UVSQ
université PARIS-SACLAY

IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY
CAMPUS DE RAMBOUILLET

Introduction aux bases de données de type Graphe

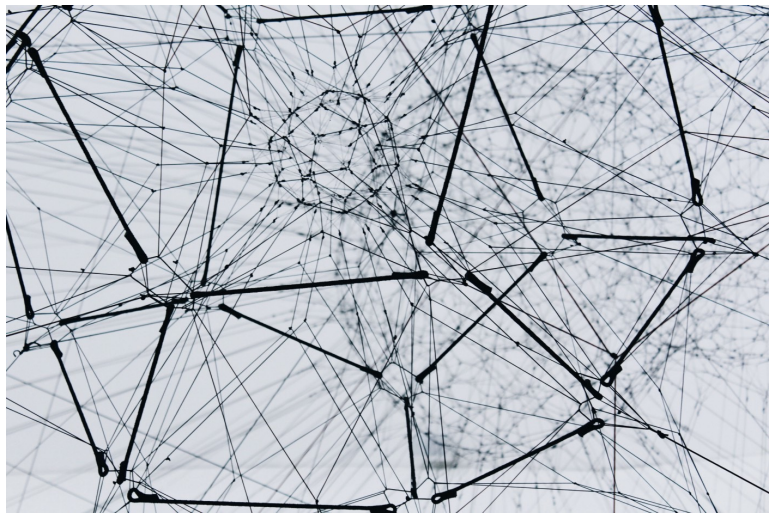
Karine ZEITOUNI

BUT3 INFO-FI

Edition 2023-24

Plan

1. BD NOSQL & BD Graphe
2. Neo4J
3. Application



1

BD NoSQL & BD Graphe

Modèle de BD relationnel ou pas !

SQL versus NoSQL

- Vous connaissez bien le modèle relationnel et SQL

Solution prédominante en bases de données sur le marché !

- Les bases de type Graphe sont un exemple de modèle non-relationnel et de système dit NoSQL

Mais au juste c'est Quoi ? Pourquoi ? Et Comment ?

Au-delà du relationnel… NoSQL

- **NoSQL**
  - Correspond aux BD non relationnelles, abréviation de Not only SQL
  - Il correspond à un mécanisme de stockage autre que des tables et la capacité de consulter et manipuler des données non-tabulaires.
  - Il existe plusieurs technologies NoSQL.

    En général, plus flexibles (supportent des données plus complexes et n'imposent pas de schéma) et s'adaptent à de gros volumes de données.

Au-delà du relationnel… NoSQL

- comparaison

Learning Neo4j 3.x - Second Edition: Effective data modeling, performance tuning and data visualization techniques in Neo4j, Jerome Baton, Rik Van Bruggen, Packt Publishing (2017)

Au-delà du relationnel… NoSQL (suite)

- **Catégories de systèmes NoSQL**
  - NoSQL utilisent des modes de stockage optimisés pour des usages spécifiques, comme pour :
    - Le stockage clé-valeur (ex. DynamoDB d'Amazon)
    - Le stockage sous forme de Document (e. MongoDB)
    - Le stockage orienté Colonnes (ex. MonetDB, HBase)
    - Le stokage sous forme de Graphes (notre focus)

BD Graphe – C'est quoi ?

- **Définition**
  - C'est une collection d'arcs et de nœuds formant une structure de graphe pour représenter et stocker les données.
    - Un nœud représente une entité du graphe, ex. personne, role, animal, etc.
    - Un arc représente une relation (un lien) entre les entités, ex. ami, joue_dans, etc.
- **Forces :**
  - Requêtes rapides car pas de jointure
  - Facile à explorer visuellement

BD Graphe – C’est quoi ? (suite)

- Chaque nœud possède un identifiant unique et une collection de propriétés
- Chaque arc possède un identifiant unique, des nœuds entrants et sortants et une collection de propriétés
- un arc est un lien orienté
- Les nœuds peuvent être reliés à d’autres nœuds suivant plusieurs liens : comme les association N-M
- Les nœud et les arcs peuvent avoir des propriétés, similaires aux valeurs d’attributs dans une table mais la structure est plus flexible qu’en relationnel, car définies par des paires (clé, valeur)

Cas d'usages des BD Graphe

Intérêt

- Permet de trouver des relations qu'on ne peut pas voir en travaillant dans une base de données SQL.

Principales applications

- Réseaux sociaux & systèmes de recommandation : on peut afficher le fil des activités, de nouveaux amis peuvent être recommandés en fonction de leur proximité aux amis actuels. Recommandation de produits
- Gestion d'actifs - banque on peut traiter les transactions financières en quasi temps réel.
- Détection de fraudes, évasion fiscale(*) on peut détecter plusieurs comptes (personnes et adresses différentes) avec pourtant la même adresse IP. Démanteler un réseau de sociétés écrans et de comptes offshore enregistrés aux noms de façade ou de proches parents.

(*) https://neo4j.com/blog/panama-papers-graph-database-download/

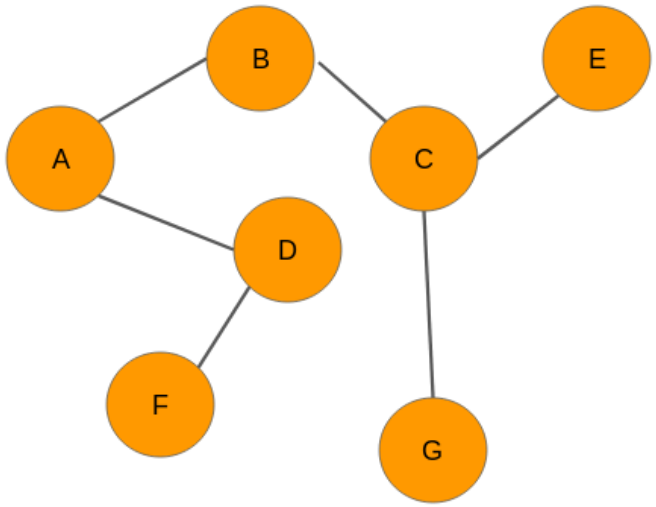
Exemple 1 – Réseau Social

Exemple 1 - Relationnel Versus Graphe

Format Relationnel
- *Liens difficiles à voir*
- *Jointures couteuses*

User	Friend
A	B
A	D
B	A
B	C
C	B
C	E
C	G
D	A
D	F
E	C
F	D
G	C

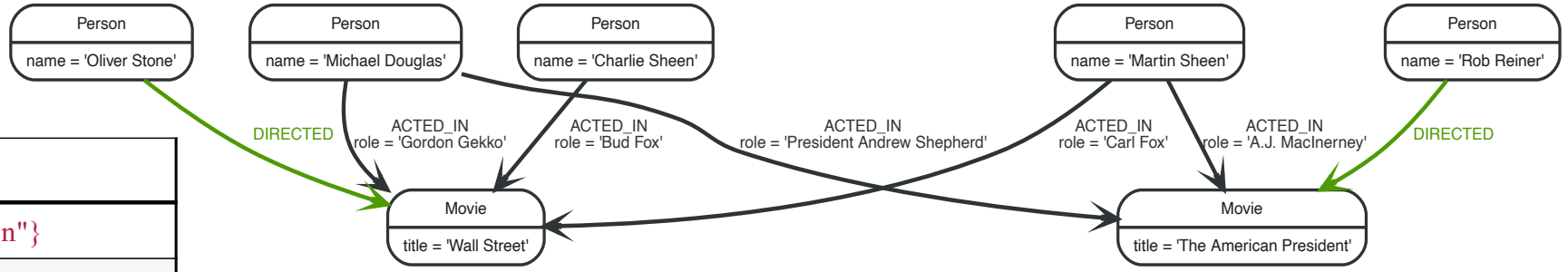
VS



Format Graphe
- *liens visibles ;*
- *beaucoup plus adapté à la navigation*
- *Plus performant*

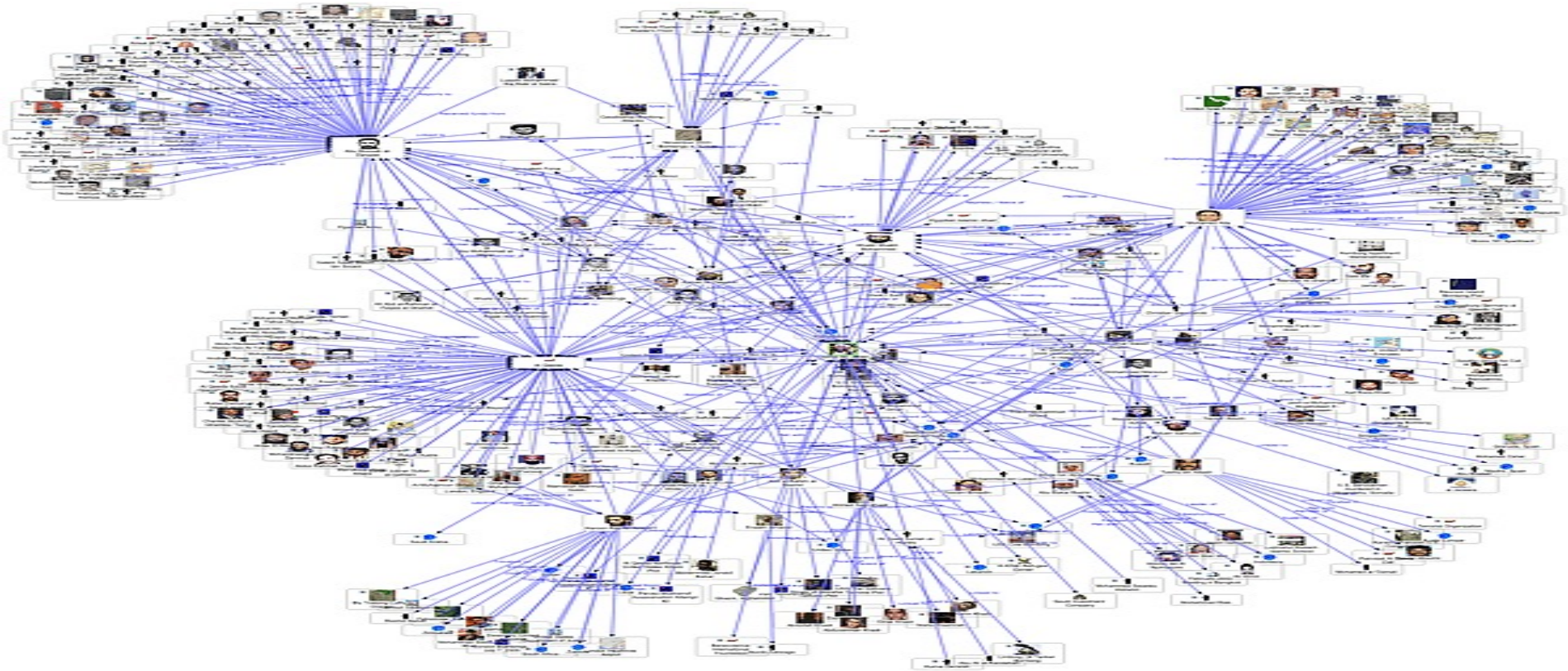
Exemple 2 – Cinéma

- Noeuds et arcs avec propriétés



Node[0]{name:"Charlie Sheen"}
Node[1]{name:"Martin Sheen"}
Node[2]{name:"Michael Douglas"}
Node[3]{name:"Oliver Stone"}
Node[4]{name:"Rob Reiner"}
Node[5]{title:"Wall Street"}
Node[6]{title:"The American President"}

Visualisation d'une BD Graphe
Réseau social large échelle



2

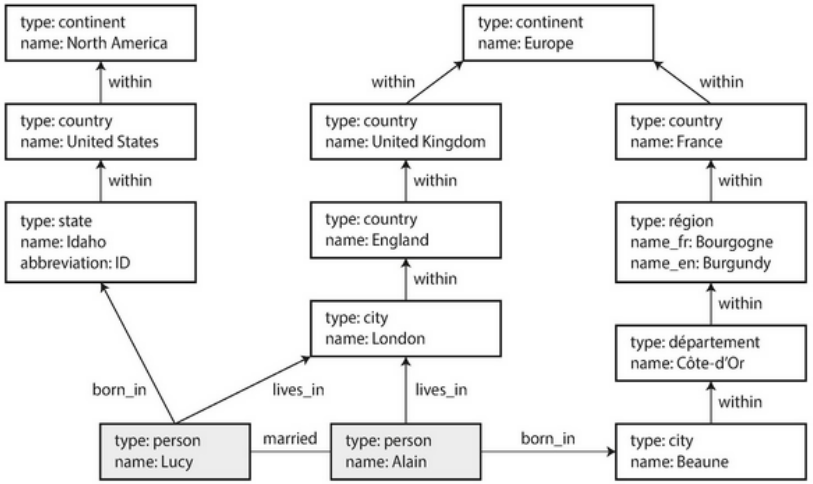
Neo4J

Gestionnaires de BD Graphe

- Neo4j est le gestionnaire de base de données de type graphe le plus connu.
- Il supporte des graphes de grande taille, et des requêtes (Cypher) retournant jusqu'à des milliers de nœuds.

- Préféré au SGBD relationnel lorsque :
  - Les données sont très liées entre elles
  - L'application requière un schéma flexible (évolutif)
  - Pour représenter les graphes de connaissances (ex. Wiki, Chatbots)
  - Lorsqu'on requête fréquemment les liens entre les objets.

Neo4J/Cypher plus simple que SQL




```
MATCH (person)-[:BORN_IN]->()-[:WITHIN*0..]->(us:Location {name:'United States'}),
(person)-[:LIVES_IN]->()-[:WITHIN*0..]->(eu:Location {name:'Europe'})
RETURN person.name
```

```
WITH RECURSIVE
-- in_usa is the set of vertex IDs of all locations within the United States
in_usa(vertex_id) AS (SELECT vertex_id FROM vertices WHERE properties->>'name' =
'United States' UNION SELECT edges.tail_vertex FROM edges
JOIN in_usa ON edges.head_vertex = in_usa.vertex_id WHERE edges.label = 'within'
),
-- in_europe is the set of vertex IDs of all locations within Europe
in_europe(vertex_id) AS ( SELECT vertex_id FROM vertices WHERE properties->>'name' =
'Europe' UNION SELECT edges.tail_vertex FROM edges
JOIN in_europe ON edges.head_vertex = in_europe.vertex_id WHERE edges.label = 'within'
),
-- born_in_usa is the set of vertex IDs of all people born in the US
born_in_usa(vertex_id) AS ( SELECT edges.tail_vertex FROM edges
JOIN in_usa ON edges.head_vertex = in_usa.vertex_id WHERE edges.label = 'born_ina),
- lives_in_europe is the set of vertex IDs of all people living in Europe
lives_in_europe(vertex_id) AS (
SELECT edges.tail_vertex FROM edges
JOIN in_europe ON edges.head_vertex = in_europe.vertex_idWHERE edges.label ='lives_in')
SELECT vertices.properties->>'name' FROM vertices
-- join to find those people who were both born in the US *and* live in Europe
JOIN born_in_usa ON vertices.vertex_id = born_in_usa.vertex_id
JOIN lives_in_europe ON vertices.vertex_id = lives_in_europe.vertex_id;
```

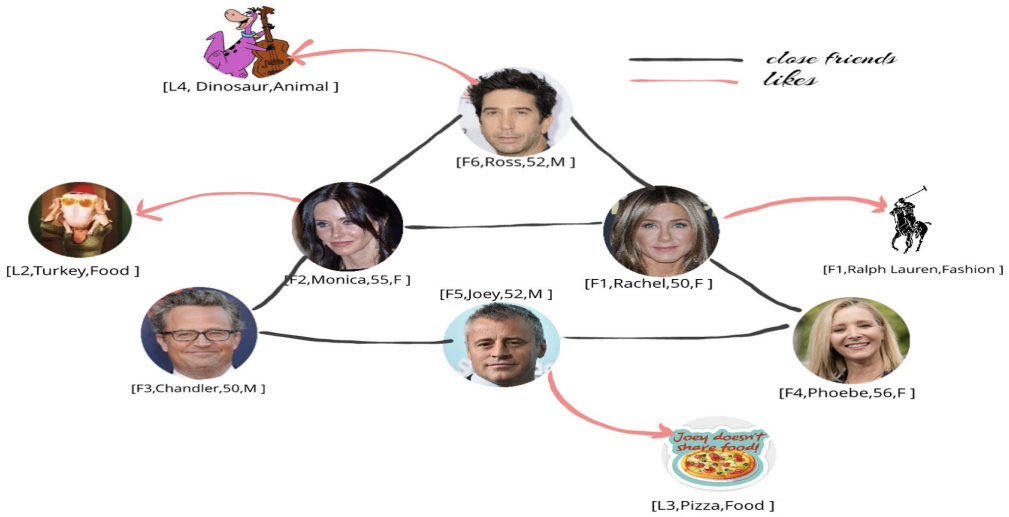
Cypher versus SQL
(Martin Kleppmann, Designing Data Intensive Applications, 2017)

3

Application

Exemple – Réseau social

- **BD FRIENDS**
  - Considérons une base de données d'un réseau social créé pour la fameuse série TV FRIENDS (1994-2004)



BD FRIENDS

- **On souhaite représenter les acteurs et leur role dans la série :**
  - Jennifer Aniston pour le rôle de Rachel Green,
  - Courteney Cox pour Monica Geller,
  - Lisa Kudrow pour Phoebe Buffay,
  - Matt LeBlanc pour Joey Tribbiani,
  - Matthew Perry pour Chandler Bing,
  - David Schwimmer pour Ross Geller
- Les relier aux pages qu'ils aiment dans les catégories :
  - food,
  - fashion,
  - animal.

Schéma relationnel de la BD FRIENDS

- En relationnel, elle correspond à deux entités :
  - 2 tables: Personnes et Pages.
  - Mais il faut 2 autres tables pour représenter les liens :
    - entre Personnes (CloseFriendsWith)
    - et de Personnes à Pages (Likes)

Schéma relationnel de la BD FRIENDS

Person

ID	Name	AGE	GENDER
1	Rachel	50	F
2	Monica	55	F
3	Chandler	50	M
4	Phoebe	56	F
5	Joey	52	M
6	Ross	52	M

CloseFriendsWith

ID_1	ID_2
1	2, 4, 6
2	3, 5
3	5
4	1, 5
6	2

Page

ID	NAME	TYPE
1	Ralph Lauren	Fashion
2	Turkey	Food
3	Pizza	Food
4	Dinosaur	Animal

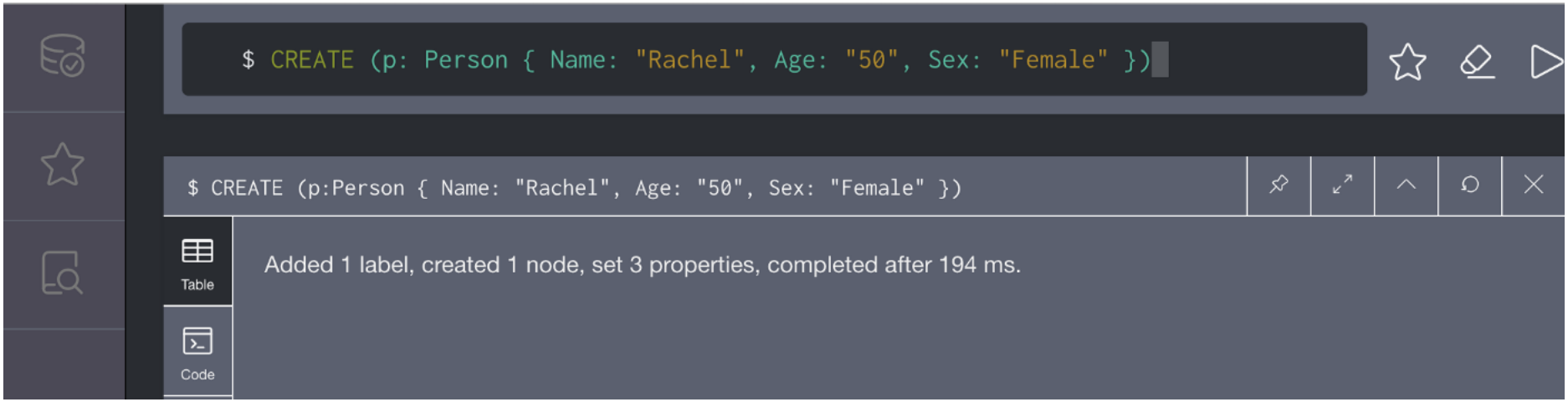
Likes

ID_USER	ID_PAGE
1	1
2	2
5	3
6	4

Création avec Cypher

- **Création d'un noeud :**
  - Syntaxe

    CREATE(variable_du_noeud: LABEL_du_noeud
    {attribut1: valeur1, attribut2: valeur2,… })



Création avec Cypher

- **Création de plusieurs noeuds à la fois :**

```
CREATE ( r: person { NAME: "Ross", Age:"52", Sex:"male" }),
( c: person { NAME: "Chandler", Age:"50", Sex:"male" }),
( j: person { NAME: "Joey", Age:"52", Sex:"male" }),
( m: person { NAME: "Monica", Age:"55", Sex:"Female" }),
( p: person { NAME: "Phoebe", Age:"56", Sex:"Female" }),
( ra: person { NAME: "Rachel", Age:"50", Sex:"Female" })
Return r, c, j, m, p, ra                    // permet d'afficher le graphe

CREATE (p: food {NAME: "Pizza", Type:"food"}), (t: food {NAME: "Turkey",
Type:"food"}), (ral: Fashion {NAME: "Ralph Lauren", Type:"Fashion"}),
(d: Animal {NAME: "Dinosaur", Type:"Animal"})
Return p, t, ral, d
```

Création avec Cypher

- ## Création d'arcs :

```
MATCH (j: person),(p: person)
WHERE j.NAME="Joey" AND p.NAME="Phoebe"
CREATE     (j)-[:Close_Friend_With]->(p),
           (p)-[:Close_Friend_With]->(j)
Return p, j
```

Rem: Les arcs étant orientés, ici on en crée dans les 2 sens

Création avec Cypher

- **Création d'arcs (suite) :**

```
MATCH       (ra: person),(m: person), (r: person), (p: person)
WHERE        ra.NAME="Rachel" AND m.NAME="Monica"
          AND r.NAME="Ross"     AND  p.NAME="Phoebe"
CREATE      (ra)-[:Close_Friend_With]->(r),
                (r)-[:Close_Friend_With]->(ra),
                (ra)-[:Close_Friend_With]->(m),
                (m)-[:Close_Friend_With]->(ra),
                (ra)-[:Close_Friend_With]->(p),
                (p)-[:Close_Friend_With]->(ra)
Return ra, r, p, m
```

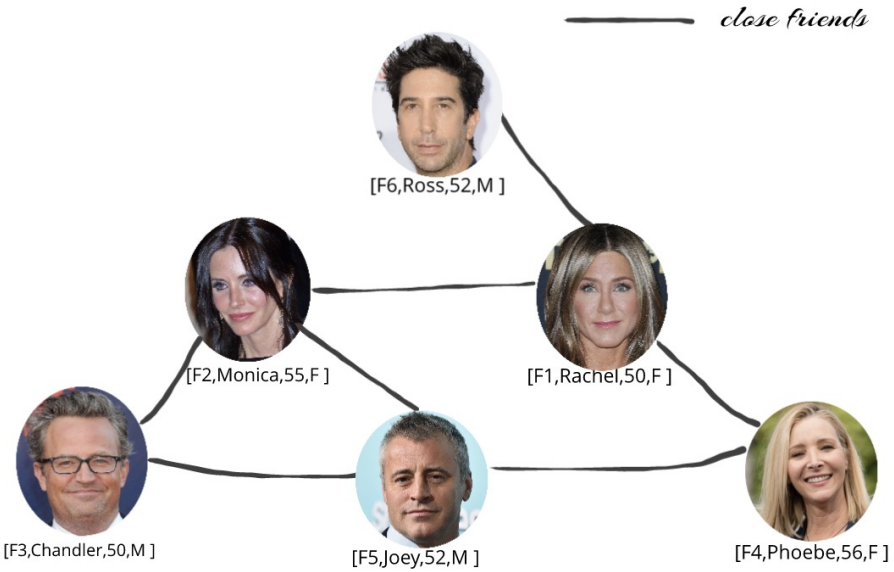
Création avec Cypher

- ## Création d'arcs (suite) :

```
MATCH     (j: person), (m: person), (c: person)
WHERE     j.NAME="Joey" AND m.NAME="Monica"
          AND c.NAME="Chandler"
CREATE    (j)-[:Close_Friend_With]->(c),
          (c)-[:Close_Friend_With]->(j),
          (m)-[:Close_Friend_With]->(j),
          (j)-[:Close_Friend_With]->(m),
          (m)-[:Close_Friend_With]->(c),
          (c)-[:Close_Friend_With]->(m)
Return j, m, c
```

Création avec Cypher

- Création des arcs :



Création avec Cypher

- **Création des arcs “Likes” :**

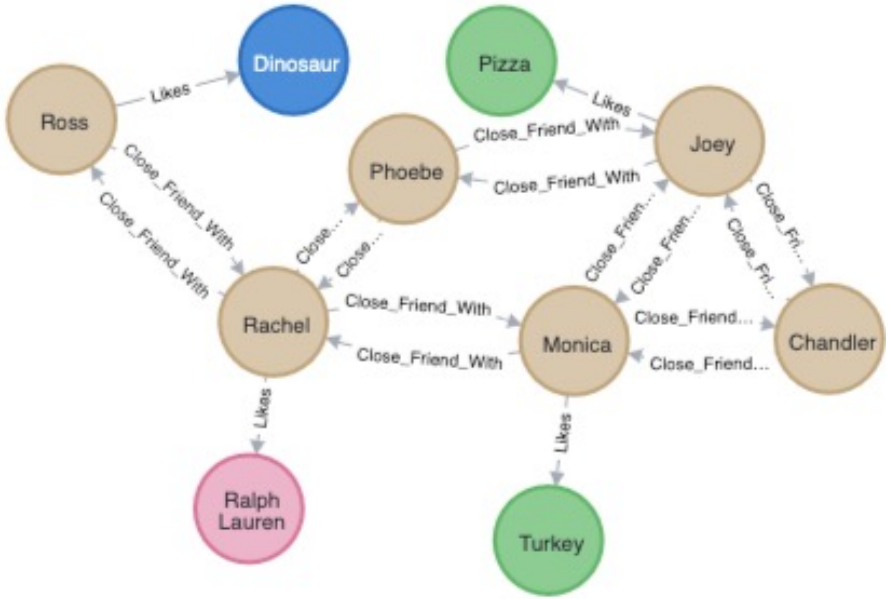
```
MATCH (ra: person), (ral: Fashion)
WHERE ra.NAME="Rachel" AND ral.NAME="Ralph Lauren"
CREATE (ra)-[:Likes]->(ral)
Return ra, ral

MATCH (r: person), (d: Animal)
WHERE r.NAME="Ross" AND d.NAME="Dinosaur"
CREATE (r)-[:Likes]->(d)
Return r, d

MATCH (m: person), (t:food)
WHERE m.NAME="Monica" AND t.NAME="Turkey"
CREATE (m)-[:Likes]->(t)
Return m, t
```

....

Vérification

- Affichage :
  MATCH(n) RETURN n
  Ou bien :
  match (n)-[r]-(m) return n, r, m

  Affichage possible tabulaire
  ou au format json



- Affichage partiel :
  MATCH(n) RETURN n LIMIT 5 // retourne l^{e} 5 1er noeuds

Requêtes Cypher

- Les amis proches de Rachel :

```
MATCH (p1:person)-[:Close_Friend_With]->(p2:person)
WHERE p1.NAME="Rachel"
Return p1,p2
```

- Les personne qui aiment « food » *(Label de nœud)* :

```
MATCH (p:person)-[:Likes]->(f:food)
Return p, f
```

- Les personne qui n'aiment pas « food »:

```
MATCH (p:person), (f:food)
WHERE NOT (p:person)-[:Likes]->(:food)
Return p, f
```

Suppression

- Suppression :

  MATCH (n) DETACH DELETE n // supprime le graphe

  Ou bien :

  MATCH (n) -[r]-() DELETE n, r

Attention ! Sans DETACH :

MATCH (n) DELETE n *//supprime uniquement les nœuds isolés*

MATCH (n)-[r]-() DELETE **r** *//supprime uniquement les liens !*

- Suppression partielle :

  MATCH (n: food) DELETE n *//Nœuds isolés ayant le label food*

  MATCH (n)-[r: LIKES]-() DELETE **r** *//supprime les liens LIKES*

Import d'un CSV

test.csv

```
Source,Target,distance
A,C,3
B,D,5
C,J,2
D,E,4
E,G,4
F,A,5
G,D,1
H,J,5
L,E,8
J,F,7
D,C,3
A,L,6
C,B,7
G,P,4
```

Exemple 1 :

```
LOAD CSV WITH HEADERS FROM "file:///C:/mon_chemin/test.csv"
AS line MERGE (n:MyNode {Name:line.Source})
MERGE (m:MyNode {Name:line.Target})
MERGE (n) -[:TO {Dist:line.distance}]-> (m)
```

Exemple 2 :

```
LOAD CSV WITH HEADERS FROM "file:///mon_chemin/terrorist_data_subset.csv" AS Ligne
MERGE (c:Country {Name:row.Country}) MERGE (a:Actor {Name: row.ActorName, Aliases:
row.Aliases, Type: row.ActorType}) MERGE (o:Organization {Name: row.AffiliationTo}) MERGE
(a)-[:AFFILIATED_TO {Start: row.AffiliationStartDate, End: row.AffiliationEndDate}]->(o)
MERGE(c)<-[:IS_FROM]-(a);
```

Syntaxe plus complète

- Se référer à Cypher Refcard:
  https://neo4j.com/docs/cypher-refcard/current/

Exercice de Conception

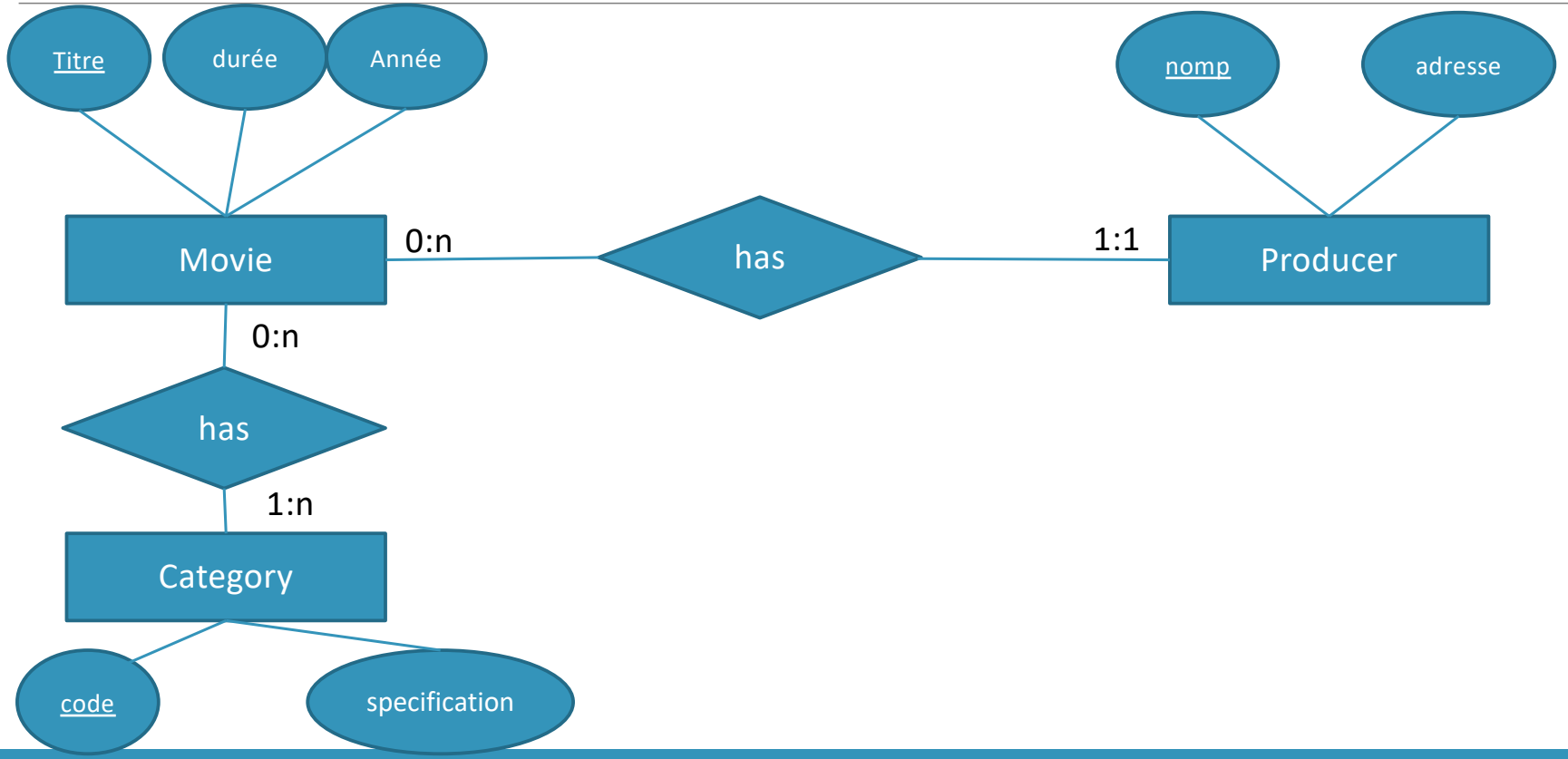
Soit la base de données Cinéma de schéma Entité-Association ci-après

Elle est instanciée dans un modèle relationnel comme suit

Questions :

1. Concevoir l'équivalent de cette base de données sous forme de graphe et créez la sous Neo4J
2. Insérez les données exemples ci-dessous

Modèle entité - association



Modèle relationnel

Table: Movie

Titre	Durée	Année	Nomprod
aliens	137	1982	Clean kill movies
Blade runner	117	1982	Sf movies
casablanca	102	1942	Classique film
Dances with wolves	180	1990	Constance film
Seven pounds	120	2007	Clean kill movies

Modèle relationnel (suite)

Table: Producer

Nomp	Adresse
Clean kill movies	45, walker street, houston
Sf movies	13, champs elysees, paris
Classique film	2, place kleber, strasbourg
Constance film	Gumpendorferstrasse 17, a-106

Table: MovieCatgory

Titre	code
aliens	a
aliens	sf
blade runner	sf
casablanca	cd
dances with wolves	w
seven pounds	w

Table: Category

Code	Specification
a	action
sf	Science fiction
cd	Comedie drame
w	war