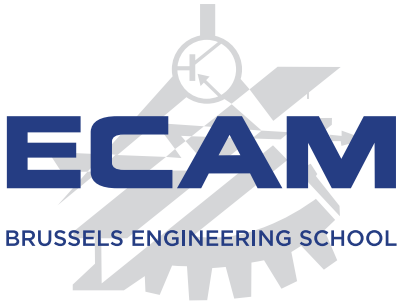


I404C NoSQL

Séance 4

Modèle orienté-graphe

Neo4j, OrientDB

Sébastien Combéfis

2019

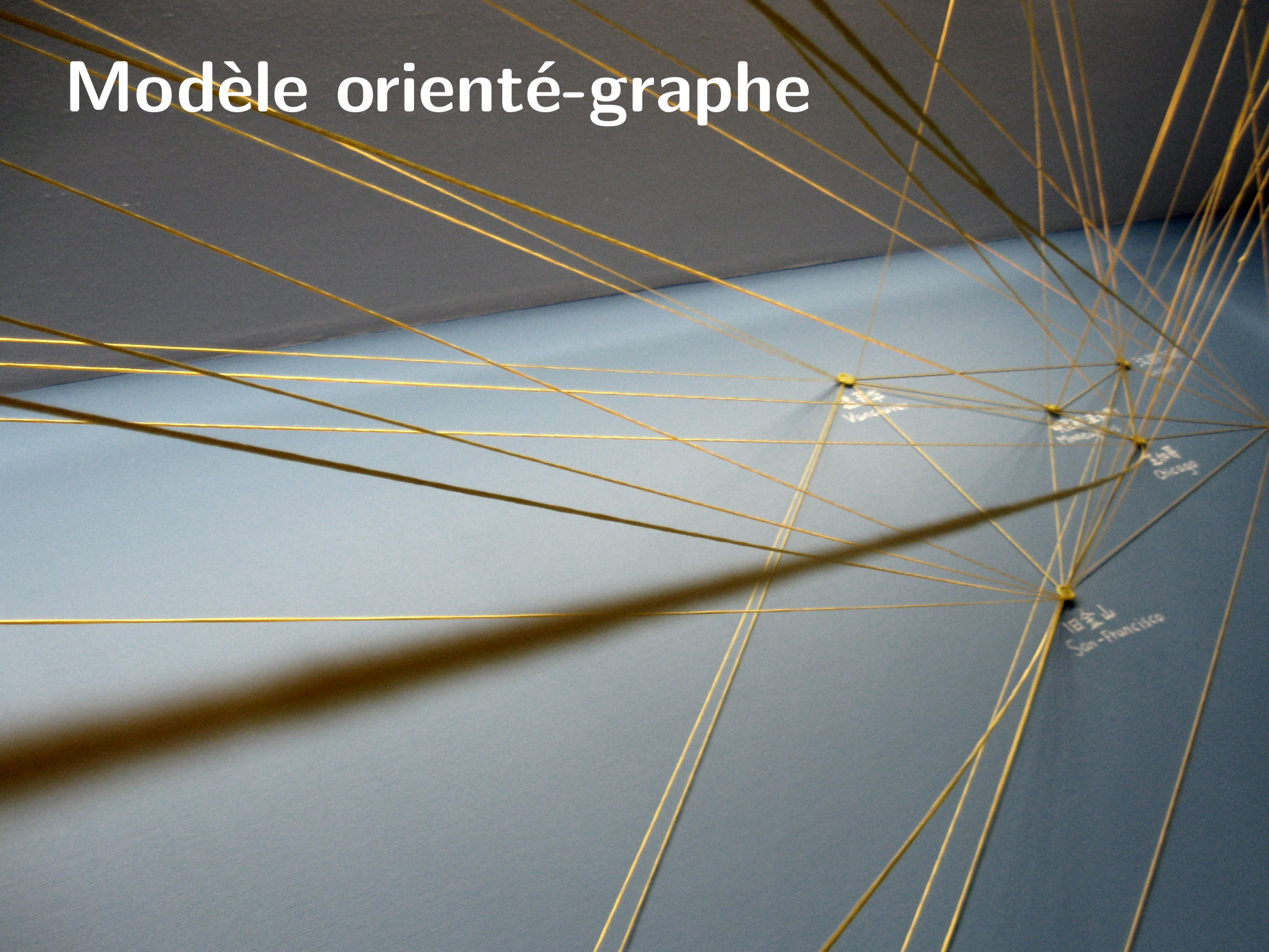
Rappels

- Modèle de base de données orienté colonnes
  - Définition et principe du stockage de colonnes
  - Moteur HBase et BigData avec HFile et HDFS
  - Moteur Cassandra et son langage de requête CQL
- Stockage des données sur le disque et optimisation
  - Comparaison du stockage des lignes ou des colonnes
  - C-Store et son modèle de stockage en deux temps
  - Projection et compression des données

Objectifs

- Le modèle orienté-graphe
  - Définition d'un graphe et modèle de données
  - Langage et framework de requêtes graphe
  - Web sémantique et langages d'interrogation
- Exemples de bases de données
  - Neo4j
  - OrientDB

Modèle orienté-graphe



Graphe (1)

- Base de données orientée-graphe stocke des entités et relations

  Les relations permettent de relier les entités entre elles

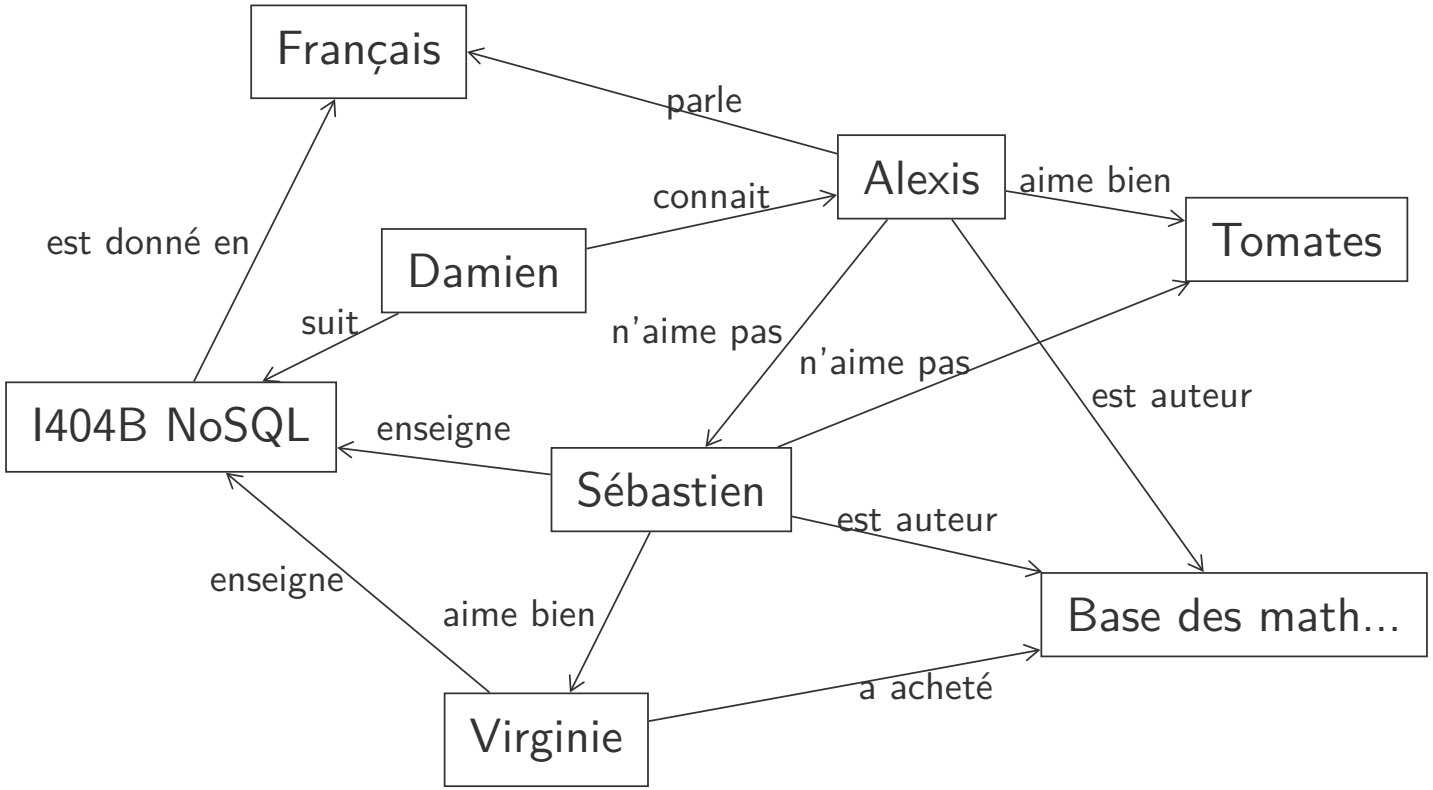
- Un nœud représente une entité et possède des propriétés

  Une instance d'un objet dans une application

- Une arête est directionnelle et possède des propriétés

  Établissement de liens entre les instances dans une application

Graphe (2)


Français
parle
Alexis
aime bien
Tomates
connait
est donné en
Damien
suit
n'aime pas
n'aime pas
est auteur
I404B NoSQL
enseigne
Sébastien
est auteur
enseigne
aime bien
Base des math...
a acheté
Virginie

Modèle de données (1)

- Un graphe est un ensemble de nœuds et d'arêtes

  Tous deux possédant plusieurs propriétés

- Les arêtes sont directionnelles et ont un type

  Certaines peuvent être bidirectionnelles dans la signification

- Deux principaux avantages sur le modèle relationnel
  - Permet de stocker plusieurs types de relations en même temps
  - Traversée du graphe sans devoir recalculer les relations

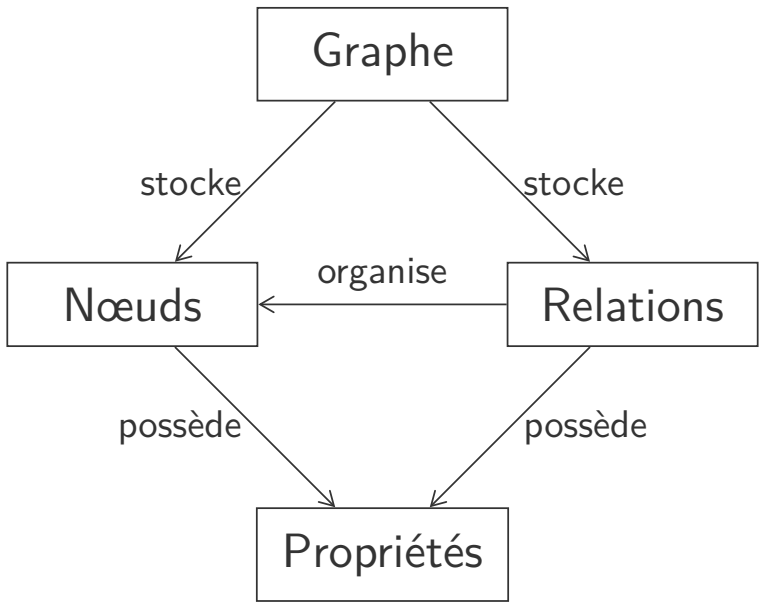
Modèle de données (2)

- Plusieurs relations entre les éléments stockés

  Que l'on peut représenter avec un graphe

- Il n'y a jamais de liens morts

  Impossible de supprimer un nœud qui intervient dans des relations

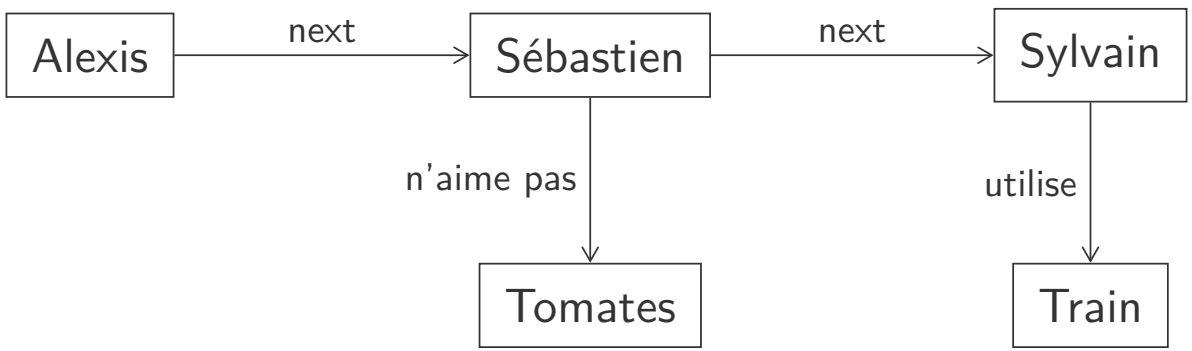


Relation

- Traversée des relations très rapide en orienté-graphe

  Relations stockées comme telles, doivent pas être construites

- Plusieurs types de relation possibles
  - Relations entre éléments de domaines différents
  - Relation secondaire indiquant une catégorie, un chemin...
  - Quad-tree pour l'indexation ou liste chainée pour l'accès trié



Cas d'utilisation

- Tout domaine qui est très riche en liens entre des entités

  Réseaux sociaux : amitié, employé/employeur, compétences...

- Routage/dispatching et services dépendants d'une localisation

  Lieux en nœuds et relations contiennent distance

- Moteur de recommandations automatiques

  « Tes amis ont aussi acheté cela »...

Cas de non utilisation

- Opération de mise à jour des nœuds pas immédiate

  Modification de propriétés des nœuds peut couter très cher

- Opérations globales sur le graphe peuvent être très couteuses

  En fonction de la base de données choisie

Web sémantique (1)

- Extension du Web pour l'utilisation de format de données

  Format de base RDF (Resource Description Framework)

- Permettre le partage des données entre applications

  Web des données pour lier et structurer l'information

- Références à des schémas de vocabulaire et concepts

  Ajout d'un sens aux informations déjà présentes sur le web

Resource Description Format (RDF)

- RDF développé fin des années 90 pour modéliser le Web

  Modélisation de ressources et de relations entre ces dernières

- Stockage de triplets représentant les relations
  - Triplets de la forme (sujet, prédicat, objet)
  - Différence avec le classique *entité-attribut-valeur* de la POO
- Exemple de représentation de *"The sky has the colour blue."*
  - Le sujet est *« le ciel »*
  - Le prédicat est *« avoir la couleur »*
  - L'objet est *« bleu »*

Web sémantique (2)

- Sémantique du texte *« Paul Schuster est né à Dresde. »*

  Ajout de propriétés aux balises `div` et `span`

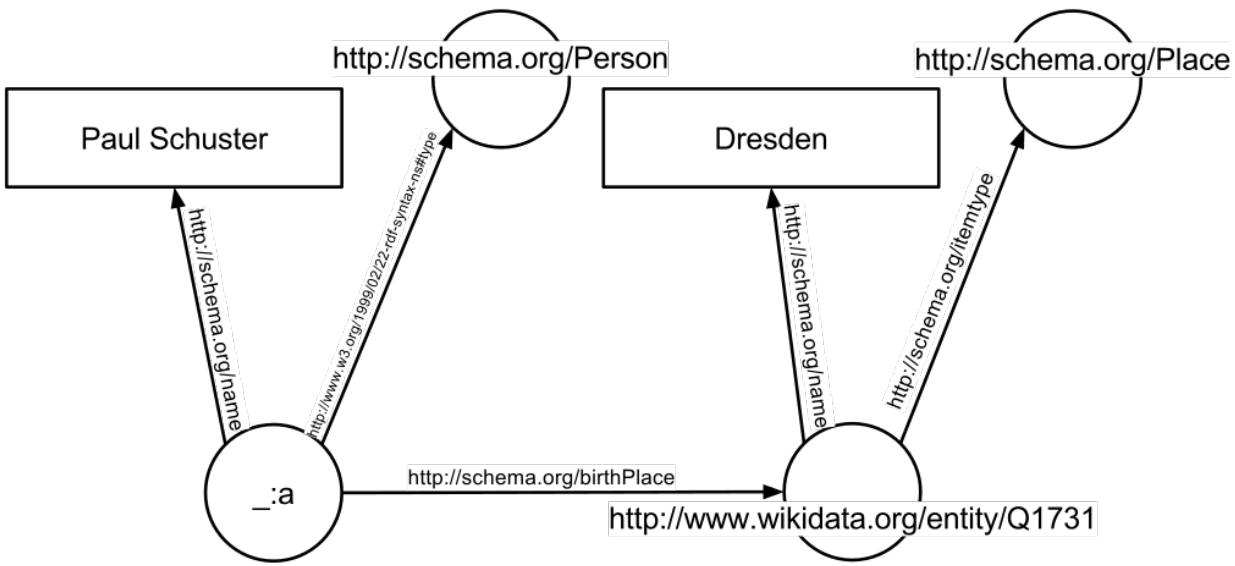
- Utilisation de la syntaxe RDFa pour décrire un mini-graphe

  Vocabulaire provenant de Schema.org et Wikidata

```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Paul Schuster</span> est né à
  <span property="birthPlace" typeof="Place" href="http://www.
  wikidata.org/entity/Q1731">
    <span property="name">Dresde</span>.
  </span>
</div>
```

Web sémantique (3)

```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Paul Schuster</span> est né à
  <span property="birthPlace" typeof="Place" href="http://www.
  wikidata.org/entity/Q1731">
    <span property="name">Dresde</span>.
  </span>
</div>
```



SPARQL (1)

- Langage de requêtes SPARQL Protocol and RDF Query Language

  Adapté à la structure spécifique des graphes RDF

- Quatre types de requêtes dans SPARQL
  - `SELECT` extrait un sous-graphe d'un graphe RDF en table
  - `CONSTRUCT` engendre un nouveau graphe complétant un autre
  - `ASK` pose une question répondue par `True/False`
  - `DESCRIBE` extrait un graphe RDF

- Description des contraintes sur les triplets avec clause `WHERE`

  Sauf pour `DESCRIBE` avec qui la clause est optionnelle

SPARQL (2)

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
       ?email
WHERE
  {
    ?person  a          foaf:Person .
    ?person  foaf:name  ?name .
    ?person  foaf:mbox  ?email .
  }
```

- Sélection de tous les triplets avec le même sujet `?person`

  Et récupère aussi les `?email` associés

- Vérification de la présence de trois relations

  Sujet est une `foaf:Person` avec un/des noms et boites mails

DBpedia (1)

- Extraction d'information structurée depuis Wikipédia

  Interrogation de la base de données avec SPARQL

- Projet lancé par deux universités allemandes publié en 2007

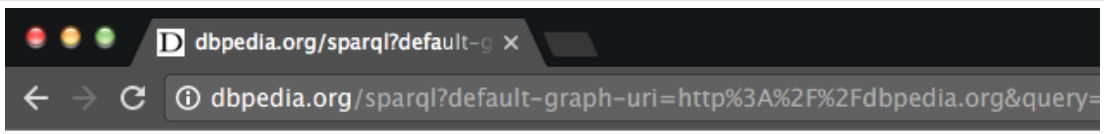
  Free University of Berlin et Leipzig University

- Permet de faire des liens entre plusieurs articles Wikipédia

  Grâce aux informations factuelles récoltées par le projet

DBpedia (2)

```
PREFIX res: <http://dbpedia.org/resource/>
SELECT ?object
WHERE
  {
    res:Edgar_F._Codd <http://purl.org/dc/terms/subject> ?object
  }
```



object
http://dbpedia.org/resource/Category:1923_births
http://dbpedia.org/resource/Category:2003_deaths
http://dbpedia.org/resource/Category:Alumni_of_Exeter_College,_Oxford
http://dbpedia.org/resource/Category:British_computer_scientists
http://dbpedia.org/resource/Category:Cellular_automatists
http://dbpedia.org/resource/Category:Fellows_of_the_Association_for_Computing_Machinery
http://dbpedia.org/resource/Category:IBM_Fellows
http://dbpedia.org/resource/Category:IBM_Research_computer_scientists
http://dbpedia.org/resource/Category:IBM_employees
http://dbpedia.org/resource/Category:Turing_Award_laureates
http://dbpedia.org/resource/Category:University_of_Michigan_alumni
http://dbpedia.org/resource/Category:Database_researchers
http://dbpedia.org/resource/Category:People_educated_at_Poole_Grammar_School
http://dbpedia.org/resource/Category:People_from_San_Jose,_California
http://dbpedia.org/resource/Category:People_from_the_Isle_of_Portland

Apache Jena

- Framework open source pour le Web sémantique

  Développé par la fondation Apache et avec Java

- API pour extraire des données et construire des graphes RDF

  Interrogation des modèles à l'aide de SPARQL

- Fournit du support pour OWL (Web Ontology Language)


Neo4j
GRAPH CONNECT
BOSTON 2013
INNOVATE. SHARE. CONNECT.
(graphs)-[:ARE]->(everywhere)
Neo4j
the world's leading graph database
Neo4j

Neo4j

- Système de gestion de graphes par Neo Technology, Inc.

  Basé à San Francisco, CA, USA et Malmö en Suède

- Transactions ACID avec stockage natif des graphes

  Le plus populaire dans les bases orientées-graphe

- Interrogation de la base avec un langage à travers HTTP

  Cypher Query Language (CQL)

Modèle de données (1)

- Le nœud est l'unité fondamentale qui constitue un graphe

  Peut être associé à des labels et peut posséder des propriétés

- Une relation connecte deux nœuds avec une direction

  Peut également posséder des propriétés

- Propriétés pour les nœuds et relations comme paires clé-valeur

  Valeurs de type nombre, chaine de caractères, booléen et listes

- Créer des ensembles de nœuds à l'aide de labels

  Un nœud peut posséder plusieurs labels

Modèle de données (2)

- Une traversée d'un graphe répond à une requête
  - Navigation de nœud en nœud
  - Interrogation avec langage déclaratif CQL
- Un chemin est une séquence de nœuds avec des relations

  Utilisé comme résultat d'une requête

Cypher Query Language (CQL)

- Interrogation d'une base Neo4j avec le langage Cypher

  Langage très simple et très puissant

- Deux clauses principales pour construire des requêtes
  - `CREATE` permet de créer une nouvelle entité
  - `MATCH` permet de chercher des entités
- Identification graphique des entités
  - Nœud représenté entre parenthèses et deux-points pour label
  - Relation représentée avec flèches et crochets pour détails
  - Propriétés représentées par un dictionnaire à la JSON

Installation de Neo4j

- Neo4j est un programme développé en Java et Scala
- Plusieurs programmes proposés après installation
  - `neo4j` permet de lancer plusieurs commandes de gestion
    - `neo4j start` démarre le serveur
    - `neo4j stop` arrête le serveur
    - `neo4j status` récupère le statut du serveur
  - `neo4j-shell` propose un client en ligne de commande

Lancement du serveur

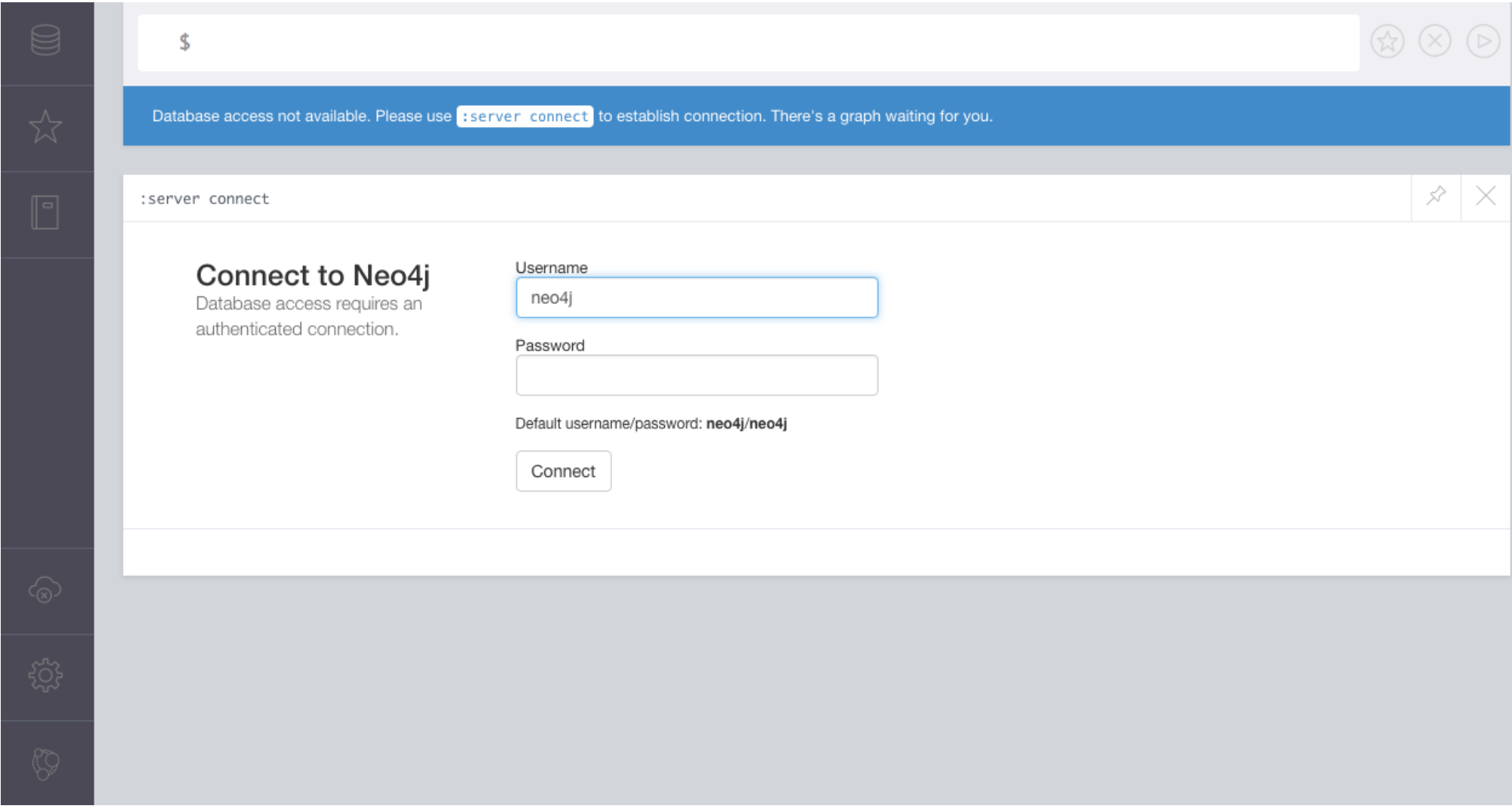
- Lancement du serveur et vérification de la connexion

  Le démarrage de Neo4j lance également un serveur web

```
& neo4j start
```

```
& neo4j-shell
Welcome to the Neo4j Shell! Enter 'help' for a list of commands
NOTE: Remote Neo4j graph database service 'shell' at port 1337

neo4j-sh (?)$
```

Browser Neo4j



Création d'un nœud

- Création d'un nouveau nœud avec la commande `CREATE`

  Spécification optionnelle de propriétés

```
neo4j-sh (?)$ CREATE (:Person {firstname: "Alexis"});
+--------------------+
| No data returned. |
+--------------------+
Nodes created: 1
Properties set: 1
Labels added: 1
557 ms

neo4j-sh (?)$
```

Chercher un nœud

- Recherche d'un nœud dans le graphe avec `MATCH`

  Spécification d'un filtre pour les nœuds désirés

- Exemple de recherche des deux nœuds déjà créés

  Stockage des résultats dans les variables `p` et `f`

```
neo4j-sh (?)$ MATCH (p:Person)
> MATCH (f:Food)
> RETURN p, f;
+-----------------------------------------------------------+
| p                                 | f                     |
+-----------------------------------------------------------+
| Node[0]{firstname:"Alexis"} | Node[1]{name:"Tomato"} |
+-----------------------------------------------------------+
1 row
50 ms

neo4j-sh (?)$
```

Ajouter une relation

- Ajout d'une relation dans le graphe avec `CREATE`

  Recherche éventuelle des nœuds avant

```
neo4j-sh (?)$ MATCH (alexis:Person {firstname: "Alexis"})
> MATCH (tomato:Food {name: "Tomato"})
> CREATE (alexis) -[r:LIKES]-> (tomato)
> RETURN r;
+--------------+
| r            |
+--------------+
| :LIKES[0]{} |
+--------------+
1 row
Relationships created: 1
321 ms

neo4j-sh (?)$
```

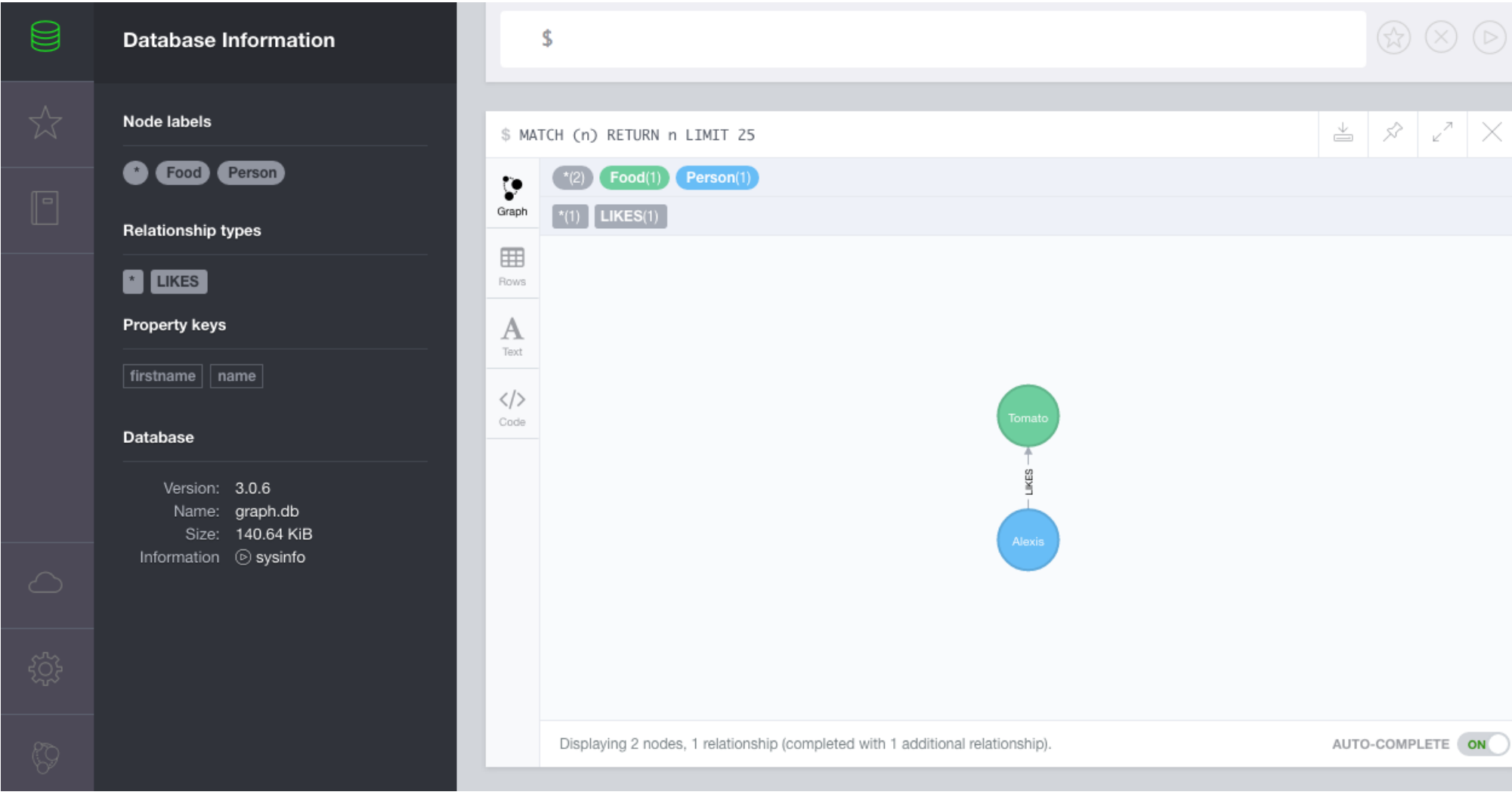
Chercher une relation

- Recherche d'une relation dans le graphe avec `MATCH`

  Variables du pattern de recherche seront remplies

```
neo4j-sh (?)$ MATCH (p:Person) -[LIKES]-> (Food {name: "Tomato"})
> RETURN p;
+------------------------------+
| p                            |
+------------------------------+
| Node[0]{firstname:"Alexis"} |
+------------------------------+
1 row
90 ms

neo4j-sh (?)$
```

Visualisation avec le browser



Modification d'un nœud

- Ajout/modification de propriétés avec la commande `SET`

  Récupération avec une requête `MATCH` puis modification

- Suppression d'une propriété en spécifiant `NULL` comme valeur

```
neo4j-sh (?)$ MATCH (p:Person {firstname: "Alexis"})
> SET p.sex = "M"
> SET p.age = 22
> RETURN p;
+-------------------------------------------+
| p                                         |
+-------------------------------------------+
| Node[0]{firstname:"Alexis",sex:"M",age:22} |
+-------------------------------------------+
1 row
Properties set: 2
202 ms

neo4j-sh (?)$
```

Filtrage des résultats

- Définition de filtre de recherche avec `WHERE`

  En spécifiant plusieurs conditions sur les entités récupérées

```
neo4j-sh (?)$ CREATE (:Person {firstname: "Damien", age: 24, sex:
 "M"});
[...]

neo4j-sh (?)$ MATCH (p:Person)
> WHERE p.age>22
> RETURN p.firstname;
+-------------+
| p.firstname |
+-------------+
| "Damien"    |
+-------------+
1 row
333 ms

neo4j-sh (?)$
```

Contrainte (1)

- Forcer l'intégrité des données à l'aide de contraintes

  Unicité des nœuds et des propriétés

- Définition d'une nouvelle contrainte avec `CONSTRAINT`

```
neo4j-sh (?)$ CREATE CONSTRAINT ON (p:Person)
> ASSERT p.firstname IS UNIQUE;
+--------------------+
| No data returned.  |
+--------------------+
Unique constraints added: 1
641 ms

neo4j-sh (?)$
```

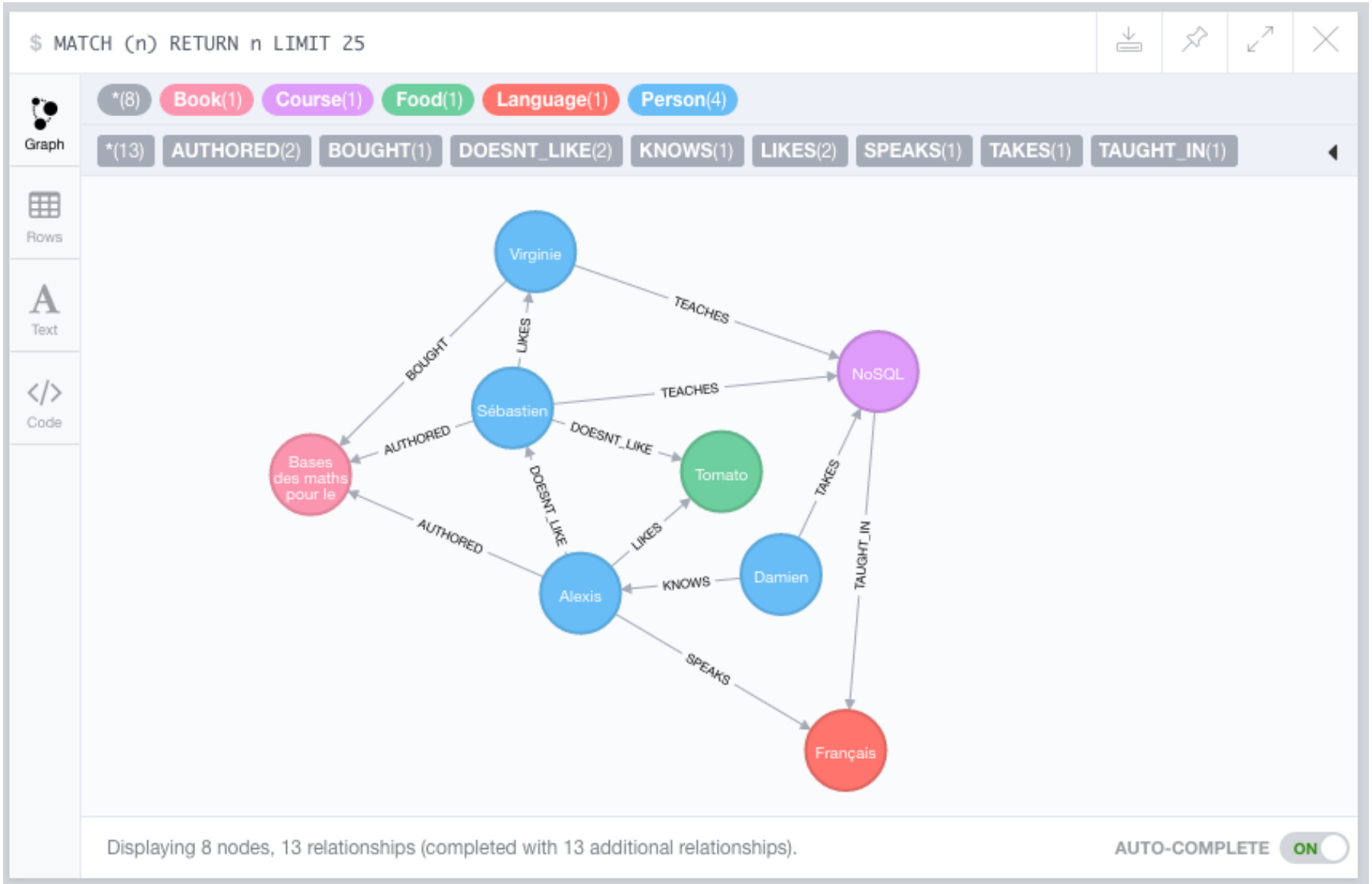
Contrainte (2)

- Contraintes vérifiées lors d'ajout ou modification des nœuds

  L'ajout d'une contrainte peut échouer s'il y a déjà des conflits

```
neo4j-sh (?)$ CREATE (:Person {firstname: "Damien"});
113 ms

WARNING: Node 3 already exists with label Person and property "
firstname"=[Damien]

neo4j-sh (?)$ CREATE CONSTRAINT ON (p:Person)
> ASSERT p.sex IS UNIQUE;
260 ms

WARNING: Unable to create CONSTRAINT ON ( person:Person ) ASSERT
person.sex IS UNIQUE:
Multiple nodes with label ‘Person‘ have property ‘sex‘ = ’M’:
  node(0)
  node(5)

neo4j-sh (?)$
```

Requête complexe (1)



Requête complexe (2)

- Récupérer tous les auteurs d'un bouquin donné

  On fixe la relation et le nœud d'arrivée

```
neo4j-sh (?)$ MATCH (p:Person) -[r:AUTHORED]-> (b:Book {title: "
Bases des maths pour le supérieur"}) RETURN p.firstname;
+-------------+
| p.firstname |
+-------------+
| "Sébastien" |
| "Alexis"    |
+-------------+
2 rows
14 ms

neo4j-sh (?)$
```

Requête complexe (3)

- Quelqu'un qui n'aime pas une personne qui en aime une autre

  Un enchainement de deux relations est à préciser dans la requête

```
neo4j-sh (?)$ MATCH (a:Person) -[r:DOESNT_LIKE]-> (b:Person) -[s:
LIKES]-> (c:Person) RETURN a.firstname, b.firstname, c.firstname;
+-----------------------------------------+
| a.firstname | b.firstname | c.firstname |
+-----------------------------------------+
| "Alexis"    | "Sébastien" | "Virginie"  |
+-----------------------------------------+
1 row
45 ms

neo4j-sh (?)$
```

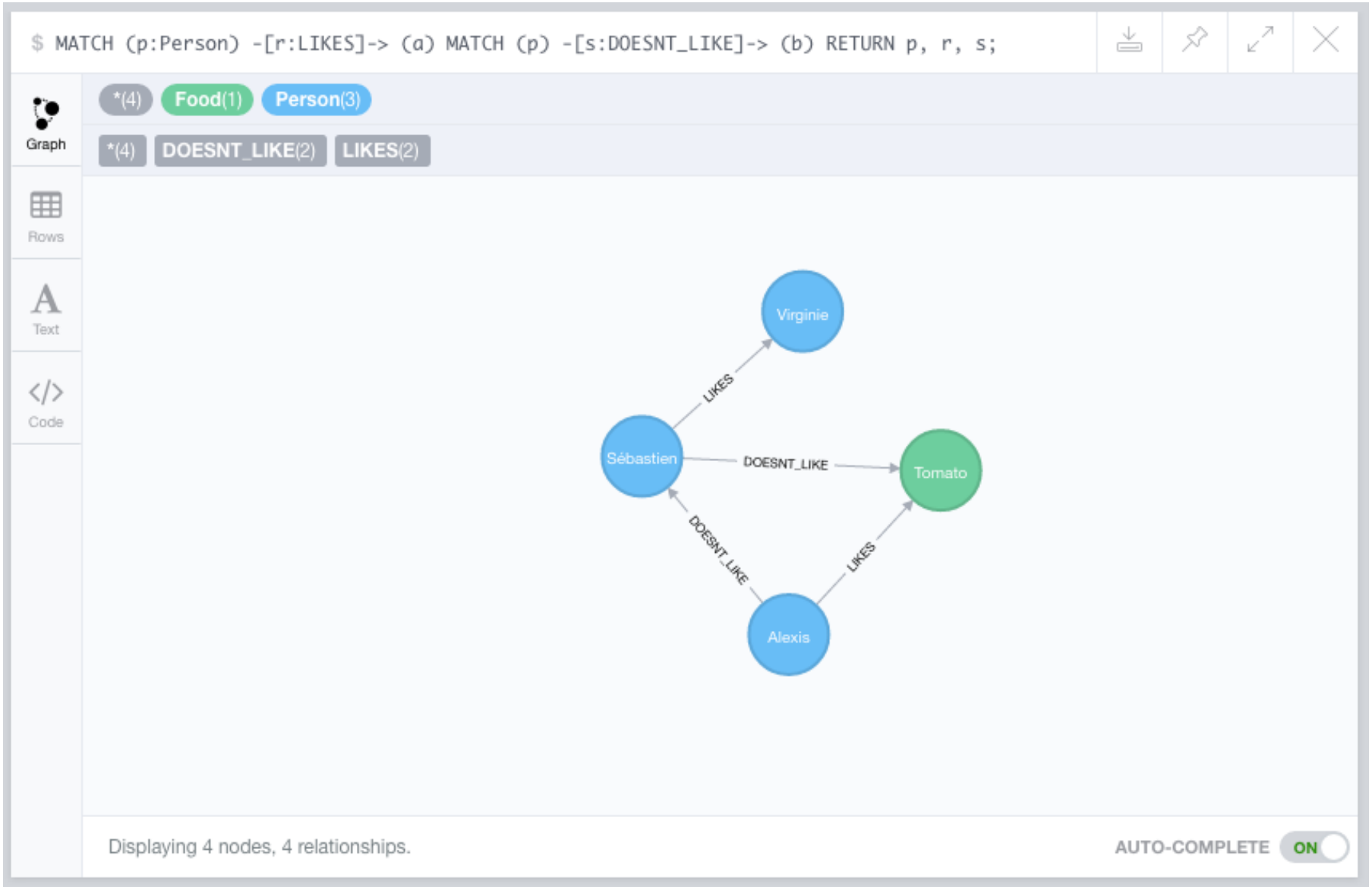
Requête complexe (4)

- Quelqu'un qui n'aime pas et aime deux choses différentes

  Deux relations sont à préciser dans la requête

```
neo4j-sh (?)$ MATCH (p:Person) -[r:LIKES]-> (a) MATCH (p) -[s:
DOESNT_LIKE]-> (b) RETURN p.firstname, a, b;
+-----------------------------------------------------------------
| p.firstname | a                                    | b
+-----------------------------------------------------------------
| "Alexis"    | Node[1]{name:"Tomato"}               | Node[2]{firstname
:"Sébastien"} |
| "Sébastien" | Node[4]{firstname:"Virginie"} | Node[1]{name:"
Tomato"}          |
+-----------------------------------------------------------------
2 rows
14 ms

neo4j-sh (?)$
```

Requête complexe (5)



Module Python neo4j

- Module Python neo4j pour interroger la base

  Connexion au serveur et ouverture d'une session

```
1  from neo4j.v1 import GraphDatabase, basic_auth
2
3  driver = GraphDatabase.driver("bolt://localhost", auth=basic_auth("
   neo4j", "neo4j"))
4  session = driver.session()
5
6  print(session)
7  session.close()
```

```
<neo4j.v1.session.Session object at 0x105475550>
```

Exécution d'une requête

- Utilisation de la méthode `run` sur la session

  Exécuter une requête CQL et récupérer un objet `Record`

```
1 result = session.run('MATCH (p:Person) RETURN p.firstname AS
  firstname')
2 for record in result:
3     print(record)
4     print(record['firstname'])
```

```
<Record firstname='Alexis'>
Alexis
<Record firstname='Damien'>
Damien
```

Requête paramétrée

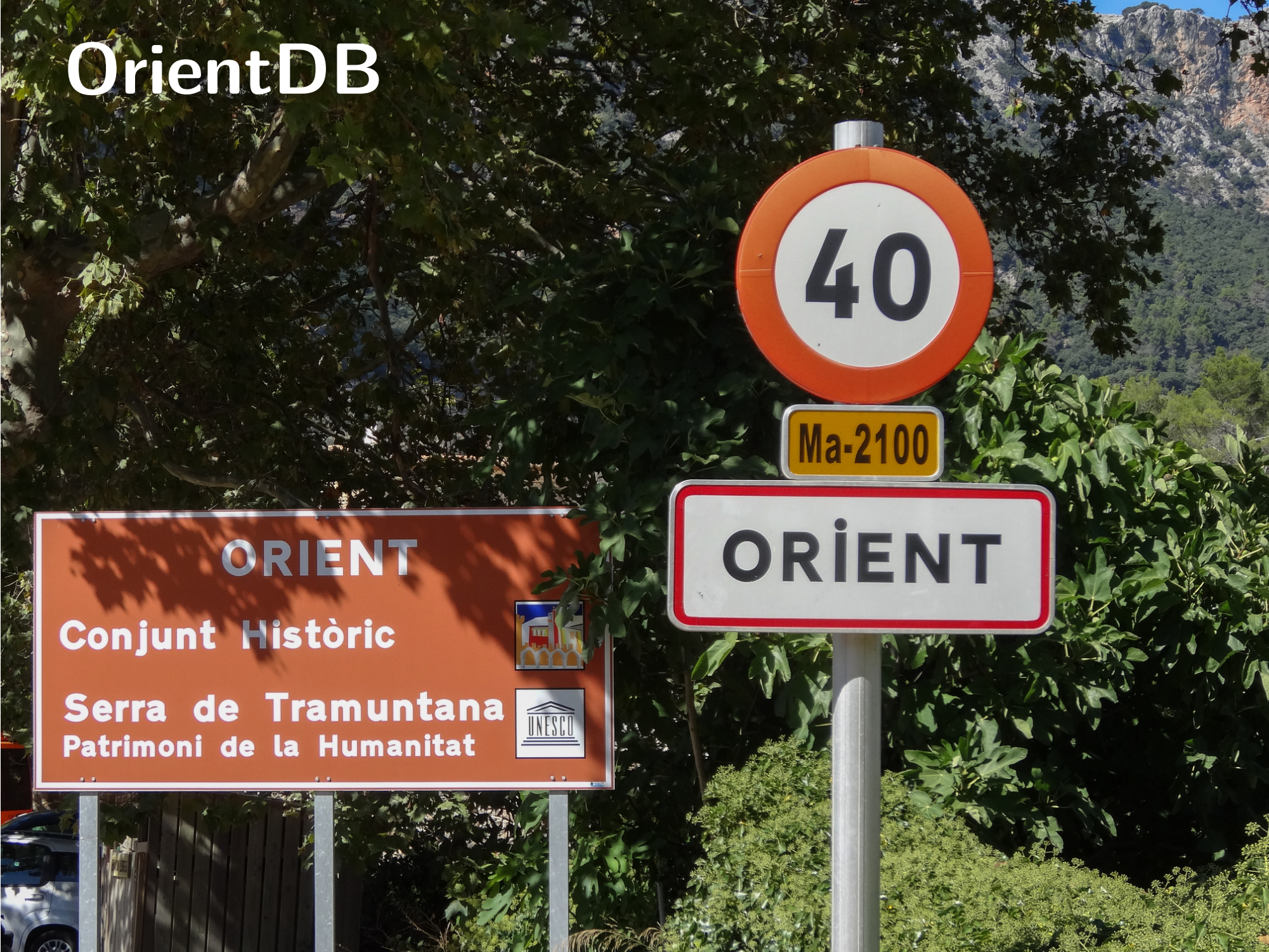
- Une requête paramétrée peut être exécutée plusieurs fois

  Valeurs fournies avec le second paramètre de la méthode `run`

```
1  request = 'MATCH (p:Person {firstname: {name}}) RETURN p'
2  people = ['Alexis', 'Sylvain', 'Damien']
3  for p in people:
4      result = session.run(request, {'name': p})
5      print(p, end=' : ')
6      try:
7          p = result.single()
8          print(p['p'].properties['age'], 'ans.')
9      except ResultError:
10         print('pas trouvé.')
```

```
Alexis : 22 ans.
Sylvain : pas trouvé.
Damien : 24 ans.
```

OrientDB


40
Ma-2100
ORİENT
ORIENT
Conjunt Històric
Serra de Tramuntana
Patrimoni de la Humanitat
UNESCO

OrientDB

- Base de données multi-domaines

  Supporte les graphes, documents, clé-valeur et objets

- Les relations entre entités sont gérées comme avec les graphes

  Supporte les requêtes de type Gremlin

- Transactions ACID supportées par le moteur d'OrientDB

  Et tolérance aux pannes à l'aide de caches

Modèle de données (1)

- Notion de classes pour représenter des enregistrements

  Notion similaire aux tables du modèle relationnel

- Une classe contient plusieurs éléments
  - Une propriété définit une colonne de la classe
  - Des contraintes peuvent être ajoutées aux propriétés
  - Une classe stocke des enregistrements

- Partition des classes en clusters

  Permet notamment de répartir les données physiquement

Modèle de données (2)

- Localisation d'un enregistrement au sein d'un cluster

  #cluster_id :cluster_position

- Établissement de relations entre les classes

  Similaire au modèle relationnel, relation de type `LINK`

- Possibilité de stocker des graphes avec des classes particulières

  Deux classes `V` *et* `E` *toujours présentes à étendre*

Relation

- Stockage du record id de la cible dans la source

  Comme un pointeur entre deux objets en mémoire

- Plusieurs types de relation
  - `LINK` pointe vers un objet
  - `LINKSET` pointe vers plusieurs objets (ensemble)
  - `LINKLIST` pointe vers plusieurs objets (liste)
  - `LINKMAP` pointe vers plusieurs objets (dictionnaire)

Installation de OrientDB

- OrientDB est un programme développé en Java
- Plusieurs programmes proposés après installation
  - `orientdb` permet de lancer plusieurs commandes de gestion
    - `orientdb start` démarre le serveur
    - `orientdb stop` arrête le serveur
    - `orientdb status` récupère le statut du serveur
  - `orientdb-console` propose un client en ligne de commande

Lancement du serveur

- Lancement du serveur et vérification de la connexion

  Indication immédiate de si un serveur a été trouvé

```
& orientdb start
```

```
& orientdb-console

OrientDB console v.2.2.5 (build 2.2.
x@r393af9c5a3e4a4408440a9376283a26d2d3d3c7b; 2016-07-20
06:03:46+0000) www.orientdb.com
Type 'help' to display all the supported commands.
Installing extensions for GREMLIN language v.2.6.0

orientdb>
```

Configuration d'un utilisateur

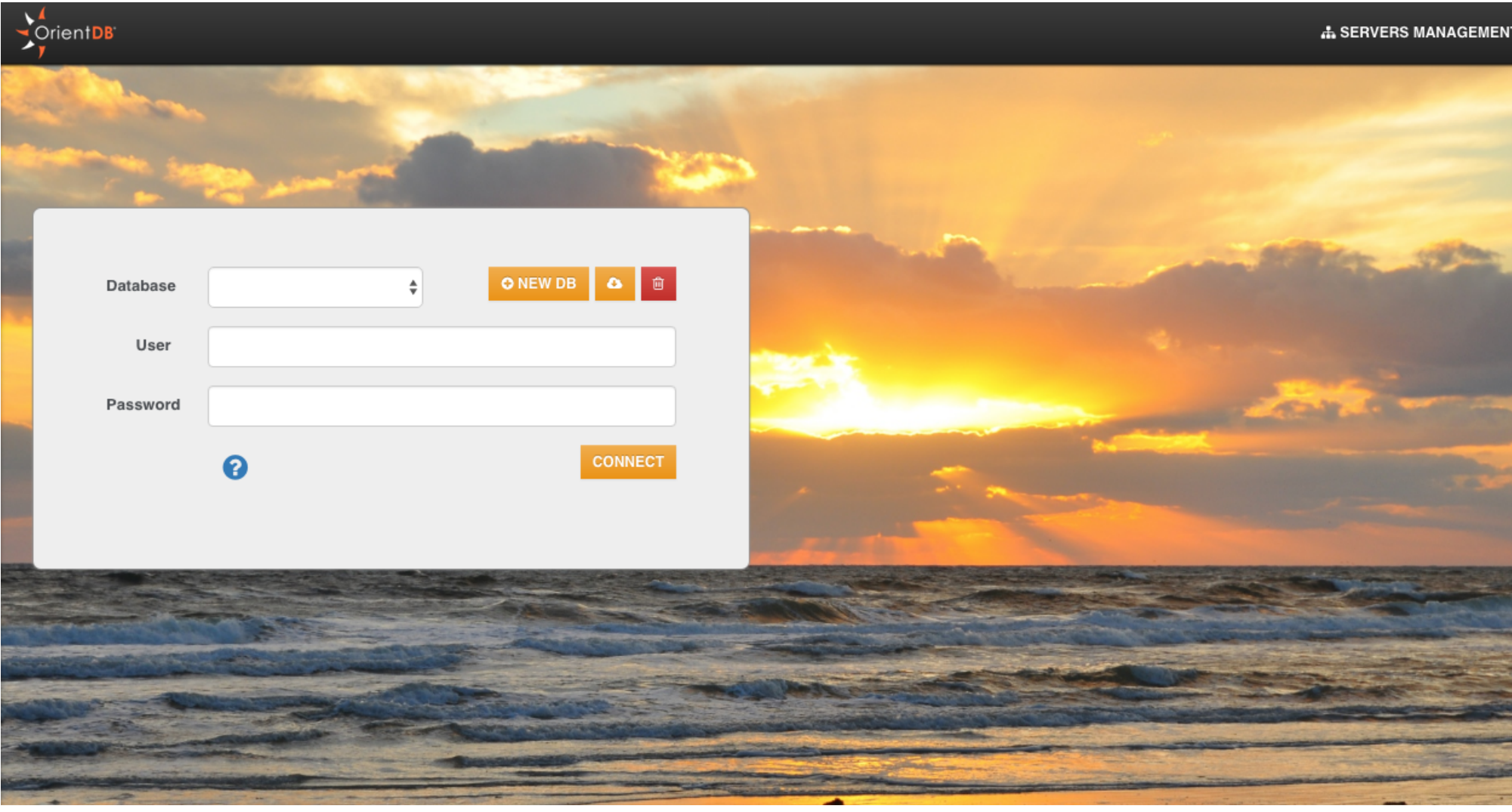
- Ajout d'un utilisateur dans `orientdb-server-config.xml`

Puis création d'une connexion avec la commande `CONNECT`

```
<user resources="*" password="admin" name="admin"/>
```

```
orientdb> CONNECT remote:localhost admin admin

Connecting to remote Server instance [remote:localhost] with user
 'admin'...OK

orientdb {server=remote:localhost/}>
```

Browser OrientDB



Connexion à une base de données

- Connexion à une base de données avec la commande `CONNECT`

Déconnexion du serveur pour connexion à la base de données

```
orientdb {server=remote:localhost/}> LIST DATABASES

Found 1 databases:

* myschool (plocal)

orientdb {server=remote:localhost/}> CONNECT REMOTE:localhost/
myschool admin admin

Disconnecting from remote server [remote:localhost/]...
OK
Connecting to database [REMOTE:localhost/myschool] with user '
admin'...OK

orientdb {db=myschool}>
```

Création d'une classe

- Création d'une nouvelle classe avec `CREATE CLASS`

  Ensuite ajout éventuel de propriété avec `CREATE PROPERTY`

- Propriétés pas obligatoires, sauf pour index ou contraintes

  OrientDB travaille en mode schéma-mixte

```
orientdb {db=myschool}> CREATE CLASS student

Class created successfully. Total classes in database now: 12.

orientdb {db=myschool}> CREATE PROPERTY student.firstname STRING

Property created successfully with id=1.
```

Information sur une classe

- Récupération d'information sur une classe avec `INFO CLASS`

  Information sur la distribution et sur les propriétés et contraintes

```
orientdb {db=myschool}> INFO CLASS student

CLASS 'student'

Records..............: 0
Default cluster......: student (id=21)
Supported clusters...: student(21), student_1(22), student_2(23),
 student_3(24)
Cluster selection....: round-robin
Oversize.............: 0.0

PROPERTIES
+----+---------+------+-----------------+---------+--------+-----
|#   |NAME     |TYPE  |LINKED-TYPE/CLASS|MANDATORY|READONLY|NOT-N
+----+---------+------+-----------------+---------+--------+-----
|0   |firstname|STRING|                 |false    |false   |false
+----+---------+------+-----------------+---------+--------+-----

orientdb {db=myschool}>
```

Ajout d'un enregistrement

- Ajout d'un nouvel enregistrement avec `INSERT INTO`

  Information sur la distribution et sur les propriétés et contraintes

- L'enregistrement est automatiquement placé sur un cluster

  Possibilité de spécifier le cluster sur lequel le placer

```
orientdb {db=myschool}> INSERT INTO student SET firstname='Alexis
', age=22

Inserted record 'student#21:0{firstname:Alexis,age:22} v1' in
0,090000 sec(s).

orientdb {db=myschool}>
```

Récupérer un enregistrement

- Récupérer un enregistrement avec `DISPLAY RECORD`

  Possibilité d'utiliser `LOAD RECORD #21:0` (plus global)

```
orientdb {db=myschool}> BROWSE CLASS student

+----+-----+-------+---------+----+
|#   |@RID |@CLASS |firstname|age |
+----+-----+-------+---------+----+
|0   |#21:0|student|Alexis   |22  |
+----+-----+-------+---------+----+

orientdb {db=myschool}> DISPLAY RECORD 0

DOCUMENT @class:student @rid:#21:0 @version:1
+----+---------+------+
|#   |NAME     |VALUE |
+----+---------+------+
|0   |firstname|Alexis|
|1   |age      |22    |
+----+---------+------+

orientdb {db=myschool}>
```

Création d'un graphe (1)

- Création de classes de type nœud et arêtes

  Une classe peut étendre une autre avec la clause `EXTENDS`

```
orientdb {db=social}> CREATE CLASS person EXTENDS V

Class created successfully. Total classes in database now: 12.

orientdb {db=social}> CREATE CLASS food EXTENDS V

Class created successfully. Total classes in database now: 13.

orientdb {db=social}> CREATE CLASS likes EXTENDS E

Class created successfully. Total classes in database now: 14.

orientdb {db=social}>
```

Création d'un graphe (2)

- Ajout de nœud et arêtes avec `CREATE VERTEX/EDGE`

  Même syntaxe que l'ajout d'enregistrements

```
orientdb {db=social}> CREATE VERTEX person SET firstname="Alexis"
Created vertex 'person#21:0{firstname:Alexis} v1' in 0,002000 sec
(s).

orientdb {db=social}> CREATE VERTEX food SET name="Tomato"
Created vertex 'food#25:0{name:Tomato} v1' in 0,003000 sec(s).

orientdb {db=social}> CREATE EDGE likes FROM (SELECT FROM person
WHERE firstname="Alexis") TO (SELECT FROM food WHERE name="Tomato
")

Created edge '[likes#29:0{out:#21:0,in:#25:0} v1]' in 0,148000
sec(s).

orientdb {db=social}>
```

Interrogation d'un graphe

- Récupération des arêtes adjacentes avec `IN/OUT/BOTH`

  Transformation des identifiants en enregistrements avec `EXPAND`

```
orientdb {db=social}> SELECT OUT() FROM person WHERE firstname="
Alexis"

+----+-------+
|#   |OUT    |
+----+-------+
|0   |[#25:0]|
+----+-------+

1 item(s) found. Query executed in 0.034 sec(s).

orientdb {db=social}> SELECT EXPAND(OUT()) FROM person WHERE
firstname="Alexis"

+----+-----+------+------+--------+
|#   |@RID |@CLASS|name  |in_likes|
+----+-----+------+------+--------+
|0   |#25:0|food  |Tomato|[#29:0] |
+----+-----+------+------+--------+

1 item(s) found. Query executed in 0.004 sec(s).
```

Module Python pyorient

- Module Python pyorient pour interroger la base

  Connexion au serveur et authentification d'un utilisateur

```
1  import pyorient
2
3  client = pyorient.OrientDB('localhost', 2424)
4  session = client.connect('admin', 'admin')
5
6  print(client.db_list())
```

```
{{'databases': {'myschool': 'plocal:/usr/local/var/db/orientdb/
myschool', 'social': 'plocal:/usr/local/var/db/orientdb/social
'}}}
```

Exécution d'une requête

- Utilisation de la méthode `command` sur la session

  Après avoir ouvert une base de données avec `open`

```
1  import pyorient
2
3  client = pyorient.OrientDB('localhost', 2424)
4  session = client.db_open('myschool', 'admin', 'admin')
5
6  result = client.query('SELECT FROM student')
7  for student in result:
8      print(student)
9      print(student.firstname)
```

```
{'@student':{'firstname': 'Alexis', 'age': 22},'version':1,'rid
':'#21:0'}
Alexis
```

Langage de requête

Gremlin (1)

- Langage pour décrire des traversée de graphes

  Fait partie du projet Apache TinkerPop™

- Utilisé avec deux types de bases de données
  - Bases orientée-graphe (OLTP) : OrientDB...
  - Manipulation de graphes (OLAP) : Apache Giraph, Hadoop...

- Langage de type procédural avec sa machine virtuelle

  Une requête est un programme décrivant la traversée à faire

Gremlin (2)

- Simple traversée d'un graphe pour récupérer des informations

  On part des nœuds et on effectue des opérations dessus

- Jeu de données MovieLens avec rating de films

```
user--rated[stars:0-5]-->movie
user--occupation-->occupation
movie--category-->category
```

```
1  # Pour chaque noeud, on prend le label, on les regroupe et compte
2  gremlin> g.V().label().groupCount()
3  ==>[occupation:21, movie:3883, category:18, user:6040]
4
5  # On prend les films, et on cherche la plus petite année
6  gremlin> g.V().hasLabel('movie').values('year').min()
7  ==>1919
8
9  # On prend Die Hard et on fait la moyenne des étoiles sur les arêtes entrantes
10 gremlin> g.V().has('movie','name','Die Hard').inE('rated').values('stars').
   mean()
11 ==>4.121848739495798
```

Gremlin (3)

- Requête plus complexe en suivant des relations dans le graphe

  Fonction `out` pour suivre une relation

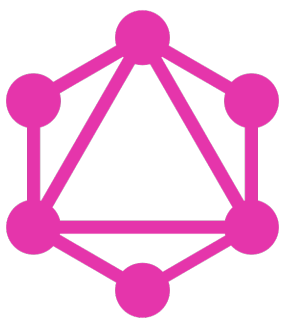
```
1  # Nom des amis des amis de Gremlin
2  g.V().has("name","gremlin").
3    out("knows").
4    out("knows").
5    values("name")
6
7  # Noms des managers dans la chaine de Gremlin au CEO
8  g.V().has("name","gremlin").
9    repeat(in("manages")).until(has("title","ceo")).
10   path().by("name")
```

GraphQL (1)

- Langage de requête développé par Facebook et publié en 2015

  Alternative aux architectures REST pour services web

- Évite de récupérer trop ou trop peu de données
  - Le client défini les données voulues de la part du serveur
  - Langage fortement typé
  - Une seule requête plutôt que d'appeler plusieurs URLs



GraphQL (2)

- Interrogation d'une API en écrivant une requête en GraphQL

  Par exemple requêtes sur la SWAPI (Star Wars API)

- Équivalent de l'appel https://swapi.co/api/films/

  En filtrant et ne gardant que l'attribut `title`

```
{
  allFilms {
    films {
      title
    }
  }
}
```

GraphQL (2)

```
GET /api/films/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
{
    "count": 7,
    "next": null,
    "previous": null,
    "results": [
        {
            "title": "A New Hope",
            "episode_id": 4,
            "opening_crawl": "It is a period of civil war.\r\nRebel spaceships, striking\r\nfrom a
            "director": "George Lucas",
            "producer": "Gary Kurtz, Rick McCallum",
            "release_date": "1977-05-25",
            "characters": [
                "https://swapi.co/api/people/1/",
                "https://swapi.co/api/people/2/",
                "https://swapi.co/api/people/3/",
                "https://swapi.co/api/people/4/",
                "https://swapi.co/api/people/5/",
                "https://swapi.co/api/people/6/",
                "https://swapi.co/api/people/7/",
                "https://swapi.co/api/people/8/",
```

GraphQL (2)


GraphiQL
Prettify
Docs
{
allFilms {
films {
title
}
}
}
QUERY VARIABLES
{
"data": {
"allFilms": {
"films": [
{
"title": "A New Hope"
},
{
"title": "The Empire Strikes Back"
},
{
"title": "Return of the Jedi"
},
{
"title": "The Phantom Menace"
},
{
"title": "Attack of the Clones"
},
{
"title": "Revenge of the Sith"
}
]
}
}
}

GraphQL (3)

- Fusion des réponses de plusieurs *« requêtes classiques »*

  Avec une requête GraphQL qui imbrique des demandes

- Équivalent de l'appel https://swapi.co/api/people/xxx/

  Et des appels aux films https://swapi.co/api/films/yyy/

```
{
  person(id: "cGVvcGxlOjE=") {
    name
    height
    filmConnection {
      films {
        title
      }
    }
  }
}
```

GraphQL (3)

```
GET /api/people/1/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "name": "Luke Skywalker",
    "height": "172",
    "mass": "77",
    "hair_color": "blond",
    "skin_color": "fair",
    "eye_color": "blue",
    "birth_year": "19BBY",
    "gender": "male",
    "homeworld": "https://swapi.co/api/planets/1/",
    "films": [
        "https://swapi.co/api/films/2/",
        "https://swapi.co/api/films/6/",
        "https://swapi.co/api/films/3/",
        "https://swapi.co/api/films/1/",
        "https://swapi.co/api/films/7/"
    ],
    "species": [
        "https://swapi.co/api/species/1/"
    ],
    "vehicles": [
```

GraphQL (3)


GraphiQL
Prettify
Docs
{
person(id: "cGVvcGxlOjE=") {
name
height
filmConnection {
films {
title
}
}
}
}
QUERY VARIABLES
{
"data": {
"person": {
"name": "Luke Skywalker",
"height": 172,
"filmConnection": {
"films": [
{
"title": "A New Hope"
},
{
"title": "The Empire Strikes Back"
},
{
"title": "Return of the Jedi"
},
{
"title": "Revenge of the Sith"
}
]
}
}
}
}

Crédits

- Photos des logos depuis Wikipédia
- https://www.flickr.com/photos/jennifer-stylls/8012538039
- https://fr.wikipedia.org/wiki/Fichier:Sw-horz-w3c-v.svg
- https://fr.wikipedia.org/wiki/Fichier:RDF_example.svg
- https://en.wikipedia.org/wiki/File:DBpediaLogo.svg
- https://www.flickr.com/photos/neotechnology/9024811631
- https://www.flickr.com/photos/michaeljohnbutton/10049404576
- https://www.flickr.com/photos/10422334@N08/5250281675
- https://en.wikipedia.org/wiki/File:Gremlin_(programming_language).png
- https://en.wikipedia.org/wiki/File:GraphQL_Logo.svg