Maestría en Ingeniería de Software

Base de Datos

Conceptos y comparaciones

Autor:
Lic. Carreira, Mariano

Docentes:
Dr. Grigera, Julian
Lic. Diclaudio, Federico

Índice

Abstract 3
Tipos de Datos 3
Estructurados 3
Semi-Estructurados 3
No Estructurados 3
Base de datos 4
Definición 4
Sistema de gestión de base de datos (SGBD) 4
Definición 4
Tipos 4
SGBD Relacional 5
Ejemplos más populares: 6
NoSQL 6
Clasificación de SGBD NoSQL 6
Ventajas 7
Métodos 7
Comparativo NoSQL 7
Benchmarking 16
Criterios de selección de motores 16
Según el tipo 16
Según el modelo de implementación 17
Según la finalidad de la investigación 17
Compatibilidad con ORM 17
Según conocimientos del autor 17
Metodología 18
Descripción 18
Ambiente 18
Hardware 18
Herramientas 19
Criterios de selección 19
Herramientas candidatas 20
Selección Final 21
Dataset 22
Comparación Estática entre motores 24
Prueba 25
1er Paso: Instalación de los motores y clientes 25
2do Paso: Instalación de JMeter 27
3er Paso: Descarga del Dataset 31
4to Paso: Creación de la base de datos en MS SQL Server 31
5to Paso: Creación de la base de datos en MongoDB 32
6to Paso: Configuración de Índices 34
7mo Paso: Queries 35
8vo Paso: Ejecución 41
8vo Paso: Resultados 42
Conclusión 46
Referencias 46

Abstract

Este proyecto de investigación compara el rendimiento de consultas y operaciones entre MongoDB y SQL Server utilizando Apache JMeter como herramienta de benchmarking. Se evalúan tres bases de datos de diferentes tamaños (pequeña, mediana y grande) para entender cómo cada sistema maneja diversas cargas de trabajo y volúmenes de datos.

Se analiza el tiempo de ejecución de consultas como único aspecto de evaluación bajo diferentes tamaños de base de datos.

Los resultados preliminares sugieren que MongoDB es robusto en el manejo de datos no estructurados, mientras que SQL Server destaca en consultas transaccionales complejas y garantía de propiedades ACID. Estos hallazgos proporcionan información valiosa, facilitando la elección del DBMS adecuado según las necesidades específicas de la aplicación y características de los datos.

Este estudio contribuye a entender las implicaciones prácticas de seleccionar entre bases de datos NoSQL y relacionales en distintos contextos de aplicación, mejorando la toma de decisiones en el diseño de aplicaciones basadas en datos.

Tipos de Datos

Estructurados

Los datos estructurados tienen un formato estandarizado que permite tanto al software como a las personas acceder a estos de forma eficaz. Por lo general, se trata de datos tabulares con filas y columnas que definen claramente sus atributos.

https://aws.amazon.com/es/what-is/structured-data/#:~:text=Structured%20data%20is%20data%20that,that%20clearly%20define%20data%20attributes.

Semi-Estructurados

Los datos semi-estructurados son una forma de datos estructurados que no obedecen a la estructura tabular de los modelos de datos asociados con bases de datos relacionales u otras formas de tablas de datos, pero que, no obstante, contienen etiquetas u otros marcadores para separar elementos semánticos y aplicar jerarquías de registros y campos dentro. los datos. Por lo tanto, también se la conoce como estructura autodescriptiva.
https://en.wikipedia.org/wiki/Semi-structured_data#cite_note-1

No Estructurados

Los datos no estructurados son información sin un modelo de datos establecido o son datos que no están ordenados de una manera predefinida.

Ejemplos comunes de datos no estructurados:

- Archivos de texto
- Archivos de video
- Informes
- Correo electrónico
- Imágenes

Las empresas están creando datos a un ritmo exponencial y la gran mayoría (entre el 80 % y el 90 %) son datos no estructurados.

https://aws.amazon.com/es/what-is/structured-data/#:~:text=Structured%20data%20is%20data%20that,that%20clearly%20define%20data%20attributes.

Base de datos

Definición

Una base de datos es una colección lógica de datos gestionada por un software específico. Una base de datos incluye no sólo los datos del usuario sino también los objetos necesarios para su gestión (por ejemplo, índices o archivos de registro).

Sistema de gestión de base de datos (SGBD)

Definición

Un sistema de gestión de bases de datos (SGBD) es un software que se utiliza para gestionar una colección de datos lógicamente relacionados (base de datos). Un SGBD forma junto con una base de datos el sistema de base de datos.

https://db-engines.com/en/article/Database+Management+System

Tipos

Un SGBD que admite el modelo de datos relacionales a menudo se denomina SGBDR. En caso de que admita otros modelos de datos, a menudo se incluye como un sistema NoSQL.

https://db-engines.com/en/article/Database+Management+System

SGBD Relacional

Los sistemas de gestión de bases de datos relacionales (SGBDR) admiten el modelo de datos relacional (orientado a tablas). El esquema de una tabla (esquema de relación) se define por el nombre de la tabla y un número fijo de atributos con tipos de datos fijos. Un registro (entidad) corresponde a una fila de la tabla y consta de los valores de cada atributo. Por tanto, una relación consta de un conjunto de registros uniformes.

Los esquemas de tablas se generan mediante la normalización en el proceso de modelado de datos.

Ciertas operaciones básicas se definen sobre las relaciones:

- Operaciones clásicas de conjuntos (unión, intersección, y diferencia)
- Selección (selección de un subconjunto de registros según ciertos criterios de filtrado para los valores de los atributos)
- Proyección (seleccionando un subconjunto de atributos/columnas de la tabla)
- Join: conjunción especial de múltiples tablas como combinación del producto cartesiano con selección y proyección.

Estas operaciones básicas, así como las operaciones de creación, modificación y eliminación de esquemas de tablas, operaciones de control de transacciones y gestión de usuarios, se realizan mediante lenguajes de bases de datos, siendo SQL un estándar bien establecido para dichos lenguajes.

Los primeros sistemas de gestión de bases de datos relacionales aparecieron en el mercado a principios de los años 1980 y desde entonces han sido el tipo de SGBD más utilizado. A lo largo de los años, muchos SGBDR se han ampliado con conceptos no relacionales, como tipos de datos definidos por el usuario, atributos no atómicos, herencia y jerarquías, por lo que a veces se les denomina SGBD relacionales de objetos.

https://db-engines.com/en/article/Relational+DBMS?ref=RDBMS

Ejemplos más populares:

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL
- IBM Db2

https://db-engines.com/en/ranking/relational+dbms

NoSQL

Los sistemas de bases de datos NoSQL son una alternativa a los SGBD relacionales convencionales. No utilizan un modelo de datos relacional y normalmente no tienen una interfaz SQL. Aunque este tipo de sistemas existe desde hace muchos años (algunos incluso más que los sistemas relacionales), el término NoSQL se introdujo por primera vez en 2009, cuando se desarrollaron muchos sistemas nuevos para hacer frente a los nuevos requisitos de los sistemas de gestión de bases de datos en ese momento, P.ej. Big Data, escalabilidad y tolerancia a fallos para grandes aplicaciones web.

El acrónimo NoSQL a menudo se entiende como "No solo SQL", lo que implica que los sistemas relacionales son una tecnología probada pero no necesariamente la opción óptima para cada tipo de uso previsto.

Clasificación de SGBD NoSQL

Los sistemas NoSQL son un grupo heterogéneo de sistemas de bases de datos muy diferentes. Por lo tanto, cada intento de clasificación fracasa en clasificar uno u otro sistema. Sin embargo, las siguientes categorías son bien aceptadas:

- Key-Value Stores
- Wide Column Stores
- Document Stores
- Graph DBMS
- RDF Stores (resource description framework)
- Native XML DBMS
- Content Stores
- Multivalue DBMS
- Search Engines
- Blockchain
- Spatial
- Time Series
- Vector
- Object Oriented
- Event Stores
- Navigational SGBD
- Multi Modelo

Ventajas

Si bien no todas las clases mencionadas anteriormente tienen las mismas ventajas generales, pero se benefician de una combinación de los siguientes aspectos:

- **Alta Performance**
- Fácil **distribución de datos en diferentes nodos** (por ejemplo, fragmentación), logrando así escalabilidad y tolerancia a fallas
- **Mayor flexibilidad** mediante el uso de un modelo de datos sin esquemas.
- **Administración más sencilla**.

Métodos

Estas ventajas se logran mediante uno o más de los siguientes enfoques:

- Sin modelo de datos relacionales normalizado.
- Abandono de uno o más de los criterios ACID.
- Posibilidades menos poderosas para consultar los datos.

https://db-engines.com/en/article/NoSQL

Comparativo NoSQL

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Key-Value	• Persiste pares Clave/Valor.	• Simplicidad. • Acceso rápido vía clave.	• No adecuada para aplicaciones complejas. • Capacidad de query limitada.	• Cache. • Datos de sesión.	• Redis. • Memcache.
Wide Column	• Persiste datos en columnas dinámicas. • No estructurada. • Mapeo multidimensional.	• Performance en queries y escritura. • Alta escalabilidad. • Flexibilidad del modelo.	• Capacidad limitada de queries. • Modelado limitado. • Soporte ACID limitado.	• Logging. • IoT. • Análisis en tiempo real. • Preferencias de usuario. • Alto volumen de datos. • Big Data. • Warehousing.	• Cassandra. • HBase. • BigTable.

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Document Stores	• Información documental. • Libre de Schema. • Anidación de documentos. • Generalmente JSON.	• Alta escalabilidad. • Flexibilidad del modelo.	• Consistencia eventual. • Falta de estandarización. • Sin Indexación secundaria. (más de 1 campo)	• BigData. • Análisis Tiempo Real.	• MongoDB. • DynamoDB.
Grafos	• Compuesta por nodos y vértices interconectados.	• Queries optimizados para relaciones.	• No permite el acceso directo a nodos según sus atributos.	• Redes sociales. • Sistemas de recomendación.	• Neo4J. • Virtuoso.
RDF (triplestore)	• Representación de la inf.en forma de	• Queries semánticas.	• Poca madurez.	• Web Semántica	• MarkLogic.

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
	tripletas (Sujeto, Predicado, Objeto). • Sub-tipo de SGBD Grafos.	• Interoperabilidad.	• Alta curva de aprendizaje.	• Relación de datos.	• Virtuoso.
XML Nativo	• Formato XML. • Soporte para XQuery, XPath y XSLT	• Soporte a metadatos. • Flexibilidad del modelo.	• Complejidad y baja performance para queries complejos. • Adopción limitada.	• WebServices. • Gestión documental.	• MarkLogic • Virtuoso
Content Stores	Administrar contenido digital. Datos no estructurados. Rico en metadata.	Soporte a versionado. Flexibilidad de datos. Soporte a metadatos. Control de acceso.	Problemas de escalabilidad. Migración compleja.	Administración de activos digitales. Gestión documental.	Jackrabbit ModeShape

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Multivalor	Similar a las relacionales, persiste en tablas. Puede asignar más de un valor a una columna. (Ej Array). Intermedio entre modelo relacional y documental.	Flexibilidad del modelo. Eficiente para datos jerárquicos.	No optimizado para joins. Soporte SQL limitado.	ERPs. Gestión de Inventarios.	Adabas UniData,UniVerse jBASE
Search Engines	Especializados en buscar datos. Derivación (reducción de palabras flexionadas a su raíz)	Soporte para expresiones de búsqueda complejas. Búsqueda de texto completo. Clasificación y agrupación de resultados Búsqueda distribuida para alta escalabilidad	Indexación compleja. Alto consumo de recursos de storage.	ECommerce. Buscadores Web. Foros. Diarios.	Elasticsearch. Splunk

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Blockchain	Descentralización. Cadena de bloques. Criptografía.	Inmutabilidad. Seguridad.	Problemas de escalabilidad. Consumo eléctrico intensivo.	Monedas Crypto. Legales. Libro contable.	BigchainDB.
Espaciales	Almacenar, manipular y consultar datos espaciales de manera eficiente. Tipos de datos dedicados para almacenar datos espaciales índices espaciales	Recuperar eficientemente puntos que están dentro de una cierta distancia de otros objetos. Calcular distancias, fusionar o intersecar objetos y calcular propiedades de objetos, como áreas de polígonos.	Desafíos de rendimiento con grandes conjuntos de datos.	GPS y Mapas. Navegación. Vehículos-Autónomos.	PostGIS. Aerospike. SpatiaLite.
Time Series	Almacenamiento de datos ordenados por	Recopila, almacena y	Soporte limitado para consultas	Los datos de series temporales	InfluxDB.

<table>
<tr><th>Tipo</th><th>Características</th><th>Ventaja</th><th>Desventajas</th><th>Escenarios</th><th>Ejemplo</th></tr>
<tr><td rowspan="2"></td><td>tiempo

Optimizado para manejar datos de series de tiempo: cada entrada está asociada con una marca de tiempo.</td><td>consulta de manera eficiente varias series temporales con altos volúmenes de transacciones</td><td>complejas,

Caso de uso especializado.</td><td>pueden ser producidos por sensores, medidores inteligentes o RFID en el llamado Internet de las Cosas.

Análisis del mercado financiero</td><td>Prometheus.

Kdb.</td></tr>
<tr><td colspan="5">
<table>
<tr><th colspan="3">Key</th><th>Value</th></tr>
<tr><td>Metric name</td><td>Dimensions (labels)</td><td>Timestamp</td><td>Sample value</td></tr>
<tr><td>http_requests_total</td><td>{status="200", method="GET"}</td><td>@1112596200</td><td>987654</td></tr>
<tr><td>http_requests_total</td><td>{status="200", method="GET"}</td><td>@1112596400</td><td>986575</td></tr>
<tr><td>http_requests_total</td><td>{status="404", method="GET"}</td><td>@1112596600</td><td>946574</td></tr>
<tr><td>http_requests_total</td><td>{status="200", method="GET"}</td><td>@1112596800</td><td>976575</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">Vectores</td><td>Optimizado para almacenamiento, indexación y consulta eficientes de datos vectoriales de alta dimensión.

Algoritmos especializados y estructuras de datos para respaldar la búsqueda de similitudes.

Normalmente puede manejar múltiples tipos de datos, incluidos numéricos, de texto y binarios.</td><td>Alto rendimiento, escalabilidad y flexibilidad.</td><td>Casos de uso especializado.</td><td>Aprendizaje automático.

Minería de datos.

Sistemas de recomendaciones
.</td><td>Kdb.

Pinecone.

Chroma.</td></tr>
<tr><td colspan="5">
</td></tr>
<tr><td>Object Oriented</td><td>Almacenar los objetos en una base de datos de una manera que</td><td>Soporte nativo a orientación de objetos.</td><td>Falta de estandarización. Desafíos de</td><td>Sistemas con modelización compleja.</td><td>InterSystems IRIS.</td></tr>
</table>

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
	corresponda a su representación en un lenguaje de programación. Modelo de datos con clases (el esquema de objetos), propiedades y métodos. Lenguajes de consulta propios tipo SQL	Previene el uso de ORM y capas intermedias.	escalabilidad.	Dominios complejos.	Actian NoSQL Database. ObjectStore. Db4o.
Event Stores	Implementar el concepto de abastecimiento de eventos. Persiste todos los eventos de cambio de estado de un objeto junto con una marca de tiempo, creando así series de tiempo para objetos individuales. El estado actual de un objeto se puede inferir reproduciendo todos los eventos de ese objeto desde el momento 0 hasta el momento actual. Registro inmutable de cambios	Alta consistencia de datos. Recupero de información histórica. Alta escalabilidad.	No se admiten modificaciones o eliminaciones de eventos ya persistentes. Alto consumo de storage.	Logging. Sistemas financieros. IoT.	EventStoreDB . IBM Db2 Event Store.

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Navigational	Permitir el acceso a conjuntos de datos sólo a través de registros vinculados. Se agrupan en SGBD jerárquicos y SGBD de red	Eficiente para datos vinculados, estructura jerárquica.	Baja adopción. Flexibilidad de consulta limitada.	Sistemas de archivos. Modelos de datos de red.	IMS (IBM) IDMS
Multi Modelo	Soporte para múltiples modelos de datos (por ejemplo, documento, gráfico, relacional) dentro de un sistema de base de datos	Versatilidad. Soporte para múltiples modelos de datos dentro de un sistema	Complejidad. Posibles compensaciones de rendimiento	Aplicaciones que requieren diversos tipos de datos y persistencia políglota	Microsoft Azure Cosmos DB. Apache Ignite.

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo

Benchmarking

Criterios de selección de motores

Según el tipo

Como primer criterio de selección, se evalúa el tipo de base de datos, se optó (en primera instancia) por seleccionar los tres tipos más utilizados en el mercado, según db-engines a Marzo de 2024.
De los tipos catalogados anteriormente, se observa que los sistemas de base de datos relacionales son los màs utilizados, con un porcentaje de mercado del 72,4%, siguiéndolo los Documentales con un 10,3%, el tercero más utilizado se encuentra bastante más abajo con un 5,3%, los sistemas de base de datos basados en Key-values.

https://db-engines.com/en/ranking_categories

De estos 3 tipos, el de clave valor es el que difiere significativamente su finalidad con respecto a los otros dos. Generalmente, como se mencionó en el cuadro anterior, los Key-Value son ampliamente utilizados en escenarios de Caché y datos de sesión, no así en datos transaccionales como los transaccionales y documentales. Por ende queda descartado de la prueba, estableciéndose para la misma solo los dos primeros.

Para conocer el método de puntaje de db-engines.com, remitirse a
https://db-engines.com/en/ranking_definition

Según el modelo de implementación

Como segundo método de selección ya del motor especìfico, se tuvo en cuenta el ranking de los 10 motores de los dos tipos seleccionados, los cuales posean licencias de utilización gratuita o de pruebas, con las mayores prestaciones en cuanto a rendimiento, de esta manera se trata de garantizar que la prueba se realiza con las mayores/mejores prestaciones posibles.

Rank			DBMS	Database Model	Score		
Mar 2024	Feb 2024	Mar 2023			Mar 2024	Feb 2024	Mar 2023
1.	1.	1.	Oracle	Relational, Multi-model	1221.06	-20.39	-40.23
2.	2.	2.	MySQL	Relational, Multi-model	1101.50	-5.17	-81.29
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	845.81	-7.76	-76.20
4.	4.	4.	PostgreSQL	Relational, Multi-model	634.91	+5.50	+21.08
5.	5.	5.	MongoDB	Document, Multi-model	424.53	+4.18	-34.25
6.	6.	6.	Redis	Key-value, Multi-model	157.00	-3.71	-15.45
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	134.79	-0.95	-4.28
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	127.75	-4.47	-15.17
9.	9.	↑ 11.	Snowflake	Relational	125.38	-2.07	+10.98
10.	10.	↓ 9.	SQLite	Relational	118.16	+0.88	-15.66

https://db-engines.com/en/ranking

Según la finalidad de la investigación

En el transcurso de la materia, se abordaron temas relacionados a los ORM y sistemas mayormente transaccionales/operacionales. Por este motivo se descartan motores de búsqueda o de clave valor.

Compatibilidad con ORM

Dado que en las clases estuvimos viendo ORM, y para poder generar conocimiento común con el otro tipo de TP, se optó por utilizar una base de datos relacional y una documental.

Según conocimientos del autor

El autor posee experiencia en bases de datos Ms SQL Server y en menor medida MongoDB.

Seleccionados

Se optó finalmente por los siguientes motores:

Motor 1:
Microsoft SQL Server 2022 (RTM) - 16.0.1000.6 (X64) Oct 8 2022 05:58:25 Copyright (C) 2022 Microsoft Corporation Developer Edition (64-bit) on Windows 10 Pro 10.0
https://info.microsoft.com/ww-landing-sql-server-2022.html?culture=en-us&country=US

Motor 2:
MongoDB Community Edition, Version 7.0.6
https://www.mongodb.com/try/download/community

Metodología

Descripción

Definir los objetivos de la pruebas

El primer punto será definir los objetivos de la prueba. En este caso el objetivo es comparar el rendimiento de diferentes motores de base de datos, tanto SQL como NoSql con gran cantidad de datos, en el orden de los millones, bajo operaciones CRUD (Create, Read, Update y Delete).

Seleccionar un método de prueba

El método de prueba será el de ejecutar exactamente los mismos scripts o sentencias (o sus equivalentes) en las diferentes bases de datos a probar. En el caso de las lecturas se repetirán X cantidad de veces para aprovechar los mecanismos de caché de los diferentes motores y así también podes analizar su comportamiento en general.

Realizar las pruebas

Para realizar las pruebas, se decidirá entre varias opciones de herramientas de benchmarking, orientadas a base de datos, con compatibilidad para los diferentes motores, tratando de evitar latencias adicionales como virtual machines, redes, containers de docker, etc.

Analizar los resultados

Una vez ejecutados los scripts y generados los resultados, se procede a su presentación e interpretación. Se tendrá en cuenta principalmente los tiempos de ejecución, y medidas accesorias que ayudarán a la interpretación de los resultados como así también a obtener conclusiones.

Ambiente

Hardware

El hardware en el que correrán las bases de datos será:

Modelo:
Laptop Lenovo ThinkPad X1

Procesador:
13th Gen Intel(R) Core(TM) i7-1370P, 1900 Mhz, 14 Core(s), 20 Logical Processor(s)

Memoria:

Installed Physical Memory (RAM)	64.0 GB
Total Physical Memory	63.7 GB
Total Virtual Memory	72.5 GB

Sistem Operativo:
Microsoft WIndows 11 - Enterprise Edition - 64 bits

Storage:
Disco SSD de 2TB de capacidad

Networking:
Las pruebas se realizan en el entorno local donde ambas bases de datos están instaladas.

Herramientas

Criterios de selección

Para la selección de la herramienta de testing se tuvieron en cuenta y analizaron las siguientes características y requerimientos:

- **Amigable**: La herramienta tiene que ser amigable e intuitiva. Se descarta a priori la codificación utilizando librerías como herramienta de pruebas, dado que pueden existir errores humanos los cuales podrían ya haberse considerado en herramientas con cierto tiempo de existencia en el mercado.

- **Compatibilidad** con diferentes motores: Debe ser compatible y poseer los conectores necesarios y actualizados con los motores y versiones a testear. (SQL Server y MongoDb)

- **Actualizada**: Como se mencionó anteriormente, debe estar actualizada para poder conectar, consumir y aprovechar las mejoras introducidas en las versiones a testear.

- **Compatibilidad con Sistema Operativo:** Debe poder correr e instalarse en el sistema operativo del hardware utilizado para la prueba, evitando cualquier overhead o latencias introducidos por máquinas virtuales o imágenes de docker.

Método de búsqueda

El método de búsqueda de herramientas candidatas fue la utilización del motor de búsqueda google utilizando las palabras clave como:

“Benchmarking db tools”
“Database testing tools”
“DB Comparison tools”
“How to compare databases performance”

Resultados Preliminares

URLs:

https://theqalead.com/tools/best-database-testing-tools/
https://www.linkedin.com/advice/0/how-do-you-load-test-benchmark-databases
https://www.c-sharpcorner.com/article/top-database-performance-testing-tools/
https://github.com/memsql/dbbench/blob/master/TUTORIAL.md
https://github.com/YuriyIvon/DatabaseBenchmark

Herramientas candidatas

HammerDB:

Url:

https://www.hammerdb.com/

Pro:

- Potente herramienta utilizada para las pruebas de base de datos relacionales.

Contra:

- No es compatible con base de datos NoSQL como Mongodb.

YCSB:

Url:

https://github.com/brianfrankcooper/YCSB

Pro:

- Una de las herramientas más potentes encontradas.
- Tenía a Yahoo como sponsor.
- Compatible con gran variedad de motores SQL y NoSQL.

Contra:

- Último release en 2019, puede estar desactualizada con respecto a los motores actuales y conectores actuales.
- Compleja en su utilización, posee una curva de aprendizaje alta.
- Es posible que necesite codificación para adaptarlo a los nuevos motores.

JMeter

Url:

https://jmeter.apache.org/

Pro:

- Herramienta conocida e instalada en el mercado como una de las más utilizadas en las pruebas de rendimiento, tanto base de datos como API.
- Compatible con Windows 11.
- Soporte a bases de datos SQL y NOSql con conexión a través de conectores.
- Configurable y adaptable.

- Simple de utilizar

Contra:

- Limitadas representación de los resultados, aunque suficientes para el objetivo de esta prueba.
- No existen versiones recientes, la más reciente es la 3.0 de 2016 (aunque pueden configurarse conectores actualizados(

DatabaseBenchmark:

Url:

https://github.com/YuriyIvon/DatabaseBenchmark

Pro:

- Open Source
- Actualizada y soporte activo
- Compatible con varios motores SQL y NoSQL

Contra:

- No posee (al menos no se encontraron) versiones con UI amigable, solo utilizable por línea de comandos.
- No soporta JOINs en queries nativos.

Selección Final

Dada la gran cantidad de recursos de aprendizaje como sitios web, tutoriales en video, etc. Además de una curva de aprendizaje baja, alta flexibilidad, adaptable a nuevos conectores y una UI altamente amigable, la seleccionada es:

Dataset

Criterios de selección

Para la selección del dataset se priorizaron las siguientes caracterìsticas:

- Tabla transaccionales largas, relacionadas con una o más tablas maestras.
- Más de 1 millón de registros.
- En lo posible datos reales, no mocks.
- Que el dataset en cuestión exista en un formato que pueda ser importado fácilmente a los diferentes motores a testear.

Metodo de busqueda

El método de búsqueda se basó en la utilización de la herramienta buscador Google, utilizando palabras clave como:

"Dataset Large"
"Dataset Huge"
"Dataset Microsoft Sql Server and Mongo"
"Dataset for benchmarking"
"MongoDb MSSql Server Dataset for benchmarking"
"Test Dataset"
"Dataset públicos Argentina"
"Dataset para benchmarking Sql Server"

Encontrados

1) **Datasets del gobierno Argentino**
URL:
https://www.datos.gob.ar/dataset
Descripción:
Set diverso de datos mayormente estadísticos del gobierno Argentino.
Pro:
- Datos reales
- Datasets de gran tamaño

Contra:
- Formatos los cuales es necesario importarlos y adaptarlos (DTL) a los tipos de datos de los motores a testear.
- No se encontraron datasets con entidades relacionadas, o al menos es necesario buscar los maestros de las relaciones en otros vínculos del sitio.

2) **Stack Exchange All sites Dataset**

URL:
https://archive.org/details/stackexchange
Descripción:
Datasets todos los sitios de StackExchange (como por ej stack overflow)

Pro:

- Datos reales
- Datasets de gran tamaño
- Posee tanto las tablas maestras como también las operacionales.

Contra:

- No poseen las claves foráneas, sólo los datos.
- Formatos los cuales es necesario importarlos y adaptarlos (DTL) a los tipos de datos de los motores a testear.

3) **StackOverflow SQL Server Datasets**

URL:
https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/

Descripción:
Subset de datos nombrados en el punto anterior, contiene solo los de stack overflow ya adaptados y listos para ser importados en una base de datos SQL Server.

Pro:

- Datos reales
- Datasets de gran tamaño
- Formateados y listos para importar en SQL Server
- Posee tanto las tablas maestras como también las operacionales.

Contra:

- No poseen las claves foráneas, sólo los datos.

Seleccionado

#3 StackOverflow SQL Server Datasets

Dentro del dataset seleccionado, se sub-seleccionaron para la prueba solo algunas tablas, teniendo como tabla principal para las consultas a los **Posts** y las tablas maestras de **Usuarios** y **PostTypes**. La tabla principal de consultas (Posts) posee **17.142.169** registros, considerados suficientes para la prueba. Tamaño **50GB.**

Fig. Subset de tablas seleccionado para la prueba.

Comparación Estática entre motores

Aspecto	MongoDB	SQL Server
Fabricante	Desarrollada por MongoDB Inc., release inicial Febrero 11, 2009.	Desarrollado por Microsoft Corporation y lanzado inicialmente el 24 de abril de 1989..
Tipo	Base de datos no relacional: orientada a documentos (estructura clave-valor)	Base de datos relacional: formato de tablas

Lenguaje de implementación	JavaScript, Python, Java, PHP, C++, C, Ruby, Perl	C, C++
Licencia	Open-Source	License required
Esquema de datos	Esquema dinámico	Esquema fijo
Lenguaje de Query	MongoDB Query Language	SQL
Escalabilidad	Horizontal	Vertical
Map Reduce	Soporte al método Map Reduce.	No soporta el método Map-Reduce.
Joins	No	Si
Transacciones	MongoDB proporciona transacciones ACID de múltiples documentos con aislamiento de instantáneas.	MS SQL Server proporciona transacciones ACID sin aislamiento de instantáneas.
Soporte XML	No	Si

https://hevodata.com/learn/mongodb-vs-sql-server/

Prueba

1er Paso: Instalación de los motores y clientes

El primer paso consiste en la instalación de ambos motores, los mismos pueden descargarse desde los sitios de sus fabricantes.

Descargas

Ms SqlServer Developer 2022:
https://go.microsoft.com/fwlink/p/?linkid=2215158&clcid=0x409&culture=en-us&country=us

Mongodb Community:
https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-7.0.7-signed.msi

Para facilitar las consultas y gestionar de manera visual y amigable dichas base de datos, procedemos luego a instalar los clientes respectivos.

MS Sql Server Management Studio (SSMS)
https://aka.ms/ssmsfullsetup

Mongodb Compass
https://downloads.mongodb.com/compass/mongodb-compass-1.42.3-win32-x64.exe?_ga=2.32022904.792464078.1711297515-188737971.1710007121

Instalación

Luego de ejecutar los instaladores y dejar seleccionadas sus opciones por defecto, los motores quedarán corriendo con las siguientes características.

Ms SQL Server:

Ip:	**127.0.0.1 (localhost)**
Port:	**1433**
Admin User:	**sa**
Password:	***** (oculto)

Mongodb Community:

Ip:	**127.0.0.1 (localhost)**
Port:	**27017**
Admin User:	**sa**
Password:	***** (oculto)
Conn. String:	**mongodb://localhost:27017**

2do Paso: Instalación de **JMeter**

Descarga

Luego de ejecutar los instaladores y dejar seleccionadas sus opciones por defecto, los motores quedarán corriendo con las siguientes características.

Java 8
https://www.java.com/es/download/ie_manual.jsp

JMeter

https://jmeter.apache.org/download_jmeter.cgi

Instalación

El primer paso de la instalación consiste en instalar Java 8. Luego de haber instalado **Java 8** se deberán extraer los archivos del zip descargado de **JMeter (apache-jmeter-5.6.3.zip).** Para ejecutar JMeter se deberá ejecutar el archivo jmeter.bat en una consola de windows.

Drivers (Conectores)

Para lograr la conexión entre **JMeter** y los motores seleccionados, es necesario tener actualizados los conectores **JDBC**.

Ms Sql Server:
https://learn.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16

Una vez descargado el archivo **sqljdbc_12.6.1.0_enu.zip** se debera descomprimir el archivo **mssql-jdbc-12.6.1.jre8.jar** dentro de la carpeta lib de **JMeter**

MongoDB:
Del mismo modo para mongoDb, como así también bibliotecas soporte necesarias deberán descargarse los driver desde https://repo1.maven.org/maven2/org/mongodb/

- mongo-java-driver-3.12.14.jar
- mongodb-driver-core-3.12.14.jar
- bson-3.12.14-javadoc.jar
- groovy-all-3.0.10-groovydoc.jar
- mongodb-driver-syn-3.12.14.jar

3er Paso: Descarga del Dataset

Para descargar el dataset, se debe hacer click en el enlace
https://downloads.brentozar.com/StackOverflow2013_201809117.7z

Luego se deben descomprimir los archivos de sql Server **.mdf** en alguna carpeta deseada.
https://www.whatisfileextension.com/es/mdf/

4to Paso: Creación de la base de datos en **MS SQL Server**

Una vez descomprimidos los archivos **mdf,** se deberá adjuntar la base de datos a la instancia del motor que instalamos.
https://learn.microsoft.com/en-us/sql/relational-databases/databases/attach-a-database?view=sql-server-ver16#SSMSProcedure

Una vez atachada, podremos consultar la cantidad de registros:

Nota: La base de datos original es llamada **StackOverflow2013**, fue renombrada a **StackOverflow2010** dado que ya se habían escrito las pruebas con ese nombre.

5to Paso: Creación de la base de datos en **MongoDB**

Para la creación y la carga del dataset en mongo, se resolvió utilizar la herramienta de migración **MongoDB Relational Migrator** la cual permite la migración de una base de datos del relacional existente a una de MongoDb de manera sencilla posibilitando el mapeo entre tablas y documentos.
https://www.mongodb.com/try/download/relational-migrator

1er Paso: Se mantiene la compatibilidad con el modelo de SQL Server, esto quiere decir que se las entidades migradas no estarán agregadas (como lo propone el modelo documental de mongo) sino que serán consideradas todas ellas como TOP-Level. Esto quiere decir, 1 colección distinta por cada una.

Fue necesario antes crear la foreign key entre las tablas **Posts, PostTypes y Users** en MSSQl Server para ser detectadas como relaciones en la herramienta de migración.

Total de registros migrados:

6to Paso: Configuración de Índices

Para poder comparar las bases de datos de manera equitativa, será la de equiparar índices, más allá del método de indexación de cada una, será necesario al menos poseer los mismos “campos” indexados en ambas.

Una vez equiparados los índices, se crean las siguientes consultas que serán luego utilizadas en la prueba.

7mo Paso: Queries

0) Consultas de Inserción, Eliminación y Modificación

a) INSERT SQL - 500000 Posts

Sql

```
-- Generate 1,000,000 rows of random data and insert into the "Posts"
table
SET IDENTITY_INSERT Posts ON
DECLARE @counter INT = 1
DECLARE @maxRows INT = 500000

WHILE @counter <= @maxRows
BEGIN
   INSERT INTO Posts (
       Id,
       AcceptedAnswerId,
       AnswerCount,
       Body,
       ClosedDate,
       CommentCount,
       CommunityOwnedDate,
       CreationDate,
       FavoriteCount,
       LastActivityDate,
       LastEditDate,
       LastEditorDisplayName,
       LastEditorUserId,
       OwnerUserId,
       ParentId,
       PostTypeId,
       Score,
       Tags,
       Title,
       ViewCount
   )
   VALUES (
       -@counter,
       ABS(CHECKSUM(NEWID())) % 1000000 + 1, -- Random value between 1
and 1,000,000
       ABS(CHECKSUM(NEWID())) % 100,
       CAST(NEWID() AS NVARCHAR(MAX)),
       GETDATE(), -- ClosedDate can be null
       ABS(CHECKSUM(NEWID())) % 100,
       NULL, -- CommunityOwnedDate can be null
```

```
        GETDATE(), -- Current date
        ABS(CHECKSUM(NEWID())) % 100,
        GETDATE(), -- Current date
        GETDATE(), -- Current date
        LEFT(CONVERT(VARCHAR(255), NEWID()), 30), --
LastEditorDisplayName can be null
        null,
        11,
        null, -- Random value between 1 and 1,000,000
        1, -- Random value between 0 and 9
        ABS(CHECKSUM(NEWID())) % 100,
        CAST(NEWID() AS NVARCHAR(300)),
        CAST(NEWID() AS NVARCHAR(500)),
        ABS(CHECKSUM(NEWID())) % 1000000
    )

    SET @counter = @counter + 1
END
SET IDENTITY_INSERT Posts OFF
```

Mongo (mongo shell script)

```
// Connect to the database
const db = connect("mongodb://localhost:27017/yourDatabaseName");

// Define the number of rows to insert
const maxRows = 500000;

// Define a function to generate random data
function randomValue(max) {
    return Math.floor(Math.random() * max) + 1;
}

// Insert rows of random data
for (let counter = 1; counter <= maxRows; counter++) {
    db.Posts.insert({
        Id: -counter,
        AcceptedAnswerId: randomValue(1000000),
        AnswerCount: randomValue(100),
        Body: new ObjectId().str, // Generate a random string
        ClosedDate: new Date(), // Current date
        CommentCount: randomValue(100),
        CommunityOwnedDate: null, // Nullable field
        CreationDate: new Date(), // Current date
        FavoriteCount: randomValue(100),
        LastActivityDate: new Date(), // Current date
        LastEditDate: new Date(), // Current date
```

```
        LastEditorDisplayName: new ObjectId().str.substr(0, 30), //
Random string, max 30 chars
        LastEditorUserId: null, // Nullable field
        OwnerUserId: 11,
        ParentId: randomValue(1000000),
        PostTypeId: randomValue(10),
        Score: randomValue(100),
        Tags: new ObjectId().str.substr(0, 300), // Random string, max
300 chars
        Title: new ObjectId().str.substr(0, 500), // Random string, max
500 chars
        ViewCount: randomValue(1000000)
    });
}
```

b) UPDATE SQL - 500000 Posts

Sql

```
UPDATE    POSTS
SET         lastEditorUserId = 11
WHERE    lastEditorUserId IS NULL
   AND     ownerUserId = 11
   AND     id < 0
```

Mongo

```
UPDATE POSTS SET lastEditorUserId = 11 WHERE lastEditorUserId IS NULL
AND ownerUserId = 11 AND id < 0;
```

c) DELETE SQL - 500000 Posts

Sql

```
DELETE POSTS WHERE id < 0
```

Mongo

```
DELETE FROM POSTS WHERE id < 0;
```

1) Consultas Simples de Selección

a) Seleccionar Todos los Registros en Posts

Sql

```
SELECT * FROM Posts;
```

Mongo

```
db.posts.find({});
```

b) Seleccionar con filtro

Sql

```
SELECT * FROM Posts WHERE PostTypeId = 1;
```

Mongo

```
db.posts.find({ PostTypeId: 1 });
```

c) Seleccionar Posts con Alta Puntuación

Sql

```
SELECT * FROM Posts WHERE Score > 100;
```

Mongo

```
db.posts.find({ Score: { $gt: 100 } });
```

d) Seleccionar Posts por Usuario Específico

Sql

```
SELECT * FROM Posts WHERE OwnerUserId = 42;
```

Mongo

```
db.posts.find({ OwnerUserId: 42 });
```

2) Consultas de Agregación

a) Contar el Número de Posts por Usuario

Sql

```
SELECT OwnerUserId, COUNT(*) FROM Posts GROUP BY OwnerUserId;
```

Mongo

```
db.posts.aggregate([ { $group: { _id: "$OwnerUserId", count: { $sum: 1 }
} } ]);
```

b) Contar el Número de Posts por Tipo con Filtro

Sql

```
SELECT PostTypeId, COUNT(*)
FROM Posts
GROUP BY PostTypeId
HAVING COUNT(*) > 1000;
```

Mongo

```
db.posts.aggregate([ { $group: { _id: "$PostTypeId", count: { $sum: 1 }
} }, { $match: { count: { $gt: 1000 } } } ]);
```

c) Promedio de Puntuaciones por Tipo de Post

Sql

```
SELECT PostTypeId, AVG(Score) AS AvgScore FROM Posts GROUP BY
PostTypeId;
```

Mongo

```
db.posts.aggregate([ { $group: { _id: "$PostTypeId", avgScore: { $avg:
"$Score" } } } ]);
```

d) Máxima Puntuación de Posts por Usuario

Sql

```
SELECT OwnerUserId, MAX(Score) AS MaxScore
FROM Posts
GROUP BY OwnerUserId;
```

Mongo

```
db.posts.aggregate([ { $group: { _id: "$OwnerUserId", maxScore: { $max:
"$Score" } } } ]);
```

3) Consultas de Unión (Joins)

a) Inner Join entre Posts y Usuarios

Sql

```
SELECT p.Title, u.DisplayName
FROM Posts p
INNER JOIN Users u ON p.OwnerUserId = u.Id
WHERE u.id = 11
```

Mongo

```
db.posts.aggregate([ { $match: { OwnerUserId: 11 } }, { $lookup: { from:
"users", localField: "OwnerUserId", foreignField: "_id", as: "owner" }
}, { $project: { Title: 1, "owner.DisplayName": 1 } } ]);
```

b) Join entre Posts, PostTypes y Usuarios

Sql

```
SELECT p.Title, pt.[Type], u.DisplayName
FROM Posts p
INNER JOIN PostTypes pt
ON p.PostTypeId = pt.Id
INNER JOIN Users u ON p.OwnerUserId = u.Id
WHERE p.id > 20000000
```

Mongo

```
db.posts.aggregate([ { $match: { _id: { $gt: 20000000 } } }, { $lookup:
{ from: "postTypes", localField: "PostTypeId", foreignField: "_id", as:
"postType" } }, { $unwind: "$postType" }, { $lookup: { from: "users",
localField: "OwnerUserId", foreignField: "_id", as: "owner" } }, {
$unwind: "$owner" }, { $project: { Title: 1, Type: "$postType.Type",
DisplayName: "$owner.DisplayName" } } ]);
```

4) Consultas Complejas

a) Subconsulta: Posts con Puntuación Mayor al Promedio

Sql

```
SELECT * FROM Posts WHERE Score > (SELECT AVG(Score) FROM Posts);
```

Mongo

```
const avgScore = db.posts.aggregate([ { $group: { _id: null, avgScore: {
$avg: "$Score" } } } ]).toArray()[0].avgScore; db.posts.find({ Score: {
$gt: avgScore } });
```

5) Consultas de Utilización de Índices

a) Selección Basada en Índices en Posts

Sql

```
SELECT * FROM Posts WHERE Id = 123;
```

Mongo

```
db.posts.find({ _id: ObjectId("60d5f60f1e8a1a3c9a6f4e91") });
```

b) Escaneo de Índices en Posts

Sql

```
SELECT * FROM Posts WHERE Title LIKE 'How%';
```

Mongo

```
db.posts.find({ Title: /^How/ });
```

Scripts SQL:
https://github.com/marianocarreira/BaseDeDatos2023/tree/main/Scripts/SQL

Scripts MongoDb (groovy):
https://github.com/marianocarreira/BaseDeDatos2023/tree/main/Scripts/MongoDb

8vo Paso: Ejecución

El plan de ejecución se planifica de la siguiente manera:

- **Ejecución 1**: Base de datos con un total de **17142169 Posts**, **2465713 Users** y **8 PostTypes**

- **Ejecucion 2:** Base de datos con un total de **4095531 Posts**, **1364747 Users** y **8 PostTypes**

- **Ejecucion 3:** Base de datos con un total de **59421 Posts**, **870164 Users** y **8 PostTypes**

Dado que una gran cantidad de motores de base de datos utilizan el Cache para resultados, se planifica la prueba de la siguiente manera:

Create/Update/Delete -> Se ejecutan solo **1 vez cada uno al iniciar cada prueba** de cada motor.

Reads -> Las Lecturas se ejecutarán en un loop de 2, o sea, se repetirá el set de consultas para cada motor, de esta manera podremos evaluar el efecto o no del cache luego de la primera ronda.

En resumen: Se realizarán 3 pruebas con distinto tamaño de base de datos, repitiendo cada una 2 veces.

Documentación de Jmeter:
https://jmeter.apache.org/usermanual/index.html

Jmeter Test File :
https://github.com/marianocarreira/BaseDeDatos2023/tree/main/Jmeter

Screenshot de las pruebas configuradas en JMeter.

8vo Paso: Resultados

Tabla general de resultados (tiempos expresados en milisegundos):

# Query	Base de datos Grande				Base de datos Mediana				Base de datos Pequeña			
	Iteración 1		Iteración 2		Iteración 1		Iteración 2		Iteración 1		Iteración 2	
	SQL	Mongo	SQL	Mongo	SQL	Mongo	SQL	Mongo	SQL	Mongo	SQL	Mongo
0.a	**133390**	**97815**	**NA**	**NA**	**119811**	**71070**	**NA**	**NA**	**119237**	**85908**	**NA**	**NA**
0.b	**14632**	**178194**	**NA**	**NA**	**20064**	**66792**	**NA**	**NA**	**12829**	**28979**	**NA**	**NA**
0.c	**27556**	**72673**	**NA**	**NA**	**24629**	**49063**	**NA**	**NA**	**18875**	**28979**	**NA**	**NA**
1.a	297046	**94958**	279148	**71329**	61512	20745	55282	15273	747	222	520	170
1.b	173623	**34917**	162827	**33022**	31765	7500	28541	6671	289	95	196	65
1.c	**22904**	34530	**22796**	34295	6750	8607	6655	8322	109	73	52	52
1.d	35	**28**	**19**	28	37	80	9	33	65	29	1	22
2.a	**2770**	52235	**2767**	51996	697	10996	682	9231	123	104	20	47
2.b	**1643**	35945	**1539**	35931	355	8724	332	8631	31	66	12	41
2.c	**22801**	50811	**22755**	50497	9735	12628	6707	12266	81	68	11	50
2.d	**24259**	56770	**24068**	55916	8344	12146	7583	12138	108	96	29	51
3.a	**138**	10545	**133**	10422	180	4912	89	4583	9	57	2	39
3.b	**2004**	62866	**1899**	58897	14	114	3	65	26	31	1	19
4.a	114398	**63621**	111877	**56065**	30257	17132	27247	12960	307	31	138	101
5.a	43	**32**	37	**24**	54	58	11	31	15	30	0	20
5.b	**28197**	40142	**26071**	36171	7251	9885	7153	8060	89	82	53	53

Referencias:
Verde : *Tiempo mejor performance.*
Celeste: *Tiempo del segundo round de cada query mejor performance.*

Consideraciones adicionales:

Se estima que el insert de 500000 registros fue la operación más costosa dado que los motores tienen que re-crear índices, crear Id autogenerados, validar y actualizar foreign-keys, etc. Por otro lado los deletes y updates se realizaron filtrando por id, el cual es un índice de tipo cluster en SQL Server (se propone como pendiente para futuras investigaciones).
https://www.quora.com/What-are-the-benefits-of-using-an-update-statement-instead-of-delete-and-insert-statements-when-updating-tables-in-a-relational-database-system-like-MySQL-or-MS-SQL-Server?no_redirect=1

Puede asumirse que de igual manera podría pasar con deletes, dado que es necesario reorganizar índices, pero en muchos motores los deletes son soft en primera instancia y luego removidos físicamente en background.
https://www.sqlshack.com/the-ghost-cleanup-task-for-sql-server-databases/

El ordenamiento de mongo por defecto es no determinista, por tanto pueden haberse transmitido mayor cantidad de datos dado el contenido de los registros, cabe aclarar que no fue agregado el ordenamiento a los queries de Mongo para no comprometer la performance. Esta es otra de las diferencias de los motores, lo que hace que sea difícil su comparación.

Se observa que la cantidad de bytes devueltos por mongo incrementa con una pendiente mayor a la de sql server, esto puede deberse al formato JSON, el cual agrega caracteres especiales como llaves {}, corchetes [], doble comillas “”, etc; mientras que SQL Server solo devuelve el nombre de las columnas como header sin importar la cantidad de registros.

Se propone como futura investigación utilizar datasets con registros totalmente clónicos para evaluar la cantidad de datos enviados, devolviendo siempre registros iguales.
Por otro lado se propone evaluar el comportamiento del borrado de registros y causa de superioridad en la prueba en base de datos SQL contra MongoDB.

MongoDb y SQL Server son diferentes motores con capacidades diferentes, una comparativa puede ser compleja, o sesgada, dado que los dos poseen características diferentes para diferentes fines, mientras que MongoDb casi que ignora la redundancia de datos en pos de tolerancia a particiones y consistencia (baja o eventual), SQL Server y el concepto de normalización proponen un rígido control de normalización y baja redundancia, lo que garantiza consistencia.
Elegir el motor correcto dependerá del correspondiente análisis de trade-off según el proyecto a implementar.

Fig. Teorema CAP

Conclusión

En general, ambos motores de base de datos mostraron mejoras significativas en la performance cuando se ejecutó el mismo query por segunda vez consecutiva con la misma cantidad de registros. MongoDB presentó mejores resultados en consultas sobre una sola colección (equivalente a una tabla en SQL), mientras que SQL Server demostró una clara ventaja en consultas que requieren la unión de varias tablas (joins). Esta ventaja de SQL Server en los joins puede deberse a la naturaleza relacional del modelo de datos utilizado, el cual se adaptó directamente de SQL a MongoDB sin aprovechar completamente el modelo documental de MongoDB. Un diseño de datos en MongoDB que incluyera relaciones como agregaciones dentro de documentos podría haber mejorado notablemente su performance, posiblemente superando a SQL Server, aunque esto hubiera comprometido la validez de las pruebas de joins solicitadas por los tutores.

En términos de agregaciones, SQL Server también mostró una superioridad significativa en rendimiento, siendo hasta diez veces más rápido en algunos casos. Además, SQL Server hace un uso más eficiente de los índices, aunque la diferencia en este aspecto no es tan pronunciada.

Se observó que, a medida que se reduce el tamaño de la base de datos, la performance tiende a igualarse entre ambos motores. Esto sugiere que, con el aumento de la cantidad de datos, la diferencia en performance tiende a ser no lineal, excediendo la tasa de crecimiento de datos.

Finalmente, si se busca mantener un modelo de baja redundancia de datos, como el relacional utilizado en esta prueba, SQL Server supera a MongoDB en performance. Sin embargo, si la redundancia de datos no es una preocupación principal y se priorizan otros aspectos como la tolerancia a particiones y la alta disponibilidad, MongoDB, con un modelo documental, podría ofrecer mejores resultados de performance.

Referencias

https://www.dataversity.net/wide-column-database/#
https://databasetown.com/wide-column-database-use-cases/#disadvantages-of-widecolumn-databases-include
https://db-engines.com/
https://www.sqlshack.com/the-ghost-cleanup-task-for-sql-server-databases/
https://www.quora.com/What-are-the-benefits-of-using-an-update-statement-instead-of-delete-and-insert-statements-when-updating-tables-in-a-relational-database-system-like-MySQL-or-MS-SQL-Server?no_redirect=1
https://jmeter.apache.org
https://www.whatisfileextension.com/es/mdf/
https://hevodata.com/learn/mongodb-vs-sql-server/

Anexo 1 - Gráficos

Los resultados se esperan en los siguientes formatos/gráficos:

Summary Report:
https://jmeter.apache.org/usermanual/component_reference.html#Summary_Report

View Results Tree:
https://jmeter.apache.org/usermanual/component_reference.html#View_Results_Tree

Response Time Graph:
https://jmeter.apache.org/usermanual/component_reference.html#Response_Time_Graph

Aggregate Report:
https://jmeter.apache.org/usermanual/component_reference.html#Aggregate_Report

Aggregate Graph:
https://jmeter.apache.org/usermanual/component_reference.html#Aggregate_Graph

1) Gráficos para base de datos con **17142169 Posts**, **2465713 Users** y **8 PostTypes**

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Selects:INSERT SQL - 500000 Posts	1	133390	133390	133390	0.00	0.00%	27.0/hour	0.00	0.00	9.0
Selects:UPDATE SQL - 500000 Posts	1	14632	14632	14632	0.00	0.00%	4.1/min	0.00	0.00	14.0
Selects:DELETE SQL - 500000 Posts	1	27556	27556	27556	0.00	0.00%	2.2/min	0.00	0.00	16.0
Selects:INSERT MONGO - 500000 Posts	1	97815	97815	97815	0.00	0.00%	36.8/hour	0.00	0.00	.0
Selects:UPDATE MONGO - 500000 Posts	1	178194	178194	178194	0.00	0.00%	20.2/hour	0.00	0.00	84.0
Selects:DELETE MONGO - 500000 Posts	1	72673	72673	72673	0.00	0.00%	49.5/hour	0.00	0.00	45.0
TOTAL	6	87376	14632	178194	57097.10	0.00%	41.2/hour	0.00	0.00	28.0

Aggregate Graph
Milliseconds
133,360
14,632
27,558
INSERT SQL - 5000 ...
UPDATE SQL - 5000 ...
DELETE SQL - 5000 ...
Average

Response Time Graph
Milliseconds
SQL 1.A (Consultas Simples-Todos los Posts)
SQL 1.B (Consultas Simples-Seleccion con Filtro)
SQL 1.C (Consultas Simples-Posts Alta Puntuacion)
SQL 1.D (Consultas Simples-Posts Usuario Especifico)
MONGO 1.A (Consultas Simples-Todos los Posts)
MONGO 1.B (Consultas Simples-Seleccionar con Filtro)
MONGO 1.C (Consultas Simples-Posts Alta Puntuacion)
MONGO 1.D (Consultas Simples-Posts Usuario Especifico)

Response Time Graph
Milliseconds
SQL 2.A (Consultas de Agregacion-Núm. de Posts por Usuario)
SQL 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
SQL 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
SQL 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)
MONGO 2.A (Consultas de Agregacion-Num. de Posts por Usuario)
MONGO 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
MONGO 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
MONGO 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)

Response Time Graph
Milliseconds
SQL 3.A (Consultas de Unión (Joins)-Inner Join entre Posts y Usuarios)
SQL 3.B (Consultas de Unión (Joins)-Join entre Posts, PostTypes y Usuarios)
MONGO 3.A (Consultas de Unión (Joins)-Inner Join entre Posts y Usuarios)
MONGO 3.B (Consultas de Unión (Joins)-Join entre Posts, PostTypes y Usuarios)

Response Time Graph
Milliseconds
SQL 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)
MONGO 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)

Response Time Graph
Milliseconds
SQL 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
SQL 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)
MONGO 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
MONGO 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)

2) Gráficos para base de datos con **4095531 Posts**, **2465713 Users** y **8 PostTypes**

Selects:INSERT SQL - 500000 Posts	1	119811	119811	119811	0.00	0.00%	30.0/hour	0.00	0.00	9.0
Selects:UPDATE SQL - 500000 Posts	1	20064	20064	20064	0.00	0.00%	3.0/min	0.00	0.00	14.0
Selects:DELETE SQL - 500000 Posts	1	24629	24629	24629	0.00	0.00%	2.4/min	0.00	0.00	16.0
Selects:INSERT MONGO - 500000 Posts	1	71070	71070	71070	0.00	0.00%	50.7/hour	0.00	0.00	.0
Selects:UPDATE MONGO - 500000 Posts	1	66792	66792	66792	0.00	0.00%	53.9/hour	0.00	0.00	84.0
Selects:DELETE MONGO - 500000 Posts	1	49063	49063	49063	0.00	0.00%	1.2/min	0.00	0.00	45.0
Selects:SQL 1.A (Consultas Simples-Todos los Posts)	2	58397	55282	61512	3115.00	0.00%	23.2/hour	0.00	0.00	216.5
Selects:SQL 1.B (Consultas Simples-Seleccion con Filtro)	2	30153	28541	31765	1612.00	0.00%	26.0/hour	0.00	0.00	216.0
Selects:SQL 1.C (Consultas Simples-Posts Alta Puntuacion)	2	6702	6655	6750	47.50	0.00%	28.6/hour	0.00	0.00	216.0
Selects:SQL 1.D (Consultas Simples-Posts Usuario Especifico)	2	23	9	37	14.00	0.00%	29.4/hour	0.00	0.00	215.5
Selects:MONGO 1.A (Consultas Simples-Todos los Posts)	2	18009	15273	20745	2736.00	0.00%	27.1/hour	0.00	0.00	.0
Selects:MONGO 1.B (Consultas Simples-Seleccionar con Filtro)	2	7085	6671	7500	414.50	0.00%	27.9/hour	0.00	0.00	.0
Selects:MONGO 1.C (Consultas Simples-Posts Alta Puntuacion)	2	8464	8322	8607	142.50	0.00%	27.7/hour	0.00	0.00	.0
Selects:MONGO 1.D (Consultas Simples-Posts Usuario Especifico)	2	56	33	80	23.50	0.00%	28.6/hour	0.00	0.00	.0
Selects:SQL 2.A (Consultas de Agregacion-Num. de Posts por Us...	2	689	682	697	7.50	0.00%	28.5/hour	0.00	0.00	216.0
Selects:SQL 2.B (Consultas de Agregacion-Contar el Número de P...	2	343	332	355	11.50	0.00%	28.5/hour	0.00	0.00	216.5
Selects:SQL 2.C (Consultas de Agregacion-Promedio de Puntuaci...	2	8221	6707	9735	1514.00	0.00%	27.5/hour	0.00	0.00	216.5
Selects:SQL 2.D (Consultas de Agregacion-Máxima Puntuación d...	2	7963	7583	8344	380.50	0.00%	27.4/hour	0.00	0.00	216.0
Selects:MONGO 2.A (Consultas de Agregacion-Num. de Posts po...	2	10113	9231	10996	882.50	0.00%	27.3/hour	0.00	0.00	.0
Selects:MONGO 2.B (Consultas de Agregacion-Contar el Número...	2	8677	8631	8724	46.50	0.00%	27.6/hour	0.00	0.00	.0
Selects:MONGO 2.C (Consultas de Agregacion-Promedio de Pun...	2	12447	12266	12628	181.00	0.00%	27.2/hour	0.00	0.00	.0
Selects:MONGO 2.D (Consultas de Agregacion-Máxima Puntuaci...	2	12142	12138	12146	4.00	0.00%	27.3/hour	0.00	0.00	.0
Selects:SQL 3.A (Consultas de Unión (Joins) -Inner Join entre Post...	2	134	89	180	45.50	0.00%	28.5/hour	0.00	0.00	217.0
Selects:SQL 3.B (Consultas de Unión (Joins) -Join entre Posts, Pos...	2	8	3	14	5.50	0.00%	28.5/hour	0.00	0.00	217.0
Selects:MONGO 3.A (Consultas de Unión (Joins) -Inner Join entre ...	2	4747	4583	4912	164.50	0.00%	28.0/hour	0.00	0.00	6.0
Selects:MONGO 3.B (Consultas de Unión (Joins) -Join entre Posts,...	2	89	65	114	24.50	0.00%	28.5/hour	0.00	0.00	1.0
Selects:SQL 4.A (Consultas Complejas-Subconsulta: Posts con Pu...	2	28752	27247	30257	1505.00	0.00%	25.7/hour	0.00	0.00	217.0
Selects:MONGO 4.A (Consultas Complejas-Subconsulta: Posts co...	2	15046	12960	17132	2086.00	0.00%	27.0/hour	0.00	0.00	6.0
Selects:SQL 5.A (Consultas de Utilización de Índices-Selección Ba...	2	32	11	54	21.50	0.00%	28.4/hour	0.00	0.00	216.5
Selects:SQL 5.B (Consultas de Utilización de Índices-Escaneo de I...	2	7202	7153	7251	49.00	0.00%	27.6/hour	0.00	0.00	217.0
Selects:MONGO 5.A (Consultas de Utilización de Índices-Selecció...	2	44	31	58	13.50	0.00%	28.4/hour	0.00	0.00	1.0
Selects:MONGO 5.B (Consultas de Utilización de Índices-Escaneo ...	2	8972	8060	9885	912.50	0.00%	27.3/hour	0.00	0.00	6.0
TOTAL	58	14835	3	119811	21908.56	0.00%	4.0/min	0.01	0.00	100.6

Aggregate Graph
Milliseconds
71,076
68,702
49,063
INSERT MONGO - 50...
UPDATE MONGO - 50...
DELETE MONGO - 50...
Average

Response Time Graph
Milliseconds
68,000
64,000
60,000
56,000
52,000
48,000
44,000
40,000
36,000
32,000
28,000
24,000
20,000
16,000
12,000
8,000
4,000
0
22:05:40
22:05:50
22:06:00
22:06:10
22:06:20
22:06:30
22:06:40
22:06:50
22:07:00
22:07:10
22:07:20
22:07:30
22:07:40
22:07:50
22:08:00
22:08:10
22:08:20
22:08:30
22:08:40
22:08:50
22:09:00
22:09:10
22:09:20
22:09:30
22:09:40
22:09:50
22:10:00
22:10:10
22:10:20
22:10:30
22:10:40
22:10:50
22:11:00
22:11:10
22:11:20
22:11:30
22:11:40
22:11:50
22:12:00
SQL 1.A (Consultas Simples-Todos los Posts)
SQL 1.B (Consultas Simples-Seleccion con Filtro)
SQL 1.C (Consultas Simples-Posts Alta Puntuacion)
SQL 1.D (Consultas Simples-Posts Usuario Especifico)
MONGO 1.A (Consultas Simples-Todos los Posts)
MONGO 1.B (Consultas Simples-Seleccionar con Filtro)
MONGO 1.C (Consultas Simples-Posts Alta Puntuacion)
MONGO 1.D (Consultas Simples-Posts Usuario Especifico)

Response Time Graph
Milliseconds
SQL 2.A (Consultas de Agregacion-Num. de Posts por Usuario)
SQL 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
SQL 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
SQL 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)
MONGO 2.A (Consultas de Agregacion-Num. de Posts por Usuario)
MONGO 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
MONGO 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
MONGO 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)

Response Time Graph
Milliseconds
SQL 3.A (Consultas de Unión (Joins)-Inner Join entre Posts y Usuarios)
SQL 3.B (Consultas de Unión (Joins)-Join entre Posts, PostTypes y Usuarios)
MONGO 3.A (Consultas de Unión (Joins)-Inner Join entre Posts y Usuarios)
MONGO 3.B (Consultas de Unión (Joins)-Join entre Posts, PostTypes y Usuarios)

Response Time Graph
Milliseconds
SQL 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)
MONGO 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)

Response Time Graph
Milliseconds
SQL 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
SQL 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)
MONGO 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
MONGO 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)

3) Gráficos para base de datos con **59421 Posts**, **2465713 Users** y **8 PostTypes**

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename Browse...

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec
Selects:UPDATE SQL - 500000 Posts	1	12829	12829	12829	0.00	0.00%	4.7/min	0.
Selects:UPDATE MONGO - 500000 Posts	1	28979	28979	28979	0.00	0.00%	2.1/min	0.
Selects:SQL 5.B (Consultas de Utilización de Índices...	8	73	53	89	13.19	0.00%	50.7/hour	0.
Selects:SQL 5.A (Consultas de Utilización de Índices...	8	4	0	15	4.77	0.00%	50.7/hour	0.
Selects:SQL 4.A (Consultas Complejas-Subconsulta...	8	222	138	307	50.27	0.00%	50.7/hour	0.
Selects:SQL 3.B (Consultas de Unión (Joins) -Join e...	8	6	1	26	7.85	0.00%	50.7/hour	0.
Selects:SQL 3.A (Consultas de Unión (Joins) -Inner J...	8	5	2	9	2.76	0.00%	50.7/hour	0.
Selects:SQL 2.D (Consultas de Agregacion-Máxima...	8	71	29	108	26.24	0.00%	50.7/hour	0.
Selects:SQL 2.C (Consultas de Agregacion-Promedi...	8	42	11	81	21.69	0.00%	50.7/hour	0.
Selects:SQL 2.B (Consultas de Agregacion-Contar e...	8	19	12	31	5.98	0.00%	50.7/hour	0.
Selects:SQL 2.A (Consultas de Agregacion-Num. d...	8	38	20	123	32.30	0.00%	50.7/hour	0.
Selects:SQL 1.D (Consultas Simples-Posts Usuario E...	8	12	1	65	20.19	0.00%	50.7/hour	0.
Selects:SQL 1.C (Consultas Simples-Posts Alta Punt...	8	79	52	109	16.41	0.00%	50.7/hour	0.
Selects:SQL 1.B (Consultas Simples-Seleccion con F...	8	217	196	289	28.19	0.00%	50.7/hour	0.
Selects:SQL 1.A (Consultas Simples-Todos los Posts)	8	650	520	747	77.50	0.00%	50.6/hour	0.
Selects:MONGO 5.B (Consultas de Utilización de Ín...	8	63	53	82	8.91	0.00%	50.7/hour	0.
Selects:MONGO 5.A (Consultas de Utilización de Ín...	8	23	20	30	2.87	0.00%	50.7/hour	0.
Selects:MONGO 4.A (Consultas Complejas-Subcon...	8	151	101	230	36.73	0.00%	50.7/hour	0.
Selects:MONGO 3.B (Consultas de Unión (Joins) -J...	8	25	19	31	3.77	0.00%	50.7/hour	0.
Selects:MONGO 3.A (Consultas de Unión (Joins) -I...	8	47	39	57	6.38	0.00%	50.7/hour	0.
Selects:MONGO 2.D (Consultas de Agregacion-Má...	8	61	51	96	13.99	0.00%	50.7/hour	0.
Selects:MONGO 2.C (Consultas de Agregacion-Pro...	8	55	50	68	6.14	0.00%	50.7/hour	0.
Selects:MONGO 2.B (Consultas de Agregacion-Co...	8	49	41	66	7.56	0.00%	50.7/hour	0.
Selects:MONGO 2.A (Consultas de Agregacion-Nu...	8	62	47	104	18.25	0.00%	50.7/hour	0.
Selects:MONGO 1.D (Consultas Simples-Posts Usu...	8	24	22	29	1.94	0.00%	50.7/hour	0.
Selects:MONGO 1.C (Consultas Simples-Posts Alta ...	8	58	52	73	6.34	0.00%	50.7/hour	0.
Selects:MONGO 1.B (Consultas Simples-Selecciona...	8	71	65	95	9.35	0.00%	50.7/hour	0.
Selects:MONGO 1.A (Consultas Simples-Todos los ...	8	196	170	222	20.73	0.00%	50.7/hour	0.
Selects:INSERT SQL - 500000 Posts	1	119237	119237	119237	0.00	0.00%	30.2/hour	0.
Selects:INSERT MONGO - 500000 Posts	1	85908	85908	85908	0.00	0.00%	41.9/hour	0.
Selects:DELETE SQL - 500000 Posts	1	18875	18875	18875	0.00	0.00%	3.2/min	0.
Selects:DELETE MONGO - 500000 Posts	1	38151	38151	38151	0.00	0.00%	1.6/min	0.
TOTAL	214	1507	0	119237	10575.17	0.00%	22.5/min	0.

Aggregate Graph
Milliseconds
71.070
66.702
49.033
INSERT MONGO - 50 ...
UPDATE MONGO - 50 ...
DELETE MONGO - 50 ...
Average

Response Time Graph
Milliseconds
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
23:15:30
23:15:40
23:15:50
23:16:00
23:16:10
23:16:20
23:16:30
23:16:40
23:16:50
23:17:00
23:17:10
23:17:20
23:17:30
23:17:40
23:17:50
23:18:00
23:18:10
23:18:20
23:18:30
23:18:40
SQL 1.A (Consultas Simples-Todos los Posts)
SQL 1.B (Consultas Simples-Seleccion con Filtro)
SQL 1.C (Consultas Simples-Posts Alta Puntuacion)
SQL 1.D (Consultas Simples-Posts Usuario Especifico)
MONGO 1.A (Consultas Simples-Todos los Po
MONGO 1.B (Consultas Simples-Seleccionar con Filtro)
MONGO 1.C (Consultas Simples-Posts Alta Puntuacion)
MONGO 1.D (Consultas Simples-Posts Usuario Especifico)

Response Time Graph
Milliseconds
SQL 2.A (Consultas de Agregacion-Num. de Posts por Usuario)
SQL 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
SQL 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
SQL 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)
MONGO 2.A (Consultas de Agregacion-Num. de Posts por Usuario)
MONGO 2.B (Consultas de Agregacion-Contar el Número de Posts por Tipo con Filtro)
MONGO 2.C (Consultas de Agregacion-Promedio de Puntuaciones por Tipo de Post)
MONGO 2.D (Consultas de Agregacion-Máxima Puntuación de Posts por Usuario)

Response Time Graph
Milliseconds
SQL 3.A (Consultas de Unión (Joins) -Inner Join entre Posts y Usuarios)
SQL 3.B (Consultas de Unión (Joins) -Join entre Posts, PostTypes y Usuarios)
MONGO 3.A (Consultas de Unión (Joins) -Inner Join entre Posts y Usuarios)
MONGO 3.B (Consultas de Unión (Joins) -Join entre Posts, PostTypes y Usuarios)

Response Time Graph
Milliseconds
SQL 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)
MONGO 4.A (Consultas Complejas-Subconsulta: Posts con Puntuación Mayor al Promedio)

Response Time Graph
Milliseconds
SQL 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
SQL 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)
MONGO 5.A (Consultas de Utilización de Índices-Selección Basada en Índices en Posts)
MONGO 5.B (Consultas de Utilización de Índices-Escaneo de Índices en Posts)