Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bases de Datos II

Investigación de Oracle NoSQL

Integrantes:
Jason Barrantes Arce (2015048456)
Steven Bonilla Zúñiga (2015056296)
Jorge González Rodríguez (2015083567)

Profesor: Erick Hernández Bonilla

02 de mayo, 2017
San José, Costa Rica

Contenido

Objetivo de la investigación

La investigación realizada fue acerca de la base de datos Oracle NoSQL, el objetivo principal de esta investigación consistió en describir las propiedades, características, estructuras, arquitectura y el uso frecuente de la misma; finalmente, realizar un prototipo que haga uso de una base de datos de Oracle NoSQL que se comunique mediante una aplicación.

Para ser más específicos, los objetivos de la investigación son los siguientes. Primero, describir adecuadamente las propiedades que tiene una base de datos Oracle NoSQL, mencionando algunas propiedades que nos permitan intuir en qué tipo de aplicaciones o herramientas de la vida real puedan llegar a utilizarse. Otro objetivo es mencionar y enlistar las características principales y secundarias de una base de este tipo, esto es sumamente importante ya que nos ayuda a conocer qué tipo comportamiento podemos esperar, y que podría llegar a darse en el caso de que queramos usarla o implementarla; además, ¿qué tipo de aplicaciones, basadas en su comportamiento y características, son las que no deberían usar ni implementarse con una base Oracle NoSQL? Cabe mencionar también que es importante detallar las estructuras que tiene y las funciones que realiza cada una, esto nos permitirá identificar cada uno de los miembros de las que está compuesta la base y el tipo de funcionalidad que cada parte posee, lo que claramente nos permite tener un mejor conocimiento del ambiente en el que se va a usar o implementar la aplicación. En adición, es importante conocer la estructura física y lógica de la base de datos ya que son primordiales para entender las bases de Oracle NoSQL. Y finalmente, conocer la sintaxis, semántica y pragmática del lenguaje que usan este tipo de bases, así como también, la forma en cómo se realiza la comunicación con la base de datos y diversos aspectos técnicos que nos permitan comprender mejor la base de datos para poder desarrollar los prototipos y realizar su adecuada implementación.

Antecedentes

Bases de datos NoSQL

Las bases de datos NoSQL fueron diseñadas para reemplazar las fallas o errores que tenían bases de datos relacionales, principalmente el gran empuje que hace que las bases de datos NoSQL se popularicen es por la aparición del Big Data, como muchos saben las aplicaciones deben procesar gran cantidad de datos a gran velocidad, estos datos pueden ser heterogéneos (de diversas estructuras). Esto obliga al mercado a actualizarse pero muchas de las bases de datos relacionales no pueden adaptarse al cambio principalmente porque no están diseñadas para eso.

Esta eventualidad concluye con los cambios que empiezan a aparecen en el hardware como el aumento del nivel de paralelismo, ya que los procesadores empiezan a ser desarrollados con más núcleos, aparece la llamada “commodity computing era” que está definida como la computación realizada por equipos básicos en vez de la hecha por las superminicomputadoras de alto costo o en computadoras de boutique, que implican el uso de un gran número de componentes para la computación en paralelo, y finalmente, la aparición de la nube.

El aumento del paralelismo es un problema, pero en cambio la aparición de la “commodity computing era” y la nube, abren nuevas posibilidades para almacenar y gestionar grandes cantidades de datos. Es decir, abren nuevas posibilidades para tener bases de datos escalables sin gran costo. A esto se le llama “scale-out” o escalado horizontal, que consiste en tener más nodos por un equipo conforme aumenta la cantidad de datos. Este tipo de nueva escala es viable para las bases de datos NoSQL quienes fueron desarrolladas con este propósito, o podría decirse que está en su ADN este tipo de comportamiento a diferencia de las relacionales que no han logrado adaptarse de manera adecuada.

Entre las diversas ventajas que ofrece el escalado horizontal se menciona que, entre más equipos, hay menores posibilidades de que el equipo completo falle aumentando la disponibilidad, el problema es que si se aplica esta política se debe

replicar la información entre nodos. Otro problema es la coordinación de los nodos con el clúster debido a que todos los equipos deben estar estrictamente conectados, lo que complica el software y lo hace concurrente a que haya pérdida de tiempo entre la comunicación de los nodos.

Las bases NoSQL son caracterizadas por cumplir con el acrónimo de BASE:

- **Basically Available (Disponibilidad):** Usa replicación para reducir la probabilidad indisponibilidad, y particiona los datos entre diversos servidores para que el problema no llegue a afectar directamente al sistema.
- **Soft State (Estado Flexible):** Suele ser inconsistente y relega el diseño de las inconsistencias a los desarrolladores de aplicaciones.
- **Consistencia Eventual:** La consistencia es eventual, por lo tanto, los sistemas asumen que los datos ingresados serán consistentes en un punto en el futuro.

Bases de datos llave / valor

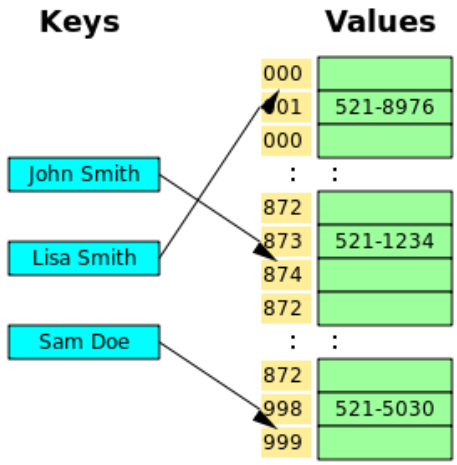
Las bases de datos llave / valor relacionan los datos mediante una llave que puede poseer uno o varios elementos. Los elementos están vinculados a una llave de acceso que permite acceder a los datos, algo muy parecido a lo que sucede con una tabla hash en donde los valores están relacionados a una única llave. Prometen un rendimiento excelente para volúmenes de datos muy grandes a cambio de eliminar ciertas funciones como verificación intrínseca de datos, referencias externas (foreign key) o triggers. Por esta razón, el proceso de analizar los datos queda referente para que lo realice la aplicación y no la base de datos, estas se dedican solamente a almacenarlos.

Son bases formadas por contenedores o cabinets, en donde cada uno de estos puede tener tantas parejas de llave-valor como quiera, dependiendo del

sistema puede haber llaves duplicadas o no. Podemos tener datos de la misma naturaleza como también podemos tener datos de diferente naturaleza.

La llave de acceso puede estar formada por diversa información, en algunos sistemas los valores podrían almacenarse como cadenas, en otros los valores podrían estar tipificados: cadena, entero, flotante, fecha, etc.

Como son bases diseñadas para únicamente almacenar información de compleja estructura, no realizan análisis de los datos, esto puede realizarlo mediante OLAP. Para intentar contrarrestar esto podría utilizar un diseño vertical en vez de un horizontal. El vertical requiere de dos contenedores para el sistema llave-valor, pero puede complicar más el diseño y puede que no sea al final la solución que se buscaba.



Berkeley DB

Berkley DB es un motor de bases diseñado especialmente para almacenar datos en llave-valor. Es una librería de manejo de base de datos con API para C, C++, Java, Perl, Python, Ruby, Tcl y otros lenguajes. La base se compone sólo de los registros cuyo formato está fijado libremente por el programa de llamada. No existe el concepto de mesa y la base de datos no se puede buscar a través de un

lenguaje de manipulación de datos como SQL. Cada registro consta de un par clave / valor, la clave no es única.

Berkeley DB fue desarrollada por la Universidad de Berkeley de California con el objetivo de lograr eliminar el código de AT&T de BSD 4.3 para la versión 4.4. Hasta que en 1996 Netscape solicitó a los autores de Berkeley BD que mejorarán y ampliarán su biblioteca. En la versión 1.86 satisficieron los requisitos de Netscape para un servidor LDAP y utilizarlo en el navegador Netscape, esta petición llevó a la creación de Sleepycat Software, que fue adquirida en febrero de 2006 por Oracle Corporation.

Entre sus principales características podemos mencionar:

- Gestión de transacciones y recuperaciones ante errores ACID.
- Posee una caché configurable para mejorar el rendimiento
- Capacidad de bloquear registros
- Gestión simplificada de backup y replicación. Podemos también realizar copias de seguridad y replicación sin apagar la base, ni detener el servidor
- Soporta gran cantidad de datos
- Los datos pueden estar encriptados
- Los datos se almacenan en el formato nativo del programa
- No tiene modo cliente – servidor
- Operaciones de apoyo Xa

Historia Oracle NoSQL

Las bases de datos NoSQL representan una reciente evolución de las arquitecturas de las bases de datos continuando con el proceso durante los últimos veinte años. En los 90 muchas de las aplicaciones empiezan a crecer verticalmente, por lo que dan origen a las arquitecturas cliente-servidor, y las arquitecturas de cliente-servidor dan paso a arquitecturas de aplicaciones web de tres niveles.

De la mano con este proceso, aumenta la demanda de análisis de datos a nivel web, por lo tanto, se agregan procedimientos de “map-reduce”. Los arquitectos de datos finalmente deciden relajar la consistencia (principalmente la consistencia transaccional) a cambio de un incremento de la escalabilidad y distribución a gran escala.

En esta época Oracle es uno de los grandes de las bases de datos principalmente en las bases de datos relacionales, pero esto no necesariamente significa que se dedican únicamente al diseño de bases relacionales, dieron un gran impacto cuando decidieron arriesgarse por una base como Oracle NoSQL.

Fue diseñada por la compañía Sleepycat Software (compañía formada por los miembros creadores de Berkeley BD) que fue comprada por Oracle en febrero de 2006, se les encargó diseñar una nueva versión de Berkeley BD que tuviera características propias que antiguamente no se habían contemplado, pero el equipo terminó reescribiendo una nueva base y diseñando Oracle NoSQL.

Finalmente se dio a conocer acerca de la base de datos Oracle NoSQL en 2011, aunque posee ciertas coherencias con Berkeley DB es dicho que es una nueva base en su totalidad.

Funcionamiento y Arquitectura

Arquitectura

Grupo de Replicación

Un grupo de replicación consta de un número de nodos de replicación que puede ser configurable. Cada grupo posee el mismo número de nodos, y este es el que dicta el número de fallas que el sistema puede llegar a tolerar, por ejemplo, un sistema con tres nodos de réplica, puede tolerar dos fallos mientras prosigue tomando información de la base.

Cada grupo se le asigna un *master node* que maneja todas las operaciones de modificación de datos como crear, modificar y eliminar. Por lo que los otros nodos son de sólo lectura que replican los datos entre ellos, aunque en caso de falla ellos pueden asimilar la posición del *master node*. Normalmente por defecto siempre se asignan tres nodos de replicación por grupo.

Nodos de almacenamiento

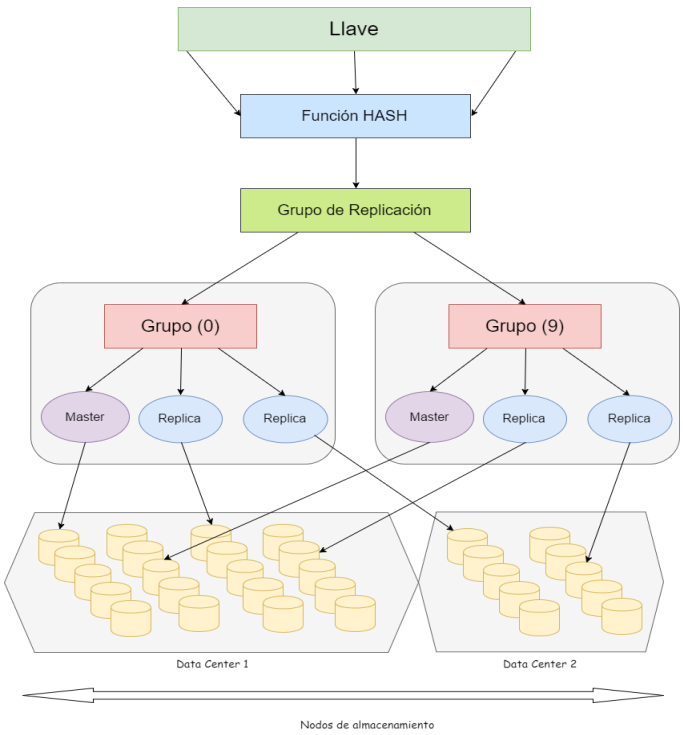
Un nodo de almacenamiento es una unidad física independiente con sus propios componentes (CPU, Memoria, Dirección IP) y su almacenamiento local persistente. Es una computadora o máquina separada de las demás. Entre más nodos tenga un sistema mayor será su capacidad de almacenamiento y los sistemas con mayor grado de replicación en los grupos pueden producir una disminución del rendimiento en unidades con menores grados de replicación.

Un nodo agente de almacenamiento (SNA) procesa la información de cada uno de los nodos de almacenamiento monitoreando el comportamiento del nodo. Es responsable de recopilar los datos operativos del nodo de almacenamiento de manera continua y entregarlos al servicio de administración cuando se lo solicite.

Un nodo de almacenamiento abastece a uno o más nodos de replicación. Los nodos de replicación sólo pueden pertenecer a un grupo de replicación, todos en un mismo grupo de replicación simple poseen los mismos datos.

Los nodos de replicación admiten la API de base de datos Oracle NoSQL a través de llamadas RMI desde el cliente y obtienen datos directamente desde o escriben datos directamente en el sistema de almacenamiento estructurado de registro, lo que proporciona un excelente rendimiento de escritura, manteniendo también estructuras de índice que proporcionan rendimiento de lectura de baja latencia. El motor de almacenamiento Oracle NoSQL Database fue pionero en el uso del almacenamiento estructurado en log en bases de datos clave / valor desde su despliegue inicial en 2003 y ha sido probado en varias soluciones *Open Source NoSQL*, como *Dynamo*, *Voldemort* y *GenieDB.*

Oracle NoSQL usa replicación para asegurarse que los datos estén disponibles en caso de falla. En caso de que el *master node* sea el que falle, se escogerá un nuevo *master node* y se le atribuyen las responsabilidades de su predecesor.



Driver Client (Controlador)

Su función consiste en mapear las llaves hash a los respectivos grupos de replicación, esto lo hace primero por medio de la topología y la tabla del estado de los grupos de replicación (RGST).

Topología

El proceso de mapeo es realizado por la topología quien mapea las llaves a particiones y a grupos de replicación, por cada grupo de replicación se debe incluir el nombre del host de los nodos de almacenamiento que aloja cada nodo de replicación en el grupo, el nombre del servidor se asocia con los nodos de replicación y el centro de los datos en el que cada nodo reside. Se carga cuando se inicializa un cliente o un nodo de replicación, y posteriormente se puede actualizar por el administrador en caso de que haya cambios.

RGST

La tabla del estado de los grupos de replicación es una tabla que almacena los nodos de replicación que hay y su actual estado. Es dinámico, por lo que requiere de mantenimiento continuo, y el encargado de realizarlo es el hilo de ejecución *Replication State Node Update.* Tiene dos propósitos, el primero consiste en identificar el *master node* en un grupo de replicación y balancear la carga en todos los nodos de réplica de un grupo de replicación. Debido a que es una estructura compartida cada cliente y nodo de réplica debe tener una propia copia. Los clientes y los nodos utilizan el *RequestDispatcher* para redireccionar las solicitudes de escrituras al *master node* y las solicitudes de lecturas a los demás nodos.

Los hilos de *Update* y el del *RequestDispatcher* recolectan la información de nodos de replicación remotos incluyendo el estado actual del nodo en su grupo de replicación, una indicación de que tan actualizado está, el tiempo de su última interacción con el nodo, el tiempo de respuesta promedio y la cantidad de

solicitudes que esperan ser procesadas. El hilo de *Update* mantiene las conexiones y restablece las que hayan caído.

Node	Type	Response	Host
Rg1-n1	Master	50	SN-21
Rg1-n2	Replica	100	SN-22
Rg1-n3	Replica	10	SN-3

Modelo de datos que implementa

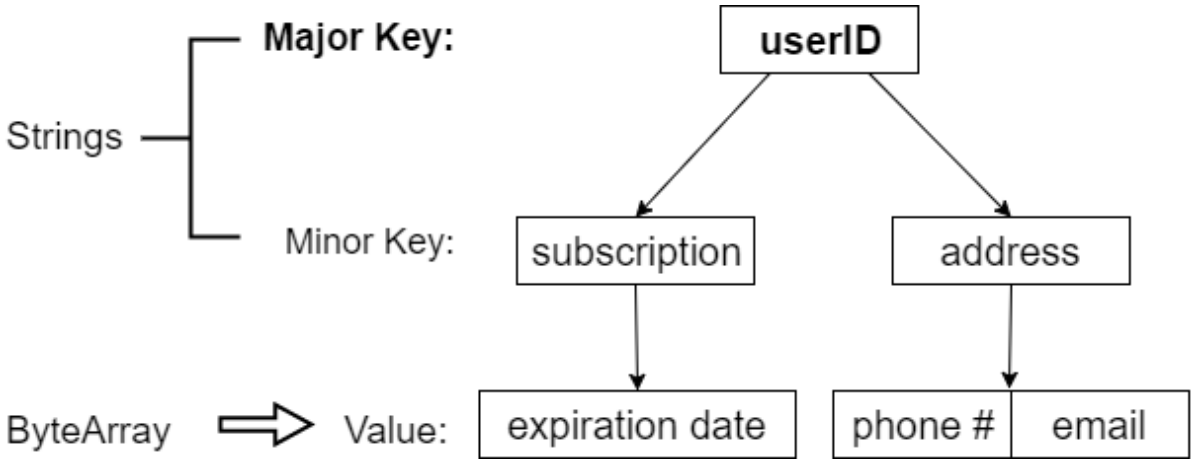
Implementa un mapeo de las llaves (strings) definidas por un usuario a valores opacos. Almacena numerosas versiones de pares llave / valor, pero mantiene la versión más reciente en la base. Estas bases utilizan “single-master replication”, el nodo master tiene el valor más actualizado para una llave mientras las réplicas de solo lectura pueden tener versiones anteriores. Las aplicaciones usan el número de la versión para asegurar la consistencia de las operaciones de borrado, lectura y edición.

Las llaves de hash de Oracle NoSQL proporcionan una distribución equitativa sobre una colección de equipos que proporcionan el espacio para la base de datos. Sin embargo, las aplicaciones pueden aprovechar las capacidades de las subllaves para lograr encontrar la localidad de los datos.

Una llave es la concatenación de una “major key” (ruta llave mayor o primaria) y una “minor key” (ruta llave menor o secundaria), las cuales deben ser especificadas en la aplicación. Todos aquellos que comparten la misma “major key” si se unen, se puede saber en dónde se encuentran los datos almacenados. Y dentro de una colección de “major key”, se encuentra la llave completa, que abarca tanto las “major key” como las “minor key” de los bucles indexados. La comprensión del prefijo hace que el almacenamiento de grupos de llaves sea eficiente.

Los valores son almacenados como un arreglo de bytes, Oracle NoSQL no realiza suposiciones acerca del contenido del arreglo. El proceso de mapeo de los arreglos de bytes a estructuras de datos (mediante la serialización y de serialización) es dejado para que lo realice la aplicación. Aplicaciones con requerimientos de datos muy simples pueden usar los valores que contienen, mediante estructuras de registro fijas. En cambio, otras aplicaciones pueden usar los valores que contienen como estructuras complejas, un conjunto de propiedades de un nombre (nombre-valor-par) u otros tipos de auto descripción.

No existen restricciones acerca del tamaño o la estructura del valor que se asigna al campo.



Características

La base de datos Oracle NoSQL posee una arquitectura "No Single Point of Failure" que consiste en una estructura que en el caso de que una pequeña parte del sistema falle, el sistema continúe funcionando. “NOSP” es una solución viable y la más adecuada cuando se necesita acceder a los datos de manera sencilla pero no eficiente (menor tiempo de respuesta posible), también es funcional en el caso de que las aplicaciones demanden más el uso de la base por lo que podrían exceder la capacidad de volumen o latencia de las soluciones tradicionales de administración de datos.

Algunos ejemplos son, cuando los datos de flujo a la hora de presionar click en sitios web de alto volumen, el procesamiento de acciones o eventos que van a

requerir un gran rendimiento por parte del equipo y las comunicaciones por redes sociales que producen volúmenes extraordinarios de datos llave / valor.

Otros ejemplos de dominios que en cambio requieren lo más novedoso en baja latencia son el seguimiento del comportamiento del comercio al por menor en línea, el acceso a perfiles de clientes, la obtención de anuncios de clientes adecuados, el almacenamiento y el reenvío de comunicaciones en tiempo real.

Algunas aplicaciones pueden hacer uso de una base de Oracle NoSQL debido a que están altamente distribuidas en los sectores, como la agregación de sensores en tiempo real y la autentificación escalable.

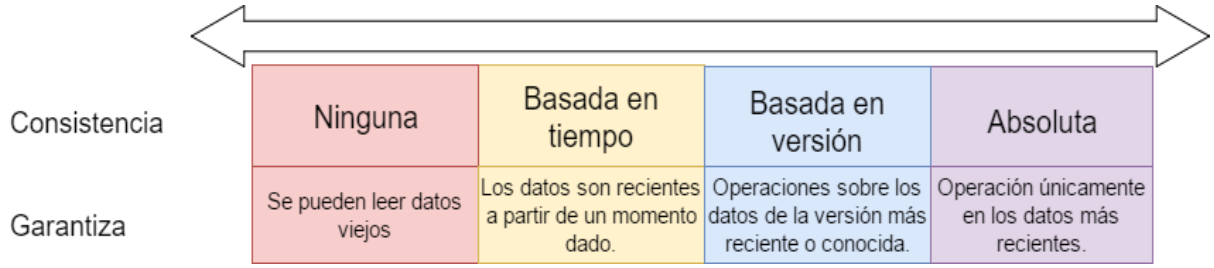
Aprovecha el motor de almacenamiento de alta disponibilidad de Oracle Berkeley DB Java Edition sobre el que añade una capa de servicios para poder usarse en entornos distribuidos.

Proporciona almacenamiento confiable, rápido y distribuido a las aplicaciones que necesitan integrarse con el procesamiento ETL (es un proceso en almacenamiento de datos que consiste en sacar datos de los sistemas fuentes y poner en un repositorio o depósito).

Oracle NoSQL no es fiel con la ideología de ACID, debido a que no soporta transacciones atómicas arbitrarias. Sin embargo, si es capaz de realizar operaciones atómicas en la misma llave e incluso permite las transacciones atómicas en grupos de llaves que comparten la misma llave. En pocas palabras, mientras se garantice que se van a realizar en un mismo nodo, van a ser capaces de hacerlo.

Consistencia

Mientras muchas bases de datos NoSQL no garantizan y no dejan manipular la consistencia de las operaciones, Oracle NoSQL le permite a la aplicación definir qué tipo de consistencia le gustaría utilizar, ya que esta es parte de sus políticas. Puede haber dos extremos, la consistencia absoluta, que garantiza que todas las lecturas deben retornar el valor más actualizado de una llave; y la consistencia débil, que puede tolerar inconsistencia de datos y retornar valores viejos cuando se realice lectura. Entre los dos extremos, se podría definir un tipo de consistencia basada en el tiempo de respuesta o basada en la versión.



Durabilidad

Oracle NoSQL provee un rango de políticas de durabilidad que garantiza que el sistema prosiga después de una caída. La aplicación podría solicitar al sistema que bloquee las peticiones de escritura que están intentando acceder a un respectivo dato hasta que se haya realizado una copia a todos los demás nodos. Esto puede desencadenar complicaciones y fallas, pero garantiza a los demás nodos que se tienen los datos correctos en caso de falla. En otro aspecto, las aplicaciones pueden solicitar a las operaciones de escritura que obtengan el dato y lo retornen una vez se haya guardado en disco. Estas políticas garantizan el mejor rendimiento, pero no garantizan la durabilidad del dato.

Integración con diversas herramientas

Hadoop

Se puede suministrar clases como KVAvroInputFormat y KVInputFormat para leer todos los datos de la base de datos OracleNoSQL directamente en Jobs MapReduce. También permite cargar los registros de Oracle NoSQL en Oracle Loader for Hadoop.

API

Las API me ofrecen una forma para acceder a los datos mediante las operaciones básicas Create, Read, Update y Delete (CRUD), con un conjunto de iteradores los que se empaquetan en un solo archivo jar. Al utilizar API de uno o más procesos de cliente que acceden a un proceso independiente del servidor de base de datos Oracle NoSQL no se obliga a la necesidad de configurar los sistemas para el desarrollo inicial y final de las pruebas. Me da un soporte Bulk sobre las transacciones. Y soporta a Avro para serializar y deserializar.

```
// Put a new key/value pair in the database, if key not already present.
Key key = Key.createKey("Katana");
String valString = "sword";

store.putIfAbsent(key, Value.createValue(valString.getBytes()));

// Read the value back from the database.
ValueVersion retValue = store.get(key);

// Update this item, only if the current version matches the version I read.
// In conjunction with the previous get, this implements a read-modify-write
String newvalString = "Really nice sword";
Value newval = Value.createValue(newvalString.getBytes());

store.putIfVersion(key, newval, retValue.getVersion());

// Finally, (unconditionally) delete this key/value pair from the database.
store.delete(key);
```

Iteradores

Oracle NoSQL permite dos tipos de iteraciones:

- **Iteración no ordenada**: El resultado obtenido no es transaccional porque la iteración se ejecuta en un nivel de aislamiento de solo lectura, lo que significa que el conjunto de resultados contendrá sólo pares clave / valor que se han escrito persistentemente en la base de datos, pero no hay garantías de consistencia semántica entre pares clave / valor.
- **Iteración ordenada**: En este caso se garantiza la consistencia semántica.

Conclusiones

Las bases de datos NoSQL funcionan y cumplen el acrónimo de BASE, pero con Oracle NoSQL existe un problema y es por la consistencia eventual, Oracle NoSQL no soporta adecuadamente la consistencia eventual. Para ser más específicos, se dará el siguiente caso: si el *master node* falla para unos valores de una llave o si es separado de otros nodos de réplica, la llave no va a estar disponible durante un tiempo mientras se espera elegir el nuevo *master node,* entonces durante ese momento, las escrituras pueden continuarse ejecutando en el nuevo *master node*, pero aquellas que todavía no han migrado al nuevo nodo se pierden.

La arquitectura de Oracle NoSQL funciona bajo una filosofía *No Single Point of Failure* que establece que el sistema debe recuperarse ante eventualidades en caso de que alguno de sus nodos falle, entre sus medidas para enfrentar eventualidades se encuentra la replicación multi-master que es utilizada en los nodos para poder asegurarse de que la información este distribuida y respaldada. Oracle NoSQL también ofrece las herramientas adecuadas, establece políticas de consistencia y de durabilidad para que el sistema se mantenga funcional durante su uso.

El modelo de datos que posee Oracle NoSQL permite que se guarden los valores mediante *major key* y *minor key* garantizando un grado de eficiencia alto para acceder a los datos, pero esto también puede llegar a ser confuso e incluso puede ser molesto para los usuarios, principalmente porque las llaves deben tener características específicas que les permitan ser reconocidas de las demás y si no se tiene el nombre adecuado de la llave a la que se desea acceder podría causarse errores.

Oracle NoSQL es una base de datos escalable horizontalmente, esto significa que entre más nodos de almacenamiento se agreguen se tiene más almacenamiento respondiendo de la misma manera. Esto garantiza que la base de datos sea capaz de crecer en función de los nodos que se agreguen. Pero la

desventaja de esto, es que en el caso de pierda uno de los nodos hay pérdida de información valiosa. Además de que entre más nodos se agreguen la información se encontrará más distribuida lo que puede ralentizar los procesos. Pero el mayor problema está en que en caso de que se pierdan nodos la información puede estar dispersa.

Las políticas de durabilidad nos hacen cuestionarnos si la información que se almacena en la base de datos está garantizada de ser permanente (hasta cierto punto), pero se nos da un ligero sentimiento de que la base de datos no será capaz de contrarrestar los problemas a la larga que puedan aparecer.

Lecciones aprendidas

El ambiente de desarrollo web en java deja mucho que desear, los servidores web que poseen no son potentes para soportar cierta cantidad de tráfico de datos y por consiguiente funciona mal o no funcionan.

Al comienzo de la programación de la página web, se presentaron varios inconvenientes a la hora de conectar Oracle NoSql con la web, principalmente por desconocer cómo funciona el ambiente de desarrollo web en java, así como sus limitaciones. Conforme se avanzaba en el proyecto se fue facilitando el manejo de la conexión con la base de forma casi automática.

La base de datos de Oracle NoSql es sumamente rápida a la hora de hacer consultas específicas, el funcionamiento y la arquitectura hacer que funcionalidades como verificar usuarios o administradores se hagan con mucha eficiencia.

A la hora de hacer llenado de datos, así como las consultas de datos notamos que para realiza inserciones la base de datos toma un poco más de tiempo porque cada vez que se desee ingresar datos a la base este dato debe ordenarse y ser asignadas a una función hash que las consultará más velozmente.

El modelo de programación MVC es cada vez más provechoso para el grupo, se ha notado que la facilidad al escribir código usando este modelo es mucho mas eficiente puesto que se pueden formar las capas de datos y que las mismas no dependan ni de la base de datos ni de las otras capas del MVC.

Realizar el algoritmo para poder consultar a las bases de datos es sumamente complicado, se deben almacenar los valores de las llaves para poder consultar y obtener los valores, en muchos casos esto causo problemas con perdida de información en la base de datos del proyecto.

Referencias

Abadi, D. (2011). *Overview of the Oracle NoSQL Database*. [online] Dbmsmusings.blogspot.com. Available at: http://dbmsmusings.blogspot.com/2011/10/overview-of-oracle-nosql-database.html [Accessed 1 May 2017].

Alberto, J. (2017). *Bases de datos clave-valor – soft·in·spain*. [online] Softinspain.com. Available at: http://softinspain.com/desarrollo/bases-de-datos-clave-valor/ [Accessed 30 Apr. 2017].

Ayudamosconocer.com. (2017). *Berkeley DB, historia, descripción, características, las aplicaciones que utilizan BDB*. [online] Available at: http://ayudamosconocer.com/significados/letra-b/berkeley-db.php [Accessed 30 Apr. 2017].

Datawarehouse4u.info. (2017). *ETL*. [online] Available at: http://datawarehouse4u.info/ETL-process.html [Accessed 29 Apr. 2017].

En.wikipedia.org. (2017). *Commodity computing*. [online] Available at: https://en.wikipedia.org/wiki/Commodity_computing [Accessed 30 Apr. 2017].

Es.wikipedia.org. (2017). *Berkeley DB*. [online] Available at: https://es.wikipedia.org/wiki/Berkeley_DB [Accessed 30 Apr. 2017].

Gpd.sip.ucm.es. (2017). *NoSQL-FdI - intro*. [online] Available at: http://gpd.sip.ucm.es/rafa/docencia/nosql/intro.html [Accessed 30 Apr. 2017].

Seltzer, M. (2011). *Oracle NoSQL Databases*. 1st ed. [ebook] Redwood Shores, California. Available at: http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf [Accessed 30 Apr. 2017].

Un poco de Java. (2017). *Un poco de Oracle NoSQL*. [online] Available at: https://unpocodejava.wordpress.com/2013/06/19/un-poco-de-oracle-nosql/ [Accessed 29 Apr. 2017].