# Comparación de Bases de Datos Orientadas a Documentos: CouchDB vs. OrientDB para el Almacenamiento de Datos No Estructurados

Daniel Alfaro, Brandon Arias, Ana Paula González, Emanuel Jiménez, José Pablo Mora, Laura Obando

*Escuela de Estadística, Universidad de Costa Rica, San José, Costa Rica*

daniel.alfarofigueroa@ucr.ac.cr
brandon.ariassandoval@ucr.ac.cr
ana.gonzalezestrada@ucr.ac.cr
emanuel.jimenezcubero@ucr.ac.cr
jose.mora20@ucr.ac.cr
laura.obandoesquivel@ucr.ac.cr

**Abstract- Ante la necesidad de almacenar, procesar y consultar datos de diversos tipos, el modelaje de datos actual exige alternativas a las bases de datos SQL, incluyendo las bases de datos NOSQL. Este artículo explora la adaptabilidad de las bases de datos orientadas a documentos como una solución para el almacenamiento de datos no estructurados con características y propiedades variables. Se demuestra la adaptabilidad de este tipo de bases de datos por medio de la construcción de una base de datos de una librería digital que almacena canciones, podcasts, películas, canales de Twitch y libros. El análisis se centra en las ventajas y diferencias de la implementación de esta base de datos usando CouchDB y OrientDB, dos softwares diseñados para base de datos orientadas a documentos. Se comparan aspectos como la consistencia, inserción de datos, recuperación de datos, facilidad de uso, herramientas de gestión, consultas y almacenamiento que ofrece cada software. En conclusión, el análisis encontró que tanto OrientDB como CouchDB son opciones viables para la implementación de bases de datos orientadas a documentos. Se recomienda OrientDB ante un escenario que requiera que los datos estén estructurados o semiestructurados y CouchDB ante escenarios donde los datos tengan características o propiedades muy variables.**

**Index terms: bases de datos orientadas a documentos, biblioteca digital, CouchDB, NOSQL, OrientDB.**

## I. INTRODUCCIÓN

Todos los pasos referenciados en este documento pueden consultarse en el repositorio: https://github.com/emajc/Investigacion-grupal-1/

La importancia de aprender a trabajar tecnologías de datos es de suma importancia para la era para la toma de decisiones actuales que pueden incidir en la mayor parte de la población. Dicho esto, dentro de estas categorías tecnológicas se destaca el manejo de datos por medio de Structured Query Language mejor conocido como SQL, definido por Microsoft como “un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos”[1], noción respaldada por la Universidad Europea , que menciona que, “por medio de SQL, se pueden trabajar los datos de múltiples maneras sencillas e intuitivas destacando comandos como la selección, relación, inserción y eliminación de datos.”[2]

Conociendo la importancia de SQL, se tiene la contraparte que corresponde a la versatilidad de los datos no estructurados conocidos como NoSQL. Estos datos tienen como característica principal que no se encuentran vinculados a tablas o relaciones, sumado a esto, “permite almacenar y consultar datos fuera de las estructuras tradicionales que se encuentran en las bases de datos relacionales” [3]. El trabajo con estas estructuras de datos siempre se va a encontrar ligado al tipo de datos y la necesidad de procesamientos que se requieran. Ante esto, el presente informe tiene como objetivo comparar por medio de dos softwares las similitudes y diferencias en la implementación de una base de datos NOSQL.

Específicamente, se busca poner a prueba las bases de datos orientadas a documentos, conocidas por su simplicidad, a no ser estructuradas y a su versatilidad en la diversidad de propiedades que puede almacenar documento por documento. En este tipo de bases de datos, cada documento equivale a una observación que correspondería a una fila.

## II. PLANTEAMIENTO DEL PROBLEMA

Para llevar a cabo el análisis de la conveniencia del uso de bases de datos orientadas a documentos, se planteó una problemática. Esta consiste en la necesidad de la implementación de una base de datos no estructurada que permita el almacenamiento de documentos con propiedades muy variables entre documentos para una biblioteca digital. Esta base de datos debía poder almacenar y diferenciar los documentos de acuerdo al tipo, que podía ser película, libro, canal de Twitch, canción o podcast. Se requería almacenar 2500 datos en total, 500 por cada tipo de documento. Más allá de poder diferenciar los tipos de documentos, no se requería relacionar los documentos por ninguna otra característica, por lo que hacer uso de bases de datos orientadas a documentos demostró ser conveniente.

## III. CARACTERÍSTICAS DE LA SOLUCIÓN IMPLEMENTADA

Para brindarle solución al problema, se procedió a construir la misma base de datos en dos softwares diferentes: CouchDB y OrientDB. CouchDB es un software diseñado específicamente para el manejo de bases de datos orientadas a documentos, mientras OrientDB está diseñado para manejar tanto bases de datos orientadas a documentos como bases de datos orientadas a graphos. Dados los objetivos de este análisis, únicamente se trabajó con las características propias de las bases de datos orientadas a documentos en OrientDB. Para el caso de OrientDB, se trabajó directamente en OrientDB Studio, mientras que para CouchDB se trabajó con la interfaz en línea Fauxton.

Para la implementación de la base de datos en CouchDB, se codificaron 2500 documentos (uno por cada tipo de documento) con diferentes propiedades. En general, se decidió uniformizar las propiedades manteniendo en todos los documentos un identificador, el nombre, el nombre del autor y el link de acceso al archivo. Para diferenciar el tipo de documento, se añadió la propiedad o característica "tipo", que indicaba cuál era el tipo de documento. Esto debido a que, a diferencia de la mayoría de softwares de manejo de bases de datos orientadas a documentos, CouchDB no brinda la opción de establecer clases para la separación de documentos. Como tal, el ingreso de documentos a la base de datos corresponde a todo el proceso y estructura de la propia base, pues no se detallan clases, tablas o ninguna estructura para su almacenamiento.

Por otro lado, en OrientDB se codificaron cinco clases diferentes para cada tipo de documento. Estas clases se especificaron como subclases de dos superclases: Multimedia y Escrito. Esto permitió añadir un nivel adicional de estructura a los documentos almacenados, característica que CouchDB no ofrece. Al almacenar documentos en clases, estos heredan las características especificadas en ellas. Por lo tanto, para la inserción de datos las propiedades o atributos de cada documento debían ser las mismas que las de todos los documentos pertenecientes a la misma clase. Aunque OrientDB Studio permite especificar propiedades o atributos diferentes al insertar los documentos, este proceso demostró no ser factible en el manejo de muchos datos, pues el ingreso tendría que ser manual en OrientDB Studio. Por lo tanto, se mantuvieron uniformes las características de todos los documentos pertenecientes a cada clase. Posteriormente, se ingresaron 2500 documentos (500 por cada clase) a la base de datos usando OrientDB Studio y el método de ingreso de datos de SQL (Insert Into).

Tanto para CouchDB como para OrientDB se logró demostrar que la implementación de una base de datos orientada a documentos era una solución factible para el problema planteado, debido a que se logró construir una base de datos en cada programa que almacenara los datos con características diferentes y de manera no estructurada. Bajo esta noción, CouchDB mostró ser más versátil y con un método de inserción más flexible al ingreso de datos no uniformes. Por otro lado, OrientDB mostró ser más adecuado para datos semi-estructurados, pero no tuvo la misma facilidad de implementación de datos no uniformes como lo tuvo CouchDB.

## IV. COMPARACIÓN COUCHDB Y ORIENTDB.

Como segundo objetivo de este análisis, se planteó la comparación de CouchDB y OrientDB de acuerdo con varios criterios. Uno por uno, se detallan las comparaciones por cada criterio estudiado.

### *A. Consistencia*

CouchDB es un software que puede procesar múltiples réplicas sobre los datos generados, aunque no en todas estas rondas de réplicas los datos se encuentran sincronizados al ser un software NoSQL. Los datos cambian o se actualizan por medio de réplicas y este sistema puede que dure un tiempo prolongado en alguno de los casos. Esto quiere decir que, aunque esté el código de interés completo, no significa que las réplicas de este código se vayan a generar al mismo ritmo que se lleva la codificación, pero en algún momento llegan a converger de manera consistente [4]. Las réplicas a las cuales nos referimos se refieren a la cantidad de copias de datos que es capaz de almacenar el sistema para contar

con disponibilidad de estos; estas copias tienen la característica de poderse ubicar en distintos servidores (por si alguno de los servidores en algún momento falla).

A pesar de su enfoque en la consistencia eventual, CouchDB tiene la capacidad de garantizar la consistencia en el nivel de los documentos a través de su soporte para las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad)[5]. Esto significa que las operaciones en los documentos son atómicas, es decir, se realizan como una única unidad o no se realizan en absoluto. Además, se asegura de que los documentos se mantengan en un estado consistente en todo momento, incluso cuando se realizan múltiples operaciones simultáneas. La característica de aislamiento garantiza que las operaciones en un documento no afecten las operaciones en otros documentos de manera inesperada. Por último, CouchDB es duradero, lo que significa que una vez que se confirma una operación, los cambios en los documentos se almacenan de forma segura y persistente, incluso en caso de falla del sistema. Esta combinación de consistencia eventual y las propiedades ACID proporciona un equilibrio sólido entre la flexibilidad de manejar grandes volúmenes de datos y la seguridad de que los datos son confiables y coherentes en todo momento.

Ahora bien; respecto a Orient DB tenemos, que su consistencia ha ido cambiando a lo largo de sus actualizaciones, lo cual significa que su consistencia actual no es la misma a la consistencia de versiones anteriores. Antes de la versión 2.17, OrientDB trabajaba su consistencia mediante transacciones, lo cual ocasiona limitaciones en la velocidad, optimización y escalabilidad de las bases de datos, es decir cumplía al pie de la letra los procedimientos ACID y su uso estaba más orientado a las bases de datos SQL, no obstante, se debe aclarar que independientemente de las versiones que hayan, siempre ha utilizado a lo largo de su evolución un lenguaje SQL.

Luego OrientDB, a partir de la versión 2.17 implementa las operaciones no transaccionales, es decir, su consistencia cambió, debido a que su forma de trabajar no era ACID, es decir, las operaciones realizadas en una base de datos no se iban a efectuar como una unidad atómica (algunas operaciones se completarán otras no), la base de datos puede pasar de base consistente a una inconsistente (por las diferentes operaciones no transaccionales), el actuar de un usuario puede afectar en el procedimiento de otro usuario, y los cambios no son constantes en la bases de datos sino que son variables. No obstante, esta nueva implementación ocasiona mejoras en las bases de datos NoSQL, dotando del software de escalabilidad y flexibilidad, además esto no significa que se haya perdido la consistencia en estas nuevas actualizaciones, sino que se implementaron nuevos mecanismos como las mismas transacciones pero de una forma automatizada cuando no hay operaciones no transaccionales (configuración global sql.graphConsistencyMode en notx_sync_repair o notx_async_repair) o en el uso de reparaciones a las bases de datos de forma asincrónica (haciendo uso del modo de consistencia notx_sync_repair) o sincrónica (usando el modo de consistencia notx_async_repair)[6].

### *B. Inserción de datos*

CouchDB puede agregar varios servidores para almacenar la información codificada con el fin de aumentar los rendimientos; este rendimiento muchas veces depende de la manera en la que se trabaje la información, porque entre más sencillo se compila el código, más sencillo será de procesar, pero si se torna complejo significa que amerita más rendimiento[7].

Tiene dos formas básicas de inserción de datos, una es por la interfaz gráfica de project Fauxton, que es muy útil y sencilla, siempre y cuando sea una base de datos pequeña, debido a que se tendrá que crear un documento tipo Jason por cada línea de la base. La segunda opción es por medio de la terminal o línea de comando, es un poco más complejo debido a la utilización de código, es decir, es menos intuitivo y amigable con el usuario, sin embargo, para insertar grandes cantidades de datos es la opción más recomendada ya que tiene la instrucción de insertar los datos de forma simultánea, donde solo se necesita un archivo tipo json que describa la base de datos que se desea insertar y una cuenta en Couch DB con una base de datos creada previamente y lista para recibir y almacenar la información.

En comparación a OrientDB, este permite la incorporación de múltiples computadoras para el trabajo conjunto y administrativo en una base de datos, lo cual permite características fundamentales en las bases de datos tipo NoSQL como lo son la escalabilidad y la flexibilidad.

Asimismo, OrientDB ofrece recomendaciones de rendimiento de hardware, debido a que el tipo de hardware (disco sólido o disco duro) con el cual se trabaja puede influir en el desempeño de la inserción de datos, pero también ha de tomarse en cuenta el tipo de código trabajo, por tanto un código complejo y de larga extensión no trabaja de la misma manera en un sistema operativo apto para lo que se le pide en comparación a un sistema operativo no apto ante tal tipo de códigos (y se puede intuir que un código más sencillo es de mejor desempeño en un sistema apto o no apto). Además OrientDB ofrece optimización su memoria virtual ante el manejo de grandes datos, los cuales podrían causar problemas por el problema antes mencionado. Además, "OrientDB está programado en JAVA, por lo tanto en sus nuevas actualizaciones hace uso de una extensión llamada "JMX" la cual permite ver la información del rendimiento de un producto (tal es el caso de OrientDB)"[8]. En términos prácticos, la inserción de datos en OrientDB se realizó por medio de OrientDB Studio, que tiene una sección "Search" para hacer consultas con código SQL. Previo a la inserción de datos, OrientDB, a diferencia de CouchDB, sí exige la construcción de estructuras para el almacenamiento de los datos, conocidas como clases. OrientDB permite establecer estas clases directamente en Studio o a través de código. Posteriormente, la inserción de datos se realizó con la función Insert de SQL, siendo la clase lo equivalente a una tabla en una base de datos SQL. La inserción de datos debe hacerse clase por clase, por lo que es necesario que el código SQL esté organizado por clase.

### *C. Recuperación de datos*

Sobre CouchDB, se destaca por su capacidad de manejar grandes cantidades de datos, en tiempos de respuesta rápidos y eficientes, claramente dependiendo también del equipo con que se esté trabajando. La característica más preciada de Couch DB en la gestión de datos es la replicación bidireccional con la que cuenta, es

decir: si se tiene una base de datos A y B, la replicación bidireccional permite que los cambios en la base A se reflejen automáticamente en la base B[9].

Por otra parte, en este sentido de recuperación de datos, también se vuelve preciado que las bases de datos se pueden abrir desde distintos equipos manteniendo una conexión con un servidor, de esta forma se puede agilizar aún más la recuperación de datos ya que se mantienen en un mismo lugar, se trabaja en tiempo real y sin necesidad de hacer descargas para luego compartirlas con distintos usuarios.

Por el lado de OrientDB, en cuanto a la recuperación de datos, se sabe que hace uso de etiquetas las cuales permiten permiten trabajar de forma optimizada en la creación y consulta de datos noSQL, hay diferentes tipos de indices que proporciona OrientDB para la recuperación. Dos de esos tipos de índices son el indice de "clave" y el indice "ID", los cuales son utilizados para la creación de bases de datos no relacionales. Los indices de clave servirían para indicar cuales serían las respectivas colecciones y los indices "ID" sirven para identificar los documentos dentro de una colección [10]. Por otra parte, OrientDB hace uso de etiquetas de "texto" las cuales sirven para la consulta de los usuarios a las bases de datos.

Por otra parte, OrientDB ofrece una arquitectura distribuida, esto significa que varios usuarios y servidores pueden leer y escribir datos. Asimismo, hace uso de las "transacciones distribuidas" esto significa que si un servidor realiza un cambio en la base de datos, esta se refleja en los demás servidores, no obstante, esto no significa que sea de forma inmediata, porque dependerá de la configuración de consistencia y del sistema operativo con el que se trabaje.

Un factor importante en el desempeño en la recuperación de datos, es que esta dependerá de la cantidad de datos y complejidad con la cual se haya sido creada la estructura del código para la base en unión a la capacidad del hardware y los sistemas operativos con los que se trabaje, y como se mencionó anteriormente eso se evidenciará en el factor velocidad y optimización en la creación y consulta de las bases de datos NoSQL.

En términos prácticos, la recuperación de datos en CouchDB demostró ser igual de compleja que en OrientDB. Sin embargo, CouchDB no ofrece una manera de exportar la base de datos, por lo que se recomienda usar OrientDB si el manejo de la base de datos se hará de manera local entre varios usuarios. En ambos casos, la manera más sencilla de importar la base de datos fue por medio de la consola. Aunque el proceso no es complejo, sí exige más pasos que usando una base de datos SQL.

*D. Facilidad de uso*

Sobre CouchDB, como se ha venido abordando de forma implícita en el texto anterior, se refleja una facilidad de uso preciada, gracias a su modelo de datos basado en documentos tipo Json, esto simplifica la comprensión, el manejo de la información y su excelente estructura dotada de un buen orden [11]. Además de eso cuenta, con una interfaz gráfica amigable "Project Fauxton", que permite a los usuarios administrar los datos, documentos y vistas de manera visual.

La replicación de datos bidireccional anteriormente abordada, facilita el uso de las bases de datos en una empresa u organización, que requiere contar con una base de datos actualizada en tiempo real. Así mismo se aprovecha una característica destacada en Couch DB, la flexibilidad de estructura, al no tener un esquema fijo, los usuarios pueden agregar horizontalmente información de importancia, sin la necesidad de reestructurar la base y sin sacrificar la consistencia.

A lo que refiere a OrientDB, este sí conlleva una mayor complejidad a la hora de crear bases de datos no relacionales, porque estos se asemejan al uso del lenguaje SQL[12], lo cual se evidencia en el código del anexo 1.

Con los ejemplos abordados en el anexo 1, se asume que para la utilización de OrientDB en la elaboración de una base de datos NoSQL se requiere de conocimientos en las herramientas de lenguaje SQL con sus respectivas variaciones.

A pesar de lo comentado, OrientDB Studio permite hacer todos los pasos de la creación de clases y la inserción de datos de manera manual directamente en la interfaz gráfica, por lo que prácticamente no se requiere ningún conocimiento de programación. Por lo tanto, se concluye que con OrientDB Studio, este software es tan fácil como CouchDB.

*E. Herramientas de gestión*

Los utilizados en este trabajo fueron Fauxton que es la interfaz en línea a la que se puede acceder de una manera super sencilla, esto "viene incluído" en la instalación del programa de CouchDB. Se puede trabajar también desde el administrador y mediante el monitoreo de clústers.

Como parte de este trabajo, se utilizaron varias herramientas clave para administrar y operar de manera eficiente la base de datos CouchDB. Utilizamos principalmente Fauxton, una interfaz en línea estándar y de fácil acceso que se incluye en una instalación estándar de CouchDB. Fauxton proporciona una plataforma gráfica fácil de usar que simplifica la gestión de documentos, la creación de vistas y la replicación de datos de seguimiento. Su fácil acceso la convierte en una herramienta valiosa para quienes desean una experiencia de usuario más intuitiva y no están familiarizados con la línea de comandos.

Además, el desarrollo de este trabajo examinó dos componentes importantes para la gestión de CouchDB. El Administrador proporciona una interfaz web dedicada para configurar y administrar aspectos importantes del servidor, como la seguridad y la configuración del clúster. Mientras tanto, el "monitoreo de clústeres" se ha vuelto esencial para monitorear y mantener el rendimiento y la sincronización en entornos de alta disponibilidad. Junto con Fauxton, estas herramientas permiten un enfoque integral para utilizar CouchDB de manera efectiva en diversos proyectos, brindando a los administradores y desarrolladores las herramientas que necesitan para una gestión eficiente de la base de datos [13].

Por otra parte, "OrientDB provee una consola como herramienta, la cual fue creada con Java y tiene la capacidad de conectarse a una interfaz web llamada OrientDB Studio"[14], la cual se puede trabajar con consultas e inserciones de información en las bases de datos no relacionales. Ha de mencionarse que la interfaz web brinda de una arquitectura simple y minimalista la cual permite ver

del producto o del sujeto a qué clase (colecciones), su id, y propiedades (columnas de información) que conforman cada documento.

Asimismo, OrientDB, ofrece herramientas de respaldo y seguridad ante situaciones de pérdida de datos o confidencialidad de datos que no deben ser tratados por cualquier persona. Dentro de estas herramientas se puede mencionar el comando "BACKUP DATABASE" la cual realiza un copia de seguridad de la base de datos y la comprime en un archivo .ZIP, la cual se podrá guardar y volver a utilizar con el comando RESTORE DATABASE ante cualquier accidente. [15]

También como se mencionó, OrientDB ofrece herramientas de seguridad, una de esas herramientas son los 3 tipos de roles con los cuales se puede interactuar en la interfaz web, estas son admin, reader y writer. El admin tiene total acceso a las bases de datos y a sus funciones, el reader como lo dice su nombre solo puede leer más no hacer ningún tipo de modificación, y el writer, que es similar al reader pero tiene la capacidad crear, actualizar y eliminar registros. Otra herramienta de seguridad son los 2 modos de trabajo que son "permitir todo excepto" que es como un tipo de superusuario pero con sus respectivas limitaciones y el otro "denegar todo excepto" que se utiliza en los usuarios comunes que utilizan las consultas de las bases de datos. Y para finalizar el apartado de seguridad, OrientDB ofrece desactivar o activar las participaciones de los usuarios en la interfaz web, lo cual es beneficioso ante situaciones de protección de las bases de datos.

Con las herramientas mencionadas anteriormente, permite que los grupos de trabajos puedan entender de forma rápida y eficaz el manejo de las bases de datos no relacionales implementando roles y sistemas de seguridad.

*F. Consultas*

Para generar consultas hay unos "pasos" que deben ser cumplidos en la codificación, como la creación de un diseño de vista (las vistas son la manera de cómo se filtran y se ordenan los documentos), la generación de los documentos necesarios; después de esto, ya se pueden realizar consultas a las vistas generadas en el link http://localhost:5984/nombredelabasededatos/_design/nombredeldocumento/_view/nombredevista donde se devuelven los resultados en formato JSON. Otra información importante con esto de las vistas es que cada una de las consultas generadas se almacena para poder acelerar una posible consulta futura idéntica.

Es importante destacar que CouchDB almacena en caché los resultados de cada consulta generada, lo que hace que futuras consultas idénticas sean más rápidas y mejora el rendimiento general del sistema. Este enfoque de visualización y consulta en CouchDB proporciona una metodología sólida para capturar y analizar datos de manera eficiente y efectiva en una variedad de aplicaciones y proyectos[16].

Además, el uso de vistas en CouchDB proporciona una mayor flexibilidad a la hora de seleccionar y filtrar datos, lo que resulta especialmente beneficioso para aplicaciones y sistemas que gestionan grandes cantidades de información[16]. Las vistas se pueden personalizar según las necesidades de un proyecto en particular, lo que brinda a los desarrolladores un control preciso sobre cómo se accede y muestra los datos. Esto mejora el rendimiento y la eficiencia al optimizar las consultas para obtener resultados precisos y relevantes[16]. Esto es importante para tomar decisiones informadas en muchas situaciones.

Para consultas más específicas, Project Fauxton cuenta con una sección llamada *Mango Query*. Mango es el sistema de consultas que utiliza CouchDB. En el anexo 2, se adjuntan ejemplos de consultas a la base de datos construida. Debido a la naturaleza de los datos almacenados en CouchDB, las consultas Mango no son muy complejas.

Por otro lado, en OrientDB, después de haber creado las bases de datos y las inserciones de los datos por medio del comando brindado a la hora de hora de instalar este producto, permite trabajar con una interfaz web a la cual se puede acceder por medio de un "user" y un "password" establecido con anterioridad en la creación de la base en el comando en el siguiente link: http://localhost:2480/studio/index.html#/

Se sabe que OrientDB hace uso del lenguaje SQL, por lo cual las consultas harán uso de este al igual que la creación e inserción de datos, por lo cual se hace uso de los SELECT * FROM, WHERE, GROUP BY, COUNT, ORDER BY [12] entre otras instrucciones más que son gran utilidad, siempre y cuando se tenga un conocimiento de lenguajes SQL. Ejemplos adicionales se incluyen en el anexo 3 de este artículo.

*G. Almacenamiento*

El almacenamiento principal con el que desarrollamos el código fue con el formato JSON (clave-valor) que simplifica tanto la codificación como la interpretación de los mismos datos[17]. También se podría trabajar por bases de datos en conjuntos de documentos relacionados, por identificadores que se hacen mediante cadenas de texto exclusivas para cada documento; y por revisiones, que consisten en que, cada vez que se actualiza el documento, este crea una actualización del mismo de forma automática sin perder la capacidad de recuperar versiones previas del documento.

Por su lado, OrientDB actualmente utiliza un almacenamiento Local Paginado, conocido como "plocal", este tipo de almacenamiento usa componentes del disco, y como ha de pensarse tales componentes son la caché del disco [18]. También dentro este almacenamiento que hace OrientDB se debe mencionar que toma protagonismo los clúster, las cuales se dividen en secciones en el disco representando una unidad atómica utilizada en el clúster. También cada clúster tiene punteros (representaciones físicas en el disco a la hora de un registro) que tienen la funcionalidad, es decir por cada registro que se haga en la base de datos, ésta tendrá un efecto en la memoria del disco. Este almacenamiento Local Paginado hace uso de índices de página y claves, los cuales permiten organizar y localizar los diferentes punteros (registros) dentro del disco.

Por otro lado, como ya se ha mencionado, para guardar una base de datos localmente, CouchDB no cuenta con una opción que permita exportar la base. OrientDB sí cuenta con una opción para exportar la base completa, lo cual hace más sencillo compartirla localmente.

## IV. CONCLUSIONES

Una vez realizado el análisis pertinente mediante los software NoSQL seleccionados correspondientes a CouchDB y OrientDB encontramos que, los 2 productos (CouchDB y OrientDB) que se utilizaron ante el problema planteado de poder gestionar el contenido multimedia en un biblioteca digital, han cumplido con éxito los objetivos propuestos, no obstante, el manejo y creación de las bases de datos no relacionales entre estos 2 productos comparten similitudes y diferencias. Es de suma importancia observar, comparar y concluir cómo las nuevas tecnologías de bases de datos no relacionales evolucionan a lo largo del tiempo y de las demandas de usuarios o del mercado. Y dentro de las grandes evoluciones que han ocurrido a lo largo del tiempo y de las demandas del mercado, está el uso de bases NoSQL, a lo cual CouchDB y OrientDB demostraron ser herramientas competentes con sus respectivas diferencias y similitudes para llevar a cabo la tarea de agregar datos a la base no relacional con diferentes tipos de variables y características.

Finalmente, se concluyó que ante el escenario planteado tanto OrientDB como CouchDB son buenas alternativas para la implementación de una base de datos orientada a documentos, prefiriendo OrientDB ante un escenario que requiera que los datos estén estructurados o semiestructurados y CouchDB ante escenarios donde los datos tengan características o propiedades muy variables.

## REFERENCIAS


[1]Access SQL, “conceptos básicos, vocabulario y sintaxis,” Soporte técnico de Microsoft, 2023. [Online]. Accedido en Oct, 9, 20230 Available:https://support.microsoft.com/es-es/office/access-sql-conceptos-b%C3%A1sicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671.

[2]Universidad Europea, “¿Qué es SQL y para qué sirve?” Blog UE - Universidad Europea,” Universidad Europea, 2022. [Online]. Accedido en: Oct, 10, 2023. Available: https://universidadeuropea.com/blog/lenguaje-programacion-sql/.

[3]IBM, “¿Qué son las bases de datos NoSQL?” IBM, 2023. [Online]. Accedido en: Oct, 11, 2023. Available: https://www.ibm.com/es-es/topics/nosql-databases.

[4] CouchDB, “Incremental replication” Apache CouchDB 3.3.0 Documentation, 2023. [Online]. Accedido en: Oct, 17, 2023 Available: 1.3. Eventual Consistency — Apache CouchDB® 3.3 Documentation.

[5] CouchDB, “ACID Properties” Apache CouchDB 3.3.0 Documentation, 2023. [Online]. Accedido en: Oct, 17, 2023. Available: 1.1. Technical Overview — Apache CouchDB® 3.3 Documentation.

[6]OrientDB, “Graph Consistency,” OrientDB Manual, 2021. [Online]. Accedido en: Oct, 13, 2023. Available: https://orientdb.com/docs/3.1.x/java/Graph-Consistency.html.

[7] CouchDB, “CouchDB Replication Protocol ” Apache CouchDB 3.3.0 Documentation, 2023. [Online]. Accedido en: Oct, 17, 2023 Available: 2.4. CouchDB Replication Protocol — Apache CouchDB® 3.3 Documentation.

[8]OrientDB, “Performance Tuning · OrientDB Manual,” OrientDB, 2022. [Online]. Accedido en: Oct, 13, 2023. Available: https://orientdb.com/docs/2.2.x/Performance-Tuning.html.

[9] C. Anderson; J. Lehnardt y N. Slater, CouchDB The Definitive Guide, Editor / O´ Reilly, [Online]. Acceso en: Oct, 17, 2023. Available: CouchDB: La guía definitiva.

[10]OrientDB, “Indexes,” OrientDB Manual, 2016. [Online]. Accedido en: Oct, 14, 2023 Available: https://orientdb.com/docs/2.2.x/Indexes.html.

[11] CouchDB, “Document Stoge” Apache CouchDB 3.3.0 Documentation, 2023. [Online]. Accedido en: Oct, 17, 2023 Available: 1.1. Technical Overview — Apache CouchDB® 3.3 Documentation.

[12]OrientDB, “OrientJS Query,” OrientDB Manual, 2016. [Online]. Accedido en: Oct, 15, 2023. Available: https://orientdwb.com/docs/2.2.x/OrientJS-Query.html

[13] Alexander Calderon, “Introduccion a Apache CouchDB,” YouTube. Aug. 01, 2020. [Online]. Available: https://www.youtube.com/watch?v=4SXQJeVKaIQ

[14]OrientDB, “Console Tool,” OrientDB Manual, 2021. [Online]. Accedido en: Oct, 15, 2023. Available: https://orientdb.org/docs/3.1.x/console/.

[15]OrientDB, “Backup and Restore,” OrientDB Manual, 2016. [Online]. Accedido en: Oct, 16, 2023. Available: https://orientdb.com/docs/2.2.x/Backup-and-Restore.html.

[16]Oscarfmdc, “CouchDB: Una introducción sencilla,” Aprender BIG DATA, May 2023, [Online]. Available: https://aprenderbigdata.com/couchdb/

[17]LT DATA CHANNEL, “🔎 [COUCHDB] ➡️ ¿QUÉ ES CouchDB? ⭐,” YouTube. Jul. 01, 2022. [Online]. Available: https://www.youtube.com/watch?v=ryqNK9Y1YRM

[18]OrientDB, “Paginated Local Storage,” OrientDB Manual, 2021. [Online]. Accedido en Oct, 17, 2023. Available: https://orientdb.com/docs/last/internals/Paginated-Local-Storage.html.


## ANEXOS

*Anexo 1: Ejemplo creación base de datos en OrientDB*

create class “nombre colección” extends V abstract #crea colección.

create class “documento” extends “clase” “#crea documentos de cada colección .

create property documento.id integer # crea id del documento con su respectivo tipo de dato.

alter property documento.id mandatory true #cada nueva fila de datos tendrá un id obligatorio.

create property documento.nombre string “crea columna de información del documento con su respectivo tipo de dato.

insert into Libros (id, nombre, autor, enlace, enfoque) values (),(),(),(),(),(),(),(),()

*Anexo 2: Ejemplo consultas CouchDB (MangoQuery)*

##Métodos de consulta

#Con Query Mango

#Con el cmd

#Consultas con el cmd

#Para ver todos los documentos de una base de datos

curl http://admin:011235@127.0.0.1:5984/practica1/_all_docs

"Paso correcto"

#Para hacer consultas sobre una clave en especifico.

curl "http://admin:011235@localhost:5984/practica1/_all_docs?include_docs=true&key=\"57c7874763aa9fd72cbc0bbd329dbed4\""

##Esta es otra forma de generar todos los documentos de la base de datos

curl http://admin:011235@localhost:5984/practica1/_all_docs?include_docs=true

##Para hacer consultas sobre una ID en especifico.

curl "http://admin:011235@localhost:5984/practica1/_all_docs?include_docs=true&id=\"57c7874763aa9fd72cbc0bbd324f0f84\""

##Me voy a Project F, creo una nueva vista en design documents, le pongo como nombre Consultas e index name Consulta0

##Esta consulta en consola refleja el cambio que se le ha hecho a todos los datos.

curl http://admin:011235@localhost:5984/practica1/_design/Consultas/_view/Consulta0

##Despues de haber creado una consulta1 en PF "(doc.documento_tipo, doc.género)"

curl http://admin:011235@localhost:5984/practica1/_design/Consultas/_view/Consulta1

##Despues de haber creado una consulta2 en PF "(doc.género,doc.cantidad_visualizaciones)"

curl http://admin:011235@localhost:5984/practica1/_design/Consultas/_view/Consulta2

*Anexo 3: Ejemplo consultas OrientDB*

SELECT * FROM libros #toma la colección Libros
SELECT * FROM peliculas WHERE director = 'Michael Mann' #toma el documento de la colección Peliculas donde el director se Michael Mann

SELECT enfoque, COUNT(*) AS cantidad
FROM libros
GROUP BY enfoque
# selecciona la información enfoque de Libros y usa el COUNT para ver cuantos libros hay según el enfoque en la columna cantidad.

SELECT nombre, duracion
FROM peliculas
WHERE duracion IS NOT NULL
ORDER BY duracion DESC

#selecciona el nombre y duración del documento películas, dando la indicación de no usar valores nulos en la información de duración, y después ordena esta consulta de forma descendente

SELECT * FROM Podcasts LIMIT 6 OFFSET 3 #selecciona del documento Podcasts los primeros 6 registros después de pasar por alto los primeros 3 registros.

UPDATE canciones
SET autor = 'desconocido'
WHERE autor = 'Don Omar'
OR enlace = 'https://open.spotify.com/intl-es/track/6KgBpzTuTRPebChN0VTyzV';

#Coge el documento canciones, y cambia la información igual a “desconocido” cuando el autor se llama Don Omarl o cuando el enlace sea “https://open.spotify.com/intl-es/track/6KgBpzTuTRPebChN0VTyzV”

DELETE VERTEX
FROM multimedia
WHERE genero = 'Entretenimiento'
or canal = 'Fabian';
# Borra los registros del documento multimedia donde genero es igual a entretenimiento y el canal sea llamado Fabian