Comparaison des outils de base de données graphes ArangoDB, CosmoDB et JanusGraph

Table des matières

Choix des moteurs 2

Installation et utilisation de chaque moteur 2

Structure du jeu de données, description des cas d'usages. 2

Etapes de chargement du jeu dans chaque moteur 2

Une comparaison des moteurs en termes d'installation/utilisation par rapport à votre jeu de données et de vos cas d'usage. ..... **Erreur ! Signet non défini.**

Analyse théorique de chaque moteur 6

Notes 7

Bibliographie 8

Choix des moteurs

Installation et d'utilisation de chaque moteur

JanusGraph et ArangoDB

JanusGraph –not ok

Cosmos DB

Structure du jeu de données, description des cas d'usages.

Chargement du jeu de données

Nous utilisons un script Python, que vous pourrez trouver sur notre dépôt *GitHub* [10] pour transformer les données du .csv en code utilisable par nos moteurs de base de données graphe.

JanusGraph –not ok

L'exécution du script python fait générer des exceptions du côté du serveur JanusGraph, au niveau de son composant Berkeley. Un Git Issues est toujours ouvert. Il est seulement possible d'insérer les nœuds. Nous avons exécuté notre script avec des images Docker Janusgraph différentes et des versions de Gremlin Python différentes, sans succès. Nous avons tout de même relevé le temps d'exécution pour l'insertion des nœuds en fonction des versions.

Voici les temps d'exécution pour l'insertion des données :

- Avec JanusGraph 0.6 et Gremlin-Python 3.6.1 :
  - Temps d'exécution d'insertion des noeuds: 600s
  - Temps d'exécution d'insertion des arêtes : **Impossible**
- Avec JanusGraph 0.5 et Gremlin-Python 3.6.1 :
  - Insertion des données (nœuds ET arêtes) via cette version de Gremlin-python **impossible**
- Avec JanusGraph 0.5 et Gremlin-Python 3.6.0:
  - Temps d'exécution d'insertion des noeuds: 800s
  - Temps d'exécution d'insertion des arêtes : **Impossible**

ArangoDB

Cosmos DB

L'expression de vos requêtes pour chaque moteur, le résultat obtenu et leurs temps d'exécution pour des jeux de données de différentes tailles.

Point de vue Utilisateur standard :

1. Les dix villes où il y a le plus d'inscrits
2. Le score d'Emmanuel Macron à Paris
3. Pour chaque candidat, la ville où il a fait le meilleur résultat
4. Les résultats des communes dans laquelle l'abstention est supérieure à 50%

JanusGraph –not ok

ArangoDB

FOR c in communes SORT c.inscrits DESC LIMIT 10 RETURN c

```aql
  FOR c IN communes
    FILTER c.libelle_commune == "Paris"
    FOR e IN elit
      FILTER e._from == c._id
      FILTER e._to == "candidats/EM"
      RETURN e.score
  ```

```aql
  FOR candidat IN candidats
  LET best = (
    FOR commune IN communes
      LET score = (
        FOR edge IN elit
          FILTER edge._from == commune._id
          FILTER edge._to == candidat._id
          RETURN edge.score
      )[0]
      RETURN {commune: commune, score: score}
  )
  LET best_commune = (
    FOR commune IN best
      FILTER commune.score == MAX(best[*].score)
      RETURN commune.commune
  )[0]
  RETURN {candidat: candidat.full_name, meilleur_resultat: best_commune.libelle_commune}
  ```

```aql
  FOR commune IN communes
  FILTER commune.abstentions > 50
```

```
LET results = (
  FOR edge IN elit
    FILTER edge._from == commune._id
    LET candidat = DOCUMENT(edge._to)
    RETURN {candidat: candidat.full_name, score: edge.score}
)
RETURN {commune: commune.libelle_commune, results: results}
```

Cosmos DB –not ok

```
def create_insert_vertices_query(data):
    """
    Create the query to insert the vertices in the graph from the `data` DataFrame.
    """
    # We will create two collections : 'candidats' and 'communes'.
    # The 'candidats' collection will contain the candidats.
    # The 'communes' collection will contain the communes.

    # We will create a list of queries to insert the vertices.
    queries = []

    # We will create a query to insert the candidats with the dict
    for candidat in candidats_dict:
        query = "g.addV('candidat').property('pk', '{0}').property('full_name', '{1}')".format(
            candidat, candidats_dict[candidat]
        )
        queries.append(query)
```

```
    for index, row in data.iterrows():
        commune_key = f"D{row['Code du departement']}C{row['Code de la commune']}"
        libelle_commune = (
            row["Libelle de la commune"].replace("'", r"\'").replace('"', "")
        )
        libelle_departement = (
            row["Libelle du departement"].replace("'", r"\'").replace('"', ""),
        )
        libelle_departement = libelle_departement[0]
        query = f"g.addV('commune').property('pk', '{commune_key}')"
        query += f".property('libelle_commune', '{libelle_commune}')"
        query += f".property('code_commune', '{row['Code de la commune']}')"
        query += f".property('code_departement', '{row['Code du departement']}')"
        query += f".property('libelle_departement', '{libelle_departement}')"
        query += f".property('inscrits', {row['Inscrits']})"
        query += f".property('abstentions', {row['Abstentions']})"
        query += f".property('pourcentage_abstentions', {row['%Abs Ins']}f)"
        query += f".property('votants', {row['Votants']})"
        query += f".property('pourcentage_votants', {row['%Vot Ins']}f)"
        query += f".property('blancs', {row['Blancs']})"
        query += f".property('pourcentage_blancs', {row['%Blancs Ins']}f)"
        query += f".property('pourcentage_blancs_sur_votants', {row['%Blancs Vot']}f)"
        query += f".property('nuls', {row['Nuls']})"
        query += f".property('pourcentage_nuls', {row['%Nuls Ins']}f)"
        query += f".property('pourcentage_nuls_sur_votants', {row['%Nuls Vot']}f)"
        query += f".property('exprimes', {row['Exprimes']})"
        query += f".property('pourcentage_exprimes', {row['%Exp Ins']}f)"
```

```
        query += f".property('pourcentage_exprimes_sur_votants', {row['%Exp
Vot']}f)"

        queries.append(query)

    return queries
```

Analyse théorique de chaque moteur

JanusGraph

ArangoDB

Mode de déploiement

ArangoDB supporte plusieurs modes de déploiement résilient pour répondre aux besoins exacts d'un projet.

Instance unique (ce que nous avons fait)
Cluster
Active Failover

Data sharding

Cosmos DB

Name	ArangoDB	JanusGraph	Microsoft Azure Cosmos DB
Primary database model	Document store Graph DBMS Key-value store Search engine	Graph DBMS	Document store Graph DBMS Key-value store Wide column store
Initial release	2012	2017	2014
APIs and other access methods	AQL Foxx Framework Graph API (Gremlin) GraphQL query language HTTP API Java & SpringData JSON style queries VelocyPack/VelocyStream	Java API TinkerPop Blueprints TinkerPop Frames TinkerPop Gremlin TinkerPop Rexster	DocumentDB API Graph API (Gremlin) MongoDB API RESTful HTTP API Table API
Partitioning methods	Sharding	yes	Sharding
Replication methods	Source-replica replication with	yes	yes

	configurable replication factor		
Consistency concepts	Eventual Consistency Immediate Consistency OneShard (highly available, fault-tolerant deployment mode with ACID semantics)	Eventual Consistency Immediate Consistency	Bounded Staleness Consistent Prefix Eventual Consistency Immediate Consistency Session Consistency

Conclusion indiquant les difficultés rencontrées, la répartition du travail entre les membres du groupe et le temps passé sur ce projet.

Difficultés rencontrées

Requêtes sur un jeu de données pas si graph-friendly,

Répartition du travail

Un sur chaque moteur de base de données. Paul a rencontré moins de difficultés dans l'utilisation d'ArangoDB que Sefkan avec JanuGraph puisque le premier est bien mieux documenté et il existe des outils pour l'utiliser...

Temps passé sur le projet

Une nuit passée au dernier moment dans une tentative de passer à un dataset plus approprié aux bases de données graphes. Mais problèmes au niveau du nettoyage, des clés de ce jeu de données, etc...

Notes

Cosmos

CosmoDB un peu chiant à mettre en place quand même pleins de boutons machin pour créer, et puis pleins de trucs à configurer avec des passwords et tt bon après normal hein

Mais bien documenté + dépôts GitHub avec des examples machin et tout

Attention aux couts cependant mais pour nous pas de pb pcq on utilise *free tier*.

Intégré PARTOUT : dans le portail Azure, avec Python, avec une extension VS… et j'en passe

Bibliographie

[1] K. Gutha, « Azure Cosmos DB Analysis », *{coding}Sight*, 13 juin 2018. [Online]. Disponible sur: https://codingsight.com/azure-cosmos-db-analysis/. [Consulté le: 10 décembre 2022]

[2] « Apache TinkerPop ». [Online]. Disponible sur: https://tinkerpop.apache.org/. [Consulté le: 10 décembre 2022]

[3] « TinkerPop Documentation ». [Online]. Disponible sur: https://tinkerpop.apache.org/docs/current/reference/. [Consulté le: 10 décembre 2022]

[4] « Gremlin Graph Database IDE, Debugging & Visualization Tool », 4 mars 2019. [Online]. Disponible sur: https://gdotv.com/. [Consulté le: 11 décembre 2022]

[5] « Créez votre compte gratuit Azure aujourd'hui | Microsoft Azure ». [Online]. Disponible sur: https://azure.microsoft.com/fr-fr/free/. [Consulté le: 10 décembre 2022]

[6] manishmsfte, « Démarrage rapide : API Gremlin avec Python - Azure Cosmos DB ». [Online]. Disponible sur: https://learn.microsoft.com/fr-fr/azure/cosmos-db/gremlin/quickstart-python. [Consulté le: 10 décembre 2022]

[7] « Microsoft Azure ». [Online]. Disponible sur: https://portal.azure.com. [Consulté le: 10 décembre 2022]

[8] « python-decouple: Strict separation of settings from code. » [Online]. Disponible sur: http://github.com/henriquebastos/python-decouple/. [Consulté le: 10 décembre 2022]

[9] M. de l'Intérieur, « Election présidentielle 2017 : résultats globaux du premier tour », *http://www.interieur.gouv.fr/Archives/Archives-elections/Election-presidentielle-2017/Election-presidentielle-2017-resultats-globaux-du-premier-tour*. [Online]. Disponible sur: http://www.interieur.gouv.fr/Archives/Archives-elections/Election-presidentielle-2017/Election-presidentielle-2017-resultats-globaux-du-premier-tour. [Consulté le: 10 décembre 2022]

[10] G. T, « TDR Project ». 28 octobre 2022 [Online]. Disponible sur: https://github.com/gu1lhem/tdm-2022. [Consulté le: 10 décembre 2022]

[11] « Do print statements in the programming language slows down the execution? », *Quora*. [Online]. Disponible sur: https://www.quora.com/Do-print-statements-in-the-programming-language-slows-down-the-execution. [Consulté le: 10 décembre 2022]