



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

Maestría en Ingeniería de Software

# Base de Datos

Conceptos y comparaciones

Autor:

**Lic. Carreira, Mariano**

Docentes:

**Dr. Grigera, Julian**

**Lic. Diclaudio, Federico**

Tipos de Datos.....	3
Estructurados.....	3
Semi-Estructurados.....	3
No Estructurados.....	3
<b>Base de datos.....</b>	<b>4</b>
Definición.....	4
<b>Sistema de gestión de base de datos (SGBD).....</b>	<b>4</b>
Definición.....	4
Tipos.....	4
SGBD Relacional.....	5
Ejemplos más populares:.....	5
NoSQL.....	6
Clasificación de SGBD NoSQL.....	6
Ventajas.....	6
Métodos.....	7
<b>Comparativo NoSQL.....</b>	<b>7</b>
<b>Benchmarking.....</b>	<b>16</b>
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.....	21
Fig. Subset de tablas seleccionado para la prueba.....	24
Prueba.....	24
1er Paso: Instalación de los motores y clientes.....	24
2do Paso: Instalación de JMeter.....	26
3er Paso: Descarga del Dataset.....	30
5to Paso: Creación de la base de datos en MongoDB.....	31
6to Paso: Configuración de las pruebas, queries y resultados.....	32
7mo Paso: Ejecución.....	33
8vo Paso: Resultados.....	34
Conclusión.....	37
<b>Referencias.....</b>	<b>39</b>

# 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](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 (el llamado sistema de gestión de bases de datos o SGBD). La base de datos y el SGBD juntos forman el sistema de base de datos. 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

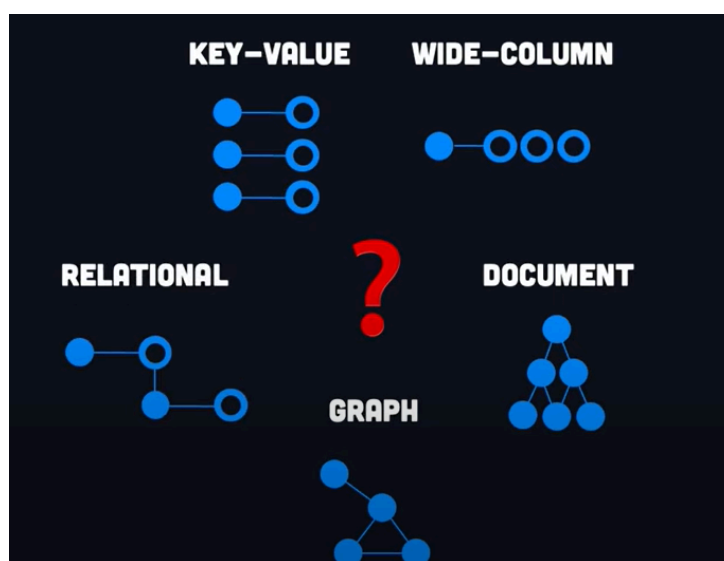
Un sistema de gestión de bases de datos (SGBD) es un software que se utiliza para gestionar una colección de datos lógicamente relacionados (base de datos). Un SGBD forma junto con una base de datos el sistema de base de datos.

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

### Tipos

Un DBMS que admite el modelo de datos relacionales a menudo se denomina RDBMS. 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 (RDBMS) 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 DBMS más utilizado. A lo largo de los años, muchos RDBMS 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 DBMS 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 DBMS 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                            | ❖ Search Engines    |
| ❖ Wide Column Stores                          | ❖ Blockchain        |
| ❖ Document Stores                             | ❖ Spatial           |
| ❖ Graph DBMS                                  | ❖ Time Series       |
| ❖ RDF Stores (resource description framework) | ❖ Vector            |
| ❖ Native XML DBMS                             | ❖ Object Oriented   |
| ❖ Content Stores                              | ❖ Event Stores      |
| ❖ Multivalued DBMS                            | ❖ 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  | logrando así escalabilidad y tolerancia a fallas                         |
| ❖ Fácil distribución de datos en diferentes nodos (por ejemplo, fragmentación), | ❖ Mayor flexibilidad mediante el uso de un modelo de datos sin esquemas. |
|   | ❖ Administración más sencilla.   |

## Métodos

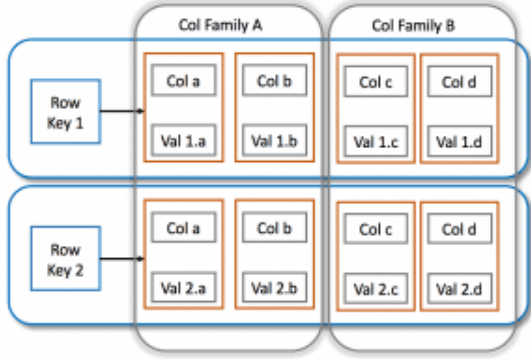
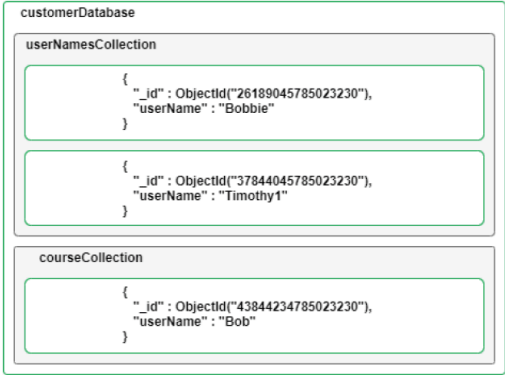
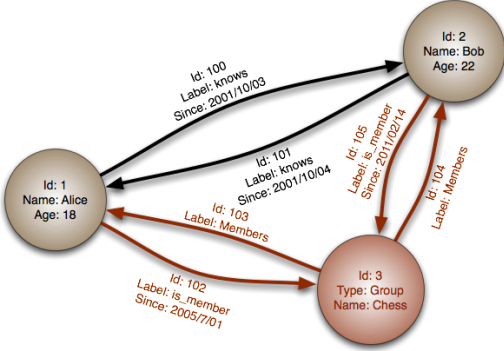
Estas ventajas se logran mediante uno o más de los siguientes enfoques:

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

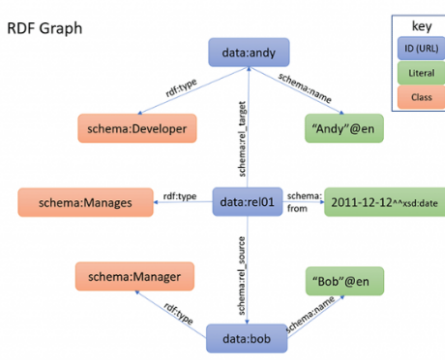
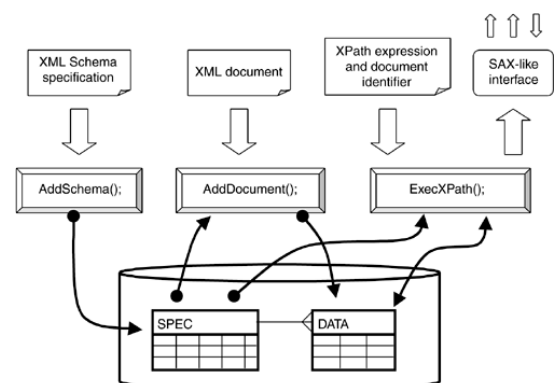
<https://db-engines.com/en/article/NoSQL>

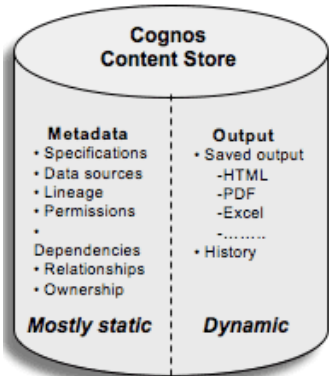
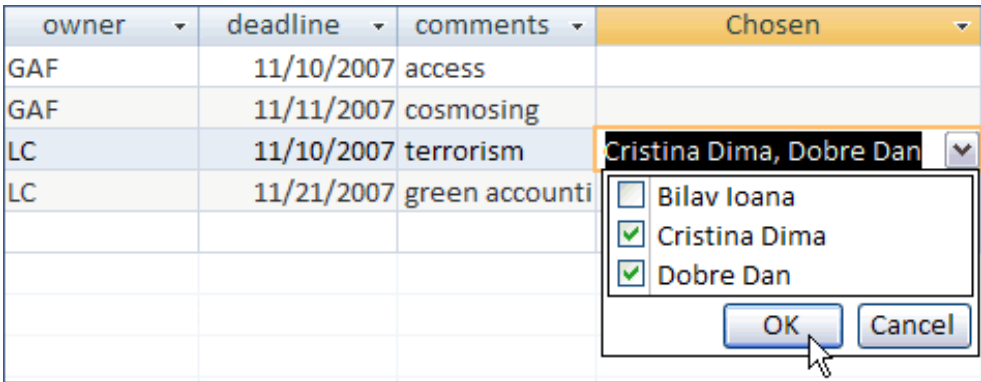
## Comparativo NoSQL

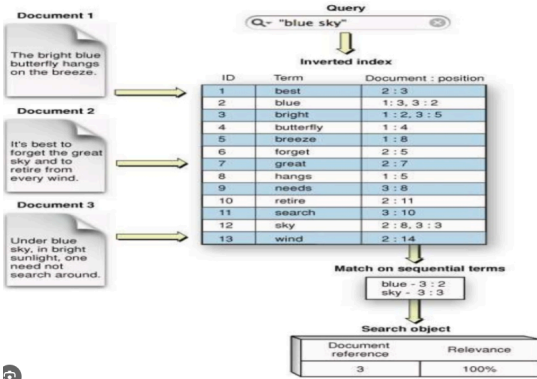

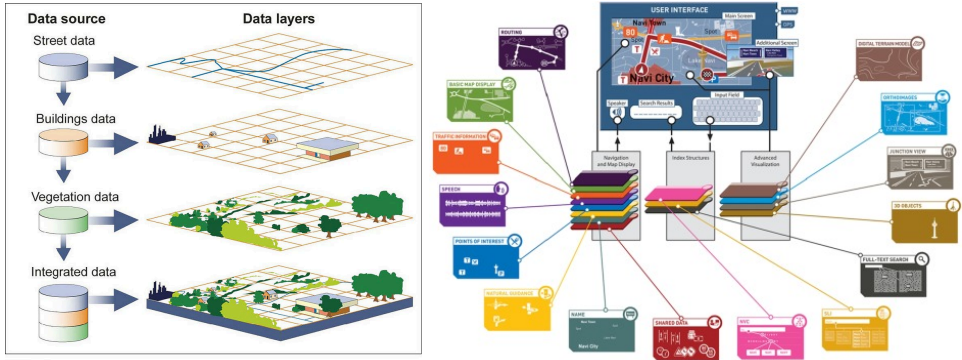
Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
Key-Value	<ul style="list-style-type: none"> <li>• Persiste pares Clave/Valor.</li> </ul>	<ul style="list-style-type: none"> <li>• Simplicidad.</li> <li>• Acceso rápido vía clave.</li> </ul>	<ul style="list-style-type: none"> <li>• No adecuada para aplicaciones complejas.</li> <li>• Capacidad de query limitada.</li> </ul>	<ul style="list-style-type: none"> <li>• Cache.</li> <li>• Datos de sesión.</li> </ul>	<ul style="list-style-type: none"> <li>• Redis.</li> <li>• Memcache.</li> </ul>
	<div> <div>Keys</div> <div>Values</div> <pre> graph LR     K1[2398239] --&gt; V1["{ 'name': 'Michal', 'Age': '31' }"]     K2[2398240] --&gt; V2["'Lorem ipsum dolor sit amet'"]     K3[2398241] --&gt; V3["{ 'name': 'Marlon Brando', 'Profession': 'Actor' }"]     K4[2398242] --&gt; V4["42"]                     </pre> </div>				
Wide Column	<ul style="list-style-type: none"> <li>• Persiste datos en columnas dinámicas.</li> <li>• No estructurada.</li> <li>• Mapeo multidimensional.</li> </ul>	<ul style="list-style-type: none"> <li>• Performance en queries y escritura.</li> <li>• Alta escalabilidad.</li> <li>• Flexibilidad del modelo.</li> </ul>	<ul style="list-style-type: none"> <li>• Capacidad limitada de queries.</li> <li>• Modelado limitado.</li> <li>• Soporte limitado. ACID</li> </ul>	<ul style="list-style-type: none"> <li>• Logging.</li> <li>• IoT.</li> <li>• Análisis en tiempo real.</li> <li>• Preferencias de usuario.</li> <li>• Alto volumen de datos.</li> <li>• Big Data.</li> <li>• Warehousing.</li> </ul>	<ul style="list-style-type: none"> <li>• Cassandra.</li> <li>• HBase.</li> <li>• BigTable.</li> </ul>

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
					
Document Stores	<ul style="list-style-type: none"> <li>• Información documental.</li> <li>• Libre de Schema.</li> <li>• Anidación de documentos.</li> <li>• Generalmente JSON.</li> </ul>	<ul style="list-style-type: none"> <li>• Alta escalabilidad.</li> <li>• Flexibilidad del modelo.</li> </ul>	<ul style="list-style-type: none"> <li>• Consistencia eventual.</li> <li>• Falta de estandarización.</li> <li>• Sin Indexación secundaria.</li> </ul>	<ul style="list-style-type: none"> <li>• BigData.</li> <li>• Análisis Tiempo Real.</li> </ul>	<ul style="list-style-type: none"> <li>• MongoDB.</li> <li>• DynamoDB.</li> </ul>
					
Grafos	<ul style="list-style-type: none"> <li>• Compuesta por nodos y vértices interconectados.</li> </ul>	<ul style="list-style-type: none"> <li>• Queries optimizados para relaciones.</li> </ul>	<ul style="list-style-type: none"> <li>• No permite el acceso directo a nodos según sus atributos.</li> </ul>	<ul style="list-style-type: none"> <li>• Redes sociales.</li> <li>• Sistemas de recomendación.</li> </ul>	<ul style="list-style-type: none"> <li>• Neo4J.</li> <li>• Virtuoso.</li> </ul>
					
RDF	<ul style="list-style-type: none"> <li>• Representación de la inf.en forma de</li> </ul>	<ul style="list-style-type: none"> <li>• Queries semánticas.</li> </ul>	<ul style="list-style-type: none"> <li>• Poca madurez.</li> </ul>	<ul style="list-style-type: none"> <li>• Web Semántica</li> </ul>	<ul style="list-style-type: none"> <li>• MarkLogic.</li> </ul>



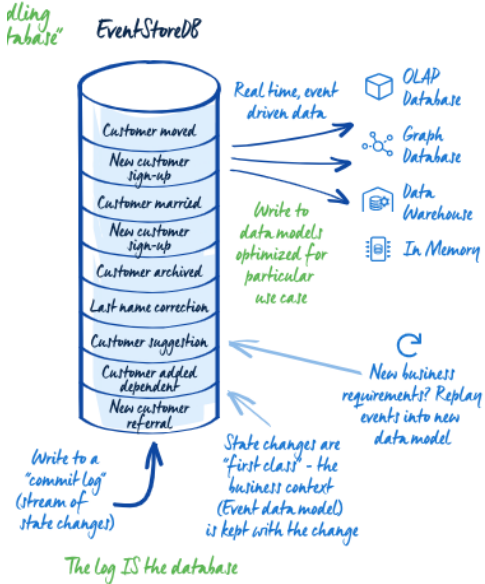
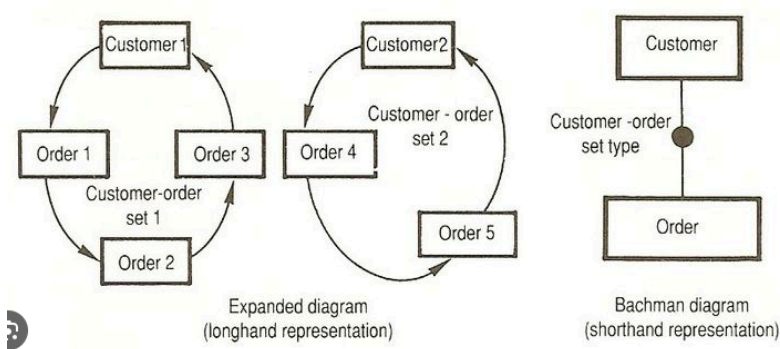
Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
	<p>tripletas ( Sujeto, Predicado, Objeto).</p> <ul style="list-style-type: none"> <li>Sub-tipo de SGBD Grafos.</li> </ul>	<ul style="list-style-type: none"> <li>Interoperabilidad.</li> </ul>	<ul style="list-style-type: none"> <li>Alta curva de aprendizaje.</li> </ul>	<ul style="list-style-type: none"> <li>Relación de datos.</li> </ul>	<ul style="list-style-type: none"> <li>Virtuoso.</li> </ul>
	<p>RDF Graph</p> 				
<b>XML Nativo</b>	<ul style="list-style-type: none"> <li>Formato XML.</li> <li>Soporte para XQuery, XPath y XSLT</li> </ul>	<ul style="list-style-type: none"> <li>Soporte a metadatos.</li> <li>Flexibilidad del modelo.</li> </ul>	<ul style="list-style-type: none"> <li>Complejidad y baja performance para queries complejos.</li> <li>Adopción limitada.</li> </ul>	<ul style="list-style-type: none"> <li>WebServices.</li> <li>Gestión documental.</li> </ul>	<ul style="list-style-type: none"> <li>MarkLogic</li> <li>Virtuoso</li> </ul>
					
<b>Content Stores</b>	<p>Administrar contenido digital.</p> <p>Datos no estructurados.</p> <p>Rico en metadata.</p>	<p>Soporte a versionado.</p> <p>Flexibilidad de datos.</p> <p>Soporte a metadatos.</p> <p>Control de acceso.</p>	<p>Problemas de escalabilidad.</p> <p>Migración compleja.</p>	<p>Administración de activos digitales.</p> <p>Gestión documental.</p>	<p>Jackrabbit</p> <p>ModeShape</p>

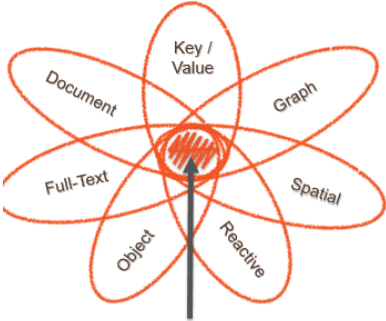
Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
					
<b>Multivalor</b>	<p>Similar a las relacionales, persiste en tablas.</p> <p>Puede asignar más de un valor a una columna. (Ej Array).</p> <p>Intermedio entre modelo relacional y documental.</p>	<p>Flexibilidad del modelo.</p> <p>Eficiente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

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 intersectar 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.

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo																							
	tiempo  Optimizado para manejar datos de series de tiempo: cada entrada está asociada con una marca de tiempo.	consulta de manera eficiente varias series temporales con altos volúmenes de transacciones	complejas,  Caso de uso especializado.	pueden ser producidos por sensores, medidores inteligentes o RFID en el llamado Internet de las Cosas.  Análisis del mercado financiero	Prometheus.  Kdb.																							
	<table><tr><th colspan="3">Key</th><th>Value</th></tr><tr><th>Metric name</th><th>Dimensions (labels)</th><th>Timestamp</th><th>Sample value</th></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>					Key			Value	Metric name	Dimensions (labels)	Timestamp	Sample value	http_requests_total	(status="200", method="GET")	@1112596200	987654	http_requests_total	(status="200", method="GET")	@1112596400	986575	http_requests_total	(status="404", method="GET")	@1112596600	946574	http_requests_total	(status="200", method="GET")	@1112596800
Key			Value																									
Metric name	Dimensions (labels)	Timestamp	Sample value																									
http_requests_total	(status="200", method="GET")	@1112596200	987654																									
http_requests_total	(status="200", method="GET")	@1112596400	986575																									
http_requests_total	(status="404", method="GET")	@1112596600	946574																									
http_requests_total	(status="200", method="GET")	@1112596800	976575																									
Vectores	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.	Alto rendimiento, escalabilidad y flexibilidad.	Casos de uso especializado.	Aprendizaje automático.  Minería de datos.  Sistemas de recomendaciones	Kdb.  Pinecone.  Chroma.																							
	Object Oriented	Almacenar los objetos en una	Soporte nativo a	Falta de estandarización	Sistemas con modelización	InterSystems IRIS.																						

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
	<p>base de datos de una manera que corresponda a su representación en un lenguaje de programación.</p> <p>Modelo de datos con clases (el esquema de objetos), propiedades y métodos.</p> <p>Lenguajes de consulta propios tipo SQL</p>	<p>orientación de objetos.</p> <p>Previene el uso de ORM y capas intermedias.</p>	<p>Desafíos de escalabilidad.</p>	<p>compleja.</p> <p>Dominios complejos.</p>	<p>Action NoSQL Database.</p> <p>ObjectStore.</p> <p>Db4o.</p>
<div> <div> <div>Object-Oriented Programming</div> <div>Polymorphism</div> <div>Inheritance</div> <div>Encapsulation</div> </div> </div>					
<b>Event Stores</b>	<p>Implementar el concepto de abastecimiento de eventos.</p> <p>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.</p> <p>El estado actual de un objeto se puede inferir reproduciendo todos los eventos de ese objeto desde el momento 0 hasta el momento actual.</p>	<p>Alta consistencia de datos.</p> <p>Recupero de información histórica.</p> <p>Alta escalabilidad.</p>	<p>No se admiten modificaciones o eliminaciones de eventos ya persistentes.</p> <p>Alto consumo de storage.</p>	<p>Logging.</p> <p>Sistemas financieros.</p> <p>IoT.</p>	<p>EventStore DB.</p> <p>IBM Db2 Event Store.</p>

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

Tipo	Características	Ventaja	Desventajas	Escenarios	Ejemplo
	 <p>Multi-Model represents the intersection of multiple models in just one product</p>				

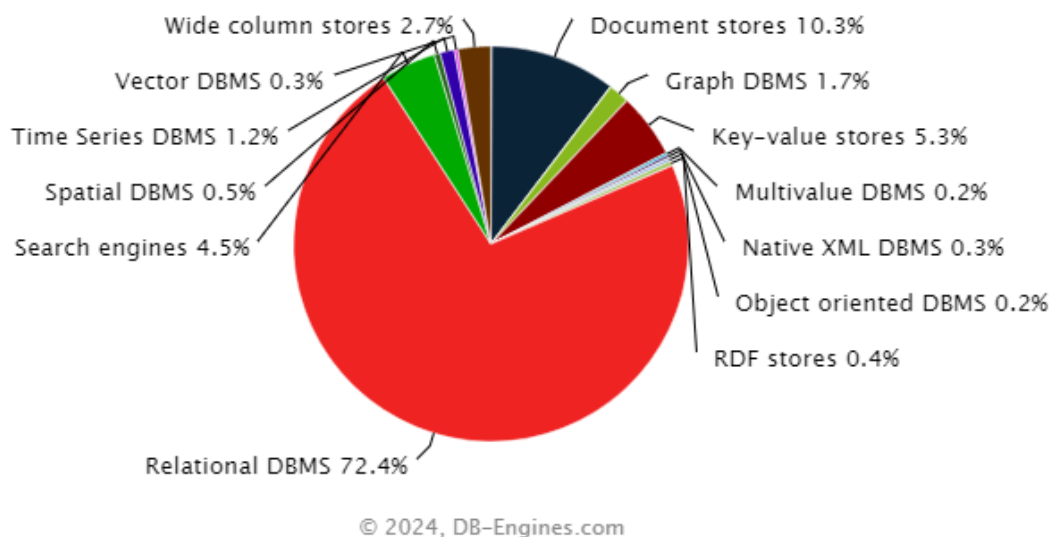
# Benchmarking

## Criterios de selección de motores

### Según el tipo

Como primer criterio de selección, se evalúa el tipo de base de datos, se optó 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 más 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.

### Ranking scores per category in percent, March 2024



[https://db-engines.com/en/ranking\\_categories](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.

Para conocer el método de puntaje de db-engines.com, remitirse a

[https://db-engines.com/en/ranking\\_definition](https://db-engines.com/en/ranking_definition)



### Según el modelo de implementación

Como segundo método de selección ya del motor específico, 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 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 poder 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 intermediarias 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 corren las bases de datos:

**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 para la realizar las pruebas, se tuvieron en cuenta y analizaron los siguientes aspectos 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://thegalead.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:

- Limitada 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 características:

- Tabla transaccionales largas, relacionadas con una o más tablas maestras.
- Más de 1 millón de registros.
- En lo posible datos reales, no moched.
- 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.

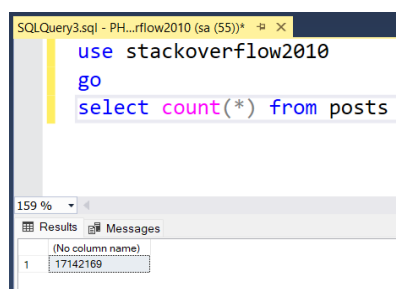
Contra:

- 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 a los **Posts** y las tablas maestras de **Usuarios** y **PostTypes**. La tabla principal de posts posee **17.142.169** registros, considerados suficientes para la prueba. Tamaño **50GB**.



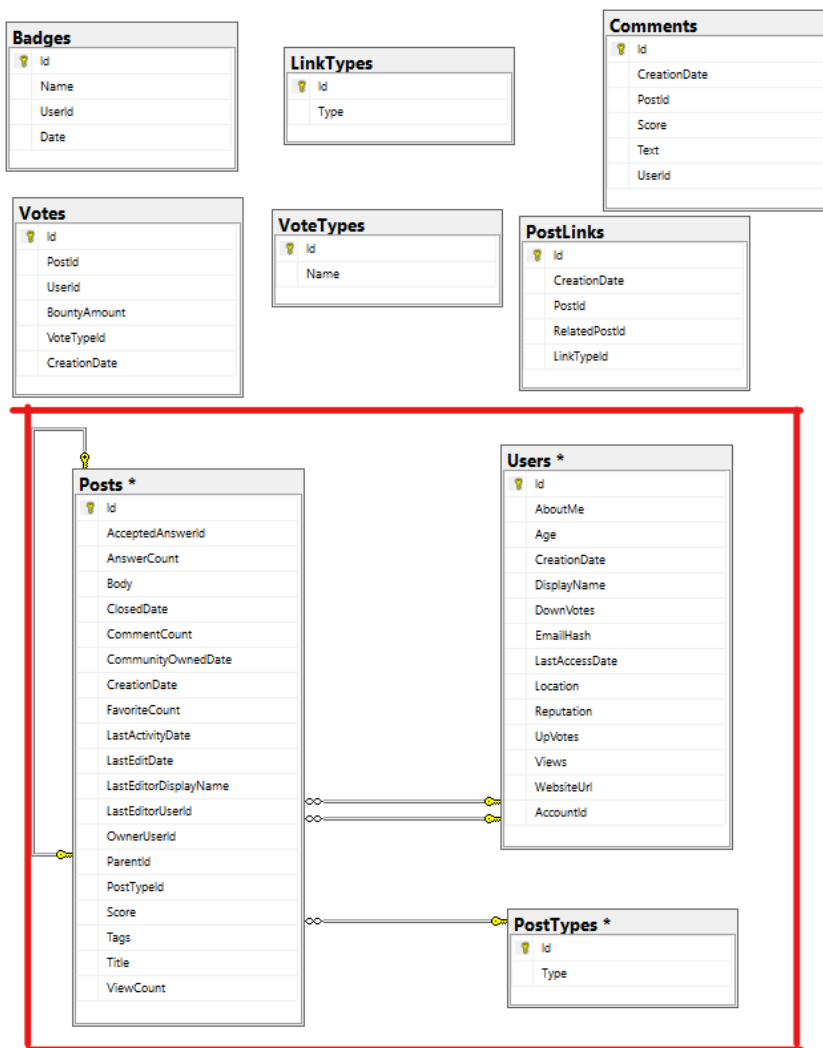


Fig. Subset de tablas seleccionado para la prueba.

## Comparación Estática

Aspecto	MongoDB	SQL Server
Developed by and Initial Release	Developed by MongoDB Inc. and initially released on February 11, 2009.	Developed by Microsoft Corporation and released initially on April 24, 1989.



Database Model	Non-Relational Database: Document-oriented (key-value structure)	Relational Database: Tables format
Implementation Language	JavaScript, Python, Java, PHP, C++, C, Ruby, Perl	C, C++
License	Open-Source	License required
Data Schema	Dynamic Schema	Fixed Schema
Query Language	MongoDB Query Language	SQL Query Language
Scalability	Horizontal	Vertical
Map Reduce	Supports Map Reduce method.	Does not support the Map-Reduce method.
Joins	No	Yes
Transaction	MongoDB provides Multi-document ACID transactions with snapshot isolation.	MS SQL Server provides ACID transactions without snapshot isolation.
XML support	No	Yes

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

## Prueba

### 1<sup>er</sup> 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>

## Benchmarking

microsoft.com/es-es/sql-server/sql-server-downloads

O bien, descarga una edición especializada gratuita.



**Developer**

SQL Server 2022 Developer es una edición gratuita con todas las características que se puede usar como base de datos de desarrollo y pruebas en un entorno que no sea de producción.

Descargar ahora



**Express**

SQL Server 2022 Express es una edición gratuita de SQL Server ideal para el desarrollo de aplicaciones de escritorio, aplicaciones web y pequeñas aplicaciones de servidor.

Descargar ahora

## Mongodb Community:

[https://fastdl.mongodb.org/windows/mongodb-windows-x86\\_64-7.0.7-signed.msi](https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-7.0.7-signed.msi)

mongodb.com/try/download/community

MongoDB. Products Resources Solutions Company Pricing

Search Support Sign In Try Free

MongoDB Atlas

MongoDB Enterprise Advanced

MongoDB Community Edition

**MongoDB Community Server**

MongoDB Community Kubernetes Operator

Tools

Atlas SQL Interface

Mobile & Edge

OR

Get started in the cloud

Try Atlas

Version

7.0.7 (current)

Platform

Windows x64

Package

msi

Download Copy link More Options

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>

learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

versión

SQL Server 2022

Filtrar por título

> Distributed Replay

> Administrador de configuración de SQL Server

> SQLCMD

> Diagnóstico de SSB

**Descarga de SSMS**

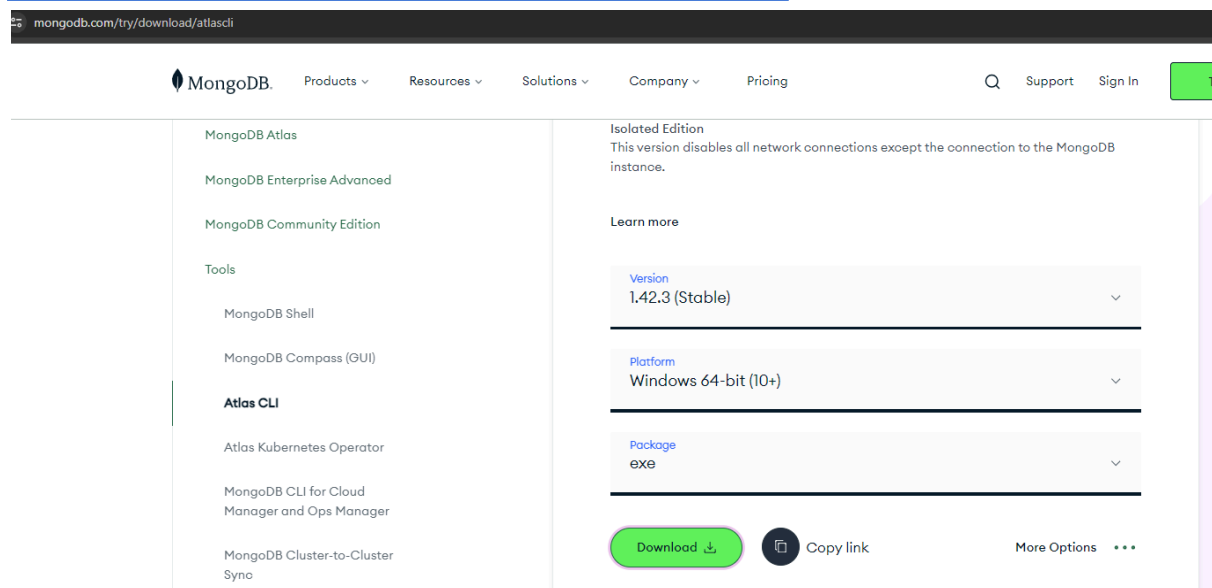
Descargar SQL Server Management Studio (SSMS) 20.0

SSMS 20.0 es la última versión de disponibilidad general (GA). Si tiene instalada una *versión preliminar* de SSMS 20, desinstálala antes de instalar SSMS 20.0. La instalación de SSMS 20 no actualiza ni reemplaza las versiones 19.x de SSMS ni las anteriores.

- Número de versión: 20.0

### Mongodb Compass

[https://downloads.mongodb.com/compass/mongodb-compass-1.42.3-win32-x64.exe?\\_ga=2.32022904.792464078.1711297515-188737971.1710007121](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**

### 2<sup>do</sup> 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](https://www.java.com/es/download/ie_manual.jsp)

java.com/es/download/ie\_manual.jsp

### Recursos de ayuda

[¿Qué es Java?](#)  
[Eliminar versiones antiguas](#)  
[Desactivar Java](#)  
[Mensajes de error](#)  
[Solución de problemas de Java](#)  
[Más ayuda](#)

### Usuarios de Windows de 64 bits

[¿Usa exploradores tanto de 32 como de 64 bits?](#)  
[Preguntas frecuentes sobre Java de 64 bits para Windows](#)

### Instalación fuera de línea

[¿Tiene problemas al descargar?](#)  
[Pruebe con el instalador fuera de línea](#)

### Información importante sobre la licencia de Oracle Java

**La licencia de Oracle Java ha cambiado para las versiones publicadas a partir de 2019.**

El acuerdo de licencia de Oracle Technology Network para Oracle Java SE es sustancialmente diferente a las licencias de Oracle Java anteriores. Esta licencia permite ciertos usos, como uso personal y de desarrollo, sin costo alguno (aunque podría haber otros usos autorizados en las licencias de Oracle Java anteriores que ya no estén disponibles). Revise las condiciones con atención al descargar y utilizar este producto. Puede consultar las preguntas frecuentes [aquí](#).

La licencia comercial y el soporte están disponibles con una suscripción de Java SE de

Descargar Java

Al descargar Java, confirma que ha leído y acepta las condiciones del acuerdo de licencia de Oracle Technology Network para Oracle Java SE

## JMeter

[https://jmeter.apache.org/download\\_jmeter.cgi](https://jmeter.apache.org/download_jmeter.cgi)

jmeter.apache.org/download\_jmeter.cgi



### Download Apache JMeter

[Twitter](#) [github](#)

We recommend you use a mirror to download our release builds, but you **must** [verify the integrity](#) of the downloaded files using signatures downloaded from our main distribution directories. Recent releases (48 hours) may not yet be available from all the mirrors.

You are currently using <https://d1cdn.apache.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors:

The **KEYS** link links to the code signing keys used to sign the product. The **PGP** link downloads the OpenPGP compatible signature from our main site. The **SHA-512** link downloads the sha512 checksum from the main site. Please [verify the integrity](#) of the downloaded file.

For more information concerning Apache JMeter, see the [Apache JMeter](#) site.

[KEYS](#)

### Apache JMeter 5.6.3 (Requires Java 8+)

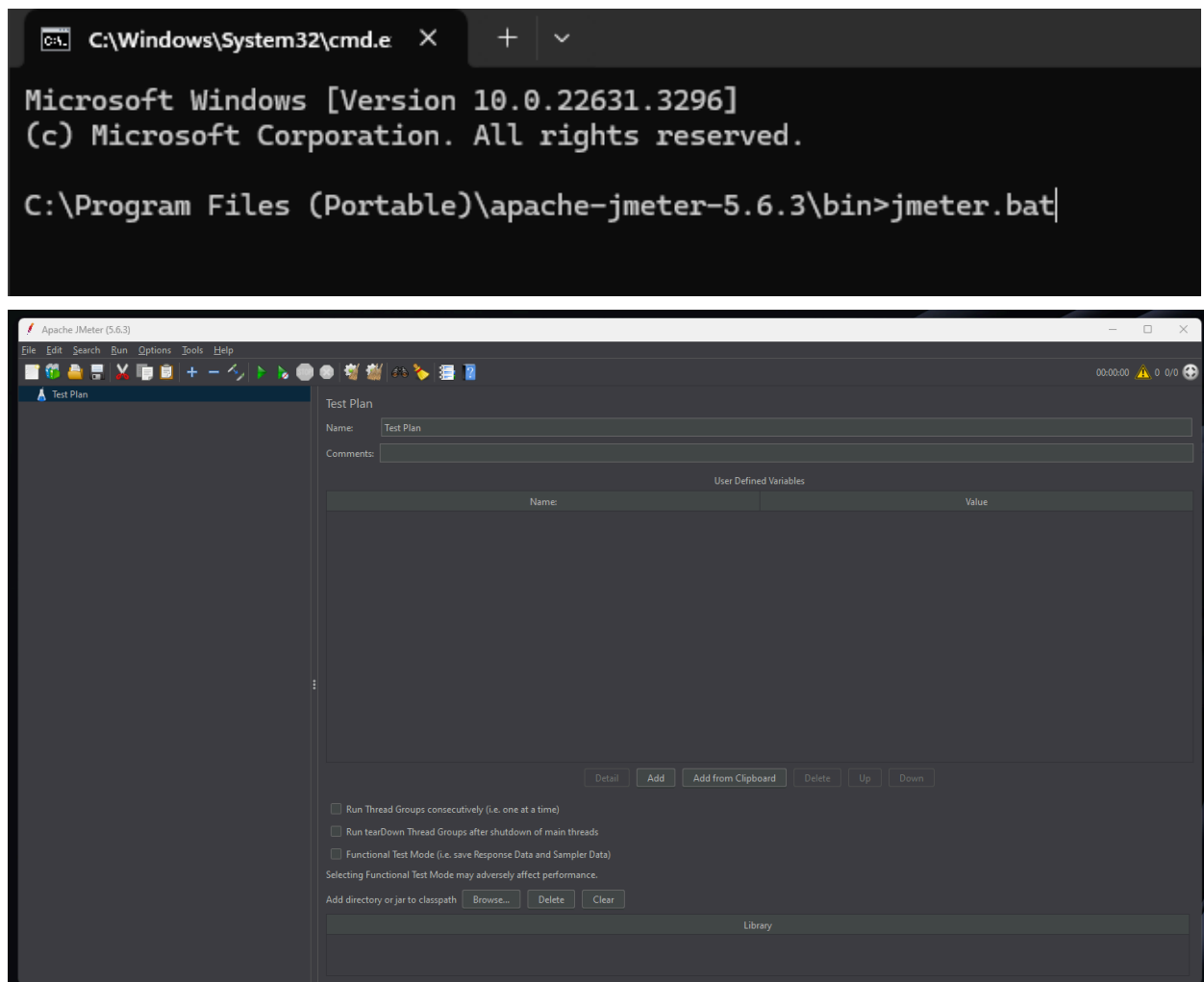
#### Binaries

[apache-jmeter-5.6.3.tgz sha512 pgp](#)  
[apache-jmeter-5.6.3.zip sha512 pgp](#)

#### Source

## 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.

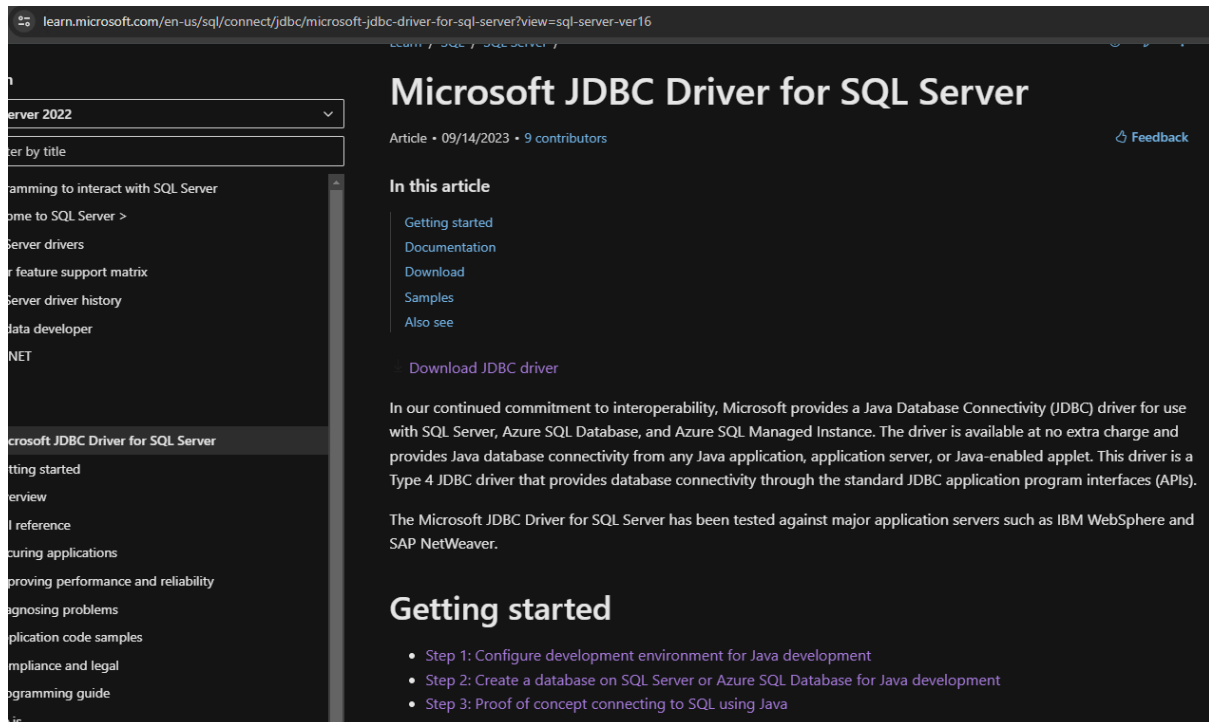


### Drivers (Conectores)

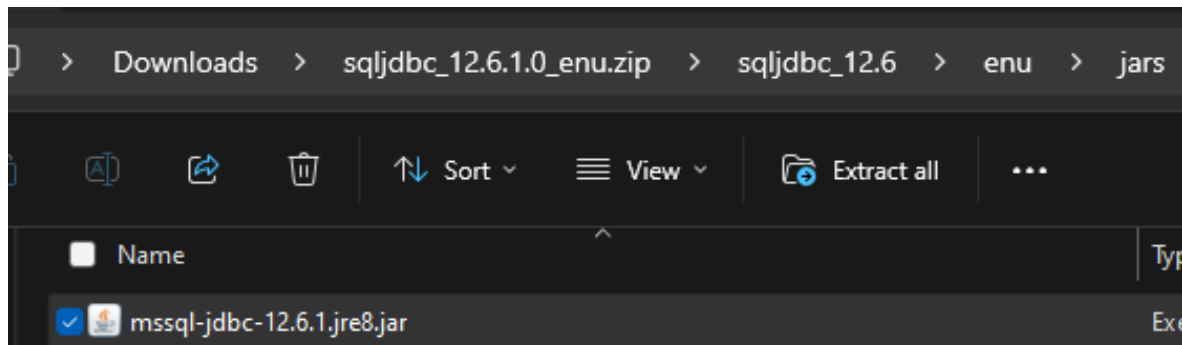
Para lograr la conexión entre **JMeter** y los motores seleccionados, es necesario tener actualizados los conectores **JDBC**.

#### Ms Sql Server:

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



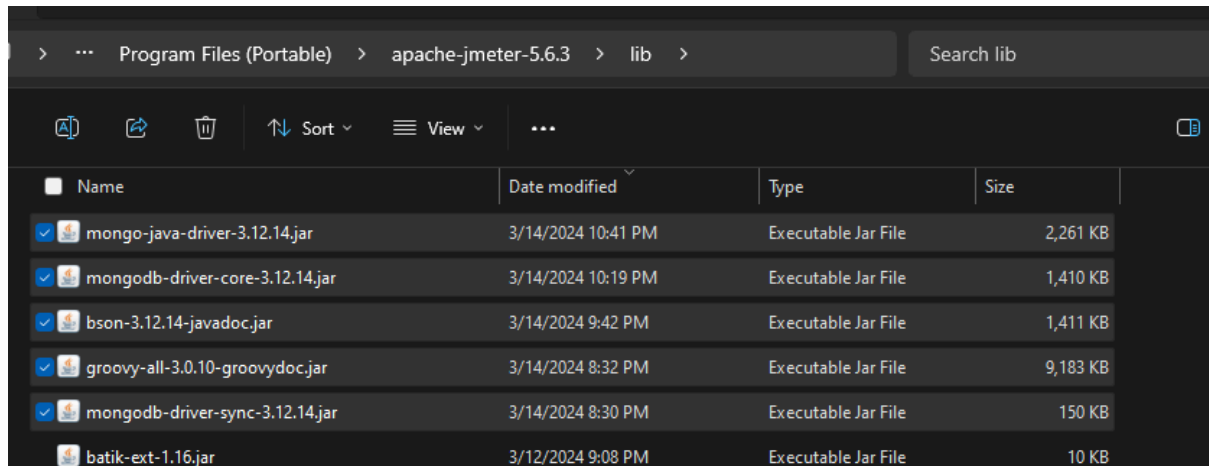
Una vez descargado el archivo **sqljdbc\_12.6.1.0\_enu.zip** se debera descomprimir el archivo **mssql-jdbc-12.6.1.jre8.jar** dentro de la carpeta lib de **JMeter**



### 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



### 3<sup>er</sup> Paso: Descarga del Dataset

Para descargar el dataset, se debe hacer click en el enlace

[https://downloads.brentozar.com/StackOverflow2013\\_201809117.7z](https://downloads.brentozar.com/StackOverflow2013_201809117.7z)

Luego se deben descomprimir los archivos de sql Server **.mdf** en alguna carpeta deseada.

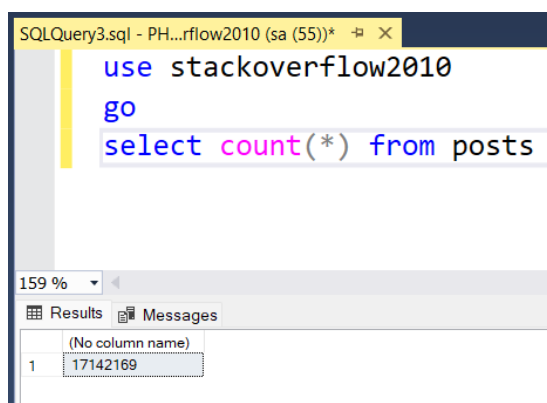
<https://www.whatisfileextension.com/es/mdf/>

### 4<sup>to</sup> 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:

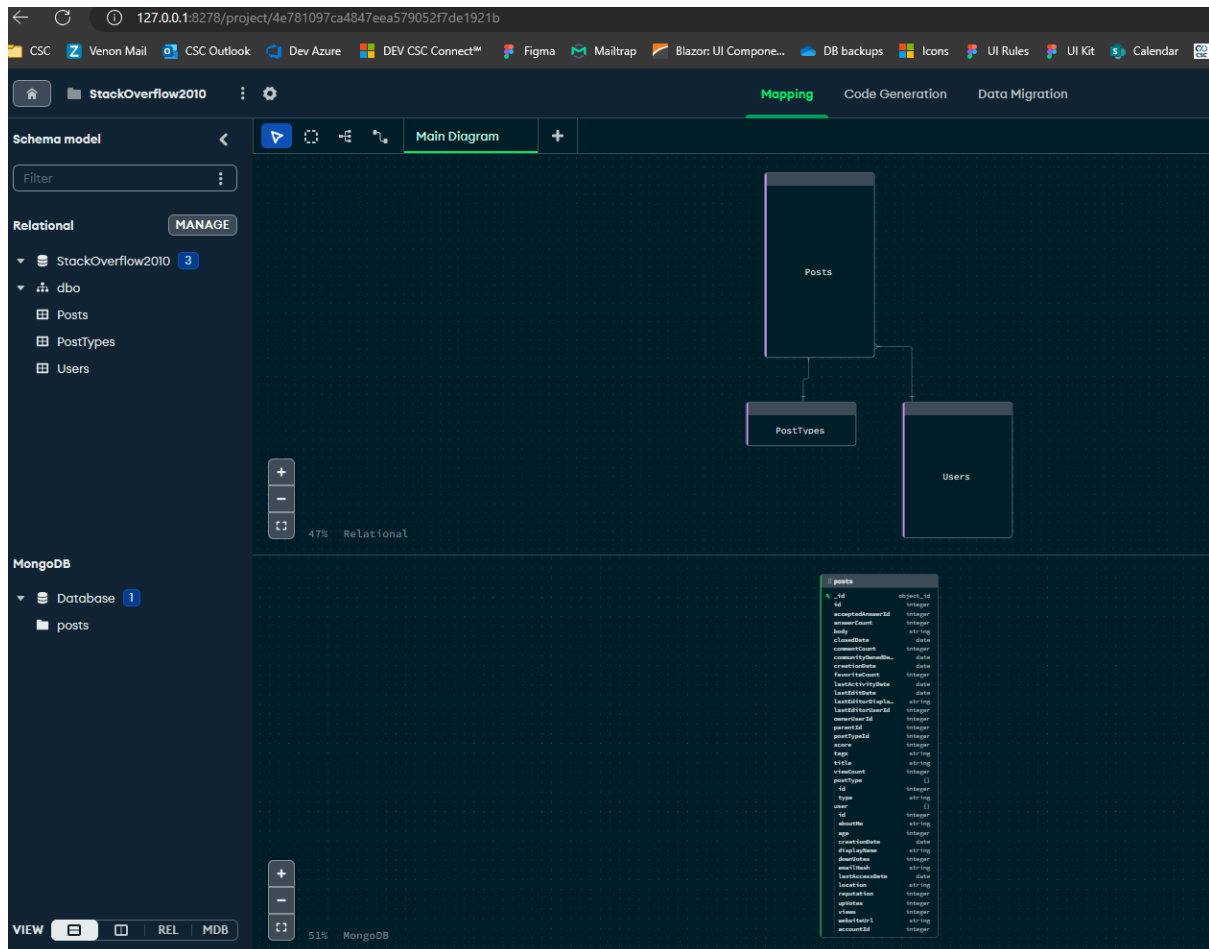


Nota: La base de datos original es llamada **StackOverflow2013**, fue renombrada a **StackOverflow2010** dado que ya se habían escrito las pruebas con ese nombre.

## 5<sup>to</sup> 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>



Fue necesario antes crear la foreign key entre las tablas **Posts**, **PostTypes** y **Users** en MSSQL Server, de esta manera se migraron como documentos anidados en el documento principal Posts.

The image shows two side-by-side screenshots. The left screenshot is a terminal window with a dark background. It shows a SQL query being executed: `usestackoverflow2010`, `go`, and `select count(*) from posts`. Below the query, there is a table with one row and one column. The column is labeled "(No column name)" and the row contains the value "17142169". The right screenshot is a MongoDB shell window with a dark background. It shows the command `> _MONGOSH` at the top, followed by `> db.posts.count()` and the result `< 17142169`. At the bottom, there is a prompt `StackOverflow2010>`.



### 6º Paso: Configuración de las pruebas, queries y resultados

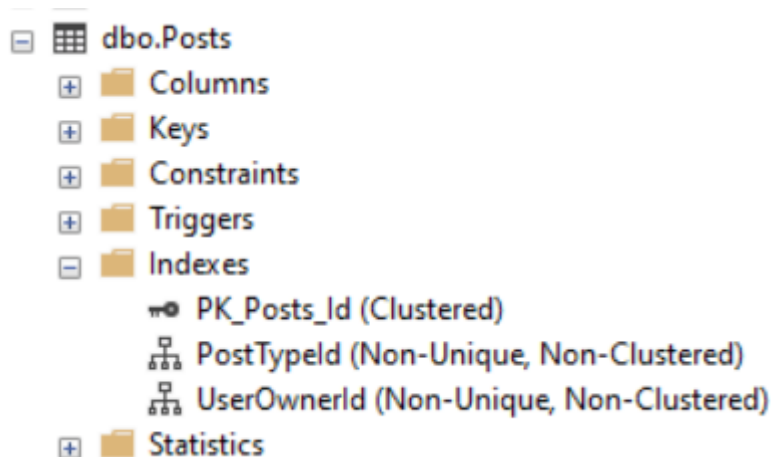
El primer paso para poder comparar las bases de datos de manera equitativa, será la de equiparar índices, más allá del método de indexación de cada una, será necesario al menos de poseer los mismos “campos” indexados en ambas.

StackOverflow2010 > posts

Documents 17.1M Aggregations Schema **Indexes 4** Validation

Create Index Refresh

Name and Definition	Type	Size	Usage
> _id_	REGULAR ⓘ	206.1 MB	2008 (since Sat Mar 16 2024)
> id_1	REGULAR ⓘ	196.1 MB	377 (since Sat Mar 16 2024)
> ownerUserId_1	REGULAR ⓘ	249.8 MB	63 (since Sat Mar 16 2024)
> postTypeId_1	REGULAR ⓘ	83.4 MB	0 (since Sat Mar 16 2024)



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

#### 1) CREATE

Se insertarán 500000 registros **Posts** con datos aleatorios en ambas bd con un Id negativo para luego reconocerlos fácilmente en los subsiguientes queries.

### 2) UPDATES

Se realizará el update en el campo **lastEditorUserId** seteando al usuario anónimo (id=11) para todos aquellos registros que posean un null en ese campo y su id sea negativo.

### 3) DELETE

Se eliminaron los 500000 registros insertados, con id negativo.

### 4) READS

Se realizarán las siguientes lecturas en ambos motores:

- a) **Agregación:** Conteo total de Posts
- b) **Primeros 1000 Posts:** con ordenación default y sus documentos hijos (MongoDb o Joins Sql).
- c) **Primeros 10000 Posts:** con ordenación default y sus documentos hijos (MongoDb o Joins Sql).
- d) **Conteo:** Total de posts creados por un usuario anónimo.
- e) **Total de Posts:** creados por usuarios anónimos.

Scripts SQL:

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

Scripts MongoDB (groovy):

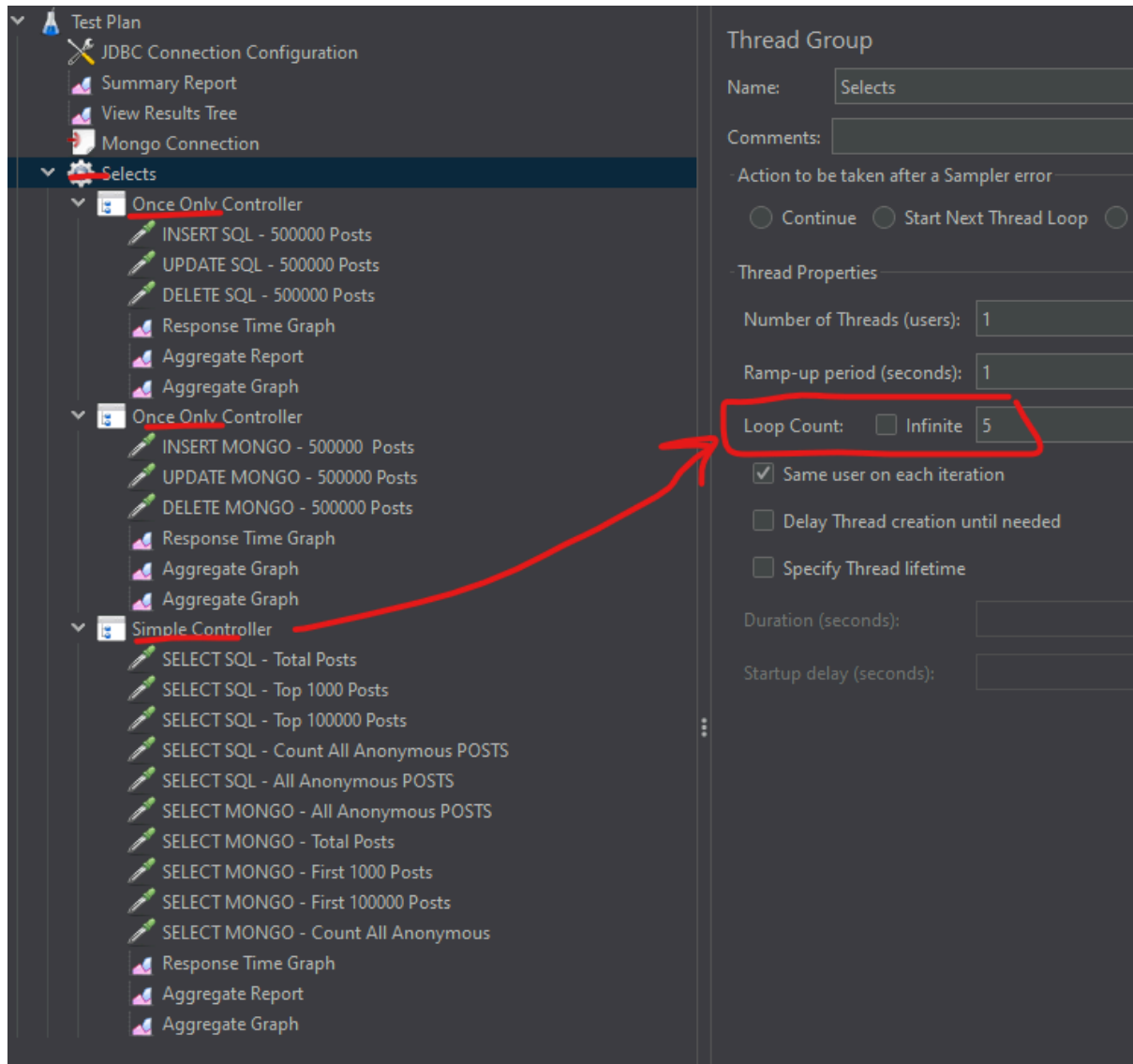
<https://github.com/marianocarreira/BaseDeDatos2023/tree/main/Scripts/MongoDb>

## 7<sup>mo</sup> Paso: Ejecución

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 la prueba de cada motor.

**Reads** -> Las Lecturas se ejecutarán en un loop de 5, o sea, 5 veces consecutivas cada una para cada motor, de esta manera podremos evaluar el efecto o no del cache luego de la primera ronda.



Documentación de Jmeter:

<https://jmeter.apache.org/usermanual/index.html>

Jmeter Test File :

<https://github.com/marianocarreira/BaseDeDatos2023/tree/main/Jmeter>

### 8<sup>vo</sup> Paso: Resultados

Los resultados se esperan en los siguientes formatos/gráficos:

**Summary Report:**

[https://jmeter.apache.org/usermanual/component\\_reference.html#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](https://jmeter.apache.org/usermanual/component_reference.html#View_Results_Tree)

**Response Time Graph:**

## Benchmarking

[https://jmeter.apache.org/usermanual/component\\_reference.html#Response\\_Time\\_Graph](https://jmeter.apache.org/usermanual/component_reference.html#Response_Time_Graph)

### Aggregate Report:

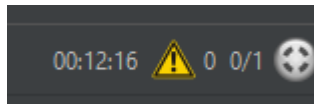
[https://jmeter.apache.org/usermanual/component\\_reference.html#Aggregate\\_Report](https://jmeter.apache.org/usermanual/component_reference.html#Aggregate_Report)

### Aggregate Graph:

[https://jmeter.apache.org/usermanual/component\\_reference.html#Aggregate\\_Graph](https://jmeter.apache.org/usermanual/component_reference.html#Aggregate_Graph)

Tiempo de ejecución total:

12 min con 42 seg.



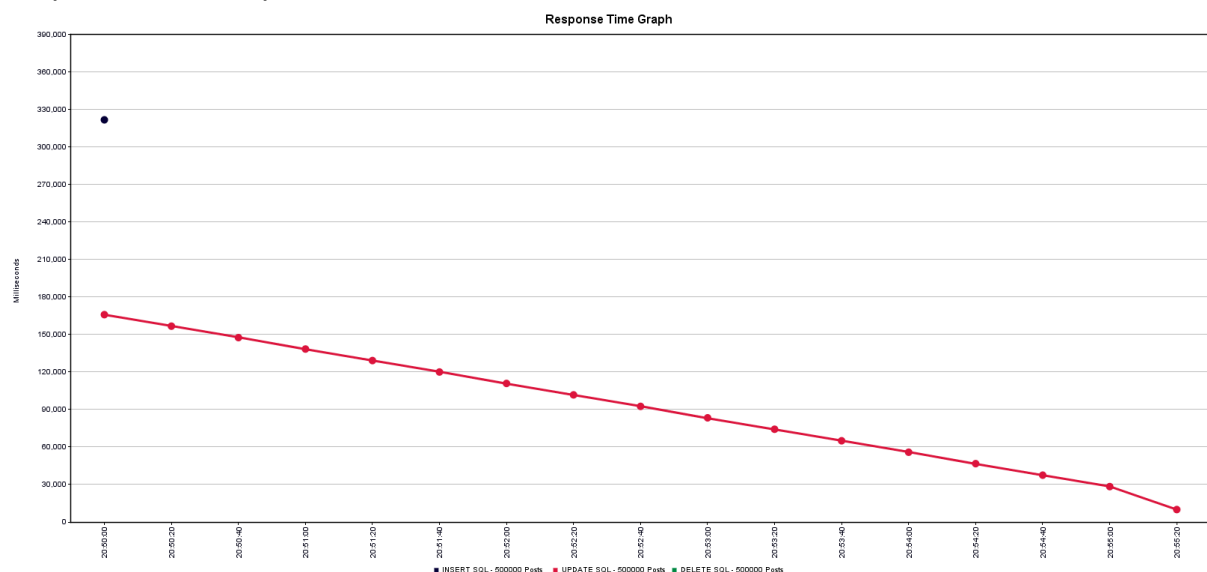
## Resultados Generales:

### Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Selects INSERT SQL - 500000 Posts	1	321780	321780	321780	0.00	0.00%	11.2/hour	0.00	0.00	9.0
Selects UPDATE SQL - 500000 Posts	1	9996	9996	9996	0.00	0.00%	6.0/min	0.00	0.00	14.0
Selects DELETE SQL - 500000 Posts	1	27228	27228	27228	0.00	0.00%	2.2/min	0.00	0.00	16.0
Selects INSERT MONGO - 500000 Posts	1	96022	96022	96022	0.00	0.00%	37.5/hour	0.00	0.00	.0
Selects UPDATE MONGO - 500000 Posts	1	53231	53231	53231	0.00	0.00%	1.1/min	0.00	0.00	84.0
Selects DELETE MONGO - 500000 Posts	1	57098	57098	57098	0.00	0.00%	1.1/min	0.00	0.00	45.0
Selects SELECT SQL - Total Posts	5	1767	1408	2165	242.77	0.00%	2.2/min	0.00	0.00	10.0
Selects SELECT SQL - Top 1000 Posts	5	683	33	3220	1268.40	0.00%	2.2/min	44.03	0.00	1212880.0
Selects SELECT SQL - Top 100000 Posts	5	5187	4232	5543	489.38	0.00%	2.2/min	3910.76	0.00	108521202.0
Selects SELECT SQL - Count All Anonymous POSTS	5	1199	1147	1266	43.44	0.00%	2.3/min	0.00	0.00	8.0
Selects SELECT SQL - All Anonymous POSTS	5	9772	7109	13054	2017.67	0.00%	2.1/min	7590.29	0.00	228988006.0
Selects SELECT MONGO - All Anonymous POSTS	5	10289	9431	11854	880.78	0.00%	2.1/min	13694.27	0.00	400063172.0
Selects SELECT MONGO - Total Posts	5	46	36	78	15.71	0.00%	2.3/min	0.00	0.00	8.0
Selects SELECT MONGO - First 1000 Posts	5	95	89	106	6.05	0.00%	2.3/min	67.99	0.00	1823165.0
Selects SELECT MONGO - First 100000 Posts	5	4429	4068	5632	603.18	0.00%	2.2/min	6096.88	0.00	170413359.0
Selects SELECT MONGO - Count All Anonymous	5	170	141	209	24.77	0.00%	2.3/min	0.00	0.00	8.0
TOTAL	56	13099	33	321780	44599.48	0.00%	4.6/min	6048.72	0.00	81341236.6

## Resultados SQL Server (Insert/Update/Delete)

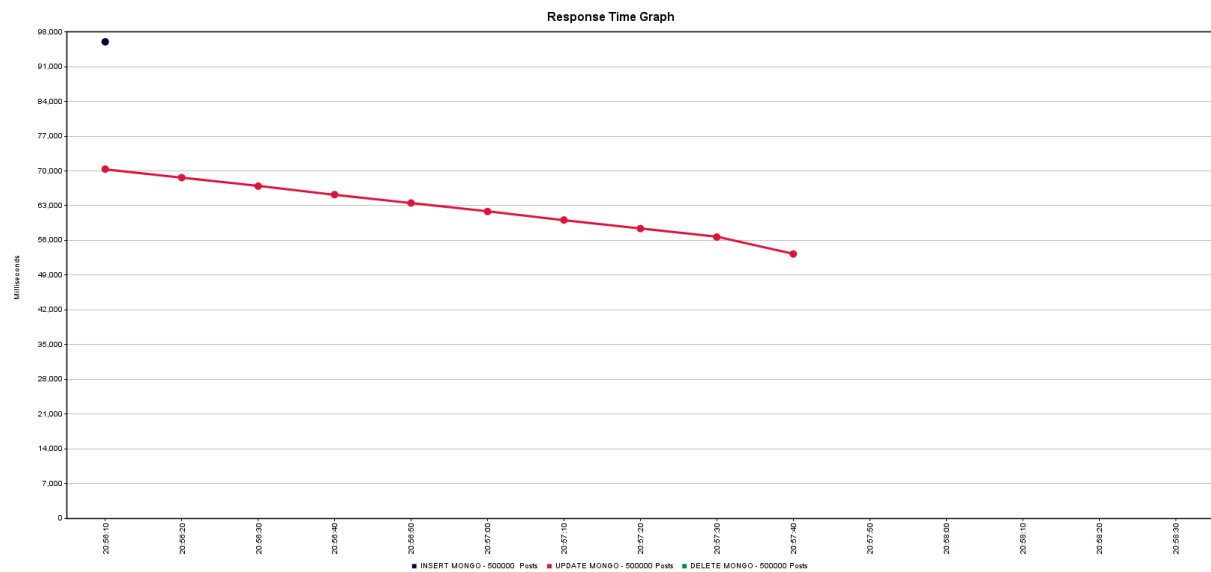
### Response Time Graph:



## Benchmarking

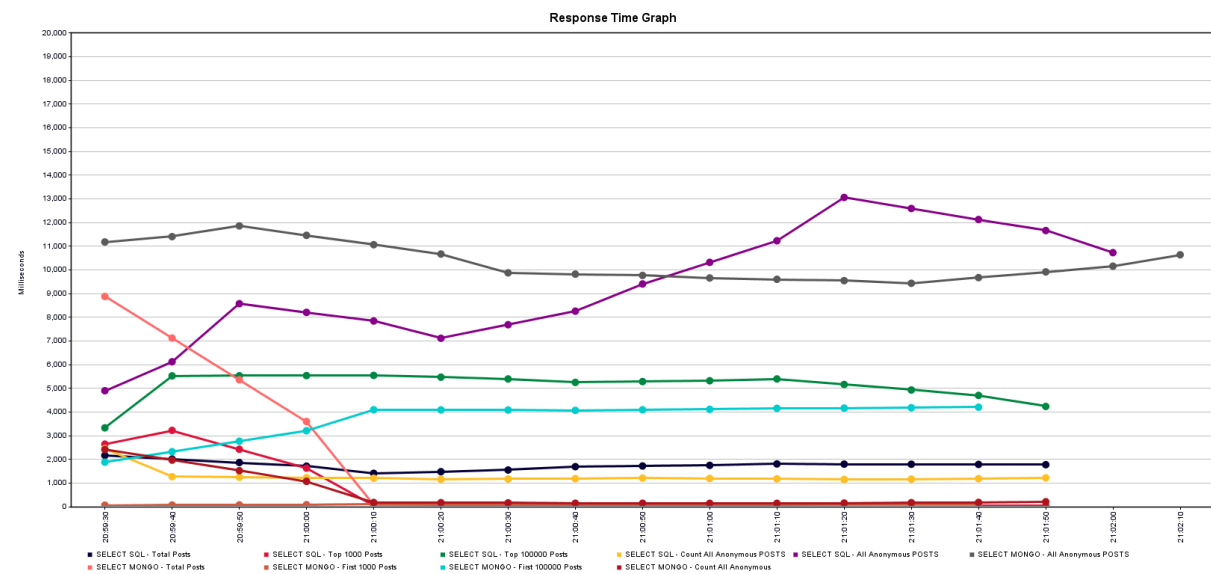
### Resultados MongoDB (Insert/Update/Delete)

Response Time Graph:



### Resultados SQL & Mongo (Reads/Selects)

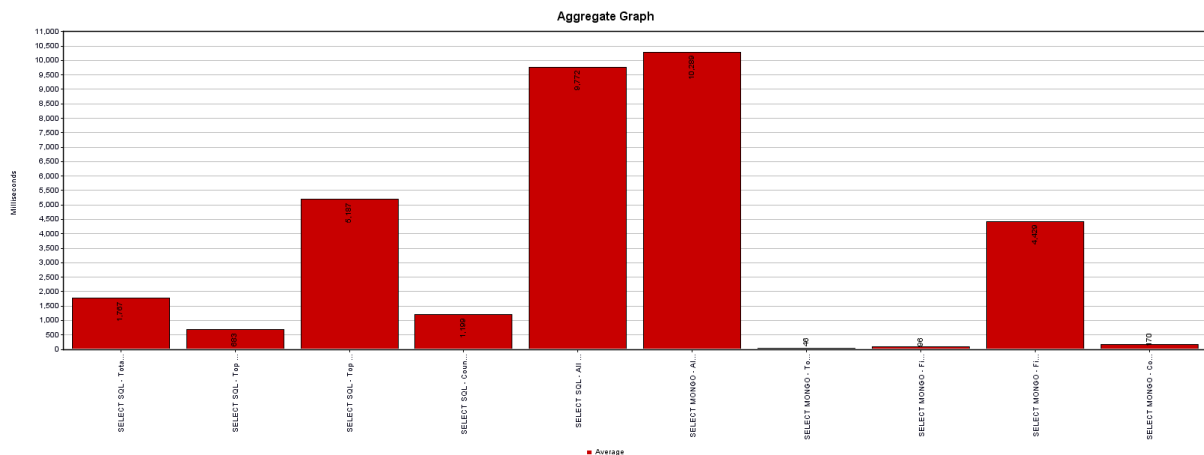
Response Time Graph:



### Aggregate Report:

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
SELECT SQL - Total Posts	5	1767	1772	1803	2165	2165	1408	2165	0.00%	2.2/min	0.00	0.00
SELECT SQL - Top 1000 Posts	5	683	54	76	3220	3220	33	3220	0.00%	2.2/min	44.03	0.00
SELECT SQL - Top 100000 Posts	5	5187	5393	5523	5543	5543	4232	5543	0.00%	2.2/min	3910.76	0.00
SELECT SQL - Count All Anonymous POSTS	5	1199	1208	1219	1266	1266	1147	1266	0.00%	2.3/min	0.00	0.00
SELECT SQL - All Anonymous POSTS	5	9772	9402	10726	13054	13054	7109	13054	0.00%	2.1/min	7950.29	0.00
SELECT MONGO - All Anonymous POSTS	5	10289	9873	10635	11854	11854	9431	11854	0.00%	2.1/min	13694.27	0.00
SELECT MONGO - Total Posts	5	46	41	41	78	78	36	78	0.00%	2.3/min	0.00	0.00
SELECT MONGO - First 1000 Posts	5	95	95	95	105	105	89	105	0.00%	2.3/min	67.99	0.00
SELECT MONGO - First 100000 Posts	5	4429	4152	4201	5632	5632	4068	5632	0.00%	2.2/min	6096.88	0.00
SELECT MONGO - Count All Anonymous	5	170	177	179	209	209	141	209	0.00%	2.3/min	0.00	0.00
TOTAL	50	3364	1408	9556	10635	13054	33	13054	0.00%	17.7/min	26205.14	0.00

### Aggregate Graph:



## Conclusión

### Inserts:

De las 3 operaciones realizadas con la misma cantidad de registros, las **inserciones** fueron las operaciones más costosas para ambos motores:

- En **SQL server** fueron hasta un poco más de **11 veces más costosas** en tiempo que los deletes (5,36 min. vs 30 seg.) y hasta **32 veces más costosas** que los updates. para la misma cantidad de registros.
- En **MongoDB** puede observarse el mismo escenario, Insert 1,6 min. vs casi 1 min, al igual que el update de 1 min. El insert fue de **sólo 0,6 veces más costoso**.

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](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/>

Se puede observar que incluso iterando y agregando elemento por elemento en **MongoDb** (ver [script](#)) fue en los inserts **3,35 veces más rápido que Ms Sql Server** y que por otro lado la relación entre las 3 operaciones fue, en tiempos de respuesta, más uniforme en MongoDB.

### Updates:

Los updates fueron **5,32 veces más rápidos en Ms SQL Server**

### Deletes:

Los deletes fueron casi **2 veces más rápidos en Ms SQL Server**

### Reads:

Total Posts:

- **MongoDB fue 38,41 veces más rápido.**

First 1000 Posts:

- **MongoDB fue 7,11 veces más rápido.**

First 100000 Posts:

- **MongoDB fue 1,17 veces más rápido.**

Count All Anonymous:

- **MongoDB fue 7 veces más rápido.**

All anonymous Posts:

- **Ambos motores presentan casi el mismo tiempo.**

Cabe notar que en aspectos de peso o cantidad de datos, MongoDB a priori devuelve mayor cantidad de datos.

**SQL Server 338 MB**

**MongoDB 572 MB**

**MongoDb devolvió un 1,7 mayor cantidad de datos que SQL Server**, las causas más probables pueden ser:

- 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.

### Conclusión final:

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.

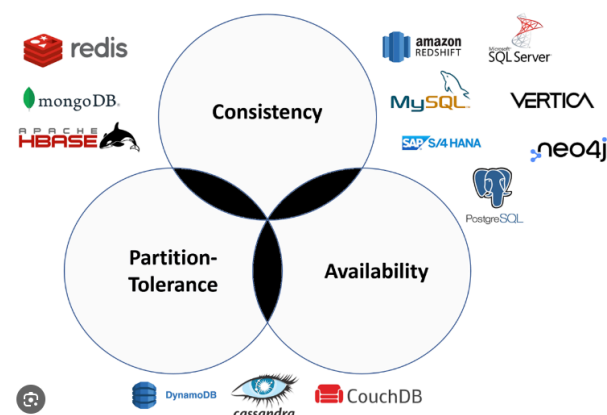


Fig. Teorema CAP

## 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://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/>