Sistema de Recomendaciones en Tiempo Real Para Una Plataforma de Comercio Electrónico Utilizando Bases de Datos de Familias de Columnas

Isaac Valle Granados, Fatima Daniela Saborío Rivera, Olger Leonardo Chinchilla Segura, Minor Andrey Porras Salgado

Escuela de estadística, Universidad de Costa Rica, San José, Costa Rica
isaac.valle@ucr.ac.cr, fatima.saborio@ucr.ac.cr, olger.chinchilla@ucr.ac.cr, minor.porrassalgado@ucr.ac.cr

***Resumen*— Debido a las limitaciones de escalabilidad horizontal y la estructura rígida de las bases de datos SQL se plantea la necesidad de buscar otro tipo de bases de datos que puedan manejar las grandes cantidades de datos que poseen las plataformas de comercio electrónico, una solución para esta limitación es utilizar bases de datos NoSQL las cuales con sacrificios de diferentes propiedades de las bases de datos SQL pueden manejar esta cantidad de datos masiva. El presente artículo explora las diferencias que existen entre dos softwares que utilizan bases de datos NoSQL, los cuales son Cassandra y HBase, esta comparación se realiza con el fin de encontrar si alguno de estos softwares funciona de una manera óptima para la problemática planteada. Para la comparación se toman características como la consistencia, el desempeño de la inserción de datos, el desempeño de la recuperación de datos, la facilidad de uso, las herramientas de gestión que poseen, los métodos de consulta y el almacenamiento. Se encuentra que ambos softwares cumplen con las funcionalidades de los tipos de bases de familias de columnas y son capaces de realizar las bases de datos necesarias para darle una solución a la problemática, sin embargo, si se busca velocidad y escalabilidad, Cassandra es la opción ideal. Si la prioridad es la precisión de datos y la recuperación ante fallos, Hbase es la mejor alternativa.**

Palabras clave: Bases de datos NoSQL, Cassandra, HBase, familias de columnas.

I. INTRODUCCIÓN

Los documentos relacionados a la metodología implementada en este artículo pueden visualizarse en el repositorio: https://github.com/redqbert/Proyecto-I-Sistemas-de-Informacion.git

El volumen de datos generado por las plataformas de comercio electrónico ha aumentado de manera significativa en los últimos años, impulsado por el crecimiento en el número de transacciones, usuarios y productos disponibles. Las constantes actividades, que incluyen compras, búsquedas y visualizaciones de productos, producen una gran cantidad de información que necesita ser procesada y analizada de forma efectiva para mejorar la experiencia del usuario y optimizar las estrategias comerciales.

Esta situación ha desafiado las capacidades de las bases de datos relacionales tradicionales, que, debido a su estructura rígida y limitaciones en escalabilidad horizontal, no son adecuadas para manejar estos volúmenes masivos de datos en tiempo real. Ante estas limitaciones, surgieron las tecnologías NoSQL [1], que sacrifican algunas propiedades de las bases de datos relacionales, como la atomicidad, consistencia, aislamiento y durabilidad (ACID), para ofrecer ventajas como un mayor rendimiento, escalabilidad, y flexibilidad en la representación de datos.

Por lo anterior, con este artículo se busca realizar una comparación utilizando dos softwares con el fin de encontrar las diferencias entre ellos y así realizar conclusiones con respecto al almacén masivo de datos y uso de bases de datos NoSQL.

Tomando en cuenta el enfoque del tema se decide trabajar con dos softwares que utilizan al tipo de sistema de familias de columnas, definida por J. Martínez como [2] un tipo de bases de datos tienen un acceso rápido al almacén de los datos y cuenta con un alta

facilidad en las tareas de lectura y escritura. Además, como su nombre lo dice, los datos se almacenan en columnas, estas se forman de familias de columnas que agrupan
columnas en la base de datos; cuentan con una clave para identificar columnas.

Como primer base de datos NoSQL del tipo de familia de columnas, se utiliza Cassandra, la cual, "Se encuentra entre las bases de datos NoSQL que han abordado las limitaciones de las tecnologías de gestión de datos anteriores, como las bases de datos SQL" [3], algunas de las ventajas que brinda utilizarla: La escalabilidad horizontal, flexibilidad y el tipo de arquitectura que posee que son distribuidas.

Asimismo, como la segunda base de datos se utilizará Hbase, como menciona la IBM [4] este es un sistema de bases de datos efectuado sobre Hadoop, este se prefiere en situaciones en las que se amerita procesar datos en tiempo real o el acceso a grandes cantidades de datos, además, este permite conjuntos de datos primitivos.

II.DESCRIPCIÓN DEL PROBLEMA

En el entorno digital actual, las plataformas de comercio electrónico(e-commerce) se han visto enfrentadas al desafío de la cantidad masiva de grandes volúmenes de datos generados por millones de usuarios y transacciones diarias en la plataforma.

Los datos que genera la plataforma son extremadamente variados y siempre se encuentran en constante cambio, lo que hace que las bases de datos relacionales tradicionales con sus propios esquemas sean bastante rígidas, y en consecuencia no sean lo suficientemente flexibles ni escalables para manejar las crecientes demandas de estos entornos.

III.SOLUCIÓN IMPLEMENTADA

Este trabajo se centrará en el desarrollo de un sistema de recomendaciones en tiempo real para una plataforma de comercio electrónico, el objetivo es aprovechar bases de datos orientadas a columnas como lo son Cassandra y HBase para analizar el comportamiento de los usuarios de un comercio virtual en cuanto a compras, búsquedas y productos visualizados y con ello generar recomendaciones personalizadas. Se analizarán en ambos sistemas: consistencia, inserción de datos, recuperación de datos, facilidad de uso, herramientas de gestión, método para hacer consultas y almacenamiento con el fin de seleccionar la solución más práctica para el problema.

Ante estas limitaciones, las bases de datos no relacionales (particularmente las orientadas a columnas) han emergido como una solución eficiente para almacenar y procesar grandes volúmenes de datos de forma distribuida, lo que significa que los datos se dividen y almacenan en
varios servidores haciendo que el manejo sea más eficiente que si solo estuviera en una única computadora.

Con el fin de dar una solución a la problemática planteada se creó la misma base de datos en los dos softwares escogidos, tomando en cuenta que el objetivo del estudio radica en el análisis de las diferencias entre estos softwares se opta por utilizar las funciones propias de ellos. Para Cassandra se utilizó Apache Cassandra para Windows en su versión 3.11.10, el cual se basa en Java, más específicamente Java Development Kit (JDK) 8, este se ejecuta desde Python en su versión 2.7.18. Por otro lado, para HBase se utilizó la versión 2.4.9, ejecutada en un entorno Ubuntu a través de la terminal WSL (Windows Subsystem for Linux) en Windows.

Para ambos softwares la base de datos tenía la misma estructura, la cual consistía en crear 6 tablas con 50 datos cada una, las cuales facilitarán la interacción que tenían los usuarios en esta plataforma de comercio, en primer lugar se tiene la tabla de interacciones del usuario con la cual se buscaba ver de forma general las acciones que realizaban cada uno de estos usuarios, luego se tiene la tabla de productos populares de la cual se pueden obtener los productos por categoría y la cantidad de interacciones que tuvieron los usuarios con cada producto. Asimismo, se creó una tabla que contiene el historial de compras de cada usuario, así como el momento en el que lo hizo, además, se creó la tabla de compras complementarias, esta lo que muestra son los productos que se compraron junto con los productos principales, esto con el fin de realizar recomendaciones a otros usuarios que compran los productos populares, para que compren los productos complementarios. También se tomó en cuenta una tabla que contiene los productos visualizados por los usuarios, esta con el fin de analizar el momento en el que visualizaron el producto en cuestión. Por último, se creó la tabla del historial de búsqueda de los usuarios, con el fin de conocer el momento en el que hicieron la búsqueda y las palabras que usaron para realizar esta búsqueda.

Es importante mencionar que Cassandra, como menciona A. Peña [5] utiliza una versión más simplificada de SQL, que se llama Cassandra Query Language (CQL), este es diferente del SQL porque se utilizan datos desnormalizados. Por otro lado, HBase utiliza el lenguaje de programación de Java.

IV.COMPARACIÓN DE LAS CARACTERÍSTICAS

Como se plantea anteriormente, uno de los contenidos de esta investigación es comparar en diferentes aspectos los softwares Cassandra y HBase. A continuación, se mencionan estas comparaciones.

A. Consistencia

En Cassandra, la consistencia se refiere a qué tan sincronizadas están las réplicas de un dato dentro del respectivo clúster. Este concepto se ve reflejado dentro del Teorema CAP, el cual dicta que un sistema distribuido no puede conseguir tener las tres siglas de este teorema [6]. Por lo tanto, los sistemas distribuidos deben optar por dos de estas propiedades, sacrificando la tercera. Cassandra prioriza la alta disponibilidad (A) y la tolerancia a particiones o fallos (P), lo que significa que sacrifica la consistencia (C), dando lugar a la "consistencia eventual". La consistencia eventual significa que después de una actualización, no todas las réplicas de los datos se actualizan inmediatamente, pero eventualmente todos los nodos del clúster contendrán la misma versión de los datos. [7]

También es importante añadir que la consistencia en Cassandra es configurable, lo que permite ajustarla según las necesidades del sistema en un momento dado. Por ejemplo, se pueden definir distintos niveles de consistencia para lecturas y escrituras, desde "una réplica" (una baja consistencia) hasta "todas las réplicas" (alta consistencia) [7]. Básicamente a mayor nivel de consistencia, más nodos tienen que confirmar la operación, lo que aumenta la confiabilidad, pero puede afectar la velocidad.

Por otro lado, en cuanto a la replicación de datos, Cassandra almacena cada fila en múltiples nodos dentro de un clúster, lo que asegura que, si un nodo falla, los datos aún estarán disponibles en otros nodos [7]. Este mecanismo garantiza la alta disponibilidad y la tolerancia a fallos, como se mencionó anteriormente.

HBase proporciona un alto nivel de consistencia tanto en lectura como en escritura [8]. Esto significa que los clientes no pueden acceder a registros inconsistentes hasta que se resuelvan las inconsistencias. A diferencia de Cassandra, que utiliza un enfoque de "consistencia eventual", HBase garantiza una consistencia fuerte, asegurando que cualquier operación de escritura sea inmediatamente visible para las lecturas subsiguientes. Esta característica es esencial para aplicaciones donde la precisión y la actualización instantánea de la información son cruciales.

Además, la replicación asíncrona puede introducir retrasos entre nodos, garantizando una consistencia fuerte dentro de cada región. Lo anterior, significa que las operaciones en una región específica están garantizadas para ser consistentes, aunque puedan existir retrasos en la propagación de datos a otras regiones Es crucial para aplicaciones que requieren datos actualizados en tiempo real. Para evaluar el impacto de estas estrategias de replicación y consistencia, se utilizan cargas de trabajo específicas. “Realizamos varios esfuerzos de evaluación comparativa para brindar resultados de rendimiento para diferentes estrategias de replicación/consistencia” [8].

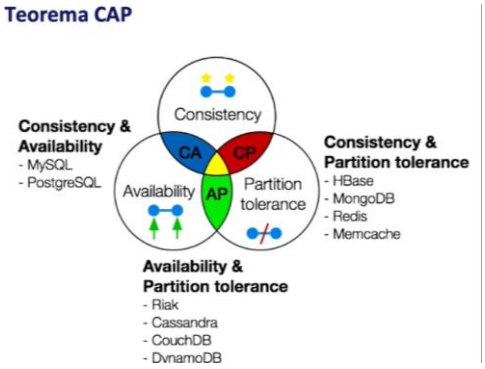

Fig. 1 Conjuntos del Teorema CAP

B. Desempeño en inserción de datos

La forma más esencial para la inserción de los datos en Cassandra es por medio del shell de CQL. Solo que, primeramente, para poder hacerlo debemos abrir otra línea de comandos ahí mismo e introducir “>cqlsh”; ya con eso, ya se puede introducir comandos CQL para la interacción con el servidor de Cassandra. Inicialmente debe crearse una Keyspace (base de datos), y ya dentro de ella es donde se puede iniciar la inserción de los datos. Además, según A. Peña [5] una ventaja del lenguaje CQL es que dispone de sentencias muy parecidas a SQL, por lo que, si ya se maneja bien este último, no debería haber mayores complicaciones a la hora de insertar los datos.

En aspectos más característicos, Cassandra específicamente no cuenta con integridad referencial, lo que ayuda a que las inserciones sean más rápidas, pues no es necesario verificar claves foráneas ni relaciones entre tablas, eliminando la sobrecarga de asegurarse de que esas referencias entre tablas sean consistentes. [9] Otros aspectos importantes a la hora de analizar el desempeño es la desnormalización de los datos y el buen diseño para un almacenamiento óptimo.

Al desarrollar un sistema de recomendaciones en tiempo real para una plataforma de comercio electrónico, es fundamental considerar el desempeño de inserción de datos en HBase. Aunque HBase tiene la capacidad de manejar grandes volúmenes de datos, su modelo de consistencia fuerte puede generar una latencia mayor en las inserciones en comparación con otras bases de datos NoSQL que optan por consistencia eventual [1].

Esta necesidad de confirmar que los datos se han escrito correctamente antes de permitir nuevas operaciones puede limitar el rendimiento en aplicaciones que requieren actualizaciones rápidas, como el análisis del comportamiento del usuario y la generación de recomendaciones personalizadas. [1] Esto es especialmente relevante en entornos donde la inmediatez de las inserciones es crucial, como en sistemas transaccionales o aplicaciones financieras, lo que podría impactar negativamente en la eficacia del sistema de recomendaciones que se busca implementar.

C. Desempeño en recuperación de datos

En Cassandra, para poder ver o describir el desempeño en recuperación de datos, es importante hacer hincapié en la desnormalización, pues se puede decir que no solo es aceptada, sino que es esencial para mejorar el rendimiento de las consultas y la recuperación. A la hora de recuperar datos, Cassandra no permite realizar joins entre las tablas de una misma base de datos, por lo que el crear tablas desnormalizadas es necesario. [9] Esto puede ser una ventaja o desventaja dependiendo el punto de vista en el que lo vea una persona, pues no puede resultar siendo un aspecto negativo a menos que la persona no sea simpatizante de la redundancia en los datos (duplicación).

Como menciona Sreekzz [10], este sistema de gestión de base de datos no relacionales cuenta con un aspecto muy importante y es su capacidad de manejar fallos, al lado de HDFS(sistema de archivos distribuido de Hadoop) proporciona resiliencia ante eventuales fallos de hardware por lo que permite la recuperación de datos a partir de copias de seguridad en el caso de que se pierda un nodo es decir una unidad básica de un sistema distribuido(por ejemplo de un cluster de computadoras)cita.

Según Al Alshammari, Al-Naymat, y Hadi [11], cuando existe corrupción o pérdida de los datos puede ser utilizado para extraer bloque de datos de HDFS la técnicas del file carving, además, la velocidad de recuperación de los datos puede verse beneficiada si existe compresión en los datos y ajustes de memoria como el tamaño de la caché.

D. Facilidad de uso

Cassandra no es muy complicado de utilizar para las personas que tengan algún conocimiento básico en instalación de softwares, pues se distribuye en formato .zip, lo que facilita su instalación, tan solo es suficiente con descomprimir el archivo en una ubicación deseada; solamente que es necesario cumplir con ciertos requisitos previos para que Cassandra funcione correctamente. Uno de ellos es que Cassandra depende de las versiones de Python 2.7.X, esto puede generar inconvenientes para aquellos usuarios que cuenten con versiones más recientes de Python como la 3.11, ya que las versiones superiores a la 2.7.X no son compatibles. Aunque es posible tener varias versiones de Python instaladas en el mismo sistema, es esencial y necesario que la variable de entorno Path esté configurada para señalar la ruta de instalación de Python 2.7. [5]

Por otro lado, Cassandra requiere también de Java 8 como entorno de ejecución. Aunque puede funcionar con versiones como Java 7 o 10, lo más recomendable es tener instalada la versión 8 y configurar la variable de entorno JAVA_HOME para que apunte a esta versión y se asegure una compatibilidad adecuada. [5]

El proceso de instalación y configuración de HBase puede ser un reto para quienes no poseen experiencia previa en sistemas operativos como Ubuntu, ya que sin conocimientos técnicos en entornos basados en Linux la curva de aprendizaje es más pronunciada, debido a que es necesario dominar comandos básicos y gestionar herramientas para lograr una instalación correcta, sin embargo, cuando se supera este obstáculo inicial, la inserción de datos en HBase resulta sencilla, aunque si se requiere cierta familiaridad con conceptos clave, como la estructura orientada a columnas, para entender plenamente el funcionamiento del sistema.

Por otro lado, el análisis de los datos almacenados en HBase presenta mayores desafíos. Para realizar consultas básicas es importante contar con habilidades de programación en Python 3, y si se desea realizar análisis más avanzados, es esencial el manejo de librerías de machine learning y otros frameworks especializados, esto implica que, aunque HBase resulta sencilla en la hora de la gestión de grandes volúmenes de datos, extraer información valiosa de ellos demanda un conocimiento más profundo en áreas como la programación y el análisis de datos.

E. Herramientas de gestión

Según J. Carpenter y E. Hewitt [12] el software Cassandra tiene varias herramientas de gestión o de modelado de datos como Hackolade, Kashlev, DevCenter, entre otros. Sin embargo, el más utilizado es el Cassandra Query Language Shell (CQLSH), este ya viene incluido en el archivo .zip de Cassandra y se encuentra en el directorio bin. Al iniciar el CQLSH, deben conectarse a un nodo del clúster, donde pueden ejecutar comandos CQL para crear y gestionar esquemas, insertar y consultar datos.

En HBase existen diversas herramientas de gestión que facilitan su uso, una de las más importantes es HBase Shell, el cual es una interfaz de comandos que permite a los usuarios realizar acciones básicas como crear y eliminar tablas, así como insertar y consultar datos. Luego se tiene Apache Phoenix [13], el cual permite realizar consultas utilizando un lenguaje más parecido a SQL, por lo que con esto se logra simplificar el acceso y el manejo de los datos, además HBase cuenta con una interfaz web que proporciona una vista general del sistema.

F. Método utilizado para hacer consultas

En Hbase se puede emplear *HappyBase*, la cual es una biblioteca de Python que facilita la interacción con HBase y que permite realizar consultas sobre grandes volúmenes de datos, además de poder hacer consultas complejas. Con *HappyBase*, es posible acceder a tablas y filas específicas, escanear tablas completas y aplicar filtros para recuperar solo los registros que cumplen con ciertas condiciones, además ofrece la capacidad de insertar y actualizar datos, lo que la convierte en una herramienta completa para gestionar y manipular información [14]. Esta funcionalidad permite optimizar flujos de trabajo, mejorando así la accesibilidad y el análisis de datos en tiempo real tal como se puede observar en el anexo 1 en el que se realizaron consultas sencillas como ver información de la persona con cierto id, en estas consultas lo que se hace es establecer una conexión mediante socket a Hbase, dicho sea de paso, *HappyBase* puede además de hacer consultas agregar datos, como si fuese el Shell de Hbase.

En el Shell del Hbase se pueden observar las listas de tablas con LIST, también observar la estructura con

DESCRIBE, además de eliminar una tabla con DISABLE y posteriormente con DROP. En en anexo 1 se muestran algunos ejemplos de consultas con HBase.

Por otro lado, para Cassandra, como menciona S. Aggarwal [15] se utiliza la consola CQLSH para interactuar con Cassandra, con el lenguaje antes mencionado CQL, este al ser muy similar al lenguaje SQL el cual es muy utilizado para la creación se bases de datos hace que personas experimentadas con SQL no tengan tantas dificultades al utilizar Cassandra por primera vez; Esta consola debe estar correctamente instalada en el dispositivo para poder utilizar todos los paquetes de Cassandra.

Asimismo, existen diferentes formas de interactuar con Cassandra mediante esta consola, tal y como explica A. Peña [5] es posible utilizarlo a partir de del shell de CQL, CQLSHELL y por medio de drivers que aplicaciones como Java y Python puedan hacer consultas con CQL. Para esta investigación se utilizaron los drivers de Java y Python para la creación de la base de datos y las consultas pertinentes a partir de esta.

Las consultas en Cassandra se pueden realizar simplemente utilizando declaraciones SELECT, estas pueden extraer toda la información de una tabla utilizando un asterisco (*), sin embargo, se pueden realizar consultas más elaboradas utilizando condiciones como lo son, por ejemplo, WHERE, ORDER BY y IN, las cuales pueden ser múltiples, utilizando conectores como AND/OR. Aunque es importante mencionar que Cassandra no admite consultas utilizando JOINS por la desnormalización de los datos, asimismo, la condición GROUP BY tampoco es admitida si no se refiere únicamente a la llave primaria. Por último, algunas consultas requieren un filtrado completo, sin embargo, Cassandra por sí sola no permite consultas en las que se hace un análisis completo, por esto se debe agregar ALLOW FILTERING [16] al final de la consulta, aunque esto no es necesariamente eficiente para consultas de muchos datos, la mejor opción es hacer un análisis del modelo de la base de datos. En el anexo 2 se muestran algunos ejemplos de consultas que se pueden realizar en Cassandra para la base de datos realizada.

G. Almacenamiento

El almacenamiento de datos es un aspecto crítico para el desarrollo de un sistema de recomendaciones en tiempo real en una plataforma de comercio electrónico. HBase se destaca por su modelo de almacenamiento orientado a columnas, donde cada fila está compuesta por una clave y múltiples atributos.

Esta estructura permite un acceso más eficiente a los datos, especialmente para consultas que requieren leer solo un subconjunto de columnas. Además, HBase opera sin esquemas rígidos, lo que brinda una gran flexibilidad en el modelado de datos y resulta ideal para manejar grandes volúmenes de información no estructurada o semi-estructurada [1]. En un entorno de Big Data, estas características son esenciales, ya que facilitan la adaptación del sistema a diferentes tipos de datos y aplicaciones, optimizando el rendimiento en consultas específicas.

Para el caso de Cassandra, según Amazon Web Service [14] esta utiliza un almacén de datos que define tablas por filas y columnas llamado almacén de tipo clave-valor. Además, la unidad básica de almacenamiento que tiene implementado son las “tablas de cadenas ordenadas (SSTables)” [17], estas se usan para poder mantener los datos en el disco duro del dispositivo que se utiliza, estas tablas son inmutables por lo que, una vez hechas, no se les puede hacer cambios. Es importante mencionar que Cassandra también permite exportar tablas dentro de la base de datos en archivos tipo CSV y ZIP, sin embargo, en el caso de la presente investigación esto no fue utilizado, se utilizó únicamente el almacenamiento en el disco duro.

V.CONCLUSIONES

Una vez finalizado el análisis de los dos softwares seleccionados, se evidencia que ambos cumplen con los objetivos planteados, se logran crear las bases de datos y hacer las consultas necesarias, sin embargo, se concluye que existen diferencias entre ellos.

En cuanto a la consistencia, Cassandra destaca por su alta disponibilidad y tolerancia a fallos mediante consistencia eventual, mientras que Hbase ofrece una consistencia fuerte en todas sus operaciones, lo que lo hace más adecuado cuando la precisión de los datos es esencial.

Sobre su desempeño en la inserción de datos, Cassandra resultó ser más eficiente, lo que la convierte en una mejor opción para sistemas que requieren actualizaciones rápidas y frecuentes, en cambio, mostró más latencias, probablemente por su enfoque en asegurar la consistencia.

Por otro lado, en la recuperación de datos, Hbase, gracias a su integración con HDFS(Hadoop Distributed File System), ofrece una mayor resiliencia en caso de fallos, lo que es ideal para entornos críticos donde la integridad de los datos es importante.

Asimismo, se abordaron aspectos relacionados con la facilidad de uso, Cassandra resultó ser más amigable para usuarios con conocimientos básicos en sistemas NoSQL, mientras que Hbase requiere un entorno más complejo y conocimientos avanzados, lo que puede representar una curva de aprendizaje más pronunciada.

Finalmente, la elección entre Cassandra y Hbase depende de las necesidades específicas del sistema. Si se busca velocidad y escalabilidad, Cassandra es la opción ideal. Si la prioridad es la precisión de datos y la recuperación ante fallos, Hbase es la mejor alternativa.

REFERENCIAS

[1] Cattaneo, M. F. P., Nocera, M. L., & Rottoli, G. D. (2014). Rendimiento de tecnologías NoSQL sobre cantidades masivas de datos. *Cuaderno Activa*, *6*(6), 11-17. https://biblat.unam.mx/es/revista/cuaderno-

activa/articulo/rendimiento-de-tecnologias-nosql-sobre-cantidades-masivas-de-datos

[2] J. Martínez, “Bases de datos NoSQL” Javi M. Solera, 2022. [Online]. Available: https://jmsolera.com/bases-de-datos-nosql/ [Accessed:Oct. 14, 2024]

[3] Apache Cassandra, “Cassandra Basics”, Apache Cassandra, 2024. [Online] Available: https://cassandra.apache.org/_/cassandra-basics.html [Accessed:Oct. 14, 2024]

[4] IBM, “¿Qué es HBase?” IBM, 2024. [Online] Available: https://www.ibm.com/mx-es/topics/hbase [Accessed:Oct. 14, 2024]

[5] A. Peña “Bases de datos Cassandra” Formadores it, 2019. [Online] Available: https://formadoresit.es/bases-de-datos-cassandra/ [Accessed:Oct. 14, 2024]

[6] A. Requena Meza, "¿Qué es Apache Cassandra?”, 2019. [Online]. Available: https://openwebinars.net/blog/que-es-apache-cassandra/ [Accessed:Oct. 14, 2024]

[7] Ksolves Team, "Consistency levels in Apache Cassandra: Guaranteeing reliability,", 2024. [Online]. Available: https://www.ksolves.com/blog/big-data/apache-cassandra/consistency-levels-in-apache-cassandra-guaranteeing-reliability [Accessed:Oct. 14, 2024]

[8] Wang, H., Li, J., Zhang, H., & Zhou, Y. (2014). Benchmarking Replication and Consistency Strategies in Cloud Serving Databases: HBase and Cassandra. En *Lecture notes in computer science* (pp. 71-82). Available: https://doi.org/10.1007/978-3-319-13021-7_6

[9] Apache Cassandra, “Cassandra documentation”, 2024. [Online]. Available: https://cassandra.apache.org/doc/3.11/cassandra/data_modeling/data_modeling_rdbms.html [Accessed:Oct. 15, 2024]

[10] Sreekzz, “Copia de seguridad y replicación para Apache HBase y Phoenix: Azure HDInsight,” *Microsoft.com* 2024. [Online] Available: https://learn.microsoft.com/es-es/azure/hdinsight/hbase/apache-hbase-backup-replication [Accessed:Oct. 15, 2024]

[11] E. Alshammari, G. Al-Naymat, and A. Hadi, “A New Technique for File Carving on Hadoop Ecosystem,” *IEEE Xplore*, 2017. [Online] Available: https://ieeexplore.ieee.org/abstract/document/8250267 [Accessed:Oct. 15, 2024]

[12] J. Carpenter and E. Hewitt, *Cassandra: The Definitive Guide*, 2nd ed., O'Reilly Media, 2020. [Online]. Available:https://cassandra.apache.org/doc/stable/cassandra/data_modeling/data_modeling_tools.html..[Accessed:Oct. 15, 2024]

[13]“Overview | Apache Phoenix,” *phoenix.apache.org*. https://phoenix.apache.org/[Accessed:Oct. 15, 2024]

[14] “User guide — HappyBase 1.2.0 documentation,” Readthedocs.io, 2024. [Online] Available: https://happybase.readthedocs.io/en/stable/user.html [Accessed:Oct. 15, 2024]

[15] S. Aggarwal, “Comenzando Con Cassandra: Usando CQL API y CQLSH”, envato tuts+, 2017. [Online] Available: https://code.tutsplus.com/es/getting-started-with-cassandra-using-cql-api-and-cqlsh--cms-28026a [Accessed:Oct. 15, 2024]

[16] Apache Cassandra, “Cassandra documentation”, Apache Cassandra, 2024. [Online] Available: https://cassandra.apache.org/doc/4.1/cassandra/cql/dml.html [Accessed:Oct. 15, 2024]

[17] “¿Cuál es la diferencia entre Cassandra y MongoDB?” Amazon Web Service, 2024. [Online] Available: https://aws.amazon.com/es/compare/the-difference-between-cassandra-and-mongodb/#:~:text=Cassandra%20almacena%20los%20datos%20como,utiliza%20en%20el%20almacenamiento%20real. [Accessed:Oct. 15, 2024]

ANEXOS

Anexo 1. Ejemplos de consultas HBase (HappyBase en WSL)

```
import happybase

# conexion al servidor thrift

connection = happybase.Connection('localhost')

connection.open()

#Quiero ver informacion de la persona con el id 1

table = connection.table('user_interactions')

user_id = '1'

for key, data in table.scan(filter=f"PrefixFilter('{user_id}_')"):

    print(key, data)

#Productos populares orden ascendente

connection = happybase.Connection('localhost')

table = connection.table('popular_products')

# Obtener productos electrónicos

electronics_products = []

for key, data in table.scan():

    if b'info:category' in data:  # Verifica si la columna 'info:category' existe

        category = data[b'details:category'].decode('utf-8')

        if category == 'Electronics':
```

```
        product_name = data.get(b'details:product_name', b'Unknown Product').decode('utf-8')

        product_id = data.get(b'details:category', b'0').decode('utf-8')

        total_interactions = int(data.get(b'details:total_interactions', b'0').decode('utf-8'))

        electronics_products.append({

            'product_name': product_name,

            'product_id': product_id,

            'total_interactions': total_interactions

        })

# ordenar los productos por total_interactions en orden descendente

sorted_products = sorted(electronics_products, key=lambda x: x['total_interactions'], reverse=True)

for product in sorted_products:

    print(f"Producto: {product['product_name']}, Categoria: {product['product_id']}, Interacciones totales: {product['total_interactions']}")

#Usuarios que buscaron libros

import happybase

# conexion al servidor thrift

connection = happybase.Connection('localhost')

connection.open()

table = connection.table('user_interactions')

for key, data in table.scan(filter="SingleColumnValueFilter('info', 'action', =, 'binary:search') AND SingleColumnValueFilter('info', 'category', =, 'binary:Books')"):

    print(key, data)
```

Anexo 2. Ejemplo de consultas Cassandra (CQL en Python y Java)

```
SELECT * FROM user_interactions WHERE user_id = 1; #Toma las interacciones del primer usuario, estas incluyen el momento en el que interactuó, el producto, la acción y la categoría del producto.


SELECT product_name, product_id, total_interactions

FROM popular_products

WHERE category = 'Electronics'

ORDER BY total_interactions DESC; #Da los productos más populares de la categoría de electrónicos.


SELECT * FROM user_viewed_products WHERE user_id = 1; #identifica el producto visualizado por el primer usuario.


SELECT co_purchased_product_name

FROM product_co_purchase

WHERE product_id = 1001

ORDER BY total_purchases DESC; #identifica el producto que compró este usuario junto con el principal que sería el 1001.


SELECT product_id,  purchase_time

FROM user_purchase_history

WHERE category = 'Electronics'

AND purchase_time > '2024-01-01'

ALLOW FILTERING; #identifica los productos comprados luego del 01/01/2024 de la categoría de electrónicos.


SELECT * FROM user_interactions

WHERE action = 'search' AND category = 'Books'

ALLOW FILTERING; #selecciona todas las columnas de la tabla interacciones del usuario, de los usuarios que buscaron cosas de la categoría libros.
```