Alumno: **Luis Alberto Jaramillo Pulido**

Práctica 5 semana 13: Actualizaciones en una BD en MongoDB

APARTADO 6.- Trabajando con la colección *aficiones* en MongoDB: Actualizaciones

a) Repasa ejemplos de update e insert: ejemplos-Libro-MongoDB-v3.pdf
b)
b) Arreglar la colección *aficiones*. Usando varias update sucesivas puedes añadir y quitar atributos sin perder sus valores. Arregla los errores más destacados que son:
b.1) MotoGP: cambiar el atributo NombreEquipo por Nombre (sin perder sus valores)
b.2) Precio: En Futbol y Baloncesto hay valores exagerados: quitar ceros hasta dejar cifras de tres dígitos.
b.3) Añadir el precio a MotoGP y Ajedrez con valor fijo de 100 (en un solo *update*)
b.4) En MotoGP: Atributo incorrecto: hay "Puntuacón" donde debería haber "Puntuacion" sin perder su valor
b.5) **(sacar nota)** En Ajedrez : unir los valores de Nombre y Apellidos en el atributo Nombre. Quitar el atributo Apellidos.

b.1) MotoGP: cambiar el atributo NombreEquipo por Nombre (sin perder sus valores)

Solución

Actualizo el nombre del campo y utilizo el operador $rename para cambiar el nombre del atributo, “NombreEquipo” por “Nombre” y “multi:true” para actualizar todos los documentos que coinciddentes.

Comando

```
db.aficiones.update( { Tema: "MotoGP" } , { $rename: { NombreEquipo: "Nombre" } } , { multi: true } )
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
  "_id" : ObjectId("5746cf74e6046b47d4073621"),
  "Tema" : "MotoGP",
  "Apodo" : "MGP",
  "NumeroPilotos" : 2.0,
  "CarrerasGanadas" : 8.0,
  "PrimerPiloto" : "Stefan Bradl",
  "SegundoPiloto" : "Alvaro Bautista",
  "Descuento" : NaN,
  "Nombre" : "Aprilia Racing Team Gresini"
}

{
  "_id" : ObjectId("5746cf74e6046b47d4073622"),
  "Tema" : "MotoGP",
  "Apodo" : "MGP",
  "NumeroPilotos" : 2.0,
  "CarrerasGanadas" : 6.0,
  "PrimerPiloto" : "Eugene Laverty",
  "SegundoPiloto" : "Yonny Hernandez",
  "Descuento" : NaN,
  "Nombre" : "Aspar Team MotoGP"
}

{
  "_id" : ObjectId("5746cf74e6046b47d4073623"),
```

```
   "Tema" : "MotoGP",
   "Apodo" : "MGP",
   "NumeroPilotos" : 2.0,
   "CarrerasGanadas" : 3.0,
   "PrimerPiloto" : "Hector Barbera",
   "SegundoPiloto" : "Loris Baz",
   "Descuento" : NaN,
   "Nombre" : "Avintia Racing"
}
```


aficiones 0.002 sec.

	_id	Tema	Apodo	Nombre
14	ObjectId("57...	MotoGP	MGP	Aspar Team MotoGP
15	ObjectId("57...	MotoGP	MGP	Avintia Racing
16	ObjectId("57...	MotoGP	MGP	Ducati Team
17	ObjectId("57...	MotoGP	MGP	Estrella Galicia 0,0 Marc ...
18	ObjectId("57...	MotoGP	MGP	LCR Honda
19	ObjectId("57...	MotoGP	MGP	Monster Yamaha Tech 3
20	ObjectId("57...	MotoGP	MGP	Movistar Yamaha MotoGP
21	ObjectId("57...	MotoGP	MGP	Repsol Honda Team
22	ObjectId("57...	MotoGP	MGP	OCTO Pramac Yakhnich
23	ObjectId("57...	MotoGP	MGP	Team SUZUKI ECSTAR

b.2) Precio: En Futbol y Baloncesto hay valores exagerados: quitar ceros hasta dejar cifras de tres dígitos.

Solución

Recorro toda la colección, y si el tema del documento es futbol o baloncesto, procedo a dividir entre 10 hasta que el precio sea menor a 1000, cuando es menor de 1000 significa que su cifra es de 3 dígitos, saliendo de la iteración y actualizando el precio del documento.

Comando

```
db.aficiones.find().forEach(
        function (doc) {
        if (doc.Tema == "Fútbol" || doc.Tema=="Baloncesto" ){
                while(doc.Precio > 1000)
                        doc.Precio= doc.Precio / 10;
        }
        db.aficiones.update( { _id: doc._id } , doc );
});
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
   "_id" : ObjectId("5746cf6be6046b47d407361e"),
   "Tema" : "Fútbol",
   "Apodo" : "Mike",
   "Nombre" : "Cristiano Ronaldo",
   "Puntuacion" : 9.0,
   "Precio" : 1000.0,
   "Caracteristicas" : {
      "Nacionalidad" : "Portugues",
      "Posiciones" : [
```

```
            "Extremo",
            "Delantero"
        ]
    },
    "ValorCalidad" : 9e-06,
    "Descuento" : 1
}
{
    "_id" : ObjectId("5746cf6be6046b47d407361f"),
    "Tema" : "Fútbol",
    "Apodo" : "Mike",
    "Nombre" : "Isco",
    "Puntuacón" : 8.0,
    "Precio" : 600.0,
    "Características" : {
        "Nacionalidad" : "Español",
        "Posiciones" : [
            "Medio",
            "MediaPunta"
        ]
    },
    "Descuento" : NaN
}
```


Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
    "_id" : ObjectId("5746cfb0e6046b47d4073660"),
    "Tema" : "Baloncesto",
    "Apodo" : "NBA",
    "Nombre" : "Allen Iverson",
    "Puntuacion" : 10.0,
    "Precio" : 500.0,
    "Mote" : "The Answer",
    "PPP" : 22.7,
    "ValorCalidad" : 2e-05,
    "Descuento" : 0.0
}
{
    "_id" : ObjectId("5746cfb0e6046b47d4073661"),
    "Tema" : "Baloncesto",
    "Apodo" : "NBA",
    "Nombre" : "Stephen Curry",
    "Puntuacion" : 9.5,
    "Precio" : 450.0,
    "Mote" : "Chef Curry",
    "PPP" : 22.4,
    "ValorCalidad" : 2.11111111111111e-05,
    "Descuento" : 0.5
}
{
    "_id" : ObjectId("5746cfb0e6046b47d4073662"),
    "Tema" : "Baloncesto",
    "Apodo" : "NBA",
    "Nombre" : "Carmelo Anthony",
    "Puntuacion" : 10.0,
    "Precio" : 400.0,
    "Mote" : "Melo",
    "PPP" : 25.2,
```

```
  "ValorCalidad" : 2.5e-05,
  "Descuento" : 0.0
}
```


b.3) Añadir el precio a MotoGP y Ajedrez con valor fijo de 100 (en un solo *update*)

Solución

En un solo update procedo a actualizar el documento, si el tema del documento es “Ajedrez” o “MotoGP” actualiza el precio a 100, y “multi:true” para actualizar todos los documentos coincidentes.

Comando

```
db.aficiones.update( { $or: [ { Tema: "Ajedrez" } , { Tema: "MotoGP" } ] } , { $set: { Precio: 100 } } , { multi: true } )
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
  "_id" : ObjectId("5746cf97e6046b47d4073647"),
  "Tema" : "Ajedrez",
  "Apodo" : "Bobby",
  "Nombre" : "Alexander Alekhine undefined undefined undefined",
  "Nacionalidad" : "Russia",
  "ELO" : 2835.0,
  "Descuento" : NaN,
  "Precio" : 100.0
}

{
  "_id" : ObjectId("5746cf97e6046b47d4073648"),
  "Tema" : "Ajedrez",
  "Apodo" : "Bobby",
  "Nombre" : "Mikhail Botvinnik undefined undefined undefined",
  "Nacionalidad" : "Russia",
  "ELO" : 2875.0,
  "Descuento" : NaN,
  "Precio" : 100.0
}
```


Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
  "_id" : ObjectId("5746cf74e6046b47d4073624"),
  "Tema" : "MotoGP",
  "Apodo" : "MGP",
  "NumeroPilotos" : 2.0,
  "CarrerasGanadas" : 15.0,
  "PrimerPiloto" : "Andrea Dovizioso",
  "SegundoPiloto" : "Andrea Iannone",
  "Descuento" : NaN,
  "Nombre" : "Ducati Team",
  "Precio" : 100.0
}
{
```

```
    "_id" : ObjectId("5746cf74e6046b47d4073625"),
    "Tema" : "MotoGP",
    "Apodo" : "MGP",
    "NumeroPilotos" : 2.0,
    "CarrerasGanadas" : 1.0,
    "PrimerPiloto" : "Jack Miller",
    "SegundoPiloto" : "Tito Rabat",
    "Descuento" : NaN,
    "Nombre" : "Estrella Galicia 0,0 Marc VDS",
    "Precio" : 100.0
}
```

b.4) En MotoGP: Atributo incorrecto: hay "Puntuacón" donde debería haber "Puntuacion" sin perder su valor

Solución

Actualizo en “MotoGP” el nombre del campo y utilizo el operador $rename para cambiar el nombre del atributo, “Puntuacón” por “Puntuacion” y “multi:true” para actualizar todos los documentos coincidentes.

Comando

```
db.aficiones.update( { Tema: "MotoGP" } , { $rename: { Puntuacón: "Puntuacion" } } , { multi: true } )
```

Salida

No hay salida para mostrar porque motogp no dispone de un número para puntuación, este campo aparece vacío.

b.5) **(sacar nota)** En Ajedrez : unir los valores de Nombre y Apellidos en el atributo Nombre. Quitar el atributo Apellidos.

Recorro toda la colección, y si el tema del documento es “Ajedrez”, procedo a unir los valores de nombre y apellidos y lo guardo en una variable, luego elimino el campo Apellidos con $unset y actualizo el “Nombre” con el operador $set del documento

Divido entre 10 hasta que el precio sea menor a 1000, cuando es menor de 1000 significa que su cifra es de 3 dígitos, saliendo de la iteración y actualizando el precio del documento.

Solución

Comando

```
db.aficiones.find().forEach(
        function(doc) {
                var cadena;
                if (doc.Tema == "Ajedrez"){
                        if(doc.Apellidos != null){
                        cadena =doc.Nombre + " " + doc.Apellidos;
                        db.aficiones.update( { _id: doc._id } , { $unset: { Apellidos: "" } } );
                        db.aficiones.update( { _id: doc._id } , { $set: { Nombre: cadena } } );
                        }
```

```
                    }
            }
)
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
   "_id" : ObjectId("5746cf97e6046b47d4073645"),
   "Tema" : "Ajedrez",
   "Apodo" : "Bobby",
   "Nombre" : "Wilhelm Steinitz",
   "Nacionalidad" : "Austria",
   "ELO" : 2880.0,
   "Descuento" : NaN,
   "Precio" : 100.0
}

{
   "_id" : ObjectId("5746cf97e6046b47d4073646"),
   "Tema" : "Ajedrez",
   "Apodo" : "Bobby",
   "Nombre" : "José Capablanca",
   "Nacionalidad" : "Cuba",
   "ELO" : 2860.0,
   "Descuento" : NaN,
   "Precio" : 100.0
}
```


c) Esos errores son ocasionados porque la colección no tenía un validador automático. Crea una colección *misAficiones* solo para tu Tema (afición) y define el validador adecuado siguiendo las pautas del método en *validar-con-Schema.js* de la carpeta /ejemplosMongodb-en-Teoria/ . Inserta en *misAficiones* los documentos de tu tema que están en colección *aficiones* (puedes hacerlo copiándolos e insertándolos con la opción del menú *insert*). Deben superar la validación para que se inserten.
Nota: ver otro ejemplo del `$jsonSchema` en https://docs.mongodb.com/manual/core/schema-validation/

Solución

Me he guiado del enlace del ejemplo puesto en el enunciado (https://docs.mongodb.com/manual/core/schema-validation/), para aplicarlo en el validador de mi afición, especifico unas reglas de validación para la colección “misAficiones” utilizando un esquema json, donde los campos Tema y Nombre siempre tienen que tener un valor(no pueden ser nulos).

Validador para mi afición

```
db.createCollection("misAficiones", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "Tema", "Nombre"],
      properties: {
        Tema: {
          bsonType: "string",
          description: "nombre del tema de aficiones"
        },
        Nombre: {
```

```
            bsonType: "string",
            description: "nombre del objeto"
          },
                        Puntuacion: {
                                bsonType: [ "double" ],
                                 minimum: 0.0,
            maximum: 10.0,
            description: "puntuacion del objeto"
          },
                                Precio: {
            bsonType: [ "double" ],
            description: "precio del objeto"
          },
                        Autor: {
            bsonType: "string",
            description: "nombre del autor del objeto"
          },
                                Fecha: {
            bsonType: "string",
            description: "fecha de creación del objeto"
          },
                                 Periodo: {
            bsonType: "string",
            description: "periodo del objeto"
          }
        }
      }
    }
}
```

Comando para insertar un documento de mi aficion

```
db.misAficiones.insert(
  {
   "Tema" : "Arte",
   "Nombre" : "La noche estrellada",
   "Puntuacion" : 7,
   "Precio" : 70.0,
          "Autor" : "Vincent Van gogh",
   "Fecha" : "1889",
   "Periodo" : "Posimpresionismo"
  }
)
```

Salida después de ejecutar la inserción

```
{
   "_id" : ObjectId("5ed23daeb9eab4aa3e6db690"),
   "Tema" : "Arte",
   "Nombre" : "La noche estrellada",
   "Puntuacion" : 7.0,
   "Precio" : 70.0,
   "Autor" : "Vincent Van gogh",
   "Fecha" : "1889",
   "Periodo" : "Posimpresionismo"
}
```

d) Queremos evitar repeticiones de datos, ej.: tener el mismo libro repetido para todos los que les guste. Para ello queremos normalizar *aficiones* para tener una colección *solodatos* con la lista de detalles de las *aficiones*. Además, para no estropear *aficiones*, crea otra colección *soloaficiones* con lo que debe quedar en *aficiones* después de normalizarla: quedarán los atributos más frecuentes en las consultas: los Campos Obligatorios del enunciado, y el identificador del nuevo objeto con los detalles en *solodatos*.

Solución

```
//Creo las 2 colecciones “soloDatos” y “soloAficiones”
db.createCollection("soloDatos")
db.createCollection("soloAficiones")
```

Proyecto todos los campos de aficiones en detalles y componentes en el primer pipeline , luego en el segundo pipeline proyecto los campos que quiero que contenga “Componentes” (Tema, apodo nombre, puntuación, precio e identificador), luego en la función inserto detalles en soloDatos y Componentes en soloAficiones.

Comando

```
db.aficiones.aggregate(
{ $project: { detalles: "$$CURRENT",Componentes: "$$CURRENT" } } ,
{ $project: { Componentes: { Tema: 1, Apodo: 1, Nombre: 1, Puntuacion: 1, Precio: 1, identificador:
"$_id" } , detalles: "$$CURRENT.detalles" } }
).forEach(
        function(doc) {
                db.soloDatos.insert(doc.detalles);
                db.soloAficiones.insert(doc.Componentes);
        }
)
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando

SoloAficiones


```
{
   "_id" : ObjectId("5ed40f5635fe4eee626994e2"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El temor del hombre sabio",
   "Puntuacion" : 8.2,
   "Precio" : 24.9,
   "identificador" : ObjectId("5746cf61e6046b47d4073615")
}

{
   "_id" : ObjectId("5ed40f5635fe4eee626994e3"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El nombre del viento",
   "Puntuacion" : 7.9,
   "Precio" : 20.7,
   "identificador" : ObjectId("5746cf61e6046b47d4073616")
}
```

SoloDatos


```
{
   "_id" : ObjectId("5746cf61e6046b47d4073615"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El temor del hombre sabio",
   "Puntuacion" : 8.2,
   "Precio" : 24.9,
   "genero" : "Fantasia",
   "autor" : "Patrick Rothfuss",
   "paginas" : 1200.0,
   "ISBN" : 9788401339639.0,
   "ValorCalidad" : 32.9317269076305,
   "Descuento" : 1.8
}

{
   "_id" : ObjectId("5746cf61e6046b47d4073616"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El nombre del viento",
   "Puntuacion" : 7.9,
   "Precio" : 20.7,
   "genero" : "Fantasia",
   "autor" : "Patrick Rothfuss",
   "paginas" : 880.0,
   "ISBN" : 9788401337208.0,
   "ValorCalidad" : 38.1642512077295,
   "Descuento" : 2.1
}
```


e) El efecto del apartado anterior es que ahora la consulta completa uniendo ambas colecciones es más compleja y más lenta. Cómo se hace?

Solución

Para realizar una coincidencia de igualdad entre un campo de los documentos de entrada con un campo de los documentos de la colección "unida", la $lookup

Realiza una unión de todos los campos de soloDatos con todos los campos del identificador que hace referencia a un elemento de la colección “soloAficiones”.

Comando

```
db.soloAficiones.aggregate( [{ $lookup: { from: "soloDatos" , localField: "identificador" , foreignField: "_id" , as: "detalles" } }] )
```

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
   "_id" : ObjectId("5ed40f5635fe4eee626994e2"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
```

```
  "Nombre" : "El temor del hombre sabio",
  "Puntuacion" : 8.2,
  "Precio" : 24.9,
  "identificador" : ObjectId("5746cf61e6046b47d4073615"),
  "detalles" : [
    {
      "_id" : ObjectId("5746cf61e6046b47d4073615"),
      "Tema" : "Libros",
      "Apodo" : "Turre",
      "Nombre" : "El temor del hombre sabio",
      "Puntuacion" : 8.2,
      "Precio" : 24.9,
      "genero" : "Fantasia",
      "autor" : "Patrick Rothfuss",
      "paginas" : 1200.0,
      "ISBN" : 9788401339639.0,
      "ValorCalidad" : 32.9317269076305,
      "Descuento" : 1.8
    }
  ]
}

{
  "_id" : ObjectId("5ed40f5635fe4eee626994e3"),
  "Tema" : "Libros",
  "Apodo" : "Turre",
  "Nombre" : "El nombre del viento",
  "Puntuacion" : 7.9,
  "Precio" : 20.7,
  "identificador" : ObjectId("5746cf61e6046b47d4073616"),
  "detalles" : [
    {
      "_id" : ObjectId("5746cf61e6046b47d4073616"),
      "Tema" : "Libros",
      "Apodo" : "Turre",
      "Nombre" : "El nombre del viento",
      "Puntuacion" : 7.9,
      "Precio" : 20.7,
      "genero" : "Fantasia",
      "autor" : "Patrick Rothfuss",
      "paginas" : 880.0,
      "ISBN" : 9788401337208.0,
      "ValorCalidad" : 38.1642512077295,
      "Descuento" : 2.1
    }
  ]
}
```


f) **(para nota)** Crea una colección *resconstruida* que sea el resultado de la unión de *solodatos* y *soloaficiones* (el resultado del apartado anterior)

Solución

// Creo colección reconstruida

db.createCollection("reconstruida")

Comando

Realiza una unión de todos los campos de soloDatos con todos los campos del identificador que hace referencia a un elemento de la colección “soloAficiones” mediante el operador $lookup, y luego el resultado de la unión lo escribo en la colección “reconstruida”

db.soloAficiones.aggregate([{ $lookup: { from:"soloDatos", localField:"identificador", foreignField:"_id", as:"detalles" } } , { $out: "reconstruida" }])

Salida

Algunos ejemplos de salida de los documentos después de ejecutar el comando


```
{
   "_id" : ObjectId("5ed40f5635fe4eee626994e2"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El temor del hombre sabio",
   "Puntuacion" : 8.2,
   "Precio" : 24.9,
   "identificador" : ObjectId("5746cf61e6046b47d4073615"),
   "detalles" : [
      {
         "_id" : ObjectId("5746cf61e6046b47d4073615"),
         "Tema" : "Libros",
         "Apodo" : "Turre",
         "Nombre" : "El temor del hombre sabio",
         "Puntuacion" : 8.2,
         "Precio" : 24.9,
         "genero" : "Fantasia",
         "autor" : "Patrick Rothfuss",
         "paginas" : 1200.0,
         "ISBN" : 9788401339639.0,
         "ValorCalidad" : 32.9317269076305,
         "Descuento" : 1.8
      }
   ]
}

{
   "_id" : ObjectId("5ed40f5635fe4eee626994e3"),
   "Tema" : "Libros",
   "Apodo" : "Turre",
   "Nombre" : "El nombre del viento",
   "Puntuacion" : 7.9,
   "Precio" : 20.7,
   "identificador" : ObjectId("5746cf61e6046b47d4073616"),
   "detalles" : [
      {
         "_id" : ObjectId("5746cf61e6046b47d4073616"),
         "Tema" : "Libros",
         "Apodo" : "Turre",
         "Nombre" : "El nombre del viento",
```

```
        "Puntuacion" : 7.9,
        "Precio" : 20.7,
        "genero" : "Fantasia",
        "autor" : "Patrick Rothfuss",
        "paginas" : 880.0,
        "ISBN" : 9788401337208.0,
        "ValorCalidad" : 38.1642512077295,
        "Descuento" : 2.1
      }
  ]
}
```


APARTADO 5.- (REVISAR el de la semana pasada con lo explicado en clase de Teoría)

Siguiendo las pautas para diseñar una BD no-sql en las diapositivas de la Teoría: diseña tú una BD de tema libre y describe qué operaciones quiere hacer. Teniendo en cuenta que sea un tema donde una BD tipo SQL no sea adecuada.

Solución

He decidido diseñar una base de datos centrados en el uso que le pueda dar una empresa a los datos de sus empleados, con el objetivo de hacer una mejor gestión de recursos, mejorar el bienestar de sus trabajadores y tener un mayor conocimiento de las estadísticas de sus empleados en el desempeño de sus trabajos en los proyectos, como por ejemplo: el tiempo que habitualmente dedican a un proyecto, cuantos proyectos tuvieron un resultado optimo, cual son las aficiones de sus empleados (para realizar actividades grupales), sus platos preferidos etc.

Según lo entendido en clase, para un mejor diseño de mi BD, he decidido responder a las siguientes preguntas de las diapositivas de teoría, que me ha ayudado de guía a la hora de hacer un mejor diseño de mi base de datos.

¿Qué objetos básicos necesito?

En mi BD los objetos básicos son tres: Empresa, Proyecto y Empleados.

¿Qué relaciones hay entre esos objetos: 1 a 1, 1 a N, N a N

- Un empleado trabaja en una empresa, y una empresa puede tener varios empleados (1 a N)

-Un empleado puede tener varios proyectos, y un proyecto puede tener varios empleados (N a N)

- Un proyecto tiene una empresa y una empresa puede tener varios proyectos (1 a N)

¿Con que frecuencia:

se añaden objetos nuevos?

- La frecuencia de agregar un nuevo proyecto o empleados no será alta, por ejemplo, que un nuevo empleado se incorpore a un proyecto puede ser de 4 o 6 a lo largo de un año, sabiendo que los proyectos tienen una duración mínima de 2 meses

se borran objetos?

-La frecuencia de eliminar un proyecto o empleado, también es baja, la frecuencia es mínima tanto si un empleado se va del proyecto o de la empresa

se modifican objetos?

- La frecuencia también es baja, dependerá del número de proyectos que este asignado el empleado a lo largo de un año.

se acceden objetos?

La frecuencia es alta, porque unos de los principales objetivos es hacer un baremo de sus empleados, el mayor número de operaciones serán de tipo consultas.

¿Cómo serán las consultas de un objeto?

¿desde la clave?

No usare la clave, su uso se reserva para el sistema

¿desde qué campos?

Campos como nombre del empleado, puntuación del empleado , afición del empleado, nombre del proyecto, salario

¿Comparando con otros campos?

¿Cómo serán las consultas de un grupos de tipos de objeto?

Pues las consultas de grupos de tipos de objeto serán por ejemplo consultas para obtener una lista de empleados que están participando en un proyecto, una lista de empleados agrupados por una profesión en común, una lista de empleados agrupados por un salario, etc.

¿Escribe qué consultas necesitas hacer a esta BD?

Consulta para conocer la media que habitualmente tarda un empleado en terminar un proyecto y a su vez sacar el nombre del empleado, correo del empleado y salario.

Consulta para que, a partir del nombre del proyecto, también pueda obtener los datos de los usuarios que están en el proyecto como el nombre y profesión.

búsqueda para obtener el salario de los empleados mediante un salario dado

búsqueda para saber que empleados participan en un determinado proyecto mediante el nombre del proyecto y se asocie a una puntuación

Búsqueda para obtener una lista de empleados mediante una puntuación, pueda tener los datos del empleado que me interesen asociados a la puntuación.

Búsqueda para obtener una lista de empleados para la realización de actividades mediante una afición/hobby dada y se asocie el nombre del usuario y el correo.

<u>En conclusión</u>

En mi base de datos, el mayor número de operaciones que van a predominar va a ser de tipo lectura, concretamente operaciones de consultas o búsquedas.

No va disponer de esquemas rígidos, puedo colocar datos anidados dentro de las colecciones, ejemplo si tengo un registro de proyectos, dentro puedo poner un registro parcial de datos del empleado(datos que me interesen en la consulta) , de manera que cuando consulte el proyecto este asociado con datos del usuario.

También en el momento que vaya a leer los datos de mi base de datos, en lugar de hacer una consulta que vaya a recolectar los datos de varias partes, solamente tengo que leer una colección (Desnormalización).

APARTADO 5.- EXTRA (REVISAR el de la semana pasada con lo explicado en clase de Teoría)

Deseamos introducir elementos compuestos, ej.: como en un equipo de futbol si incluimos cada jugador con sus datos personales. Y queremos hacer muchas consultas sobre esos elementos compuestos ej.: datos personales. ¿Conviene normalizar o desnormalizar?. ¿Cómo debería quedar la representación de la colección?

Solución

¿Conviene normalizar o desnormalizar?

Después de la explicación de la clase de teoría, he llegado a la conclusión que lo conveniente sería normalizar, es decir ponerlo en 2 colecciones con la intención de hacer mas sencillas las consultas que se hagan entre equipo y jugadores .

He decidido normalizar porque la idea es de separar determinadas propiedades de un objeto “Equipos de Futbol” y separarlos en 2 colecciones “Equipo” y “Jugadores”, almacenándolos como documentos aparte en otra colección.

Con eso evitaría redundancia en el caso de que los jugadores estén en equipos nacionales, y así tener una sola copia de jugadores y que se referencia en varios documentos.

Luego las relaciones entre los objetos “Equipos y jugadores” serán de 1-N:

“Un equipo puede tener varios jugadores y un jugador puede tener 2 equipos (equipo de club y selecciones nacionales)”

En contra estaría que las operaciones de actualización no serían atómicas, que cuando tienes varias colecciones “Equipos y jugadores”, no puedes garantizar que haya una actualización atómica, es decir no se garantiza que la actualización se haga en una sola colección.

¿Cómo debería quedar la representación de la colección?

Una colección “Equipos” que contenga los datos del equipo con una lista de id de jugadores y otra colección “Jugadores” que contenga los datos del jugador y un id del equipo.

Colección equipo


```
{
   "_id" : ObjectId("5746cf61e6046b47d4073615"),
   "Name" : "Barcelona",
   "League" : "La Liga Santander",
```

```
   "Country" : "Spain",
   "Season" : "2020-2021",
   "Jugadores" : {[
         ObjectId("9746cf61e6046b47d4073613"),//id del jugador de Messi
         ObjectId("8746cf61e6046b41mh273616"),//id del jugador Griezman
         ]},
}

{
   "_id" : ObjectId("5746cf61e6046b47d4073611"),
   "Name" : "Real Madrid",
   "League" : " La Liga Santander ",
   "Country" : "Spain",
   "Season" : "2020-2021",
   "Jugadores" : {[
         ObjectId("1746cf61e6046b47d4073610"),//id del jugador luka modric
         ObjectId("5746cf3840ju6b47d407361f"),// id del jugador de Eden Hazard
         ]},
}
```

Colección de jugadores

```
//Barcelona

{
   "_id" : ObjectId("9746cf61e6046b47d4073613"),//id del jugador
   "Name" : "Messi",
   "Town" : "Argentina",
   "Age" : "32",
   "idEquipo" : "5746cf61e6046b47d4073615",//id del equipo Barcelona
}
{
   "_id" : ObjectId("8746cf61e6046b41mh273616"),//id del jugador
   "Name" : “Antoine Griezmann",
   "Town" : "France",
   "Age" : "29",
   "idEquipo" : "5746cf61e6046b47d4073615",//id del equipo Barcelona
}
//Real Madrid
{
   "_id" : ObjectId("1746cf61e6046b47d4073610"),//id del jugador de luka modric
   "Name" : "Luka Modric",
   "Town" : "Croatia",
   "Age" : "34",
   "idEquipo" : "5746cf61e6046b47d4073611",//id del equipo Real madrid
}
{
   "_id" : ObjectId("5746cf3840ju6b47d407361f "),//id del jugador de Eden Hazard
```

```
   "Name" : "Eden Hazard ",
   "Town" : "Belgium ",
   "Age" : "29",
   "idEquipo" : "5746cf61e6046b47d4073611",//id del equipo Real madrid
}
```