PROJET GDELT

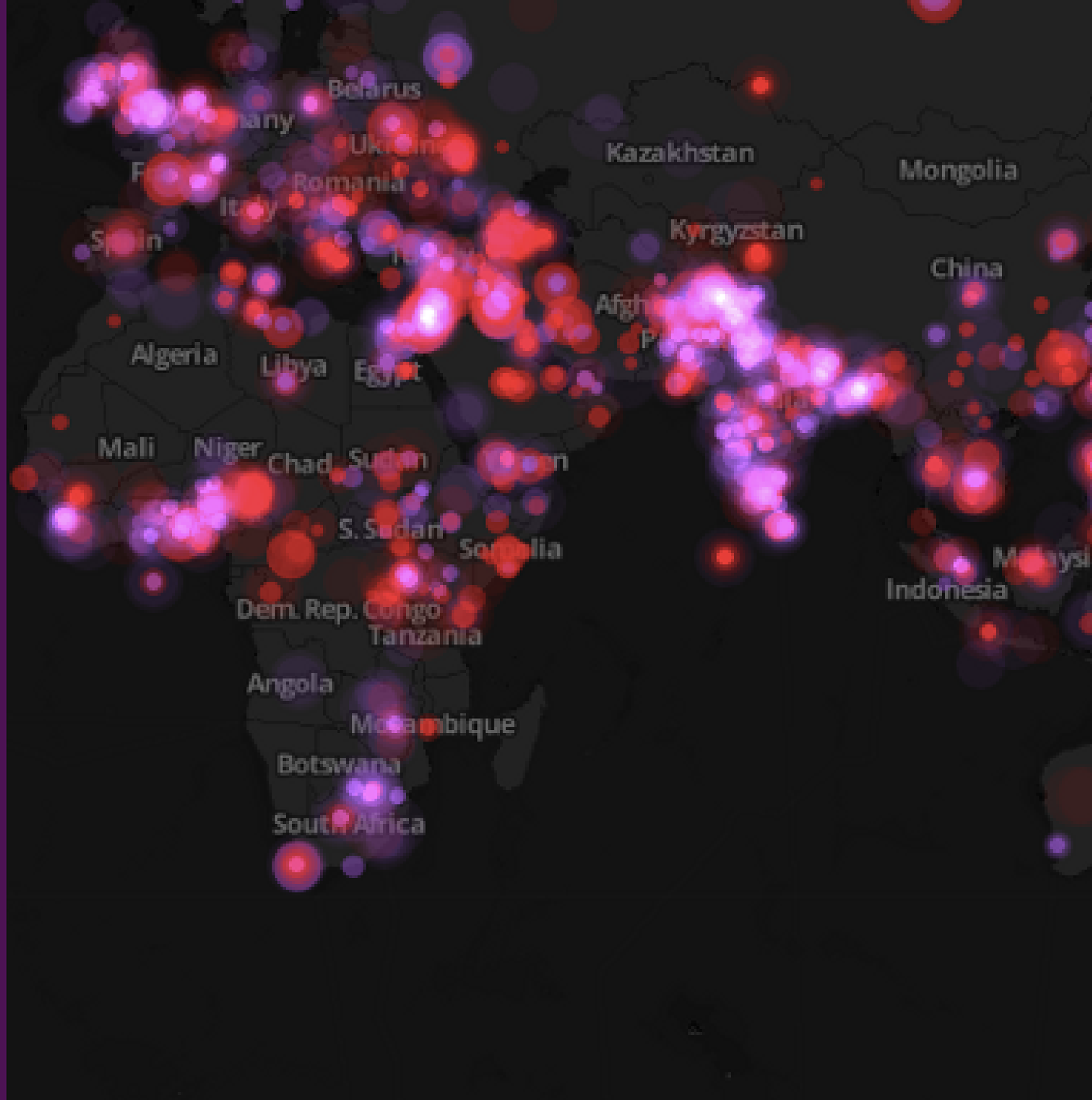
How was 2021 ?

Sommaire

Contexte

Objectif :

- Analyser l'impact media de l'année 2021 via le jeu des données GDELT.
- Proposer un système de stockage distribué, résilient et performant pour les données GDELT.



GDELT Dataset

Events

Ce dataset regroupe les informations des événements en capturant les 2 participants à cet événement.

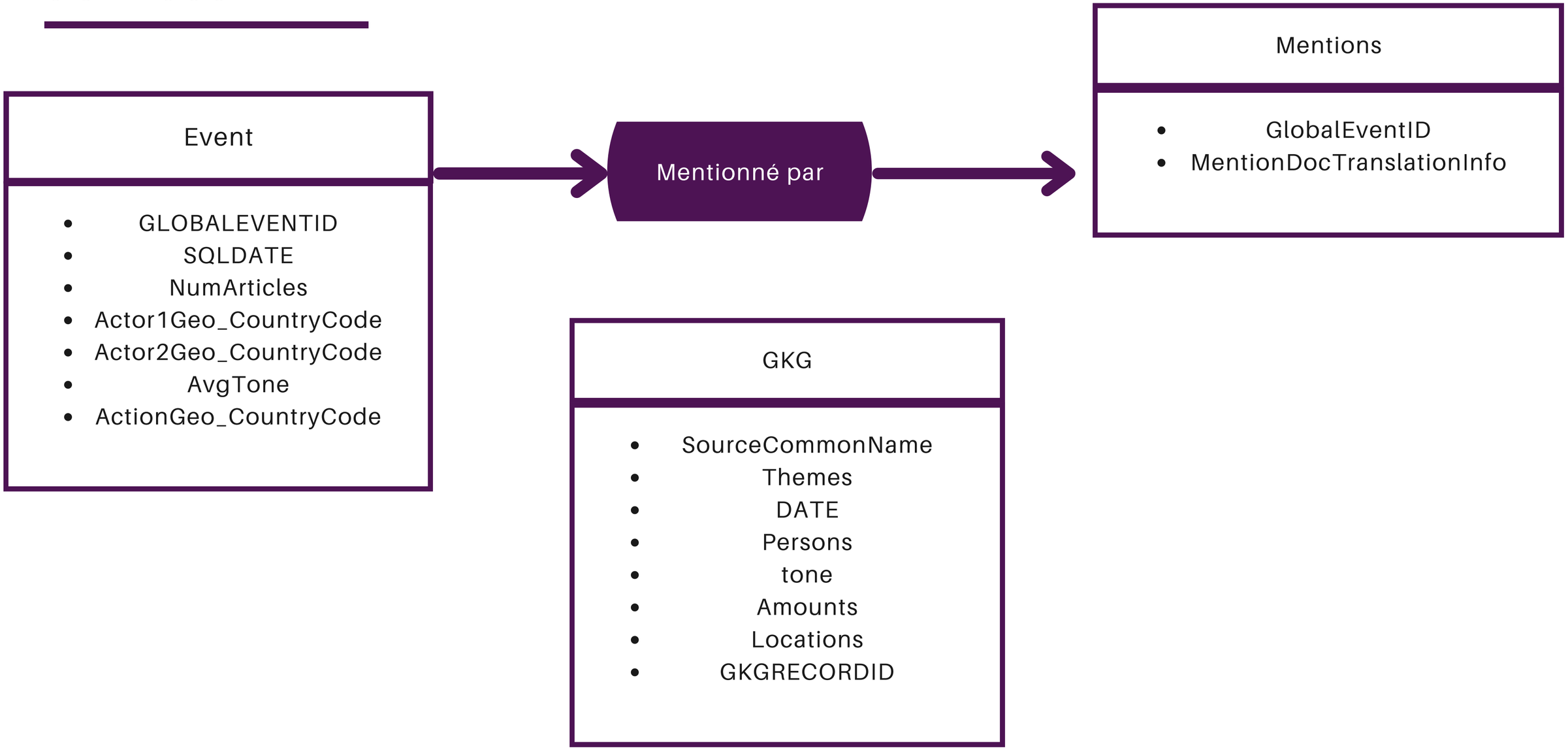
Mentions

Cette table regroupe toutes les mentions pour un évenement .

GKG

Global Knowledge Graph permet de connecter les participants, organisations, lieux et thèmes.

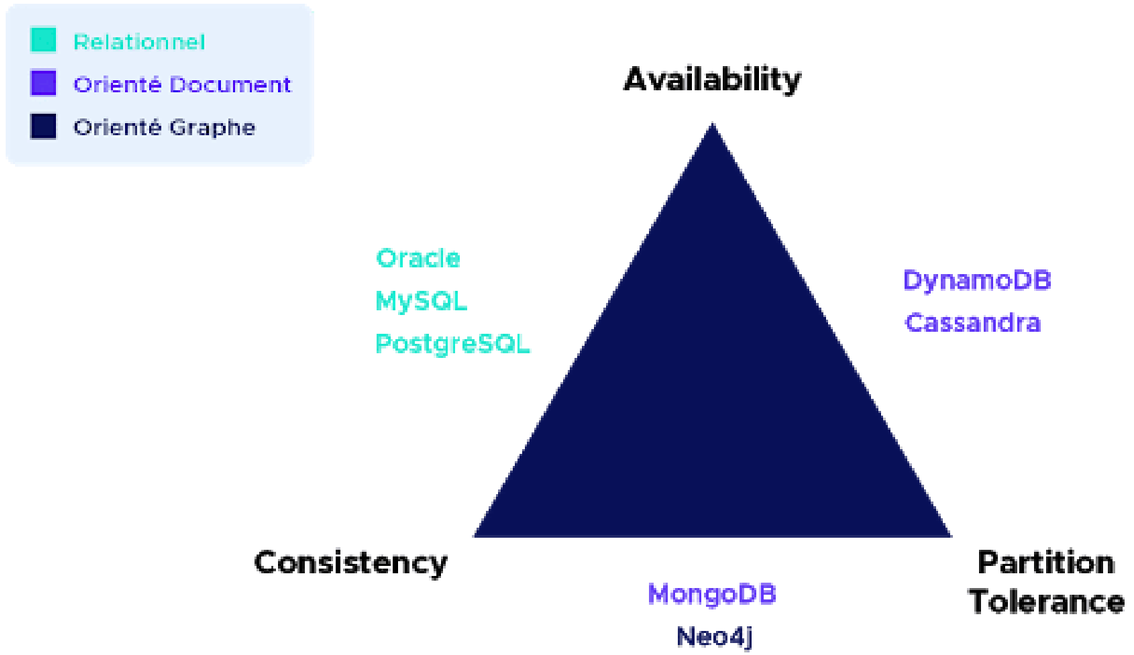
Modèle des données



Choix de la technologie:

MongoDB:

- Fonctionnalité.
- Cohérence
- Tolérance au partitionnement
- Sharding - Multi-indexing.
- Rapidité de requêtage



Gdelt

Extraction des données

Luigi

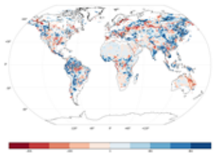
Stockage des données + requêtes

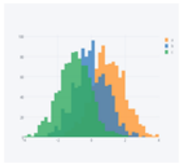
6 machines

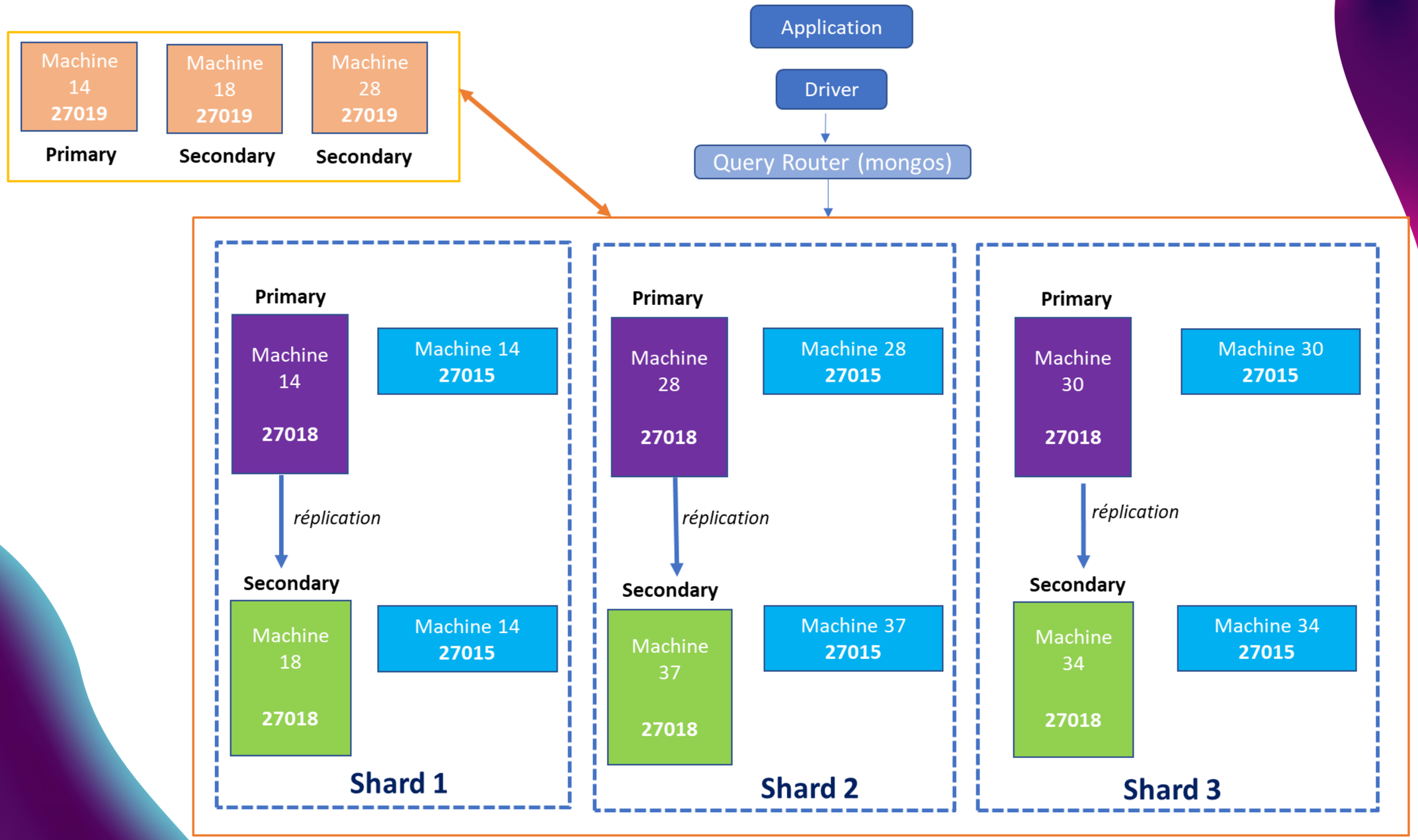
mongoDB

jupyter

Requêtes + résultats + analyses

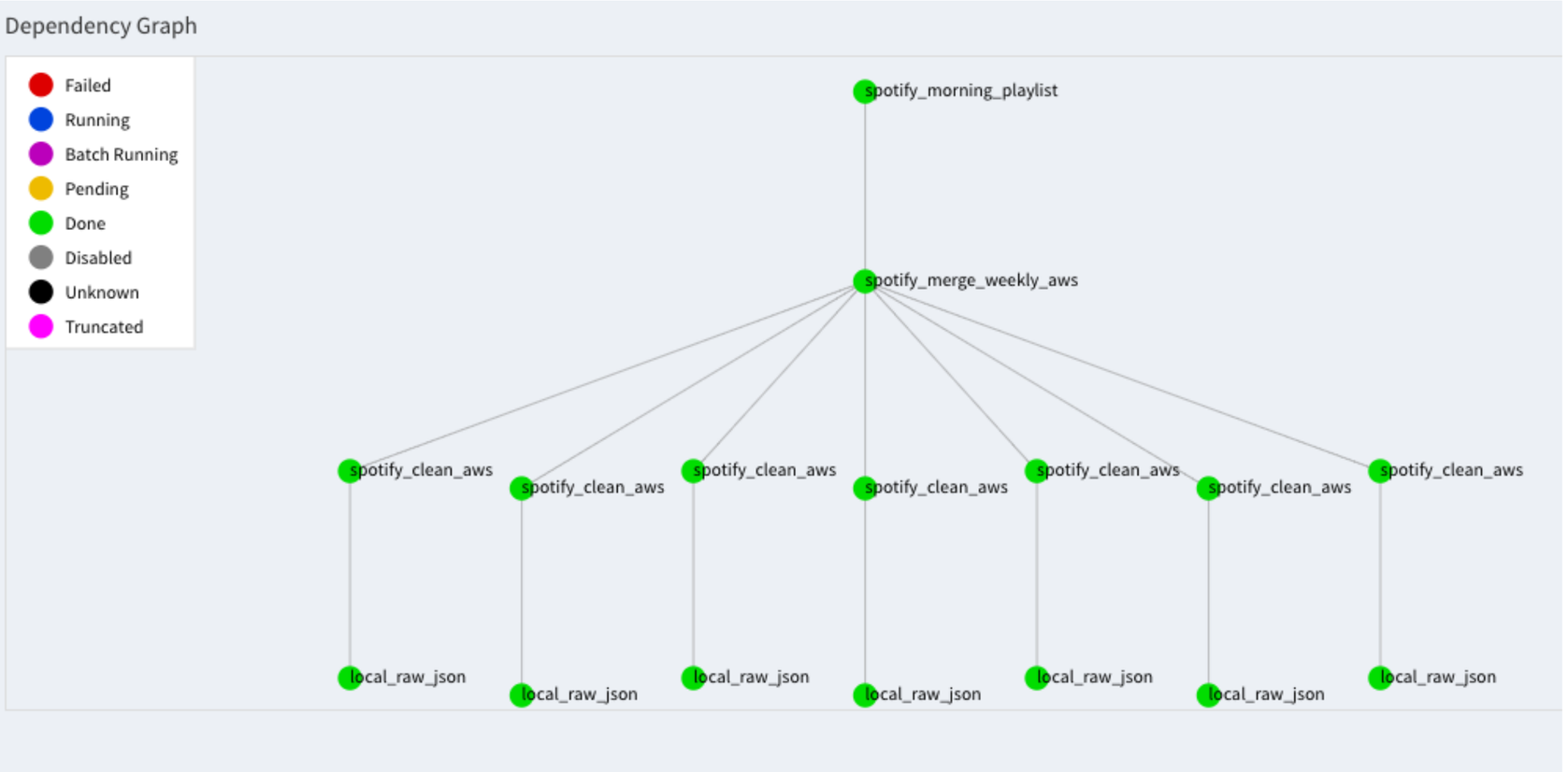


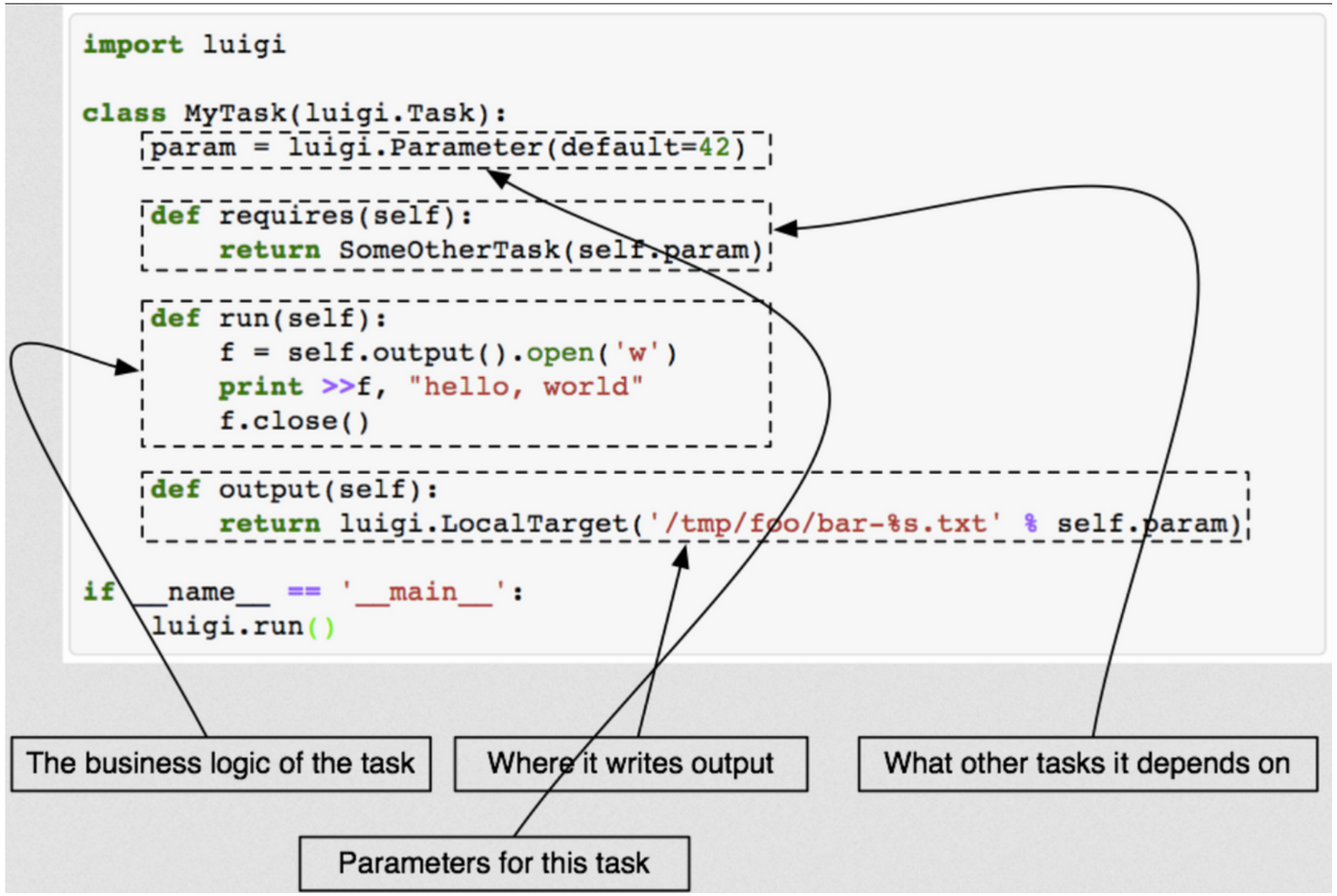


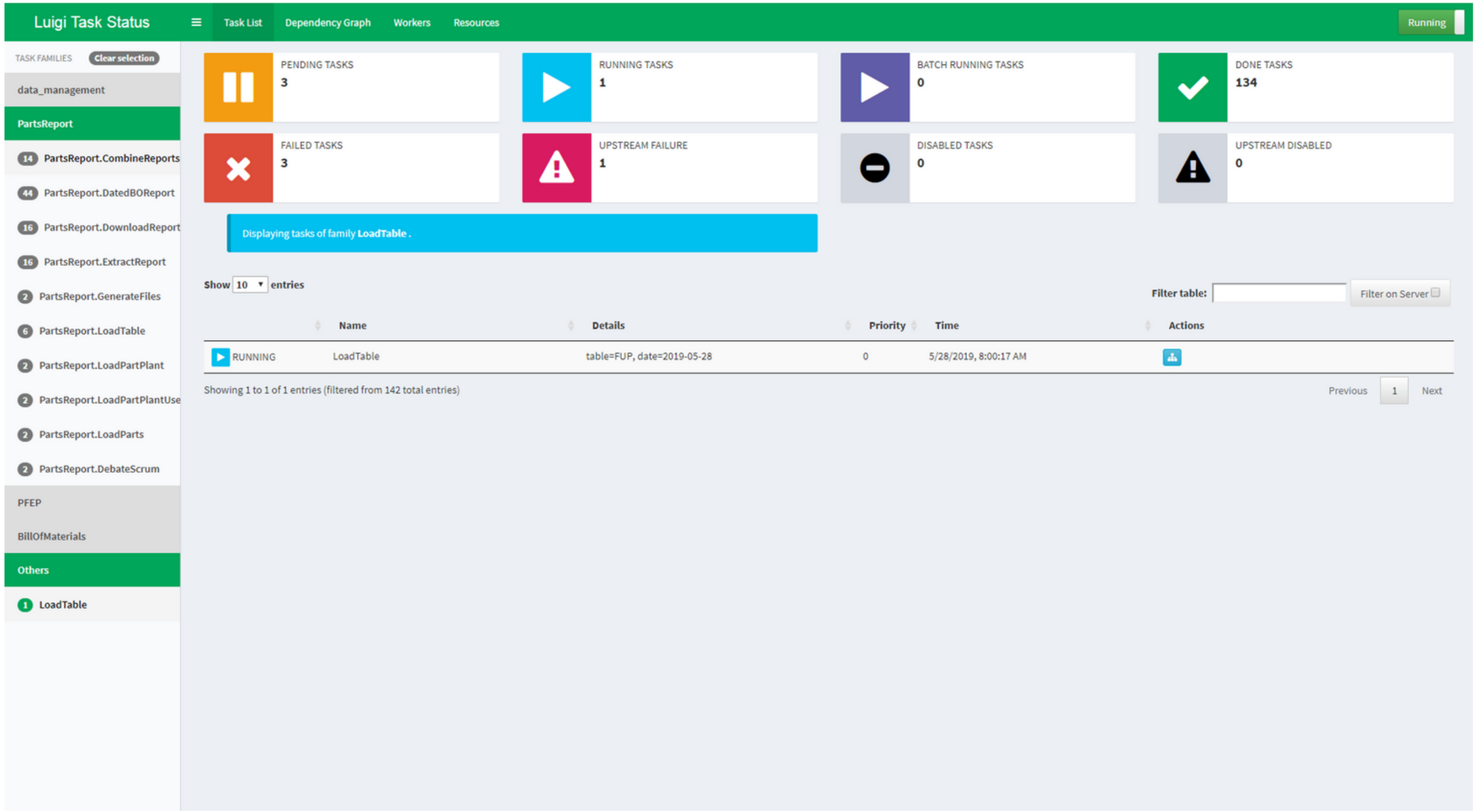

Machine 14 27019
Machine 18 27019
Machine 28 27019
Primary
Secondary
Secondary
Application
Driver
Query Router (mongos)
Primary
Machine 14 27018
Machine 14 27015
réplication
Secondary
Machine 18 27018
Machine 14 27015
Shard 1
Primary
Machine 28 27018
Machine 28 27015
réplication
Secondary
Machine 37 27018
Machine 37 27015
Shard 2
Primary
Machine 30 27018
Machine 30 27015
réplication
Secondary
Machine 34 27018
Machine 34 27015
Shard 3

ETL

Luigi


Dependency Graph
Failed
Running
Batch Running
Pending
Done
Disabled
Unknown
Truncated
spotify_morning_playlist
spotify_merge_weekly_aws
spotify_clean_aws
spotify_clean_aws
spotify_clean_aws
spotify_clean_aws
spotify_clean_aws
spotify_clean_aws
spotify_clean_aws
local_raw_json
local_raw_json
local_raw_json
local_raw_json
local_raw_json
local_raw_json
local_raw_json


import luigi
class MyTask(luigi.Task):
param = luigi.Parameter(default=42)
def requires(self):
return SomeOtherTask(self.param)
def run(self):
f = self.output().open('w')
print >>f, "hello, world"
f.close()
def output(self):
return luigi.LocalTarget('/tmp/foo/bar-%s.txt' % self.param)
if __name__ == '__main__':
luigi.run()
The business logic of the task
Where it writes output
What other tasks it depends on
Parameters for this task


Luigi Task Status
Task List
Dependency Graph
Workers
Resources
Running
TASK FAMILIES
Clear selection
data_management
PartsReport
14 PartsReport.CombineReports
44 PartsReport.DatedBOReport
16 PartsReport.DownloadReport
16 PartsReport.ExtractReport
2 PartsReport.GenerateFiles
6 PartsReport.LoadTable
2 PartsReport.LoadPartPlant
2 PartsReport.LoadPartPlantUse
2 PartsReport.LoadParts
2 PartsReport.DebateScrum
PFEP
BillOfMaterials
Others
1 LoadTable
PENDING TASKS
3
RUNNING TASKS
1
BATCH RUNNING TASKS
0
DONE TASKS
134
FAILED TASKS
3
UPSTREAM FAILURE
1
DISABLED TASKS
0
UPSTREAM DISABLED
0
Displaying tasks of family LoadTable .
Show 10 entries
Filter table:
Filter on Server
Name
Details
Priority
Time
Actions
RUNNING
LoadTable
table=FUP, date=2019-05-28
0
5/28/2019, 8:00:17 AM
Showing 1 to 1 of 1 entries (filtered from 142 total entries)
Previous
1
Next

STACKTRACE (most recent call first)

Full Raw

__init__.py in **handle** at line **894**

```
889.         """
890.         rv = self.filter(record)
891.         if rv:
892.             self.acquire()
893.             try:
894.                 self.emit(record)
895.             finally:
896.                 self.release()
897.         return rv
898.
899.     def setFormatter(self, fmt):
```

record	`<LogRecord: xxx>`
rv	`True`
self	`<EventHandler (WARNING)>`

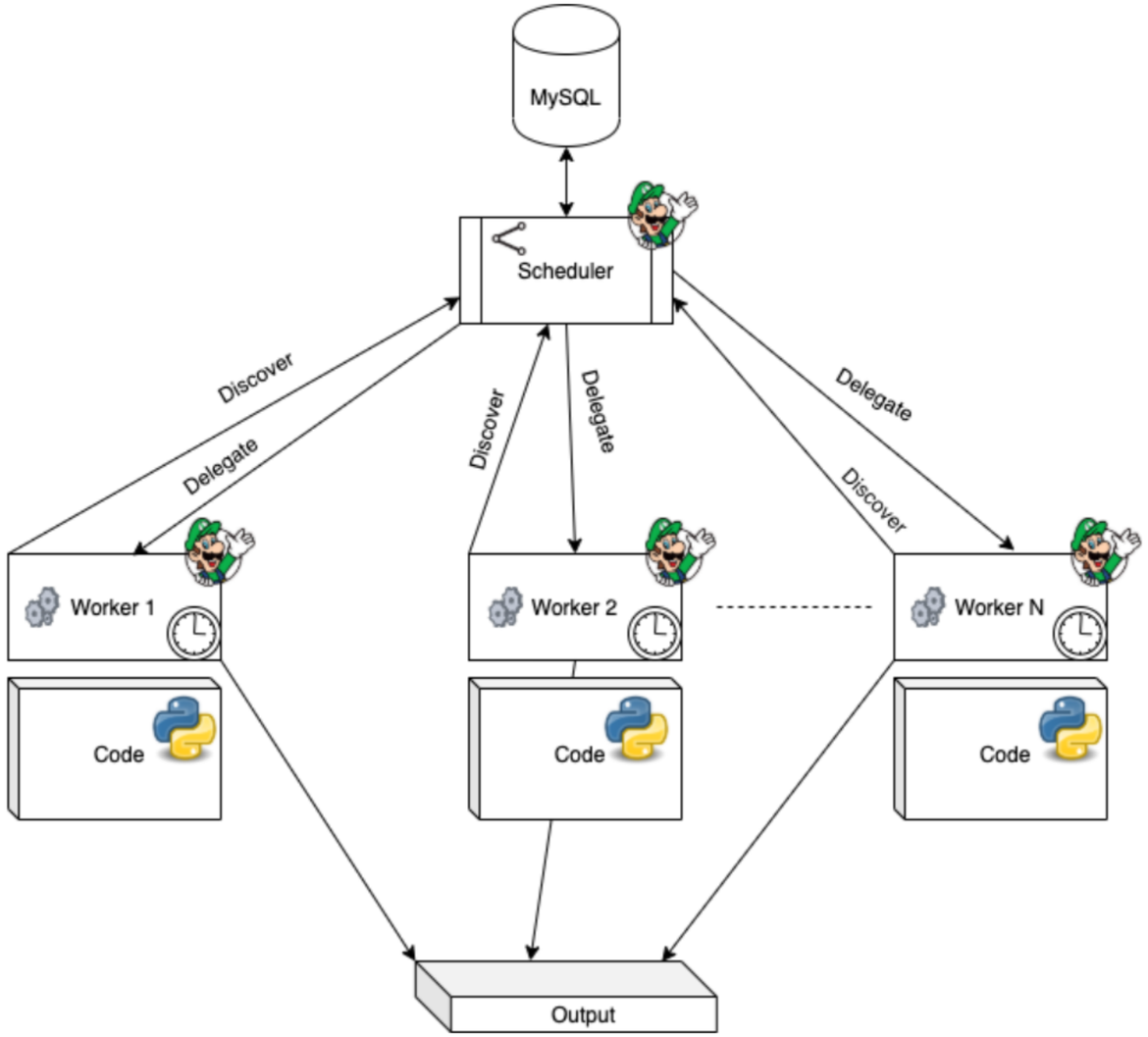
__init__.py in **handle** at line **1524**

__init__.py in **_log** at line **1514**

__init__.py in **warning** at line **1390**

xxx/apiclient.py in **_get** at line **52**

```
47.             'x-api-key': self.api_key,
48.         }
49.
50.     def _get(self, url, **kwargs):
51.         # Log warning
52.         log.warning('Test warning', extra={'test': 'value'})
```

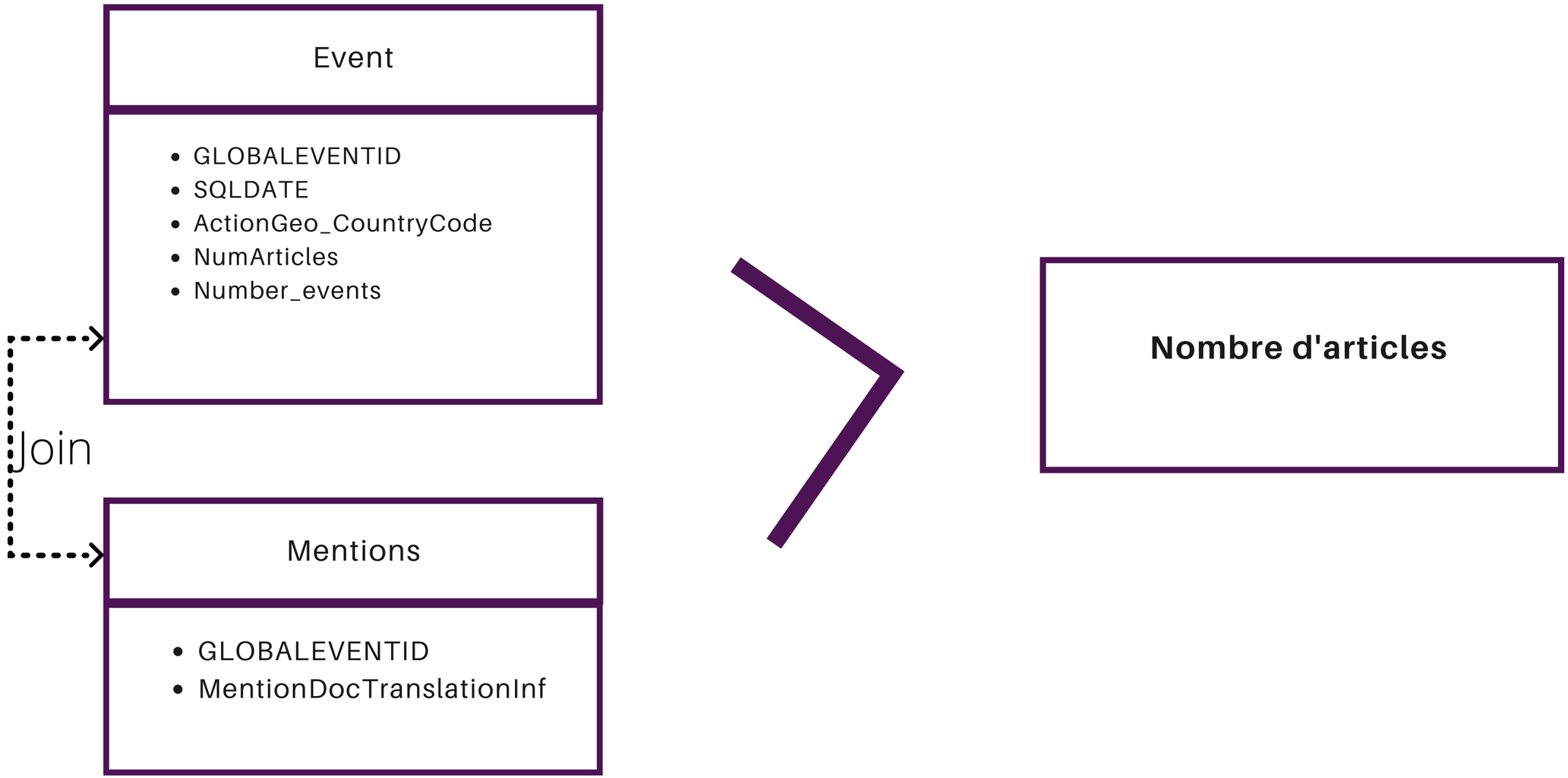


Luigi workers communication with the scheduler


LATENCY
LATENCY EVERYWHERE

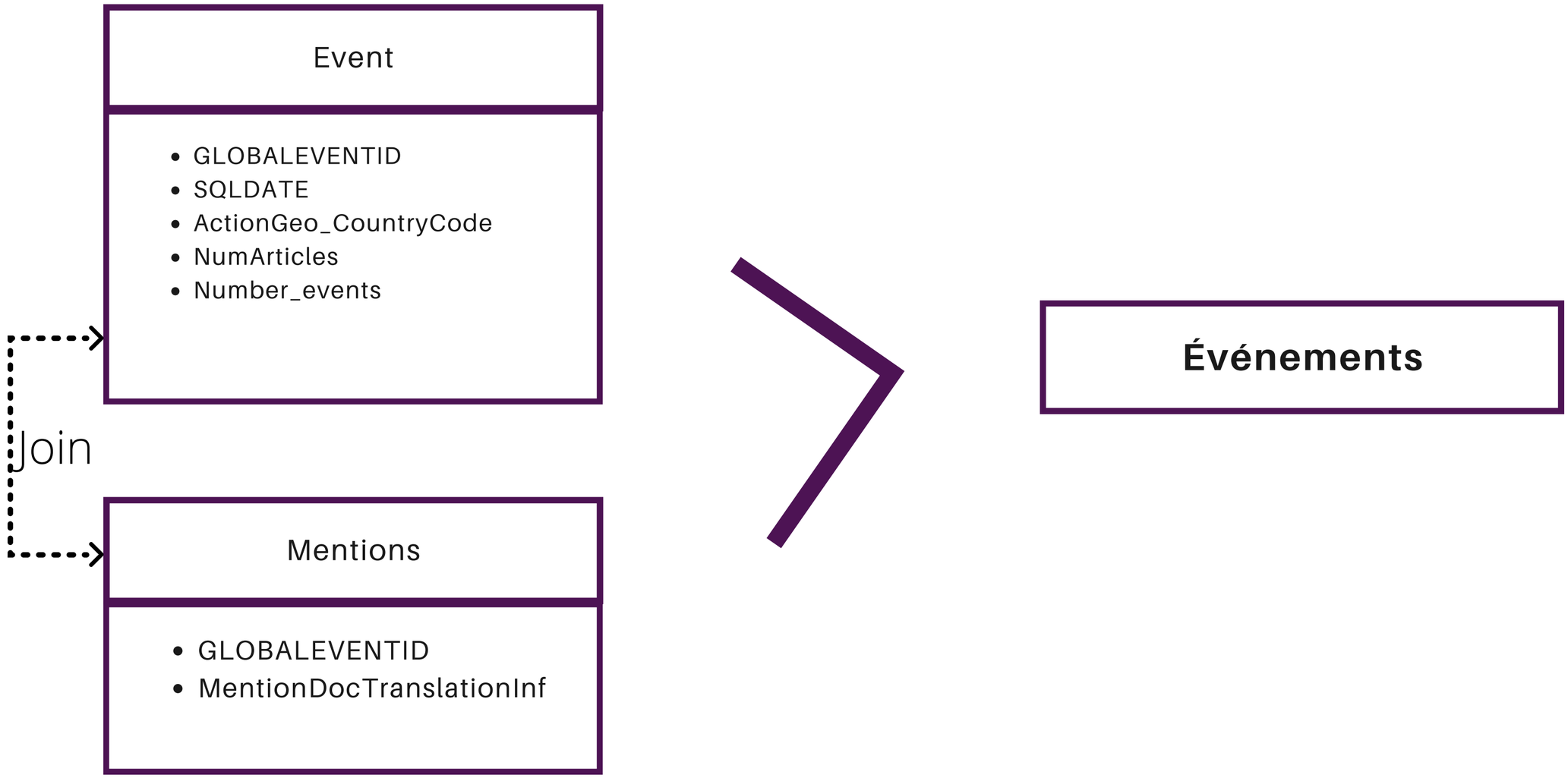
Requête 1 :

Afficher le nombre d’articles/évènements qu’il y a eu pour chaque triplet (jour, pays de l’évènement, langue de l’article)



Requête 2 :

Pour un pays donné en paramètre, affichez les évènements qui y ont eu place triées par le nombre de mentions (tri décroissant); permettez une agrégation par jour/mois/année



Requête 3 :

Pour une source de donnés passée en paramètre (gkg.SourceCommonName) affichez les thèmes, personnes, lieux dont les articles de cette sources parlent ainsi que le nombre d'articles et le ton moyen des articles (pour chaque thème/personne/lieu); permettez une agrégation par jour/mois/année.

GKG

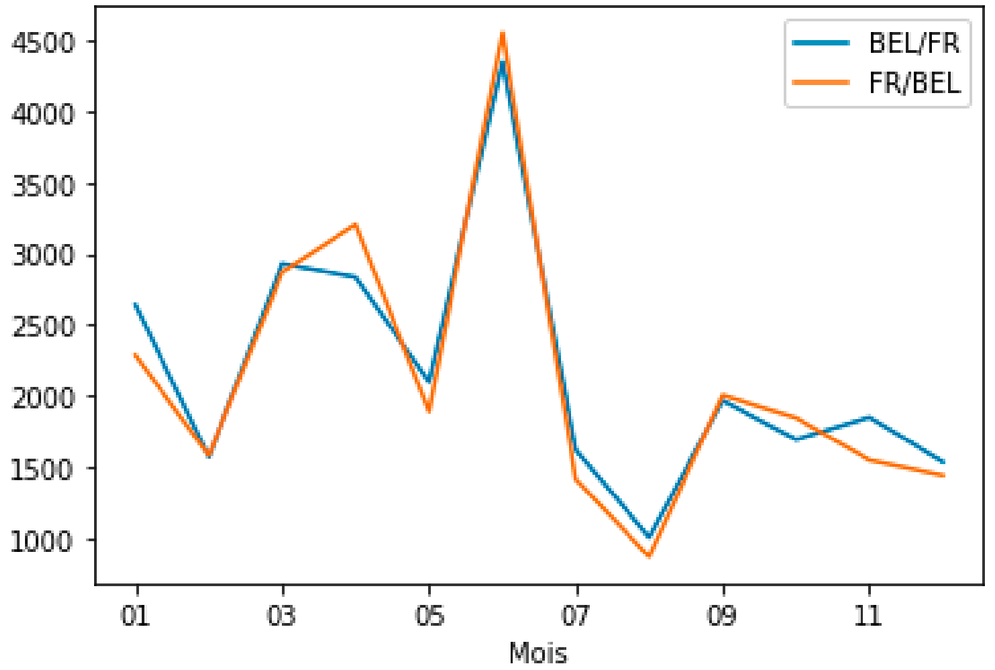
- GKGRECORDID
- SourceCommonName
- DATE
- Themes
- Persons
- Tone
- Amounts
- Locations

Themes
Locations
Persons

Requête 4 :

L’évolution des relations entre deux pays (specifies en paramètre) au cours de l’année. Vous pouvez vous baser sur la langue de l’article, le ton moyen des articles, les themes plus souvent citées, les personalités ou tout element qui vous semble pertinent.

Event
• GLOBALEVENTID • SQLDATE • Actor1CountryCode • Actor2CountryCode • ActionGeo_CountryCode • NumArticles • AvgTone



Demo