UNIVERSIDAD TÉCNOLOGICA NACIONAL
FACULTAD REGIONAL GENERAL PACHECO

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

Base de Datos II

Conceptos Básicos NoSQL

ING. ARIEL HERRERA

Contenido

1: Introducción a las Bases de Datos NoSQL

1.1 Qué son las Bases de Datos NoSQL

- **Definición:**
  Las bases de datos NoSQL son sistemas de gestión de datos diseñados para almacenar, gestionar y acceder a grandes volúmenes de datos distribuidos, sin depender de estructuras relacionales tradicionales (tablas y esquemas rígidos).
    - NoSQL significa "Not Only SQL," lo que implica flexibilidad para manejar datos estructurados y no estructurados.

1.2 Diferencias con Bases de Datos Relacionales (SQL)

Característica	SQL	NoSQL
Estructura de Datos	Tablas relacionales	Clave-valor, documentos, grafos, columnas.
Flexibilidad del esquema	Estricto	Esquemas flexibles.
Escalabilidad	Vertical (más recursos a un servidor)	Horizontal (más servidores).
Lenguaje de Consulta	SQL estándar	Varia según el motor.

1.3 Características principales

1. **Escalabilidad Horizontal:** Permite agregar servidores para manejar más datos y consultas.
2. **Alta Disponibilidad:** Los datos están replicados para mejorar la tolerancia a fallos.
3. **Velocidad:** Optimizado para lecturas/escrituras rápidas en entornos de grandes datos.

2: Introducción Teorica a MongoDB

El modelo de datos documental está fuertemente orientado a agregados, dado que una base de datos consiste en un conjunto de agregados denominados «documentos». Las bases de datos documentales se caracterizan por que definen un conjunto de estructuras y tipos permitidos que pueden ser almacenados, y además es posible acceder a la estructura del agregado, teniendo como ventaja que se consigue más flexibilidad en el acceso.
En este sentido se pueden realizar consultas a la base de datos según los

campos del agregado, se pueden recuperar partes del agregado en vez del agregado completo, y además se pueden crear índices basados en el contenido del agregado. Dado que cada agregado tiene asociado un identificador, es posible realizar búsquedas del estilo clave-valor.
MongoDB es una base de datos NoSQL orientada a documentos creada por la compañía 10gen en el año 2007. Se caracteriza por que almacena los datos en documentos de tipo JSON con un esquema dinámico denominado «BSON».

Documentos

Los documentos son la unidad básica de organización de la información en MongoDB, y desempeñan un papel equivalente a una fila en las bases de datos relacionales. Un documento es un conjunto ordenado de claves que tienen asociados valores, y que se corresponden con algunas estructuras de datos típicas de los lenguajes de programación tales como tablas hash o diccionarios.
En general, los documentos contendrán múltiples pares clavevalor, como, por ejemplo, {"Nombre":"Juan","País":"España"}.
Sus principales características son:

• Las claves son cadenas, por lo que se permite cualquier carácter con un par de excepciones:
– La clave no pueden contener el carácter nulo «\0».
– El punto «.» y el «$» deben evitarse, pues tienen propiedades especiales.

• MongoDB es sensitivo tanto a las mayúsculas/minúsculas como a los tipos de datos. Así, por ejemplo, los siguientes documentos se consideran distintos: {"Edad":3}, {"Edad":"3"}, {"edad":3}, {"edad":"3"}.

• Los documentos no pueden tener claves duplicadas. Así, por ejemplo, el siguiente documento es incorrecto:
{"edad":3,"edad":56}.

• Los pares clave-valor están ordenados en los documentos. Por ejemplo, el documento {"x":3,"y":5} no es igual que {"y":5,"x":3}. Es importante no definir las aplicaciones pensando en el orden de los campos, pues MongoDB puede reordenarlos automáticamente en determinadas situaciones.

• Los valores de un documento pueden ser de diferentes tipos.

Tipos de datos

Los principales tipos de datos soportados por los documentos en MongoDB son:
• Nulo: representa el valor nulo o bien un campo que no existe. Por ejemplo, {"x":null}.

• Booleanos: representa el tipo booleano, que puede tomar los valores de true o false. Por ejemplo, {"x":true}.

• Números: distingue entre números reales, como, por ejemplo, {"x":3.14}, y números enteros, como, por ejemplo, {"x":45}.

• Cadenas: cualquier cadena de caracteres, como, por ejemplo, {"x":"Ejemplo"}.

• Fechas: almacena la fecha en milisegundos, pero no la zona horario. Por ejemplo, {"x":new Date()}.

• Expresiones regulares: se pueden usar expresiones regulares para realizar consultas.

• Arrays: se representa como un conjunto o lista de valores. Por ejemplo, {"x":["a","b","c"]}.

• Documentos embebidos: los documentos pueden contener documentos embebidos como valores de un documento padre. Por ejemplo, {"x":{"y":45}}.

• Identificadores de objetos: es un identificador de 12 bytes para un documento. Por ejemplo, {"x": OjectId()}.

• Datos binarios: es una cadena de bytes arbitraria que no puede ser manipulada directamente desde el Shell y que sirve para representar cadenas de caracteres no UTF8.

• Código Javascript: los documentos y las consultas pueden contener código JavaScript. Por ejemplo, {"x: function () {…}}.

3: Introducción Práctica con MongoDB

3.1 Instalación de MongoDB

- Descargar desde la página oficial de MongoDB.
- Instalar siguiendo el asistente para tu sistema operativo.
- Iniciar MongoDB desde la terminal:

3.2 Operaciones CRUD

Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) son las acciones básicas que se pueden realizar sobre los documentos de una colección en MongoDB. A continuación se muestran ejemplos de cómo se ejecutan estas operaciones:

Crear

Esta operación se utiliza para crear una base de datos o cambiar a una existente.

```
use MiBaseDeDatos
```

Nota: Cuando utilice el comando use y la base de datos no exista, aparentemente la creará, pero la creación resultará efectiva cuando cree explícitamente una colección.

Estas operaciones insertan nuevos documentos en una colección existente. Se puede insertar un solo documento o varios a la vez.

```
db.usuarios.insertOne({nombre: "Ana", edad: 28})

db.usuarios.insertMany([{nombre: "Luis"}, {nombre: "María"}])
```

Leer

Permite consultar documentos dentro de una colección. Se pueden usar filtros simples o complejos.

```
db.usuarios.find()

db.usuarios.find({edad: {$gt: 25}})
```

Actualizar

Modifica uno o más documentos existentes. Se puede actualizar parcialmente o reemplazar campos específicos.

```
db.usuarios.updateOne({nombre: "Ana"}, {$set: {edad: 29}})

db.usuarios.updateMany({}, {$set: {activo: true}})
```

Eliminar

Elimina uno o más documentos de una colección según los criterios especificados.

```
db.usuarios.deleteOne({nombre: "Luis"})

db.usuarios.deleteMany({activo: false})
```

3.3 Operadores de Búsqueda

Los operadores de búsqueda permiten construir consultas avanzadas sobre los documentos de una colección. En lugar de buscar solo coincidencias exactas, puedes aplicar condiciones complejas como comparaciones, combinaciones lógicas, verificaciones de tipo o existencia de campos, y patrones de texto. A continuación se detallan los principales tipos de operadores:

Comparación

Permiten comparar valores de campos:
- $eq: Igual a.
- $ne: Distinto de.
- $gt: Mayor que.
- $gte: Mayor o igual que.
- $lt: Menor que.
- $lte: Menor o igual que.

Lógicos

Permiten combinar múltiples condiciones:
- $and: Todas las condiciones deben cumplirse.
- $or: Al menos una condición debe cumplirse.
- $not: Niega una condición.
- $nor: Ninguna de las condiciones debe cumplirse.

Elementos

Permiten consultar la existencia o el tipo de un campo:
- $exists: Verifica si un campo existe.
- $type: Verifica el tipo de dato de un campo.

Evaluación

Permiten evaluar expresiones o patrones personalizados:
- $regex: Coincidencia con expresiones regulares.
- $expr: Evalúa expresiones que involucran múltiples campos.
- $mod: Coincidencia con un resto de división (útil para valores numéricos).

Ejemplo:

```
db.usuarios.find({$or: [{edad: {$lt: 30}}, {nombre: /María/i}]})
```

Nota: Cuando MongoDB haces una consulta find donde usas barras (/) y la bandera i (minúscula) en el valor del campo nombre, estás realizando una búsqueda utilizando una expresión regular (regex) que es insensible a mayúsculas y minúsculas.
Es como realizar en SqlServer una consulta con like = '%María%'

3.4 Comandos de Colecciones

En esta sección se describen comandos utilizados para gestionar colecciones en MongoDB. Las colecciones son conjuntos de documentos y cumplen el rol que tendrían las tablas en bases de datos relacionales.

- db.createCollection(nombre): Crea una nueva colección vacía con el nombre especificado.

```
db.createCollection("productos")
```

- db.nombre.drop(): Elimina la colección especificada y todos sus documentos.

```
db.productos.drop()
```

- db.nombre.renameCollection(nuevoNombre): Cambia el nombre de una colección existente.

```
db.usuarios.renameCollection("clientes")
```

3.5 Agregaciones Avanzadas

El framework de agregación de MongoDB permite procesar datos de multiples documentos y transformarlos según diferentes etapas. Estas etapas se agrupan en un pipeline que puede filtrar, agrupar, ordenar y proyectar datos de forma muy flexible y eficiente.

```
db.ventas.aggregate([
  {$match: {anio: 2024}},
  {$group: {_id: "$mes", total: {$sum: "$monto"}}},
  {$sort: {total: -1}}
])
```

Operadores comunes:

$match: Filtra documentos que cumplen una condición.

$group: Agrupa documentos por un campo y aplica funciones agregadas como $sum.

$sort: Ordena los documentos.

$project: Define qué campos mostrar en la salida.

$unwind: Descompone arreglos en documentos individuales.

$limit: Limita la cantidad de documentos devueltos.

$skip: Omite una cantidad de documentos antes de mostrar los resultados.

```
db.ventas.aggregate([
  {$match: {anio: 2024}},
  {$group: {_id: "$mes", total: {$sum: "$monto"}}},
  {$sort: {total: -1}}
])
```

3.6 Joins con $lookup

MongoDB permite realizar operaciones de tipo “join” utilizando el operador $lookup, el cual une documentos de dos colecciones distintas en una sola salida. Esto es útil para combinar información relacionada distribuida en múltiples colecciones.

```
db.pedidos.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "cliente_id",
      foreignField: "_id",
      as: "datos_cliente"
    }
  }
])
```

Une documentos de colecciones relacionadas.

```
db.pedidos.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "cliente_id",
      foreignField: "_id",
      as: "datos_cliente"
    }
  }
])
```

Une documentos de colecciones relacionadas.

3.7 Create View

En MongoDB, una vista es una consulta predefinida que se comporta como una colección, pero no almacena datos físicamente. Se utiliza para simplificar consultas complejas, ocultar detalles de la estructura de datos, o aplicar restricciones de acceso. Se define utilizando el comando db.createView().

```
db.createView(
  "vista_empleados_activos",  //Normbre de la vista
  "empleados",                // Nombre de la colección
  [
    { $match: { estado: "activo" } },      //Filtro
    {
      $project: {                          //Campos
        _id: 0,
        nombre: 1,
        departamento: 1
      },
    },
  ]
)
```