- Instalación de MongoDB y librería PyMongo.

```
bash

pip install pymongo
```

- Conexión a MongoDB.

```
python

from pymongo import MongoClient

# Conectar al servidor MongoDB

cliente = MongoClient('mongodb://localhost:27017')

# Seleccionar base de datos y colección

db = cliente['mi_base_datos']

coleccion = db['usuarios']
```

- Insertar un documento.

```
python

documento = {

    "id": "usuario_123",

    "nombre": "Juan",

    "edad": 30,

    "email": "juan@mail.com",

    "pedidos": [

        {"id": "pedido_1", "total": 100},

        {"id": "pedido_2", "total": 200}

    ]

}

coleccion.insert_one(documento)
```

- Leer documentos.

```
python

# Consultar todos los documentos

for doc in coleccion.find():

    print(doc)

# Consultar por una clave específica

resultado = coleccion.find_one({"id": "usuario_123"})

print(resultado)
```

- Actualizar documentos.

```
python

# Actualizar el correo electrónico de un usuario

coleccion.update_one(

    {"id": "usuario_123"},

    {"$set": {"email": "nuevo_correo@mail.com"}}

)
```

- Eliminar documentos.

```
python

# Eliminar un documento

coleccion.delete_one({"id": "usuario_123"})
```

- Comparación entre bases de datos documentales y clave-valor:

Aspecto	Documental	Clave-Valor
Modelo	Documentos JSON	Pares clave-valor
Estructura	Jerárquica	Simple
Consultas	Complejas y detalladas	Directas
Casos de uso	Aplicaciones dinámicas	Cachés y sistemas simples
Relaciones	Moderadas	Limitadas

Tabla 5. Comparación entre bases de datos documentales y clave-valor. Fuente: elaboración propia.

Bases de datos columnar

Las bases de datos columnares son una solución especializada para el almacenamiento y procesamiento de grandes volúmenes de datos, diseñadas para optimizar consultas analíticas y cargas de trabajo intensivas en lectura. Si estás empezando, este modelo te permitirá comprender cómo se gestionan datos masivos en aplicaciones modernas como análisis de datos, *big data* y sistemas de inteligencia empresarial.

- ¿Qué es una base de datos columnar?:

Una base de datos columnar organiza y almacena los datos por columnas en lugar de filas. Este enfoque permite acceder rápidamente a valores específicos de una columna, lo que las hace ideales para consultas analíticas donde solo se necesitan ciertos campos.

- Definición: una base de datos columnar almacena los datos de manera que los valores de cada columna se agrupan y se almacenan juntos. Este modelo reduce el uso de I/O al acceder solo a las columnas necesarias para una consulta.

- Ejemplo de estructura tradicional (fila):

ID	Nombre	Edad	País
1	Juan	30	España
2	Ana	25	México

Tabla 6. Estructura tradicional. Fuente: elaboración propia.

Tema 2. Almacenamiento escalable

- Ejemplo de almacenamiento columnar:
  - Columna ID: [1, 2].
  - Columna nombre: [Juan, Ana].
  - Columna edad: [30, 25].
  - Columna país: [España, México].
- Características principales de las bases de datos columnares:

1. Almacenamiento por columnas:

- Los datos de cada columna se almacenan en bloques contiguos.

2. Compresión eficiente:

- Los valores de cada columna suelen ser homogéneos, lo que facilita la compresión y reduce el tamaño del almacenamiento.

3. Consultas optimizadas:

- Al acceder solo a las columnas necesarias, se reduce el tiempo de lectura y se mejora el rendimiento de las consultas.

4. Escalabilidad horizontal:

- Diseñadas para ser distribuidas y manejar grandes volúmenes de datos.

5. Optimización para OLAP *(online analytical processing)*:

- Especializadas en cargas de trabajo analíticas, no transaccionales.

- Arquitectura de una base de datos columnar:

1. Almacenamiento segmentado:

- Cada columna se almacena como un archivo o segmento independiente.

2. Indexación avanzada:

- Índices especializados, como índices bitmap, mejoran el acceso a los datos.

3. Compresión por columna:

- Utilizan algoritmos como *run-length encoding* (RLE) o Delta Encoding para optimizar el almacenamiento.

4. Procesamiento distribuido:

- Las bases columnares están diseñadas para funcionar en clústeres de nodos, distribuyendo las consultas entre múltiples servidores.

5. Motores de consultas optimizadas:

- Soportan lenguajes como SQL adaptados a operaciones masivas.

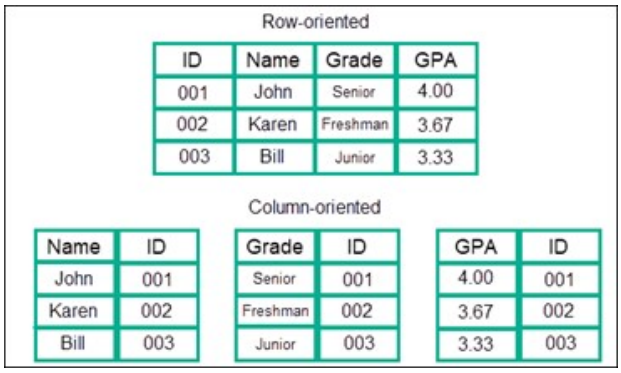

Figura 10. Imagen representativa almacenamiento datos en base de datos columnar. Fuente: Koech, 2023.

- Casos de uso de las bases de datos columnares:

1. Análisis de grandes volúmenes de datos:

- Exploración y procesamiento de datos históricos.

2. Inteligencia de negocios (BI):

- Alimentación de paneles de control y generación de reportes rápidos.

3. *Big data* y ciencia de datos:

- Preparación de datos para modelos predictivos y análisis avanzado.

4. Sistemas de recomendación:

- Procesamiento de grandes catálogos para personalizar experiencias de usuario.

5. Procesamiento de *logs:*

- Análisis de datos de registros de aplicaciones o sistemas.

- Ventajas de las bases de datos columnares:

1. Rendimiento en consultas:

- Consultas más rápidas al acceder únicamente a las columnas relevantes.

2. Ahorro de espacio:

- Los datos comprimidos ocupan menos espacio en disco.

3. Optimización para lectura:

- Diseñadas específicamente para operaciones de lectura intensiva.

4. Escalabilidad:

- Capaces de manejar *petabytes* de datos distribuidos.

5. Compatibilidad con herramientas de análisis:

- Las herramientas de BI más populares del mercado (Power BI, Tableau, Looker, Qlik View) tienen compatibilidad completa con el almacenamiento de bases de datos columnares.

- Limitaciones de las bases de datos columnares:

1. No aptas para transacciones:

- No son ideales para aplicaciones OLTP *(online transactional processing),* debido a sus tiempos de escritura.

2. Complejidad en actualizaciones:

- Las actualizaciones de datos son más costosas en comparación con bases de datos relacionales.

3. Curva de aprendizaje:

- Requieren un entendimiento previo para configurarlas y optimizarlas adecuadamente.

Apache Cassandra es una base de datos NoSQL de tipo columnar diseñada para escalar horizontalmente. A continuación, veremos cómo usarla con Python, con este ejemplo práctico con Apache Cassandra:

- Instalación de Cassandra y librería Cassandra-Driver:

```
bash

pip install cassandra-driver
```

- Conexión a Cassandra:

```
python

from cassandra.cluster import Cluster

# Conexión al clúster de Cassandra

cluster = Cluster(['127.0.0.1'])

session = cluster.connect()

# Crear un keyspace

session.execute("""

    CREATE KEYSPACE IF NOT EXISTS mi_keyspace

    WITH replication = { 'class': 'SimpleStrategy', 'replication_factor':
'1' }

""")

session.set_keyspace('mi_keyspace')
```

- Crear una tabla:

```
python

# Crear una tabla tipo columnar

session.execute("""

    CREATE TABLE IF NOT EXISTS usuarios (

        id UUID PRIMARY KEY,

        nombre TEXT,

        edad INT,

        pais TEXT

    )

""")
```

- Insertar datos:

```
python

import uuid

session.execute("""

    INSERT INTO usuarios (id, nombre, edad, pais)

    VALUES (%s, %s, %s, %s)

""", (uuid.uuid4(), "Juan", 30, "España"))
```

- Consultar datos:

```
python

rows = session.execute('SELECT * FROM usuarios')

for row in rows:

    print(f"ID: {row.id}, Nombre: {row.nombre}, Edad: {row.edad}, País:
{row.pais}")
```

Una tabla comparativa de las BBDD Columnares, respecto de las bases de datos SQL -relacionales, se muestra a continuación:

Aspecto	Columnares	Relacionales
Modelo	Por columnas	Por filas
Optimización	Consultas analíticas	Operaciones transaccionales
Escalabilidad	Horizontal	Vertical
Casos de uso	*Big data*, BI	Aplicaciones OLTP
Escritura Actualización	Costosa	Rápida

Tabla 7. Tabla comparativa entre las bases de datos columnares y las bases de datos SQL-relacionales.

Fuente: elaboración propia.

Bases de datos de grafos

Las bases de datos de grafos son una solución moderna y especializada para modelar, almacenar y consultar datos que tienen relaciones complejas. Si estás iniciándote en el mundo de las bases de datos, estas son unas herramientas poderosas para comprender y resolver problemas reales: redes sociales, sistemas de recomendaciones y análisis de rutas.

- ¿Qué es una base de datos de grafos?:

Una base de datos de grafos está diseñada para almacenar datos y sus relaciones en forma de un grafo. Este modelo utiliza **nodos, aristas y propiedades** para representar tanto los datos como las conexiones entre ellos.

- Definición:

Es un sistema de almacenamiento que emplea un modelo basado en grafos para capturar relaciones explícitas entre datos, optimizando consultas que dependen de múltiples niveles de conexión.

- Elementos clave:

1. Nodo *(vertex)*: representa una entidad o un objeto: personas, lugares o productos.
2. Arista *(edge)*: representa una relación entre nodos. Las aristas tienen direcciones y pueden tener propiedades.
3. Propiedades: atributos asociados tanto a nodos como a aristas: el nombre de un nodo o el peso de una relación.

- Ejemplo. Red de amigos:

1. Nodo: persona (Juan, Ana, Carlos).
2. Arista: «es amigo de».
3. Propiedad de la arista: nivel de confianza (por ejemplo, 0,9).

Visualización:

```
css

Juan --[es amigo de: 0.9]--> Ana

Ana --[es amigo de: 0.7]--> Carlos
```

- Características principales de las bases de datos de grafos:

1. Modelo natural:

- Ideal para representar sistemas con relaciones complejas.

2. Consultas optimizadas:

- Las relaciones directas permiten consultas rápidas sin necesidad de *joins* complejos.

3. Escalabilidad horizontal:

- Pueden distribuirse en varios nodos de almacenamiento.

4. Flexibilidad de esquema:

- No requieren una estructura fija: los nodos y aristas pueden tener diferentes propiedades.

5. Rendimiento para relaciones profundas:

- Excelentes para explorar conexiones profundas entre entidades.

- Casos de uso de las bases de datos de grafos:

1. Redes sociales:

- Modelado de relaciones entre usuarios, seguimiento de amigos o interacciones.

2. Sistemas de recomendación:

- Relacionar usuarios con productos según preferencias.

3. Gestión de rutas:

- Optimización de rutas en redes de transporte.

4. Detección de fraude:

- Identificar patrones sospechosos en redes de transacciones.

5. Gestión de conocimientos:

- Representación de relaciones jerárquicas y semánticas en bases de datos ontológicas.

- Arquitectura de una base de datos de grafos:

1. Almacenamiento en grafos:

- Los datos se almacenan como nodos y aristas en lugar de tablas y filas.

2. Indexación:

- Índices especializados que optimizan la localización de nodos y relaciones.

3. Lenguajes de consultas específicos:

- Usan lenguajes como Cypher (Neo4j) o Gremlin, para consultas en grafos.

4. Procesamiento distribuido:

- Algunos motores, como JanusGraph, soportan análisis distribuidos en grandes volúmenes de datos.

5. Motores de consultas:

- Optimizados para recorrer grafos en profundidad y encontrar patrones complejos.

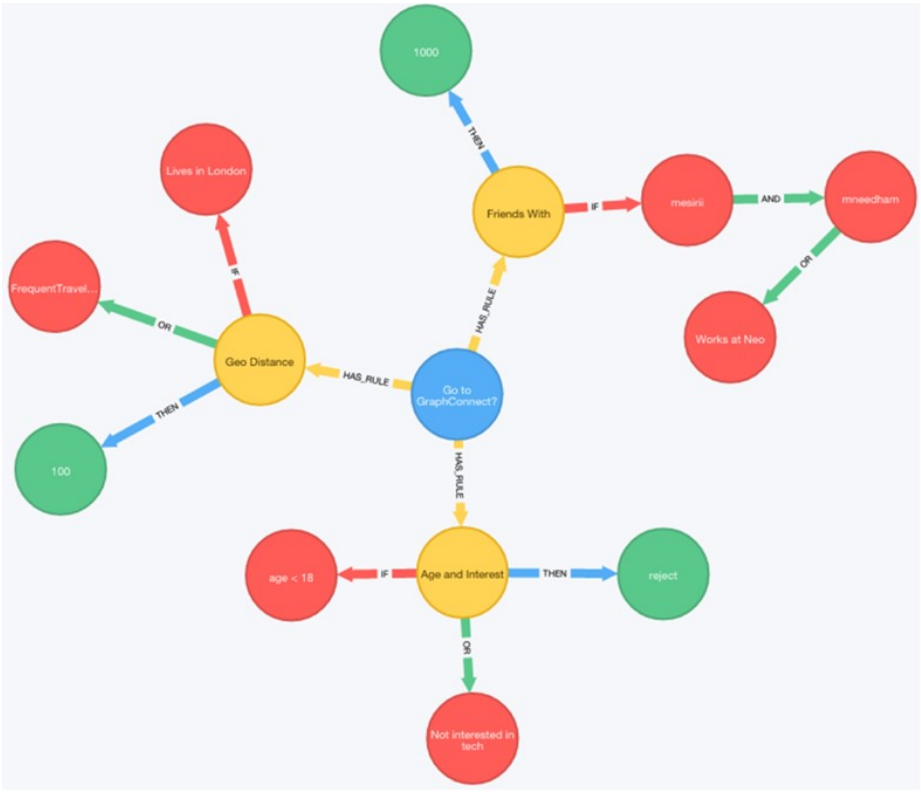

Figura 11. Imagen representativa del almacenamiento de datos en bases de datos NoSQL grafos. Fuente: Bachman, 2016.

- Ventajas de las bases de datos de grafos:

1. Relaciones complejas:

- Diseñadas para manejar sistemas con conexiones complejas entre entidades.

2. Rendimiento de consultas relacionales:

- Realizan consultas más rápidas que las bases relacionales cuando se trabaja con relaciones profundas.

3. Flexibilidad:

-

Permiten agregar nuevos tipos de nodos o relaciones sin modificar el esquema existente.

4. Visualización intuitiva:

1. Los grafos son fáciles de entender y visualizar.

5. Soporte para algoritmos de grafos:

- Soportan algoritmos como búsqueda de rutas más cortas, centralidad y detección de comunidades.
- Limitaciones de las bases de datos de grafos:

1. Transacciones complejas:

- No son ideales para sistemas OLTP tradicionales con muchas transacciones concurrentes.

2. Escalabilidad limitada en consultas simples:

- No son tan eficientes como las bases relacionales para consultas no relacionales.

3. Adopción más lenta:

- Requieren un cambio de paradigma en el diseño de datos.

Neo4j es una de las bases de datos de grafos más populares, utilizada ampliamente en aplicaciones de redes sociales y análisis de datos. A continuación, veremos un ejemplo práctico:

- Instalación de Neo4j y librería Py2neo:

```
bash

pip install py2neo
```

- Conexión a Neo4j:

```
python

from py2neo import Graph

# Conexión al servidor Neo4j

graph = Graph("bolt://localhost:7687", auth=("neo4j", "contraseña"))
```

- Crear nodos y relaciones:

```
python

# Crear nodos y relaciones

graph.run("""

    CREATE (juan:Persona {nombre: 'Juan', edad: 30})

    CREATE (ana:Persona {nombre: 'Ana', edad: 25})

    CREATE (juan)-[:AMIGO_DE {nivel_confianza: 0.9}]->(ana)

""")
```

- Consultar el grafo.

```
python

# Consulta de nodos y relaciones

consulta = graph.run("MATCH (p:Persona)-[r:AMIGO_DE]->(amigo) RETURN p,
amigo, r").data()

for resultado in consulta:

    print(f"{resultado['p']['nombre']} es amigo de {resultado['amigo']
['nombre']} con confianza {resultado['r']['nivel_confianza']}")
```

- Actualizar un nodo.

```
python

# Actualizar la edad de Ana

graph.run("MATCH (ana:Persona {nombre: 'Ana'}) SET ana.edad = 26")
```

En la siguiente tabla, mostramos una comparación de bases de datos grafos y bases de datos relacionales:

Aspecto	Grafos	Relacionales
Modelo	Nodos y aristas	Tablas y filas
Optimización	Consultas de relaciones complejas	Operaciones transaccionales
Escalabilidad	Horizontal	Vertical
Casos de uso	Redes, recomendaciones	OLTP
Visualización	Intuitiva	Tabular

Tabla 8. Comparativa entre las bases de datos de grafos y las bases de datos relacionales. Fuente: elaboración propia.

- Escenarios ideales para usar bases de datos NoSQL:

1. Grandes volúmenes de datos: cuando los datos superan las capacidades de las bases relacionales.

- Ejemplo: gestión de *logs* en aplicaciones con millones de usuarios.

2. Datos en tiempo real: procesamiento y análisis instantáneo.

- Ejemplo: aplicaciones de monitoreo de IoT.

3. Alta disponibilidad y escalabilidad: sistemas que requieren *uptime* continuo y crecimiento dinámico.

- Ejemplo: plataformas de comercio electrónico globales.

4. Diversidad de formatos: almacenamiento de datos con estructuras variadas.

- Ejemplo: almacenes de documentos en aplicaciones móviles.

5. Requerimientos bajos de consistencia: cuando es aceptable una consistencia eventual.

- Ejemplo: sistemas de mensajería instantánea.

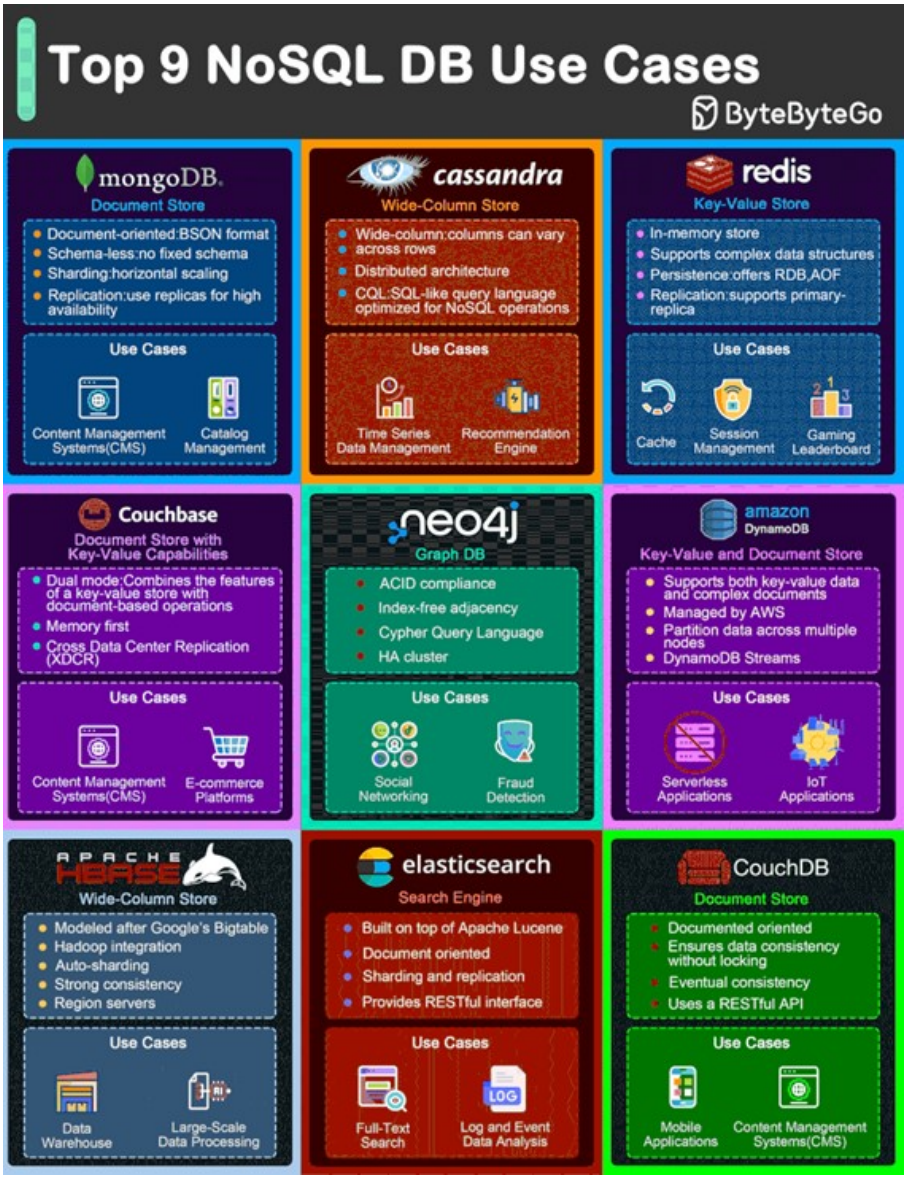

Figura 12. Infografía sobre herramientas de bases de datos NoSQL. Fuente: Al Bukari, 2024.

2.3. Amazon DocumentDB

Amazon DocumentDB es un servicio de base de datos completamente gestionado, escalable y compatible con MongoDB que permite a los desarrolladores, empresas y organizaciones crear, administrar y operar bases de datos documentales de forma fácil y eficiente.

Amazon DocumentDB está diseñado para facilitar la implementación de bases de datos de tipo NoSQL en la nube, proporcionando las características y beneficios típicos de la nube como la escalabilidad, la alta disponibilidad y la seguridad, sin tener que preocuparse por la administración compleja del *hardware* o del *software.*

Este servicio está optimizado para la compatibilidad con aplicaciones que utilizan MongoDB, lo que permite migrar o integrar estas aplicaciones a Amazon Web Services (AWS) sin necesidad de cambiar el código de las aplicaciones. Amazon DocumentDB permite a las organizaciones almacenar datos en documentos JSON y realizar consultas avanzadas de manera eficiente.

A continuación, detallaremos los aspectos clave de Amazon DocumentDB, sus características, su arquitectura, sus beneficios y casos de uso más comunes.

Aquí tienes el enlace de la guía de usuario de AWS Document DB:
https://docs.aws.amazon.com/es_es/documentdb/latest/developerguide/get-started-guide.html

---

¿Qué es Amazon DocumentDB?

Amazon DocumentDB es un servicio de base de datos documental totalmente gestionado, diseñado para almacenar y consultar documentos JSON de forma escalable y eficiente. Está basado en la arquitectura de Amazon Aurora, que es conocida por su alta disponibilidad y rendimiento en bases de datos SQL, pero optimizada para el almacenamiento de documentos de tipo NoSQL, como los documentos JSON utilizados comúnmente en aplicaciones de MongoDB.

Aunque Amazon DocumentDB es compatible con MongoDB a nivel de API, es un servicio independiente que se ejecuta en la infraestructura de AWS, lo que permite aprovechar la infraestructura de AWS para obtener una mayor seguridad, escalabilidad y facilidad de uso sin tener que gestionar servidores o infraestructura subyacente.

Entre sus principales ventajas, Amazon DocumentDB ofrece una capacidad de escalado automático, réplicas de alta disponibilidad y copias de seguridad automáticas, lo que facilita la gestión de bases de datos a gran escala sin necesidad de intervención manual. Esto hace que el servicio sea ideal para aplicaciones que requieren una base de datos altamente disponible y escalable, como las aplicaciones móviles, sitios web dinámicos y sistemas de análisis de datos.

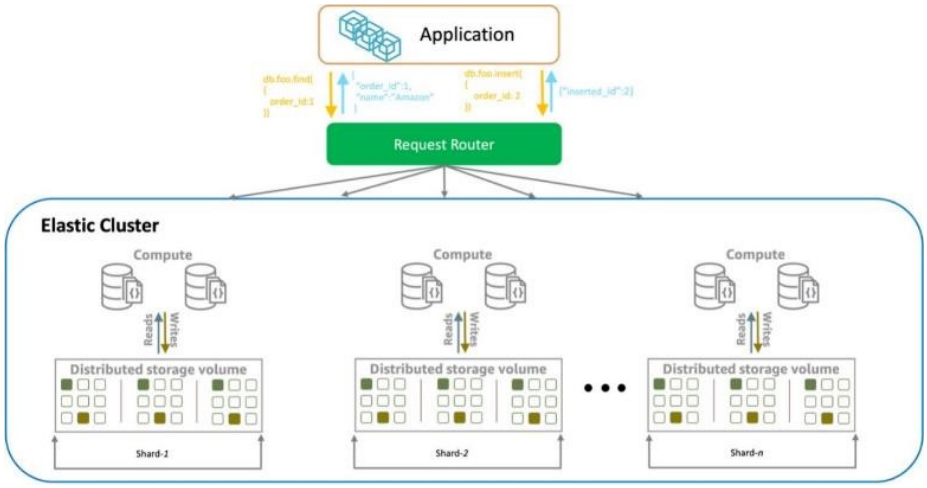

Figura 13. Diagrama arquitectura DocumentDB. Fuente: Unny y Callaghan, 2024.

Características de Amazon DocumentDB

- **Compatibilidad con MongoDB:** Amazon DocumentDB es totalmente compatible con las API de MongoDB, lo que significa que las aplicaciones que ya utilizan MongoDB pueden migrar a DocumentDB sin necesidad de modificar su código. Esto incluye las operaciones de lectura y escritura, así como las consultas de agregación.

- **Escalabilidad automática:** Amazon DocumentDB está diseñado para ser escalable, lo que significa que puede adaptarse al crecimiento de tus datos de forma automática. Puedes escalar horizontalmente añadiendo réplicas de lectura o verticalmente aumentando el tamaño de las instancias de la base de datos.

- **Alta disponibilidad y durabilidad:** Amazon DocumentDB proporciona copias de seguridad automáticas y réplicas en múltiples zonas de disponibilidad (AZ) dentro de una región de AWS, lo que garantiza la alta disponibilidad y la recuperación ante desastres. Los datos se almacenan en un sistema distribuido y redundante para asegurar la durabilidad.

- **Seguridad:** Amazon DocumentDB ofrece características de seguridad avanzadas, como cifrado en reposo mediante el uso de AWS Key Management Service (KMS) y control de acceso granular utilizando AWS Identity and Access Management (IAM). También ofrece integración con Amazon Virtual Private Cloud (VPC) para asegurar la comunicación entre las instancias de la base de datos.

- ***Backups* automáticos y restauración:** DocumentDB realiza copias de seguridad automáticas de tus datos, lo que permite restaurarlos fácilmente a cualquier punto en el tiempo, dentro del período de retención (de uno a treinta y cinco días).

- **Consultas rápidas:** DocumentDB está optimizado para realizar consultas rápidas en grandes volúmenes de datos. Utiliza índices automáticos para mejorar el rendimiento de las consultas sin necesidad de configurar manualmente los índices.

- **Integración con otros servicios de AWS:** Amazon DocumentDB se integra fácilmente con otros servicios de AWS, como Amazon Lambda, Amazon S3, Amazon Kinesis y AWS Glue, lo que facilita la creación de soluciones de análisis y procesamiento de datos de manera integral.

Casos de uso de Amazon DocumentDB

Amazon DocumentDB es ideal para varios casos de uso en aplicaciones modernas que requieren bases de datos documentales escalables y gestionadas. A continuación, se describen algunos de los casos de uso más comunes:

- **Aplicaciones web y móviles:** DocumentDB es ideal para aplicaciones web y móviles que requieren almacenar datos en formato JSON, como registros de usuario, preferencias y contenido dinámico. Al ser compatible con MongoDB, estas aplicaciones pueden migrar fácilmente a DocumentDB sin tener que modificar el código.

- **Comercio electrónico:** las plataformas de *e-commerce* pueden beneficiarse de Amazon DocumentDB para almacenar catálogos de productos, detalles de clientes y transacciones. La flexibilidad en la estructura de datos permite a las empresas de comercio electrónico añadir nuevos atributos o productos sin tener que modificar un esquema rígido.

- **Gestión de contenido:** las plataformas de gestión de contenido (CMS) que requieren almacenar artículos, imágenes, metadatos y comentarios de manera flexible pueden aprovechar DocumentDB. La capacidad de manejar datos jerárquicos y anidados permite representar la estructura compleja de los artículos y el contenido asociado.

- **Internet de las cosas** (IoT): Amazon DocumentDB es adecuado para almacenar y gestionar los grandes volúmenes de datos generados por dispositivos IoT, como registros de sensores, ubicaciones de dispositivos y estados de dispositivos. La escalabilidad de DocumentDB permite manejar fácilmente los flujos de datos masivos y ofrecer consultas rápidas para procesarlos.