ISTEA

Bases de No Datos Relacionales

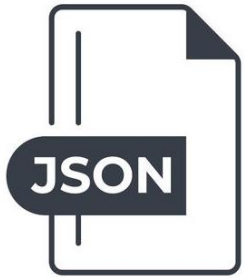
Revisión de la actividad planteada como repaso

ISTEA

Bases de No Datos Relacionales

- Bases de datos documentales
- Ventajas y desventajas
- MongoDB

- Bases de datos orientadas a documentos
  - Una base de datos de documentos almacena y consulta datos como documentos tipo YAML, XML, JSON o BSON.
  - El poder de este tipo de almacenamiento es la independencia que existe entre sí de cada unidad de información, respecto de su estructura.



Unidad autónoma de información

- Si bien poseen una estructura, puede diferir de la de otros documentos (datos semiestructurados)
- Garantiza la flexibilidad
- Organiza la información en forma de árbol jerárquico

- Bases de datos orientadas a documentos

Ventajas

- Aseguran una escritura rápida, dando prioridad a la disponibilidad de la escritura sobre la consistencia de los datos.
- La mayoría de bases de datos documentales cuentan con potentes motores de búsqueda y avanzadas propiedades de indexación
- Simplifican las tareas de adición o actualización de datos en las apps.
- Pueden albergar numerosos tipos de datos.
- Los documentos no cuentan con un esquema predefinido

- Bases de datos orientadas a documentos

Desventajas

- No utilizan el lenguaje SQL. Es decir no existe un lenguaje estandarizado para la creación de estas bases de datos.
- No siempre pueden garantizar las propiedades ACID de atomicidad, consistencia, integridad y durabilidad.
- No tienen una gran comunidad detrás y existe mucha menos información acerca de estas bases de datos.
- Los índices pueden ocupar mucha memoria RAM, sobre todo en las bases documentales que manejan un gran volumen de datos.

- Bases de datos orientadas a documentos

Veamos algunos ejemplos de documentos

Ejemplos de documentos JSON

```
{
“nro”:123,
”dato”:”prueba”
}
```

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
    },
    {
      "name": "Madame Uppercut",
      "age": “39”,
      "secretIdentity": "Jane Wilson",
      "powers": [ "Million  tonne punch", "Damage resistance"]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown"
    }
  ]
}
```

Ejemplos de documentos XML

```
<?xml versión=”1.0” standalone=”no”?>

<!DOCTYPE serie SYSTEM “series.dtd”>
<series>
        <series tipo=”comedia” sistema=”PG-14” ejemplares=”5” año=“1987”>
        <titulo>The Nany</titulo>
        <productor>Fran Drescher</productor>
        <director> Peter Marc Jacobson</director>
        <actores>
                <actor>Fran Drescher</actor>
                <actor>Charles Shaughnessy</actor>
        </actores>
</series>
```

Ejemplo de documento BSON

Un objeto BSON consiste en una lista ordenada de elementos. Cada elemento consta de un campo nombre, un tipo y un valor. Los nombres son de tipo String y los tipos pueden ser: byte,int32, int64, uint64, double, decimal128, date, objectId, array

Para el siguiente **JSON**

{"hello": "world"}

Su equivalente **BSON** seria:

x16\x00\x00\x00	// tamaño total del documento
\x02	// tipo de dato
hello\x00	// nombre del atributo
\x06\x00\x00\x00**world**\x00	// valor del atributo
\x00	// representa el final del documento

El mejor expositor

- MongoDB es una base de datos NoSQL orientada a documentos que apareció a mediados de la década de 2000.
- Su nombre surge de la palabra en inglés “humongous” (que significa enorme).
- MongoDB guarda estructuras de datos en **documentos tipo JSON** (JavaScript Object Notation) con un esquema dinámico, pero internamente almacena los datos en formato **BSON** (Binary JavaScript Object Notation)

El mejor expositor

- MongoDB es una base de datos NoSQL orientada a documentos que apareció a mediados de la década de 2000.
- Su nombre surge de la palabra en inglés “humongous” (que significa enorme).
- MongoDB guarda estructuras de datos en **documentos tipo JSON** (JavaScript Object Notation) con un esquema dinámico, pero internamente almacena los datos en formato **BSON** (Binary JavaScript Object Notation)

Es la Base de Datos NoSQL líder en bases de datos de propósitos generales, documentales y de código abierto

Comparación entre JSON y BSON

	JSON	BSON
Tipo	Los archivos JSON se escriben en formato de texto.	Los archivos BSON se escriben en binario.
Velocidad	JSON es rápido de leer, pero más lento de compilar.	BSON es lento de leer, pero más rápido de construir y escanear.
Espacio	Los datos JSON son ligeramente más pequeños en tamaño de bytes.	Los datos BSON son ligeramente más grandes en tamaño de bytes.
Codificar y decodificar	Podemos enviar JSON a través de APIs sin necesidad de codificar y decodificar.	Los archivos BSON se codifican antes de almacenarlos y se decodifican antes de mostrarlos.

Comparación entre JSON y BSON

	JSON	BSON
Analizar	JSON es un formato legible por humanos que no requiere análisis.	Los archivos BSON se codifican antes de almacenarlos y se decodifican antes de mostrarlos.
Tipos de datos	JSON tiene un conjunto específico de tipos de datos: cadena, booleano, número para tipos de datos numéricos, matriz, objeto y nulo.	A diferencia de JSON, BSON ofrece tipos de datos adicionales, como bindata para datos binarios, decimal128 para numéricos.
Uso	Se utiliza para enviar datos a través de la red (principalmente a través de APIs)	Las bases de datos utilizan BSON para almacenar datos.

Analogía de términos entre el modelo relacional y MongoDB

Modelo relacional	Modelo MongoDB
Instancia de base de datos	Instancia de base de datos
Base de datos o Esquema	Base de datos
Tabla	Colección
Fila	Documento
Identificador de fila o RowID	_id

¿Cómo se modelan las relaciones utilizando documentos?

¿Cómo se modelan las relaciones utilizando documentos?

Básicamente existen dos mecanismos:

- **Referenciar**
- **Embeber**

¿Cómo se modelan las relaciones utilizando documentos?

Embeber: Dentro de una colección registramos un documento que contenga subcolecciones (colecciones).

Referenciar: Se realiza una relación entre varias colecciones agregando los identificadores únicos de una colección en los documentos de la otra.

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

Esto dependerá básicamente

de la forma en que se necesite acceder los datos.

Ventajas de embeber documentos:

- Simplifica las consultas
- Mejora el rendimiento

Consideraciones:

- Tamaño del conjunto de trabajo
- Tamaño del documento

Tip

Considerar la duplicación de datos

Pregunta corta / respuesta rápida

Pregunta corta / respuesta rápida

El tip mencionado sobre embeber documentos…

¿A qué concepto se opone si hablamos de bases de datos relacionales?

Ventajas de **referenciar** documentos:

- Consistencia
- Escalabilidad

Consideraciones:

- Consultas más complejas
- Costo de las uniones

Conclusión…

Los documentos embebidos son una forma eficiente y limpia de almacenar datos relacionados.

En general, al diseñar esquemas para bases documentales, se prefiere embeber de forma predeterminada y usar referencias y combinaciones del lado de la aplicación o de la base de datos, solo cuando valgan la pena.

Cuanto más a menudo una carga de trabajo determinada pueda recuperar un solo documento y tener todos los datos que necesita, más alto será el rendimiento de la aplicación.

Tips…

1. **Lo que se accede junto, se guarda junto.**
2. Si se accede regularmente a un subconjunto de elementos relacionados, entonces embeber ese subconjunto.
3. Si accede regularmente a algunos campos de un documento al que se hace referencia, embeber esos campos.
4. Si una lista puede crecer de forma ilimitada, entonces debe ir en su propia colección

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

- Relaciones **Uno a Uno** con **documentos embebidos**

Supongamos las siguientes colecciones:

Personas
{ _id: “u0001",
nombre: “Juan Martín Hernandez" }

Direcciones
{persona_id: “u0001",
calle: “Malabia 2277",
ciudad: “CABA",
provincia: “CABA",
codPostal: "1425" }

Personas

{ _id: “u0001",
nombre: “Juan Martín Hernandez“,
dirección:
{
calle: “Malabia 2277",
ciudad: “CABA",
provincia: “CABA",
codPostal: "1425"
}
}

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

- Relaciones **Uno a Muchos** con **documentos embebidos**

Personas
{ _id: “u0001",
nombre: “Juan Martín Hernandez" }

Direcciones
{persona_id: “u0001",
calle: “Malabia 2277",
ciudad: “CABA",
provincia: “CABA",
codPostal: "1425" },

{persona_id: “u0001",
calle: “Salta 236",
ciudad: “Mar del Plata",
provincia: “Buenos Aires",
codPostal: "214" }

Personas

{ _id: “u0001",
nombre: “Juan Martín Hernandez“,
dirección:[
{
calle: “Malabia 2277",
ciudad: “CABA",
provincia: “CABA",
codPostal: "1425"
},
{
calle: “Salta 236",
ciudad: “Mar del Plata",
provincia: “Buenos Aires",
codPostal: "214"
}]
}

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

Cuando usamos referencias, el crecimiento de los datos de cada lado de las relaciones, determinará dónde conviene almacenar la referencia, es decir en qué colección.

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

- Relaciones **Uno a Muchos** con **documentos referenciados**

Colección libros
{titulo: "MongoDB: The Definitive Guide",
autor:["K. Chodorow", "M. Dirolf"],
fechaPublicacion: ISODate("2010-09-24"),
paginas: 216,
editor: { nombre: "O'Reilly Media", USAState: "CA" }
}
{titulo: "50 Tips and Tricks for MongoDB…",
autor: "K. Chodorow",
fechaPublicacion: ISODate("2011-05-06"),
paginas: 68,
editor: { nombre: "O'Reilly Media", USAState: "CA" }
}

Colección Editores
{ nombre: "O'Reilly Media",
anioFundacion: 1980,
USAState: "CA“,
libros: [**987654321**,**1234567890**] }

Colección Libros
{_id: **987654321**
titulo: "MongoDB: The Definitive Guide",
autor:["K. Chodorow", "M. Dirolf"],
fechaPublicacion: ISODate("2010-09-24"),
paginas: 216, }
{_id: **1234567890**
titulo: "50 Tips and Tricks for MongoDB…",
autor: "K. Chodorow",
fechaPublicacion: ISODate("2011-05-06"),
paginas: 68}

Cuándo referenciar y cuándo embeber según el tipo de relación que se quiera representar

- Relaciones **Uno a Muchos** con **documentos referenciados**

Colección libros
{titulo: "MongoDB: The Definitive Guide",
autor:["K. Chodorow", "M. Dirolf"],
fechaPublicacion: ISODate("2010-09-24"),
paginas: 216,
editor: { nombre: "O'Reilly Media", USAState: "CA" }
}
{titulo: "50 Tips and Tricks for MongoDB…",
autor: "K. Chodorow",
fechaPublicacion: ISODate("2011-05-06"),
paginas: 68,
editor: { nombre: "O'Reilly Media", USAState: "CA" }
}

Colección Editores
{ **_id: "oreilly"**,
nombre: "O'Reilly Media",
USAState: "CA“ }

Colección Libros
{_id: 987654321
titulo: "MongoDB: The Definitive Guide",
autor:["K. Chodorow", "M. Dirolf"],
fechaPublicacion: ISODate("2010-09-24"),
paginas: 216,
idEditor: "oreilly"}
{_id: 1234567890,
titulo: "50 Tips and Tricks for MongoDB…",
autor: "K. Chodorow",
fechaPublicacion: ISODate("2011-05-06"),
paginas: 68,
idEditor: "oreilly"}

Pregunta corta / respuesta rápida

Pregunta corta / respuesta rápida

¿Cuál es el tamaño máximo de un documento en MongoDB?

Tipos de datos en MongoDB

Tipo	Descripción
String	cualquier cadena de caracteres codificada en utf-8. Debe ir entre comilla doble
Int32, Int 64	enteros de hasta 32 bits ("2090845882") y 64 bits ("2090845886852") No debe ir entre comillas dobles.
Boolean	valor booleano (true o false). Sin comillas.
Timestamp	un dato de 64bits que representa una fecha con hora.
Date	un dato de 32bits que representa una fecha.
Array	agrupación de elementos a manera de lista indizada a partir de la posición 0.
Null	se refiere a un valor nulo o vacío.
Undefined	indica un dato que no ha sido definido.
ObjectID	es un tipo de datos BSON de 12 bytes.
Object	un documento embebido

Identificando tipos de datos en MongoDB

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
                {
                  "name": "Molecule Man",
                  "age": 29,
                  "secretIdentity": "Dan Jukes",
                  "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
                },
                {
                  "name": "Madame Uppercut",
                  "age": “39”,
                  "secretIdentity": "Jane Wilson",
                  "powers": [ "Million  tonne punch", "Damage resistance"]
                },
                {
                  "name": "Eternal Flame",
                  "age": 1000000,
                  "secretIdentity": "Unknown"
                }
                                ]
}
```