Bases de datos NoSQL

Autor: Miguel Benayas Penas, Marzo 2021

Índice:

1. Introducción

En este proyecto se procederá a la realización de consultas a partir de la colección “movies”. Se trata de 26 consultas explicadas en este documento donde, a su vez, se muestran los resultados obtenidos.

Las líneas de código presentes en el script se mostrarán en negrita a lo largo del documento. Los resultados de las consultas se mostrarán en cursiva y en verde.

2. Consultas realizadas

2.1 Pasos preliminares

En primer lugar se deben importar los datos de la colección. Esto se hace mediante el siguiente código:

```
const contents = [
  {
        content: "C:\\Users\\usuario\\Desktop\\Master UCM\\2 Bases de datos NoSQL - Alvaro Bravo\\TAREA\\movies.json",
    collection: "movies",
                                                idPolicy: "drop_collection_first", //overwrite_with_same_id|always_insert_with_new_id|insert_with_new_id_if_id_exists|skip_documents_with_existing_id|abort_if_id_already_exists|drop_collection_first|log_errors
    //Use the transformer to customize the import result
    //transformer: (doc)=>{ //async (doc)=>{
    //  doc["importDate"]= new Date()
    //  return doc; //return null skips this doc
    //}
  }
];

mb.importContent({
  connection: "localhost",
  database: "clases",
  fromType: "file",
```

batchSize: 2000,
contents})

En primer lugar se declaran los “contents” donde se especifica por ejemplo qué archivo se va a importar, como se va a llamar a la colección (movies) y la política de eliminar la colección si esta ha sido importada antes. El uso del “drop” es relevante al ser los cambios en la colección permanentes con mongoDB.

La segunda parte consiste en definir en qué servidor (“localhost”) y base de datos (“clases”) se va a importar contents.

El resultado al correr estas líneas es satisfactorio, sin fallos:

```
A total of 28795 document(s) have been imported into 1 collection(s).
{
  "movies": {
    "nInserted": 28795,
    "nModified": 0,
    "nSkipped": 0,
    "failed": 0
  }
}
```

Se procede a observar que, en efecto, la nueva base de datos “clases” aparece en el servidor.

show dbs
```
admin   0.000GB
clases  0.002GB
config  0.000GB
local   0.000GB
```

Se selecciona clases y se comprueba que es con la que se va a trabajar.

use clases
db
```
"clases"
```

Finalmente, se comprueba que la colección movies está presente en la base de datos clases.

show collections
```
movies
```

2.2 Consultas

En este subapartado se detallan las consultas realizadas para la práctica. En las las consultas libres se ha tratado de reflejar la versatilidad del pipeline de agregación mostrada en las clases. Es por ello que se ha optado por el uso de fechas y comandos como “$lookup” y “$replaceWith”.

// 1) Analizar con find la colección.
db.movies.find()
/* 1 createdAt:11/7/2019, 8:35:00 AM*/
{
"_id" : ObjectId("5dc41d848cc90c0c00a2e258"),
"title" : "Caught",
"year" : 1900,
"cast" : [ ],
"genres" : [ ]
},

/* 2 createdAt:11/7/2019, 8:35:00 AM*/
{
"_id" : ObjectId("5dc41d848cc90c0c00a2e259"),
"title" : "After Dark in Central Park",
"year" : 1900,
"cast" : [ ],
"genres" : [ ]
},
…

// 2) Contar cuántos documentos (películas tiene cargado).
db.movies.find().count()
28795

// 3) Insertar una película.
var nuevo_item = { "title": "paraborrar", "year": 2050, "cast": ["c1", "c2"], "genres": ["g1", "g2"] }
db.movies.insertOne(nuevo_item)
{
"acknowledged" : true,
"insertedId" : ObjectId("605b13939252df445c7f60a5")
}
db.movies.find().count() // vemos que la cuenta está ahora en 28795+1=28796
28796

// 4) Borrar la película insertada en el punto anterior.
db.movies.deleteMany({ "title": "paraborrar" }) // mejor que deleteOne() por si hemos dado (sin querer) varias veces al insertOne
{
"acknowledged" : true,
"deletedCount" : 1
}
db.movies.find().count() // vuelve a ser 28795
28795

// 5) Contar cuantas películas tienen actores (cast) que se llaman “and".
var query = { "cast": { $in: ["and"] } }
db.movies.find(query).count()
93

// 6) Actualizar los documentos cuyo actor (cast) tenga por error el valor “and” como si realmente fuera un actor. Para ello, se debe sacar únicamente ese valor del array cast.
var query = { "cast": { $in: ["and"] } }
var operacion = { $pull: { "cast": "and" } }
db.movies.updateMany(query, operacion)
{
"acknowledged" : true,
"matchedCount" : 93,
"modifiedCount" : 93
}

// 7) Contar cuantos documentos (películas) no tienen actores (array cast).
var query = { "cast": { $size: 0 } }
db.movies.find(query).count()
986

// 8) Actualizar TODOS los documentos (películas) que no tengan actores (array cast),
// añadiendo un nuevo elemento dentro del array con valor Undefined.
var query = { "cast": { $size: 0 } }
var operacion = { $push: { "cast": "Undefined" } }
db.movies.updateMany(query, operacion)
{
"acknowledged" : true,
"matchedCount" : 986,
"modifiedCount" : 986
}

```
var query = { "cast": "Undefined" }
db.movies.find(query)
/* 1 createdAt:11/7/2019, 8:35:00 AM*/
{
        "_id" : ObjectId("5dc41d848cc90c0c00a2e258"),
        "title" : "Caught",
        "year" : 1900,
        "cast" : [
                "Undefined"
        ],
        "genres" : [ ]
},
…

// 9) Contar cuantos documentos (películas) no tienen Género (array genres)
var query = { "genres": { $size: 0 } }
db.movies.find(query).count()
901

// 10) Actualizar TODOS los documentos (películas) que no tengan géneros (array genres),
// añadiendo un nuevo elemento dentro del array con valor Undefined.
var query = { "genres": { $size: 0 } }
var operacion = { $push: { "genres": "Undefined" } }
db.movies.updateMany(query, operacion)
{
        "acknowledged" : true,
        "matchedCount" : 901,
        "modifiedCount" : 901
}

// Casos donde tanto cast como genres son Undefined
var query = { "genres": "Undefined" }
var query2 = { "cast": "Undefined" }
var filter = { $and: [query, query2] }
db.movies.find(filter)
/* 1 createdAt:11/7/2019, 8:35:00 AM*/
{
        "_id" : ObjectId("5dc41d848cc90c0c00a2e258"),
        "title" : "Caught",
        "year" : 1900,
        "cast" : [
                "Undefined"
        ],
        "genres" : [
```

```
                "Undefined"
        ]
},
```

// 11) Mostrar el año más reciente / actual que tenemos sobre todas las películas.

```
db.movies.aggregate(
   [
      { $sort: { "year": -1 } }, { $limit: 1 }, { $project: { "_id": 0, "year": 1 } }
   ]
)// no modifica colección
{
        "year" : 2018
}
```

// 12) Contar cuántas películas han salido en los últimos 20 años.
// Debe hacerse desde el último año que se tienen registradas películas en la colección, mostrando el resultado total de esos años.

En esta consulta se agrupa por años, se ordenan, se cogen los 20 más recientes y se suman los valores de la columna “total”.

```
var fase1 = { $group: { "_id": "$year", "total": { $sum: 1 } } }
var fase2 = { $project: { "_id": 0, "year": "$_id" "total": 1 } }
var fase3 = { $sort: { "year": -1 } }
var fase4 = { $limit: 20 }
var fase5 = { $group: { "_id": null, "total": { $sum: "$total" } } }
db.movies.aggregate([fase1, fase2, fase3, fase4, fase5])
{
        "_id" : null,
        "total" : 4787
}
```

// 13) Contar cuántas películas han salido en la década de los 60 (del 60 al 69 incluidos).

```
var fase1 = { $group: { "_id": "$year", "total": { $sum: 1 } } }
var fase2 = { $match: { "_id": { $gte: 1960, $lte: 1969 } } }
var fase3 = { $group: { "_id": null, "total": { $sum: "$total" } } }
db.movies.aggregate([fase1, fase2, fase3])
{
        "_id" : null,
        "total" : 1414
}
```

// 14) Mostrar el año u años con más películas mostrando el número de películas de ese año.
// Revisar si varios años pueden compartir tener el mayor número de películas.

Al agrupar por el número de películas guardando los años en un vector (“array”), podemos observar a simple vista si varios años comparten el tener el mayor número de películas tras usar el comando “$limit”. El resultado es que solo hay un año con el mayor número de películas (1919).

```
var docu = { "year": "$year" }
var fase1 = { $group: { "_id": "$year", "numero_peliculas": { $sum: 1 } } }
var fase2 = { $project: { "_id": 0, "year": "$_id", "numero_peliculas": 1 } }
var fase3 = { $group: { _id: "$numero_peliculas", "number_years": { $sum: 1 }, "items": {
$push: docu } } }
var fase4 = { $project: { "_id": 0, "numero_peliculas": "$_id", "years": "$items" } }
var fase5 = { $sort: { "numero_peliculas": -1 } }
var fase6 = { $limit: 1 }
var fase7={$project:{"pelis":"$numero_peliculas","_id":"$years.year"}}
db.movies.aggregate([fase1, fase2, fase3, fase4, fase5, fase6,fase7])
{
        "pelis" : 634,
        "_id" : [
                1919
        ]
}
```

// 15) Mostrar el año u años con menos películas mostrando el número de películas de ese año.
// Revisar si varios años pueden compartir tener el menor número de películas.

Como la consulta anterior solo que invirtiendo la ordenación. En este caso hay tres años con el menor número de películas.

```
var docu = { "year": "$year" }
var docu = { "year": "$year" }
var fase1 = { $group: { "_id": "$year", "numero_peliculas": { $sum: 1 } } }
var fase2 = { $project: { "_id": 0, "year": "$_id", "numero_peliculas": 1 } }
var fase3 = { $group: { _id: "$numero_peliculas", "number_years": { $sum: 1 }, "items": {
$push: docu } } }
var fase4 = { $project: { "_id": 0, "numero_peliculas": "$_id", "years": "$items" } }
var fase5 = { $sort: { "numero_peliculas": 1 } }
var fase6 = { $limit: 1 }
db.movies.aggregate([fase1, fase2, fase3, fase4, fase5, fase6])
```

```
{
        "numero_peliculas" : 7,
        "years" : [
                {
                        "year" : 1902
                },
                {
                        "year" : 1907
                },
                {
                        "year" : 1906
                }
        ]
}
```

// 16) Guardar en nueva colección llamada “actors” realizando la fase “$unwind” por actor.
// Después, contar cuantos documentos existen en la nueva colección.

Seleccionamos aquellos documentos que contienen algún valor en el campo “cast”. La “fase5” se necesita para evitar el error por duplicado de llaves.

```
var fase1 = { $set: { "size_of_cast": { $size: "$cast" } } }
var fase2 = { $match: { "size_of_cast": { $gte: 1 } } }
var fase3 = { $project: {"size_of_cast": 0 } }
var fase4 = { $unwind: "$cast" }
var fase5 = { $project: {"_id": 0 } } // Para evitar claves duplicadas en la db
var fase6 = { $out: "actors" }

db.movies.aggregate([fase1, fase2, fase3, fase4,fase5,fase6])

db.actors.find().count()
83224

show collections
actors
movies
```

// 17) Sobre actors (nueva colección), mostrar la lista con los 5 actores que han participado en más películas mostrando el número de películas en las que ha participado.

```
var fase1 = { $match: { "cast": { $ne: 'Undefined' }}}
var fase2 = { $group: { "_id": "$cast", "numero_peliculas": { $sum: 1 } } }
var fase3 = { $sort: { "numero_peliculas": -1 } }
var fase4 = { $limit: 5 }
db.actors.aggregate([fase1,fase2,fase3,fase4])
/* 1 */
{
        "_id" : "Harold Lloyd",
        "numero_peliculas" : 190
},
```

// 18) Sobre actors (nueva colección), agrupar por película y año mostrando las 5 en las que más actores hayan participado, mostrando el número total de actores.

```
var fase2= { $group: { "_id": { "title": "$title" , "year":"$year" } , "cuenta":{$sum:1 }}}
var fase3 = { $sort: { "cuenta": -1 } }
var fase4 = { $limit: 5 }
db.actors.aggregate([fase2,fase3,fase4])
/* 1 */
{
        "_id" : {
                "title" : "The Twilight Saga: Breaking Dawn - Part 2",
                "year" : 2012
        },
        "cuenta" : 35
},
```

// 19) Sobre actors (nueva colección), mostrar los 5 actores cuya carrera haya sido la más larga. Para ello, se debe mostrar cuándo comenzó su carrera, cuándo finalizó y cuántos años han pasado

Se ignoran los 'Undefined' y se agrupa por actor, guardando los años en un vector. A partir esos vectores se añaden tres columnas a la tabla con el máximo, mínimo y su resta, "anos". Finalmente se ordena por la columna "anos" y se muestran las cinco carreras más largas.

```
var fase1={ $match: { "cast": { $ne: 'Undefined' }}}
var docu={"year": "$year" }
var fase2 = { $group: { _id: "$cast", "years": { $push: docu } } }
var fase3= { $project: { "_id": 1, "years": "$years.year" } }
var fase4={ $addFields: { "comienza": {$min: "$years"}, "termina":{$max: "$years"} } }
var fase5={ $addFields: { "anos": {$subtract: ["$termina","$comienza"]} } }
```

```
var fase6={ $project: { "_id": 1, "comienza": 1 , "termina": 1 ,"anos":1} }
var fase7 = { $sort: { "anos": -1 } }
var fase8 = { $limit: 5 }
db.actors.aggregate([fase1,fase2,fase3,fase4,fase5,fase6,fase7,fase8])
/* 1 */
{
        "_id" : "Harrison Ford",
        "comienza" : 1919,
        "termina" : 2017,
        "anos" : 98
},
```

// 20) Sobre actors (nueva colección), Guardar en nueva colección llamada “genres” realizando la fase $unwind por genres. Después, contar cuantos documentos existen en la nueva colección

```
var fase1 = { $unwind: "$genres" }
var fase2 = { $project: {"_id": 0 } } // Para evitar claves duplicadas en la db
var fase3 = { $out: "genres" }
db.movies.aggregate([fase1,fase2,fase3])

db.genres.find().count()
33910
```

// 21) Sobre genres (nueva colección), mostrar los 5 documentos agrupados por “Año y Género” que más número de películas diferentes tienen mostrando el número total de películas.

El paso clave de esta agregación es “fase4”, donde se agrupa por año, género y películas. De esta forma se eliminan los duplicados tras aplicar “$project”.

```
var docu = { "title": "$title" }
var fase1= { $group: { "_id": { "year": "$year" , "genre":"$genres" } , "pelis": { $push: docu } }}
var fase2={ $unwind: "$pelis" }
var fase3= { $project: { "_id": 0,"ano_genero":"$_id", "pelis": "$pelis.title" } }
var fase4={$group:{"_id": { "ano_genero": "$ano_genero" , "pelis":"$pelis" }, "count":{$sum:1}}}
// Ajustamos al formato de la diapositiva, tras haber filtrado los duplicados en fase4
var fase5={$project:{"_id": { "year": "$_id.ano_genero.year" , "genre":"$_id.ano_genero.genre" }, "pelis":"$_id.pelis"}}

var fase6={$group:{"_id":"$_id", "count":{$sum:1}}}
var fase7 = { $sort: { "count": -1 } }
var fase8 = { $limit: 5 }
```

```
db.genres.aggregate([fase1,fase2,fase3,fase4,fase5,fase6,fase7,fase8])
/* 1 */
{
        "_id" : {
                "year" : 1919,
                "genre" : "Drama"
        },
        "count" : 291
},
```

// 22) Sobre genres (nueva colección), mostrar los 5 actores y los géneros en los que han participado con más número de géneros diferentes, se debe mostrar el número de géneros diferentes que ha interpretado.

Análogamente a la consulta anterior, se agrupa por actores y géneros para eliminar duplicados mediante las fases 3 y 4. Después, se agrupan los actores almacenando los géneros en arrays y se crea una columna con la dimensión de estos arrays, “numgeneros”. Finalmente, se ordena por numgeneros.

```
var fase1={ $match: { "cast": { $ne: 'Undefined' }}}
var fase2={ $unwind: "$cast" }
var fase3={ $group: { "_id": { "cast": "$cast" , "genres":"$genres" },"count":{$sum:1} }}
var fase4={$project:{"_id":"$_id.cast","genress":"$_id.genres"}}
var docu = { "generos": "$genress" }
var fase5={$group:{"_id":"$_id","numgeneros":{$sum:1},"generos": { $push: docu }}}
var fase6={$project:{"_id":1,"numgeneros":1, "generos":"$generos.generos"} }
var fase7 = { $sort: { "numgeneros": -1 } }
var fase8 = { $limit: 5 }
db.genres.aggregate([fase1,fase2,fase3,fase4,fase5,fase6,fase7,fase8])
/* 1 */
{
        "_id" : "Dennis Quaid",
        "numgeneros" : 20,
        "generos" : [
                "Satire",
                "Comedy",
                "Adventure",
                "Science Fiction",
                "Drama",
                "Animated",
                "Romance",
                "Horror",
                "Sports",
                "Suspense",
```

```
                "Action",
                "Musical",
                "Biography",
                "Fantasy",
                "Western",
                "Family",
                "Thriller",
                "Dance",
                "Disaster",
                "Crime"
        ]
},
```

// 23) Sobre genres (nueva colección), mostrar las 5 películas y su año correspondiente en los que más géneros diferentes han sido catalogados, mostrando esos géneros y el número de géneros que contiene.

```
var docu = { "generos": "$genres" }
var fase2={ $group: { "_id": { "title": "$title", "year":"$year"},"generos": { $push: docu } }}
var fase3={$project:{"_id":1, "generos":"$generos.generos"} }
var fase4={ $unwind: "$generos" }
var fase5={ $group: { "_id": { "_id": "$_id" , "generos":"$generos" },"count":{$sum:1} }}
var fase6={$project :{"_id":"$_id._id", "generos":"$_id.generos"}}
var docu = { "generos": "$generos" }
var fase7={$group:{"_id":"$_id","numgeneros":{$sum:1},"generos": { $push: docu }}}

var fase8 = { $sort: { "numgeneros": -1 } }
var fase9 = { $limit: 5 }
var fase10={ $project: {generos:"$generos.generos"} }
db.genres.aggregate([fase2,fase3,fase4,fase5,fase6,fase7,fase8,fase9,fase10])
/* 1 */
{
        "_id" : {
                "title" : "American Made",
                "year" : 2017
        },
        "generos" : [
                "Comedy",
                "Thriller",
                "Historical",
                "Drama",
                "Biography",
                "Action",
                "Crime"
```

```
        ]
},
```

// 24) Consulta libre sobre el pipeline de agregación
// Crear la colección "guerras". Contar cuantas películas salieron en el último año de alguna guerra de la colección "guerras".

Se usa el comando “$lookup” ya que se quiere seleccionar documentos con valores coincidentes en dos campos de dos bases de datos diferentes, “year” en movies y “fin” en guerras.

```
db.guerras.drop()// evitar insertar muchas veces y generar duplicados
db.guerras.insert([
  { "_id" : 1, "nombre" : "Guerra Mundial I", "inicio" : 1914, "fin" : 1918 },
  { "_id" : 2, "nombre" : "Guerra Mundial II", "inicio" : 1939, "fin" : 1945 },
  { "_id" : 3, "nombre" : "Guerra Civil Espanola", "inicio" : 1936, "fin" : 1939 }
])

fase1=    {   $lookup: { from: "movies", localField: "fin", foreignField: "year",   as: "movies_docs" } }
fase2={ $unwind: "$movies_docs" }
var fase3={ $group: { "_id": null,"count":{$sum:1} }}
db.guerras.aggregate([fase1,fase2,fase3])
{
        "_id" : null,
        "count" : 882
}
```

// 25) Consulta libre sobre el pipeline de agregación
// Remplazar el documento de la Segunda Guerra Mundial de la colección "guerras" añadiendo un nuevo campo con la duración de esa guerra y guardar ese resultado en una nueva colección

Para esta consulta se hace uso del comando “$replaceWith”, tras haber impuesto la condición con el comando “$match” de solo la Segunda Guerra Mundial.

```
db.guerras1.drop()
var fase1={ $match: { "nombre": "Guerra Mundial II" } }
var fase2=   { $replaceWith: { _id: "$_id", nombre: "$nombre", inicio:"$inico", fin:"$fin", duracion: { $subtract: [ "$fin", "$inicio"]} } }
var fase3 = { $out: "guerras1" }
db.guerras.aggregate([fase1,fase2,fase3])
db.guerras1.find()
{
        "_id" : 2,
```

```
        "nombre" : "Guerra Mundial II",
        "fin" : 1945,
        "duracion" : 6
}
```

// 26) Consulta libre sobre el pipeline de agregación

// Calcular cuántos años han pasado desde el fin de las guerras de la colección “guerras”

Se crea un nuevo campo con la fecha actual. Se selecciona solo el año en las fechas mediante “$replaceWith”. Finalmente, se crea el campo “tiempo_desde_fin“ con la resta del año actual y el del fin de las guerras.

```
var fase1=   { $set: { current_date: "$$NOW" } }
var fase2=       { $replaceWith:{  _id:  "$_id",  nombre:  "$nombre",  inicio:"$inico",
fin:"$fin",current_date:{ $year: "$current_date" }}}
var fase3={ $addFields: {tiempo_desde_fin:{ $subtract: [ "$current_date","$fin"]}} }
var fase4={ $project: {"_id":0,"nombre": 1 ,"tiempo_desde_fin":1 } }
db.guerras.aggregate([fase1,fase2,fase3,fase4])
/* 1 */
{
        "nombre" : "Guerra Mundial I",
        "tiempo_desde_fin" : 103
},

/* 2 */
{
        "nombre" : "Guerra Mundial II",
        "tiempo_desde_fin" : 76
},

/* 3 */
{
        "nombre" : "Guerra Civil Espanola",
        "tiempo_desde_fin" : 82
}
```