el crudo, que es cuando se depositan los archivos o datos obtenidos de los diferentes proveedores para que estos sean procesados

posteriormente, hasta las capas de datos procesados y de fácil consumo, es por ello que se clasifica el almacenamiento en dos, el crudo y el procesado.

2.4.1.1. Almacenamiento crudo

Para la capa de almacenamiento crudo se tiene una clara decisión, que consisten en implementar el servicio de S3, ya que como se ha visto a lo largo de esta investigación, provee un servicio de almacenamiento virtualmente ilimitado a un costo bastante bajo, además de dar otras ventajas como el poderse integrar con herramientas de análisis y transformación de datos, lo que lo hace la elección ideal para los usuarios de AWS.

2.4.1.2. Almacenamiento procesado

Para la parte de almacenamiento procesado se considera utilizar una arquitectura Lambda o Kappa, es válido utilizar un servicio de S3 para este tipo de datos, no obstante, el utilizar una base de datos presenta muchas más ventajas en este caso, dado la facilidad que brinda para realizar consultas en un lenguaje fácil de aprender como lo es el SQL, además de poder indexar y servir los datos a gran velocidad.

Es aquí donde la elección de base de datos tiene la mayor ventaja y surge la pregunta, cuál de todos los modelos de bases de datos se debe utilizar, de los que Amazon provee. La respuesta es, dependiendo de la naturaleza de los datos que se van a trabajar, si los datos tienen estructura relacional, donde se realizaran muchos cambios en corto tiempo.

Así mismo los datos a consultar son usualmente los más recientes, entonces utilizar una base de datos relacional es la mejor opción, en donde se

recomienda darle prioridad a las versiones de Aurora que brinda AWS por su velocidad y almacenamiento virtualmente ilimitado, cabe resaltar que este es de 64 Terabytes.

Por otro lado, si es necesario utilizar una base de datos propietaria, los servicios de RDS son la única opción. Ahora bien, si el volumen de datos a consultar en la capa de reporteria es considerable ya que se debe ser capaz de servir datos de fechas lejanas, utilizar una base de datos columnar como Redshift, es la mejor opción.

Al analizar la estructura de una arquitectura tradicional que consiste en la implementación de un Repositorio de datos que contenga uno o más Data Marts, Redshift es una opción superior a una base de datos relacional, ya que un repositorio de datos debe ser capaz de procesar una gran cantidad de datos con tiempos de respuesta bajos.

2.4.2. Herramientas de extracción

Para las herramientas de extracción, se han analizado diferentes opciones que AWS proporciona, donde las mejores se consideran dos, siendo la primera Airflow, dado que permite realizar todo tipo de procesos de extracción, transformación y carga, dando libertad incluso para calendarizar los procesos en cualquier momento, así como proveer diferentes integraciones de terceros que se pueden implementar fácilmente.

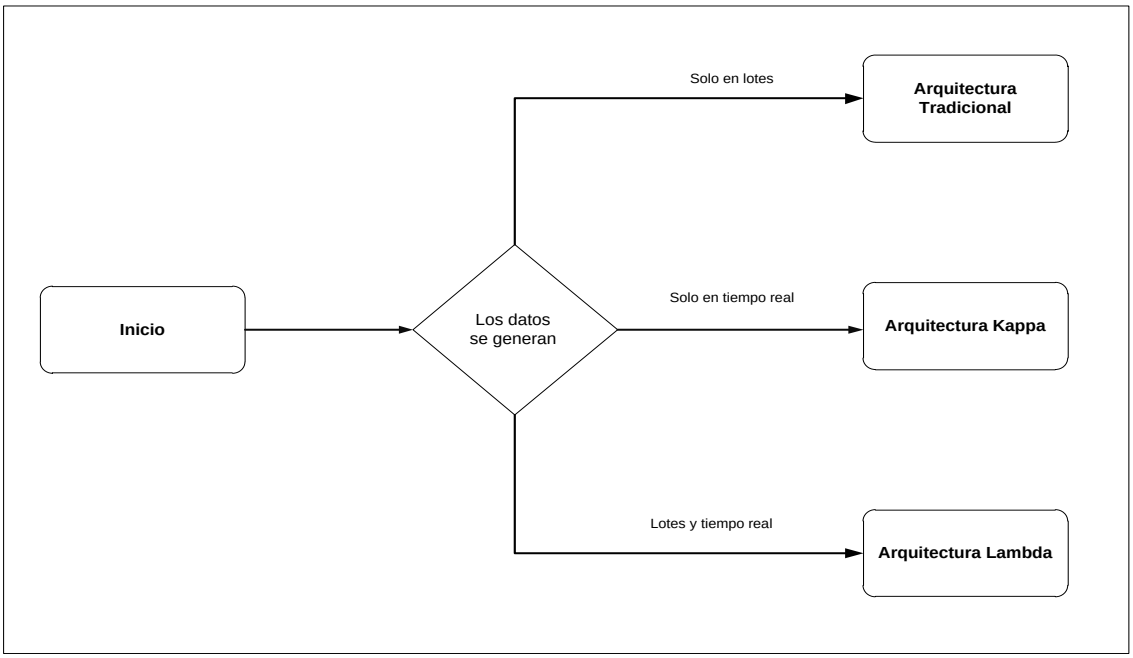
No obstante, tiene la limitante de que los procesos que se consideran deben ser cortos, o de ejecución rápida, ya que procesos que demoren mucho tiempo, pueden generar un bloqueo de recursos que atrase otros procesos.

Es aquí donde utilizar Glue es una opción viable, cuando se necesita procesar bloques de datos muy grandes, ya que al utilizar Glue, se puede reservar recursos específicos para el proceso, así como duplicarlos fácilmente, si esto fuera necesario para garantizar que el proceso sea lo más rápido posible. Finalmente se puede calendarizar el proceso de Glue utilizando Airflow, por lo que este pude convertirse en el centro de todos los procesos, aunque estos no necesariamente sean ejecutados por Airflow.

2.5. Propuesta de arquitectura

Se han analizado tres arquitecturas con propósitos aparentemente similares, que es permitir obtener, transformar y servir los datos a los usuarios finales, con la finalidad de que estos usuarios puedan ser asistidos en el proceso de toma de decisiones y reduciendo las decisiones realizar por instinto. Según los casos de uso, una arquitectura puede resultar más apropiada que otra, es por ello que se presenta el siguiente diagrama para asistir el proceso de tomar una decisión:

Figura 17. **Decisión de arquitectura**



Fuente: elaboración propia.

2.5.1. Propuesta final

Si bien se ha establecido que según la frecuencia de los datos que se generan, debe ser la decisión de implementación, en el mundo real, el escenario más común que se encuentra es la necesidad de implementar una arquitectura para datos en lotes y tiempo real, por lo que se puede pensar que la arquitectura Lambda es la más apropiada de utilizar en la mayoría de los casos de uso.

No obstante, existe un requerimiento adicional que la arquitectura Lambda no es capaz de resolver o al menos no de una manera sencilla, el cual consiste en proveer visibilidad sobre Data histórica, si bien la capa de lotes debería ser capaz de proveer este tipo de datos en la capa de servicio, cuando existe la

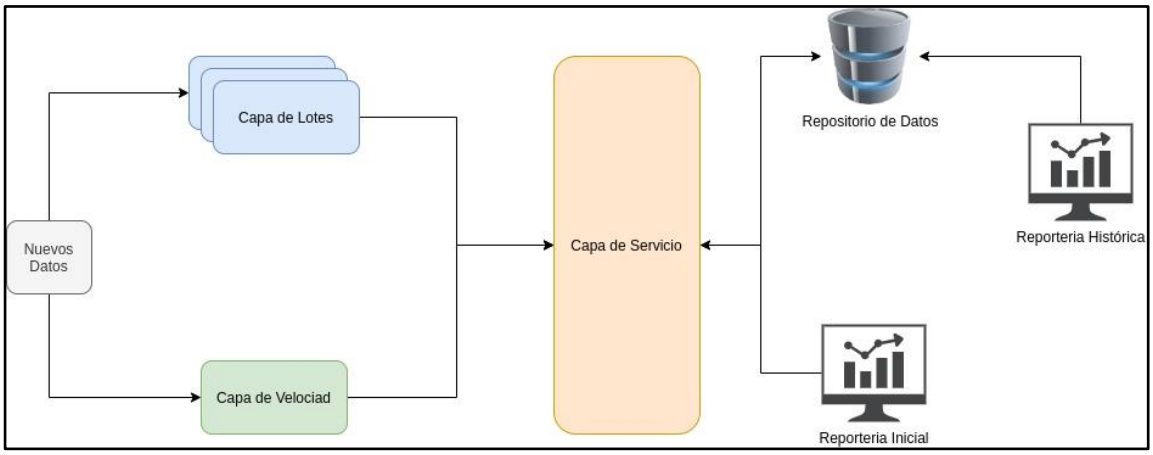
necesidad de analizar datos históricos el cual consiste en un volumen considerable, la arquitectura Lambda puede ser no la más apropiada.

Este caso de uso resuena como el caso para una arquitectura tradicional donde se implemente un repositorio de datos con al menos uno o más Data Marts. Es por ello que la propuesta final consiste en una arquitectura hibrida que combine la arquitectura Lambda con una arquitectura tradicional.

Para combinar estas dos arquitecturas, basta con agregar un bloque adicional, en el que los datos generados en la capa de servicio sean llevados a la maquinaria para que estos sean procesados en un Data Mart que permita consumir y analizar los datos históricos de la organización.

Finalmente, la capa de velocidad puede ser optimizada, si se desea; utilizando los principios de la arquitectura Kappa, lo que permite tener diferentes temas de datos en tiempo real y poderlos cambiar fácilmente con los datos en lotes; con lo que se concluye que la mejor opción de arquitectura es la combinación de las tres arquitecturas, combinando sus ventajas para poder brindar un ecosistema robusto para suplir todas las necesidades de datos de la organización.

Figura 18. **Arquitectura hibrida**



Fuente: elaboración propia.

CONCLUSIONES

1. Se logró diseñar una arquitectura hibrida que combina la arquitectura Lambda con una arquitectura tradicional, comparando las ventajas para poder ofrecer un ecosistema robusto y así implementar los diferentes bloques conceptuales cubriendo las necesidades de datos de la organización.

2. Se definieron los conceptos modernos que constituyen una arquitectura de datos, para identificar si existen componentes adicionales o bien componentes que no son necesarios en una arquitectura de datos en tiempo real.

3. Se analizaron las tres arquitecturas que permiten obtener, transformar y servir los datos a los usuarios finales, con base al análisis se determinó que la arquitectura Lambda es la más apropiada de usar en las necesidades que presenta actualmente la organización.

4. De las herramientas que AWS provee para administrar datos y facilitar los procesos, resaltan las funciones Lambda que, ejecuta porciones de código sin necesidad de administrar servidores, otra de las herramientas es Data Pipelines que facilita la creación de flujos de datos que usan los recursos más comunes, Glue es otra herramienta de procesos de extracción, transformación y carga, enfocado esencialmente en procesos de carga de grandes cantidades de datos, la última de las herramientas mencionada es Airflow que permite calendarizar y ejecutar cualquier clase de flujo de datos.

5. La arquitectura hibrida que es la elegida para responder a las necesidades de la organización, es una combinación de Lambda y tradicional que agrega un bloque adicional, en el que los datos generados en la capa de servicio son llevados a la maquinaria donde estos son procesados en un Data Mart permiten consumir y analizar los datos históricos de la organización.

RECOMENDACIONES

1. Realizar un análisis profundo para determinar las implicaciones que puede tener una organización al implementar una arquitectura hibridad, así como determinar los elementos necesarios para poder tener una implementación exitosa.

2. Considerar las ventajas y desventajas de las herramientas de AWS, para poder implementar las políticas de manejo y gestión de riesgos del sistema en la organización con el personal o Ingeniero a cargo.

3. Analizar de manera exhaustiva, las diferentes herramientas que AWS provee para administrador datos y facilitar los procesos de la organización, esto con el objetivo del aprendizaje del mismo, capacitación del equipo y diseño de ambientes de desarrollo, pruebas y producción, previo a realizar la implementación de la arquitectura hibrida.

4. Investigar un poco en referencia a las nuevas tecnologías que son de utilidad para el éxito y productividad de la organización, porque permanecer estáticos en el mundo actual donde los avances tecnológicos están a la mano, no es recomendable ya que cada poco tiempo surge nuevas tecnologías con el único objetivo esencial de facilitar la gestión y operatividad empresarial.

BIBLIOGRAFÍA

1. AMAZON AWS. *Arquitectura Lambda.* [en linea]. <https://docs.aws.amazon.com/es_es/wellarchitected/latest/analytics-lens/reference-architecture-3.html>. [Consulta: julio 2021].

2. AMAZON WEB. *SERVICES*. *Introducción a Amazon Web Services*. [en linea]. <http://aws.amazon.com/es/>. [Consulta: marzo de 2021].

3. _____. *What Is Amazon Elasticsearch Service?* [en línea]. <http://docs.aws.amazon.com/-elasticsearch-service/-latest/ developer guide/what-is-amazon-elasticsearch-service.html>. [Consulta: febrero de 2021].

4. ARISER. *Historia del Big Data – Del comienzo del análisis de datos a nuestros días*. [en línea]. <https://bigdataparacuriosos.wordpress.com/historia-big-data/>. [Consulta: enero de 2021].

5. BAQUIA, Manuel Daza. *¿Es el Big Data el fin de la Bussiness Intelligence?* [en línea]. <http://www.baquia.com/emprendedores/ 2013-07-19-manuel-daza-bigdata-business-intelligence -bidiscovery-informacion -datos-empresas-olapanalisis-qlikview-ibmsmart-cities-redes-sociales>. [Consulta: marzo de 2021].

6. BARKER, Dan. *The Real Original Source of the Phrase Big Data. Dan Barker*. [en línea]. <http://barker.co.uk/bigdata>. [Consulta: marzo de 2021].

7. BARR Jeff; NARIN, Attila; VARIA, Jinesh. *Building Fault Tolerant Applications on AWS.* [en línea]. <http://aws.amazon. com/es/whitepapers/>. [Consulta: febrero de 2021].

8. BARRANCO FRAGOSO, Ricardo. *¿Qué es Big Data?* [en línea]. <https://www.ibm.com/developerworks/ssa/local/im/que-es-bigdata/>. [Consulta: enero de 2021].

9. BBC. *La ciencia de encontrar perlas en el Big Data*. [en línea]. <http://www.bbc.com/mundo/noticias/2013/03/130310_tecnologia_ big_data_datos_informacion_perladp>. [Consulta: marzo de 2021].

10. BORJA. *¿Qué es Redis?* [en línea]. <http://www.antweb.es/servidores/redis-todo-lo-que-debes-saber>. [Consulta: enero de 2021].

11. BROZINSKY, Murray. *The future of Big Data: Deep Learning.* [en línea]. <http://www.talix.com/blog/the-future-of-big-data-deep-learning/>. [Consulta: marzo de 2021].

12. BRYANT, KATZ y LAZOWSKA. *Big-Data Computing: Creating revolutionary breakthroughs in commerce, science, and society.* [en línea]. <https://cra.org/ccc/wpcontent/uploads/sites/2/2015/05/ Big_Data.pd>. [Consulta: marzo 2021].

13. CANTABRIATIC. *NoSQL y el teorema de CAP*. [en línea]. <http://www.cantabriatic. com/nosql-y-el- teorema -de-cap/>. [Consulta: diciembre de 2020].

14. CARTER, Philip. *Big Data Analytics: Future Architectures, Skills and Roadmaps for the CIO*. [en línea]. <https://docplayer.net/2544559-Big-data-analytics-future-architectures-skills-and-roadmaps-for-the-cio.html>. [Consulta: abril 2021].

15. CHEE, Brian J.S.; FRANKLIN, Curtis Jr. *W*hat is a Cloud? Technologies and Strategies of the Ubiquitous Data Center. *Boca Ratón*. USA: Taylor & Francis Group. 2010. 288 p.

16. COX, Michael y ELLSWORH, David. *Application-Controlled Demand Paging for Out-of-Core Visualization*. Estados Unidos: NASA Ames Research Center. 1997. 12 p.

17. DIJKSTRA, Edsger. *The Structure of the Multiprogramming System*. [en línea]. <https://www.cs.utexas.edu/users/EWD/ewd01xx/EWD196.PDF>. [Consulta: marzo de 2021].

18. GARCÍA, Luis Miguel. *Arquitectura Lambda: Principios de Arquitectura para sistemas Big Data en tiempo real.* [en línea]. <https://unpocodejava.wordpress.com/2013/09/07/arquitecturalambda principios-de-arquitectura-parasistemas-big-data-entiempo-real/>. [Consulta: abril de 2021].

19. IBM. *What is cloud computing?* [en línea]. <https://www.ibm.com/cloudcomputing/learn-more/what-is-cloud-computing/>. [Consulta: mayo de 2021].

20. Keen, Peter, MORTON, Michael. *Definition of DSS. A Research Perspective*. Massachusetts: Massachusetts Institute of Technology, 1980. 50 p.

21. MARQUINA, Julián. *La vida en la nube: Big data y cloud computing*. [en línea]. <http://www.julianmarquina.es/la-vida-en-la-nube-big-datay-cloud-computing/>. [Consulta: diciembre de 2020].

22. MARTÍNEZ, Desiree *Big Data, Customer Intelligence*. [en línea]. <http://artyco.com/principales- diferencias- entrebusiness-intelligence-y big-data/>. [Consulta: noviembre de 2020].

23. Tomcy John, PANKAJ Misra. *Data Lake for Enterprises*. [en linea]. <https://learning.oreilly.com/library/view/data-lafor/9781787281349 for/9781787281349/301fc654-0f78-4058bbde036b938bee.xhtml>. [Consulta: abril de 2021].