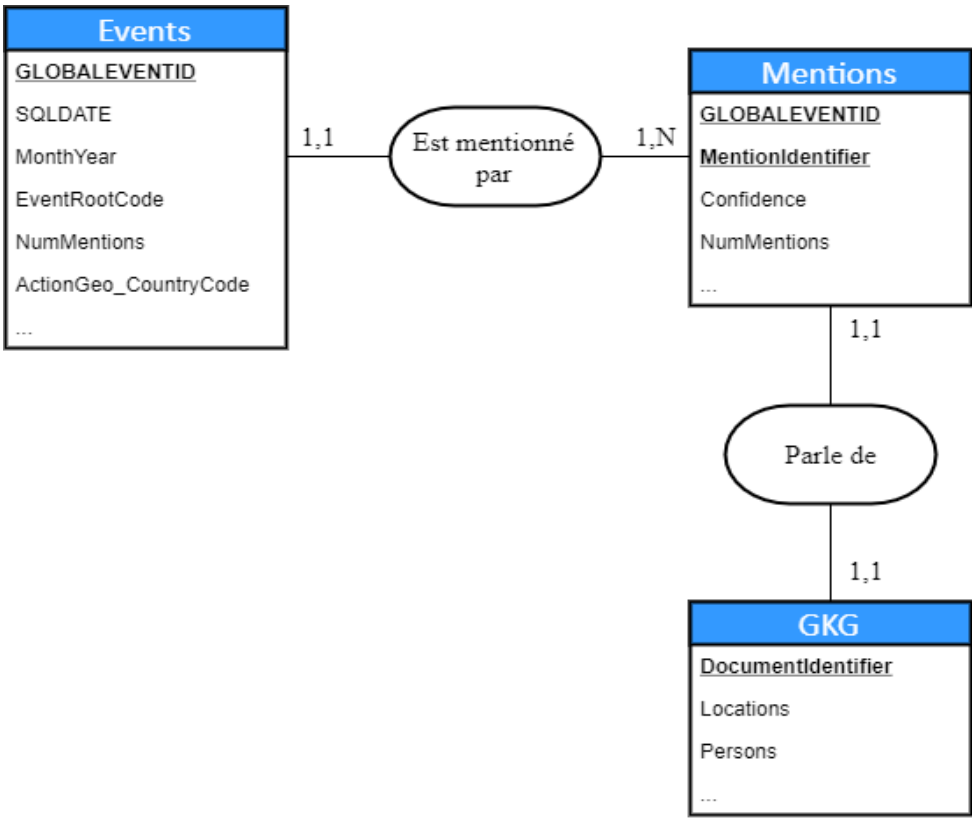
Réalisation : Mansour Hmidan, Sofien Bentaher, Ahmed Ben Amor, Abdelkrim Mahlal , Haythem Chettaoui, Djafar Hamiche, Kamel Boucherba, Aymen Cheikh , Lamine Ouachek

GDELT : l’architecture

GDELT est la base de données ouverte la plus grande, la plus complète et la plus haute résolution de la société humaine jamais créée. Création d'une plate-forme qui surveille les médias d'information du monde entier dans presque tous les coins de chaque pays dans des formats imprimés, diffusés et Web, dans plus de 100 langues, à chaque instant de chaque jour et qui s'étend du 1er janvier 1979 à nos jours, avec mises à jour, ont nécessité un éventail sans précédent d'innovations techniques et méthodologiques, de partenariats et de tout nouvelles mentalités pour rassembler tout cela et en faire une réalité. La création d'une base de données d'un quart de milliard d'enregistrements géoréférencés couvrant le monde entier sur 30 ans, couplée aux réseaux massifs qui connectent toutes les personnes, organisations, lieux, thèmes et émotions sous-jacents à ces événements, a nécessité non seulement de résoudre des défis sans précédent pour créer la base de données, mais aussi une "réimaginassions" de la façon dont nous interagissons et pensons aux données à l'échelle de la société.

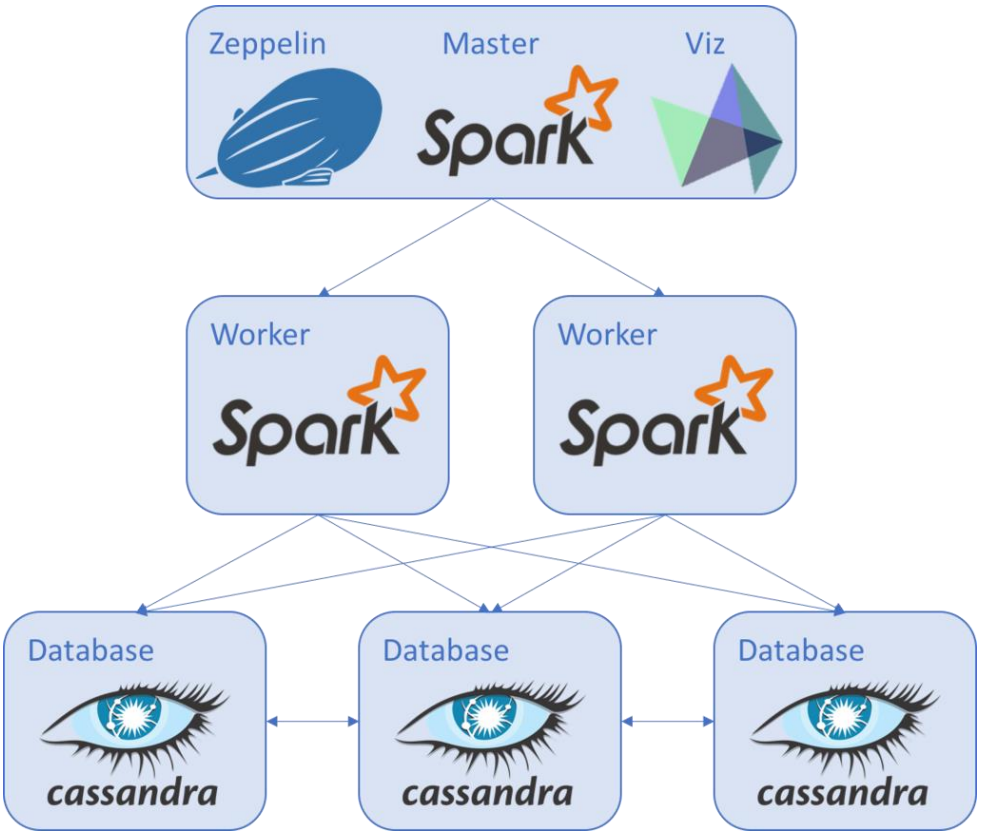
1- Modèle de données



2- Architecture

Pour ce projet, on a opté pour l'architecture suivante:

- Utiliser Spark comme ETL avec son langage natif et le traitement et les jointures intermédiaires via spark SQL
- Déposer les fichier GDELT dans S3 sous format parquet pour accélérer le processus de lecture
- Utiliser zepplin notebook



3- Préparation de donnée

- Importation des packages nécessaires

```
import sys.process._
import java.net.URL
import java.io.File
import java.io.File
import java.nio.file.{Files, StandardCopyOption}
import java.net.HttpURLConnection
import org.apache.spark.sql.functions._
import sqlContext.implicits._
import org.apache.spark.input.PortableDataStream
import java.util.zip.ZipInputStream
import java.io.BufferedReader
import java.io.InputStreamReader
import org.apache.spark.sql.SQLContext
import com.amazonaws.services.s3.AmazonS3Client
import com.amazonaws.auth.BasicAWSCredentials
import org.apache.spark.sql.types.IntegerType
import com.amazonaws.services.s3.AmazonS3Client
```

```
import sys.process._
import java.net.URL
import java.io.File
import java.io.File
import java.nio.file.{Files, StandardCopyOption}
import java.net.HttpURLConnection
import org.apache.spark.sql.functions._
import sqlContext.implicits._
import org.apache.spark.input.PortableDataStream
import java.util.zip.ZipInputStream
import java.io.BufferedReader
import java.io.InputStreamReader
import org.apache.spark.sql.SQLContext
import com.amazonaws.services.s3.AmazonS3Client
import com.amazonaws.auth.BasicAWSCredentials
import org.apache.spark.sql.types.IntegerType
import com.amazonaws.services.s3.AmazonS3Client
```

Took 5 sec. Last updated by anonymous at March 07 2020, 2:12:37 PM.

- Telechargement des ficher masterfilelist.txt et masterfile_translation.txt

```
%pyspark
import urllib

print('Beginning file download with urllib2...')

url = 'http://data.gdeltproject.org/gdeltv2/masterfilelist.txt'
urllib.urlretrieve(url, '/tmp/masterfilelist.txt')
```

```
Beginning file download with urllib2...
('/tmp/masterfilelist.txt', <httplib.HTTPMessage instance at 0x7fb131b7cbd8>)
```

Took 2 sec. Last updated by anonymous at March 07 2020, 11:16:52 AM.

```
%pyspark
import urllib

print('Beginning file download with urllib2...')

url = 'http://data.gdeltproject.org/gdeltv2/masterfilelist-translation.txt", "/tmp/masterfilelist_translation.txt"'
urllib.urlretrieve(url, '/tmp/masterfilelist_translation.txt')
```

```
Beginning file download with urllib2...
('/tmp/masterfilelist_translation.txt', <httplib.HTTPMessage instance at 0x7fb131b7c8c0>)
```

Took 1 sec. Last updated by anonymous at March 07 2020, 11:17:01 AM.

- Recuperer le fichier masterfile.txt et le spliter dans un RDD avec 3 colonne qui sont size, hash et url

```
%pyspark
from pyspark.sql import Row
data_file = sc.textFile('s3://gdelt2020/masterfilelist.txt')
textSplit = data_file.map(lambda line : line.split(" "))
dataRDD = textSplit.map(lambda item : Row (size=item[0], hash=item[1], url=item[2]))
print("succes")
```

```
succes
```

Took 0 sec. Last updated by anonymous at March 07 2020, 11:17:12 AM.

```
%pyspark
from pyspark.sql import Row
data_file = sc.textFile('s3://gdelt2020/masterfilelist_translation.txt')
textSplit = data_file.map(lambda line : line.split(" "))
dataRDD_translation = textSplit.map(lambda item : Row (size=item[0], hash=item[1], url=item[2]))
print("succes")
```

```
succes
```

Took 0 sec. Last updated by anonymous at March 07 2020, 11:23:03 AM.

- Convertion les donnée du RDD à Data Frame

```
%pyspark
dataDF = dataRDD.toDF()
```

Took 19 sec. Last updated by anonymous at March 07 2020, 1:00:12 PM.