Ottone, Juan Manuel Nro: 13615/8
Portal, Maria de los Angeles Nro: 13196/1

Bases de Datos 2 2020 -TP2

Parte 1: Bases de Datos NoSQL y Relacionales

1. ¿Cuáles de los siguientes conceptos de RDBMS existen en MongoDB? En caso de no existir, ¿hay alguna alternativa? ¿Cuál es?
• Base de Datos
• Tabla / Relación
• Fila / Tupla
• Columna

En mongoDB existe el concepto de **base de datos** pero, a diferencia de las bases de datos relacionales, hablamos de base de datos orientada a documentos.
Al tratarse de base de datos no relacionales, éstas no poseen **relaciones**, sin embargo, hay maneras de agregar relaciones en el caso de que sean necesarias pero la idea en este tipo de base de datos es tener las menos posibles.
El término **tabla** no existe, lo que tenemos son conjuntos de **colecciones.**
Tampoco existen los conceptos de **filas** o **tuplas**, más bien hablamos de **documentos**, los cuales son objetos json que dentro de MongoDB se conocen como bson, binary json. Ésto implica que no existe un esquema predefinido.
No hay **columnas**, sino **campos**.
Una colección puede tener un número indeterminado de documentos. Cada documento puede tener su propia estructura la cual es simple y está compuesta por pares clave/valor, la clave es el nombre del campo y el valor es su contenido.

2. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?

MongoDB antes de la versión 4.0 garantizaba transacciones ACID a nivel de documento, esto quiere decir que las operaciones realizadas en los subdocumentos dentro de un documento cumplían con la condición de que si un error ocurría la base de datos se encargaba de hacer rollback al estado anterior al inicio de la operación.
Con la versión 4.0, se agregan transacciones ACID (Atomicity, Consistency, Isolation, Durability) entre múltiples documentos.
Los métodos para el manejo de transacciones en mongoDB se explican a continuación.

Por un lado tenemos métodos para gestionar la sesión:

- `startSession()`, las transacciones están asociadas a una sesión, es por ello que siempre necesitaremos abrir una sesión.
- `endSession()`, cuando cerremos la sesión, si hay alguna transacción ejecutándose, esta se abortará.

Y por otro los métodos para controlar la transacción:

- `Session.startTransaction()`, este método nos sirve para empezar la transacción multi-documento dentro de una sesión.
- `Session.commitTransaction()`, con este método confirmaremos los cambios que hayamos realizado dentro de la transacción.
- `Session.abortTransaction()`, aborta la ejecución de la transacción y por lo tanto no graba las modificaciones hechas a lo largo de la transacción.

3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?

Los índices son: estructuras de datos especiales que almacenan una pequeña porción de los datos de la colección de tal forma que sea fácil de recorrer. El índice almacena el valor de un campo específico o de un conjunto de campos, ordenados por el valor de dichos campos. El orden de cada una de las entradas del índice permite realizar consultas de igualdad o rango de manera eficiente.

Tipos de índice: MongoDB proporciona varios tipos de índice diferentes para admitir tipos específicos de datos y consultas.

Campo único: Además del `_id` índice definido por MongoDB, se admite la creación de índices ascendentes/descendentes definidos por el usuario en un solo campo de un documento. Para un índice de campo único y operaciones de clasificación, el orden de clasificación (es decir, ascendente o descendente) de la clave no importa, porque MongoDB puede atravesar el índice en cualquier dirección.

Índice compuesto: MongoDB también admite índices definidos por el usuario en múltiples campos. El orden de los campos que figuran en un índice compuesto tiene importancia. The index sorts first by `userid` and then, within each `userid` value, sorts by `score`.`{ userid: 1, score: -1 }useriduseridscore` Para índices compuestos y operaciones de clasificación, el orden de clasificación (es decir, ascendente o descendente) de las claves de índice puede determinar si el índice puede admitir una operación de clasificación.

Índice Multikey: MongoDB utiliza índices de múltiples claves para indexar el contenido almacenado en matrices. Si indexa un campo que contiene un valor de matriz, MongoDB crea entradas de índice separadas para *cada* elemento de la matriz. Estos índices multiciclo permiten que las consultas seleccionen documentos que contienen matrices haciendo coincidir elementos o elementos de las matrices. MongoDB determina automáticamente si

se crea un índice multiciclo si el campo indexado contiene un valor de matriz; no es necesario que especifique explícitamente el tipo.

Índice geoespacial: Para admitir consultas eficientes de datos de coordenadas geoespaciales, MongoDB proporciona dos índices especiales: índices 2d que utilizan geometría plana al devolver resultados e índices 2dsphere que utilizan geometría esférica para devolver resultados.

Índices de texto: MongoDB proporciona un `text` de índice que admite la búsqueda de contenido de un String en una colección. Estos índices no almacenan palabras de *detección* específicas del idioma (por ejemplo, "the", "a", "or").

Índices hash: Para admitir sharding basada en hash, MongoDB proporciona un Hashed Index, que indexa el hash del valor de un campo. Estos índices tienen una distribución de valores más aleatoria a lo largo de su rango, pero *solo* admiten coincidencias de igualdad y no pueden admitir consultas basadas en rangos.

4. ¿Existen claves foráneas en MongoDB?

En mongoDB no existen las claves foráneas, sin embargo, se pueden simular guardando en un campo de un documento el id de otro documento como referencia. De esta manera, haciendo una segunda consulta podríamos llegar al documento referenciado.
Otro método para aplicar referencias entre documentos es mediante DBRef, que es una convención para representar un documento. Incluyen el nombre de la colección y, en algunos casos, el nombre de la base de datos, además del valor del campo _id.
Es importante aclarar que necesitamos mantener la integridad de los datos por nosotros mismos.
Para muchos casos de uso en MongoDB, el modelo de datos desnormalizados donde los datos relacionados se almacenan en un solo documento será óptimo. No obstante, en algunos casos, tiene sentido almacenar información relacionada en documentos separados, generalmente en diferentes colecciones o bases de datos.