Du SQL aux bases de données graphes
Exemple de Neo4j

•••

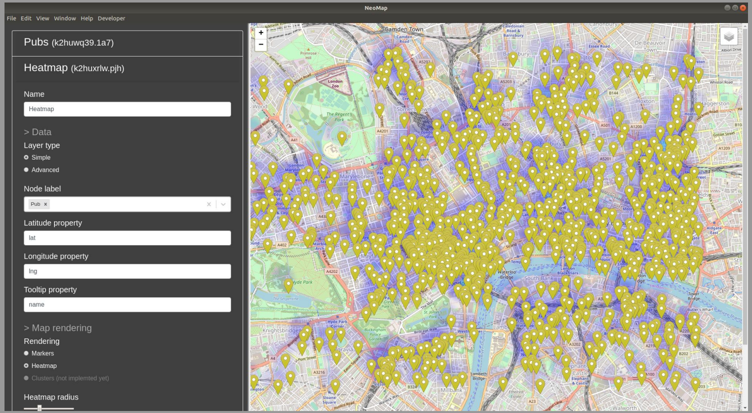
Estelle Scifo

PyConFR 2019, Bordeaux

About me

- Physicist & Data scientist (Luxembourg/World/Remote)
- Graph enthousiast
  - Last project: neomap, visualization tool for Neo4j (written in React)
- Slides and code samples available on my github

  github.com/stellasia/pyconfr19
- Get in touch via twitter/linkedin!



Pubs (k2huwq39.1a7)

Heatmap (k2huxrlw.pjh)

Name: Heatmap

> Data

Layer type: [✓ Simple] Advanced

Node label: Pub

Latitude property: lat

Longitude property: lng

Tooltip property: name

> Map rendering

Rendering: Markers [✓ Heatmap] Clusters (not implemted yet)

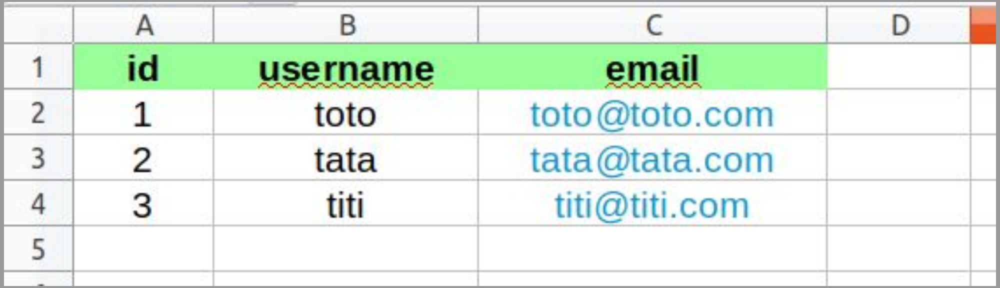
Heatmap radius

twitter: @st3llasia

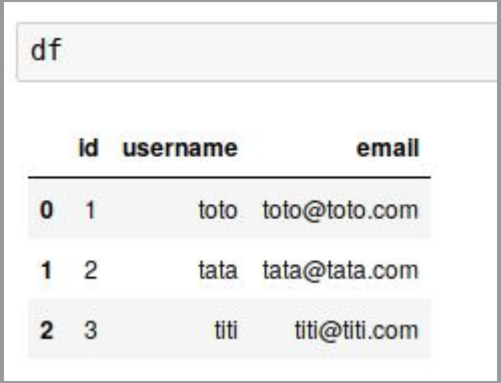
github: stellasia

linkedin: estellescifo

Représentation des données



	A	B	C	D
1	id	username	email	
2	1	toto	toto@toto.com	
3	2	tata	tata@tata.com	
4	3	titi	titi@titi.com	
5				



```
df
```

	id	username	email
0	1	toto	toto@toto.com
1	2	tata	tata@tata.com
2	3	titi	titi@titi.com

```
id,username,email
1,toto,toto@toto.com
2,tata,tata@tata.com
3,titi,titi@titi.com
```

Modèle de données (SQL)

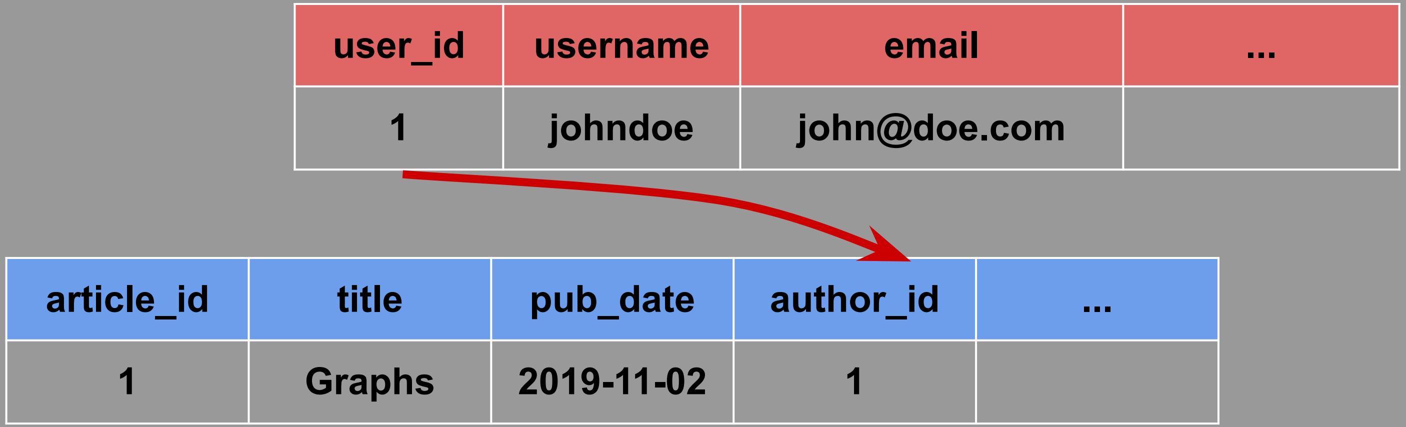
article_id	title	pub_date	...
1	Graphs	2019-11-02	

Modèle de données (SQL)

user_id	username	email	...
1	johndoe	john@doe.com	

article_id	title	pub_date	...
1	Graphs	2019-11-02	

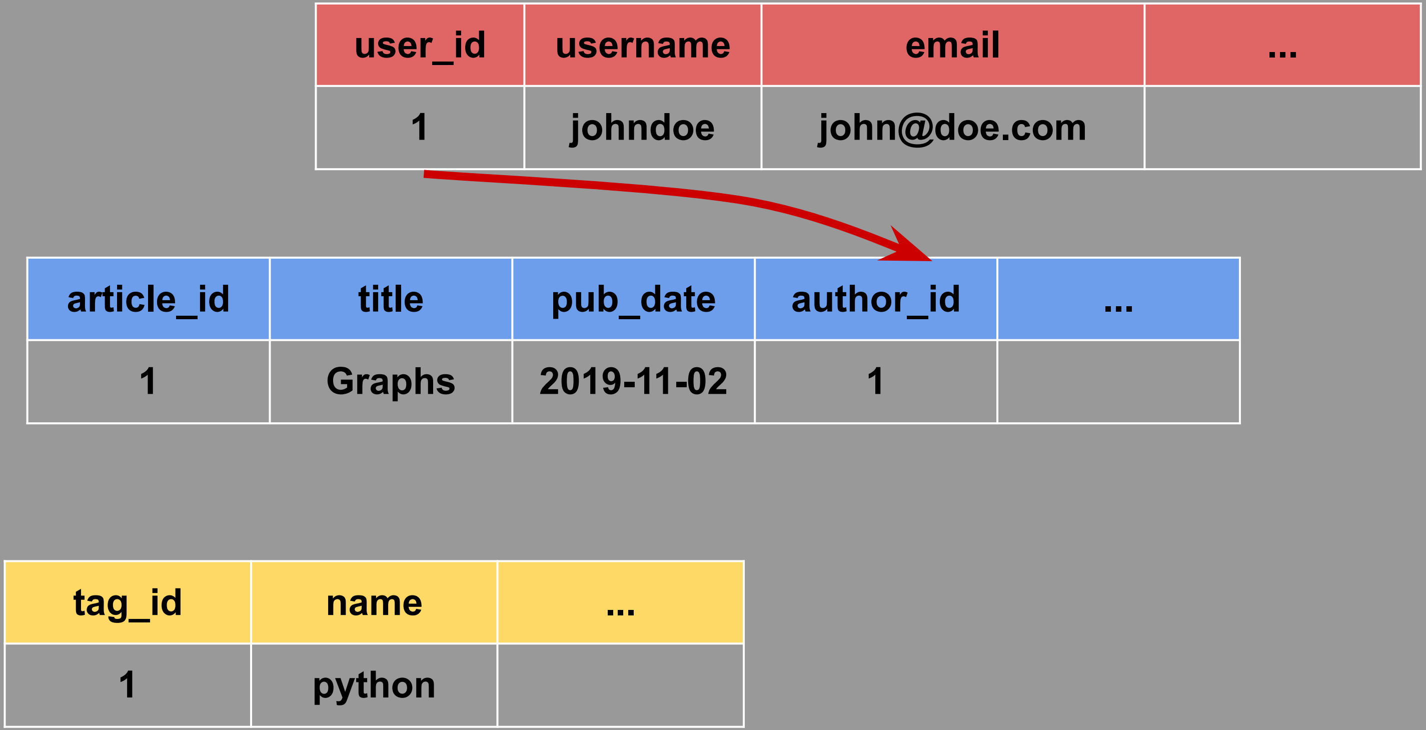
Modèle de données (SQL)



user_id	username	email	...
1	johndoe	john@doe.com	

article_id	title	pub_date	author_id	...
1	Graphs	2019-11-02	1	

Modèle de données (SQL)

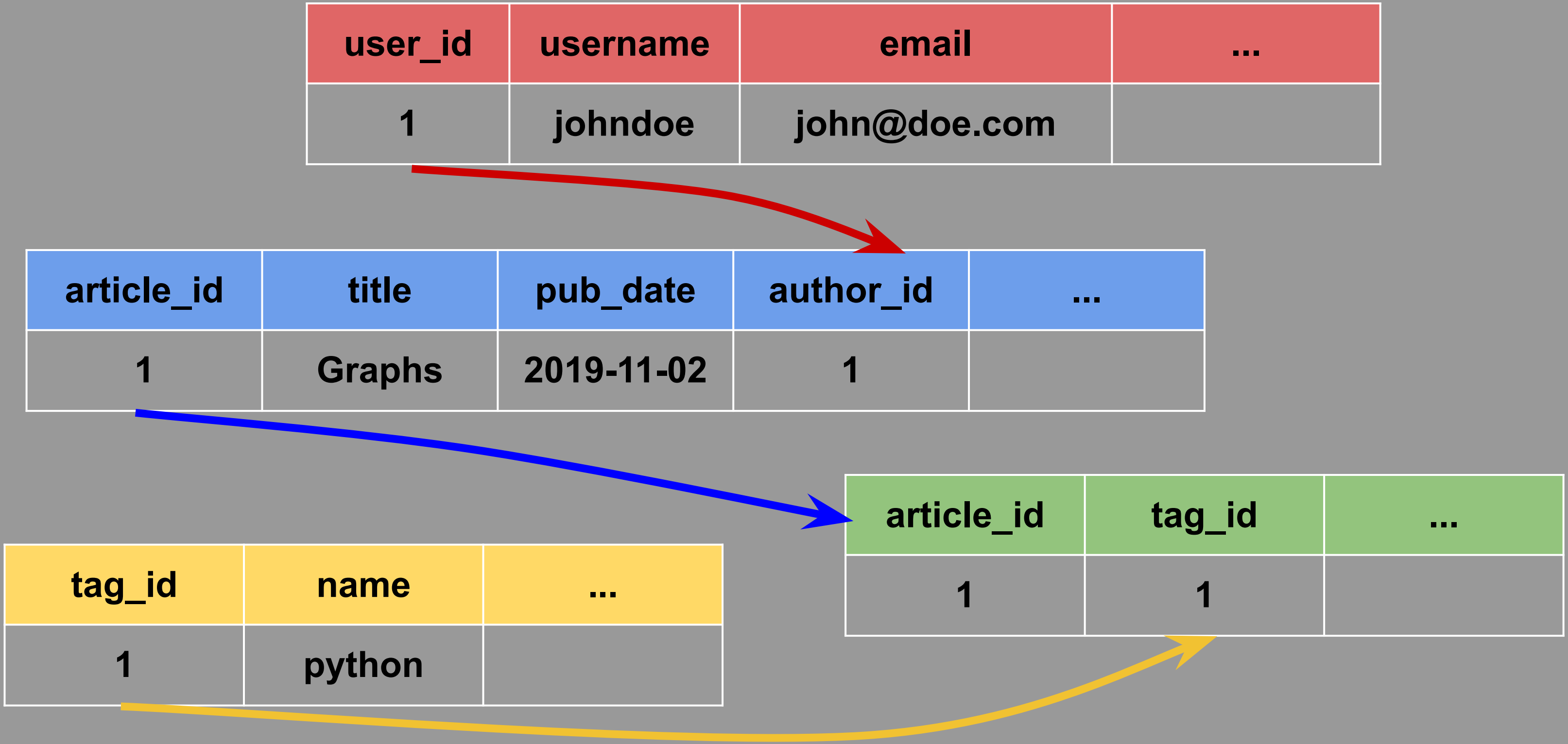


user_id	username	email	...
1	johndoe	john@doe.com	

article_id	title	pub_date	author_id	...
1	Graphs	2019-11-02	1	

tag_id	name	...
1	python	

Modèle de données (SQL)



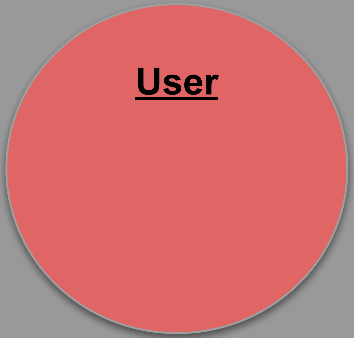
user_id	username	email	...
1	johndoe	john@doe.com	

article_id	title	pub_date	author_id	...
1	Graphs	2019-11-02	1	

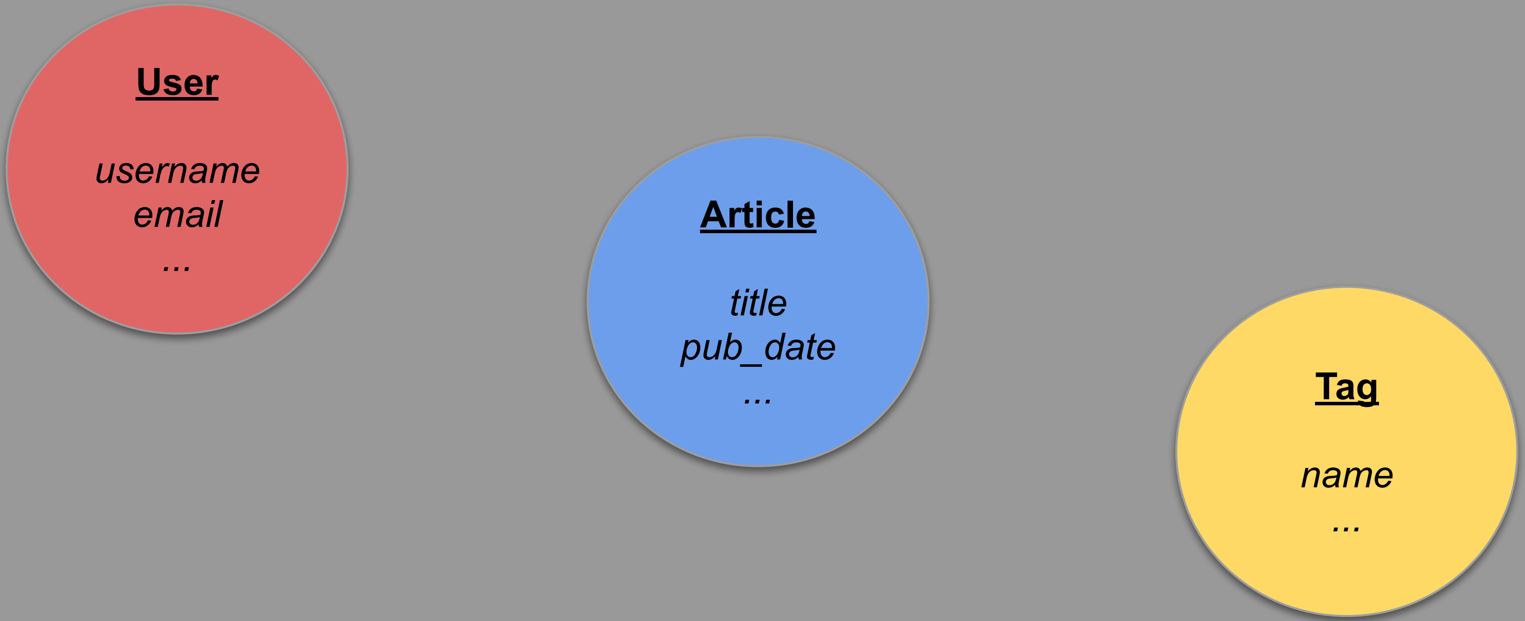
article_id	tag_id	...
1	1	

tag_id	name	...
1	python	

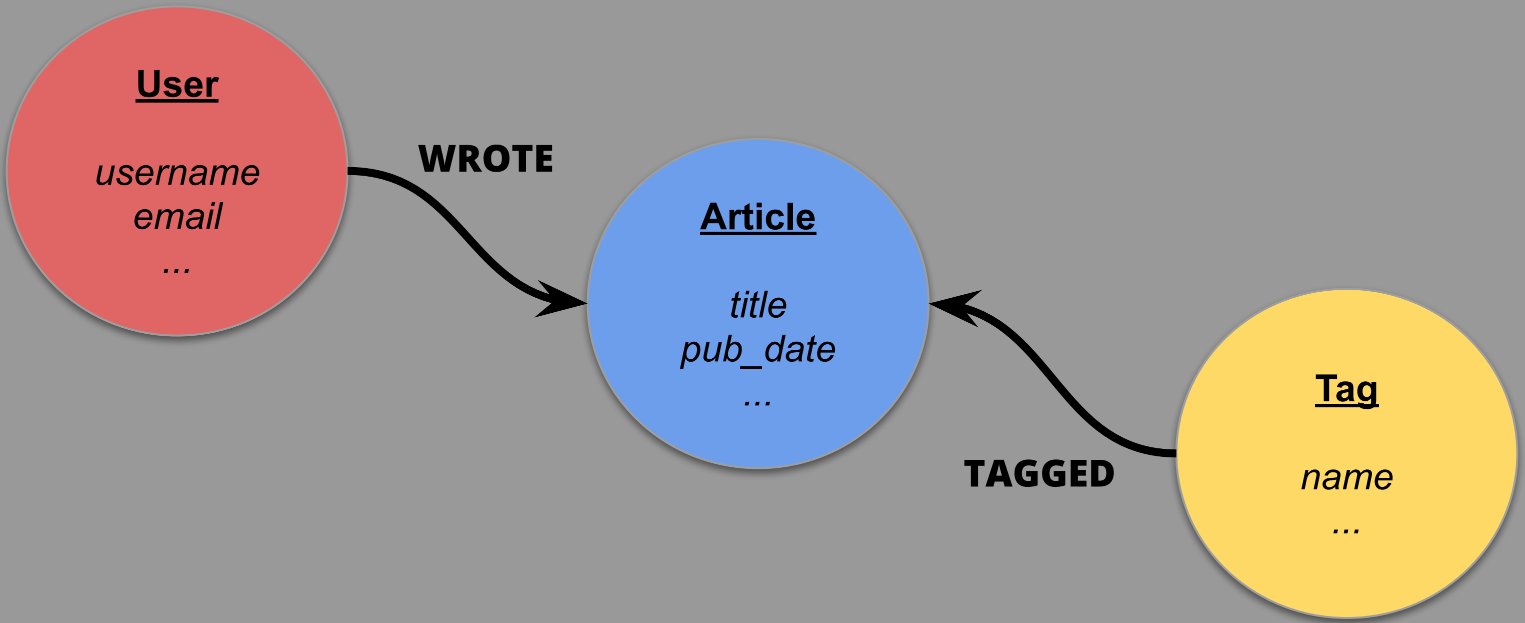
Des tables aux graphes



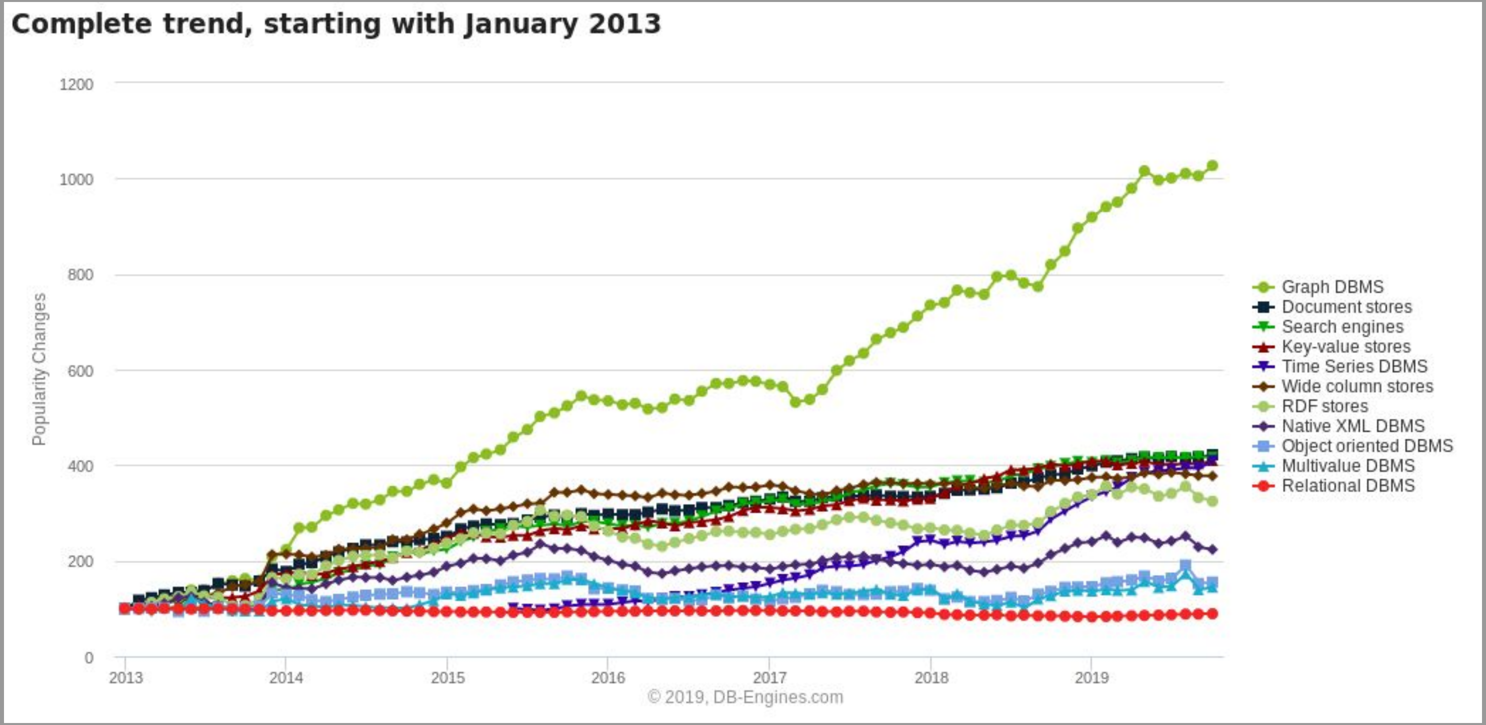
Des tables aux graphes



Des tables aux graphes



Bases de données graphes



Bases de données graphes : exemples

Amazon Neptune

ArangoDB

OrientDB®

JanusGraph

neo4j

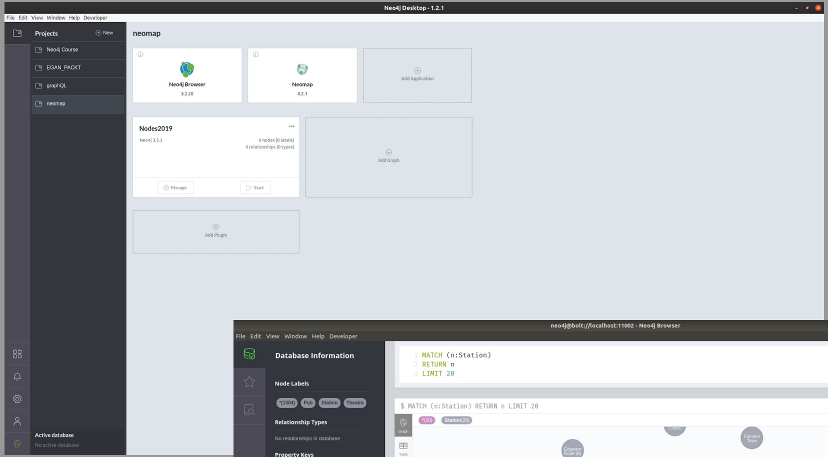
ORACLE Spatial & Graph SIG

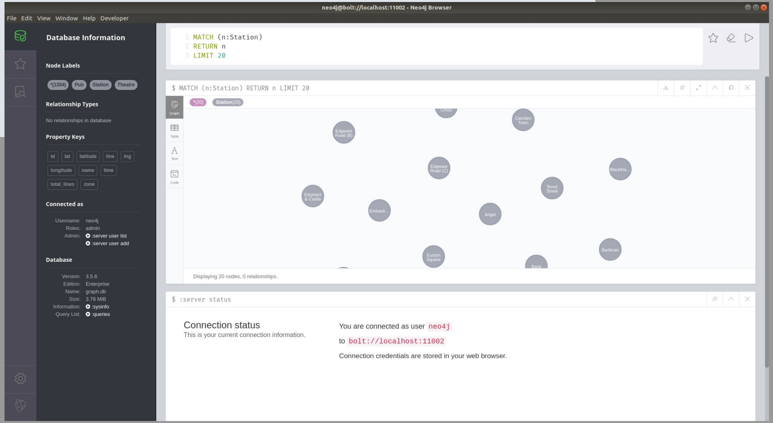
Dgraph

RedisGraph

Neo4j

- Java
- Version gratuite + enterprise
- Neo4j browser & Neo4j desktop
- Langage de requêtes : Cypher
- Driver python officiel





```
MATCH (n:Station)
RETURN n
LIMIT 20
```

No relationships in database

Username: neo4j
Roles: admin
Admin: :server user list
:server user add

Version: 3.5.6
Edition: Enterprise
Name: graph.db
Size: 3.78 MiB
Information: :sysinfo
Query List: :queries

```
$ MATCH (n:Station) RETURN n LIMIT 20
```

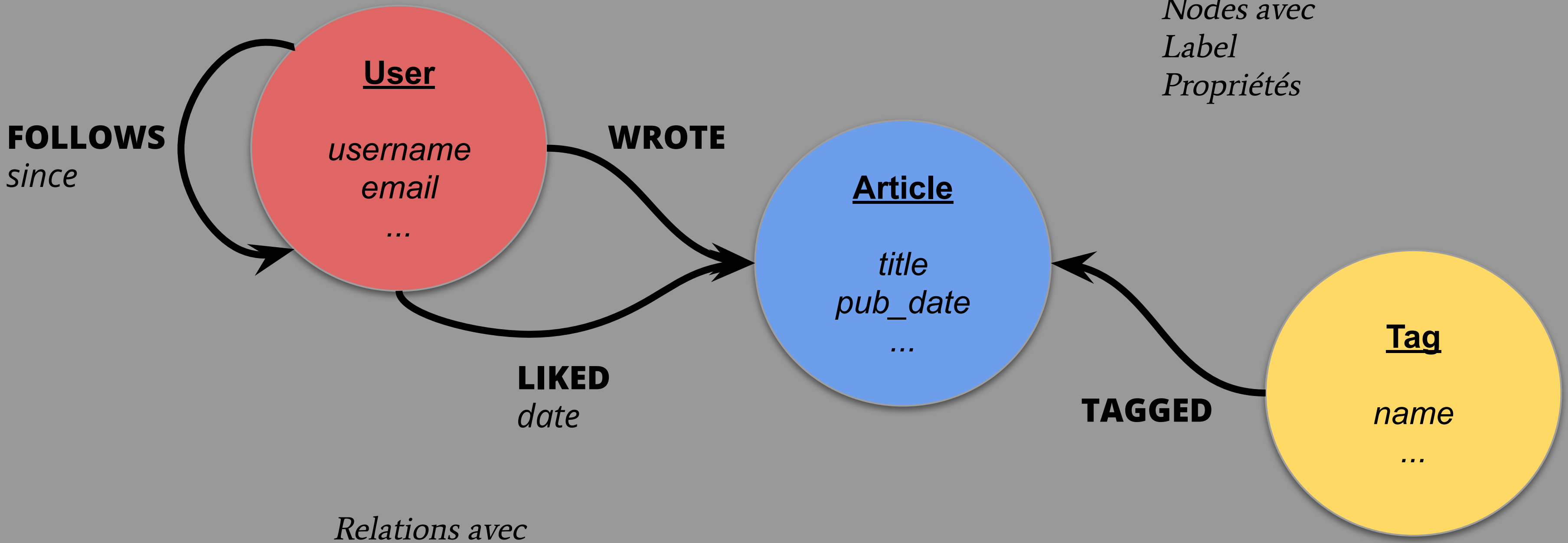
Displaying 20 nodes, 0 relationships.

```
$ :server status
```

Connection status
This is your current connection information.

You are connected as user neo4j
to bolt://localhost:11002
Connection credentials are stored in your web browser.

Modèle de données : nodes, relations & propriétés



Nodes avec
Label
Propriétés

Relations avec
Type
Propriétés

Cypher

SQL ⇔ Cypher

```
INSERT INTO user(name) VALUES ('toto')
```

```
CREATE (:User {name: "toto"})
```

SQL ⇔ Cypher

```
SELECT u.name FROM user AS u WHERE u.id = 1
```

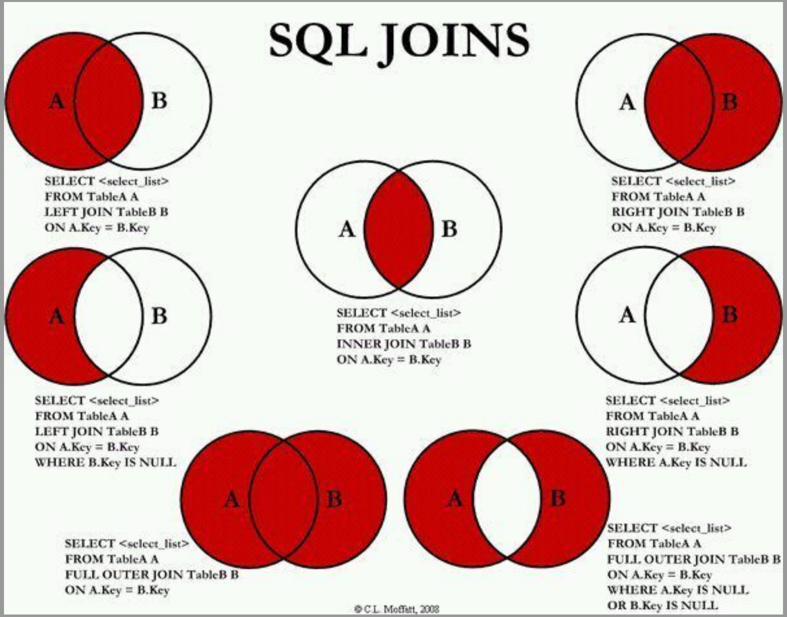
```
MATCH (u:User {id: 1}) RETURN u.name
```

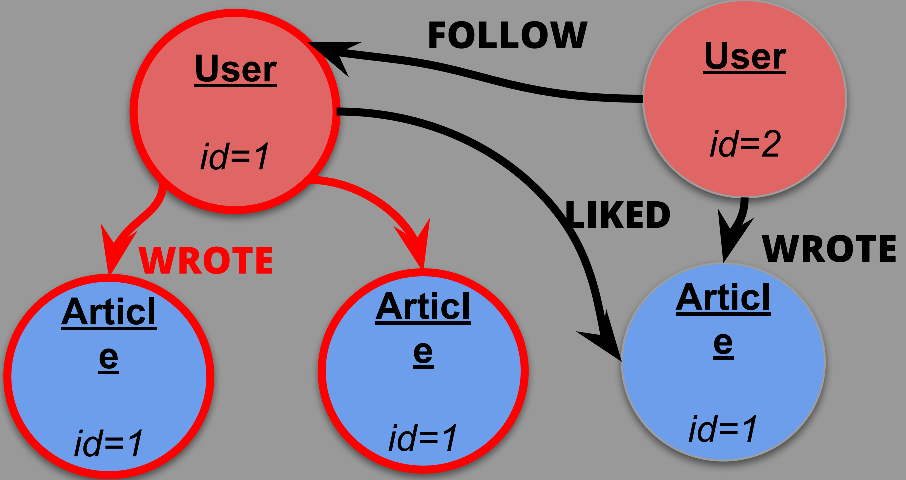
JOIN VS parcours de graphes : pattern matching

```sql
SELECT u.name, a.title
FROM user AS u
JOIN article a ON u.id = a.author_id
WHERE u.id = 1
```

```cypher
MATCH (u:User {id: 1})-[:WROTE]->(a:Article)
RETURN u.name, a.title
```

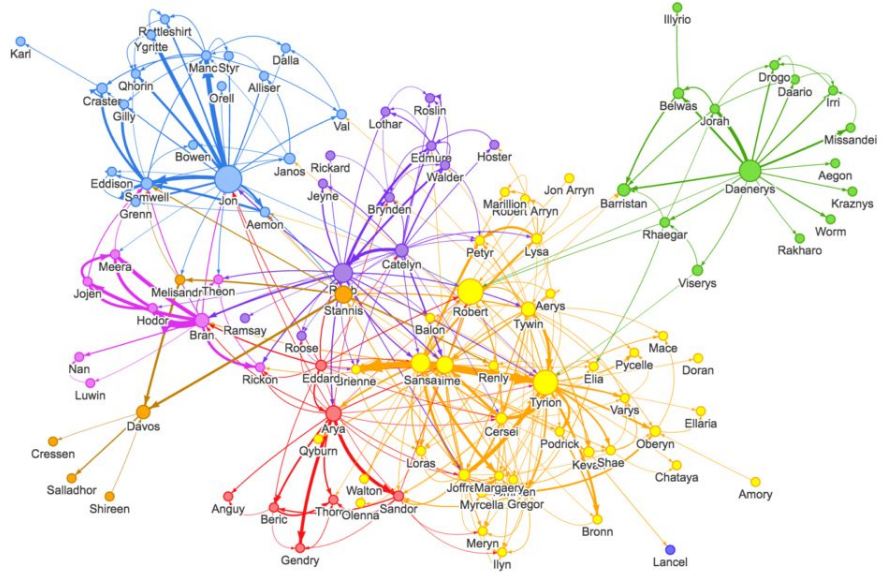
JOIN VS Pattern matching (:User)-[:WROTE]->(:Article)


SQL JOINS
A
B
SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key = B.Key
SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key = B.Key
SELECT <select_list> FROM TableA A INNER JOIN TableB B ON A.Key = B.Key
SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key = B.Key WHERE B.Key IS NULL
SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL
SELECT <select_list> FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key
SELECT <select_list> FROM TableA A FULL OUTER JOIN TableB B ON A.Key = B.Key WHERE A.Key IS NULL OR B.Key IS NULL
© C.L. Moffatt, 2008



Applications

- Knowledge graphs;
- Recommendations;
- Machine Learning :
    - Détection de communauté;
    - Détection d'influenceurs;
    - Prédiction de liens
      (données manquantes ou futurs)
- ...



Recommendations

- Articles aimés par un utilisateur que je suis

```
MATCH  (me:User {username: “estelle”})
       -[:FOLLOW]->(:User)
       -[l:LIKED]->(a:Article)

WHERE NOT (me)-[:LIKED]->(a)
RETURN a.title
ORDER BY l.date DESC
LIMIT 5
```

Import de données

Les utilisateurs

```
id,username,email
1,toto,toto@toto.com
2,tata,tata@tata.com
3,titi,titi@titi.com
```

```
LOAD CSV WITH HEADERS FROM ‘file:///users.csv’ AS row
CREATE (:User { id: row.id,
                username: row.username,
                email: row.email
           }
       )
```

Les articles

```
id,title,author_id
1,Un titre,1
2,Un autre,1
3,Encore un,2
```

```
LOAD CSV WITH HEADERS FROM ‘file:///articles.csv’ AS row
CREATE (a:Article { id: row.id,
                    title: row.title,
                  }
       )
MATCH (u:User {id: row.author_id})
CREATE (u)-[:WROTE]->(a)
```

Driver python

Driver Python

- Driver python officiel :

```
pip install neo4j
```

- Connexion via le protocole `bolt` :

```
from neo4j import GraphDatabase

uri = "bolt://localhost:7687" # default port

driver = GraphDatabase.driver(uri, auth=("neo4j", "password"))
```

Driver Python

```
cypher_query = """
        MATCH (u:User {id: {user_id}})-[:WROTE]->(a:Article)
        RETURN a.title
    """

with driver.session() as session:
    result = session.run(
            cypher_query,
            parameters={user_id:1}
    )
```

Object Graph Mapper [OGM]

neomodel

py2neo

Object Graph Mapper [OGM]

neomodel

```
from neomodel import (
       StructuredNode, UniqueIdProperty, StringProperty, DateProperty
)

class User(StructuredNode):
    uid = UniqueIdProperty()
    username = StringProperty(unique_index=True)
    email = StringProperty()

class Article(StructuredNode):
    uid = UniqueIdProperty()
    title = StringProperty()
    pub_date = DateProperty()
```

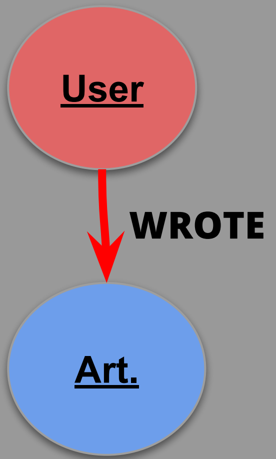
Object Graph Mapper [OGM]

neomodel

```
u = User(username=”toto”, email=”toto@toto.com”)
u.save()


u = User.nodes.get(email=”toto@toto.com”)
print(u.username)
```

Relations

```
from neomodel import RelationshipTo, RelationshipFrom

class User(StructuredNode):
    ...
    articles = RelationshipTo('Acticle', 'WROTE')

class Article(StructuredNode):
    ...
    author = RelationshipFrom('User', 'WROTE')
```



```
u = User.nodes.get(email=”toto@toto.com”)
print(u.articles.all())
```

Recommendations

```
class User(StructuredNode):
    ...

    def reco(self):
        results, columns = self.cypher(cypher_query)
        return [Article(**row[0]) for row in results]
```

```
u = User.nodes.get(email=”toto@toto.com”)
print(u.reco())
```

Pour aller plus loin

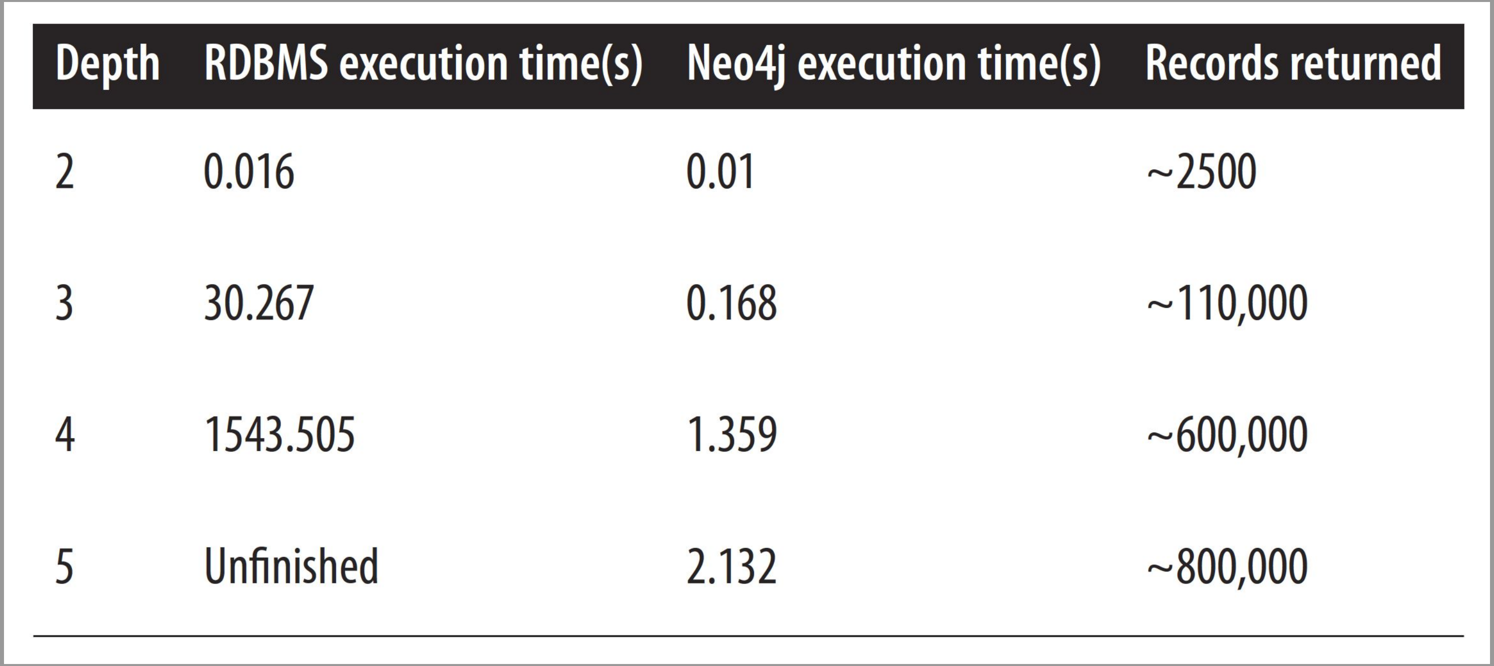
- GraphQL APIs
  - Neo4j graphQL plugin

- Algorithmes de graphes
  pour apprendre de vos données graphes :
  - Graph algorithms library, a Neo4j plugin

```
query {
  User(username: ”toto”) {
    email
    articles {
        title
    }
  }
}
```

twitter: @st3llasia
github: stellasia
linkedin:estellescifo

Performances



Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000