

INTRODUCTION À NEO4J

(me:Human {name: 'Michel'})-[:PRESENTS]->(neo4j:Engine {name: 'Neo4j'})

PRÉSENTATIONS

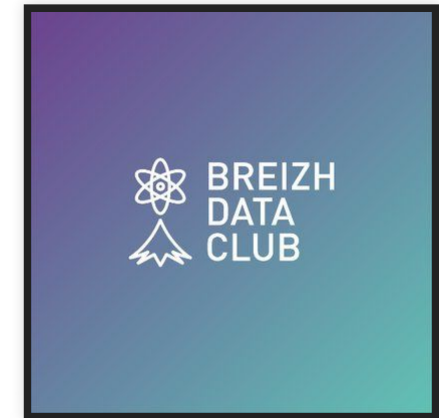
Michel Caradec (mcaradec@hotmail.com)



www.cegid.com



data-bzh.fr



breizhdataclub.org

Project Manager, Software/Data Engineer

AGENDA

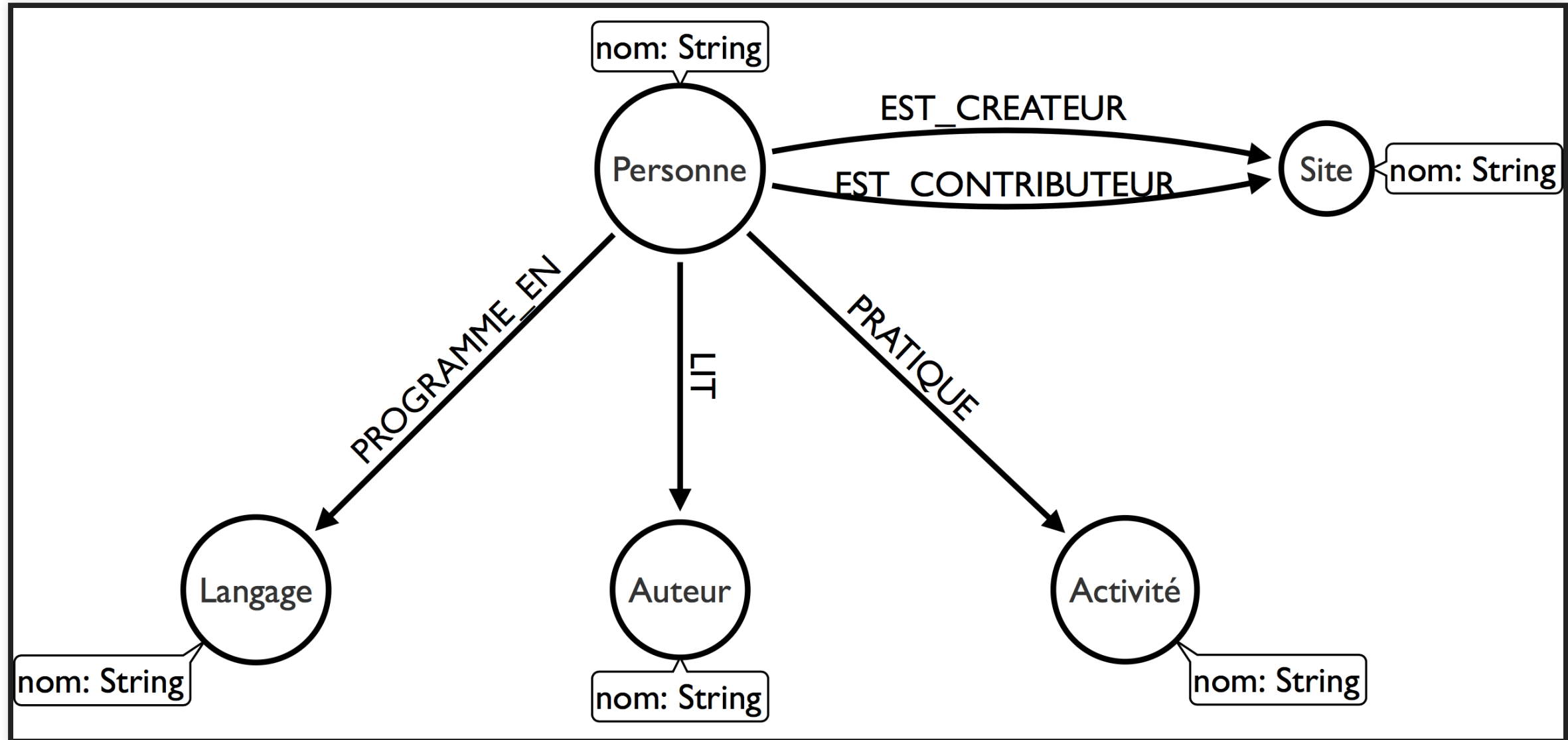
- Etude de cas.
- Cypher Query Language.
- Utilisation avancée de Cypher.
- Pourquoi Neo4j ?
- Cas d'usage.

ETUDE DE CAS

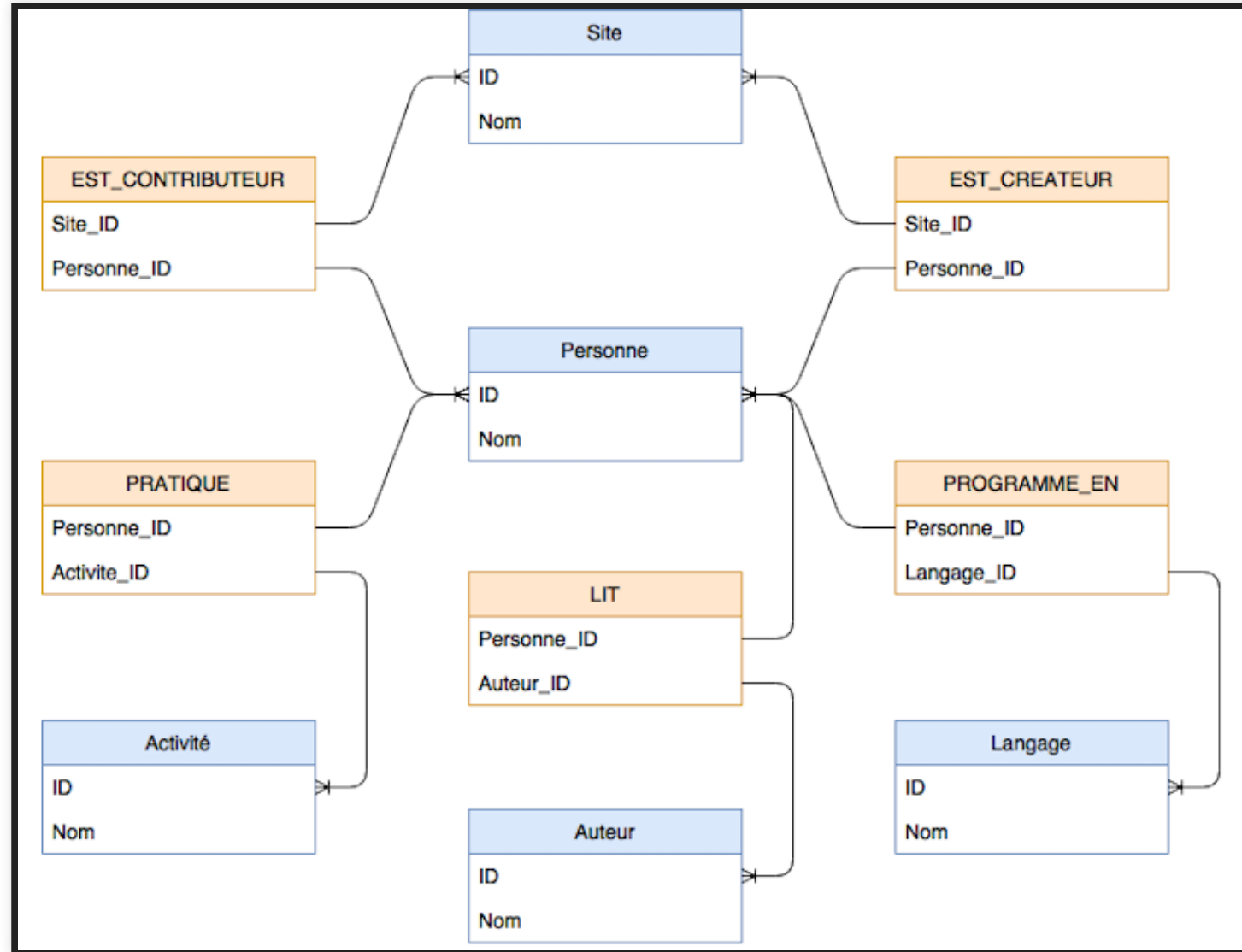
EXEMPLE DE RÉSEAU : DATA-BZH

- Colin **EST CREATEUR** de Data-Bzh.
- Colin, Michel, Tristan **SONT CONTRIBUTEURS** sur Data-Bzh.
- Colin, Michel, Tristan **PROGRAMMENT EN R**.
- Colin **LIT** Proust.
- Michel **PRATIQUE** le Trail.

MÉTA-MODÈLE



MODÉLISATION SQL



MODÉLISATION SQL

```
CREATE TABLE Personne(ID integer PRIMARY KEY, Nom text);
CREATE TABLE Site(ID integer PRIMARY KEY, Nom text);
CREATE TABLE Activite(ID integer PRIMARY KEY, Nom text);
CREATE TABLE Auteur(ID integer PRIMARY KEY, Nom text);
CREATE TABLE Langage(ID integer PRIMARY KEY, Nom text);

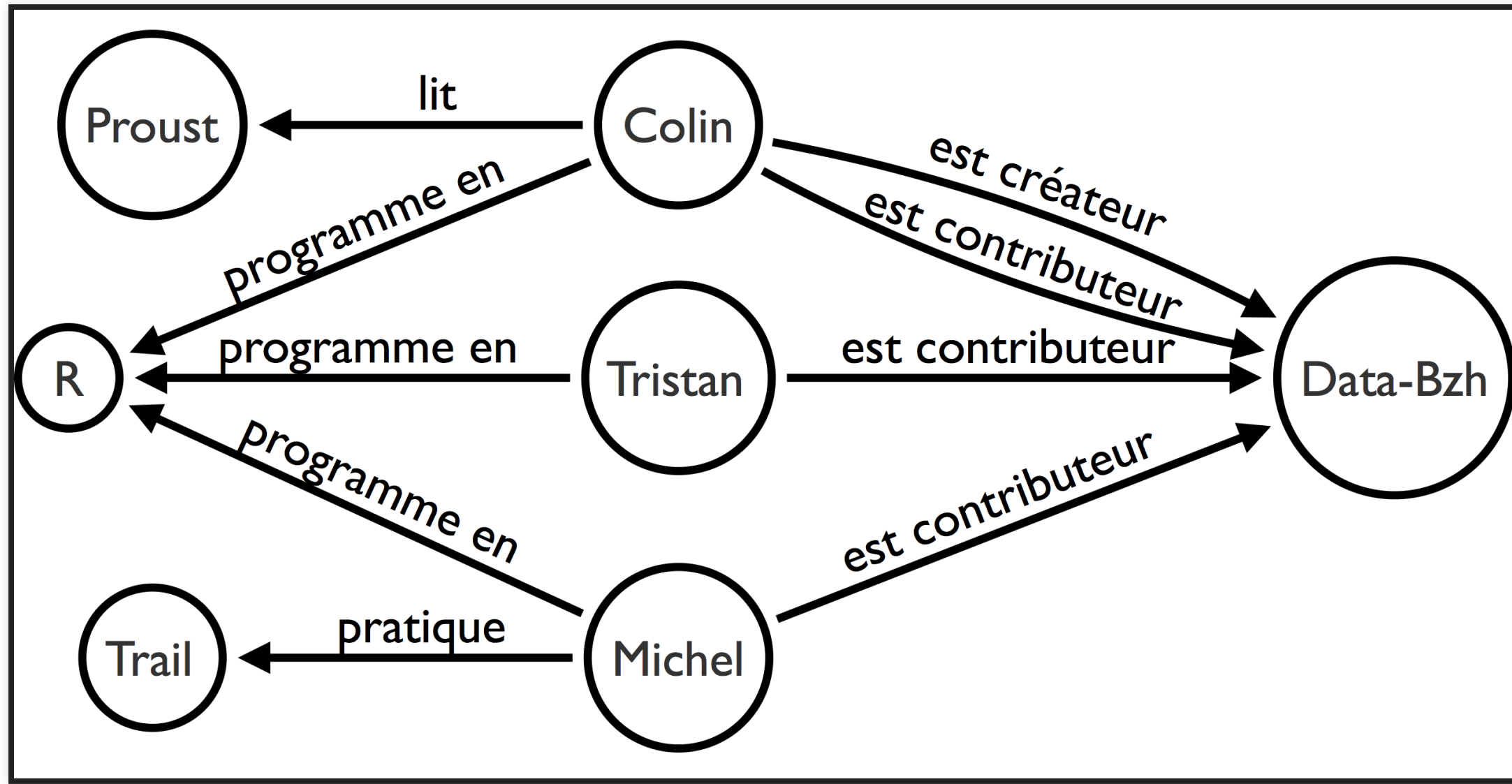
CREATE TABLE EST_CREATEUR (
    Personne_ID integer REFERENCES Personne(ID), Site_ID integer REFERENCES Site(ID));
CREATE TABLE EST_CONTRIBUTEUR (
    Site_ID integer REFERENCES Site(ID), Personne_ID integer REFERENCES Personne(ID));
CREATE TABLE PROGRAMME_EN (
    Personne_ID integer REFERENCES Personne(ID), Langage_ID integer REFERENCES Langage);
CREATE TABLE LIT (
    Personne_ID integer REFERENCES Personne(ID), Auteur_ID integer REFERENCES Auteur(ID));
CREATE TABLE PRATIQUE (
```


ALIMENTATION SQL

```
INSERT INTO Personne (ID, Nom) VALUES (1, 'Colin'), (2, 'Michel'), (3, 'Tristan')
INSERT INTO Site (ID, Nom) VALUES (1, 'Data-Bzh')
INSERT INTO Activite (ID, Nom) VALUES (1, 'Trail')
INSERT INTO Auteur (ID, Nom) VALUES (1, 'Proust')
INSERT INTO Langage (ID, Nom) VALUES (1, 'R')

INSERT INTO EST_CREATEUR (Site_ID, Personne_ID) VALUES (1, 1)
INSERT INTO EST_CONTRIBUTEUR (Personne_ID, Site_ID) VALUES (1, 1), (2, 1), (3, 1)
INSERT INTO PROGAMME_EN (Personne_ID, Langage_ID) VALUES (1, 1), (2, 1), (3, 1)
INSERT INTO LIT (Personne_ID, Auteur_ID) VALUES (1, 1)
INSERT INTO PRATIQUE (Personne_ID, Activite_ID) VALUES (2, 1)
```

Modélisation plus naturelle ?



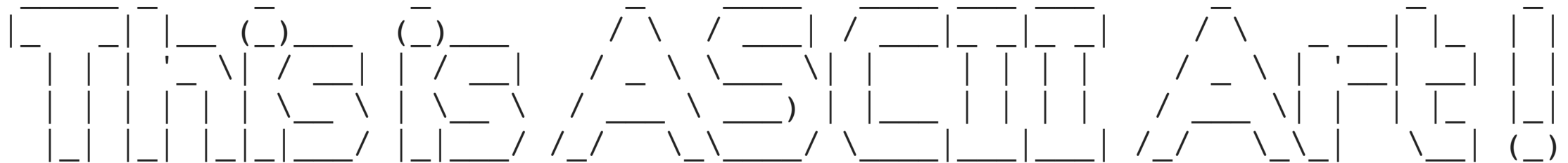
NEO4J



- Société **Neo4j, Inc.**
- Base de données **orientée graphes, NoSQL.**
- Première version en **février 2000.**
- Deux éditions :
 - **Neo4j Community.**
 - **Neo4j Entreprise.**

CYPHER QUERY LANGUAGE

Modéliser et interroger en **ASCII Art**.



THIS IS ASCII Art!

FORMALISME

- Phrase : **Sujet** - **Verbe** - **Objet**.
- Chemin : **noeud départ** - **relation** - **noeud arrivée**.
 - **Noeuds** encadrés par des parenthèses ().
 - **Relations** encadrées par des crochets [].
 - Connexions matérialisées par des flèches -->.

(noeud départ)-[:RELATION]->(noeud arrivée)

Ex.: (Michel)-[:CONTRIBUE]->(Data-Bzh)

FORMALISME

- **Propriétés** sur les noeuds et relations.
- **Labels** sur les noeuds.
- Les relations sont **-dirigées->**.
- Les **variables** identifient les noeuds et relations dans la requête.

```
(me:Person {name: 'Michel', interest: 'Data'})-[:CONTRIBUE {since: 2016}]->(data_bzh:Site {url: 'www.data-bzh.fr'})
```

CRÉATION DES NOEUDS

```
CREATE
(colin:Personne {nom: 'Colin'}),
(michel:Personne {nom: 'Michel'}),
(tristan:Personne {nom: 'Tristan'}),
(data_bzh:Site {nom: 'Data-Bzh'}),
(trail:Activite {nom: 'Trail'}),
(proust:Auteur {nom: 'Proust'}),
(r:Langage {nom: 'R'})
```

CRÉATION DES RELATIONS

CREATE

```
(colin)-[:EST_CREATEUR]->(data_bzh),  
(colin)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(michel)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(tristan)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(colin)-[:PROGRAMME_EN]->(r),  
(michel)-[:PROGRAMME_EN]->(r),  
(tristan)-[:PROGRAMME_EN]->(r),  
(colin)-[:LIT]->(proust),  
(michel)-[:PRATIQUE]->(trail)
```


COMMANDE CYPHER COMPLÈTE

CREATE

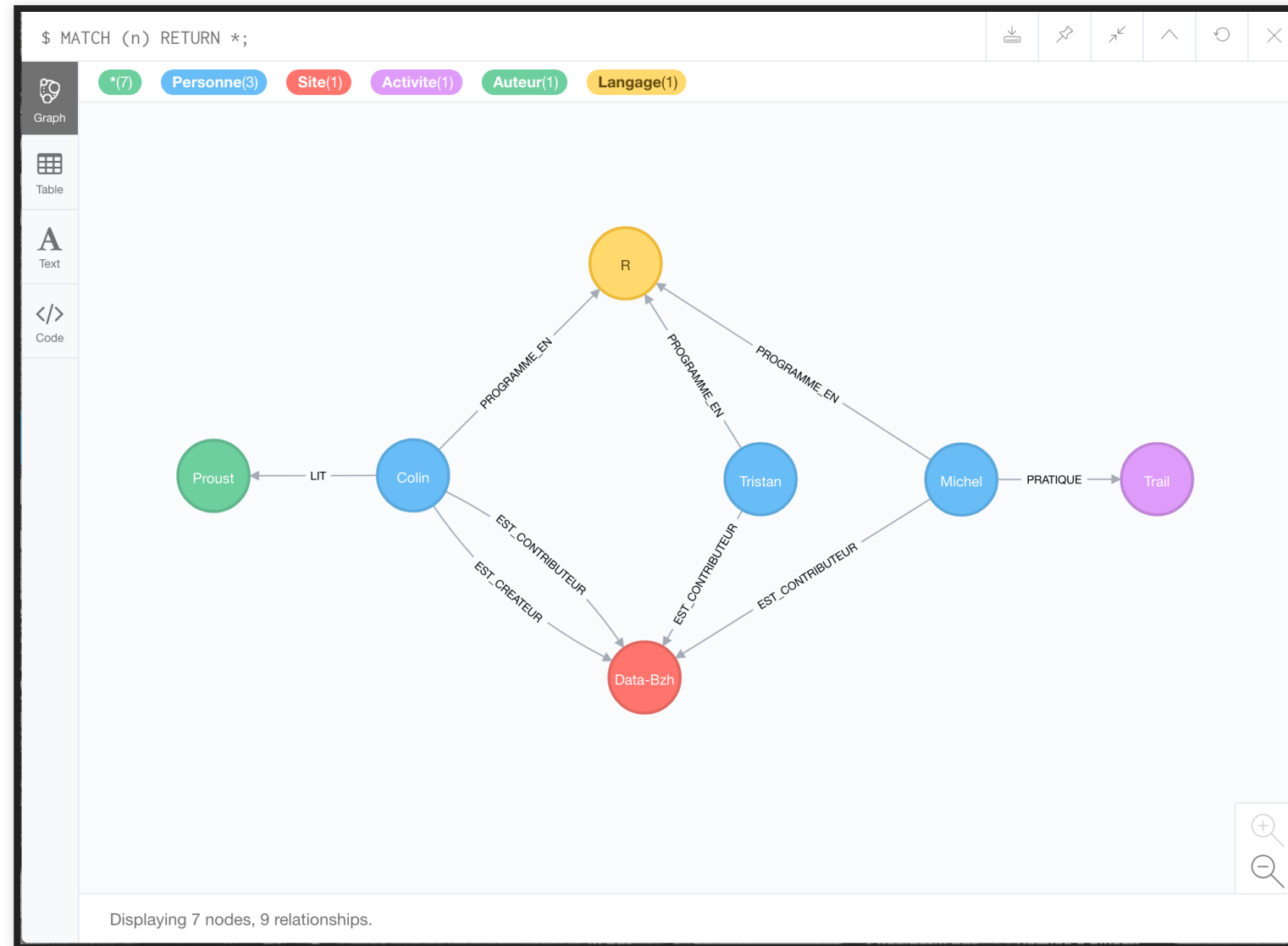
```
(colin:Personne {nom: 'Colin'}),  
(michel:Personne {nom: 'Michel'}),  
(tristan:Personne {nom: 'Tristan'}),  
(data_bzh:Site {nom: 'Data-Bzh'}),  
(trail:Activite {nom: 'Trail'}),  
(proust:Auteur {nom: 'Proust'}),  
(r:Langage {nom: 'R'}),  
(colin)-[:EST_CREATEUR]->(data_bzh),  
(colin)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(michel)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(tristan)-[:EST_CONTRIBUTEUR]->(data_bzh),  
(colin)-[:PROGRAMME_EN]->(r),  
(michel)-[:PROGRAMME_EN]->(r),  
(tristan)-[:PROGRAMME_EN]->(r),  
(colin)-[:LIT]->(proust),  
(michel)-[:PRATIQUE]->(trail);
```

RECHERCHE SIMPLE

- **MATCH** : recherche de motifs (patterns) dans le graphe.
- **RETURN** : retourne un résultat :
 - Sous-graphe.
 - Rowset.

```
MATCH (n)  
RETURN *;
```

RECHERCHE SIMPLE



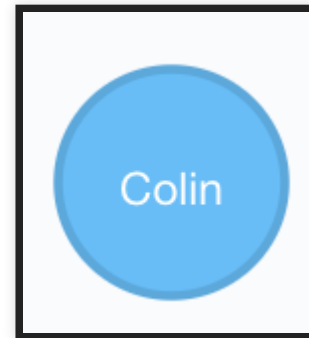
RECHERCHE PAR PATTERN

Recherche d'une **personne**.

```
MATCH (n:Personne {nom: 'Colin'})  
RETURN n;
```

OU

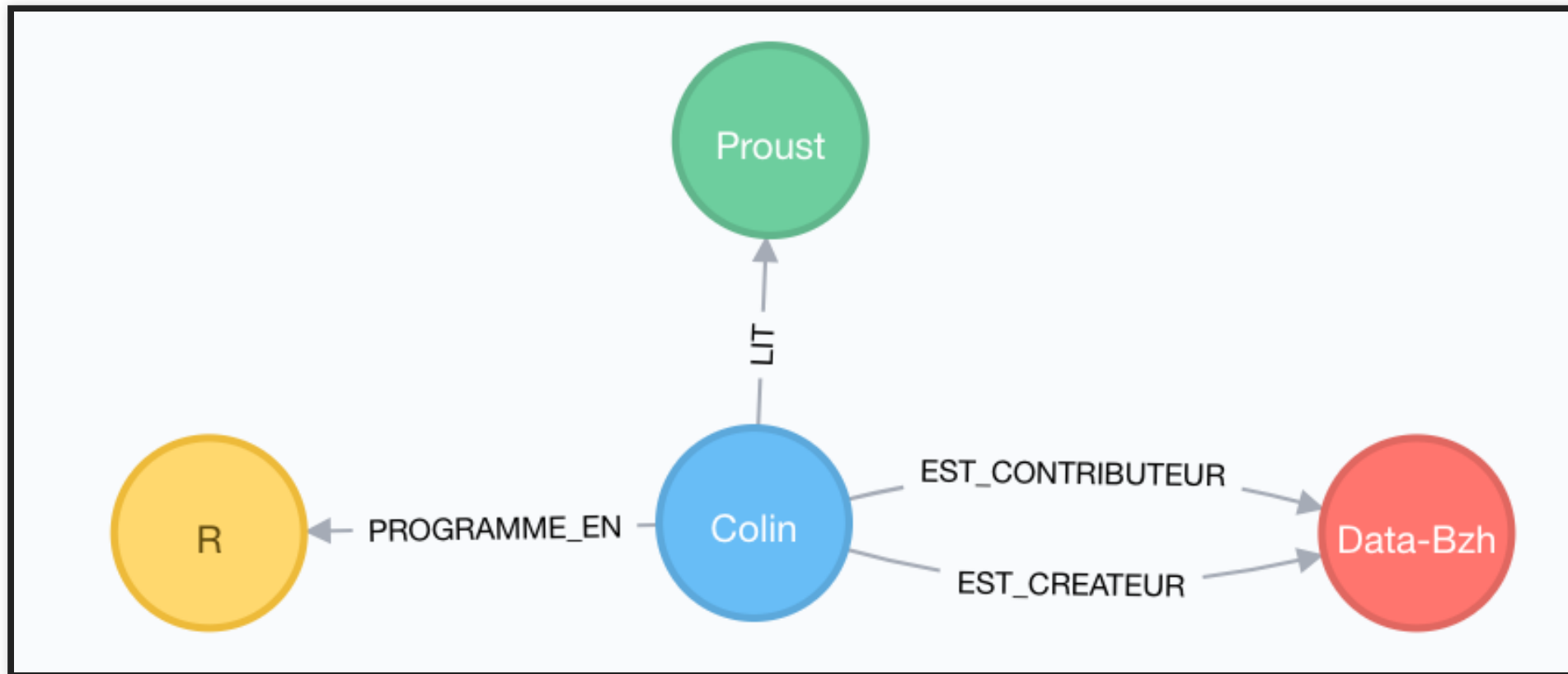
```
MATCH (n:Personne)  
WHERE n.nom = 'Colin'  
RETURN n;
```



RECHERCHE PAR PATTERN

Recherche d'une personne et de ses relations.

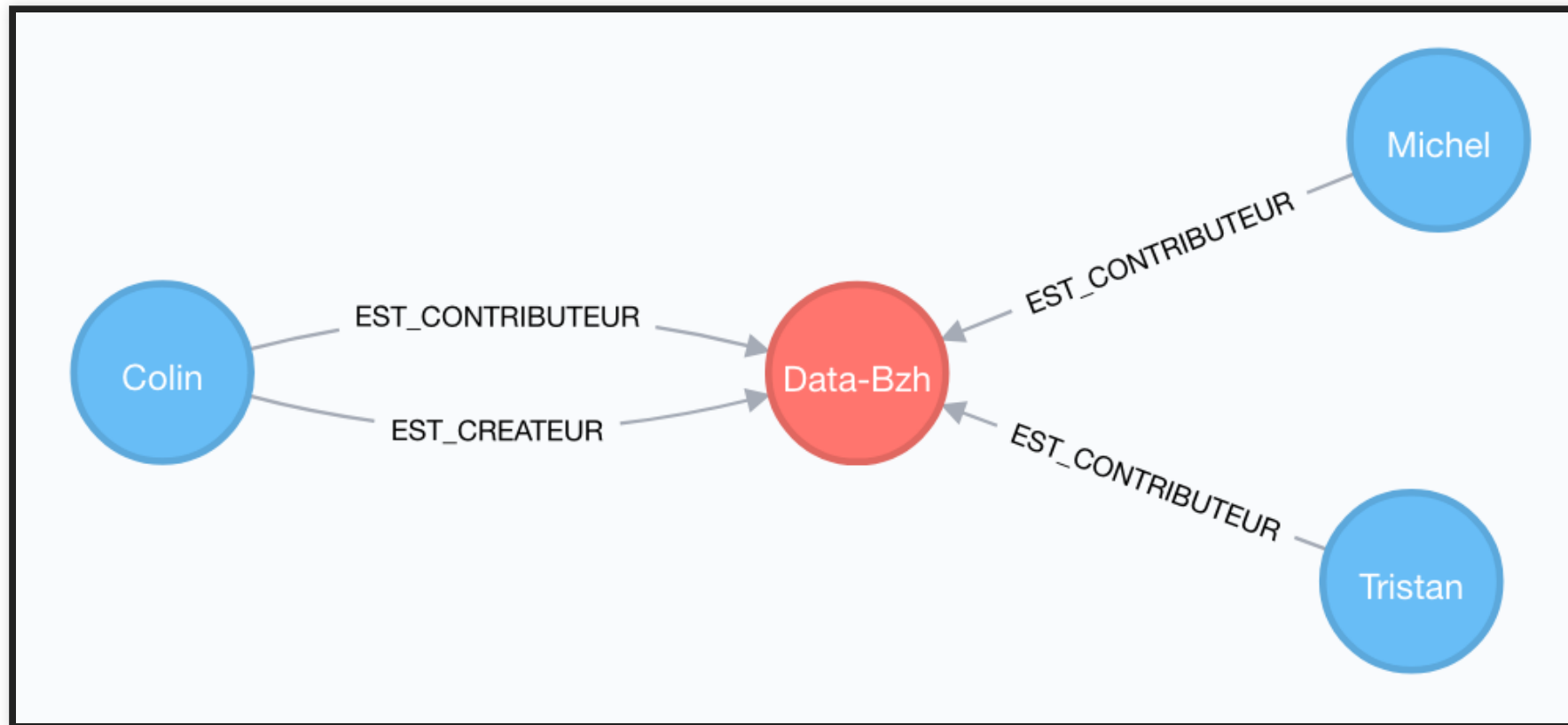
```
MATCH (c:Personne {nom: 'Colin'})-[]-(o)
RETURN *;
```



RECHERCHE PAR PATTERN

Recherche des contributeurs d'un site.

```
MATCH (c)-[:EST_CONTRIBUTEUR]->(s:Site {nom: 'Data-Bzh'})  
RETURN *;
```



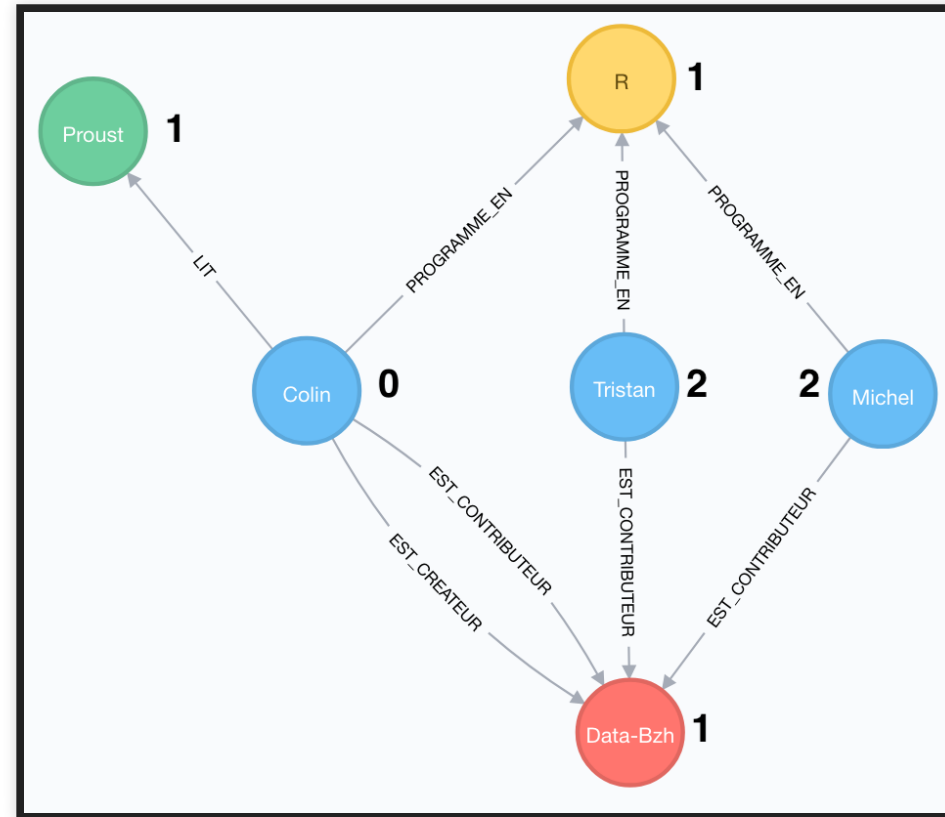
POSSIBLE AUSSI EN SQL

```
SELECT Nom
FROM Personne, EST_CONTRIBUTEUR, Site
WHERE Personne.ID = EST_CONTRIBUTEUR.Personne_ID
      AND EST_CONTRIBUTEUR.Site_ID = Site.ID
      AND Site.Nom = 'Data-Bzh'
```

Quel est l'intérêt de l'approche graphe ?

RECHERCHE MULTI-NIVEAUX

```
MATCH (n:Personne {nom: 'Colin'})-[*..2]-(o)  
RETURN *;
```

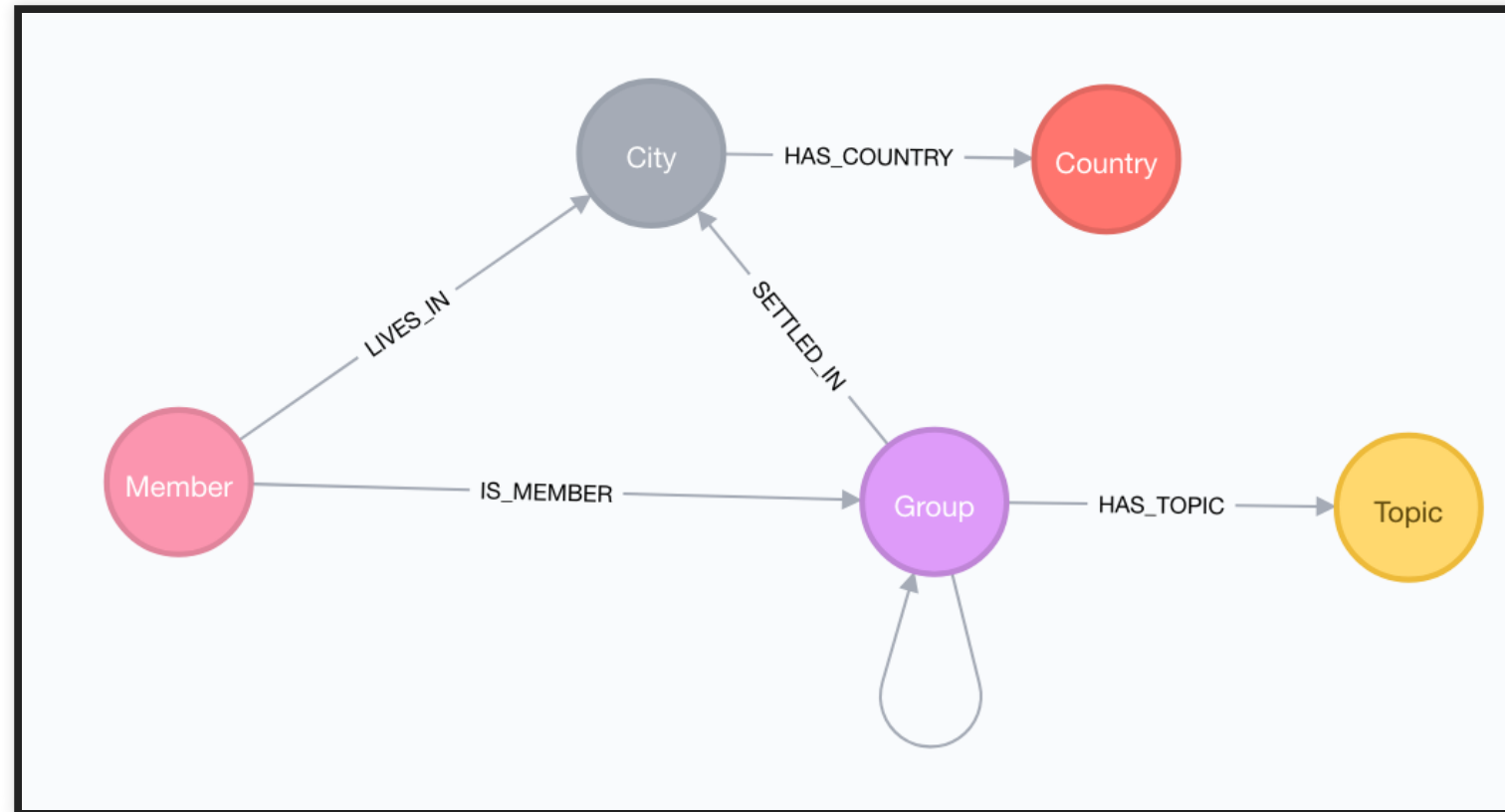


Beaucoup plus compliqué en SQL (sans parler des performances) !

UTILISATION AVANCÉE DE CYPHER

LES MEETUP RENNAIS

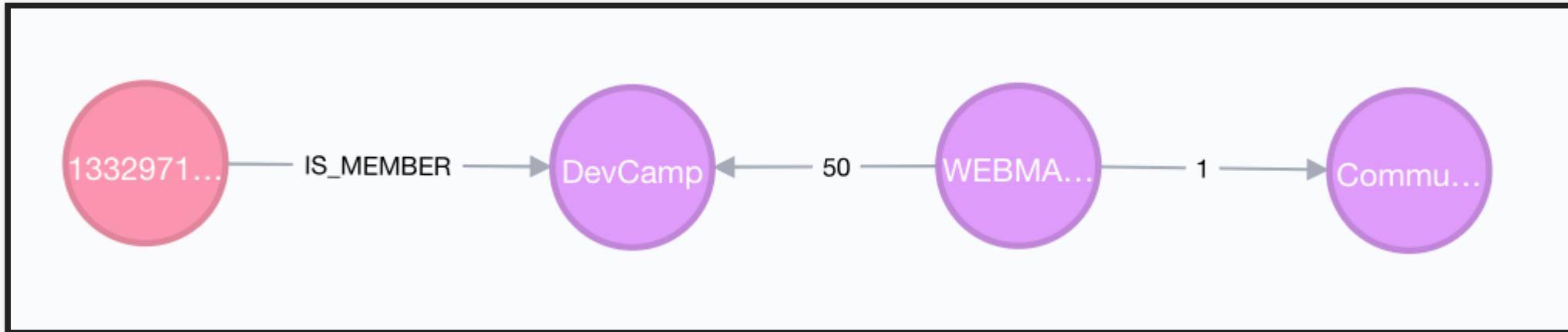
```
CALL db.schema( );
```



Données collectées le 24/09/2017.

CHEMIN LE PLUS COURT

```
WITH
  133297112 AS Me, // Michel
  "Communauté vidéo iPhone Bretagne" AS TargetGroupName,
  ["Meetup La French Tech Rennes St Malo"] AS ExcludeGroupNames
MATCH (me:Member {id: Me})-[:IS_MEMBER]->(g_me:Group)
WHERE NOT g_me.name IN ExcludeGroupNames
MATCH path=shortestPath((g_me:Group)-[:HAS_COMMON_MEMBERS*..3]-(g_target:Group {name
RETURN me, path, length(path) AS RelCount
ORDER BY RelCount ASC
LIMIT 1;
```



FILTRE COLLABORATIF

```
WITH
    // Identifiant de la personne pour laquelle effectuer une recommandation
    133297112 AS Me // Michel
    // 1. Groupes (g_reco) des membres (other) de mes (me) groupes (g_me)...
MATCH (me:Member {id: Me})-[:IS_MEMBER]->(g_me:Group)<-[:IS_MEMBER]-(other:Member)-[
WHERE
    // ... dont je ne suis pas membre
    NOT (me)-[:IS_MEMBER]->(g_reco)
WITH
    // Groupes dont je ne suis pas membre
    DISTINCT g_reco AS RecommendedGroups,
    // Nombre de membres en commun avec ce groupe
    COUNT(DISTINCT other) AS CommonMemberCount
    ORDER BY CommonMemberCount DESC
// 2. Noms des groupes recommandés, avec le nombre de membres en commun (tableau)
```

FILTRE COLLABORATIF

Groupes	NbMembresCommuns
"UX Rennes"	670
"Le Shift"	669
"Agile Rennes"	636
"La Cordée Rennes - partage, événements et bonne humeur"	634
"RennesJS"	545

PAGE RANK

- Détermine la **popularité** d'une page web.
- Rapport entre les **liens sortants** (mentions) et les **liens entrants** (citations).
- Algorithme **itératif**.

$$PageRank(A) = \frac{(1-d)}{n} + d \sum_{i=1}^n \frac{PageRank(T_i)}{OutLinks(T_i)}$$

PAGE RANK

Utilisation de l'extension **Neo4j Graph Algorithms**.

```
CALL algo.pageRank.stream('Group', 'MEMBERS_ALSO_GO', {iterations: 30})
YIELD node, score
RETURN
  node.name AS Groupe,
  score AS Score
ORDER BY score DESC
LIMIT 5;
```

PAGE RANK

Groupe	Score
"Meetup La French Tech Rennes St Malo"	1.08
"WEBMARKETING Rennes"	1.08
"La Cordée Rennes - partage, événements et bonne humeur"	1.01
"UX Rennes"	1.01
"Le Shift"	1.01

CLUSTERING

Détection de communautés selon la méthode de **Louvain**.

```
CALL algo.louvain(  
  'MATCH (g:Group) RETURN id(g) AS id',  
  'MATCH (g1:Group)-[rel:HAS_COMMON_MEMBERS]-(g2:Group) '  
+ 'WHERE rel.weight > 100 RETURN id(g1) as source, id(g2) as target',  
  {graph: "cypher", writeProperty: "community"}  
);
```

Persistance dans chaque noeud dans la propriété **community**.

CLUSTERING

```
// Communauté avec le plus grand nombre de membres.  
MATCH (g:Group)  
WITH g.community AS Community, COUNT(g) AS Count  
ORDER BY Count DESC  
WITH head(collect(Community)) AS TopCommunity  
// Membres de la communauté.  
MATCH (g:Group {community: TopCommunity})  
RETURN TopCommunity, collect(g.name) AS Groupes;
```

Exploitation de la propriété **community**.

TopCommunity Groupes

101

"Meetup E-Learning Rennes", "Le Hubzh", "ZikLab Rennes", "Meetup Codéveloppement professionnel et managérial Rennes", "Dolibarr Rennes et alentours", "Économie Circulaire Rennes", "Prosperer", "IMIE'TING", "JCE Rennes", "Le bien-être au service des professionnels", "Meetup Photographie Pro (Rennes)"

POURQUOI NEO4J ?

- **Expressivité** du modèle, simplicité d'utilisation.
- **Evolutivité** du modèle (NoSQL).
- Stockage orienté **relations** (optimisé pour les graphes).
- **Cypher Query Language**.
- **Extensibilité** (API).

CAS D'USAGE

- Réseaux sociaux.
- Recommandation.
- Logistique, géo-spatial.
- Master Data Management.
- Gestion ressources IT.
- Détection de fraudes.
 - [Panama Papers.](#)
 - [Paradise Papers.](#)

POUR ALLER PLUS LOIN AVEC NEO4J

- Cypher : <https://neo4j.com/developer/cypher/>.
 - Awesome Procedures On Cypher (APOC) : <https://github.com/neo4j-contrib/neo4j-apoc-procedures>.
 - Graph Algorithm : <https://github.com/neo4j-contrib/neo4j-graph-algorithms>.
- Open Cypher : <https://www.opencypher.org/>.

POUR ALLER PLUS LOIN AVEC NEO4J

- API, User Defined Functions, Server Extensions : <https://neo4j.com/developer/java/>.
- Drivers : Java, Python, R, .NET...
- Neo4j Desktop : <https://neo4j.com/download/>.
- Neo4j Sandbox : <https://neo4j.com/sandbox-v2/>.
- Neo4j Cloud : <https://neo4j.com/cloud/>.

ARTICLES

Data-Bzh

- <http://data-bzh.fr/groupees-meetup-a-rennes-partie-1/>
- <http://data-bzh.fr/groupees-meetup-a-rennes-partie-2/>
- <http://data-bzh.fr/groupees-meetup-a-rennes-partie-3-neo4j/>

Théorie des graphes

- <https://github.com/michelcaradec/Graph-Theory>

CONCLUSION

MATCH (everybody:Human)

MATCH (neo4j:Engine {name: 'Neo4j'})

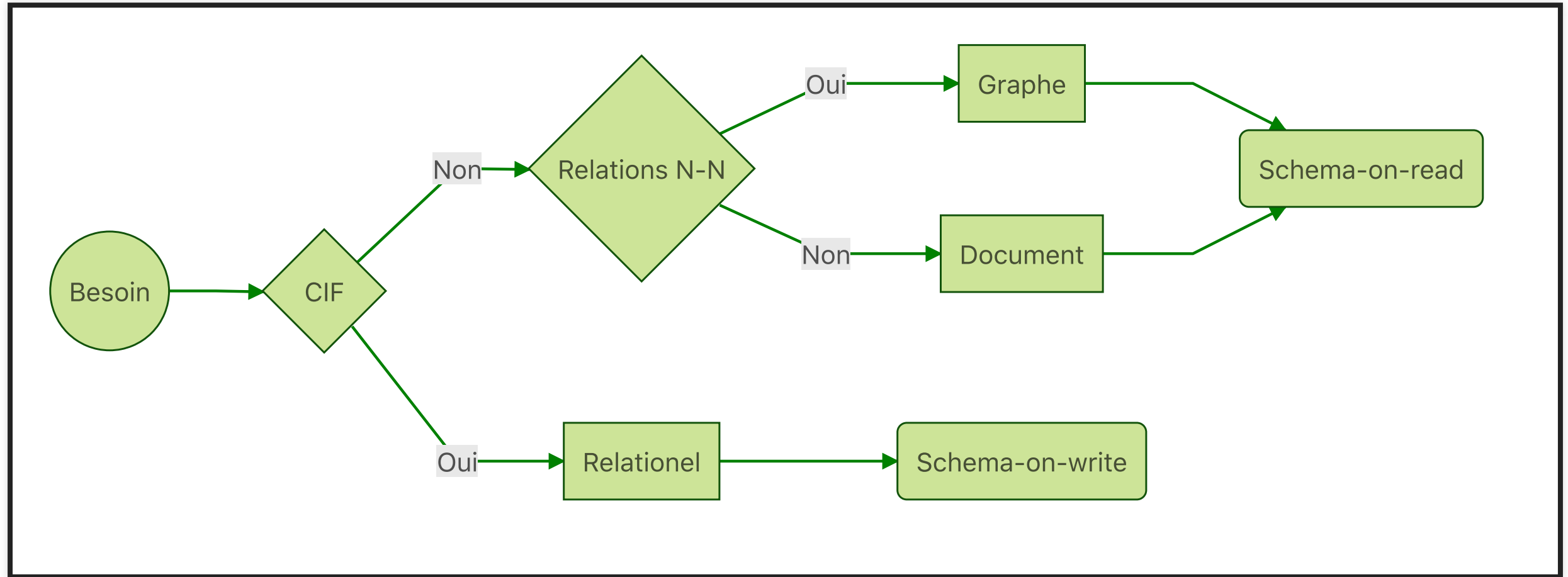
MERGE (everybody)-[:LIKE]->(neo4j);

Merci de votre attention.

QUESTIONS & RÉPONSES

ONE MORE THING...

QUEL MODÈLE DE DONNÉES ?



Critères de décision non-exhaustifs.

REPRESENTING A PROPERTY GRAPH USING A RELATIONAL SCHEMA

```
CREATE TABLE vertices (  
  vertex_id integer PRIMARY KEY,  
  properties json  
);  
  
CREATE TABLE edges (  
  edge_id integer PRIMARY KEY,  
  tail_vertex integer REFERENCES vertices (vertex_id),  
  head_vertex integer REFERENCES vertices (vertex_id),  
  label text,  
  properties json  
);  
  
CREATE INDEX edges_tails ON edges (tail_vertex);  
CREATE INDEX edges_heads ON edges (head_vertex);
```

Designing Data-Intensive Applications

Martin Kleppmann, éditions O'Reilly, ISBN 978-1-449-37332-0