Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

Base de Datos NoSQL: MongoDB vs. Cassandra en operaciones CRUD (Create, Read, Update, Delete)

Jessica Nataly Castillo[1], Jonathan Ricardo Garcés[2], Milton Patricio Navas[3], Diego Fernando Jácome Segovia[4]

1 Universidad Técnica de Cotopaxi, jessica.castillo@utc.edu.ec

2 Universidad de las Fuerzas Armadas ESPE, jrgarces3@espe.edu.ec

3 Universidad de las Fuerzas Armadas ESPE, mpnavas@espe.edu.ec

4. Universidad Técnica de Cotopaxi, diego.jacome@utc.edu.ec

RESUMEN

En el pasado, se utilizaron bases de datos relacionales debido a su rico conjunto de características, capacidades de consulta y gestión de transacciones en muchas aplicaciones. Sin embargo, no son capaces de almacenar y procesar datos de gran eficacia y no son muy eficientes para realizar transacciones y unirse a las operaciones. Recientemente, emerge un nuevo paradigma, bases de datos NoSQL, para superar algunos de estos problemas, que son más convenientes para el uso en entornos web. En esta investigación se examinarán las características esenciales de los sistemas de gestión de base de datos NoSql como son MongoDB y Cassandra. Nos centraremos en las operaciones CRUD (Create, Read, Update, Delete). Para nuestro ejemplo vamos a crear dos bases de datos, para ver cuál de los dos es más eficiente administrando grandes cantidades de datos.

Keywords: Bases de datos NoSQL, CRUD Operations, MongoDB, Cassandra.

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

NoSQL Database: MongoDB vs. Cassandra in CRUD operations (Create, Read, Update, Delete)

ABSTRACT

In the past, relational databases were used because of their rich feature set, query and transaction management capabilities in many applications. However, they are not able to store and process highly efficient data and are not very efficient at transacting and joining operations. Recently, a new paradigm emerges, NoSQL databases, to overcome some of these problems, which are more convenient for use in web environments. This research will examine the essential characteristics of NoSql database management systems such as MongoDB and Cassandra. We will focus on CRUD (Create, Read, Update, Delete) operations. For our example we are going to create two databases, to see which one is more efficient by managing large amounts of data.

Keywords: NoSQL Databases, CRUD Operations, MongoDB, Cassandra

1. INTRODUCCIÓN

Las bases de datos NoSQL surgen como una necesidad para el manejo y procesamiento de enormes cantidades de datos con una escalabilidad horizontal en las aplicaciones, es decir, la necesidad de mejorar el rendimiento mediante la adición de nuevos nodos computacionales a los ya existentes [1], para las cuales, son útiles cuando se trabaja con una gran cantidad de datos, cuando la naturaleza de los datos no requiere de un modelo relacional. [2]

Al principio del nuevo milenio, los desarrolladores comenzaron a darse cuenta de que sus datos no encajaban en el modelo relacional y algunos de ellos comenzaron a desarrollar otras arquitecturas para almacenar grandes cantidades de datos, por lo cual, en la hora de elegir una base de datos hoy en día, el problema más complejo, es decidir la mejor arquitectura para el almacenamiento de datos y recuperación de datos. [3]

En la actualidad existen muchos sistemas que poseen inmensas cantidades de datos con millones de transacciones al día contra la base de datos, por lo cual, si hablamos de bases de datos NoSQL, existen gran variedad de estas que se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, bases de datos documentales, y bases de datos orientadas a grafos. [4] Pero el problema principal que encontramos hoy en día, es que aunque todas se denominan NoSQL, no sabemos cuál escoger para crear una aplicación adecuada que cumpla con las expectativas de los clientes.

Por lo tanto, el propósito de este estudio es comparar el rendimiento entre MongoDB [5] y Cassandra [6], mediante las operaciones CRUD (create, read, update y delete) inserción, consulta, actualización y borrado sobre un determinado keyspace/collection. Se tomaron estas bases de datos porque son parte de los tres principales SGBD NoSQL junto con Hbase [7]. Además, se eligieron estas operaciones, porque son las principales para la gestión de datos en las aplicaciones de BD, ofreciendo un punto de partida para diseñadores y desarrolladores, a la hora de elegir un SGBD para una aplicación.

I. BASE DE DATOS NOSQL

NoSQL es un término utilizado para referirse a las bases de datos no relacionales. Por lo tanto, abarca la mayoría de los almacenes de datos que no se basan en los principios de RDBMS convencionales y se utilizan para el manejo de grandes volúmenes de datos a escala de Internet. [8]

Las bases de datos NOSQL son un conjunto de bases de datos que no se ajustan al modelo de bases de datos relacionales y sus características son: 1) no tienen esquemas, 2) no usan SQL 3) no permiten joins, 4) no garantizan la propiedad ACID (atomicidad, consistencia, aislamiento y durabilidad), 5) escalan bien horizontalmente, 6) resuelven el problema de los altos volúmenes de información y la inmensa cantidad de consultas y transacciones diarias, en resumen no son relacionales.

Los sistemas NoSQL se utilizan normalmente para grandes conjuntos de datos que es accedido y manipulado en una escala de web. Más de estos sistemas llegaron desde la perspectiva de la industria en lugar que la comunidad de investigación. [9] Estos sistemas crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales RDBMS no solucionaban.

Las bases de datos NoSQL están diseñados para manejar todo tipo de fallas. Existen gran variedad de fallas de hardware que pueden ocurrir, por lo cual, el sistema debe estar preparado por lo que es más funcional a tener en cuenta esas preocupaciones como eventuales ocurrencias de algunos acontecimientos excepcionales. [10]

A. MongoDB

MongoDB [5] es una base de datos documental de código abierto y líder en bases de datos NoSQL. MongoDB está escrito en c++. Ofrece alta disponibilidad, escalabilidad y particionamiento a costa de consistencia y soporte transaccional. En términos prácticos, esto significa que en lugar de tablas y filas, MongoDB utiliza documentos para hacerla flexible, escalable y rápida. [8]

Guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON para almacenar sus documentos), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Los documentos BSON sirven para mantener una lista ordenada de elementos, estos elementos tiene tres componentes: 1) un nombre de campo, 2) un tipo de datos y 3) un valor. [11] Estos documentos pueden tener esquemas diferentes, lo que significa que el esquema puede cambiar a medida que evoluciona la aplicación.

B. Cassandra

Casandra [6] es un sistema de gestión de base de datos diseñado para manejar grandes cantidades de datos repartidos por muchos servidores, mientras proporciona un servicio de alta disponibilidad sin ningún punto único de fallo. [12]

Sus características son: 1) su esquema es muy flexible y no requiere esquema de base de datos de diseño en un principio. Añadir o eliminar

campos es muy conveniente; 2) soporta consultas de alto rango, es decir que pueden variar en consultas clave; 3) posee alta escalabilidad: un punto único de fallo no afecta todo el clúster y apoyan la expansión lineal. [13]

Netflix, Twitter, Urban Airship, Reddit, Cisco, OpenX, Digg, Cloudkick, Ooyala son algunas de las empresas que utilizan Cassandra, para hacer frente a grandes conjuntos de datos activos e interactivos en línea. Cassandra posee más de 300 Terabytes (TB) de datos en más de 400 máquinas. [14]

II. MONGODB VS CASSANDRA

En esta sección se describe MongoDB y Cassandra, que son las bases de datos elegidas para el análisis y pruebas. Las principales características que deben analizarse son las operaciones CRUD (create, read, update y delete) inserción, consulta, actualización y borrado sobre un keyspace/collection por medio de gráficos estadísticos para mostrar cuál de ellas es la más óptima. Estas bases de datos fueron seleccionadas con el fin de comparar las dos bases de datos ya mencionadas para dar un punto de partida a los diseñadores y desarrolladores, a la hora de elegir una SGBD para una aplicación.

A. Características de Comparación

Con el fin de entender mejor las diferencias entre MongoDB y Cassandra estudiamos algunas de las características de las bases de datos NoSQL, tales como: el lenguaje de desarrollo, tipo de almacenamiento, protocolos, replicación, uso y otras características. Todas esas características se muestran en la Tabla 1.

Tabla 1. Características de MongoDB y Cassandra.

Característica	MongoDB	Cassandra
Tipo	Orientado a Documentos	Orientado a Columnas
Lenguaje de desarrollo	C++	Java
Lenguaje de Consulta	Objetos y métodos personalizados, basado en JavaScript	CQL
Tipo de almacenamiento	Documentos BSON	Columnas

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

Protocolo	TCP/IP	TCP/IP
Locks	Si	Si
Triggers	No	Si
Concurrencia	Actualización instantánea	MVCC
Sistemas Operativos	Linux/ MAC OS/ Windows	Linux/ MAC OS/ Windows
Replicación	Master-Slave	Multi-Master
Tolerancia a Fallos	Alto	Excepcionalmente Alto
Características mas importantes	Conserva algunas propiedades SQL, como la consulta y el índice	Escala linealmente, Es distribuida, Escala de forma horizontal.
Áreas de uso	Almacenamiento y registro de eventos, Sistemas gestión de contenidos (CMS)	Sistemas bancarios, Sistemas financieros, Publicidad
Código Abierto	Si	Si

Mediante el análisis de las propiedades básicas, es posible concluir que existen similitudes cuando se trata de tipos de archivos utilizados, consultas, transacciones, locks, almacenamiento de datos, código abireto y sistemas operativos.

En cuanto a términos de uso, MongoDB tiene un mejor uso para Sistemas de Gestion de Contenidos (CMS), mientras tiene consultas dinámicas con datos escritos frecuentemente. Cassandra está optimizado para almacenar e interactuar con grandes cantidades de datos que se pueden utilizar en diferentes áreas, tales como, finanzas o publicidad.

III. EXPERIMENTOS Y RESULTADOS

A continuación se presenta una serie de pruebas de rendimiento entre MongoDB y Cassandra. Para las pruebas se usó un equipo con procesador Intel (R) Core (TM) i7 CPU de 1.73 GHz y con 4 GB de memoria RAM con un sistema operativo Windows 7 de 64 bits. Se usó la versión 3.2.7 de MongoDB y la versión 3.7.0 en Cassandra. Las pruebas se hicieron para las cuatro operaciones: inserción, actualización, consulta y borrado.

Para comparar la eficiencia de las bases de datos NoSQL se ejecutó tres veces para cada tamaño de muestra elegido. Los tamaños para la inserción, consulta, modificación y eliminación de un conjunto de datos fueron de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos; y se tomó el promedio del tiempo de ejecución para cada tamaño dado.

Para realizar las operaciones CRUD para la base de datos de MongoDB se tomó en cuenta los siguientes campos: _id, created_on, value. Para realizar las operaciones CRUD en la base de datos de Cassandra se tomó los siguientes campos: id, name, number.

En la primera prueba se realizó la función de insertar, para la cual se tomó un conjunto de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos, después se utilizó un comando apropiado para insertar en una base de datos vacía, después se tomó el tiempo de cuanto se demoró la operación, estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes.

Para la prueba en Cassandra, se utilizó en siguiente comando para insertar datos:

```
INSERT INTO prueba.users ( id, name, number)
VALUES ( 1, 'Juan Perez', 25156051 );
```

La Tabla 2 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación insertar en Cassandra.

Tabla 2. Resultados de la operación inserción en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	9.914	9.930	9.443	**9.762**
5000	20.303	21.816	21.166	**21.095**
10000	33.841	33.424	34.002	**33.756**
50000	146.734	147.426	148.565	**147.575**
100000	209.601	205.350	208.808	**207.920**

Para la prueba en MongoDB, se utilizó en siguiente comando para insertar datos:

```
db.prueba.insert (
  {
    _id: '579ece2ac691d91feafde781',
    created_on: '2015-05-10T02:36:25.338Z',
    value: 0.193427741171179
  }
);
```

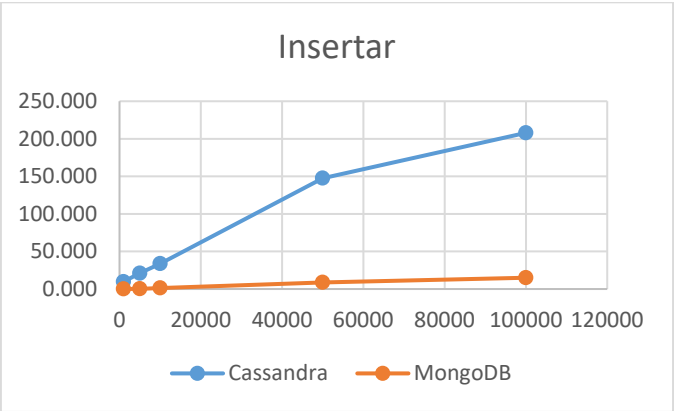
La Tabla 3 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación insertar en MongoDB.

Tabla 3. Resultados de la operación inserción en MongoDB.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.008	0.006	0.007	**0.007**
5000	0.261	0.309	0.286	**0.285**
10000	1.263	1.258	1.471	**1.331**
50000	8.796	8.724	8.834	**8.785**
100000	15.001	15.002	15.022	**15.008**

Para las operaciones de inserción los resultados se muestran en la Figura 1. Existe una gran diferencia en tiempos promedio entre las dos bases de datos, por lo tanto, los resultados obtenidos favorecieron a MongoDB.

Figura 1. Resultados de las pruebas de inserción



Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

IV. BASE DE DATOS NOSQL

NoSQL es un término utilizado para referirse a las bases de datos no relacionales. Por lo tanto, abarca la mayoría de los almacenes de datos que no se basan en los principios de RDBMS convencionales y se utilizan para el manejo de grandes volúmenes de datos a escala de Internet. [8]

Las bases de datos NOSQL son un conjunto de bases de datos que no se ajustan al modelo de bases de datos relacionales y sus características son: 1) no tienen esquemas, 2) no usan SQL 3) no permiten joins, 4) no garantizan la propiedad ACID (atomicidad, consistencia, aislamiento y durabilidad), 5) escalan bien horizontalmente, 6) resuelven el problema de los altos volúmenes de información y la inmensa cantidad de consultas y transacciones diarias, en resumen no son relacionales.

Los sistemas NoSQL se utilizan normalmente para grandes conjuntos de datos que es accedido y manipulado en una escala de web. Más de estos sistemas llegaron desde la perspectiva de la industria en lugar que la comunidad de investigación. [9] Estos sistemas crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales RDBMS no solucionaban.

Las bases de datos NoSQL están diseñados para manejar todo tipo de fallas. Existen gran variedad de fallas de hardware que pueden ocurrir, por lo cual, el sistema debe estar preparado por lo que es más funcional a tener en cuenta esas preocupaciones como eventuales ocurrencias de algunos acontecimientos excepcionales. [10]

A. MongoDB

MongoDB [5] es una base de datos documental de código abierto y líder en bases de datos NoSQL. MongoDB está escrito en c++. Ofrece alta disponibilidad, escalabilidad y particionamiento a costa de consistencia y soporte transaccional. En términos prácticos, esto significa que en lugar de tablas y filas, MongoDB utiliza documentos para hacerla flexible, escalable y rápida. [8]

Guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON para almacenar sus documentos), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Los documentos BSON sirven para mantener una lista ordenada de elementos, estos elementos tiene tres componentes: 1) un nombre de campo, 2) un tipo de datos y 3) un valor. [11] Estos documentos pueden tener esquemas diferentes, lo que significa que el esquema puede cambiar a medida que evoluciona la aplicación.

B. Cassandra

Casandra [6] es un sistema de gestión de base de datos diseñado para manejar grandes cantidades de datos repartidos por muchos servidores, mientras proporciona un servicio de alta disponibilidad sin ningún punto único de fallo. [12]

Sus características son: 1) su esquema es muy flexible y no requiere esquema de base de datos de diseño en un principio. Añadir o eliminar campos es muy conveniente; 2) soporta consultas de alto rango, es decir que pueden variar en consultas clave; 3) posee alta escalabilidad: un punto único de fallo no afecta todo el clúster y apoyan la expansión lineal. [13]

Netflix, Twitter, Urban Airship, Reddit, Cisco, OpenX, Digg, Cloudkick, Ooyala son algunas de las empresas que utilizan Cassandra, para hacer frente a grandes conjuntos de datos activos e interactivos en línea. Cassandra posee más de 300 Terabytes (TB) de datos en más de 400 máquinas. [14]

V. MONGODB VS CASSANDRA

En esta sección se describe MongoDB y Cassandra, que son las bases de datos elegidas para el análisis y pruebas. Las principales características que deben analizarse son las operaciones CRUD (create, read, update y delete) inserción, consulta, actualización y borrado sobre un keyspace/collection por medio de gráficos estadísticos para mostrar cuál de ellas es la más óptima. Estas bases de datos fueron seleccionadas con el fin de comparar las dos bases de datos ya mencionadas para dar un punto de partida a los diseñadores y desarrolladores, a la hora de elegir una SGBD para una aplicación.

A. Características de Comparación

Con el fin de entender mejor las diferencias entre MongoDB y Cassandra estudiamos algunas de las características de las bases de datos NoSQL, tales como: el lenguaje de desarrollo, tipo de almacenamiento, protocolos, replicación, uso y otras características. Todas esas características se muestran en la Tabla 1.

Tabla 4. Características de MongoDB y Cassandra.

Característica	MongoDB	Cassandra
Tipo	Orientado a Documentos	Orientado a Columnas
Lenguaje de desarrollo	C++	Java
Lenguaje de Consulta	Objetos y métodos personalizados,	CQL

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

	basado en JavaScript	
Tipo de almacenamiento	Documentos BSON	Columnas
Protocolo	TCP/IP	TCP/IP
Locks	Si	Si
Triggers	No	Si
Concurrencia	Actualización instantánea	MVCC
Sistemas Operativos	Linux/ MAC OS/ Windows	Linux/ MAC OS/ Windows
Replicación	Master-Slave	Multi-Master
Tolerancia a Fallos	Alto	Excepcionalmente Alto
Características mas importantes	Conserva algunas propiedades SQL, como la consulta y el índice	Escala linealmente, Es distribuida, Escala de forma horizontal.
Áreas de uso	Almacenamiento y registro de eventos, Sistemas gestión de contenidos (CMS)	Sistemas bancarios, Sistemas financieros, Publicidad
Código Abierto	Si	Si

Mediante el análisis de las propiedades básicas, es posible concluir que existen similitudes cuando se trata de tipos de archivos utilizados, consultas, transacciones, locks, almacenamiento de datos, código abireto y sistemas operativos.

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

En cuanto a términos de uso, MongoDB tiene un mejor uso para Sistemas de Gestion de Contenidos (CMS), mientras tiene consultas dinámicas con datos escritos frecuentemente. Cassandra está optimizado para almacenar e interactuar con grandes cantidades de datos que se pueden utilizar en diferentes áreas, tales como, finanzas o publicidad.

VI. EXPERIMENTOS Y RESULTADOS

A continuación se presenta una serie de pruebas de rendimiento entre MongoDB y Cassandra. Para las pruebas se usó un equipo con procesador Intel (R) Core (TM) i7 CPU de 1.73 GHz y con 4 GB de memoria RAM con un sistema operativo Windows 7 de 64 bits. Se usó la versión 3.2.7 de MongoDB y la versión 3.7.0 en Cassandra. Las pruebas se hicieron para las cuatro operaciones: inserción, actualización, consulta y borrado.

Para comparar la eficiencia de las bases de datos NoSQL se ejecutó tres veces para cada tamaño de muestra elegido. Los tamaños para la inserción, consulta, modificación y eliminación de un conjunto de datos fueron de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos; y se tomó el promedio del tiempo de ejecución para cada tamaño dado.

Para realizar las operaciones CRUD para la base de datos de MongoDB se tomó en cuenta los siguientes campos: _id, created_on, value. Para realizar las operaciones CRUD en la base de datos de Cassandra se tomó los siguientes campos: id, name, number.

En la primera prueba se realizó la función de insertar, para la cual se tomó un conjunto de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos, después se utilizó un comando apropiado para insertar en una base de datos vacía, después se tomó el tiempo de cuanto se demoró la operación, estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes.

Para la prueba en Cassandra, se utilizó en siguiente comando para insertar datos:

```
INSERT INTO prueba.users ( id, name, number)
VALUES ( 1, ‘Juan Perez’, 25156051 );
```

La Tabla 2 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación insertar en Cassandra.

Tabla 5. Resultados de la operación inserción en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	9.914	9.930	9.443	**9.762**
5000	20.303	21.816	21.166	**21.095**
10000	33.841	33.424	34.002	**33.756**
50000	146.734	147.426	148.565	**147.575**
100000	209.601	205.350	208.808	**207.920**

Para la prueba en MongoDB, se utilizó en siguiente comando para insertar datos:

```
db.prueba.insert (
  {
    _id: '579ece2ac691d91feafde781',
    created_on: '2015-05-10T02:36:25.338Z',
    value: 0.193427741171179
  }
);
```

La Tabla 3 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación insertar en MongoDB.

Tabla 6. Resultados de la operación inserción en MongoDB.

	Tiempo en Segundos

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

Registros	Prueba 1	Prueba 2	Prueba 3	Promedio
1000	0.008	0.006	0.007	**0.007**
5000	0.261	0.309	0.286	**0.285**
10000	1.263	1.258	1.471	**1.331**
50000	8.796	8.724	8.834	**8.785**
100000	15.001	15.002	15.022	**15.008**

Para las operaciones de inserción los resultados se muestran en la Figura 1. Existe una gran diferencia en tiempos promedio entre las dos bases de datos, por lo tanto, los resultados obtenidos favorecieron a MongoDB.

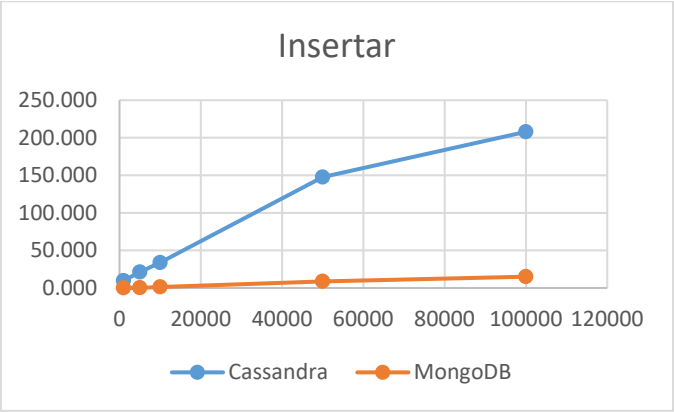


Figura 1. Resultados de las pruebas de inserción

En la segunda prueba se realizó la función de consultar, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una consulta. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes.

Para la prueba en Cassandra, se utilizó en siguiente comando para consultar datos:
SELECT * FROM users limit numeroRegistros;

Con este comando se pudo establecer el límite que va a tener la consulta. El numeroResgistros representa los datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas respectivamente.

La Tabla 4 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación consultar en Cassandra.

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

Tabla 7. Resultados de la operación consultar en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.088	0.077	0.079	**0.081**
5000	0.148	0.154	0.130	**0.144**
10000	0.219	0.207	0.245	**0.224**
50000	0.839	0.931	0.838	**0.879**
100000	1.884	1.823	1.637	**1.781**

Para la prueba en MongoDB, se utilizó en siguiente comando para consultar datos:

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

La Tabla 5 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación consultar en MongoDB.

Tabla 8. Resultados de la operación consultar en MongoDB

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.006	0.004	0.004	**0.005**
5000	0.008	0.006	0.006	**0.007**
10000	0.007	0.006	0.007	**0.007**
50000	0.070	0.040	0.060	**0.057**
100000	0.140	0.080	0.120	**0.113**

Para las operaciones de consulta los resultados se muestran en la Figura 2. Ambas bases de datos realizaron bien sus lecturas en cada keyspace/collection, por lo tanto, los resultados obtenidos favorecieron a MongoDB.

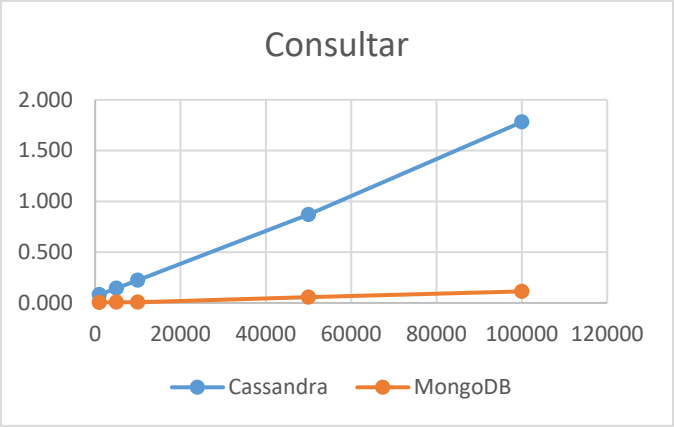


Figura 2. Resultados de las pruebas de consulta

En la tercera prueba se realizó la función de actualizar, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una actualización respectivamente. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes, para formar un resultado único.

Para la prueba en Cassandra, se utilizó en siguiente comando para actualizar datos:

```
UPDATE prueba.users SET name = 'Fabian Qhispe',
number = 1517284 WHERE id = 1;
```

La Tabla 6 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación actualizar en Cassandra.

Tabla 9. Resultados de la operación actualización en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	8.737	8.938	8.768	**8.814**
5000	20.429	20.614	20.931	**20.658**
10000	36.110	35.236	35.977	**35.774**
50000	150.701	152.382	152.046	**151.710**
100000	262.216	267.141	271.729	**267.029**

Para la prueba en MongoDB, se utilizó en siguiente comando para actualizar datos:

```
db.prueba.update (
  { _id: '57a79476599eca05ca646475' },
```

```
{ create_on: '2015-10-27T01:21:30.890Z' }
{ $push: { value: 0.181145454206268 } }
, true
);
```

La Tabla 7 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación actualizar en MongoDB.

Tabla 10. Resultados de la operación actualización en MongoDB.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	2.317	2.201	2.229	**2.249**
5000	10.557	11.425	11.427	**11.136**
10000	15.002	15.005	15.003	**15.003**
50000	15.002	15.003	15.004	**15.003**
100000	15.003	15.004	15.004	**15.004**

Para las operaciones de actualización los resultados se muestran en la Figura 3. Existe una gran diferencia en tiempos promedio entre las dos bases de datos, por lo tanto, los resultados obtenidos favorecieron a MongoDB

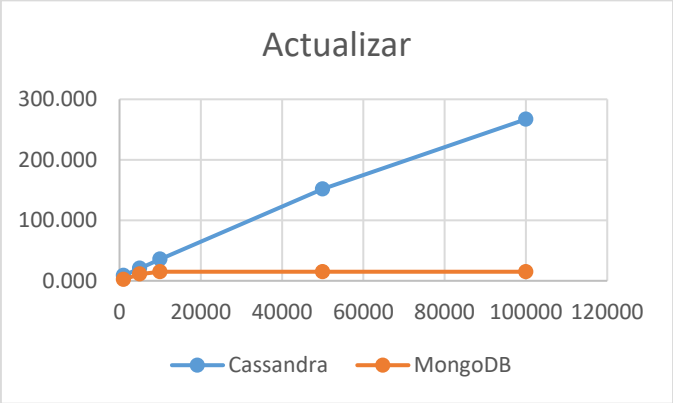


Figura 3. Resultados de las pruebas de actualización

En la última prueba se realizó la función de eliminación, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una

eliminación respectivamente. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes, para formar un resultado único.

Para la prueba en Cassandra, se utilizó en siguiente comando para eliminar datos:

TRUNCATE prueba.users;

La Tabla 8 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación eliminar en Cassandra.

Tabla 11. Resultados de la operación eliminación en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.811	0.667	0.733	**0.737**
5000	0.995	0.886	0.915	**0.932**
10000	0.943	1.038	1.096	**1.026**
50000	1.451	1.776	1.593	**1.607**
100000	2.515	1.991	2.371	**2.292**

Para la prueba en MongoDB, se utilizó en siguiente comando para eliminarr datos:

db.prueba.remove({});

La Tabla 9 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación eliminar en MongoDB.

Tabla 12. Resultados de la operación eliminación en MongoDB.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.056	0.057	0.059	**0.057**
5000	0.269	0.272	0.266	**0.269**
10000	0.609	0.641	0.662	**0.637**
50000	3.289	3.156	3.213	**3.219**
100000	5.208	5.096	5.169	**5.158**

Para las operaciones de eliminación los resultados se muestran en la Figura 4. Ambas bases de datos realizaron bien su función de eliminación en cada keyspace/collection, por

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304
lo tanto, los resultados obtenidos favorecieron a Cassandra.

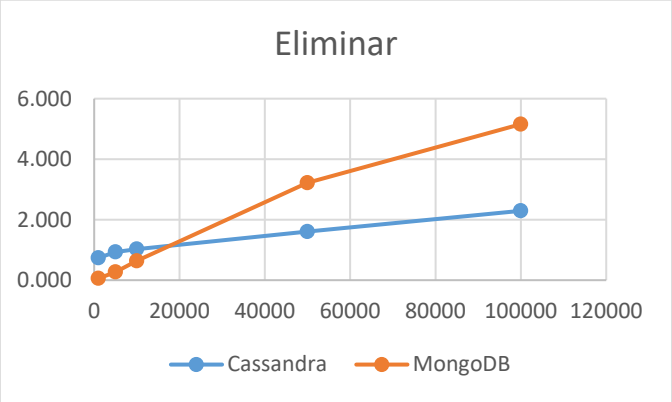


Figura 4. Resultados de las pruebas de eliminación

En general, los resultados sugieren que MongoDB es más rápido que Cassandra para las operaciones de inserción, actualización y consulta de datos. Para las operaciones de eliminación Cassandra fue más rápido que MongoDB. Estos resultados fueron razonables, ya que estas pruebas se realización en un solo equipo. Además no se requirió reuniones (todos los datos estaban en un mismo keysapce/collection) y no había costos de comunicación.

En la segunda prueba se realizó la función de consultar, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una consulta. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes.

Para la prueba en Cassandra, se utilizó en siguiente comando para consultar datos:

```
SELECT * FROM users limit numeroRegistros;
```

Con este comando se pudo establecer el límite que va a tener la consulta. El numeroResgistros representa los datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas respectivamente.

La Tabla 4 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación consultar en Cassandra.

Tabla 13. Resultados de la operación consultar en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.088	0.077	0.079	**0.081**

5000	0.148	0.154	0.130	**0.144**
10000	0.219	0.207	0.245	**0.224**
50000	0.839	0.931	0.838	**0.879**
100000	1.884	1.823	1.637	**1.781**

Para la prueba en MongoDB, se utilizó en siguiente comando para consultar datos:

db.prueba.find({});

La Tabla 5 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación consultar en MongoDB.

Tabla 14. Resultados de la operación consultar en MongoDB

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.006	0.004	0.004	**0.005**
5000	0.008	0.006	0.006	**0.007**
10000	0.007	0.006	0.007	**0.007**
50000	0.070	0.040	0.060	**0.057**
100000	0.140	0.080	0.120	**0.113**

Para las operaciones de consulta los resultados se muestran en la Figura 2. Ambas bases de datos realizaron bien sus lecturas en cada keyspace/collection, por lo tanto, los resultados obtenidos favorecieron a MongoDB.

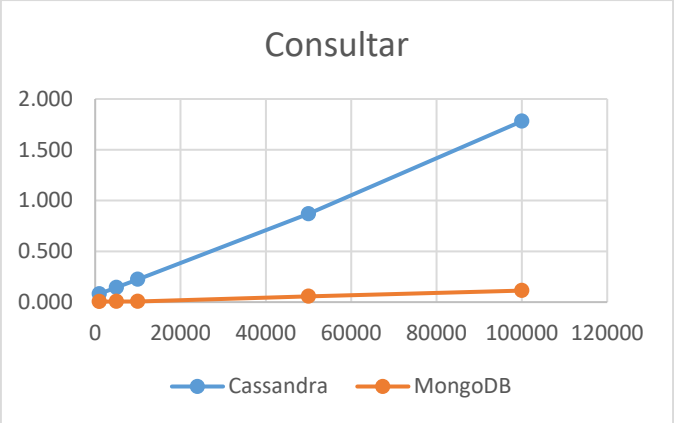


Figura 5. Resultados de las pruebas de consulta

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

En la tercera prueba se realizó la función de actualizar, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una actualización respectivamente. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes, para formar un resultado único.

Para la prueba en Cassandra, se utilizó en siguiente comando para actualizar datos:

```
UPDATE prueba.users SET name = 'Fabian Qhispe',
number = 1517284 WHERE id = 1;
```

La Tabla 6 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación actualizar en Cassandra.

Tabla 15. Resultados de la operación actualización en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	8.737	8.938	8.768	**8.814**
5000	20.429	20.614	20.931	**20.658**
10000	36.110	35.236	35.977	**35.774**
50000	150.701	152.382	152.046	**151.710**
100000	262.216	267.141	271.729	**267.029**

Para la prueba en MongoDB, se utilizó en siguiente comando para actualizar datos:

```
db.prueba.update (
   { _id: '57a79476599eca05ca646475' },
   { create_on: '2015-10-27T01:21:30.890Z' }
   { $push: { value: 0.181145454206268 } }
   , true
);
```

La Tabla 7 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación actualizar en MongoDB.

Tabla 16. Resultados de la operación actualización en MongoDB.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	2.317	2.201	2.229	**2.249**
5000	10.557	11.425	11.427	**11.136**
10000	15.002	15.005	15.003	**15.003**
50000	15.002	15.003	15.004	**15.003**
100000	15.003	15.004	15.004	**15.004**

Para las operaciones de actualización los resultados se muestran en la Figura 3. Existe una gran diferencia en tiempos promedio entre las dos bases de datos, por lo tanto, los resultados obtenidos favorecieron a MongoDB

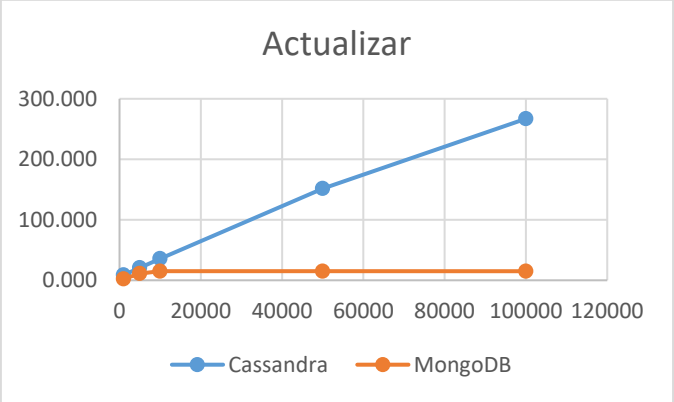


Figura 6. Resultados de las pruebas de actualización

En la última prueba se realizó la función de eliminación, para la cual se tomaron los mismos datos ya insertados en la primera prueba. En cada inserción de un conjunto de datos de 1.000, 5.000, 10.000, 50.000 y 100.000 columnas/documentos se realizó una eliminación respectivamente. Estas operaciones se repitieron 3 veces y al final se promediaron los 3 tiempos resultantes, para formar un resultado único.

Para la prueba en Cassandra, se utilizó en siguiente comando para eliminar datos:

TRUNCATE prueba.users;

La Tabla 8 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación eliminar en Cassandra.

Revista Publicando, 4 No 11. (1). 2017, 79-107. ISSN 1390-9304

Tabla 17. Resultados de la operación eliminación en Cassandra.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.811	0.667	0.733	**0.737**
5000	0.995	0.886	0.915	**0.932**
10000	0.943	1.038	1.096	**1.026**
50000	1.451	1.776	1.593	**1.607**
100000	2.515	1.991	2.371	**2.292**

Para la prueba en MongoDB, se utilizó en siguiente comando para eliminarr datos:

db.prueba.remove({});

La Tabla 9 muestra el tiempo resultante en segundos de las 3 pruebas, incluyendo el promedio final de la operación eliminar en MongoDB.

Tabla 18. Resultados de la operación eliminación en MongoDB.

	Tiempo en Segundos			
Registros	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Promedio**
1000	0.056	0.057	0.059	**0.057**
5000	0.269	0.272	0.266	**0.269**
10000	0.609	0.641	0.662	**0.637**
50000	3.289	3.156	3.213	**3.219**
100000	5.208	5.096	5.169	**5.158**

Para las operaciones de eliminación los resultados se muestran en la Figura 4. Ambas bases de datos realizaron bien su función de eliminación en cada keyspace/collection, por lo tanto, los resultados obtenidos favorecieron a Cassandra.

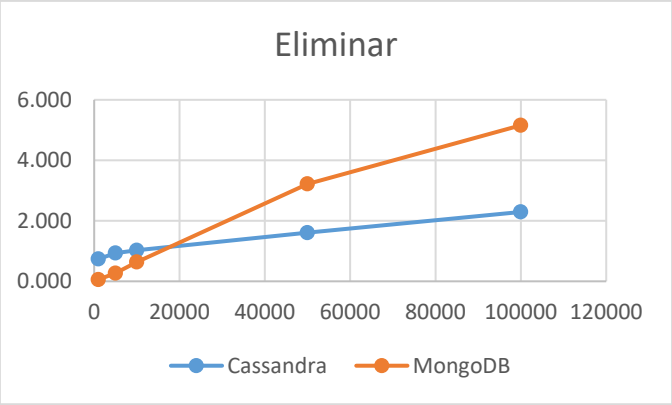


Figura 7. Resultados de las pruebas de eliminación

En general, los resultados sugieren que MongoDB es más rápido que Cassandra para las operaciones de inserción, actualización y consulta de datos. Para las operaciones de eliminación Cassandra fue más rápido que MongoDB. Estos resultados fueron razonables, ya que estas pruebas se realización en un solo equipo. Además no se requirió reuniones (todos los datos estaban en un mismo keysapce/collection) y no había costos de comunicación.

CONCLUSIONES

El desarrollo de la Web necesita bases de datos capaces de almacenar y procesar grandes datos con eficacia, la demanda de alto rendimiento al leer y escribir, así que la base de datos relacional tradicional enfrenta muchos retos nuevos. Bases de datos NoSQL han ganado popularidad en los últimos años y han tenido éxito en muchos sistemas de producción.

En este trabajo se analizó y evaluó dos de las más populares bases de datos NoSQL: MongoDB y Cassandra, estas bases de datos resultan ser más eficientes con el manejo de grandes cantidades de datos.

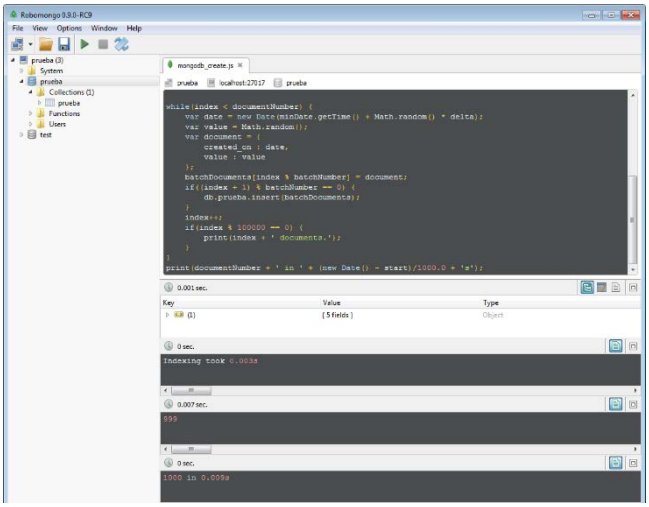
En los experimentos realizados, los tiempos registrados para las operaciones de CRUD (inserción, actualización, borrado y consultas) sobre un mismo keysapce/collection favorecieron a MongoDB, siendo este el más eficiente y rápido cuando se trata de base de datos no relacionales. Aunque se requieren experimentos más exhaustivos y muchos otros tipos de pruebas.

En conclusión MongoDB mostro mejores resultados para casi todas las pruebas realizadas, por lo cual, puede brindar mayor flexibilidad y puede proporcionar menor tiempo de ejecución independientemente del tamaño utilizado en la evaluación de una base de datos.

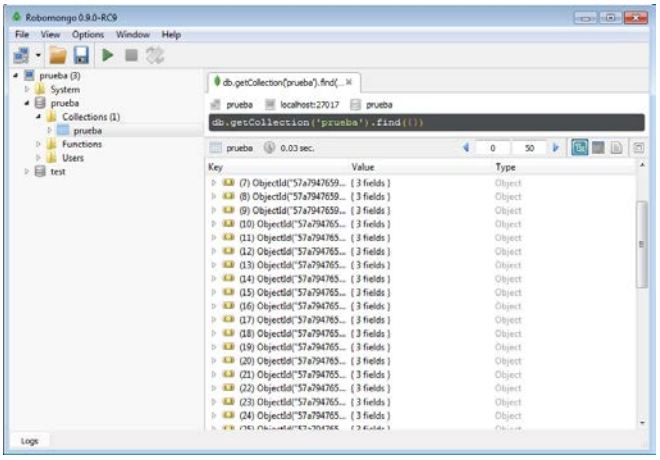
ANEXOS

MongoDB

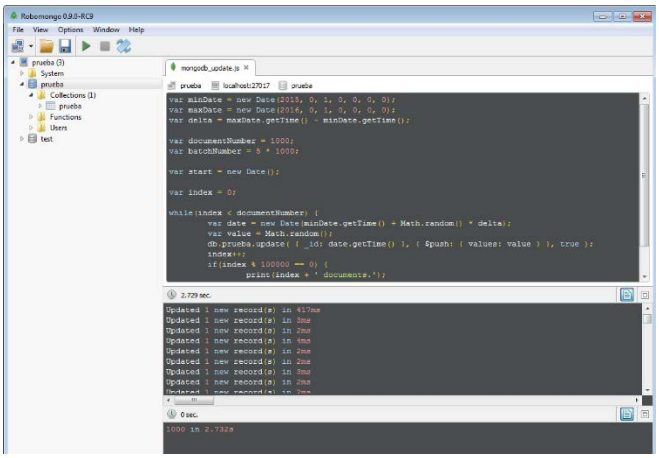
- **Pruebas de Inserción:**



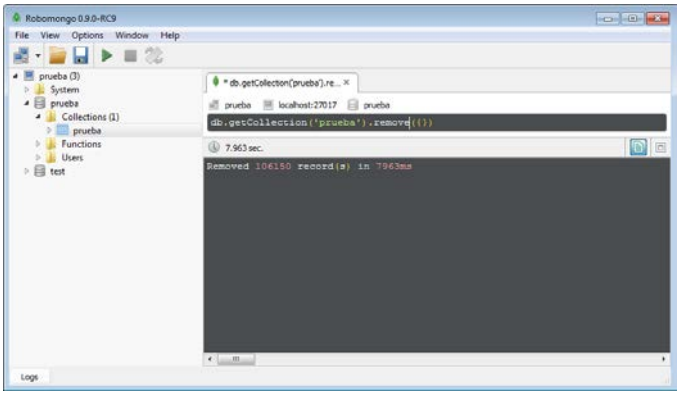
- **Pruebas de Selección:**



- **Pruebas de Modificación:**

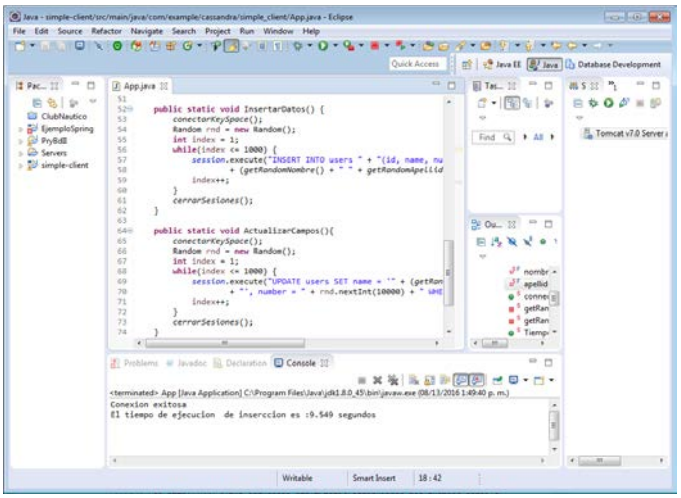


- **Pruebas de Eliminación:**

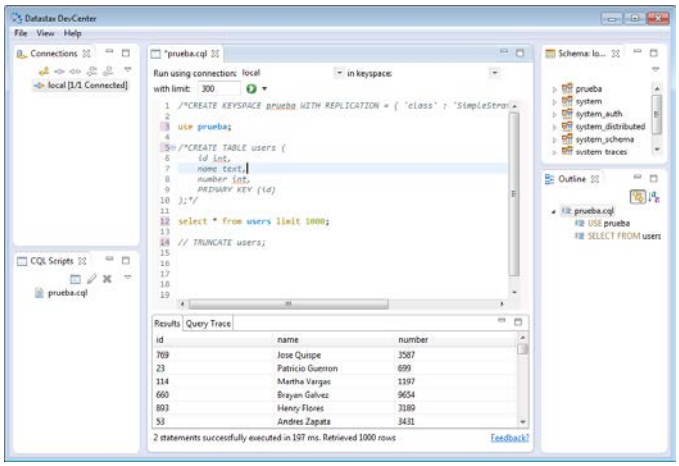


Cassandra

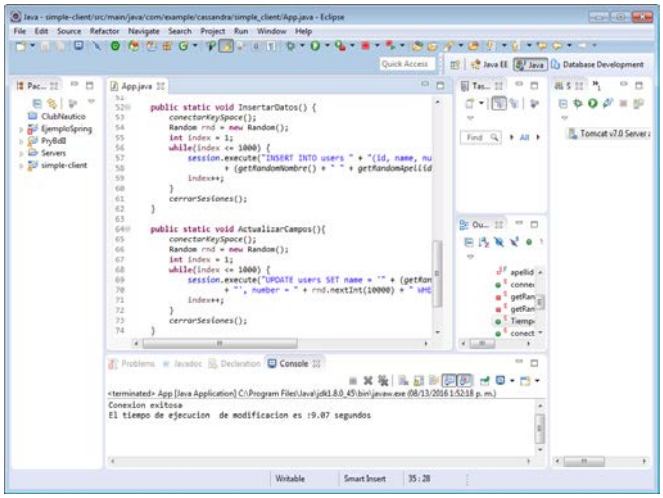
- **Pruebas de Inserción:**



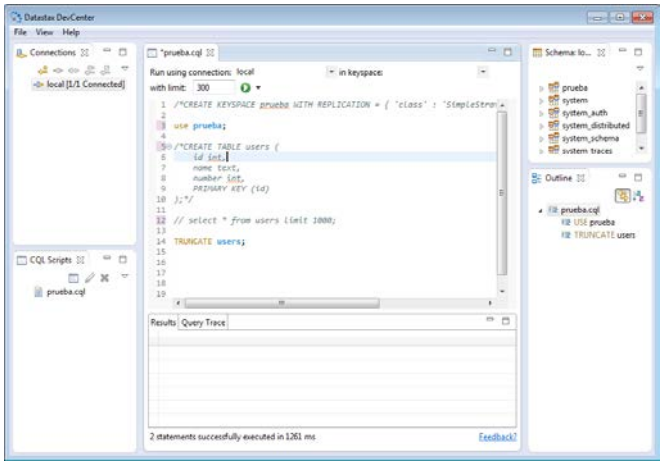
- **Pruebas de Selección:**



- **Pruebas de Modificación:**



- **Pruebas de Eliminación:**



2. REFERENCIAS BIBLIOGRÁFICAS

[1] Kuznetsov, S. D., & Poskonin, A. V. (2014). NoSQL data management systems. Programming and Computer Software, 40(6), 323-332.

[2] Moniruzzaman, A. B. M., & Hossain, S. A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison.arXiv preprint arXiv: 1307.0191.

[3] Zaki, A. K. (2014). NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges. International Journal of Research in Engineering and Technology (IJRET), 3(15), 403-409.

[4] Bhatewara, A., & Waghmare, K. (2012). Improving network scalability using NoSQL database. Int. J. Adv. Comput. Res. (IJACR), 2(6), 4.

[5] MongoDB for GIANT Ideas. (2016). MongoDB. Retrieved 6 June 2016, from http://www.mongodb.org/

[6] The Apache Cassandra Project. (2016). Cassandra.apache.org. Retrieved 6 June 2016, from http://cassandra.apache.org/

[7] Apache HBase – Apache HBase™ Home. (2016). Hbase.apache.org. Retrieved 6 June 2016, from http://hbase.apache.org/

[8] Edward, S. G., & Sabharwal, N. (2015). Introducing MongoDB. In Practical MongoDB (pp. 25-28). Apress.

[9] Dharmasiri, H. M. L., & Goonetillake, M. D. J. S. (2013, December). A federated approach on heterogeneous NoSQL data stores. In Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on (pp. 234-239). IEEE.

[10] Abramova, V., & Bernardino, J. (2013, July). NoSQL databases: MongoDB vs cassandra. In Proceedings of the International C* Conference on Computer Science and Software Engineering (pp. 14-22). ACM.

[11] Truică, C. O., Boicea, A., & Trifan, I. (2013). CRUD Operations in MongoDB. In Proceedings of the 2013 international Conference on Advanced Computer Science and Electronics Information, Ed. Atlantis Press.

[12] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. (2011, November). Security issues in nosql databases. In 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (pp. 541-547). IEEE.

[13] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In Pervasive computing and applications (ICPCA), 2011 6th international conference on (pp. 363-366). IEEE.

[14] Chandra, D. G., Prakash, R., & Lamdharia, S. (2012, November). A study on cloud database. In Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on (pp. 513-519). IEEE.