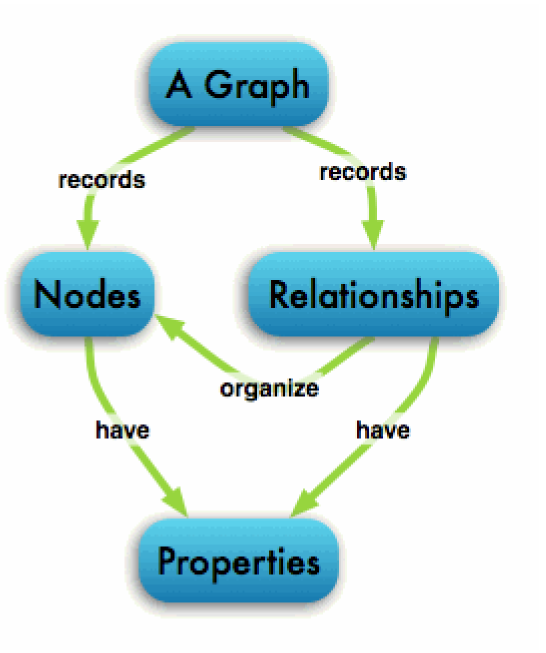
Modern Database Concepts

Practicals: Graph databases

Doc. RNDr. Irena Holubova, Ph.D.
holubova@ksi.mff.cuni.cz

Neo4j

- Open source graph database
  - The most popular
- Initial release: 2007
- Written in: Java
- API: Java, Gremlin, Cypher, …
- OS: cross-platform
- Stores data in nodes connected by directed, typed relationships
  - With properties on both
  - Called property graph



http://www.neo4j.org/

Cypher

- Neo4j graph query language
  - For querying and updating
- Declarative – we describe what we want, not how to get it
  - Not necessary to express traversals
- Human-readable
  - Inspired by SQL and SPARQL

http://docs.neo4j.org/chunked/stable/cypher-query-lang.html

Cypher Clauses

- START: Starting points in the graph, obtained via index lookups or by element IDs.
- MATCH: The graph pattern to match, bound to the starting points in START.
- WHERE: Filtering criteria.
- RETURN: What to return.
- CREATE: Creates nodes and relationships.
- DELETE: Removes nodes, relationships and properties.
- SET: Set values to properties.
- FOREACH: Performs updating actions once per element in a list.
- WITH: Divides a query into multiple, distinct parts.

Cypher Examples

Creating Nodes

```
CREATE (n);
0 rows available after 8 ms, consumed after another 0 ms
Added 1 nodes

CREATE (a {name : 'Andres'}) RETURN a;
+---------------------+
| a                   |
+---------------------+
| ({name: "Andres"}) |
+---------------------+

1 row available after 13 ms, consumed after another 0 ms
Added 1 nodes, Set 1 properties


CREATE (n  {name : 'Andres', title : 'Developer'});
0 rows available after 13 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties
```

Cypher Examples

Creating Relationships

```
MATCH  (a {name:"Andres"})
CREATE (a)-[r:FRIEND]->(b {name:"Jana"} )
RETURN r;
+------------+
| r          |
+------------+
| [:FRIEND]  |
+------------+

1 row available after 27 ms, consumed after another 1 ms
Added 1 node, Created 1 relationship, Set 1 property

MATCH (a {name:"Andres"})
MATCH (b {name:"Jana"})
CREATE (a)-[r:RELTYPE {name : a.name + '<->' + b.name }]->(b)
RETURN r;
1 row available after 18 ms, consumed after another 1 ms
Created 1 relationship, Set 1 property
```

Cypher Examples

Creating Paths

```
CREATE p = (andres {name:'Andres'})-[:WORKS_AT]->(neo)<-[:WORKS_AT]-
    (michael {name:'Michael'})
RETURN p;
+-----------------------------------------------------------------------+
| p                                                                     |
+-----------------------------------------------------------------------+
| ({name: "Andres"})-[:WORKS_AT]->()<-[:WORKS_AT]-({name: "Michael"}) |
+-----------------------------------------------------------------------+

1 row available after 188 ms, consumed after another 22 ms
Added 3 nodes, Created 2 relationships, Set 2 properties
```

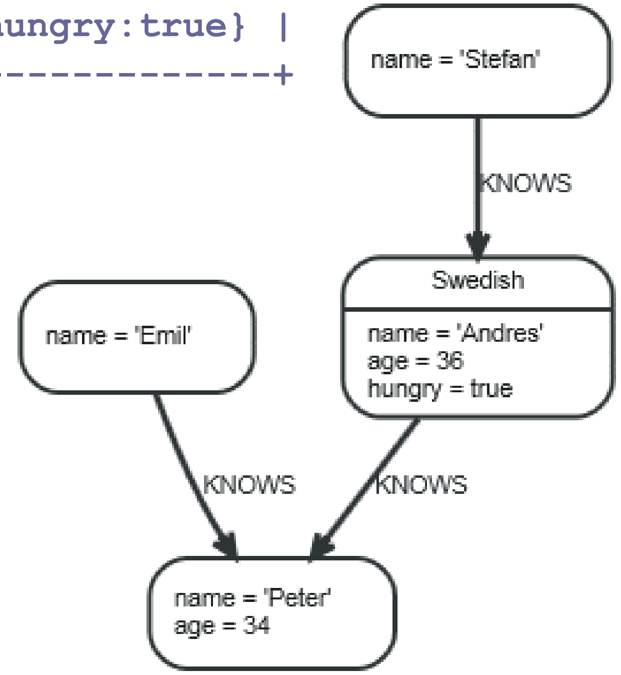
all parts of the pattern not already in scope are created

Cypher Examples

Changing Properties

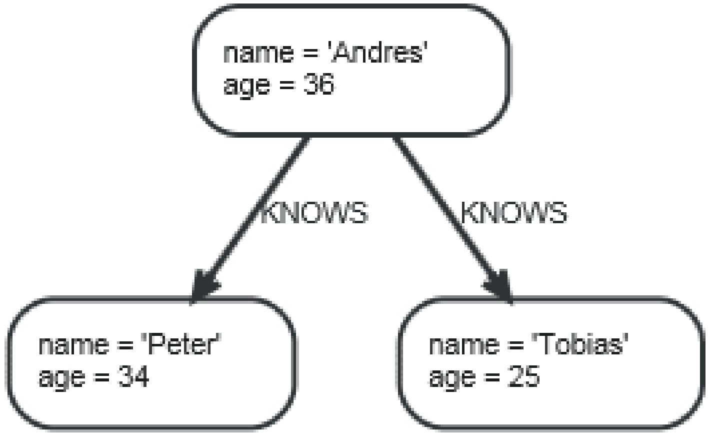
```
MATCH (n { name: 'Andres' })
SET n.surname = 'Taylor'
RETURN n
| n                                                              |
+----------------------------------------------------------------+
| Node[0]{surname:"Taylor",name:"Andres",age:36,hungry:true}     |
+----------------------------------------------------------------+
1 row
Properties set: 1
```

```
MATCH (n { name: 'Andres' })
SET n.name = NULL RETURN n
+-------------------------------+
| n                             |
+-------------------------------+
| Node[0]{hungry:true,age:36}   |
+-------------------------------+
1 row
Properties set: 1
```



Cypher Examples

Delete



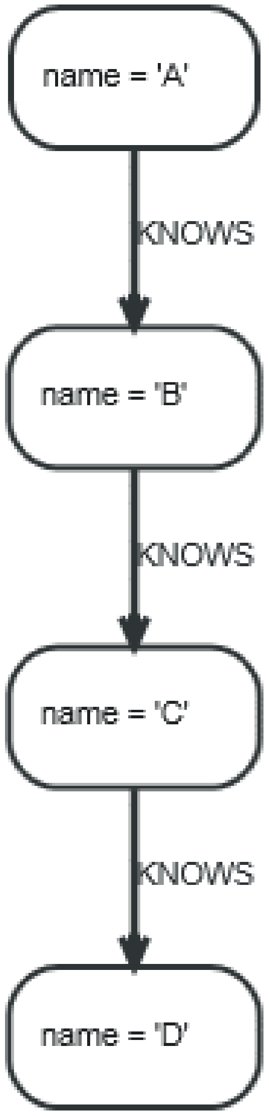
```
MATCH (n { name: 'Andres' })
DETACH DELETE n
+--------------------+
| No data returned. |
+--------------------+
Nodes deleted: 1
Relationships deleted: 2
```

```
MATCH (n { name: 'Andres' })-[r:KNOWS]->()
DELETE r
+--------------------+
| No data returned. |
+--------------------+
Relationships deleted: 2
```

Cypher Examples

Foreach

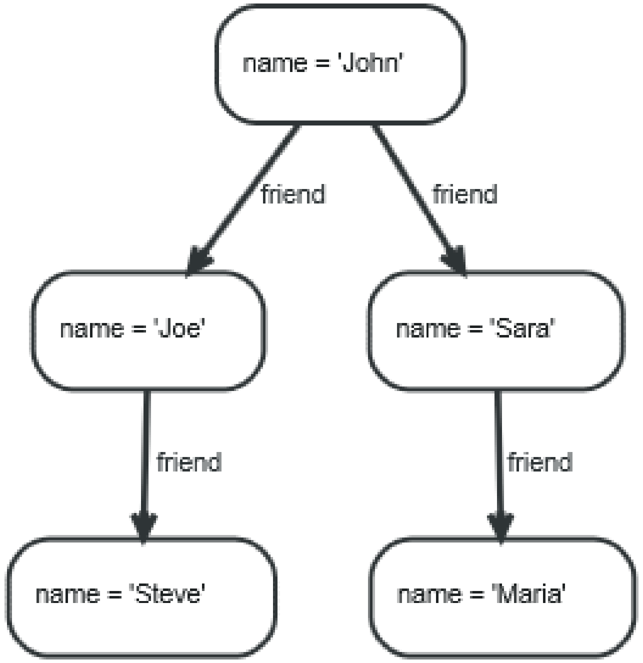
```
MATCH p =(begin)-[*]->(END)
WHERE begin.name = 'A' AND END.name = 'D'
FOREACH (n IN nodes(p)| SET n.marked = TRUE )
+--------------------+
| No data returned.  |
+--------------------+
Properties set: 4
```



Cypher Examples

Querying

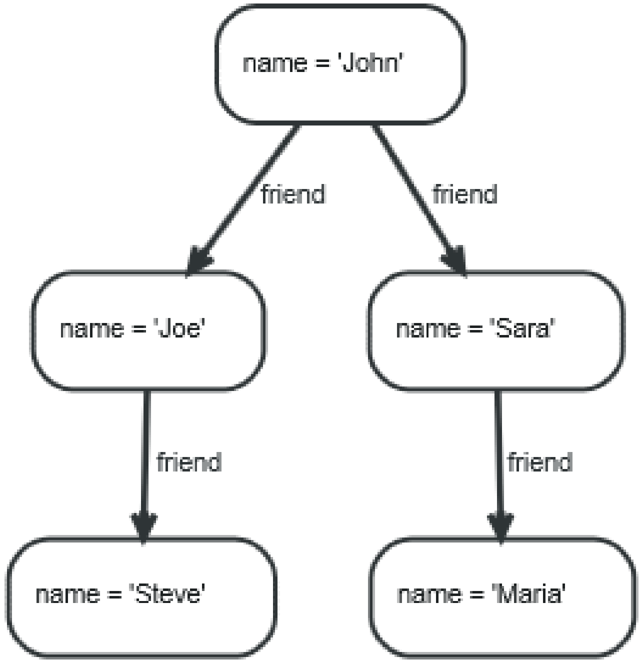
```
MATCH (john {name: 'John'})-[:friend]->()-[:friend]->(fof)
RETURN john.name, fof.name
+----------------------+
| john.name | fof.name |
+----------------------+
| "John"    | "Maria"  |
| "John"    | "Steve"  |
+----------------------+
2 rows
```



Cypher Examples

Querying

```
MATCH (user)-[:friend]->(follower)
WHERE user.name IN ['Joe', 'John', 'Sara', 'Maria', 'Steve'] AND
   follower.name =~ 'S.*'
RETURN user.name, follower.name
+----------------------------+
| user.name | follower.name |
+----------------------------+
| "Joe"     | "Steve"       |
| "John"    | "Sara"        |
+----------------------------+
2 rows
```

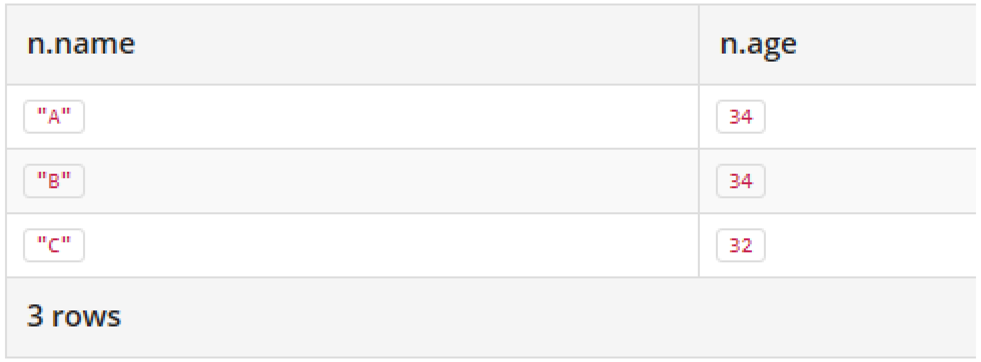


Cypher Examples

Order by

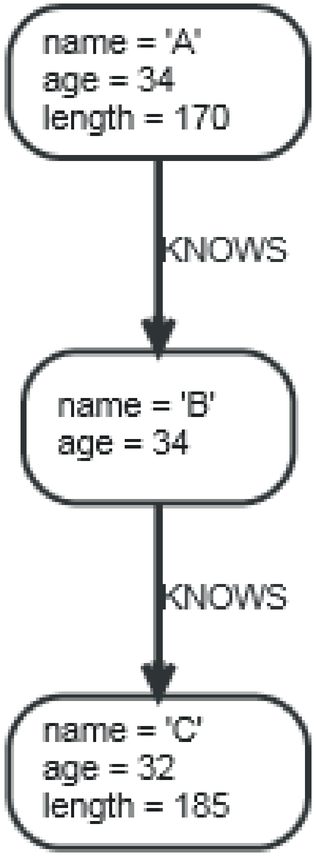
```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.name
```

We can use:
- multiple properties
- asc/desc



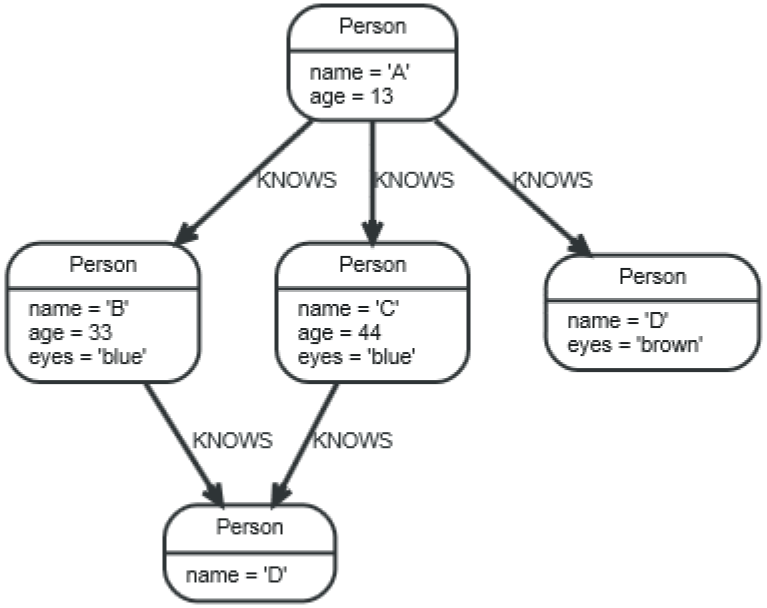
n.name	n.age
"A"	34
"B"	34
"C"	32

3 rows

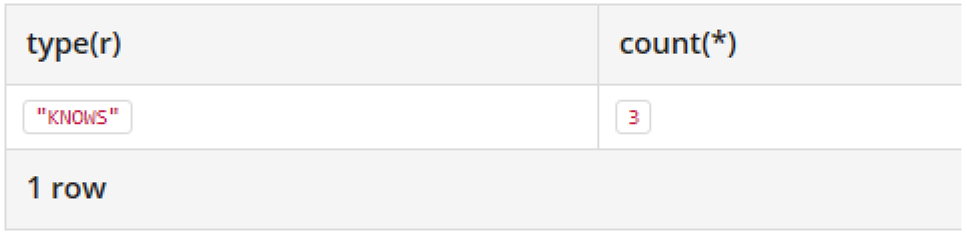


Cypher Examples

Count



```
MATCH (n { name: 'A' })-[r]->()
RETURN type(r), count(*)
```



type(r)	count(*)
"KNOWS"	3

1 row

Cypher

- And there are many other features
  - Other aggregation functions
    - Count, sum, avg, max, min
  - LIMIT n - returns only subsets of the total result
    - SKIP n = trimmed from the top
    - Often combined with order by
  - Predicates ALL and ANY
  - Functions
    - LENGTH of a path, TYPE of a relationship, ID of node/relationship, NODES of a path, RELATIONSHIPS of a path, …
  - Operators
  - …

Assignment

- Chose your unique problem domain
  - E.g., the results of football matches of various teams
- For your selected problem domain, think about an application that uses Neo4j graph (create a graph with different types of nodes and edges, both with various properties, create several non-trivial queries)
  - You can use Cypher or any other API (Java, Gremlin, …)
- Submit a script with respective commands for Neo4j + explanatory <u>comments</u>

References

- Neo4j http://www.neo4j.org/
- Neo4j Manual http://docs.neo4j.org/chunked/stable/
- Neo4j Cypher most up-to-date commands:

  https://neo4j.com/developer/cypher-query-language/
- Neo4j Cypher Refcard:

  https://neo4j.com/docs/cypher-refcard/current/
  - Neo4j testing console
    - http://console.neo4j.org/
- Gremlin –getting started:
  - https://github.com/tinkerpop/gremlin/wiki/Getting-Started