Introduction

Please find two other learning partners,

- form a standing group and
- tell them what you already know about
  - graphs,
  - graph databases and
  - Neo4j.

Graph Data - Modelling and Quering

with Neo4j and Cypher

Iryna Feuerstein

Open Data Science Conference

12 October 2017

Agenda

- What are graphs?
  - Definition
  - Typical use cases
  - Not so typical use cases
- Starting with Neo4j and Cypher
  - Starting the database
  - Brief look at the configuration
  - CRUD operations with Cypher
  - Node operations
  - Relation operations
- Quering for paths and patterns
- Using graph algorithms
  - apoc library

Graph

Definition

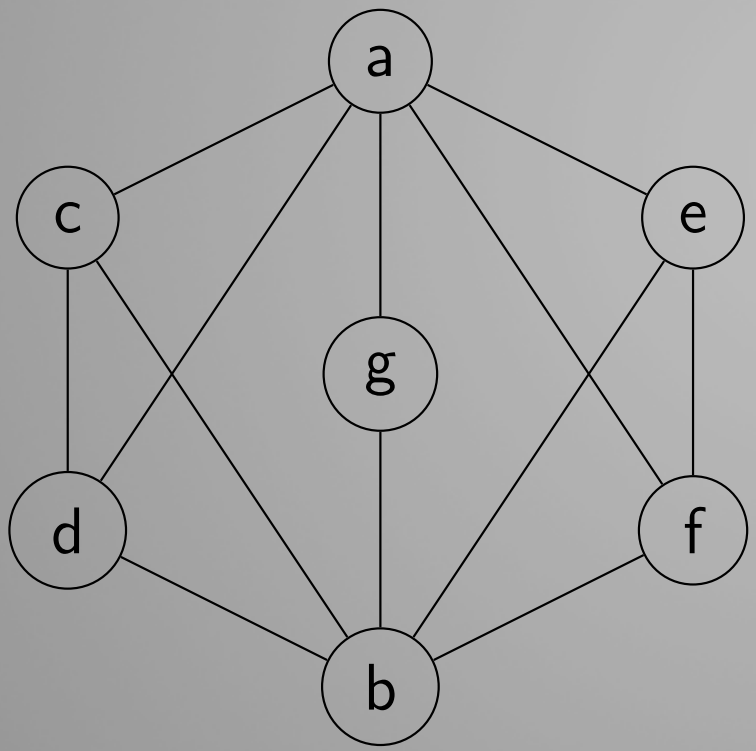
Graph is an ordered pair $G = (V, E)$ comprising a set V of *vertices*, *nodes* or *points* together with a set E of *edges*, *arcs* or *lines*, which are 2-element subsets of V.[1]

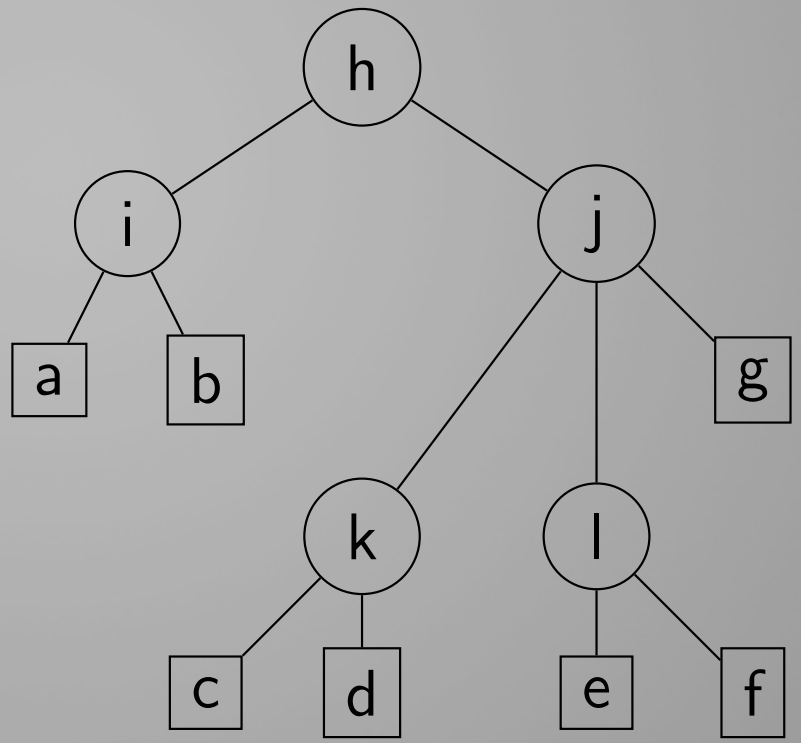
[1] en.wikipedia.org/wiki/Graph_(discrete_mathematics)

Graph

Definition

Graph is an ordered pair $G = (V, E)$ comprising a set V of *vertices*, *nodes* or *points* together with a set E of *edges*, *arcs* or *lines*, which are 2-element subsets of V.[1]



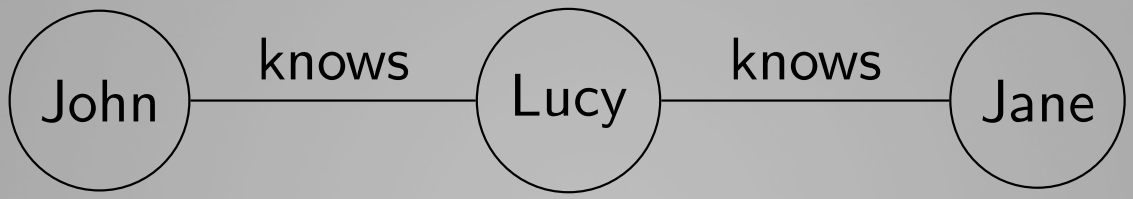


[1] en.wikipedia.org/wiki/Graph_(discrete_mathematics)

Use Cases

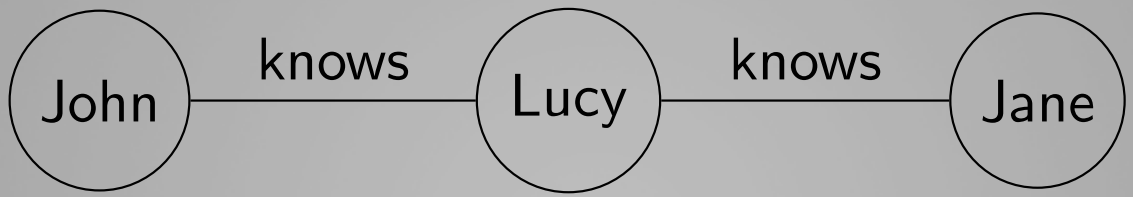
Use Cases

- Networks
  - Social networks

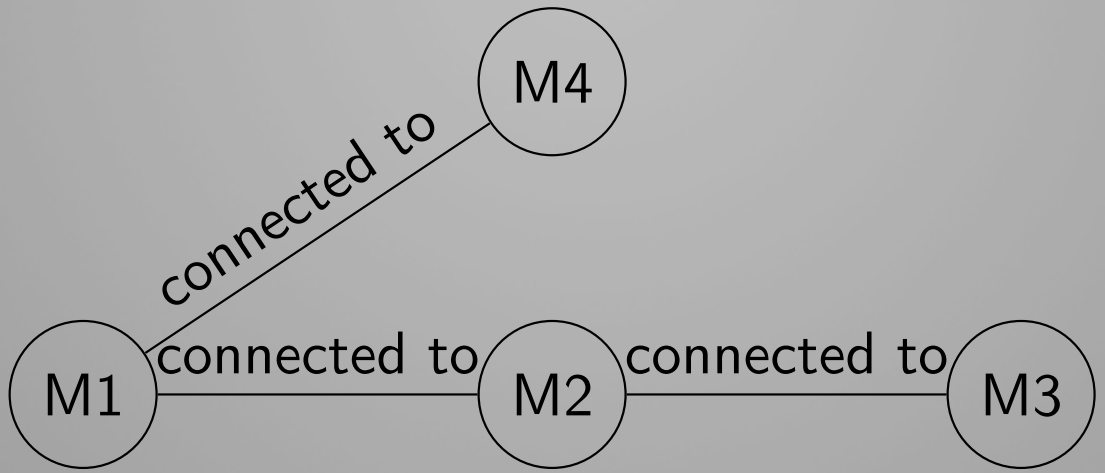


Use Cases

- Networks
  - Social networks

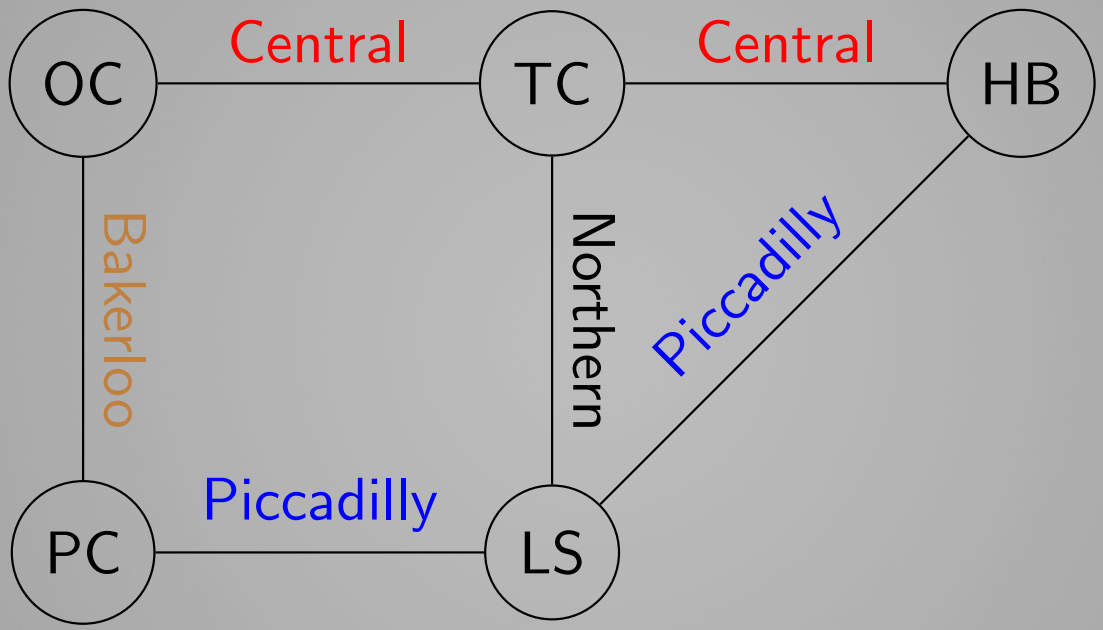


  - Computer networks



Use Cases

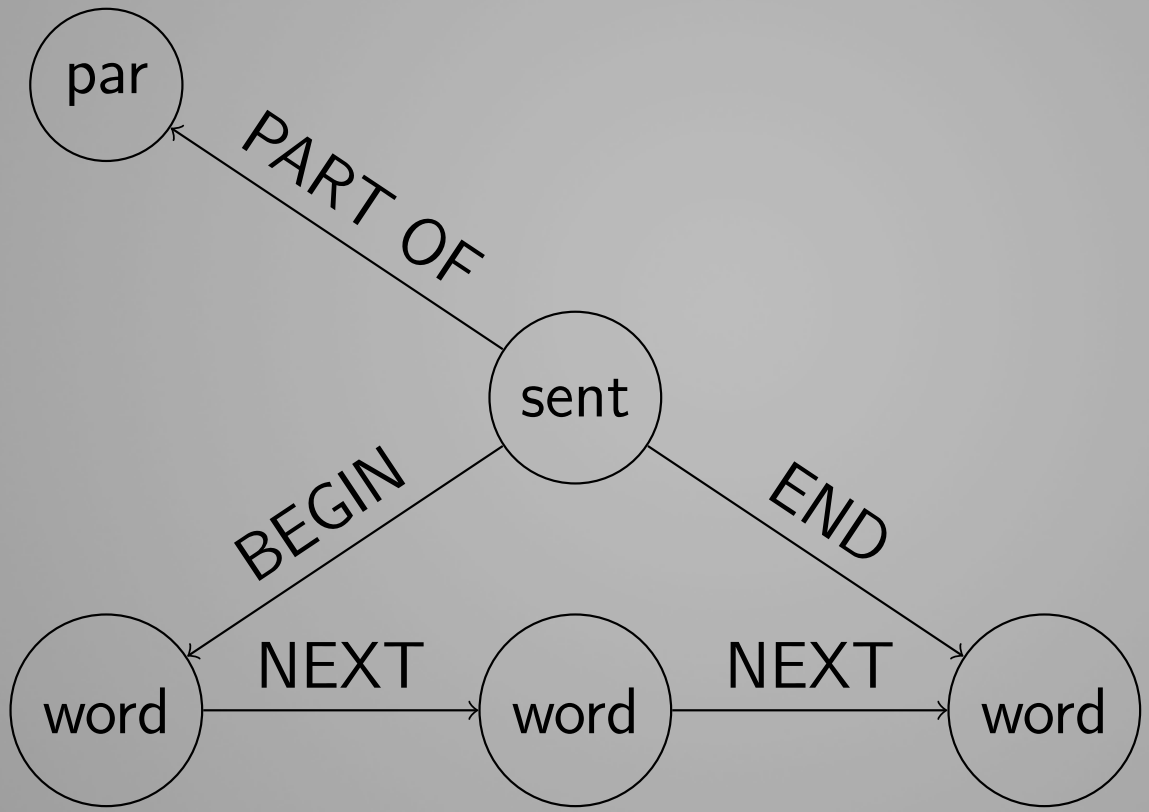
- Networks
  - Transport networks



OC = Oxford Circus
TC = Tottenham Court Road
HB = Holborn
LS = Leicester Square
PC = Piccadilly Circus

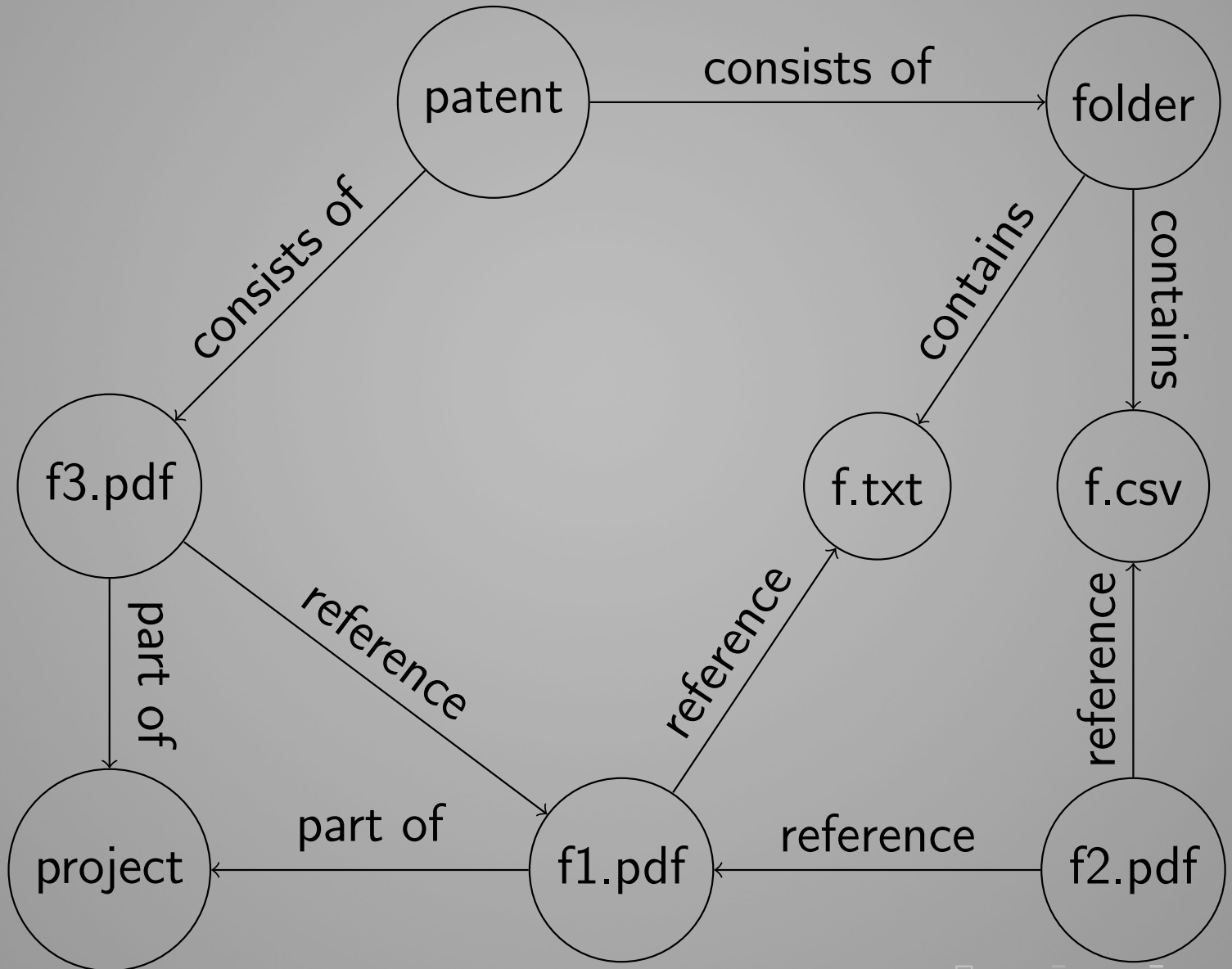
Use Cases

- Natural Language Processing



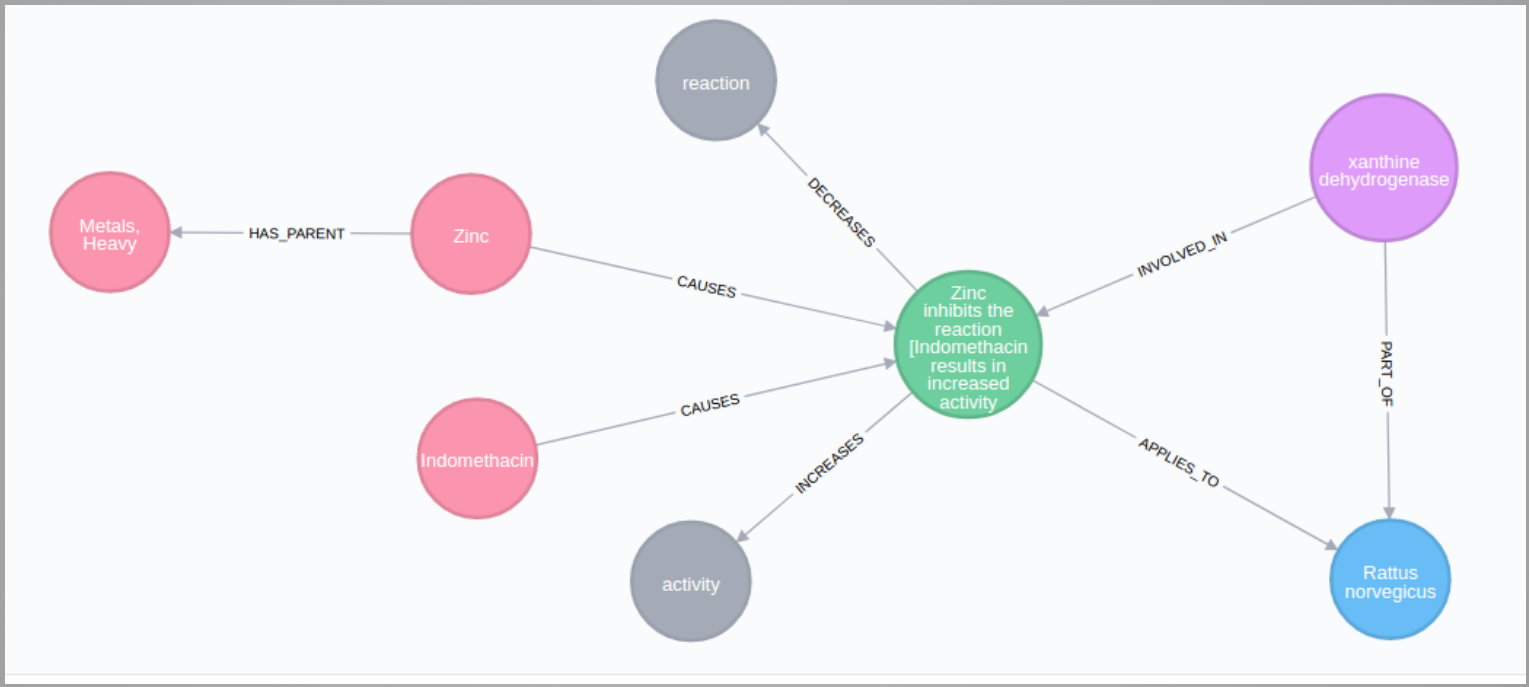
Use Cases

- Document management



Use Cases

- Biochemistry / Genomics



[1] http://ctdbase.org/

Installation

- Find the right installation file for your OS at `neo4j-training-files/neo4j` on the flash drive and install the software.

Installation

- Find the right installation file for your OS at `neo4j-training-files/neo4j` on the flash drive and install the software.
- Copy both JAR-files from the `neo4j-training-files/plugins` directory into `NEO4J_HOME/plugins` directory of your Neo4j installation.

Installation

- Find the right installation file for your OS at `neo4j-training-files/neo4j` on the flash drive and install the software.
- Copy both JAR-files from the `neo4j-training-files/plugins` directory into `NEO4J_HOME/plugins` directory of your Neo4j installation.
- Replace the `NEO4J_HOME/conf/neo4j.conf` configuration file with the one found on the flash drive at `neo4j-training-files/conf/neo4j.conf`.

Installation

- Find the right installation file for your OS at `neo4j-training-files/neo4j` on the flash drive and install the software.
- Copy both JAR-files from the `neo4j-training-files/plugins` directory into `NEO4J_HOME/plugins` directory of your Neo4j installation.
- Replace the `NEO4J_HOME/conf/neo4j.conf` configuration file with the one found on the flash drive at `neo4j-training-files/conf/neo4j.conf`.
- Copy the `neo4j-training-files/data/odsc.db` folder into your `NEO4J_HOME/data/databases/` directory

Starting Neo4j

- Start the database with `NEO4J_HOME/bin/neo4j start`

Starting Neo4j

- Start the database with
  `NEO4J_HOME/bin/neo4j start`
- Go to `http://localhost:7474` within you browser

Demonstration

Important configuration entries

dbms.active_database=odsc.db
dbms.security.auth_enabled=false
dbms.security.procedures.unrestricted=algo.*,apoc.*
apoc.import.file.enabled=true

Live coding session

- create node
  CREATE (c:Chemical {name: 'Helium'}) RETURN c
- update node
  MERGE (c:Chemical {name: 'Helium'}) SET c.symbol = 'He' RETURN c
- delete node
  - without relations
    MATCH (c:Chemical {name:'Helium'}) DELETE c
    MATCH (c:Chemical)
    WHERE c.name = 'Helium'
    DELETE c
  - with existing relations
    MATCH (c:Chemical {name:'Helium'})
    DETACH DELETE c

Live coding session

- create relation
  - between new nodes
    CREATE (c:Chemical chemicalName:'Helium')-[:BELONGS_TO]-¿(g:ChemicalGroup groupName:'Noble gases') RETURN c,g
  - between existing nodes
    MATCH (g:ChemicalGroup groupName:'Noble gases'), (p:ChemicalGroup groupName:'Gases') CREATE (g)-[:HAS_PARENT]-¿(p) RETURN g,p
- update relation
  MATCH ()-[r:BELONGS_TO]-() SET r.updateTime = timestamp() RETURN r
- delete relation
  MATCH ()-[r:BELONGS_TO]-() DELETE r

Live coding session

Examples:

- MATCH (g:Gene) WHERE g.geneSymbol = 'CTSD' RETURN g
- MATCH (g:Gene)¡-[:ASSOCIATED_WITH]-(d:Disease) WHERE g.geneSymbol = 'CTSD' RETURN g, d
- MATCH (g:Gene)¡-[:ASSOCIATED_WITH]-(d:Disease) WHERE g.geneSymbol = 'CTSD' RETURN g, count(d)
- MATCH (g:Gene)¡-[:ASSOCIATED_WITH]-(d:Disease) WITH g, count(d) as diseases WHERE diseases ¿ 50 RETURN g.geneName, g.geneSymbol, diseases ORDER BY diseases DESC
- MATCH (g:Gene)¡-[:ASSOCIATED_WITH]-(d:Disease)-[:ASSOCIATED_WITH]-(otherGene:Gene) WHERE g.geneSymbol = 'CTSD' AND d.diseaseName = 'Osteoarthritis' RETURN otherGene.geneName, otherGene.geneSymbol

Live coding session

- MATCH p = (c:Chemical)-[*2]-(d:Disease) where d.diseaseName STARTS WITH 'Osteo' RETURN p LIMIT 20
- MATCH (c:Chemical)¡-[:HAS_PARENT*3..4]-(descendant:Chemical) WITH c, count(descendant) AS descendants , collect(descendant.chemicalName) as names ORDER BY descendants DESC LIMIT 10 RETURN c.chemicalName, names[1..10], descendants
- MATCH (c:Chemical) WHERE c.chemicalName = 'Zinc Acetate' MATCH (d:Disease) WHERE d.diseaseName = 'Alzheimer Disease' MATCH p = (c)-[*1..3]-(d) RETURN p LIMIT 20

Calling procedures

- CALL db.schema
- CALL dbms.procedures
- call dbms.functions
- CALL apoc.help('dijkstra')

Live coding session

Definition

In a connected graph, the normalized closeness centrality (or closeness) of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes.[2]

Centrality Example

MATCH (node:Chemical) WHERE node.chemicalName CONTAINS 'Vitamin' WITH collect(node) AS nodes CALL apoc.algo.closeness(['HAS_PARENT'],nodes,'BOTH') YIELD node, score RETURN node, score ORDER BY score DESC

[2] https://en.wikipedia.org/wiki/Centrality#Closeness_centrality