MongoDB

Ing. Luis Guillermo Molero Suárez

Qué es MongoDB

En el mundo de las bases de datos, los sistemas de bases de datos más comunes y populares son RDBMS (sistemas de gestión de bases de datos relacionales). Ahora, si queremos desarrollar una aplicación que maneje un gran volumen de datos, entonces debemos elegir una de esas bases de datos que siempre proporcione soluciones de almacenamiento de datos de alto rendimiento. Para que podamos lograr el rendimiento en la solución en términos de almacenamiento de datos y recuperación de datos con precisión, velocidad y confiabilidad. Ahora, si categorizamos las soluciones de bases de datos, entonces hay principalmente dos tipos de categorías de bases de datos disponibles, es decir, RDBMS o bases de datos relacionales como SQL Server, Oracle, etc., y otro tipo es la base de datos NoSQL como MongoDB, CosmosDB, etc.

La base de datos NoSQL está siendo en realidad una alternativa de la base de datos SQL convencional y, además, este tipo de base de datos proporciona principalmente todos los tipos de características que normalmente están disponibles en los sistemas RDBMS. Hoy en día, las bases de datos NoSQL se vuelven mucho más populares en comparación con el pasado debido al diseño simple, la provisión para escalado horizontal y vertical y también para un control fácil y simple sobre los datos almacenados. Este tipo de base de datos

básicamente rompe la tradición normal de estructura de almacenamiento de datos de la base de datos relacional. Proporciona a los desarrolladores la posibilidad de almacenar datos en la base de datos según los requisitos reales de su programa. Este tipo de facilidad no podemos lograr mediante el uso de la base de datos RDBMS tradicional.

Tipos de NoSQL

Las bases de datos NoSQL se pueden clasificar en cuatro tipos diferentes como se indica a continuación:

De estos cuatro tipos, la base de datos de documentos es la base de datos más popular y ampliamente utilizada en el mundo actual. Este tipo de bases de datos siempre se diseñan sobre la base de un enfoque orientado a documentos para almacenar datos. Este tipo de bases de datos siempre desempeña un papel importante para agregar los datos de los documentos y también representa los datos de una forma muy fácil de buscar u organizada. MongoDB es la base de datos más utilizada en la industria del desarrollo como base de datos de documentos. En las bases de datos de documentos, se ha cambiado el concepto básico de tabla y fila en comparación con la base de datos SQL. Aquí la fila ha sido reemplazada por el término documento, que es una estructura de datos mucho más flexible y basada en modelos. Podemos almacenar datos jerárquicos en un solo documento en la base de datos de documentos. En realidad, la base de datos de documentos siempre admite el modelo de datos semiestructurados. En base de datos de documentos. La tabla ha sido

reemplazada por el término Colección, que es un contenedor de múltiples documentos con la misma estructura o diferentes estructurados.

¿Qué es MongoDB?

MongoDB es una de las bases de datos NoSQL de código abierto más populares escritas en C ++. En febrero de 2015, MongoDB es el cuarto sistema de administración de bases de datos más popular. Fue desarrollado por una empresa 10gen que ahora se conoce como MongoDB Inc.

MongoDB es una base de datos orientada a documentos que almacena datos en documentos similares a JSON con esquema dinámico. Significa que puede almacenar sus registros sin preocuparse por la estructura de datos, como la cantidad de campos o los tipos de campos para almacenar valores. Los documentos MongoDB son similares a los objetos JSON.

¿Quién usa MongoDB?

En la industria de TI actual, hay una gran cantidad de empresas que utilizan MongoDB como un servicio de base de datos para las aplicaciones o los sistemas de almacenamiento de datos. Según la encuesta realizada por Siftery en MongoDB, hay alrededor de 4000+ empresas confirmadas que están utilizando MongoDB como base de datos. Algunos de los nombres clave son:

Castlight Health, IBM, Citrix, Twitter, T-Mobile, Zendesk, Sony, BrightRoll,

Cuadrangular, HTC, InVision, Intercomunicador, etc.

¿Cómo almacena MongoDB los datos?

Como sabe, RDMS almacena datos en formato de tablas y utiliza un lenguaje de consulta estructurado (SQL) para consultar la base de datos. RDBMS también tiene un esquema de base de datos predefinido basado en los requisitos y un conjunto de reglas para definir las relaciones entre los campos en las tablas.

Pero MongoDB almacena datos en documentos a pesar de las tablas. Puede cambiar la estructura de los registros (que se llama como documentos en MongoDB) simplemente agregando nuevos campos o eliminando los existentes. Esta capacidad de MongoDB le ayuda a representar relaciones jerárquicas, almacenar matrices y otras estructuras más complejas fácilmente. MongoDB proporciona alto rendimiento, alta disponibilidad, fácil escalabilidad y replicación y fragmentación automática listas para usar.

¿Por qué y dónde debería usar Mongo DB?

Dado que MongoDB es una base de datos NoSQL, debemos comprender cuándo y por qué debemos usar este tipo de base de datos en las aplicaciones de la vida real. Ya que en circunstancias normales, MongoDB siempre es el

preferido por los desarrolladores o jefes de proyecto cuando nuestra principal preocupación es el trato con gran volumen de datos con un alto rendimiento. Si queremos insertar miles de registros en un segundo, MongoDB es la mejor opción para eso. Además, el escalado horizontal (agregar nuevas columnas) no es un proceso tan fácil en ningún sistema RDBMS. Pero en el caso de MongoDB, es mucho más fácil ya que es una base de datos sin esquema. Además, este tipo de trabajo puede ser manejado directamente por la aplicación de forma automática. No es necesario ningún tipo de trabajo administrativo para realizar ningún tipo de escalado horizontal en MongoDB. MongoDB es bueno para los siguientes tipos de situaciones:

• Tipo de comercio electrónico de aplicaciones basadas en productos

• Sistemas de gestión de blogs y contenidos

• Registro de alta velocidad, almacenamiento en caché, etc. en tiempo real

• Necesidad de mantener datos geoespaciales en función de la ubicación

• Para mantiene datos relacionados con los tipos de redes sociales y redes

• Si la aplicación es un mecanismo débilmente acoplado, significa que el diseño puede cambiar en cualquier momento.

Ventajas de MongoDB

Dado que, MongoDB no es solo una base de datos que solo puede realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) con los datos. Excepto estos, MongoDB también contiene tantas características importantes

debido a que MongoDB se convierte en la base de datos más popular en la categoría NoSQL. Algunas de las características importantes son:

1. MongoDB es una base de datos de tipo de documento sin esquema.

2. Campo de soporte de MongoDB, consulta basada en rango, expresión regular o regex, etc. para buscar los datos de los datos almacenados.

3. MongoDB es muy fácil de escalar hacia arriba o hacia abajo.

4. MongoDB básicamente usa memoria interna para almacenar los conjuntos de datos temporales de trabajo para los que es mucho más rápido.

5. MongoDB admite índices primarios y secundarios en cualquier campo.

6. MongoDB admite la replicación de la base de datos.

7. Podemos realizar el balanceo de carga en MongoDB usando Sharding. Escala la base de datos horizontalmente utilizando Sharding.

8. MongoDB se puede utilizar como un sistema de almacenamiento de archivos que se conoce como GridFS.

9. MongoDB proporciona las diferentes formas de realizar operaciones de agregación en los datos, como la canalización de agregación, la reducción de mapas o los comandos de agregación de un solo objetivo.

10. MongoDB puede almacenar cualquier tipo de archivo que puede ser de cualquier tamaño sin afectar nuestra pila

11. MongoDB básicamente usa objetos JavaScript en lugar de procedimiento.

12. MongoDB admite un tipo de recopilación especial como TTL (Time-To-Live) para el almacenamiento de datos que caducan en un momento determinado

13. El esquema de base de datos dinámico utilizado en MongoDB se llama BSON

Soporte de plataforma e idioma

Al igual que otros sistemas RDBMS, MongoDB también proporciona soporte oficial para una gran cantidad de lenguajes de programación y marcos. Los controladores Mongo están disponibles para los siguientes idiomas y marcos populares: C, C ++, C # y .NET, Java, Node.js, Perl, PHP, Bibliotecas PHP, Frameworks y Herramientas, Python, Rubí, Mongoide (Ruby ODM)

Comparación entre el esquema SqlDB y el esquema MongoDB

Dado que, si no estamos muy familiarizados con los sistemas de base de datos MongoDB y conocemos principalmente los sistemas RDBMS, las tablas a continuación representan las traducciones simples de terminología de SQL DB al esquema MongoDB: servidor SQL, MongoDB, Base de datos, Base de datos, Mesa, Colección, Fila, Documento, Índice, Índice, Columna, Campo, Unión, Vinculación e incrustación, Dividir, Fragmentación, Replicación, ReplSet, Limitaciones de MongoDB.

Dado que, en la sección anterior, discutimos principalmente sobre las ventajas de MongoDB. Pero a pesar de estos beneficios, MongoDB también tiene algunas limitaciones como:

• Dado que MongoDB no es tan fuerte ACID (Atómico, Consistencia, Aislamiento y Durabilidad) como en comparación con la mayoría de los sistemas RDBMS.

• No puede manejar transacciones complejas

• En MongoDB, no hay provisión para procedimientos almacenados o funciones o activadores, por lo que no hay posibilidades de implementar ninguna lógica comercial en el nivel de la base de datos, lo que se puede hacer en cualquier sistema RBMS.

Para más información, visite las siguientes páginas:

https://docs.mongodb.com/manual/

Mongodb Basics

https://www.mongodb.com/basics

En la práctica

Para poder experimentar con los ejemplos de código, necesitará acceso a una base de datos MongoDB.

Puede descargar una base de datos gratuita de MongoDB en:

https://www.mongodb.com

O comience de inmediato con un servicio en la nube de MongoDB en:

https://www.mongodb.com/cloud/atlas

Instalar el controlador MongoDB

Intentemos acceder a una base de datos MongoDB con Node.js.

Para descargar e instalar el controlador oficial de MongoDB, abra el CMD y ejecute lo siguiente:

Descargue e instale el paquete mongodb:

```
C:\Users\luisg>npm install mongodb
```

Ahora ha descargado e instalado un controlador de base de datos mongodb.

Node.js puede usar este módulo para manipular bases de datos MongoDB:

```
var mongo = require ('mongodb');
```

MongoDB Crear base de datos

Para crear una base de datos en MongoDB, comience creando un objeto MongoClient, luego especifique una URL de conexión con la dirección IP correcta y el nombre de la base de datos que desea crear.

MongoDB creará la base de datos si no existe y se conectará a ella.

Ejemplo

Cree una base de datos llamada "mydb":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

Guarde el código anterior en un archivo llamado "demo_create_mongo_db.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_create_mongo_db.js
```

Lo que te dará este resultado:

```
Database created!
```

Importante: En MongoDB, no se crea una base de datos hasta que obtiene contenido.

MongoDB espera hasta que haya creado una colección (tabla), con al menos un documento (registro) antes de que realmente cree la base de datos (y la colección).

MongoDB Create Collection

Una colección en MongoDB es lo mismo que una tabla en MySQL

Crear una colección

Para crear una colección en MongoDB, use el método createCollection ():

Ejemplo

Crea una colección llamada "clientes":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.createCollection("customers", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_createcollection.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_createcollection.js
```

Lo que te dará este resultado:

```
Colección creada!
```

Importante: en MongoDB, una colección no se crea hasta que obtiene contenido.

MongoDB espera hasta que haya insertado un documento antes de que realmente cree la colección.

Inserto de MongoDB

Insertar en la colección

Para insertar un registro, o documento como se llama en MongoDB, en una colección, usamos el método insertOne ().

Un documento en MongoDB es lo mismo que un registro en MySQL

El primer parámetro del método insertOne () es un objeto que contiene los nombres y valores de cada campo en el documento que desea insertar.

También necesita una función de devolución de llamada donde puede trabajar

con cualquier error o el resultado de la inserción:

Ejemplo

Inserte un documento en la colección "clientes":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = { name: "Company Inc", address: "Highway 37" };
  dbo.collection("customers").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_insert.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_insert.js
```

Lo que te dará este resultado:

```
1 document inserted
```

Nota: Si intenta insertar documentos en una colección que no existe, MongoDB creará la colección automáticamente.

MongoDB Find

En MongoDB usamos los métodos find y findOne para buscar datos en una colección.

Al igual que la instrucción SELECT se usa para buscar datos en una tabla en una base de datos MySQL.

Encuentra uno

Para seleccionar datos de una colección en MongoDB, podemos usar el método findOne ().

El método findOne () devuelve la primera aparición en la selección.

El primer parámetro del método findOne () es un objeto de consulta. En este ejemplo usamos un objeto de consulta vacío, que selecciona todos los documentos de una colección (pero devuelve solo el primer documento).

Ejemplo

Encuentre el primer documento en la colección de clientes:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").findOne({}, function(err, result) {
    if (err) throw err;
    console.log(result.name);
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_findone.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_findone.js
```

Lo que te dará este resultado:

```
Company Inc.
```

Filtrar el resultado

Al buscar documentos en una colección, puede filtrar el resultado mediante un objeto de consulta.

El primer argumento del método find () es un objeto de consulta y se utiliza para limitar la búsqueda.

Ejemplo

Busque documentos con la dirección "Park Lane 38":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: "Park Lane 38" };
  dbo.collection("customers").find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_query.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_query.js
```

Lo que te dará este resultado:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8e , name: 'Ben', address: 'Park Lane 38'
}
]
```

Filtrar con expresiones regulares

Puede escribir expresiones regulares para encontrar exactamente lo que está buscando.

Las expresiones regulares solo se pueden usar para consultar cadenas.

Para buscar solo los documentos donde el campo "dirección" comienza con la letra "S", use la expresión regular / ^ S /:

Ejemplo

Busque documentos donde la dirección comience con la letra "S":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: /^S/ };
  dbo.collection("customers").find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_query_s.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_query_s.js
```

Lo que te dará este resultado:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8b , name: 'Richard', address: 'Sky st
331' },
  { _id: 58fdbf5c0ef8a50b4cdd9a91 , name: 'Viola', address: 'Sideway
1633' }
]
```

MongoDB Sort

Ordenar el resultado

Utilice el método sort () para ordenar el resultado en orden ascendente o descendente. El método sort () toma un parámetro, un objeto que define el orden de clasificación.

Ejemplo

Ordene el resultado alfabéticamente por nombre:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var mysort = { name: 1 };
  dbo.collection("customers").find().sort(mysort).toArray(function(err, r
esult) {
    if (err) throw err;
    console.log(result);
```

```
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_sort.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_sort.js
```

Lo que te dará este resultado:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a86, name: 'Amy', address: 'Apple st 652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e, name: 'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a, name: 'Betty', address: 'Green Grass
1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90, name: 'Chuck', address: 'Main Road
989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87, name: 'Hannah', address: 'Mountain
21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a84, name: 'John', address: 'Highway 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88, name: 'Michael', address: 'Valley
345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85, name: 'Peter', address: 'Lowstreet
4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b, name: 'Richard', address: 'Sky st
331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89, name: 'Sandy', address: 'Ocean blvd
2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c, name: 'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d, name: 'Vicky', address: 'Yellow Garden
2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91, name: 'Viola', address: 'Sideway
1633'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f, name: 'William', address: 'Central st
954'}
]
```

Orden descendiente

Utilice el valor -1 en el objeto de clasificación para ordenar de forma descendente.

```
{ name: 1 } // ascending
{ name: -1 } // descending
```

Ejemplo

Ordene el resultado en orden inverso alfabéticamente por nombre:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var mysort = { name: -1 };
  dbo.collection("customers").find().sort(mysort).toArray(function(err, r
esult) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_sort_desc.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_sort_desc.js
```

Lo que te dará este resultado:

```
[
  { _id: 58fdbf5c0ef8a50b4cdd9a8f, name: 'William', address: 'Central st
954'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91, name: 'Viola', address: 'Sideway
1633'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d, name: 'Vicky', address: 'Yellow Garden
2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c, name: 'Susan', address: 'One way 98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89, name: 'Sandy', address: 'Ocean blvd
2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b, name: 'Richard', address: 'Sky st
331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85, name: 'Peter', address: 'Lowstreet
4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88, name: 'Michael', address: 'Valley
345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a84, name: 'John', address: 'Highway 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87, name: 'Hannah', address: 'Mountain
21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90, name: 'Chuck', address: 'Main Road
989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a, name: 'Betty', address: 'Green Grass
1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e, name: 'Ben', address: 'Park Lane 38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86, name: 'Amy', address: 'Apple st 652'}
]
```

Eliminar documento

Para eliminar un registro o documento como se llama en MongoDB, usamos el método deleteOne ().

El primer parámetro del método deleteOne () es un objeto de consulta que define qué documento eliminar.

Nota: Si la consulta encuentra más de un documento, solo se elimina la primera aparición.

Ejemplo

Borre el documento con la dirección "Montaña 21":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: 'Mountain 21' };
  dbo.collection("customers").deleteOne(myquery, function(err, obj) {
    if (err) throw err;
    console.log("1 document deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_delete.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_delete.js
```

Lo que te dará este resultado:

```
1 document deleted
```

Eliminar muchos

Para eliminar más de un documento, use el método deleteMany ().

El primer parámetro del método deleteMany () es un objeto de consulta que define qué documentos eliminar.

Ejemplo

Elimine todos los documentos donde la dirección comience con la letra "O":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: /^O/ };
  dbo.collection("customers").deleteMany(myquery, function(err, obj) {
    if (err) throw err;
    console.log(obj.result.n + " document(s) deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_delete_many.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_delete_many.js
```

Lo que te dará este resultado:

```
2 document(s) deleted
```

El objeto de resultado

El método deleteMany () devuelve un objeto que contiene información sobre cómo la ejecución afectó a la base de datos.

La mayor parte de la información no es importante de entender, pero un objeto dentro del objeto se llama "resultado", que nos dice si la ejecución fue correcta y cuántos documentos se vieron afectados.

El objeto de resultado se ve así:

```
{ n: 2, ok: 1 }
```

Puede utilizar este objeto para devolver el número de documentos eliminados:

Ejemplo

Devuelve el número de documentos eliminados:

```
console.log(obj.result.n);
```

Lo que producirá este resultado:

```
2
```

MongoDB Drop

Colección Drop

Puede eliminar una tabla o colección como se llama en MongoDB, utilizando el método drop ().

El método drop () toma una función de devolución de llamada que contiene el objeto de error y el parámetro de resultado que devuelve verdadero si la colección se eliminó correctamente; de lo contrario, devuelve falso.

Ejemplo

Elimina la tabla "clientes":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").drop(function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_drop.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_drop.js
```

Lo que te dará este resultado:

```
Collection deleted
```

db.dropCollection

También puede usar el método dropCollection () para eliminar una tabla (colección).

El método dropCollection () toma dos parámetros: el nombre de la colección y una función de devolución de llamada.

Ejemplo

Elimina la colección "clientes", usando dropCollection ():

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.dropCollection("customers", function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_dropcollection.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_dropcollection.js
```

Lo que te dará este resultado:

```
Collection deleted
```

MongoDB Drop

Colección Drop

Puede eliminar una tabla o colección como se llama en MongoDB, utilizando el método drop ().

El método drop () toma una función de devolución de llamada que contiene el objeto de error y el parámetro de resultado que devuelve verdadero si la colección se eliminó correctamente; de lo contrario, devuelve falso.

Ejemplo

Elimina la tabla "clientes":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").drop(function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_drop.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_drop.js
```

Lo que te dará este resultado:

```
Collection deleted
```

db.dropCollection

También puede usar el método dropCollection () para eliminar una tabla (colección).

El método dropCollection () toma dos parámetros: el nombre de la colección y una función de devolución de llamada.

Ejemplo

Elimina la colección "clientes", usando dropCollection ():

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.dropCollection("customers", function(err, delOK) {
    if (err) throw err;
    if (delOK) console.log("Collection deleted");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_dropcollection.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_dropcollection.js
```

Lo que te dará este resultado:

```
Collection deleted
```

Actualización de Node.js MongoDB

Actualizar documento

Puede actualizar un registro o documento como se llama en MongoDB, utilizando el método updateOne ().

El primer parámetro del método updateOne () es un objeto de consulta que define qué documento actualizar.

Nota: Si la consulta encuentra más de un registro, solo se actualiza la primera aparición.

El segundo parámetro es un objeto que define los nuevos valores del documento.

Ejemplo

Actualice el documento con la dirección "Valley 345" a name = "Mickey" y address = "Canyon 123":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: "Valley 345" };
  var newvalues = { $set: {name: "Mickey", address: "Canyon 123" } };
  dbo.collection("customers").updateOne(myquery, newvalues, function(err,
 res) {
    if (err) throw err;
    console.log("1 document updated");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_update_one.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_update_one.js
```

Lo que te dará este resultado:

```
1 document updated
```

Actualizar solo campos específicos

Cuando se usa el operador $ set, solo se actualizan los campos especificados:

Ejemplo

Actualice la dirección de "Valley 345" a "Canyon 123":

```
...
  var myquery = { address: "Valley 345" };
  var newvalues = { $set: { address: "Canyon 123" } };
  dbo.collection("customers").updateOne(myquery, newvalues, function(err,
 res) {
...
```

Actualizar muchos documentos

Para actualizar todos los documentos que cumplen con los criterios de la consulta, use el método updateMany ().

Ejemplo

Actualice todos los documentos donde el nombre comience con la letra "S":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://127.0.0.1:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { address: /^S/ };
  var newvalues = {$set: {name: "Minnie"} };
  dbo.collection("customers").updateMany(myquery, newvalues, function(err
, res) {
    if (err) throw err;
    console.log(res.result.nModified + " document(s) updated");
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_update_many.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_update_many.js
```

Lo que te dará este resultado:

```
2 document(s) updated
```

El objeto de resultado

Los métodos updateOne () y updateMany () devuelven un objeto que contiene información sobre cómo la ejecución afectó a la base de datos.

La mayor parte de la información no es importante de entender, pero un objeto dentro del objeto se llama "resultado", que nos dice si la ejecución fue correcta y cuántos documentos se vieron afectados.

El objeto de resultado se ve así:

```
{ n: 1, nModified: 2, ok: 1 }
```

Puede utilizar este objeto para devolver el número de documentos actualizados:

Ejemplo

Devuelve el número de documentos actualizados:

```
console.log(res.result.nModified);
```

Lo que producirá este resultado:

```
2
```

Límite de MongoDB

Limite el resultado

Para limitar el resultado en MongoDB, usamos el método limit ().

El método limit () toma un parámetro, un número que define cuántos documentos devolver.

Considere que tiene una colección de "clientes":

```
customers
[
  { _id: 58fdbf5c0ef8a50b4cdd9a84 , name: 'John', address: 'Highway 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85 , name: 'Peter', address: 'Lowstreet
4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86 , name: 'Amy', address: 'Apple st
652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87 , name: 'Hannah', address: 'Mountain
21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88 , name: 'Michael', address: 'Valley
345'},
  { _id: 58fdbf5c0ef8a50b4cdd9a89 , name: 'Sandy', address: 'Ocean blvd
2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8a , name: 'Betty', address: 'Green Grass
1'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8b , name: 'Richard', address: 'Sky st
331'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8c , name: 'Susan', address: 'One way
98'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8d , name: 'Vicky', address: 'Yellow
Garden 2'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8e , name: 'Ben', address: 'Park Lane
38'},
  { _id: 58fdbf5c0ef8a50b4cdd9a8f , name: 'William', address: 'Central st
954'},
  { _id: 58fdbf5c0ef8a50b4cdd9a90 , name: 'Chuck', address: 'Main Road
989'},
  { _id: 58fdbf5c0ef8a50b4cdd9a91 , name: 'Viola', address: 'Sideway
1633'}
]
```

Ejemplo

Limite el resultado para devolver solo 5 documentos:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.collection("customers").find().limit(5).toArray(function(err, resul
t) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Guarde el código anterior en un archivo llamado "demo_mongodb_limit.js" y ejecute el archivo:

```
C:\Users\luisg>node demo_mongodb_limit.js
```

Lo que te dará este resultado:

```
customers
[
  { _id: 58fdbf5c0ef8a50b4cdd9a84 , name: 'John', address: 'Highway 71'},
  { _id: 58fdbf5c0ef8a50b4cdd9a85 , name: 'Peter', address: 'Lowstreet
4'},
  { _id: 58fdbf5c0ef8a50b4cdd9a86 , name: 'Amy', address: 'Apple st
652'},
  { _id: 58fdbf5c0ef8a50b4cdd9a87 , name: 'Hannah', address: 'Mountain
21'},
  { _id: 58fdbf5c0ef8a50b4cdd9a88 , name: 'Michael', address: 'Valley
345'}
]
```

Como puede ver en el resultado anterior, solo se devolvieron los 5 primeros documentos.

MongoDB Join

Unirse a las colecciones

MongoDB no es una base de datos relacional, pero puede realizar una combinación externa izquierda utilizando la etapa $ lookup.

La etapa $ lookup le permite especificar a qué colección desea unirse a la colección actual y qué campos deben coincidir.

Considere que tiene una colección de "pedidos" y una colección de "productos":