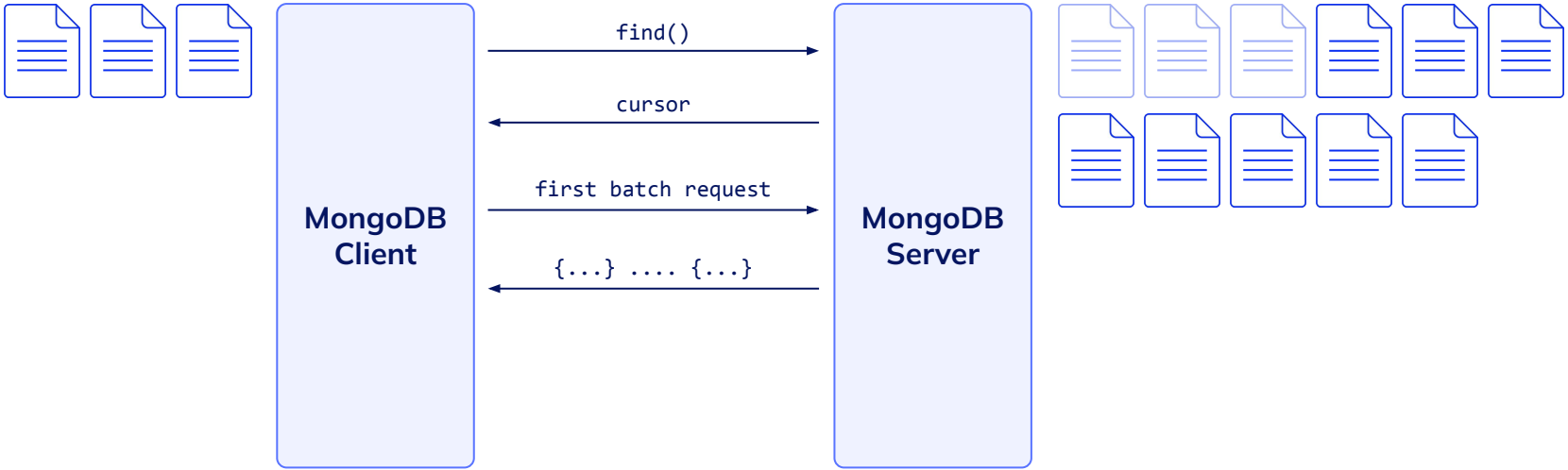
MongoDB Fundamentos

Módulo 3

Operaciones Bulk en MongoDB

Cursor in MongoDB



Como la mayoría de los sistemas de bases de datos, MongoDB proporciona llamadas ***API*** que permiten insertar o recuperar múltiples documentos en una sola operación.

Estas interfaces "**`Array`**" o "**`Bulk`**" mejoran notablemente el rendimiento de la base de datos al reducir drásticamente la cantidad de viajes de ida y vuelta entre el cliente y las bases de datos. Para comprobar lo fundamental que es esta optimización, se puede imaginar que hay un grupo de personas que es necesario llevar al otro lado del río. Se cuenta con un barco que puede llevar
a 100 personas a la vez, pero por alguna razón, solo lleva a una persona en cada viaje, no es inteligente, ¿verdad?

No aprovechar las inserciones de matriz es similar: esencialmente está enviando paquetes de red que podrían tomar cientos de documentos con solo un documento en cada paquete.

Optimización de lecturas masivas usando .batchSize()

Al recuperar datos utilizando un cursor, se puede especificar el número de filas recuperadas en cada operación con la cláusula **`batchSize`**.
Por ejemplo, debajo tenemos un cursor donde **`limit`** controla el número total de filas que procesaremos, mientras que **`arraySize`** controla el número de documentos recuperados de la base de datos **mongoDB** en cada solicitud de red.

```
cursor = useDb.collection('millions').find().limit(limit).batchSize(arraySize);

for (let doc = await cursor.next(); doc != null; doc = await cursor.next()) {
    counter++;
}
```

Tengamos en cuenta que el operador **batchSize** en realidad no devuelve una matriz al programa, sólo controla la cantidad de documentos recuperados en cada viaje de ida y vuelta de la red. Todo esto sucede "bajo el capó" desde el punto de vista de sus programas.

De forma predeterminada, MongoDB establece un valor bastante alto para **.batchSize** de 101 documentos, y es posible que su rendimiento se deteriore fácilmente si jugamos sin criterio con él. Sin embargo, si se obtienen muchas filas pequeñas de una tabla remota, se puede obtener una mejora significativa en el rendimiento al aumentar la configuración.

A continuación se verá el efecto de manipular **batchSize()** con distintos valores, al utilizar otro método del cursor llamado **objsLeftInBatch()** que devuelve el número de documentos restantes en el lote actual. Veamos la próxima pantalla.

Se crea este script y se ejecuta en la consola de Mongo Shell.

```
print('--------------- batches ------------------')

var cursor = db.personas.find()
print('Hay', cursor.count(), ' documentos por leer')

while(cursor.hasNext()) {
  var doc = cursor.next()
  print(doc.nombre,'----> quedan ',cursor.objsLeftInBatch(),' documentos por leer')
}
```

Luego de la ejecución de script, se obtendrá en consola:

```
> load('scripts/batch.js')
--------------- batches ------------------

Hay 19  documentos por leer

Sigmund ----> quedan  18  documentos por leer
Lulu ----> quedan  17  documentos por leer
Katarina ----> quedan  16  documentos por leer
Nedra ----> quedan  15  documentos por leer
Shaina ----> quedan  14  documentos por leer
Retha ----> quedan  13  documentos por leer
Salvador ----> quedan  12  documentos por leer
Ilene ----> quedan  11  documentos por leer
Michelle ----> quedan  10  documentos por leer
Veronica ----> quedan  9  documentos por leer
Christophe ----> quedan  8  documentos por leer
Sarah ----> quedan  7  documentos por leer
Greg ----> quedan  6  documentos por leer
Piper ----> quedan  5  documentos por leer
Mavis ----> quedan  4  documentos por leer
Gunner ----> quedan  3  documentos por leer
Maxie ----> quedan  2  documentos por leer
Alexie ----> quedan  1  documentos por leer
Arthur ----> quedan  0  documentos por leer
true
>
```

Se puede apreciar que si no se establece el valor de **batchSize**, el **batch** o lote contiene todos los documentos leídos (19) porque es el valor por defecto de este parámetro de 101 documentos. Significa que toda la información viaja en el paquete de red de la consulta y no se necesitan hacer peticiones de red adicionales.

A través del método **objsLeftInBatch** es posible ver los documentos restantes que quedan en el **batch**. Al leer secuencialmente el cursor, va bajando el número de documentos que quedan en el **batch**. Ahora se modifica el script y se aplica un **batchSize** de 8:

```
print('--------------- batches ------------------')
var cursor = db.personas.find()
print('Hay', cursor.count(), ' documentos por leer')

cursor.batchSize(8)

while(cursor.hasNext()) {
  var doc = cursor.next()
 print(doc.nombre,'----> quedan ',cursor.objsLeftInBatch(),' documentos por leer')
}
```

Al ejecutar, se obtiene en la consola:

```
> load('scripts/batch.js')
--------------- batches ------------------

Hay 19  documentos por leer

Sigmund ----> quedan  7  documentos por leer
Lulu ----> quedan  6  documentos por leer
Katarina ----> quedan  5  documentos por leer
Nedra ----> quedan  4  documentos por leer
Shaina ----> quedan  3  documentos por leer
Retha ----> quedan  2  documentos por leer
Salvador ----> quedan  1  documentos por leer
Ilene ----> quedan  0  documentos por leer
Michelle ----> quedan  7  documentos por leer
Veronica ----> quedan  6  documentos por leer
Christophe ----> quedan  5  documentos por leer
Sarah ----> quedan  4  documentos por leer
Greg ----> quedan  3  documentos por leer
Piper ----> quedan  2  documentos por leer
Mavis ----> quedan  1  documentos por leer
Gunner ----> quedan  0  documentos por leer
Maxie ----> quedan  2  documentos por leer
Alexie ----> quedan  1  documentos por leer
Arthur ----> quedan  0  documentos por leer
true
>
```

Se puede apreciar que, luego de la consulta, hay 8 documentos traídos al cliente que, al consumirse, generan otra petición de red para traer los 8 siguientes para, de esa manera, recargar el **`batch`** y seguir mostrando la información. Luego, como son 19 documentos y ya se tienen agotados los dos batches de 8 documentos, restan 3 documentos que se cargan en la última operación de red.

Es importante remarcar que la consulta inicial de 19 documentos quedó almacenada en un cursor del lado del servidor MongoDB, y que el cliente trae del cursor esos documentos en lotes o batches según el valor dispuesto en el **`batchSize`** (con valor por defecto de 101 documentos).

El cliente sólo tiene que hacer la consulta inicial, luego las operaciones necesarias de red para traer la información según el tamaño del lote o **`batch`** son totalmente automáticas y transparentes al cliente.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!