



# Neo4j

---

USSI57

Clément Marie-Brisson

Alexis Guillotin

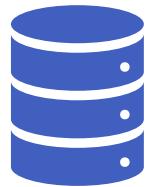
# Sommaire

---



## Introduction Neo4j

Description générale  
API/protocoles  
Exemple cas d'usages



## Implémentation d'un cas d'usage

Déploiement sur une VM  
Dataset utilisé  
Création du graphe



## Conclusion

# Introduction Neo4j

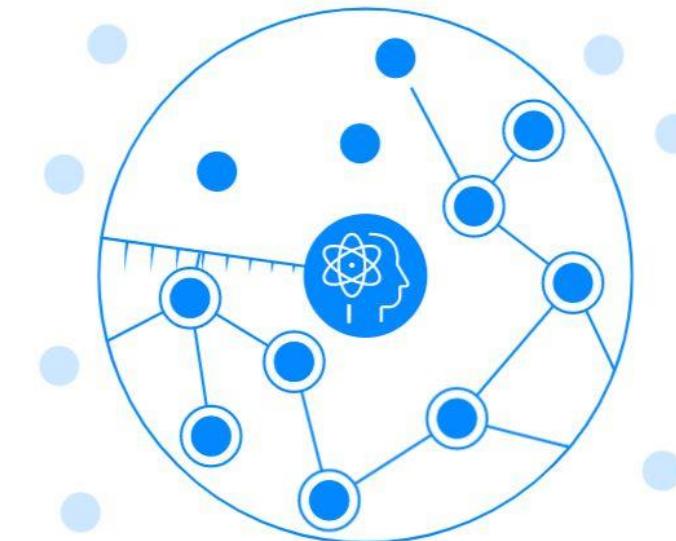


# Description générale

- Open-source
- Base de données de graphes
- NoSQL
- Compatible ACID
- Depuis 2007



neo4j



# API / Langages



Les API :



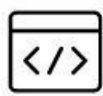
Neo4j Desktop

Pour la gestion des bases de données locales



Bibliothèque GraphQL

Pour des applications modernes basées sur les API



Neo4j Browser

Accéder à votre base de données n'importe où

Les langages compatibles :

- Python
- JavaScript
- GraphQL
- Java
- Spring Boot
- .NET
- Go



# Protocoles



## Bolt vs HTTP

### HTTP

#### Request [317 bytes]

```
POST http://localhost:7474/db/data/transaction/commit
Content-Type: application/json
Host: localhost:7474
User-Agent: py2neo/3 HTTPStream/1.5.0 Python/3.5.0-final-0 (linux)
X-Stream: true
Authorization: Basic bmVvNGo6cGFzc3dvcmQ=

{"statements": [{"statement": "RETURN 1", "parameters": {}, "resultDataContents": ["REST"]}]}
```

#### Response [213 bytes]

```
200 OK
Date: Mon, 18 Apr 2016 09:23:12 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Content-Length: 65
Server: Jetty(9.2.9.v20150224)

{"results": [{"columns": ["1"], "data": [{"rest": [1]}]}, {"errors": []}]}
```

JSON payload

### Bolt

#### Request [22 bytes]

```
00:0C:B2:10:88:52:45:54:55:52:4E:20:31:A0:00:00
00:02:B0:3F:00:00
```

Stateful: client and auth info only sent once per session

#### Response [39 bytes]

```
00:00:B1:70:A1:86:66:69:65:6C:64:73:91:81:31:00
00:00:04:B1:71:91:01:00:00:00:0A:B1:70:A1:84:74
79:70:65:81:72:00:00
```

Type-safe  
binary  
payload

Significantly  
smaller requests  
and responses

# Exemples cas d'usages



## Détection et analyse des fraudes

L'analyse en temps réel des relations entre les données est essentielle pour découvrir les réseaux de fraude et autres escroqueries sophistiquées avant que les fraudeurs et les criminels ne causent des dommages durables.

[Voir Cas d'utilisation →](#)



## Surveillance de l'infrastructure réseau et base de données pour les opérations informatiques

Les bases de données de graphes sont intrinsèquement plus adaptées que le RDBMS pour donner un sens aux interdépendances complexes essentielles à la gestion des réseaux et de l'infrastructure informatique.

[Voir Cas d'utilisation →](#)



## Moteur de recommandation et système de recommandation de produits

Les moteurs de recommandation basés sur des graphiques aident les entreprises à personnaliser leurs produits, contenus et services en exploitant une multitude de connexions en temps réel.

[Voir Cas d'utilisation →](#)

# Exemples cas d'usages



## Gestion des données de référence

Organisez et gérez vos données de base avec le modèle de base de données graphique flexible et sans schéma afin d'obtenir des informations en temps réel et une vue à 360° de vos clients.

[Voir Cas d'utilisation →](#)



## Graphiques des médias sociaux et des réseaux sociaux

Exploitez facilement les connexions sociales ou inférez des relations basées sur l'activité lorsque vous utilisez une base de données graphique pour alimenter votre application de réseau social.

[Voir Cas d'utilisation →](#)



## Gestion des identités et des accès

Suivez rapidement et efficacement les utilisateurs, les actifs, les relations et les autorisations lorsque vous utilisez une base de données graphique pour la gestion des identités et des accès.

[Voir Cas d'utilisation →](#)

# Exemples cas d'usages



- Détection de réseaux de fraudeurs :

Défi : détecter **rapidement** des modèles de fraude complexe

Données utilisées : adresses IP, postales ou électroniques.

→ détection de réseaux de fraude ou de vols d'identité

→ interception de **20 % supplémentaires** de tentatives de fraude.

→ temps de traitement pour identifier des réseaux de fraude a été **divisé par 10**.

⇒ **Gain économique énorme !**



- Moteur de recommandation :

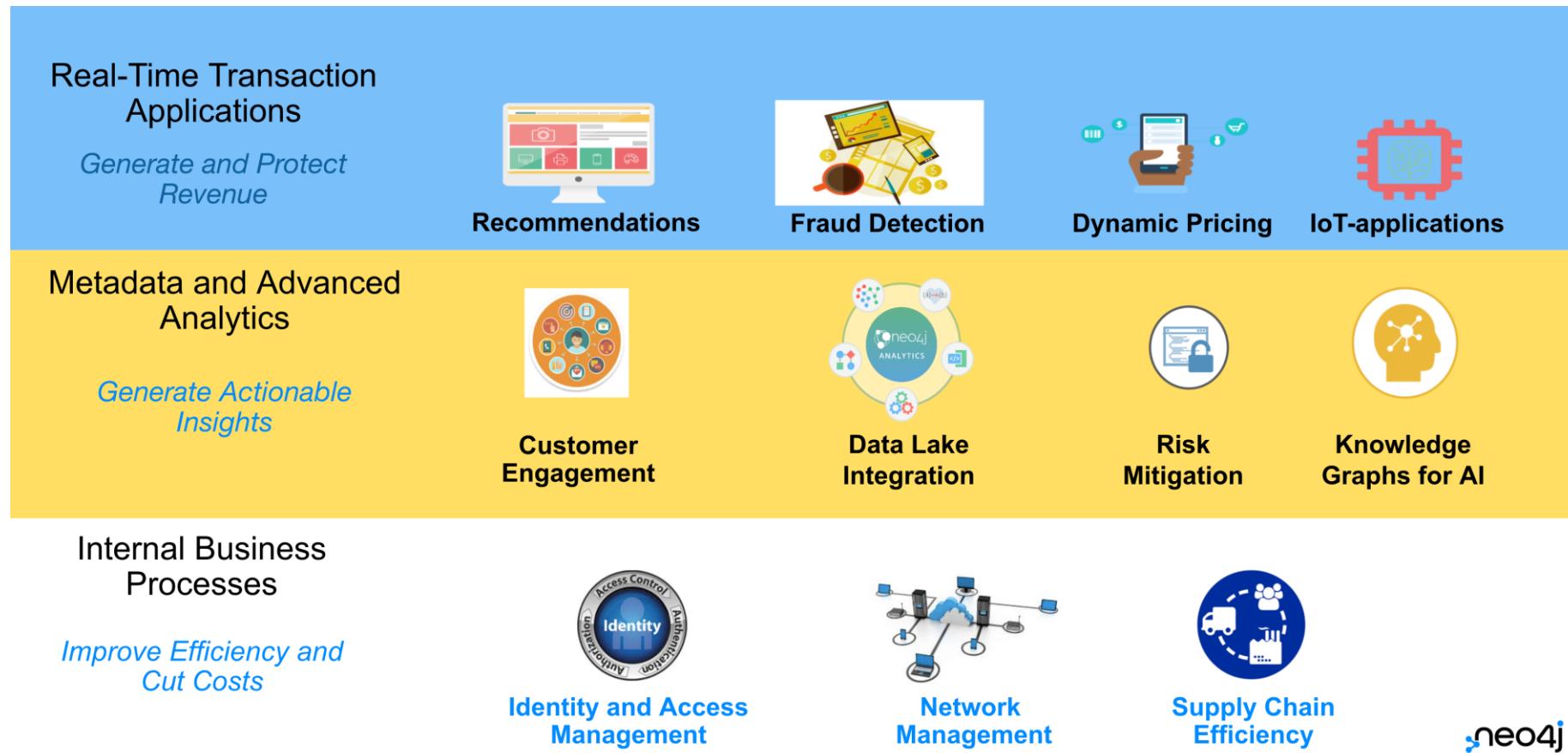
→ propose des produits similaires à ceux que les utilisateurs ont aimés.

→ recommandation en temps réel : plus l'acheteur renseigne ce qu'il veut, plus l'algorithme parcourt le graph à la recherche du meilleur match.

⇒ **Algorithme d'aide à la décision en temps réel.**



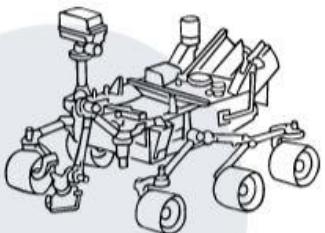
# Exemples cas d'usages



# Exemples cas d'usages



Neo4j fait économiser à la NASA des **millions de dollars et deux années de travail** sur sa mission vers Mars.



Si je devais annoncer aujourd'hui à nos enquêteurs que nous renonçons à utiliser Neo4j, ça créerait un **énorme tollé.**"



Neo4j est "**des milliers de fois plus rapide** que notre précédente solution MySQL, avec des requêtes qui demandent **10 à 100 fois moins de code.**"



"Sur deux ans, nous avons identifié un bénéfice d'exploitation de **deux millions d'euros**, et cette valeur est même structurellement sous-estimée."

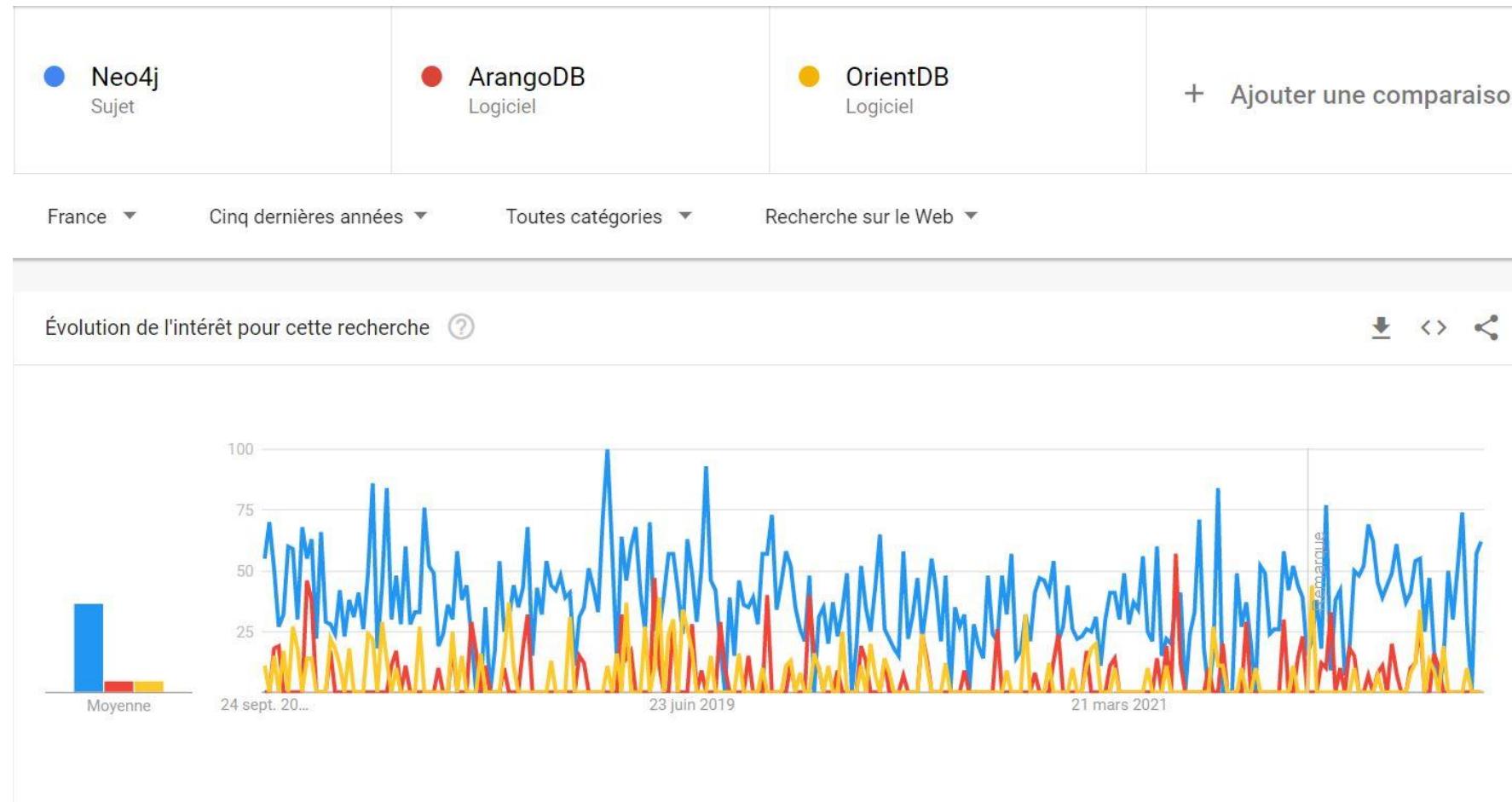


# Comparatif avec d'autres bases de données

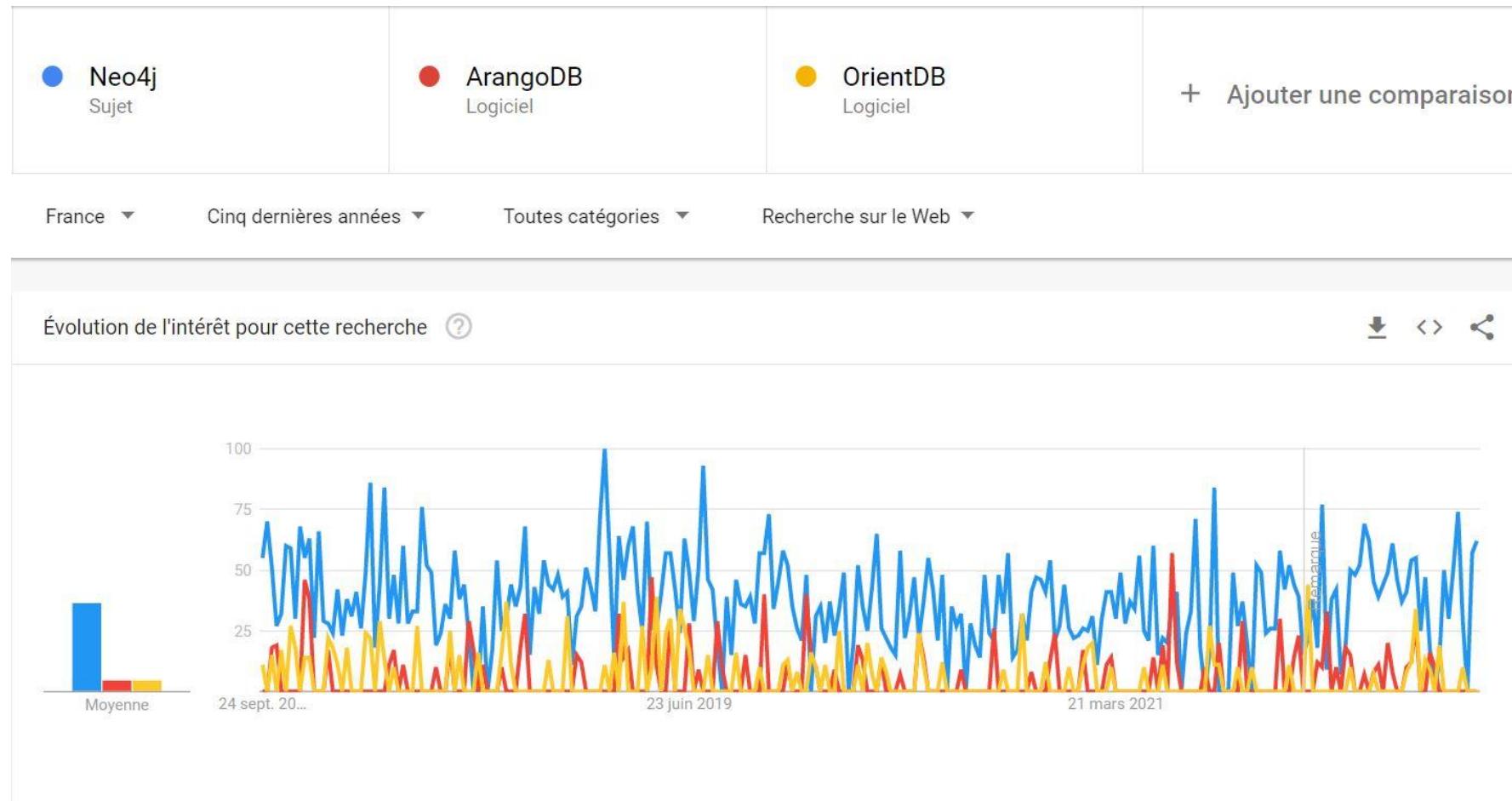


Name	Language(s)	Notes
AllegroGraph	SPARQL	RDF triplestore
ArangoDB	AQL, JavaScript, GraphQL	Base de données orientée documents, Base de données orientée graphe et système clé-valeur
DEX/Sparksee	C++, Java, .NET, Python	Base de données orientée graphe
DataStax Enterprise Graph	Java, C#, C++, Python, Node.js, Gremlin (Apache Tinkerpop)	Base de données distribuées s'appuyant sur Apache Cassandra et intégrant un moteur graphe inspiré par TitanDB.
Dgraph	Go, Javascript, Python, GraphQL	Base de données orientée graphe avec un support natif de GraphQL.
FlockDB	Scala	Base de données orientée graphe
IBM DB2	SPARQL	RDF triplestore depuis la DB2 10
InfiniteGraph	Java	Base de données orientée graphe
JanusGraph	Java, Gremlin (Apache Tinkerpop)	Base de données orientée graphe
MarkLogic	Java, JavaScript, SPARQL, XQuery	Base de données multi-modèle et RDF triplestore
Neo4j	Cypher, Gremlin (Apache Tinkerpop)	Base de données orientée graphe
OpenLink Virtuoso	C++, C#, Java, SPARQL	Middleware et RDF triplestore
Oracle	SPARQL 1.1	RDF triplestore
OrientDB	Java, Structured_Query_Language, Gremlin (Apache Tinkerpop)	Base de données multi-modèle, Base de données orientée documents et base de données orientée graphe.
OWLIM	Java, SPARQL 1.1	RDF triplestore
Sqrrl Enterprise	Java	base de données orientée graphe
Wikibase	SPARQL	RDF triplestore
Agensgraph	Cypher, SQL	Base de donnée orientée graphe, dérivée de PostgreSQL

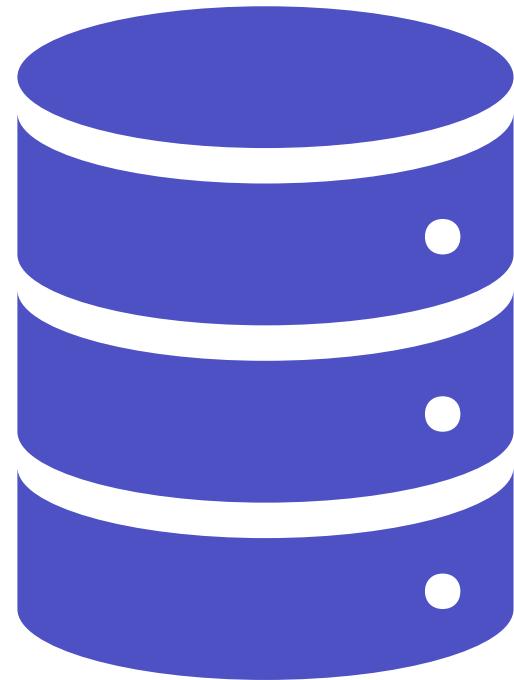
# Comparatif avec d'autres bases de données



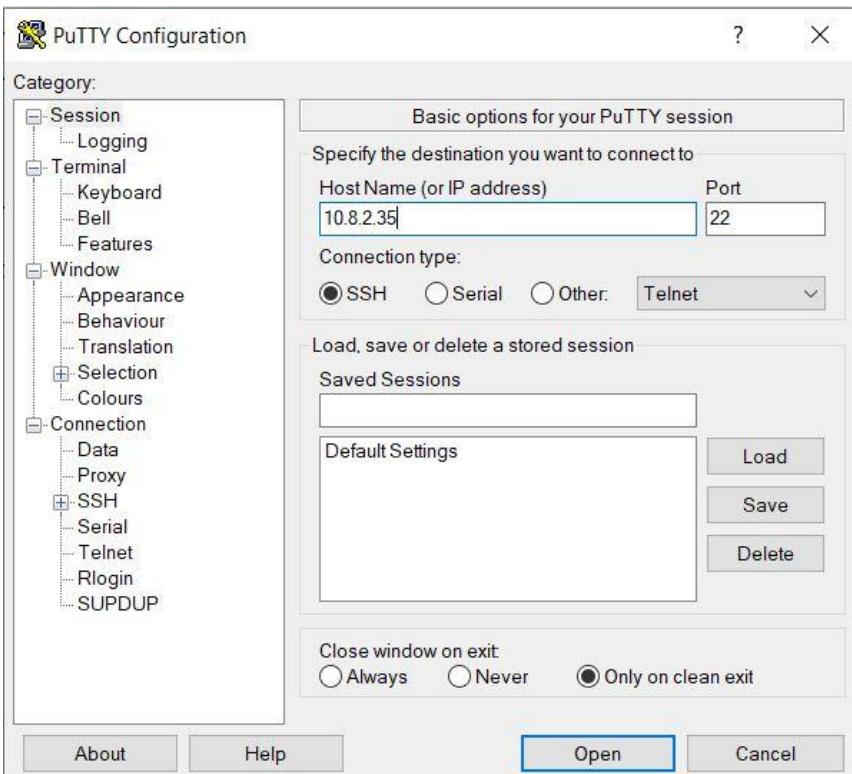
# Les points forts / faibles



# Implémentation d'un cas d'usage



# Déploiement sur une VM – 1 Installer Neo4j



1. *\$ sudo apt install neo4j*
2. *\$ sudo systemctl enable neo4j.service*
3. *\$ sudo systemctl start neo4j.service*

# Déploiement sur une VM – 1 Installer Neo4j

- *\$ sudo systemctl status neo4j.service*  
→ Vérifier que le service est « enable » et « running »

```
output
● neo4j.service - Neo4j Graph Database
  Loaded: loaded (/lib/systemd/system/neo4j.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2020-08-07 01:43:00 UTC; 6min ago
    Main PID: 21915 (java)
       Tasks: 45 (limit: 1137)
      Memory: 259.3M
        CGroup: /system.slice/neo4j.service
          ...

```

# Déploiement sur une VM – 2 Se connecter

- \$ *cypher-shell*

→ Mettre le *username* et le mot de passe par défaut : *neo4j* – *neo4j* et changer le mot de passe

```
cypher-shell prompt
username: neo4j
password: *****
Password change required
new password: *****
Connected to Neo4j 4.1.0 at neo4j://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j>
```

# Déploiement sur une VM – 3 Accès remote

- \$ sudo nano  
*/etc/neo4j/neo4j.conf*  
→ Décommenter la ligne pour autoriser l'accès en remote

```
...
*****
# Network connector configuration
*****
# With default configuration Neo4j only accepts local connections.
# To accept non-local connections, uncomment this line:
dbms.default_listen_address=0.0.0.0
...
```

# Déploiement sur une VM – 3 Accès remote

- *Depuis un shell :*

```
$ cypher-shell -a neo4j://10.8.2.35:7687
```

# Déploiement sur une VM – 3 Accès remote

- Depuis une interface graphique - Neo4j Desktop:



# Déploiement sur une VM – 3 Accès remote

- Depuis une interface graphique – Neo4j Desktop:

Example Project

[Add](#)

Username/Password    Kerberos authentication

**Username**

 neo4j

**Password**

 .....|

Use encrypted connection

[Back](#) [Save](#)



# Dataset utilisé



- <https://www.kaggle.com/datasets/rdoume/beerreviews>

DATADOME · UPDATED 4 YEARS AGO

▲ 129    New Notebook    Download (29 MB)    :

## Beer Reviews

1.5 millions of beers reviews



Data    Code (13)    Discussion (3)    Metadata

---

### About Dataset

**Context**

This is the dataset discussed in the talk "How to hire and test for data skills: A one-size-fits-all interview kit" from <https://conferences.oreilly.com/strata/strata-ny-2017/public/schedule/detail/59542>

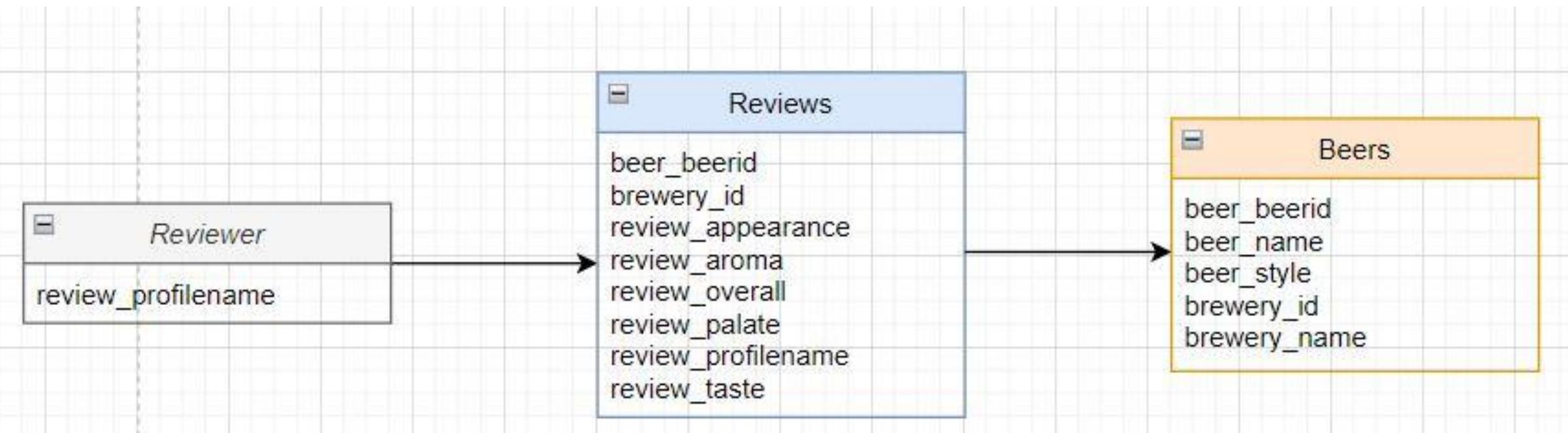
**Content**

**Usability** ⓘ  
7.06

**License**  
Unknown

**Expected update frequency**  
Not specified

# Dataset utilisé



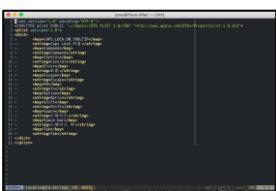
# LA CRÉATION DU GRAPHE



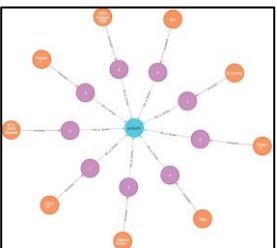
1. Configuration de l'environnement NEO4J (neo4j.conf, packages)



2. Préparation du graph (nœuds et relations)



3. Quelques exemples de rendus et possibilités.



# 1. La configuration de l'environnement

Ajout des procédures apoc.\* : package facilitant le requêtage

```
call dbms.components() yield name, versions, edition unwind  
versions as version return name, version, edition; //check neo4j  
version
```

1. Télécharger la version appropriée de apoc sur Github.
2. Placer le .jar dans le sous-dossier ‘plugins’
3. Redémarrer NEO4J

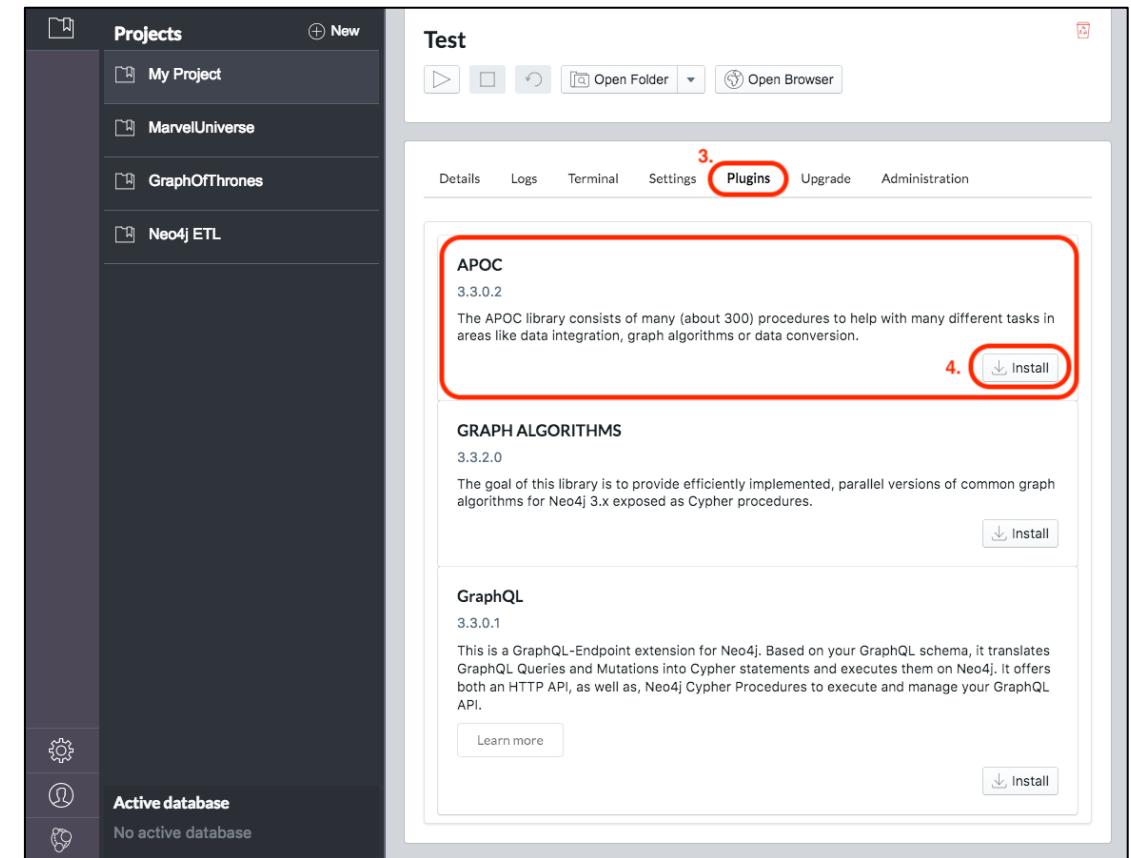
Ajout du dataset beer\_reviews.csv à la VM

```
file:/var/lib/neo4j/import/beer_reviews.csv
```

Configuration de l'utilisation mémoire de NEO4J

```
neo4j-admin memrec --memory=4G
```

```
# Based on the above, the following memory settings are recommended:  
dbms.memory.heap.initial_size=2g  
dbms.memory.heap.max_size=2g  
dbms.memory.pagecache.size=512m
```



## 2. Préparation du Graphe



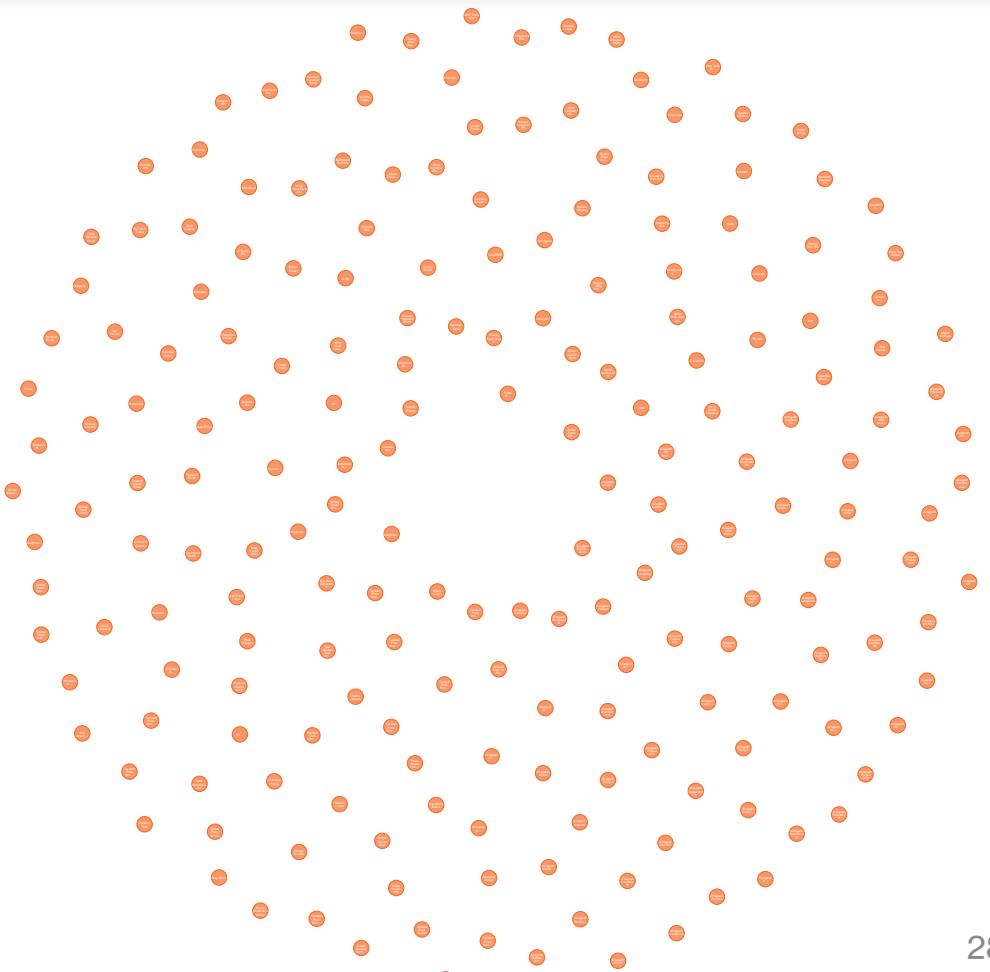
1. Création des nœuds
2. Création des relations
3. Les difficultés rencontrés et les solutions apportées
  1. La création des requêtes
  2. La mémoire nécessaire
  3. Le temps de traitement

# 2.1 Cr ation des n euds



## 1. Beers

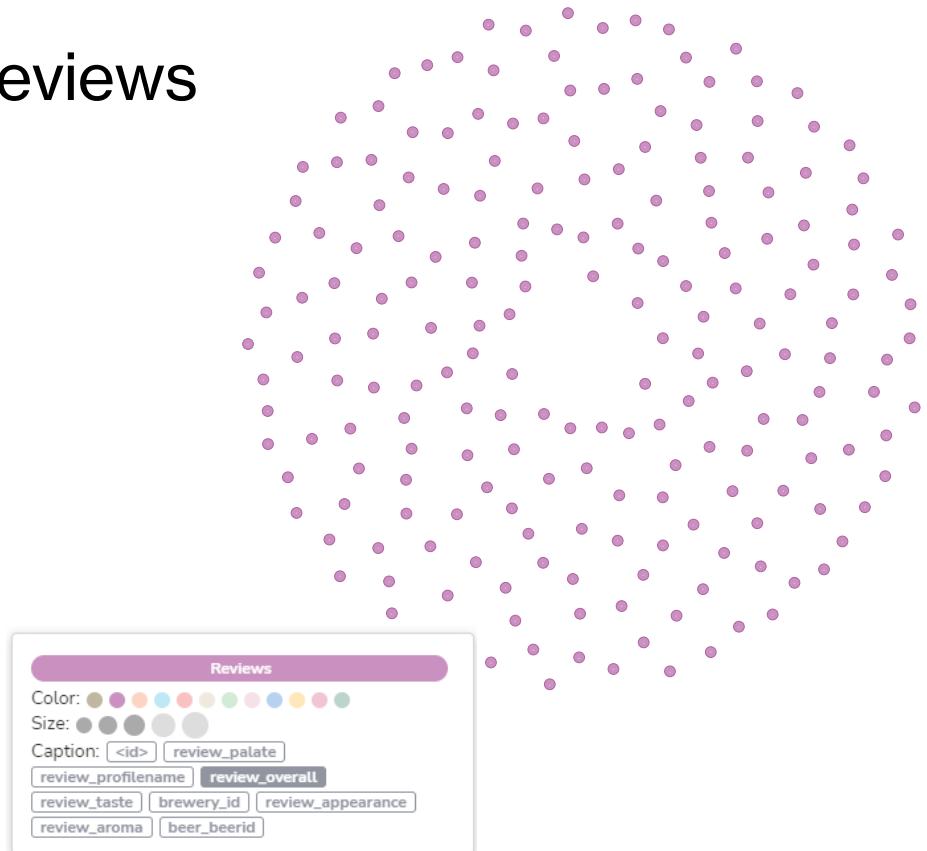
```
LOAD CSV WITH HEADERS FROM 'file:///beer_reviews.csv' AS row
WITH
    toInteger(row.brewery_id) as brewery_id,
    toInteger(row.beer_beerid) as beer_beerid,
    toString(row.brewery_name) as brewery_name,
    toString(row.beer_name) as beer_name,
    toString(row.beer_style) as beer_style,
    toInteger(row.beer_abv) as beer_abv
WHERE brewery_id < 25
MERGE (b:Beers {brewery_id: brewery_id, beer_beerid:beer_beerid})
SET
    b.brewery_id = brewery_id,
    b.beer_beerid = beer_beerid,
    b.brewery_name = brewery_name,
    b.beer_name = beer_name,
    b.beer_style = beer_style,
    b.beer_abv = beer_abv
RETURN
    count(b);
```



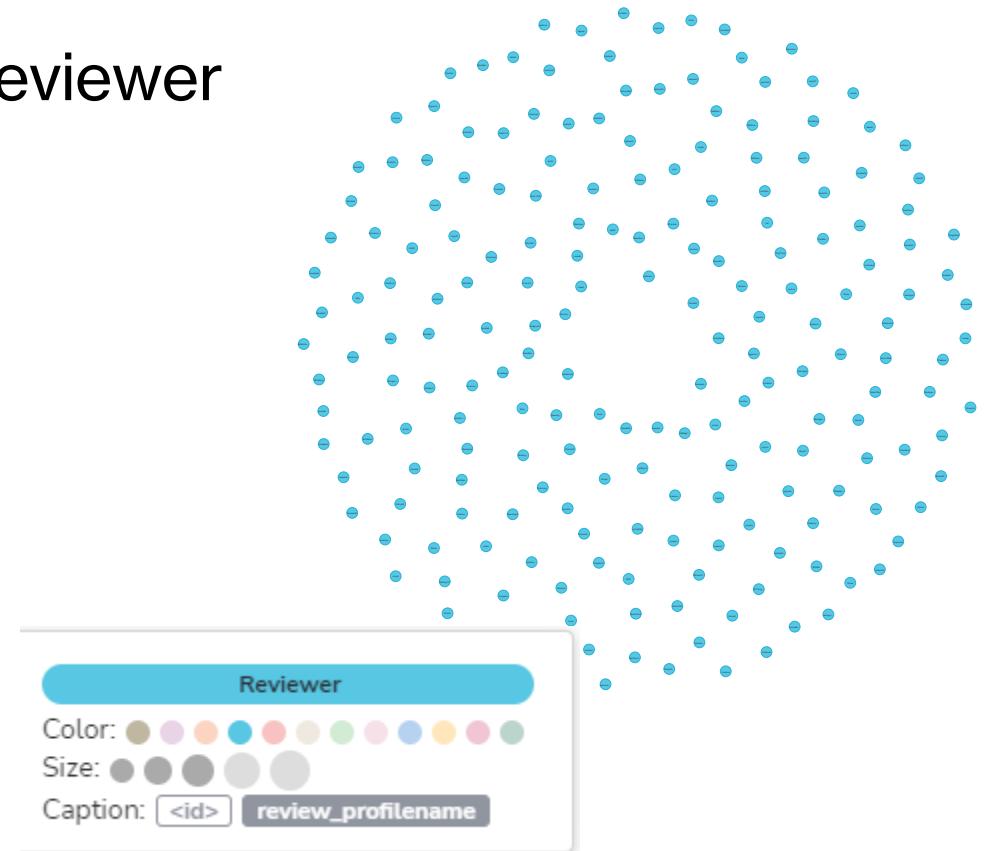
## 2.1 Cr ation des n euds



2. Reviews



3. Reviewer

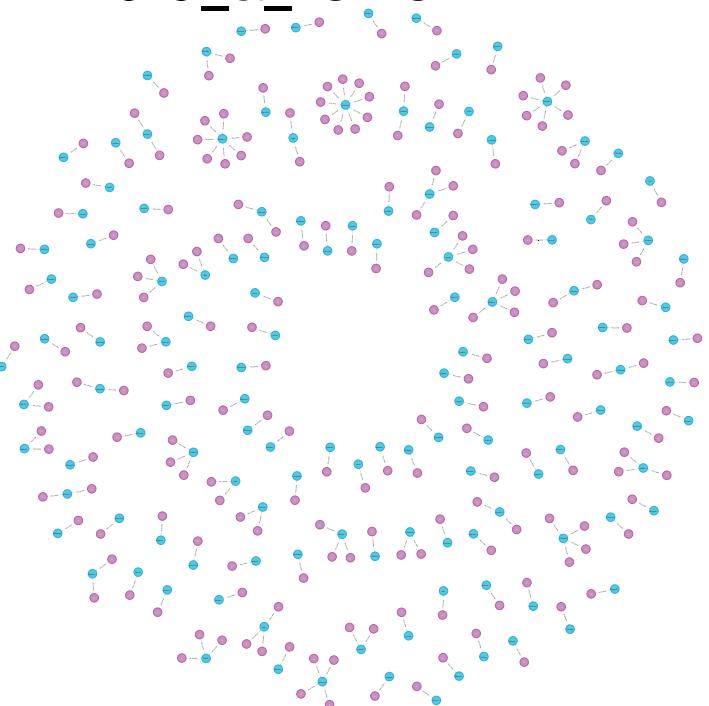


## 2.2 Création des relations

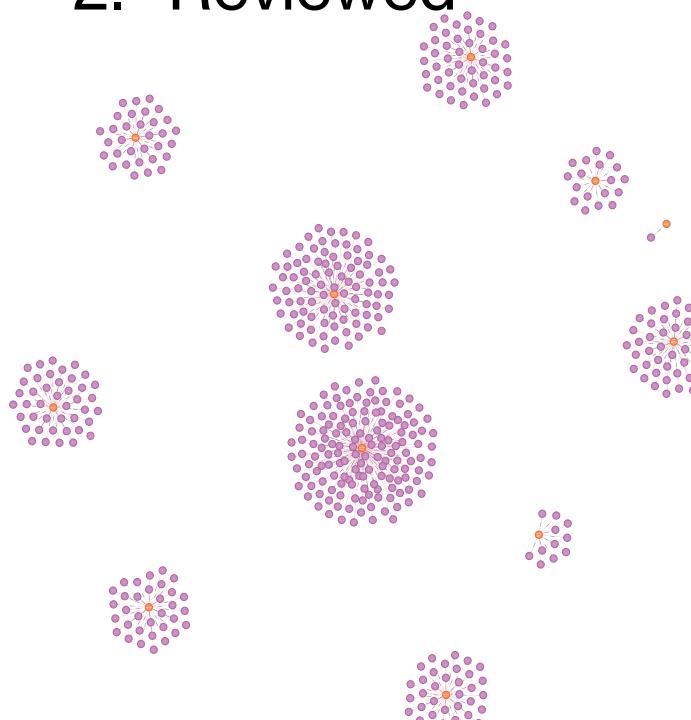


```
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM 'file:///beer_reviews.csv' AS row
WITH toInteger(row.brewery_id) AS brewery_id, toInteger(row.beer_beerid) AS beer_beerid, toString(row.review_profilename) AS review_profilename
MATCH (p:Reviews {brewery_id: brewery_id, beer_beerid: beer_beerid, review_profilename: review_profilename})
MATCH (o:Beers {brewery_id: brewery_id, beer_beerid: beer_beerid})
MERGE (o)-[rel:reviewed {brewery_id: brewery_id, beer_beerid: beer_beerid, review_profilename: review_profilename}]->(p)
RETURN count(rel);
```

1. did\_a\_review



2. Reviewed



## 2.3 Les difficultés rencontrées et les solutions apportées

### 1. La création des requêtes

```
1 LOAD CSV WITH HEADERS FROM 'file:///beer_reviews.csv' AS row
2   WITH
3     toIntegerOrNull(row.brewery_id) as brewery_id,
4     toIntegerOrNull(row.beer_beerid) as beer_beerid,
5     toStringOrNull(row.review_profilename) as review_profilename,
6     toIntegerOrNull(row.review_appearance) as review_appearance,
7     toIntegerOrNull(row.review_aroma) as review_aroma,
8     toIntegerOrNull(row.review_overall) as review_overall,
9     toIntegerOrNull(row.review_taste) as review_taste
10    RETURN brewery_id,beer_beerid,review_palate,review_profilename,review_appearance,
11        review_aroma,review_overall,review_taste
```

### 2. La mémoire nécessaire



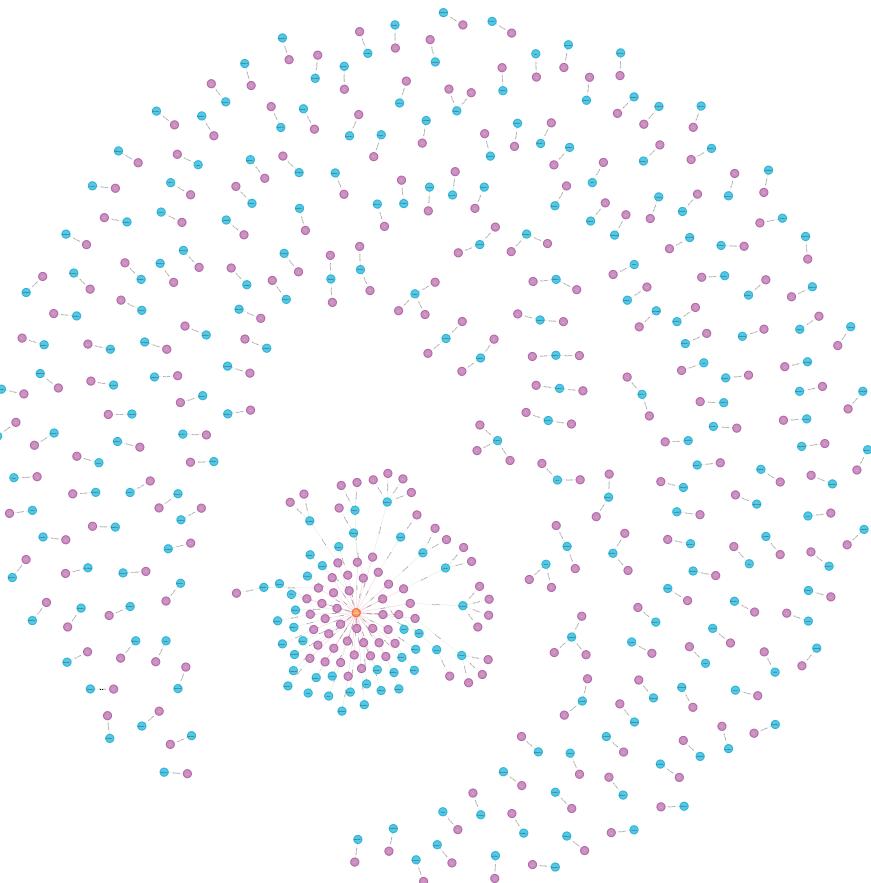
### 3. Le temps de traitement



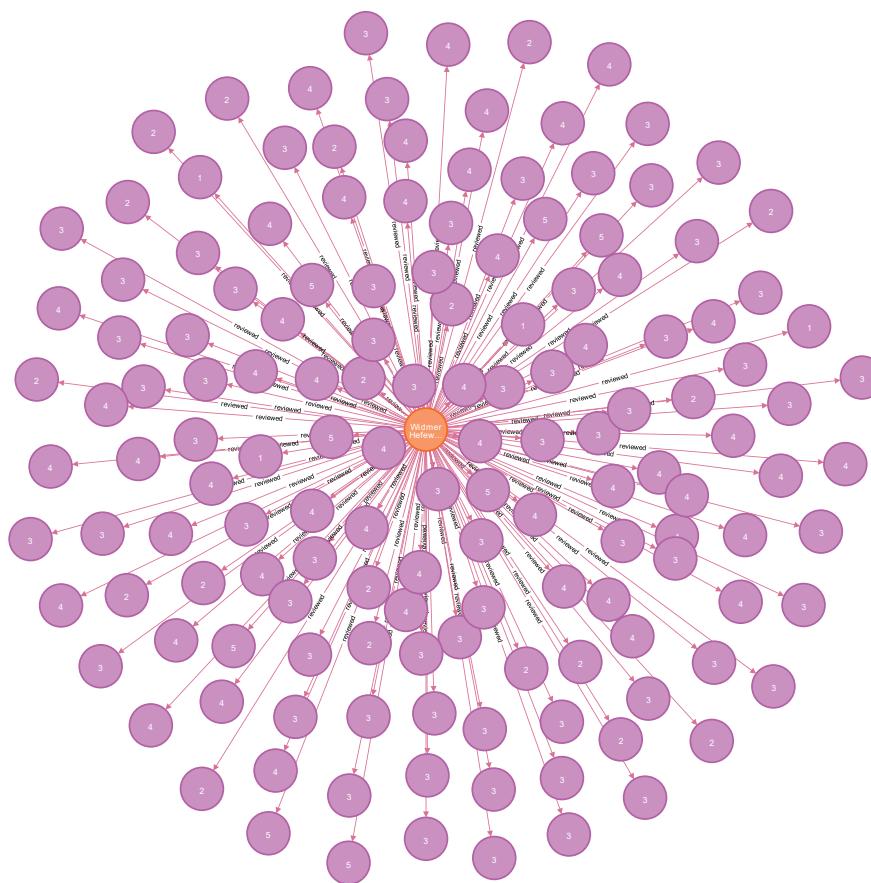
# 3. Exemples et possibilités



1. Afficher tous les nœuds et relations



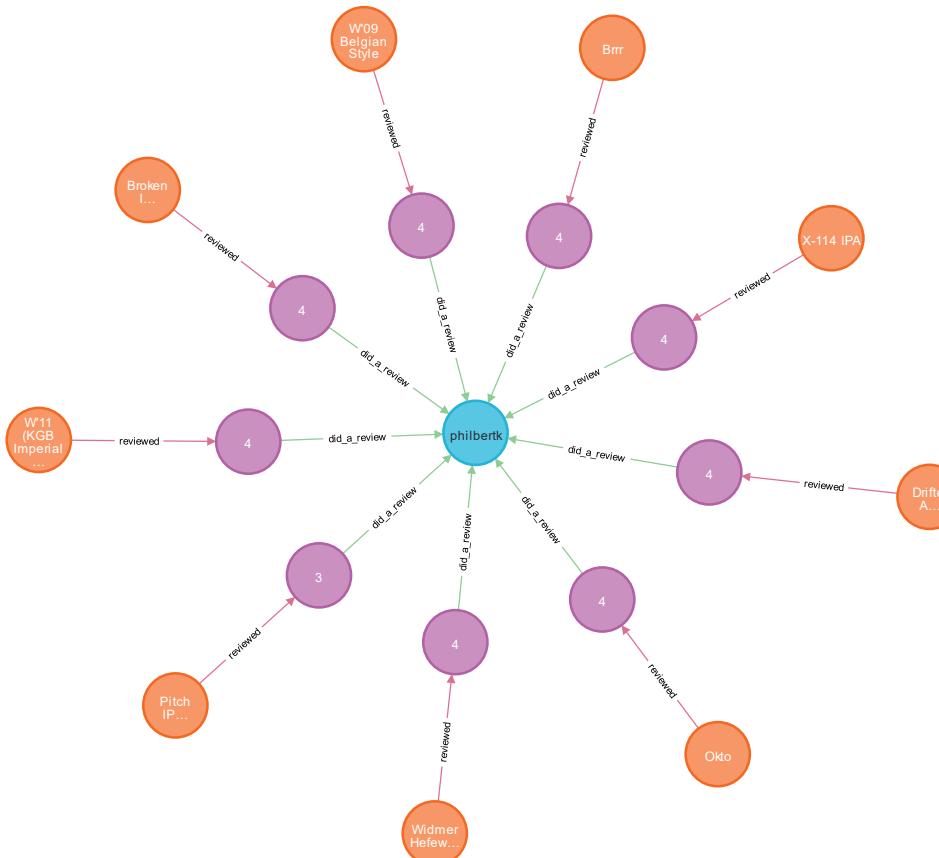
2. Trouver l'ensemble des reviews d'une bière



# 3. Exemples et possibilités



## 3. Trouver l'ensemble des reviews et des bières d'un utilisateur



# 3. Exemples et possibilités



Algorithme de recommandation de bière en fonction de la note globale:

```
//Displaying 5 most similar beers to "Widmer Hefeweizen" according to the overall review:  
| //selecting Reviewers who did a Review which reviewed the Beer called "Widmer Hefeweizen"  
MATCH (a:Beers {beer_name: "Widmer Hefeweizen"})-[:reviewed]->(b:Reviews)-[:did_a_review]->(c:Reviewer)  
| //selecting Beers being reviewed by the profilenames who also reviewed "Widmer Hefeweizen"  
MATCH (c)-[:did_a_review]->(e:Reviews)-[:reviewed]->(h:Beers)  
| //We only keep profilenames who rated our beer at min 4  
WHERE b.review_overall>=4  
RETURN h.beer_name as similar_beers, count(e) AS number_of_reviews  
ORDER BY number_of_reviews DESC  
SKIP 1 //we skip first row as it is the beer we are comparing  
LIMIT 5
```

similar_beers	number_of_reviews
"Drifter Pale Ale"	7
"Brrr"	5
"W'11 (KGB Imperial Stout)"	5
"Deadlift Imperial IPA"	3
"X-114 IPA"	3

# Démonstration



# Conclusion

