Entrega Mongodb

“Los lenguajes de programación más utilizados”

Guillermo Sánchez-Mariscal

1. Introducción

Este conjunto de datos contiene datos sobre los lenguajes de programación más populares de 2004 a 2020.

El dataset lo he obtenido en Kaggle (https://www.kaggle.com/muhammadkhalid/most-popular-programming-languages-since-2004), viene en formato csv. A priori, puede parecer que el conjunto de datos es escaso en variables, pero me parecía muy interesante de analizar y de mostrar.

Para importar los datos he utilizado el siguiente script:

```
const contents = [
    {
        content: "Most Popular Programming Languages from 2004 to 2020 V3.csv",
        collection: "languagesByMonth",
        idPolicy: "overwrite_with_same_id",
    }
];

mb.importContent({
    connection: "localhost",
    database: "p_languages",
    fromType: "file",
    batchSize: 2000,
    contents
})
```

Guardo en la colección `languagesByMonth` de la base de datos `p_languages` los datos del .csv
Si hacemos un `.find({})` para ver los datos, vemos como se importaran:

_id	Date	Abap	Ada	C/C++	C#	Cobol	Dart	D
60167ea7f719e431dc62	April 2005	0,34	0,36	9,85	5,42	0,41	0	2
60167ea7f719e431dc62	April 2006	0,36	0,25	8,48	6,67	0,44	0	2
60167ea7f719e431dc62	April 2007	0,38	0,25	8,11	6,79	0,39	0	1
60167ea7f719e431dc62	April 2008	0,36	0,22	8,03	7,33	0,4	0	1
60167ea7f719e431dc62	April 2009	0,38	0,23	8,67	7,67	0,37	0	1

Al ver los datos, me he dado cuenta de que podían estar mejor organizados, y que hay campos que se deberían modificar, pero antes de hacerlo voy a explicar que significa cada campo.

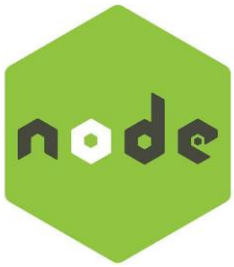
Explicación de los datos:

- La columna 1 (`Date`), nos dice el mes y el año en el que se extraen los datos.
- En las columnas 2 a la 29, almacena los distintos lenguajes de programación de los que se obtiene el porcentaje de uso, donde la clave es el nombre del lenguaje y el valor es el porcentaje de uso sobre el 100%

No es la mejor estructura de los datos puesto que, para cada fecha se repiten todos los lenguajes, incluso lenguajes que aún no existían. Para mi lo mejor sería organizarlo por lenguaje de programación y sólo utilizar las fechas para cada lenguaje, cuando ya estaba en uso. Además, voy a cambiar el formato de la fecha para poder ordenar correctamente.

La verdad es que todo lo relacionado con mongo, y estructuras de bases de datos me parece muy interesante y decidí hacer el servidor de una API REST para poder acceder a estos datos desde una aplicación web, si se quisiese, separando cliente y servidor.

Todo esto lo he realizado utilizando node y express.



2. Programación del servidor

La idea de hacer todo esto era poder conectarlo con una vista, y subirlo a algún servicio en la nube que para poder acceder de forma remota.

Lo primero que hago es conectar el cliente de mongo a mi base de datos. Después utilizo express para que, desde el puerto 3000 de mi localhost escuche las peticiones que se realicen. Y finalmente con el método .get de express defino que respuesta devuelvo cuando me hacen el request, en mi caso van a ser las distintas queries. Todas las queries realizadas las explicaré en el siguiente aprtado.

Código del servidor:

```
const initServer = async () => {

  //Creo el cliente
  const client = await MongoClient.connect(url, {
      useUnifiedTopology: true
    })

  const db = client.db('p_languages')
  const languages = db.collection('languagesByMonth')

  app.listen(3000, function() {
    console.log('Listening on 3000')
  })

  app.get('/', (req, res) => {
    // Todo index page to view results
    res.sendFile(__dirname + '/index.html')
  })

  app.get('/example_query', async function(req, res) {
    const items = await languages.find().toArray({})
    res.send(items)
  })
}
```

2. Queries

2.1 Limpieza y puesta a punto de los datos

Cambiar el tipo de fecha a Date

La fecha viene dada como un string (Ej, Abril 2005)

```
db.languagesByMonth.updateMany(
    { },
    [{ "$set": { "Date": { "$toDate": "$Date" } }}]
);
```

	_id	Date	Abap	Ada	C/C++
1	60167ea7f719e431dc62	1/12/2020 1:00:00	0,41	0,61	6,11
2	60167ea7f719e431dc62	1/11/2020 1:00:00	0,39	0,59	5,88

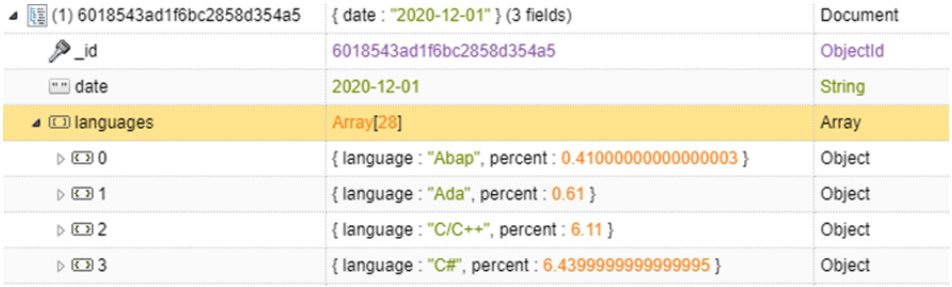
Podemos observar que ahora la fecha nos permite ordenar y realizar más operaciones con esta si fuese necesario.

Cambiar la estructura del dataset

Si no hacía este cambio en la estructura me hubiese costado poder hacer agrupaciones, sacar máximos . etc... Puesto que las propias claves eran los nombres de los lenguajes. He usado el siguiente Código.

```
app.get('/change_data_structure', async function(req, res) {

  const languages_arr = []
  const cols = await languages.findOne({},{"_id":0, "Date":0})

  for (var key in cols) {
    languages_arr.push(key)
  }

  console.log(languages_arr)
  const rows = await languages.find({},{_id:0}).toArray({})

  db.collection('languages').drop()
  db.createCollection("languages")

  rows.forEach(row => {
    let date = moment(row['Date']).format('YYYY-MM-DD')
    languages_dict = {}
    languages_dict['date'] = date
    languages_dict['languages'] = []
    languages_arr.forEach(lan => {
      languages_dict['languages'].push({'language' : lan, 'percent' : row[lan] })
    })
    db.collection('languages').insertOne(languages_dict)
  })

  res.send('Ok')
})
```

Lo guardo en otra colección para no sobreescribir la original. La estructura es la siguiente:



(1) 6018543ad1f6bc2858d354a5	{ date : "2020-12-01" } (3 fields)	Document
_id	6018543ad1f6bc2858d354a5	ObjectId
date	2020-12-01	String
languages	Array[28]	Array
0	{ language : "Abap", percent : 0.41000000000000003 }	Object
1	{ language : "Ada", percent : 0.61 }	Object
2	{ language : "C/C++", percent : 6.11 }	Object
3	{ language : "C#", percent : 6.4399999999999995 }	Object

2.1 Queries de extracción de información

Porcentaje de uso de cada lenguaje por año

```
db.languages.aggregate([
  { "$unwind": "$languages" },
  { "$group": {
    "_id": {
        year: {'$year': '$date'},
        lan : '$languages.language'
    },
    "total_percent": {"$sum": '$languages.percent'},
    "count": {"$sum": 1},
  }},
  { "$sort": { "year": 1 } },
  { "$addFields": {
      "mean_percent": {"$divide": ["$total_percent","$count"] },
    } },
    { $project : { year : "$_id.year" ,mean_percent : 1 ,total_percent: 1, lan : "$_id.lan" ,"_id":0} }
])
```

	total_percent	mean_percent	year	lan
1	208,22	29,7457	2004	Java
2	0	0	2004	Swift
3	2,64	0,3771	2004	Abap
4	67,8	9,6857	2004	C/C++
5	1,38	0,1971	2004	Objective-C
6	0,24	0,0343	2004	Scala
7	0	0	2004	Kotlin
8	58,14	8,3057	2004	Visual Basic
9	0	0	2004	Dart
10	49,8	7,1143	2004	Perl
11	0,79	0,1129	2004	Rust
12	2,82	0,4029	2004	R

Esta consulta me parece interesante para futuras consultas por lo que me lo voy a guardar en una vista auxiliar. Se hace agregando esta línea después de terminar la agregación.

```
saveAsView("lan_by_year",{dropIfExists:true})
```

Lenguajes de programación de la lista

```
db.lan_by_year.aggregate([
    {
        "$group":{
            _id: {
                lan:'$lan'
            },
        }
    },
     { $project : { _id:0, language : "$_id.lan"} }
])
```

	language
1	Perl
2	Delphi
3	Python
4	Lua
5	Scala
6	C/C++
7	Groovy
8	Matlab
9	Swift
10	Ruby
11	Visual Basic
12	JavaScript
13	Objective-C
14	R
15	TypeScript
16	Java
17	Ada
18	Abap
19	Rust
20	Julia
21	Dart
22	Haskell
23	Kotlin
24	PHP
25	Go
26	Cobol
27	C#
28	VBA

Porcentajes de 2020 ordenados de mayor a menor

Hago uso de la vista lan_by_year guardada anteriormente

```
db.lan_by_year.find({year:2020}).sort({total_percent:-1})
```

mean_percent	year	lan
30,83	2020	Python
17,4109	2020	Java
8,3391	2020	JavaScript
6,7964	2020	C#
5,9955	2020	PHP
5,7918	2020	C/C++
4,0009	2020	R
2,8045	2020	Objective-C
2,36	2020	Swift
1,7955	2020	Matlab
1,7409	2020	TypeScript
1,6436	2020	Kotlin
1,4	2020	Go
1,2373	2020	Ruby
1,2045	2020	VBA
0,9936	2020	Scala
0,9436	2020	Rust
0,8036	2020	Visual Basic
0,4864	2020	Dart
0,4655	2020	Perl
0,4609	2020	Lua
0,4464	2020	Abap
0,4427	2020	Ada
0,4245	2020	Groovy
0,3382	2020	Cobol
0,3182	2020	Julia
0,2773	2020	Haskell
0,2473	2020	Delphi

Porcentajes de uso de Java de todos los años

```
db.lan_by_year.find({lan:'Java'}).sort({year:-1})
```

mean_percent	year	lan
17,4109	2020	Java
19,8017	2019	Java
21,7025	2018	Java
22,1167	2017	Java
24,0233	2016	Java
25,4033	2015	Java
26,11	2014	Java
26,4975	2013	Java
27,2625	2012	Java
28,3092	2011	Java
28,4792	2010	Java
28,0567	2009	Java
29,7225	2008	Java
30,6108	2007	Java

Lenguajes de programación que más se ha usado en cada año

```
db.lan_by_year.aggregate([
    {
        "$group":{
            _id: {
                year:'$year'
            },

            maxused: { $max: "$total_percent" }
        }
    },
    {
        $lookup:
         {
            from: "lan_by_year",
            localField: "maxused",
            foreignField: "total_percent",
            as: "enrollee_info"
        }
    },
    {$project: {_id: 0, year :"$_id.year", language: '$enrollee_info.lan'}
])
```

	year	language
1	2004	["Java"]
2	2005	["Java"]
3	2006	["Java"]
4	2007	["Java"]
5	2008	["Java"]
6	2009	["Java"]
7	2010	["Java"]
8	2011	["Java"]
9	2012	["Java"]
10	2013	["Java"]
11	2014	["Java"]
12	2015	["Java"]
13	2016	["Java"]
14	2017	["Java"]
15	2018	["Python"]
16	2019	["Python"]
17	2020	["Python"]

El campo es un array porque podrían coincidir más de un lenguaje, aunque sería raro.

Máximo porcentaje de uso de cada lenguaje

```
db.lan_by_year.aggregate([
    {
        "$group":{
            _id: {
                lan:'$lan'
            },

            maxused: { $max: "$mean_percent" }
        }
    },

    {$project: {_id: 0, lan :"$_id.lan", max_percent: '$maxused'}},
    {$sort: {max_percent:-1}}
])
```

	lan	max_percent
1	Python	30,83
2	Java	30,6108
3	PHP	20,1567
4	C/C++	12,595
5	C#	9,7767
6	JavaScript	8,5757
7	Visual Basic	8,3057
8	Perl	7,1143
9	Objective-C	6,4317
10	R	4,0708
11	Swift	3,4558
12	Matlab	3,3383
13	Ruby	2,8483
14	Delphi	2,7271
15	VBA	2,4358

Mejor año de uso para cada lenguaje

Para esto he utilizado pipelines, son relativamente nuevas y te permiten hacer un join de dos campos.

```
db.lan_by_year.aggregate([
    {
        "$group":{
            _id: {
                lan:'$lan'
            },

            maxused: { $max: "$total_percent" }
        }
    },
    {
        $lookup:
        {
            from: "lan_by_year",
            let: {total_percent: "$total_percent", lan: "$lan", year :"$year"},
              pipeline: [
              { $match:
                 { $expr:
                    { $and:
                       [
                         { $eq: [ "$_id.lan",  "$$lan" ] },
                          { $eq: [ "$maxused",  "$$total_percent" ] }
                       ]
                    }
                 }
              },
            ],
            as: "enrollee_info"
        }
    },

])
```

Lenguajes de programación con mejor tendencia desde 2005

```
db.lan_by_year.aggregate([
    {$match: {$or: [{year:2005}, {year:2020}]}},
    { "$sort": { "year": -1 } },
    {
        "$group":{
            _id: {
                lan:'$lan'
            },

            percents: { $push: "$mean_percent" }
        }
    },
    {$project: {_id: 0, lan :"$_id.lan", increase_percent: {
        $subtract: [
            {$arrayElemAt: [ "$percents",0 ]},
            {$arrayElemAt: [ "$percents",1 ]}
        ]
    }
    { "$sort": { "increase_percent": -1 } },
])
```

	lan	increase_percent
1	Python	27,6092
2	R	3,5517
3	Objective-C	2,6737
4	Swift	2,36
5	TypeScript	1,7409
6	Kotlin	1,6436
7	Go	1,4
8	Scala	0,977
9	C#	0,858
10	Rust	0,8078
11	Dart	0,4864
12	Ruby	0,4798
13	Groovy	0,3437
14	Julia	0,3182
15	Lua	0,2026
16	Ada	0,1136
17	Abap	0,0872
18	JavaScript	0,0508
19	Haskell	0,0389
20	Cobol	-0,0927
21	VBA	-0,3513
22	Matlab	-0,4545
23	Delphi	-2,2319
24	C/C++	-3,4123
25	Perl	-6,1604
26	Visual Basic	-6,5547
27	Java	-12,8116
28	PHP	-13,682

3. Conclusiones

He terminado ya con la parte del servidor. Me faltaría conectarlo con una vista que la tengo a medias todavía. Mongo permite mejorar el rendimiento a la hora de obtener datos, y en un futuro me gustaría montar una estructura con replicasets. Me parece algo menos intuitivo que sql pero seguramente sea porque lo llevo utilizando más tiempo.