Développement d'applications Cloud - Rendu n°4
Optimisation de l'infrastructure de données

Équipe 5 :
Johny Lin
Jérémy Goldschild
Jean-Louis Delebecque
Ariane Huckel

Lien du git: https://github.com/jdelebec/dev_app_cloud

Rappel cas d'usage:

R1	Quels sont les sujets étudiés à Harvard ?
R2	Donner les bureaux de start-up situés en Californie et qui contiennent Headquarters dans leur description.
R3	Trouver toutes les levées de fonds de plus 50 millions, qui sont en dollars et ont eu lieu en 2010 et indiquer le nom de la start-up associée (somme des levées de fonds).
R4	Le nom des individus travaillant chez Facebook dont leur bureau est en Californie.
R5	Dans quelle organisation, les individus ayant eu un diplôme en Business Administration travaillent-ils le plus ? Trier par région.
R6	Quelle est la distribution du nombre de bureaux par ville ?
R7	Parmi les start-ups qui ont réussi à soulever un maximum de fonds(plus de 30% au-dessus de la valeur des fonds moyens), lesquelles sont françaises ? Trier par ville.
R8	Pour chaque année, donner le total de fonds levés (doit comprendre tous les types de monnaie différentes converties en $). Trier dans l'ordre descendant, par année et pays de start-up.

I - Fusion des tables pour la dénormalisation

Pour passer d'un csv à un json nous avons utilisé ce code:

```
jsonfile = open('relationships.json', 'w')
fieldnames = ("id","relationship_id","person_object_id","relationship_object_id","start_at","end_at","is_pa
reader = csv.DictReader( csvfile, fieldnames)
for row in reader:
    json.dump(row, jsonfile)
    jsonfile.write('\n')
```

Ensuite pour effectuer la dénormalisation nous avons effectuer deux left join (afin de garder les lignes n'ayant pas de clé sur les deux tables merged)

```
merged2 = objet.merge(offices, left_on='entity_type', right_on = 'object_id' ,  how ='left')
merged2.to_csv("object_offices.csv", index=False)
```

Nous avons donc deux csv :

- **object_office** : merge de object et office
- **people_degree**: merge de people et degree

Ensuite nous avons effectué les vérifications: (vérifier shape des csv une fois importé, regarder la dernière ligne, tester la différence entre les types de jointures et leur résultat (left et outer ont le même résultat).

Dans la première jointure beaucoup de colonnes avaient le même nom de chaque côté, un x ou y leur a été attribué ce qui donne moins de sens aux requêtes.
Alors on a modifié tous ces noms de colonnes.

```
merged2.rename(columns={"country_code_x": "country_code_object",
merged2
```

Les **modifications de noms de colonnes** pour la table object_offices sont les suivantes:

```
"country_code_x" ->   "country_code_object"
"state_code_x"   ->   "state_code_object"
"city_x"         ->   "city_object"
"region_x"       ->   "region_object"
"description_x"  ->   "description_object"
"description_y"  ->   "description_offices"
"region_y"       ->   "region_offices"
"city_y"         ->   "city_offices"
"state_code_y"   ->   "state_code_offices"
"country_code_y" ->   "country_code_offices
```

Cette même jointure a créé une table de 54 colonnes, ce qui fait beaucoup pour une table et beaucoup de colonnes ne sont pas utilisées pour nos requêtes. Nous allons donc les retirer.

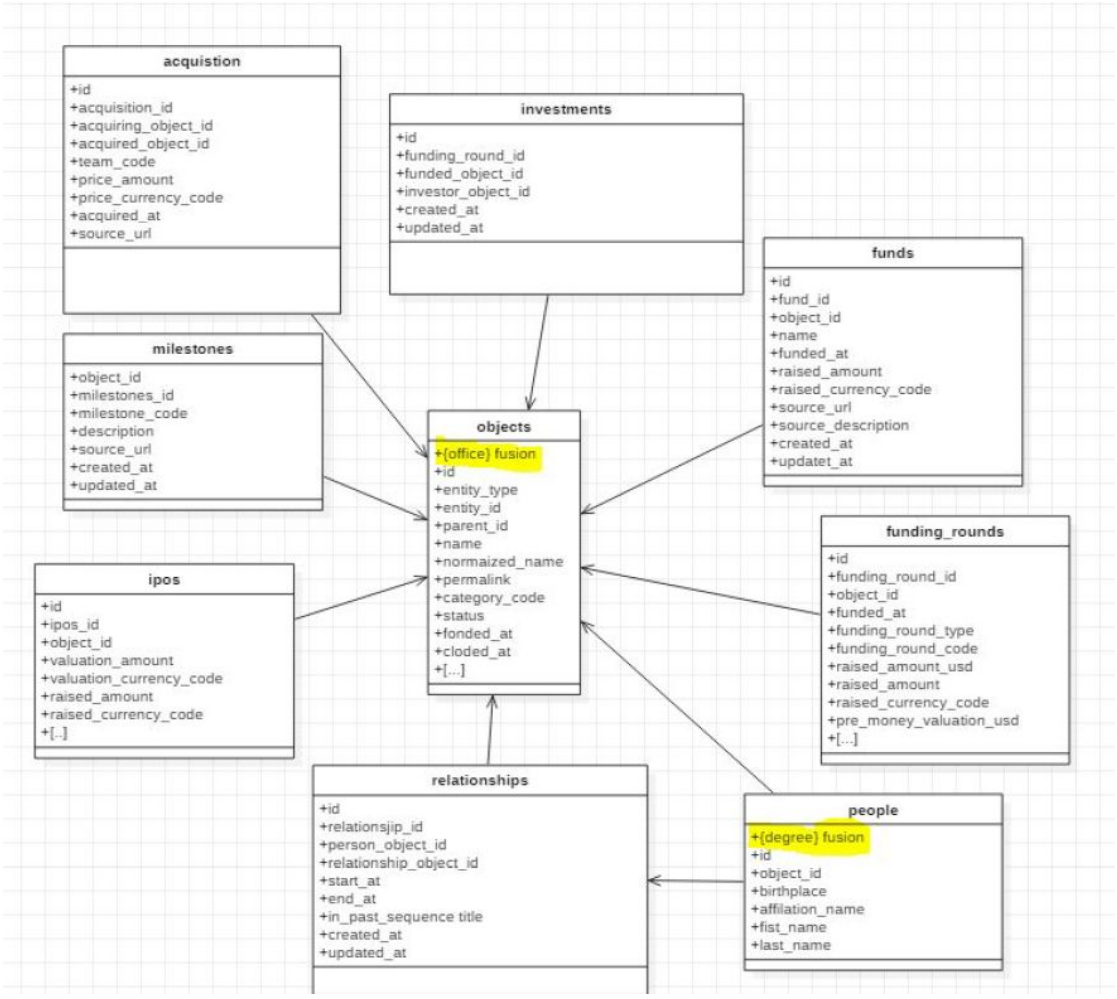
On retire donc 27 colonnes :

'entity_id', 'parent_id', 'normalized_name', 'permalink', 'category_code', 'founded_at', 'closed_at', 'domain', 'homepage_url', 'twitter_username', 'logo_url', 'logo_width', 'logo_height', 'short_description', 'overview', 'tag_list', 'first_milestone_at', 'last_milestone_at', 'milestones', 'relationships', 'created_by', 'created_at_x', 'updated_at_x', 'object_id', 'office_id', 'created_at_y', 'updated_at_y'

```
merged2.drop(columns=[ 'entity_id', 'parent_id',  'normalized_name', 'permalink',
```

Nous avons donc bien un ensemble de tables (sous format json) respectant notre deuxième dénormalisation.
Avec nos deux nouvelles tables:

- **object_office** : qui correspond à objects (sur l'uml)
- **people_degree**: qui correspond à people



II - Importation des données

Nous possédons dans notre groupe un ensemble de 8 virtuals machines aux adresses suivantes :

- devicimongodb032 (Jean-Louis) : RS3
- devicimongodb133 (Jean-Louis) : RS4
- devicimongodb056 (Ariane) : RS5
- devicimongodb157 (Ariane) : RS6
- devicimongodb163 (Johny) : RS1
- devicimongodb062 (Johny) : RS2
- devicimongodb150 (Jérémy) : CS1 / RS7
- devicimongodb049 (Jérémy) : MS / RS8

Création du ConfigServer (CS1) :

```
administrateur@devicimongodb150:~$ mongod -f conf/mongo_configSvr.conf
about to fork child process, waiting until server is ready for connections.
forked process: 24933
child process started successfully, parent exiting
administrateur@devicimongodb150:~$ mongo --host devicimongodb150 --port 27018
MongoDB shell version v4.4.1
connecting to: mongodb://devicimongodb150:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b5175b5f-6a5e-4f46-b9da-fb5a82bf348f") }
MongoDB server version: 4.4.1
```

```
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "devicimongodb150:27018",
        "ok" : 1,
        "$gleStats" : {
                "lastOpTime" : Timestamp(1607264063, 1),
                "electionId" : ObjectId("000000000000000000000000")
        },
        "lastCommittedOpTime" : Timestamp(0, 0),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607264063, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1607264063, 1)
}
configSvr:SECONDARY>
configSvr:PRIMARY>
configSvr:PRIMARY>
```

Création des shard RS1 et RS2 :

```
administrateur@devicimongodb163:~$ mongod -f conf/mongo_RS1.conf
about to fork child process, waiting until server is ready for connections.
forked process: 1706
child process started successfully, parent exiting
administrateur@devicimongodb163:~$ mongo --host devicimongodb163 --port 27017
MongoDB shell version v4.4.1
connecting to: mongodb://devicimongodb163:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b6093f21-6bc9-4827-84b9-dca77adeaaef") }
MongoDB server version: 4.4.1
```

```
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "devicimongodb163:27017",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607264259, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1607264259, 1)
}
RS1:SECONDARY>
RS1:PRIMARY>
RS1:PRIMARY>
```

```
administrateur@devicimongodb062:~$ mongod -f conf/mongo_RS2.conf
about to fork child process, waiting until server is ready for connections.
forked process: 1954
child process started successfully, parent exiting
```

```
administrateur@devicimongodb062:~$ mongo --host devicimongodb062 --port 27017
MongoDB shell version v4.4.1
connecting to: mongodb://devicimongodb062:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5ac77ea9-f93b-4c2e-b9f0-c076d87abb52") }
MongoDB server version: 4.4.1
```

```
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "devicimongodb062:27017",
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607264753, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1607264753, 1)
}
RS2:SECONDARY>
RS2:PRIMARY>
RS2:PRIMARY>
```

Création du mongos (MS)

```
administrateur@devicimongodb049:~$ cd conf
administrateur@devicimongodb049:~/conf$ ls
mongo_RS1.conf  mongo_RS2.conf  mongo_configSvr.conf  mongos.conf
administrateur@devicimongodb049:~/conf$ nano mongos.conf
```

Changement du fichier de config “mongos.conf” pour cibler CS1 :
configdb=configSvr/devicimongodb150:27018

```
administrateur@devicimongodb049:~/conf$ mongos -f mongos.conf
{"t":{"$date":"2020-12-06T14:33:07.855Z"},"s":"W",  "c":"SHARDING", "id":24132,   "ctx":"main","msg":"Running a sharded
cluster with fewer than 3 config servers should only be done for testing purposes and is not recommended for production.
"}
about to fork child process, waiting until server is ready for connections.
forked process: 3338
child process started successfully, parent exiting
```

```
administrateur@devicimongodb049:~/conf$ mongo --host devicimongodb049 --port 30000
MongoDB shell version v4.4.1
connecting to: mongodb://devicimongodb049:30000/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a4798d22-ec0a-4a4f-aca1-51325d969258") }
MongoDB server version: 4.4.1
```

Ajout des shards :

```
mongos> sh.addShard("RS1/devicimongodb163:27017")
{
        "shardAdded" : "RS1",
        "ok" : 1,
        "operationTime" : Timestamp(1607265544, 8),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607265544, 8),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("RS2/devicimongodb062:27017")
{
        "shardAdded" : "RS2",
```

Vérification du statut :

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("5fcce73f796e889c20ec4e35")
  }
  shards:
        {  "_id" : "RS1",  "host" : "RS1/devicimongodb163:27017",  "state" : 1 }
        {  "_id" : "RS2",  "host" : "RS2/devicimongodb062:27017",  "state" : 1 }
  active mongoses:
        "4.4.1" : 1
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                512 : Success
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                RS1     512
                                RS2     512
```

Création de la base de données :

```
mongos> use startup
switched to db startup
mongos> sh.enableSharding("startup")
{
        "ok" : 1,
        "operationTime" : Timestamp(1608035457, 6),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1608035457, 6),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

Nous avons importé manuellement dans la base de données la collection "object_office" puis nous avons créé un index sur notre clé de sharding “id” afin de pouvoir répartir la collection sur les shards.

```
mongos> db.object_office.createIndex({id: 1})
{
        "raw" : {
                "RS2/devicimongodb062:27017" : {
                        "createdCollectionAutomatically" : false,
                        "numIndexesBefore" : 1,
                        "numIndexesAfter" : 2,
                        "commitQuorum" : "votingMembers",
                        "ok" : 1
                }
        },
        "ok" : 1,
        "operationTime" : Timestamp(1608037291, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1608037291, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.shardCollection("startup.object_office", {id: 1})
{
        "collectionsharded" : "startup.object_office",
        "collectionUUID" : UUID("23756159-0f5b-40fb-bb3d-8dabb449b82a"),
        "ok" : 1,
        "operationTime" : Timestamp(1608037332, 19),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1608037332, 19),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

```
{  "_id" : "startup",  "primary" : "RS2",  "partitioned" : true,  "version" : {  "uuid" : UUID("d33f3362-2341-43f5-955a-bd24221aa384"),  "lastMod" : 1 } }
        startup.fundings_rounds
                shard key: { "object_id" : 1 }
                unique: false
                balancing: true
                chunks:
                        RS2     1
                { "object_id" : { "$minKey" : 1 } } -->> { "object_id" : { "$maxKey" : 1 } } on : RS2 Timestamp(1, 0)
        startup.object_office
                shard key: { "id" : 1 }
                unique: false
                balancing: true
                chunks:
                        RS1     5
                        RS2     5
                { "id" : { "$minKey" : 1 } } -->> { "id" : "c:189360" } on : RS1 Timestamp(2, 0)
                { "id" : "c:189360" } -->> { "id" : "c:242500" } on : RS1 Timestamp(3, 0)
                { "id" : "c:242500" } -->> { "id" : "c:3478" } on : RS1 Timestamp(4, 0)
                { "id" : "c:3478" } -->> { "id" : "c:85511" } on : RS1 Timestamp(5, 0)
                { "id" : "c:85511" } -->> { "id" : "p:144345" } on : RS1 Timestamp(6, 0)
                { "id" : "p:144345" } -->> { "id" : "p:199065" } on : RS2 Timestamp(6, 1)
                { "id" : "p:199065" } -->> { "id" : "p:250426" } on : RS2 Timestamp(1, 6)
                { "id" : "p:250426" } -->> { "id" : "p:6077" } on : RS2 Timestamp(1, 7)
                { "id" : "p:6077" } -->> { "id" : "r:30235" } on : RS2 Timestamp(1, 8)
                { "id" : "r:30235" } -->> { "id" : { "$maxKey" : 1 } } on : RS2 Timestamp(1, 9)
        startup.people_degree
                shard key: { "object_id" : 1 }
                unique: false
                balancing: true
                chunks:
                        RS1     1
                        RS2     2
                { "object_id" : { "$minKey" : 1 } } -->> { "object_id" : "p:215235" } on : RS1 Timestamp(2, 0)
                { "object_id" : "p:215235" } -->> { "object_id" : "p:70189" } on : RS2 Timestamp(2, 1)
                { "object_id" : "p:70189" } -->> { "object_id" : { "$maxKey" : 1 } } on : RS2 Timestamp(1, 2)
        startup.relationships
                shard key: { "person_object_id" : 1 }
                unique: false
                balancing: true
                chunks:
                        RS1     2
                        RS2     2
                { "person_object_id" : { "$minKey" : 1 } } -->> { "person_object_id" : "p:187403" } on : RS1 Timestamp(2, 0)
                { "person_object_id" : "p:187403" } -->> { "person_object_id" : "p:239162" } on : RS1 Timestamp(3, 0)
                { "person_object_id" : "p:239162" } -->> { "person_object_id" : "p:58045" } on : RS2 Timestamp(3, 1)
                { "person_object_id" : "p:58045" } -->> { "person_object_id" : { "$maxKey" : 1 } } on : RS2 Timestamp(1, 3)
```

Choix des clés de sharding

Clé de sharding	Collections
id	object_offices
object_id	people_degree
object_id	fundings_rounds
person_object_id	relationnships

Ajout des shard RS3 et RS4 :

```
mongos> sh.addShard("RS3/devicimongodb032:27017")
{
        "shardAdded" : "RS3",
        "ok" : 1,
        "operationTime" : Timestamp(1607291827, 7),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607291827, 7),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("RS4/devicimongodb133:27017")
{
        "shardAdded" : "RS4",
        "ok" : 1,
        "operationTime" : Timestamp(1607291859, 7),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1607291859, 7),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

```
shards:
      {  "_id" : "RS1",  "host" : "RS1/devicimongodb163:27017",  "state" : 1 }
      {  "_id" : "RS2",  "host" : "RS2/devicimongodb062:27017",  "state" : 1 }
      {  "_id" : "RS3",  "host" : "RS3/devicimongodb032:27017",  "state" : 1 }
      {  "_id" : "RS4",  "host" : "RS4/devicimongodb133:27017",  "state" : 1 }
active mongoses:
      "4.4.1" : 1
autosplit:
      Currently enabled: yes
balancer:
      Currently enabled:  yes
      Currently running:  no
      Failed balancer rounds in last 5 attempts:  0
      Migration Results for the last 24 hours:
              1024 : Success
              3 : Failed with error 'aborted', from RS1 to RS3
databases:
      {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
              config.system.sessions
                      shard key: { "_id" : 1 }
                      unique: false
                      balancing: true
                      chunks:
                              RS1     256
                              RS2     256
                              RS3     256
                              RS4     256
```

Nous avons fait de même par la suite pour passer à 6 shards puis 8 shards.

Lors du load balancing pour 6 shards, RS6 n'a pas été utilisé pour répartir les données des collections.

III - Requêtes MQL

Correctif des données:

Pour pouvoir effectuer des opérations sur les nombres (int), nous avons dû convertir certaines colonnes de string vers int:

Nous avons utilisé ce code pour la conversion dans mongo.

```
db.getCollection("object_office").find({funding_total_usd: {$exists: true}}).forEach(function(obj) {
    obj.funding_total_usd = new NumberInt(obj.funding_total_usd);
    db.getCollection("object_office").save(obj);
})
```

Ce code ne marche qu'en local pour des problèmes de temps d'exécution, nous avons donc réimporter les datasets (object_offices et fundings_rounds) depuis un serveur local via studio 3t

Nous avons fait de même pour la date dans object_offices

```
db.getCollection("object_office").find({funding_total_usd: {$exists: true}}).forEach(function(obj) {
    obj.funding_total_usd = new NumberInt(obj.funding_total_usd);
    db.getCollection("object_office").save(obj);
})
```

Requêtes:

1. **Quels sont les sujets étudiés à Harvard ?**

```
db.getCollection("people_degree").distinct("subject",{"institution":"Harvard Business School"})
```

2. **Donner les bureaux de start-up situés en Californie et qui contiennent Headquarters dans leur description.**

```
db.getCollection("object_office").find({
    "state_code_offices":"CA", "entity_type": "Company",
    "description_offices": /Headquarters/i })
```

3. **Trouver toutes les levées de fonds de plus 50 millions, qui sont en dollars et ont eu lieu en 2010 et indiquer le nom de la start-up associée (somme des levées de fonds).**

```
opLookUp = {
    $lookup: {
        "from": "object_office",
        "localField": "object_id",
        "foreignField": "id",
        "as": "object_office"
    }}

opProject = {
    $project: {
        "raised_amount_usd": 1,
        "funding_round_id": 1,
        "object_id": 1,
        "object_office.name" : 1,
        "object_office.id" : 1
    }}

opMatch = {
    $match: {
        "raised_currency_code": "USD",
        "funded_at": /2010/i,
        "raised_amount_usd": {
            "$gt": 50000000,
            "$not": { "$lte": 50000000.0 }
        }}}

db.getCollection("fundings_rounds").aggregate([opLookUp, opMatch, opProject])
```

4. **Le nom des bureaux situés en Californie dont des employés travaillent chez Facebook.**

```
opLookUp = {
$lookup: {
            "from": "relationships",
           "localField": "object_id",
           "foreignField": "person_object_id",
           "as": "relation"
    }}

opMatch = {
$match: {
        "affiliation_name": "Facebook"
    }}

opProject = {
$project: {
        "facebook_company" :
        "$relation.relationship_object_id",
        "first_name" : 1,
        "last_name" : 1,
        "object_id":1,
        "_id" : 0
    }}

opOut = { $out: "facebook_relationship" }

db.getCollection("people_degree").aggregate([opLookUp,opMatch, opProject]);

company = db.getCollection("facebook_relationship").distinct("facebook_company");

db.getCollection("object_office").find({"id" : {"$in" : company},"state_code_object" : "CA"},{"name" :1, "id" :1 ,"state_code_offices" : 1});
```

5. **Dans quelle organisation, les individus ayant eu un diplôme à Stanford *University travaillent-ils le plus ? Trier par organisation.***

```
opProject = {
$project: {
        "affiliation_name": 1,
        "institution": 1,
    }}

opGroup = {
$group: {
        _id : "$affiliation_name", total : {$sum : 1}
    }}

opSort = {
$sort: { total: -1 }
};

db.getCollection("people_degree").aggregate([opProject,opGroup,opSort])

db.getCollection("object_office").aggregate({$group:{_id:"$city_offices",nombre:{$sum:+1}}})
```

On avait que 1 résultat en triant par diplôme on va trier par institution (école ou université afin d'avoir plus de résultat)

6. **Quelle est la distribution du nombre de bureaux par ville ?**

1) db.getCollection("object_office").count()
   Il y a 480327 bureaux en tout.

2) db.getCollection("object_office").find({"city_offices": ""}).count()
   Il y a 372776 bureaux auxquels une ville n'est pas référencée

On cherche donc la répartition de 107 551 bureaux.

```
db.getCollection("object_office").aggregate({$group:{_id:"$city_offices",nombre:{$sum:+1}}})
```

7. **Parmi les start-ups qui ont réussi à soulever un maximum de fonds(plus de 30% au dessus de la valeur des fonds moyens), lesquelles sont françaises ?**

1) db.getCollection("object_office").find({"country_code_object": "FRA"}).count()
   2286 start-ups en France
2) Fond moyen :
   a) db.getCollection("fundings_rounds").aggregate([{$group: {_id:null, moy: {$avg:"$raised_amount_usd"} } }])
      -> $7540355.5017193165

```
opLookUp = {
    $lookup: {
        "from": "object_office",
        "localField": "object_id",
        "foreignField": "id",
        "as": "object_office"
    }}

opProject = {
    $project: {
        "raised_amount_usd": 1,
        "funding_round_id": 1,
        "object_id": 1,
        "object_office.name" : 1,
        "object_office.id" : 1,
        "object_office.city_object":1,
        "object_office.country_code_offices":1

    }}

opMatch = {
    $match: {
        "raised_amount_usd": {
            "$gt": 7540355*(1.3),
            "$not": { "$lte": 7540355*(1.3) }
        },
        "raised_currency_code":"EUR",
        "object_office.country_code_offices" : "FRA"
        } }

db.getCollection("fundings_rounds").aggregate([opLookUp, opMatch, opProject])
```

8. **Pour chaque année, donner le total de fonds levés . Trier dans l'ordre descendant, par année et pays de start-up**

```
opProject = {
    $project: {
    "country_code_object": 1,
    "funding_total_usd" : 1,
    "total" : 1,
     year : {$year : "$last_funding_at"}
    }}

opGroup = {
    $group: {
        _id : {year: "$year" , pays : "$country_code_object"} ,
        fonds : {$sum: "$funding_total_usd"}    ,
    }}

opSort = {
    $sort: { fonds: -1 }
}

db.getCollection("object_office").aggregate([opProject,opGroup,opSort])
```

IV - Performances et test avec shard

1) Configuration avec 2 shards

R1 :

```
db.getCollection("people_degree").explain(true).distinct("subject",{"institution":"Harvard Business School"})
```

```
"executionStats" : {
    "nReturned" : 1880.0,
    "executionTimeMillis" : 95.0,
    "totalKeysExamined" : 0.0,
    "totalDocsExamined" : 267694.0,
    "executionStages" : {
        "stage" : "SHARD_MERGE",
```

Le temps d'exécution au premier essai de la première requête est de **95 millisecondes.**

R2 :

```
db.getCollection("object_office").explain(true).find({
    "state_code_offices":"CA", "entity_type": "Company",
    "description_offices": /Headquarters/i })
```

```
"executionStats" : {
    "nReturned" : 2640.0,
    "executionTimeMillis" : 253.0,
    "totalKeysExamined" : 0.0,
    "totalDocsExamined" : 480327.0,
    "executionStages" : {
        "stage" : "SHARD_MERGE",
```

Le temps d'exécution au premier essai de la 2ème requête est de **253 millisecondes.**

R3:

```
db.getCollection("fundings_rounds").explain(true).aggregate([opLookUp, opMatch, opProject])
```

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 117.0,
    "executionTimeMillis" : 56.0,
    "totalKeysExamined" : 0.0,
    "totalDocsExamined" : 52928.0,
    "executionStages" : {
        "stage" : "PROJECTION_DEFAULT",
```

Le temps d'exécution au premier essai de la 3ème requête est de **56 millisecondes.**

R4 :

```
db.getCollection("people_degree").explain(true).aggregate([opMatch, opProject]);
```

```
"executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 38.0,
    "executionTimeMillis" : 69.0,
    "totalKeysExamined" : 0.0,
    "totalDocsExamined" : 118564.0,
    "executionStages" : {
        "stage" : "PROJECTION_DEFAULT",
```

```
company = db.getCollection("facebook_relationship").explain(true).distinct("facebook_company");
```

```
"executionStats" : {
    "nReturned" : 84.0,
    "executionTimeMillis" : 3.0,
    "totalKeysExamined" : 0.0,
    "totalDocsExamined" : 84.0,
    "executionStages" : {
        "stage" : "SINGLE_SHARD",
```

```
db.getCollection("object_office").explain(true).find({"id" : {"$in" : company},"state_code_object" : "CA"},{"name" :1, "id" :1 ,"state_code_offices" : 1}
```

```
"executionStats" : {
    "nReturned" : 131.0,
    "executionTimeMillis" : 4.0,
    "totalKeysExamined" : 306.0,
    "totalDocsExamined" : 203.0,
    "executionStages" : {
        "stage" : "SINGLE_SHARD",
```

R5:

```
db.getCollection("object_office").explain(true).aggregate({$group:{_id:"$city_offices",nombre:{$sum:+1}}})
```

Le temps d'exécution au premier essai de la 5eme requête est de **206 millisecondes.**

R6:

```
db.getCollection("object_office").explain(true).aggregate({$group:{_id:"$city_offices",nombre:{$sum:+1}}})
```

Le temps d'exécution au premier essai de la 6eme requête est de **502 millisecondes.**

R7:

```
db.getCollection("fundings_rounds").explain(true).aggregate([ opMatch, opProject])
```

Le temps d'exécution au premier essai de la 8eme requête est de **43 millisecondes.**

R8 :

```
db.getCollection("object_office").explain(true).aggregate([opProject,opGroup,opSort])
```

Le temps d'exécution au premier essai de la 8eme requête est de **1042 millisecondes.**

Temps en ms par requête	R1	R2	R3	R4	R5	R6	R7	R8
1	95	253	56	75	206	502	43	1042
2	93	265	63	79	250	496	43	1070
3	93	277	56	78	224	505	42	1112
4	93	250	57	70	205	493	45	1122
5	92	244	58	75	209	514	48	1102
6	91	264	65	73	241	493	44	1115
7	94	242	53	93	222	473	47	1116
8	91	253	59	74	198	478	43	1064
9	91	289	55	79	217	503	43	1074
10	90	245	52	76	225	489	44	1103
Moyenne	92.25	256.37	57.12	76.12	218.62	494.87	44.2	1094.5

2) Configuration avec 4 shards

Temps en ms par requête	R1	R2	R3	R4	R5	R6	R7	R8
1	68	159	62	24	69	88	43	189
2	67	152	56	29	66	92	42	204
3	78	146	59	25	66	83	46	207
4	66	145	55	27	70	94	51	192
5	65	149	55	24	68	85	41	198
6	74	146	53	26	67	89	42	202
7	71	148	56	24	64	87	41	203
8	71	159	53	23	65	101	45	202
9	69	143	55	26	69	88	44	194
10	74	138	56	25	66	89	44	217
Moyenne	70	148.5	55.62	25.12	67	89	43.37	200.37

3) Configuration avec 6 shards

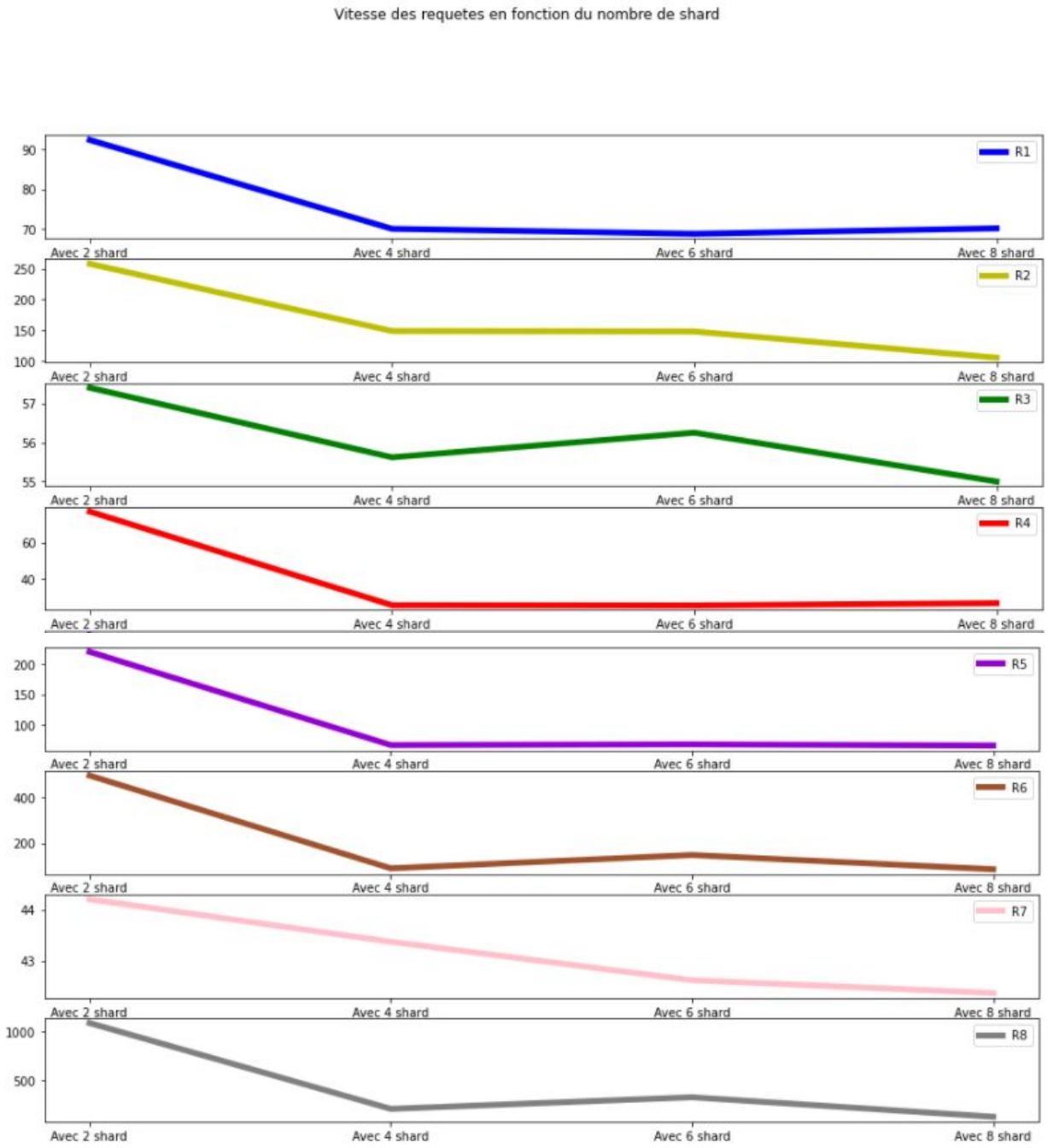
Temps en ms par requête	R1	R2	R3	R4	R5	R6	R7	R8
1	66	145	57	25	78	152	44	318
2	68	144	57	27	67	146	42	322
3	73	150	56	27	67	145	43	318
4	66	140	55	26	68	151	42	317
5	63	151	55	24	70	146	41	316
6	66	159	66	23	65	189	45	319
7	71	164	55	24	68	147	41	320

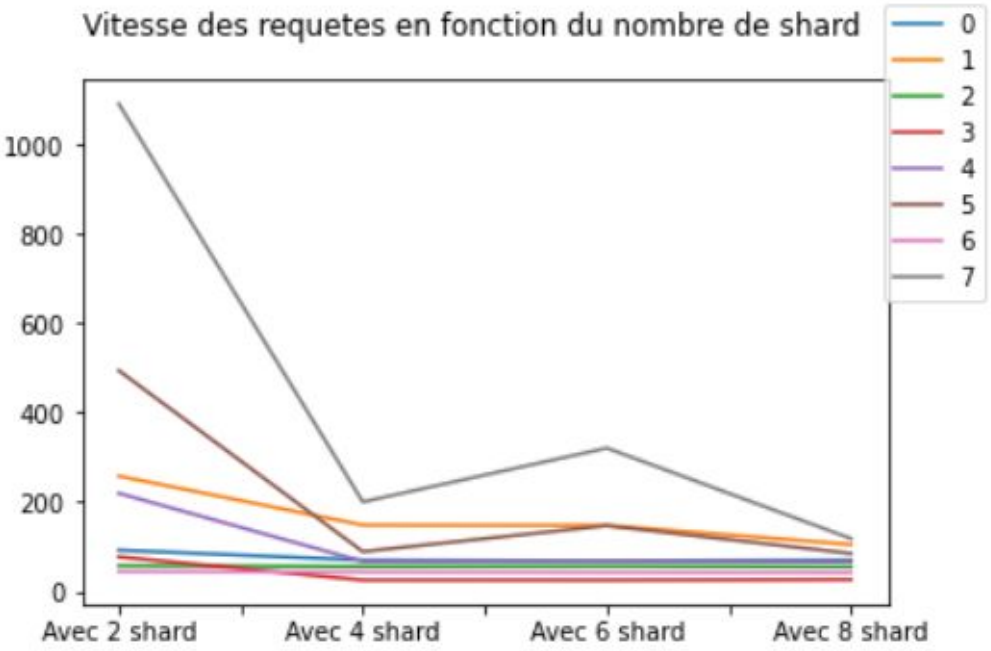
8	75	143	54	23	66	146	48	356
9	68	150	59	24	72	144	43	331
10	72	140	56	28	67	140	40	322
Moyenne	68.75	147.75	56.25	25	68.12	147.12	42.62	320.87

4) Configuration avec 8 shards

Temps en ms par requête	R1	R2	R3	R4	R5	R6	R7	R8
1	72	105	55	24	72	84	41	121
2	72	107	55	25	68	86	42	116
3	64	106	57	28	64	82	43	116
4	72	102	56	26	63	84	42	117
5	76	104	55	28	69	83	44	116
6	65	113	54	27	69	86	45	119
7	71	101	54	26	66	89	44	120
8	68	111	55	34	66	89	42	119
9	74	103	55	28	64	84	41	118
10	67	102	55	22	64	83	41	122
Moyenne	70.12	105	55	26.5	66.25	84.87	42.37	118.25

Graphiques de performance





Lien du git: https://github.com/jdelebec/dev_app_cloud