Databses II: Graph Databases

Igor Wojnicki

Department of Applied Computer Science, AGH UST

May 19, 2022

Model

- nodes
- relationships: a directed edge, between 2 vertices
- properties: an arbitrary set of key-value pairs at edges or vertices
- labels: one or more at edges or vertices (edges MUST be labeled)

$G(V, E, L, A, lab_V(), lab_E(), att_V(), att_E())$

Represent a graph in RDBMS?

- vertex and edge tables
- labels/attributes
- How to find a path: recursive SQL...

Query languages?

- Graph Transformations / Graph Grammars
- SPARQL (RDF)
- Cypher (Neo4j)

Cypher: ASCII-Art

`http://neo4j.com/docs/cypher-refcard/3.1/`

Finding a pattern/path/subgraph

```
(sensor)-[:IN]->(segment)-[:HAS]->(config)-[:REGARDS]->(lamp)
```

Variables: `sensor`, `segment`, `config`, `lamp`

O'Reilly's Graph Databases ebook, free.

Why Neo4J?

- Nice language ;)
- HTTP/REST API with JSON
- BOLT binary interface for speed

HTTP/REST API

POST request:
http://localhost:7474/db/data/transaction/commit

```
{
  "statements" : [ {
    "statement" : "CREATE (n) RETURN id(n)"
  } ]
}
```

Response:

```
{
  "results" : [ {
    "columns" : [ "id(n)" ],
    "data" : [ {
      "row" : [ 6 ],
      "meta" : [ null ]
    } ]
  } ],
  "errors" : [ ]
}
```

Cypher, binding

```
(sensor:D {class:'traffic', name:'CAT_C1_2'})
   -[:IN]->
(segment:S)
   -[:HAS {lclass:'me2'}]->
(config:C)
   -[:REGARDS {power:70}]->
(lamp:L {name:'KRK_01223198_41'})
```

Queries

```
MATCH
  (sensor:D {class:'traffic', name:'CAT_C1_2'})
   -[:IN]->
  (segment:S)
RETURN sensor,segment
```

Queries, aggregates

```
MATCH
  (sensor:D {class:'traffic'})
RETURN count(sensor)

MATCH
  (sensor:D {class:'traffic'}) -[:IN]-> (segment:S)
RETURN segment.name, count(sensor)
```

WHERE clause

```
MATCH
  (sensor:D {class:'traffic'})
   -[:IN]->
  (segment:S)
WHERE  sensor.confidence>0.9
RETURN sensor,segment
```

Operators: math, comparison, boolean, string, list, property.
See: `http://neo4j.com/docs/developer-manual/current/cypher/syntax/operators/`

Each node has a unique id

```
MATCH (n) WHERE id(n)=425 RETURN n;
```

A Path

```
MATCH (l:L),(s:S),
p = ((l)-[*..10]-(s))
RETURN nodes(p)
```

```
MATCH (l:L),(s:S),
p = ((l)-[*..10]-(s))
RETURN relationships(p)
```

```
MATCH (l:L),(s:S),
p = shortestPath((l)-[*..10]-(s))
RETURN p
```

http://neo4j.com/docs/developer-manual/current/cypher/syntax/patterns/

New data

```
CREATE (d:D {name:'TR1', class:'traffic'})
          -[:IN]->
       (s:S {name:'1'})
```

Adding a vertex

```
MATCH (s:S {name:"1"})
MERGE
(s)
   - [:HAS {lclass:'me2'}] ->
(c:C)
```

Removing

```
MATCH (d)-->(s {name:'2'}) DELETE d,s
```

You are not allowed to remove a vertex if there is an edge!

```
MATCH (d)-[e]->(s {name:'2'}) DELETE e,d,s
```

Alternatively:

```
MATCH (s) DETACH DELETE s;
```

Adding a vertex/edge

```
MATCH (s:S {name:"1"})
MERGE
  (s) - [:HAS {lclass:'me2'}] -> (c:C)
```

A vertex without an edge?

```
MATCH (n:TMP)
WHERE NOT EXISTS ((n)--())
RETURN n
```

Updating labels and properties

```
MATCH (sensor:D {class:'traffic', name:'CAT_C1_2'})
SET sensor.traffic=12332
RETURN sensor

MATCH (sensor:D {class:'traffic', name:'CAT_C1_2'})
SET sensor:UNDERGROUND
RETURN sensor

MATCH
(sensor:D {class:'traffic', name:'CAT_C1_2'})
REMOVE sensor:UNDERGROUND
RETURN sensor

MATCH
(sensor:D {class:'traffic', name:'CAT_C1_2'})
REMOVE sensor.traffic
RETURN sensor
```

Index, Constraint

Warp speed...

```
CREATE INDEX lamp_name_idx
  FOR (n:L) ON (n.name)

DROP INDEX lamp_name_idx

CREATE CONSTRAINT lamp_name_unique
  FOR (n:L) REQUIRE n.name IS UNIQUE

DROP CONSTRAINT lamp_name_unique
```

Cypher, Import CSV

```
LOAD CSV WITH HEADERS FROM
"file:///e1gs.csv"
AS line
MERGE (x {label:line.flabel,idx:line.fidx})
MERGE (y {label:line.tlabel,idx:line.tidx})
MERGE (x) -[:E]-> (y)
```

- a *label* must be given, it cannot be a value from CSV
- similarly for a *property name*