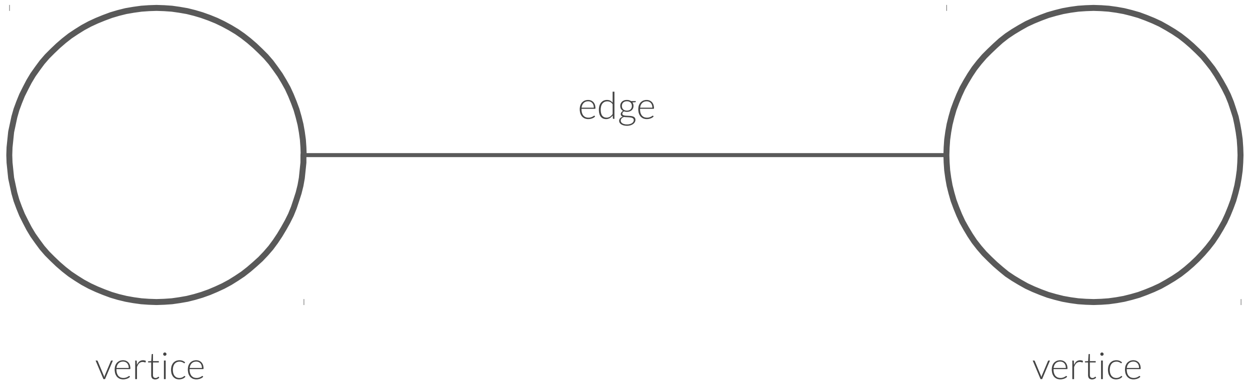
graph databases

idwall tech talk

05 09 18

graphs

graphs



graph theory

graph theory

- ordered pair *G = (V, E)*, where V is a set of **vertices** and E is a set of **edges**
- undirected graph
- directed graph
- properties
  - clustering coefficient
  - betweenness centrality
- algorithms

examples

- chess
- roads
- computer networks
- social networks
- internet
- artificial neural networks
- ...

graph databases

graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn’t expect

graph databases

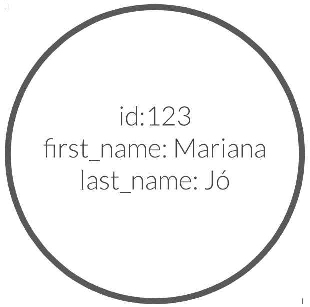
- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn't expect

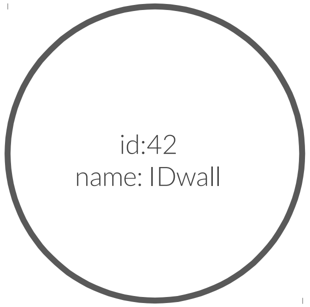
- graph dbs
  - are about *relationships*

graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn't expect

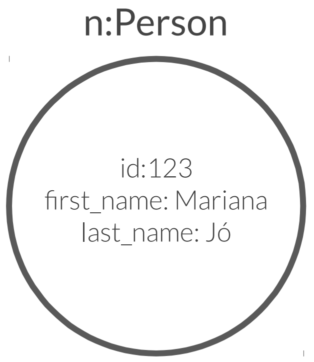
- graph dbs
  - are about *relationships*
  - have nodes with
    - **properties**, just like `{key: "value"}`

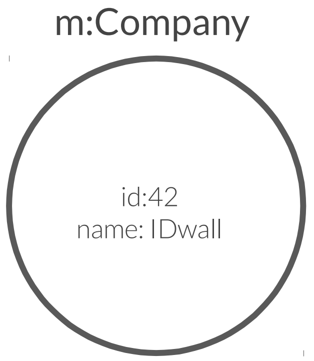




graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn't expect
- graph dbs
  - are about *relationships*
  - have nodes with
    - **properties**, just like `{key: "value"}`
    - **labels**, to define types of things

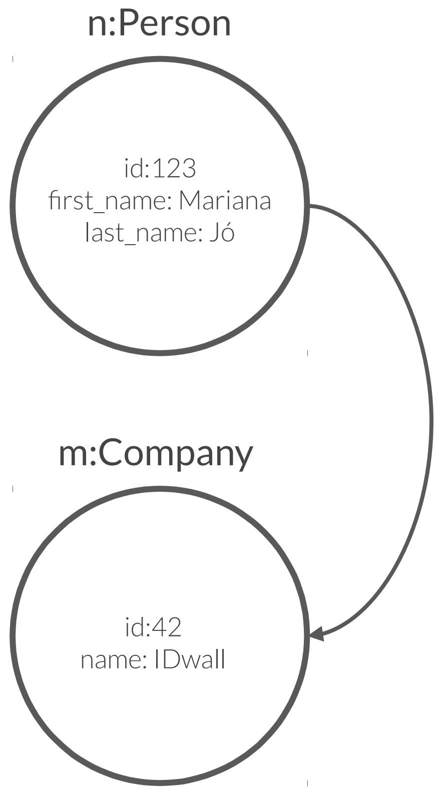




graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn't expect

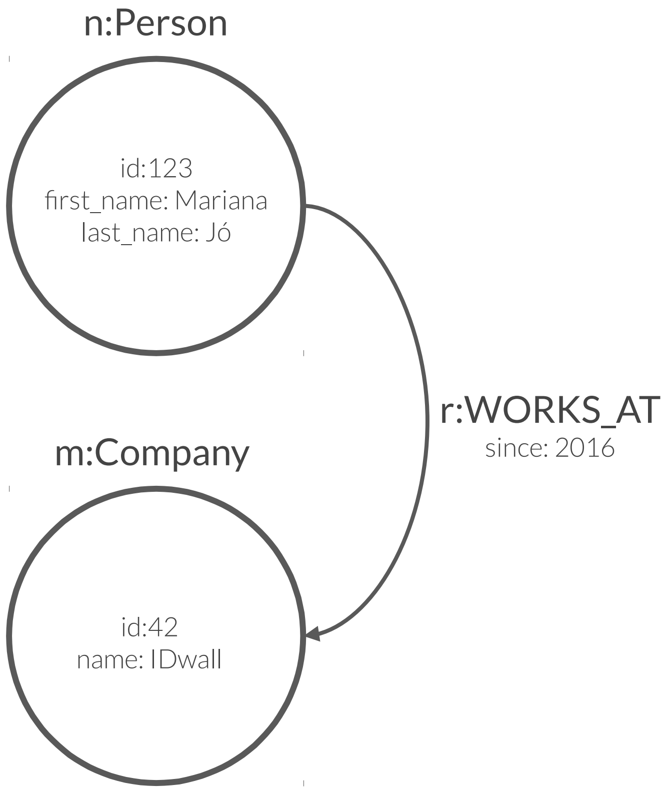
- graph dbs
  - are about *relationships*
  - have nodes with
    - **properties**, just like `{key: "value"}`
    - **labels**, to define types of things
  - they are connected by **relationships**



graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn't expect

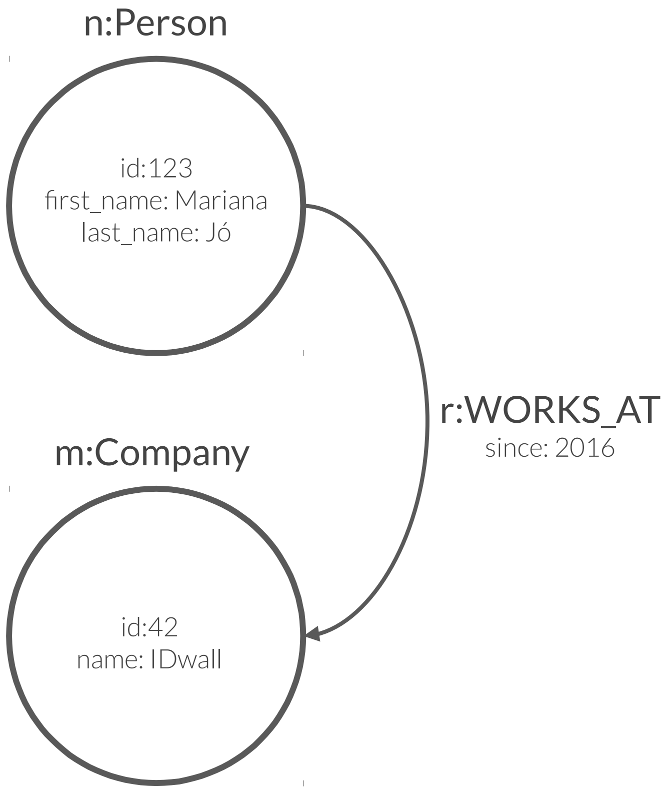
- graph dbs
  - are about *relationships*
  - have nodes with
    - **properties**, just like `{key: "value"}`
    - **labels**, to define types of things
  - they are connected by **relationships**
    - which can have **type**, **direction** and **properties**



graph databases

- relational dbs
  - are about *constraints*
  - they get rigid when representing relationships
  - indirect relationships are hard
  - make it hard to answer things you didn’t expect

- graph dbs
  - are about *relationships*
  - have nodes with
    - **properties**, just like `{key: “value”}`
    - **labels**, to define types of things
  - they are connected by **relationships**
    - which can have **type**, **direction** and **properties**
  - let you add relationships at your will
  - graph dbs are *simpler*



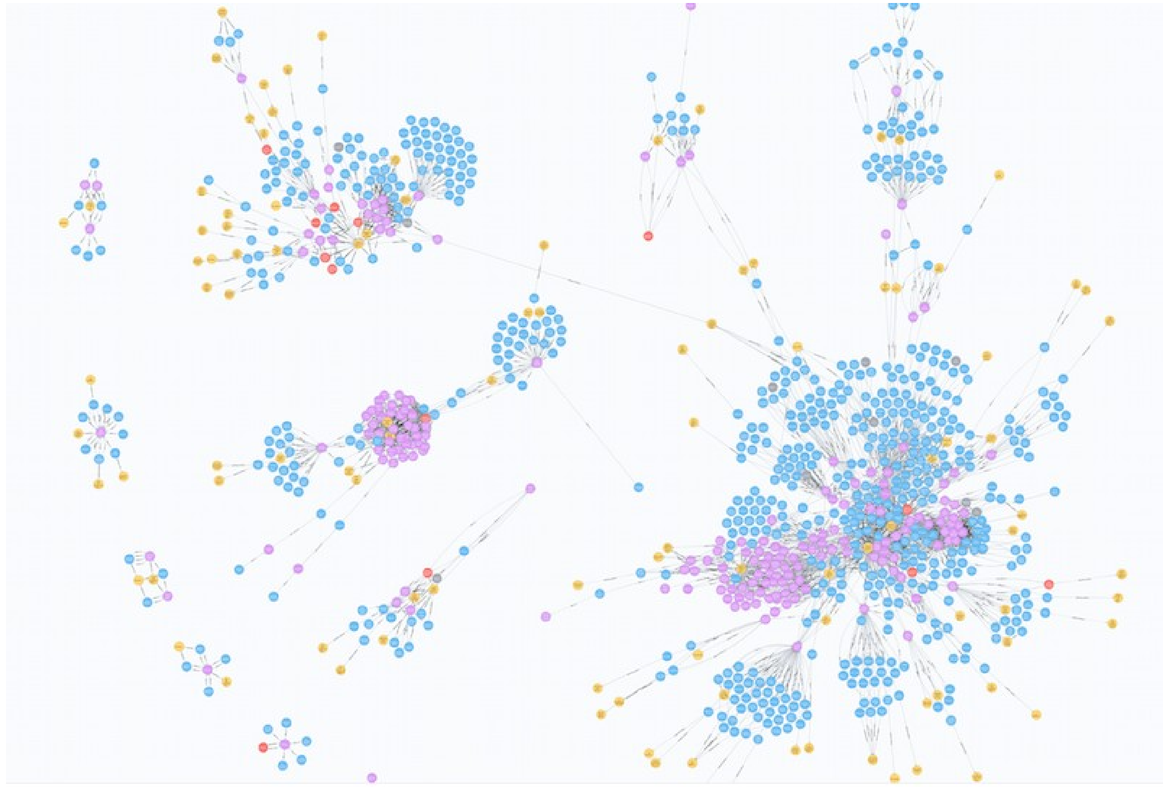
graphdb examples

- social networks (twitter, linkedin, facebook, …)
- recommendation systems
- logistics
- fraud detection
- …

neo4j

neo4j

- desktop demo



cypher

cypher

- `(nodes)`
  - `()   |   (:Person)   |   (p:Person)`
  - `p.name`

cypher

- `(nodes)`
  - `()    |    (:Person)    |    (p:Person)`
  - `p.name`

- `[relationships]`
  - `-->    |    -[:WORKS_AT]->  |  -[r:WORKS_AT]->`
  - `-[{since:2016}]->`

cypher

- (nodes)
  - `()   |   (:Person)   |   (p:Person)`
  - `p.name`

- [relationships]
  - `-->   |   -[:WORKS_AT]->  |  -[r:WORKS_AT]->`
  - `-[{since:2016}]->`

```
MATCH (node:Label) RETURN node.property

MATCH (node1:Label1)-->(node2:Label2)
WHERE node1.propertyA = {value}
RETURN node2.propertyA, node2.propertyB
```

cypher

- (nodes)
  - `()   |   (:Person)   |   (p:Person)`
  - `p.name`

- [relationships]
  - `-->   |   -[:WORKS_AT]->  |  -[r:WORKS_AT]->`
  - `-[{since:2016}]->`

```
MATCH (p:Person)-->(c:Company)
WHERE p.first_name = {“Mariana”}
RETURN c.name
```

cypher

- `(nodes)`
  - `()   |   (:Person)   |   (p:Person)`
  - `p.name`

- `[relationships]`
  - `-->   |   -[:WORKS_AT]->  |  -[r:WORKS_AT]->`
  - `-[{since:2016}]->`

```
MATCH (p:Person {name:”Mariana”})-->(c:Company)
RETURN c.name
```

demo!

references & stuff

- *Graph databases will change your freaking life.* https://www.youtube.com/watch?v=3vIeFxDGoEs
- *Build Intelligent Fraud Prevention with ML and Graphs.* https://www.youtube.com/watch?v=QkQLIDFIkyc
- *Stop complex fraud in its tracks.* https://www.youtube.com/watch?v=kSZHFIBDIfM
- *Study: IDwall in graphs.* https://idwall.atlassian.net/wiki/spaces/~mariana/pages/31686703/Study+IDwall+in+graphs
- *Analyzing Panama Papers with Neo4j.* https://neo4j.com/blog/analyzing-panama-papers-neo4j/
- *Navegando por grafos com Python.* https://www.youtube.com/watch?v=BWp_2xeVzoc
- *Graph Databases Book*, Ian Robinson, Jim Webber and Emil Eifrem O'Reilly. https://neo4j.com/graph-databases-book/?ref=home
- *Graph of Thrones.* https://www.lyonwj.com/2016/06/26/graph-of-thrones-neo4j-social-network-analysis/

flw vlw