Evaluación de diferentes bases de datos NoSQL aplicado a una plataforma de Entrenamiento Auditivo para Músicos

Agustín Queirolo
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
agustinqu@gmail.com

Franco Wanseéle
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
franco.wanseele@gmail.com

***Resumen*—En este proyecto se partió de la interrogante de cómo saber qué Base de Datos No Relacional es la mas adecuada o para una situación dada. Para ello se plantea la realidad del proyecto de grado de uno de los integrantes del grupo, una aplicación para el entrenamiento auditivo para músicos. Para un mayor análisis de la realidad se utiliza un Modelo Entidad Relación (MER) y así tener un arco más claro para poder realizar una evaluación de los diferentes atributos de calidad señalados en el paper "*Choosing the right NoSQL database for the job: a quality attribute evaluation*" [João Ricardo Lourenço(2015)]. De esta evaluación se llegó a la conclusión de que *Cassandra* y *MongoDB* serían las mejores opciones. Partiendo de dichos resultados se analizó de forma práctica la eficiencia de las bases de datos NoSQL mencionadas. De aquí se obtuvo que *MongoDB* es más eficiente cuando se realiza una cantidad grande de consultas de lectura, las cuales serán operaciones predominantes en la realidad planteada.**

Luego de haber elegido *MongoDB*, se pasó a modelar la realidad y realizar consultas que serán necesarias para el proyecto de grado.

I. Introducción

La presente investigación forma parte del proyecto de grado de uno de los autores, la cual consistirá en el modelado de una aplicación de entrenamiento auditivo para músicos. Previo al modelado se deberá analizar la realidad presentada para saber cuales serán los principales flujos de uso y de esta manera saber qué base de datos NoSQL elegir. Esta elección se realizará en base al análisis, no solamente de rendimiento, sino de diversos atributos de calidad.

La investigación se realizará en base al paper "*Choosing the right NoSQL database for the job: a quality attribute evaluation*" [João Ricardo Lourenço(2015)] y se analizarán las bases de datos no relacionales que allí se presentan. Posteriormente se realizará un análisis de rendimiento basado en la herramienta "*Yahoo Cloud Serving Benchmark*" y así obtener un análisis más minucioso en cuanto al desempeño, haciendo énfasis en las cualidades de interés, según el análisis de la realidad planteado.

II. Realidad planteada

La plataforma consistirá en una aplicación móvil mediante la cual ciertos usuarios tendrán la posibilidad de entrenar su oído musical.

El foco de la aplicación está en que existan docentes que puedan subir a la aplicación configuraciones musicales (asignadas a un curso) y posteriormente, cuando un estudiante de dicho curso acceda a esta configuración, la aplicación genere de forma aleatoria dictados melódicos, en base a la configuración mencionada. La idea será que el estudiante tenga la opción de generar la cantidad de dictados que desee ya que con los mismos podrá practicar de la siguiente manera: el estudiante podrá reproducir los dictados generados y deberá escribirlo en un pentagrama. Una vez finalizado se tendrá la opción de ver solución, para lo cual la aplicación graficará en un pentagrama el dictado melódico reproducido, teniendo el estudiante la posibilidad de comparar su resultado con la solución correcta.

Por otro lado se deberá gestionar los cursos, módulos, estudiantes, docentes e institutos. Cada curso estará compuesto por diferentes módulos, los cuales contendrán las configuraciones de los dictados. En base a estas últimas es que se generarán los dictados de forma aleatoria, los cuales los estudiantes podrán escuchar e intentar resolver de forma reiterada, teniendo una calificación por cada intento.

Los usuarios de la aplicación pueden ser docentes o estudiantes, para los cuales pueden dictar o cursar uno o más cursos respectivamente. Por otro lado, un docente podrá pertenecer a uno o más institutos y a su vez, cada curso podrá estar asociado a un instituto. Cabe destacar que la aplicación contemplará el caso de docentes particulares, es decir, docentes que no pertenezcan a ningún instituto pero sí puedan crear cursos. Además existirá el concepto de curso personal, el cual todo usuario tendrá uno por defecto. Dicho curso no es dictado ni cursado por nadie, la idea es que solamente sea accesible por el propietario, para que dentro del mismo se puedan crear configuraciones (tanto los docentes como los estudiantes) para poder testear las configuraciones antes de subirlas o incluso poder entrenarse a uno mismo (en caso de ser estudiante).

Los cursos y los módulos tendrán un nombre y una descripción correspondiente a cada uno. Las configuraciones de los dictados tendrán un nombre, una descripción, un conjunto

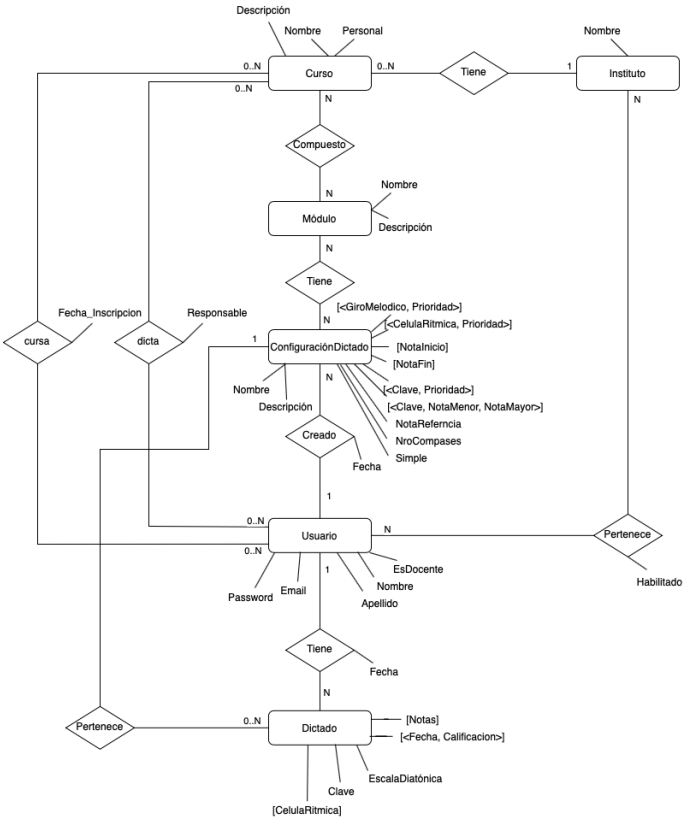

Figura 1: Modelo Entidad Relación (MER).

de giros melódicos asociados con una prioridad para dicha configuración, un conjunto de células rítmicas (las cuales serán un conjunto de figuras musicales) asociadas con una prioridad para dicha configuración, las claves de sol y fa, cada una con una cierta prioridad para dicha configuración y para cada una de las claves (sol y fa) se tendrá la nota más baja y la nota más alta entre las cuales se generará el dictado. Además se tendrá un conjunto de notas de inicio y notas de fin (notas en las cuales el dictado podrá comenzar o terminar), una nota de referencia, el número de compases que tendrá el dictado y si dicho dictado es simple o compuesto (esto determinará qué tipos de figuras podrá contener el dictado). Los usuarios tendrán un nombre, apellido, email y una password (la cual se guardará cifrada). Cada dictado se generará en una cierta fecha, y el mismo tendrá un conjunto de notas, un conjunto de células rítmicas, una clave, una escala diatónica y se guardará la calificación y la fecha de todos los intentos que el estudiante intentó escribir dicho dictado melódico. Además los docentes podrán ser responsables de un curso, y los estudiantes tendrán una fecha de inscripción a los cursos. El instituto tendrá un nombre y cada docente deberá ser habilitado para pertenecer a un instituto.

Para dicha realidad se realizó el modelo entidad relación (MER) presentado en Figura 1. A modo de complementar dicho modelado, se detallan a continuación las siguientes restricciones no estructurales:

- Un usuario no puede cursar y dictar una misma materia
- No puede haber docentes o estudiantes que dicten o cursen un curso personal

III. Motivación para usar BDNR

La aplicación presentada tiene el potencial de ser utilizada por un gran número de institutos educativos, docentes y estudiantes por lo que se quiere que el modelado de dicha realidad pueda soportar un gran número de usuarios, lo cual conlleva una gran cantidad de datos en la base de datos. Las bases de datos relacionales, según lo visto en el curso, no son ideales para manejar un sistema con gran escalabilidad. Esto motiva a elegir una BDNR para el proyecto de grado.

Por otro lado, si bien la principal funcionalidad de la aplicación será la de generar dictados melódicos, se entiende que en el contexto de enseñar música se podría querer incorporar nuevas funcionalidades como pueden llegar a ser ejercicios de diferente índole dentro de la música. Es por esto que se elige una base de datos no relacional, ya que éstas permiten cierta flexibilidad al momento de incorporar nuevos cambios que puedan impactar en el modelado de la base de datos.

IV. Marco teórico

Las bases de datos no relacionales, según el artículo [João Ricardo Lourenço(2015)] se dividen en cuatro categorías diferentes, según su modelo de datos y almacenamiento: forma de almacenamiento clave-valor, almacenamiento de documentos, almacenamiento de columnas y bases de datos de grafos. Esta clasificación se debe a que cada tipo de base de datos ofrece diferentes soluciones para contextos específicos.

La presente investigación se centrará en analizar diferentes bases de datos no relacionales, dejando de lado las relacionales, analizando, no solamente el rendimiento, sino diversos atributos de calidad. Dicho análisis será aplicado a la realidad planteada y así poder elegir una o más de una base de datos no relacional, dependiendo del caso concreto del problema al que se quiera apuntar.

Este análisis comenzará analizando el teorema CAP [João Ricardo Lourenço(2015)] el cual establece que ningún sistema distribuido puede garantizar simultáneamente consistencia, disponibilidad y tolerancia al particionamiento, explicadas a continuación:

- La **consistencia** consiste en garantizar que la lectura de datos retornará la escritura más reciente, para un registro dado. Eso quiere decir que siempre que se haga alguna modificación en algún dato, dicho cambio debe reflejarse en todos los nodos de la base de datos, garantizando que siempre se accede a la información correcta desde cualquiera de dichos nodos.
- La **disponibilidad** consiste en que un nodo en funcionamiento debe retornar siempre una respuesta razonable en un período de tiempo razonable.
- La **tolerancia al particionamiento** se refiere a que el sistema debe seguir funcionando a pesar de que algunos nodos no se encuentren disponibles, ya que la información es consistente en todos los nodos.

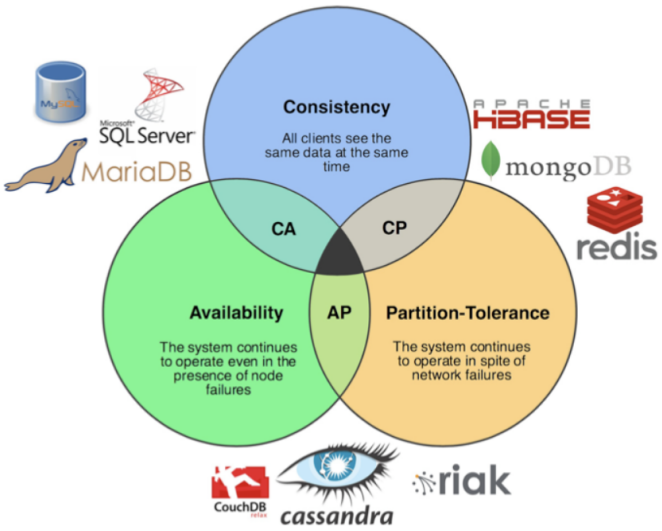

Figura 2: Imagen tomada del artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*"

Como se puede ver en la Figura 2, y como además se explica en dicho artículo, muchas bases de datos "eligen" ser CA, CP o AP, dependiendo qué aspectos desean garantizar con mayor solidez.

V. Evaluación de atributos de calidad

Como primera evaluación, se tomarán las bases de datos no relacionales tratadas en el artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*" [João Ricardo Lourenço(2015)] las cuales son: *Aerospike, Cassandra, Couchbase, CouchDB, HBase, MongoDB y Voldemort*.

V-A. Disponibilidad y Consistencia

En el artículo en el cual se basa el presente trabajo menciona que las bases de datos NoSQL tienen como objetivo lograr un alto grado de disponibilidad en lugar de ofrecer coherencia. Ambos términos son de cierta forma opuestos ya que si se aumenta el grado de disponibilidad disminuye la garantía de consistencia.

Analizando ambos atributos con la realidad que se tiene como objeto de estudio, en primer lugar, la coherencia no es algo que se requiera (al menos en un alto grado) para las funcionalidades centrales de la aplicación, las cuales van a ser la generación y reproducción de dictados melódicos. En este punto cabe recordar que la consistencia se considera como la premisa de que todos los nodos ven los mismos datos al mismo tiempo. Teniendo esto en cuenta, la configuración de un dictado que esté disponible en un curso no deberá de ser modificada, ya que si fuese así, en caso de que un estudiante hubiese generado dictados con la configuración vieja, quedará inconsistente con la nueva configuración modificada. Por tal motivo, una vez queden disponibles estos datos, los mismos sólo serán accedidos para ser leídos y en base a esos generar los dictados propiamente dichos, para los cuales no se necesita un alto grado de consistencia.

Desde el lado de la disponibilidad, sería algo deseable ya que, debido a que el algoritmo que genera los dictados podría eventualmente consumir un alto grado de procesamiento (ya que depende directamente de la complejidad de su configuración para generarlo), el poder acceder a la configuración de los dictados en un tiempo razonable disminuiría el tiempo total en que se demore construir los dictados.

Posibilidades: Según el artículo existen diferentes bases de datos no relacionales las cuales ofrecen diferentes formas de ajustar el equilibrio entre consistencia y disponibilidad, para las cuales se mencionan *Dynamo, Cassandra, CouchDB y MongoDB*. Es importante destacar que en este punto que para la realidad planteada, podría ser favorable una base de datos que tenga una mayor disponibilidad y no tanta consistencia, no es de las principales características que se va a priorizar al momento de elegir una. Por tal motivo va a ser interesante considerar las bases de datos mencionadas las cuales permiten un cierto ajuste en este punto.

V-B. Durabilidad

La durabilidad consiste en que los datos validados se envían al disco luego de una transacción exitosa. A pesar de que la consistencia y la durabilidad están relacionadas, ya que, si un sistema sufre problemas de consistencia, su durabilidad también se verá afectada, teniendo posibles pérdidas de datos, podría ser beneficioso poder maximizar este requisito.

Como se mencionó anteriormente, las funcionalidades principales de la aplicación van a consistir en consultas y no tanto en modificación de datos. El punto interesante es que, si bien no se van a estar modificando una gran cantidad de datos, si se van a estar insertando dictados, y esto se va a hacer con una frecuencia bastante considerable. Este es el motivo por el cual se querrá tener un buen desempeño en este atributo. En este punto se destaca que los sistemas documentales tienen como objetivo la durabilidad al utilizar un sistema de versión única. Dado esto, *MongoDB* es una buena opción de cara a tener una buena durabilidad en el sistema. Además, dicho sistema de base de datos es quien pierde menos datos en caso de falla del nodo cuando se utiliza la replicación asincrónica, según estudios presentados en el artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*". *Cassandra* se encuentra en segundo lugar y *Aerospike* y *Couchbase* son quienes pierden grandes cantidades de datos, según dicho estudio, el cual abarcó las bases de datos mencionadas.

V-C. Escalabilidad

La escalabilidad trata sobre la habilidad de un sistema de manejar una carga de trabajo mayor. Las Bases de Datos No Relacionales han sido creadas con el objetivo de mejorar la escalabilidad, por lo que todas presentan mejor escalabilidad que SQL. En la realidad planteada, si bien es un proyecto de grado y originalmente no se sabe con certeza el número de usuarios de la aplicación, se considera que, por el hecho de elegir una base de datos NoSQL, esté atributo será contemplado.

V-D. Performance

Las bases de datos NoSQL se dividen principalmente en dos categorías: lectura y escritura.

Las bases de datos de tipo *Column Store* son más optimizadas para operaciones de escritura, por lo que son mejores en este ámbito que los otros tipos de BDNR. De todas las BDNR mencionadas, *Cassandra* al ser de tipo *Column Store* fue construida para ser *write-optimized*, por lo cual tiene un mejor desempeño a la hora de hacer operaciones de escritura.

Dada la realidad planteada, las operaciones de lectura serán un atributo el cual se va a querer optimizar. Las funcionalidades principales de la aplicación, las cuales serán las relacionadas a la generación y reproducción de dictados, van a requerir estar haciendo lecturas constantemente a la base de datos. Para la generación de dictados, si bien se va a estar escribiendo en la base de datos, se va a tener que leer de base de datos la configuración. Además, si bien se van a guardar los dictados, cada vez que se quiera reproducir un dictado por parte del estudiante se va a tener que leer los metadatos del dictado para así poder construir el archivo multimedia para reproducirlo. Es por este motivo que la lectura será algo de gran importancia para el desempeño de la aplicación. Por otro lado, como se mencionó, la escritura, si bien estará presente en la generación de dictados, no es algo de suma importancia, ya que si dicha operación no es eficiente y tiene períodos de demora, la reproducción de dictados por parte del estudiante no se va a ver muy retrasada. Además, en el momento que el docente desee guardar la configuración de un dictado, no será de gran importancia que esta operación se haga de manera rápida, ya que no entorpece el uso de la aplicación.

V-E. Fiabilidad

La fiabilidad trata sobre la probabilidad del sistema de operar sin fallas durante un periodo de tiempo dado. Las BDNR que utilizan técnicas de una sola versión como Redis, Couchbase, MongoDB o Neo4J son más fiables ya que resuelven fiablemente el problema de operaciones de escritura concurrentes. Acerca de la consistencia, *MongoDB, CouchDB, Neo4J, Cassandra y HBase* ofrecen buenas garantías. En conclusión si uno quiere buena fiabilidad en temas de concurrencia, las mejores opciones son *MongoDB y Neo4J*.

V-F. Conclusión y resumen sobre atributos de calidad

Primero que nada cabe destacar que no todos los atributos de calidad mencionados en el artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*" [Chakraborttii(2015)] fueron contrastados con la realidad planteada ya que se seleccionaron los que realmente se considera que tendrán un impacto en el desempeño de la aplicación.

En las Figura 3 y Figura 4 se pueden observar algunas tablas presentes en el artículo en el cual se basa el presente trabajo. Esta última tabla habla de las características de cada una de las bases de datos no relacionales que se han mencionado previamente, mientras que la otra tabla ofrece un breve resumen de los atributos discutidos.

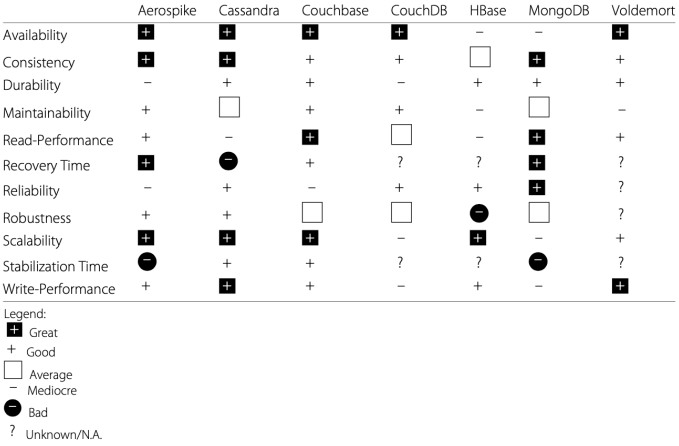

Figura 3: Imagen tomada del artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*".

Al haber analizado todos los atributos pertinentes, se considera que una buena opción sería utilizar *MongoDB* como base de datos del sistema. Esto se debe a que posee buena consistencia, fiabilidad, escalabilidad y durabilidad. Por otro lado, *MongoDB* tiene un gran desempeño a la hora de realizar operaciones de lectura, el tipo de operaciones que se supone serán más comunes en el sistema y, si se eligiera una BDNR con un mal desempeño se podría llegar a entorpecer la aplicación. Por último, *MongoDB* es una BDNR usada en el curso y por lo tanto conocida; usar una BDNR conocida en el proyecto de grado reduciría la complejidad de un proyecto suficientemente complejo.

Por otro lado se maneja la opción de utilizar la base de datos de *Cassandra* ya que presenta atributos de calidad similares a los mencionados para *MongoDB*, pero además presenta una fortaleza en el desempeño de la disponibilidad.

VI. Análisis de eficiencia

La presente sección estará basada en el artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*", [Chakraborttii(2015)] en el cual se presenta la herramienta *Yahoo Cloud Serving Benchmark* permite comparar el rendimiento relativo de los sistemas de gestión de bases de datos NoSQL. Aunque se estudió y aprendió como utilizar la herramienta YCSB

La presente sección estará basada en el artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*", [Chakraborttii(2015)] en el cual se presenta la herramienta *Yahoo Cloud Serving Benchmark* permite comparar el rendimiento relativo de los sistemas de gestión de bases de datos NoSQL. Aunque se estudió y aprendió como utilizar la herramienta YCSB, hasta llegamos a correr pruebas con ella, consideramos que es mejor basar nuestras conclusiones en los datos provistos por el paper por la abundancia de estos datos y por lo bien graficados que están.

En el artículo mencionado se puede ver un análisis comparativo de MongoDB, Cassandra, Redis y OrientDB. El análisis

	Aerospike	Cassandra	Couchbase	CouchDB	HBase	MongoDB	Voldemort
Category	Key-Value	Column-Store	Document-Store	Document-Store	Column-Store	Document-Store	Key-Value
CAP	AP	AP/CP	CP	AP	CP	CP	AP
Consistency	Configurable (several options)	Configurable (several options)	Eventual Consistency	Eventual Consistency	Configurable (strong and eventual consistency)	Configurable (several options)	Read-Repair (client handles conflicts)
Durability	Notified written to replica nodes	Configurable (several options)	Configurable (several options)	Configurable (notified written to at least one disk)	Configurable (several options)	Configurable (several options)	Notified written to desired nodes
Querying	Internal API	Internal API, SQL like (CQL)	Internal API (MapReduce)	Internal API (MapReduce)	Internal API	Internal API, MapReduce, complex query support	Internal API (get, put delete)
Concurrency Control	Read-commited isolation level (support for optimistic concurrency control)	MVCC	MVCC (application can select Optimistic or Pessmistic locking)	MVCC (application can select Optimistic or Pessmistic locking)	Optimistic locking with MVCC	Master-slave with multi-granularity locking	Optimistic locking with MVCC
Partitioning Scheme	Proprietary (Paxos based)	Consistent Hashing	Consistent Hashing	Consistent Hashing (third party)	Range Based	Consistent Hashing	Consistent Hashing
Native Partitioning	Yes	Yes	Yes	No	Yes	Yes	Yes

Figura 4: Imagen tomada del artículo "*Choosing the right NoSQL database for the job: a quality attribute evaluation*".

se hará sobretodo entre Mongo y Cassandra ya que son las bases de datos no relacionales que mejor desempeño tienen, según el análisis presentado anteriormente.

VI-A. MongoDB

Para el análisis de la base de datos no relacional MongoDB se obtuvo dos gráficas en donde se varía la proporción entre operaciones de lectura y operaciones de inserción o escritura; y cómo esto cambia el rendimiento (*throughput*), lo cual se encuentra en la Figura 5 y Figura 6. También se distingue por color la cantidad de operaciones realizadas.

En estas gráficas mencionadas se puede apreciar que cuando la cantidad de operaciones en chica (1000, 10.000), el rendimiento no cambia significativamente a medida que aumenta la proporción de operaciones de lectura. Sin embargo, cuando la cantidad de operaciones es alta, el rendimiento aumenta considerablemente a medida que aumenta la proporción de lecturas.

Por otro lado se tienen las gráficas presentes en la Figura 7 y Figura 8. La primera trata de la proporción entre las operaciones de lectura y modificación mientras que la segunda es sobre la proporción de operaciones "Scan" e inserción.

La gráfica presente en la Figura 7 es igual a las dos anteriores, mostrando que el throughput aumenta cuando la proporción de operaciones de lectura aumenta si la cantidad de operaciones es grande. En la Figura 8 se observa una disminución de rendimiento cuando la proporción de operaciones "Scan" aumenta.

Al analizar estas cuatro gráficas se puede inferir que MongoDB tiene un mejor rendimiento cuando la mayoría de las operaciones son de tipo lectura.

VI-B. Cassandra

En la Figura 9, 10, 11 y 12 se puede observar cuatro gráficas análogas a las presentadas para MongoDB.

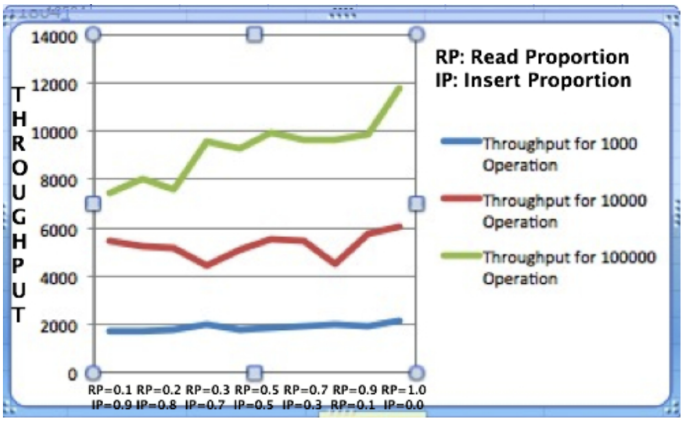

Figura 5: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

En estas gráficas se puede apreciar una curva cóncava y, casi siempre, descendiente. Por lo que el rendimiento es menor cuando la proporción de operaciones "Read" o "Scan" es menor (salvo en la figura 10 en las curvas roja y verde).

VI-C. Comparación de Desempeño

En el artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*"también se incluyen varias gráficas mostrando un análisis comparativo con cada base de datos, donde se analiza tanto el tiempo de ejecución como el rendimiento. En las gráficas de las Figuras 13 y 14 varía el tipo de operaciones y su proporción. Para la realidad planteada, y en base al análisis realizado en secciones anteriores, se sabe que se requerirá muchas consultas de lectura y no tantas del tipo de inserción; por tal motivo se

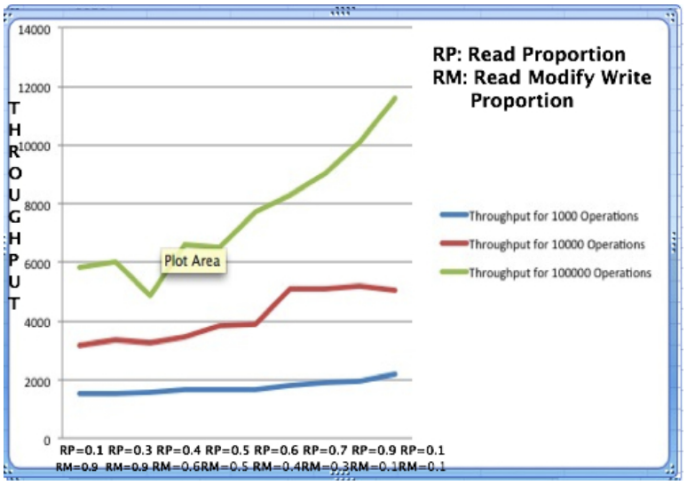

Figura 6: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

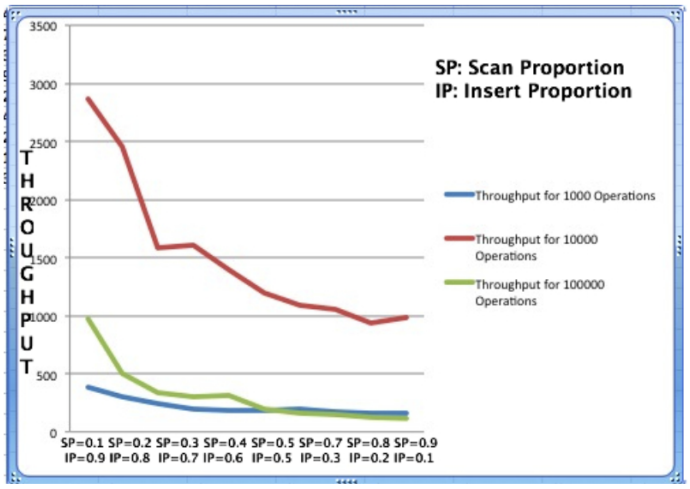

Figura 8: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

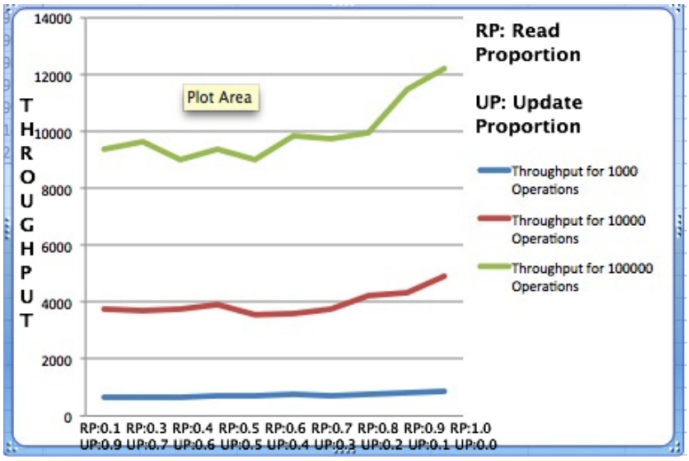

Figura 7: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

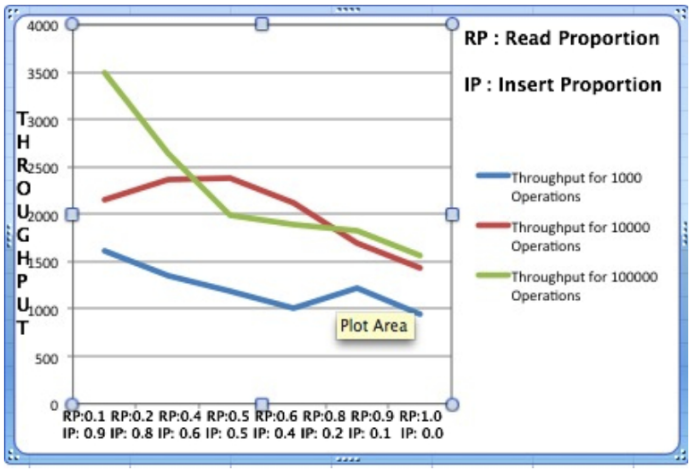

Figura 9: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

elige analizar la gráfica, en donde el 90 % de las operaciones son de tipo lectura y el resto de tipo inserción.

Se observa como Cassandra, a medida que crece el número de operaciones, ve su tiempo de ejecución aumentar y su rendimiento mantenerse prácticamente igual. Por otro lado, MongoDB bajo la misma situación ve su tiempo de ejecución aumentar mucho menos (e igual al resto de las BDNR en la gráfica) y su rendimiento aumenta considerablemente.

Al ver los resultados de este análisis se confirma la decisión por MongoDB como base de datos no relacional para la realidad planteada, ya que tiene un buen desempeño al hacer una gran cantidad de operaciones de lectura.

VII. Modelado de la realidad

Algunas consideraciones a tener en cuenta al momento de modelar la realidad planteada es que se va a querer tener cierto tipo de datos ya dados de alta en la aplicación para que, al momento de construir una configuración, dichos datos sean listados y sean simplemente seleccionados de una lista por el usuario. Estos van a ser las escalas diatónicas, los giros melódicos y las células rítmicas. Al momento de realizar una configuración o de generar un dictado, se manejará una copia de dichos datos (en la colección configuración y dictado respectivamente) y no una relación. Esto es porque, si una célula rítmica es utilizada en un dictado y si en algún momento dicha célula rítmica es modificada, si este cambio modifica la configuración del dictado, los dictados que fueron generados con la configuración vieja quedarían inconsistentes con la nueva configuración. Lo mismo sucedería para el resto de los datos mencionados.

Teniendo en cuenta esto último, sumado al análisis de la realidad realizado anteriormente es que se realizó un modelado

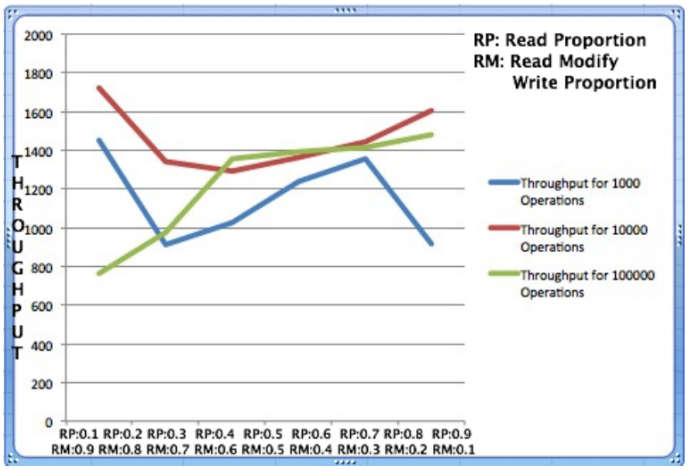

Figura 10: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

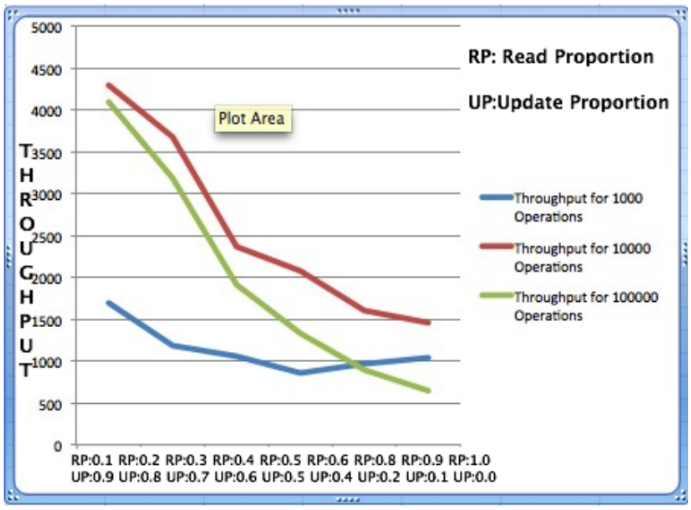

Figura 11: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

de la realidad usando MongoDB. Este modelado, que puede ser encontrado en el anexo, contiene las siguientes colecciones:

- Curso
- Modulo
- ConfiguracionDictado
- Dictado
- Usuario
- Instituto
- GiroMelodico
- CelulaRitmica
- EscaladaDiatonica

VIII. Creación de Consultas

Luego de haber terminado el modelado de la base de datos, se pensó cuales serían las consultas más realizadas

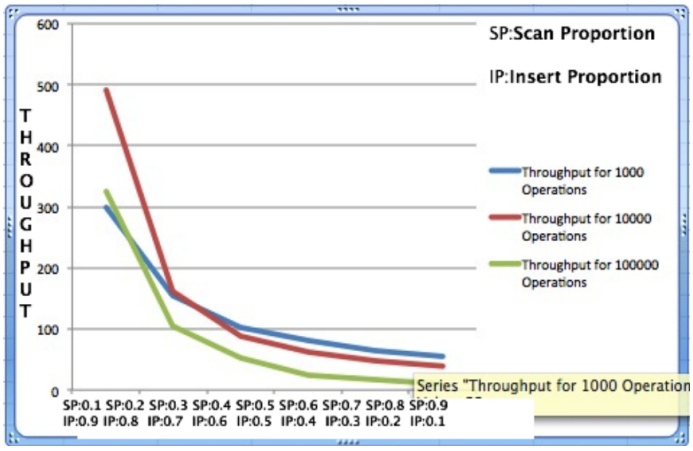

Figura 12: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

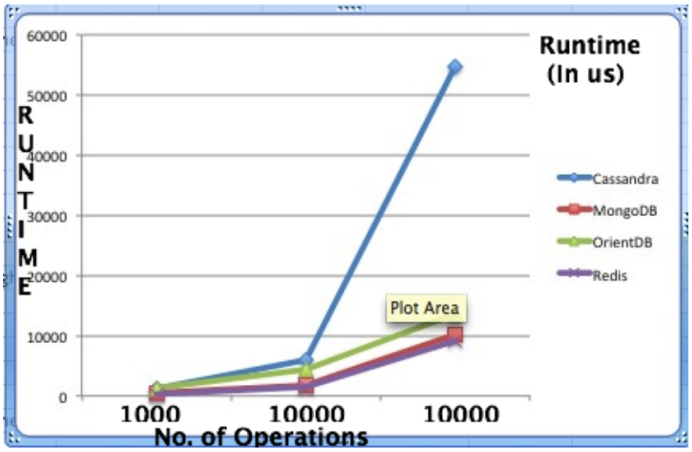

Figura 13: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

por el sistema. Luego de haber decidido esas consultas, se pasó a implementarlas y testearlas. A continuación se pueden observar todas las consultas ideadas, las cuales se considera son las mas representativas:

Se quiere obtener los cursos de un usuario, teniendo el userId:

```
db.Usuario.findOne(
{_id: userId},
{_id:0, cursos: 1 }
)
```

Esta consulta solamente obtiene un arreglo con los id de los cursos, si se quiere obtener mas información se puede usar la siguiente consulta:

```
db.Usuario.aggregate(
[
{$match: {_id: userId}},
```

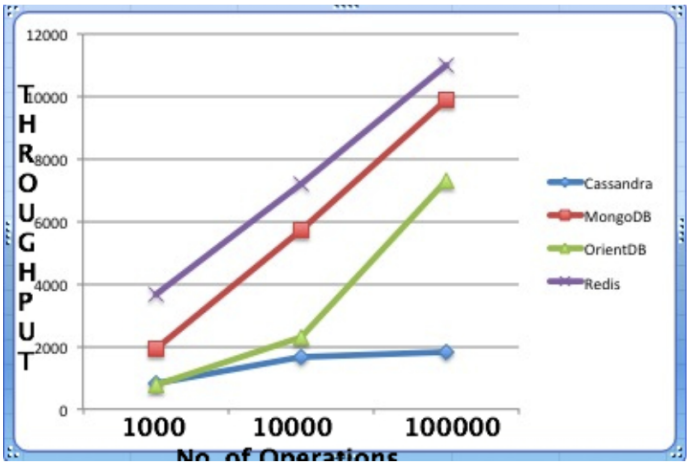

Figura 14: Imagen tomada del artículo "*Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*".

```
{$lookup: {
    from: "Curso",
    localField: "cursos",
    foreignField: "_id",
    as: "cursos"}
    },
{$project: {_id: 0, nombre: 1, cursos:1}}
])
```

Se quiere obtener los detalles una configuración de dictado, teniendo la configuracionDictadoId:

```
db.ConfiguracionDictado.findOne(
{ _id: configuracionDictadoId }
)
```

Se quiere obtener todos los dictados de un usuario (filtrado por una configuración), teniendo el userId:

```
db.Usuario.findOne(
{ _id: userId},
{ _id: 0, dictados: 1 }
).toArray()
```

Esta consulta solamente obtiene un arreglo con los id de los dictados, si se quiere obtener mas información se puede usar la siguiente consulta:

```
db.Usuario.aggregate(
[
{$match: {_id: userId}},
{$lookup: {
from: "Dictado",
localField: "dictados",
foreignField: "_id",
as: "dictados"}},
{$project: {_id: 0, nombre: 1, dictados:1}}
])
```

Se quiere insertar un dictado, teniendo los datos en la variable insertData:

```
db.Dictado.insert( insertData )
```

También se tiene que insertar el dictado en la colección Usuario.

```
db.Usuario.update(
{ _id: userId },
{ $push: { dictados: dictadoData }}
)
```

Se quiere actualizar la configuración de un dictado, teniendo dictadoId y nuevaData:

```
db.Dictado.update(
{ _id: dictadoId},
{ $set: { nuevaData }}
)
```

VIII-A. Desempeño de las consultas

De las consultas anteriores, las que serán utilizadas con mayor frecuencia serán obtener los cursos y obtener los dictados, ambas operaciones de lectura.

Seguido de esto, las operaciones de consultar la configuración de un dictado e insertar un dictado serán las segundas más utilizadas. Si bien esta involucrada una operación de inserción, la misma estará siempre acompañada de una operación de lectura, ya que en base a la configuración es que se generará el dictado de forma aleatoria.

Por último se tiene la operación de actualizar una configuración de un dictado. Esta operación será de las menos utilizadas ya que, como se explico en secciones anteriores, una vez que un estudiante genere dictados en base a una configuración, esta última no podrá ser modificada.

Según lo mencionado, se considera que en la sección "Modelado de la realidad"se cubren las operaciones que son de interés para la realidad abordada.

IX. Conclusiones

Se han analizado varios atributos para definir qué base de datos no relacional será la más indicada para la realidad planteada. En primer lugar se abordó la realidad explícitamente en texto, posteriormente se realizó un Modelo Entidad-Relación para después modelar la realidad. Se han analizado atributos de calidad y también se han contemplado los resultados de un análisis de eficiencia hecho con *Yahoo! Cloud Serving Benchmark*.

Se ha concluido que MongoDB es la mejor opción de las contempladas. Esto no quiere decir que el resto de las BDNR sean malas, o que algunas no serían adecuadas para la realidad planteada; sino que al contrastar con la realidad planteada se concluyó que MongoDB es la mejor alternativa. Esto se debe a su buen nivel de consistencia, durabilidad, fiabilidad y gran desempeño a la hora de realizar operaciones de lectura con una gran cantidad de datos.

El análisis realizado en el presente trabajo ha sido de gran valor para el proyecto de grado de uno de los integrantes,

que fue de donde se tomó la realidad estudiada. Además el análisis realizados a lo largo de la presente investigación, si bien se basan en una realidad concreta, ayuda como guías generales al estudio de pros y contras de distintas bases de datos no relacionales, lo cual se cree que puede ser de utilidad el extrapolarlos a otras realidades en el futuro.

X. Anexo

A continuación se puede observar el modelado de la realidad en MonngoDB:

```
Curso {
   _id: ObjectId,
   descripcion: string,
   personal: boolean,
   modulos: [ ObjectId ],
}

Modulo {
   _id: ObjectId,
   nombre: string,
   descripcion: string,
   configuracionDictados: [ ObjectId ],
}

ConfiguracionDictado {
   // Configuraciones generales
   _id: ObjectId,
   fechaCreacion: Date,
   usuarioCreador: ObjectId,
   nombre: string,
   descripcion: string,
   // Configuraciones dictado melodico
   reglaGiroMelodico:
     [{notas: [String], prioridad: int }],
   tesitura:
     [{ clave: string,
       notaMenor: string,
       notaMayor: string
     }],
   notasInicio: [ string ],
   notasFin: [ string ],
   clavePrioridad:
     [{ clave: string, prioridad: int }],
   reglaEscalaDiatonica: [{
{
nombre: String,
notasAlteradasSostenido: [String],
    notasAlteradasBemol: [String]
},
    prioridad: string}],.
   // Configuraciones dictado ritmico
   reglaCelulaRitmica:
     [{
       figuras: [String],
       simple: boolean,
       prioridad: int
     }],
   cantidadCompases: int,
   reglaCompas:
     [{ numerador: int,
       denominador: int,
       prioridad: int
```

```
    }],
  simple: boolean,
  notaReferencia: string,
}

Dictado {
  _id: ObjectId,
  configuracionDictadoId: ObjectId,
  fechaDeCreacion: Date,
  notas: [ string ]
  celulaRitmica: [[ string ]]
  clave: string,
  escalaDiatonica: [ string ],
  resolucion:
    [{ fecha: Date, nota: string}]

}

Usuario {
  _id: ObjectId,
  nombre: string,
  apellido: string,
  email: string,
  password: string
  esDocente: boolean,
  cursos:
    [{ cursoId: ObjectId,
      fechaInscripcion: date,
      responsable: boolean
    }],
  institutos:
    [{
      institutoId: ObjectId,
      habilitado: boolean
    }],
  dictados: [ ObjectId ],
 }

Instituto {
  _id: ObjectId,
  cursos: [ ObjectId ],
}

GiroMelodico {
  _id: ObjectId,
  notas: [ string ],
}

CelulaRitmica {
  _id: ObjectId,
  figuras: [ String ],
  simple: boolean,
}

EscalaDiatonica {
  _id: ObjectId,
  nombre: String,
  notasAlteradasSostenido:
    [String],
  notasAlteradasBemol:
    [String]
}
```

REFERENCIAS

[Chakraborttii(2015)] Chandranil Chakraborttii. Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool., 2015. URL https://www.researchgate.net/publication/331287438_Performance_Evaluation_of_NoSQL_Systems_Using_Yahoo_Cloud_Serving_Benchmarking_Tool.

[João Ricardo Lourenço(2015)] Paulo Carreiro Marco Vieira1 Jorge Bernardino João Ricardo Lourenço, Bruno Cabral. Choosing the right NoSQL database for the job: a quality attribute evaluation, 2015.