Aplicación big data sobre el efecto del coronavirus en el mundo

Bases de Datos Avanzadas
Escuela Superior Informática (UCLM)
Mario Pérez Sánchez-Montañez, David Camuñas Sánchez, Francesco Zingariello
7 de mayo de 2020

Resumen

Este proyecto tratará sobre la creación de una aplicación *big data*, para la cual se realizará un estudio sobre el efecto del coronavirus (**COVID-19**) en el mundo. Para ello se usará las diversas técnicas utilizadas para el tratamiento masivo de información

Keywords: Big Data, *Data Warehouse*, NBA

Introducción

El **Big Data** (*macro datos, datos masivos, inteligencia de datos*), se conoce como el análisis masivo de datos. Esto es una cantidad de datos tan sumamente grande, que las aplicaciones que se dedicaban al procesamiento de datos que tradicionalmente se venían usando no son capaces de tratar y poner en valor en un tiempo razonable. Este término ha estado en uso desde la década de *1990*.

Este término también hace referencia, a las nuevas tecnologías que hacen posible el almacenamiento y procesamiento de datos, además del uso que se hace de la información obtenida a través de ellas.

Tipos de datos que existen

A continuación, se dará una breve explicación de los tipos de datos existentes, y de los cuales hace uso esta tecnología (***Big Data***).

Datos estructurados

Este tipo de datos se suelen usar en el tratamiento de datos. Ya que sus principales características son que pueden ser almacenados en tablas, y que

tienen definido su formato.
Algunos datos de este tipo son:

- **Números.**
- **Cadenas de caracteres.**
- **Fechas.**

Datos semiestructurados

Los ***datos semiestructurados*** tienen una tipo de estructura, pero esta no es lo suficientemente regular, como para permitir su gestión y estructuración como si fuese similar a los datos estructurados. Este tipo de datos, posee ciertos patrones comunes que lo describen y dan información sobre las relaciones entre los mismos.

Un ejemplo de utilización, sería el lenguaje de marcas *HTML*, el cual sirve para el desarrollo de paginas web, donde estas pautas son representadas por su sistema de etiquetas.

Datos no estructurados

Este tipo de datos, se trata de datos que han sido obtenidos en su formato original. Estos no se encuentran especificados en ningún formato, el cual permita que sean almacenados como se almacenarían los anteriores tipos de datos (*estructurados* y *semiestructurados*). Ya que no disponen de una estructura definida (longitud, formato, etc) para poder desglosar la información.

Estos datos se pueden encontrar, por ejemplo: en presentaciones (*PowerPoints*), correos, documentos de texto *(Word, PDF, documentos de google*, etc).

Obtención de los datos

Los datos utilizados se han obtenido de ***EU Open Data Portal*** [2]. El conjunto de datos contiene los últimos datos públicos disponibles sobre COVID-19, incluida una actualización diaria de la situación, la curva epidemiológica y la distribución geográfica global (UE / EEE y el Reino Unido, en todo el mundo).

Para asegurar la precisión y confiabilidad de los datos, este proceso se refina constantemente. Esto ayuda a controlar e interpretar la dinámica de la pandemia de COVID-19 no solo en la Unión Europea (UE), el Espacio Económico Europeo (EEE), sino también en todo el mundo. Todos los días, entre las 6.00 y las 10.00 CET, un equipo de epidemiólogos examina hasta 500 fuentes relevantes para recopilar las últimas cifras. El análisis de datos es seguido por el proceso estándar de inteligencia epidémica del ECDC para el cual cada entrada de datos se valida y documenta en una base de datos del ECDC [4].

Creación de la base de datos

La base de datos se ha construido utilizando *MongoDB* [5] y *Azure Cosmos DB* [1]. De esta forma la base de datos es accesible desde cualquier parte.

MongoDB es una base de datos documental, el elemento esencial es el documento que normalmente se los agrupa en colecciones de documentos similares. Una base de datos en MongoDB es un conjunto de colecciones.

MongoDB almacena datos en documentos flexibles, similares a JSON, lo que significa que los campos pueden variar de un documento a otro y la estructura de datos se puede cambiar con el tiempo. El modelo de documento se asigna a los objetos en el código de su aplicación, lo que facilita el trabajo con los datos.

En nuestro caso, la base datos estará formada por una sola colección y el número de documentos varía con el tiempo, pero actualmente contiene más de 15000. La estructura de un documento se puede observar en la figura 1

Fields	Global Probability	Type
[covid.Worldwide]	100.0%	Collection
_id	100.0%	ObjectId
cases	100.0%	Int32
continentExp	100.0%	String
countriesAndTerritories	100.0%	String
countryterritoryCode	100.0%	String
dateRep	100.0%	Date
day	100.0%	Int32
deaths	100.0%	Int32
geoId	100.0%	String
month	100.0%	Int32
popData2018	100.0%	Double
year	100.0%	Int32

Figura 1: Estructura de un documento [3]

La distribución de tipos de datos se puede ver en la Figura 2. Siendo el tipo numérico el predominante con cinco elementos del documento de este tipo, seguido del tipo cadena con cuatro elementos y el formato fecha con un elemento.

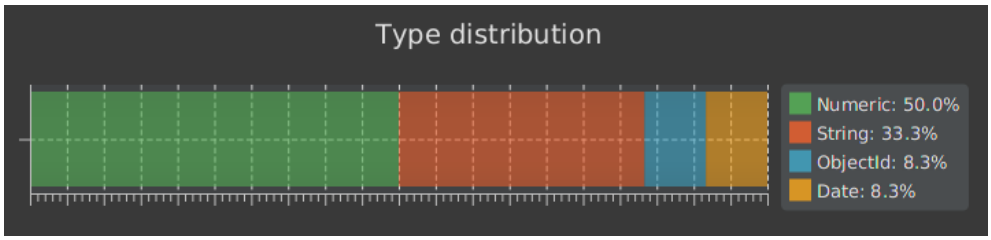

Figura 2: Distribución de tipos de datos [3]

Almacenamiento de datos

Para automatizar el proceso de almacenamiento de datos hemos creado un *script* en *Python* [6], que descarga los datos de la fuente mencionada anteriormente en formato *xlsx* y actualiza los datos en la base de datos, indicando simplemente el identificador (uri) proporcionado por *Azure Cosmos DB*.

```
1  import pandas as pd
2  import wget
3  import os
4  from pymongo import MongoClient
5
6  url = 'https://www.ecdc.europa.eu/sites/default/files' \
7  '/documents/COVID-19-geographic-disbtribution-worldwide.xlsx'
8
9
10 def excel_to_json():
11   filename = wget.download(url)
12   data = pd.read_excel(filename)
13   os.remove(filename)
14   return data.to_dict('records')
15
16
17 def upload_data(uri):
18   client = MongoClient(uri)
19   database = client['covid']
20   collection = database['Worldwide']
21   collection.drop()
22   collection.insert_many(excel_to_json())
23
24
25 upload_data(os.sys.argv[1])
```

De esta forma, cada día ejecutamos el *script* y actualizamos los datos de la base de datos en cuestión de minutos, ya que como hemos comentado anteriormente los datos de la fuente se actualizan a diario.

Ejecución de consultas

La ejecución de consultas la hemos realizado gracias a la creación de una clase *Python*, con la cual, crearemos objetos que nos permitirán conectar fácilmente con la base de datos y obtener datos específicos por país, fecha o continente. Esta clase se comporta de forma similar a un agente, utilizado en otras asignaturas para realizar la conexión con la base de datos.

```
1  from pymongo import MongoClient
2
3
4  def parse_date(date):
5    date = date.split("/")
6    day = int(date[0])
7    month = int(date[1])
8    year = int(date[2])
9
10   return day, month, year
11
12
13 class CovidClient:
14   def __init__(self, dir_bd, name_bd, collection):
15     self.__collection = MongoClient(dir_bd)[name_bd][collection]
16
17   def get_total_deaths_country(self, country):
18     try:
19       data = self.__collection.find({"countriesAndTerritories": country}, {"deaths"})
20       deaths = 0
21
22       for x in data:
23         deaths += int(x['deaths'])
24
25       return deaths
26     except Exception as e:
27       return e
28
29   def get_total_deaths_continent(self, continent):
30     try:
31       data = self.__collection.find({"continentExp": continent}, {"deaths"})
32       deaths = 0
33
34       for x in data:
35         deaths += int(x['deaths'])
36
37       return deaths
38     except Exception as e:
39       return e
40
41   def get_total_cases_country(self, country):
42     try:
43       data = self.__collection.find({"countriesAndTerritories": country}, {"cases"})
44       cases = 0
45
46       for x in data:
47       cases += int(x['cases'])
48
49       return cases
50     except Exception as e:
51       return e
```

```
52
53  def get_total_cases_continent(self, continent):
54      try:
55          data = self.__collection.find({"continentExp": continent}, {"cases"})
56          cases = 0
57
58          for x in data:
59              cases += int(x['cases'])
60
61          return cases
62      except Exception as e:
63          return e
64
65  def get_worst_day_country(self, country):
66      try:
67          data = self.__collection.find({"countriesAndTerritories": country}, {"deaths", "
        dateRep"})
68          deaths = None
69          date = None
70
71          for x in data:
72              if deaths is None:
73                  deaths = int(x['deaths'])
74                  date = x['dateRep']
75              else:
76                  if int(x['deaths']) > deaths:
77                      deaths = int(x['deaths'])
78                      date = x['dateRep']
79
80          return date, deaths
81      except Exception as e:
82          return e
83
84  def get_total_deaths(self):
85      try:
86          data = self.__collection.find({}, {"deaths"})
87          deaths = 0
88
89          for x in data:
90              deaths += int(x['deaths'])
91
92          return deaths
93      except Exception as e:
94          return e
95
96  def get_total_cases(self):
97      try:
98          data = self.__collection.find({}, {"cases"})
99          cases = 0
100
101         for x in data:
102             cases += int(x['cases'])
103
104         return cases
105     except Exception as e:
106         return e
107
108 def get_data_date_country(self, country, date):
109     try:
```

```
110         day, month, year = parse_date(date)
111         data = self.__collection.find(
112         {
113           "countriesAndTerritories": country,
114           "day": day,
115           "month": month,
116           "year": year
117         },
118         {"countriesAndTerritories", "cases", "deaths"}
119         )
120 
121         return data
122       except Exception as e:
123         return e
124 
125     def get_data_country(self, country):
126       try:
127         # only days with cases or deaths
128         data = self.__collection.find({"countriesAndTerritories": country,"$or": [{"cases": {
          "$gt": 0}},{ "deaths": {"$gt": 0}}]},{ "countriesAndTerritories", "cases", "deaths", "
          dateRep"})
129 
130         return data
131       except Exception as e:
132         return e
```

Gracias a esta clase podremos realizar multitud de consultas interesantes a la base de datos, algunas de ellas las mostraremos a continuación.

0.1. Ejemplos de consultas

Referencias

1. https://azure.microsoft.com/es-es/services/cosmos-db/. Azure cosmos db.
2. https://data.europa.eu/euodp/en/data/dataset/covid-19-coronavirus-data. Eu open data portal.
3. https://studio3t.com/. Studio 3t.
4. https://www.ecdc.europa.eu/en/covid-19-pandemic. European centre for disease prevention and control.
5. https://www.mongodb.com/. Mongodb.
6. https://www.python.org/. Python software fundation.