NEo4J

Documentation

Abhijith M

Contents

[Basics 3](#_Toc28181174)

[Create node 3](#_Toc28181175)

[SET – add property 3](#_Toc28181176)

[Create Node with property and values 3](#_Toc28181177)

[Directed relation – add relationship bwn two nodes 3](#_Toc28181178)

[Foreach – create multiple nodes 3](#_Toc28181179)

[Create node with label only 3](#_Toc28181180)

[Create node with label and properties 3](#_Toc28181181)

[Create multiple nodes with label and property 3](#_Toc28181182)

[Add relationship to two nodes 4](#_Toc28181183)

[Load csv 4](#_Toc28181184)

[Match 4](#_Toc28181185)

[OR – match two ids 4](#_Toc28181186)

[WHERE 4](#_Toc28181187)

[Order by – name ascending order 5](#_Toc28181188)

[Limit result 5](#_Toc28181189)

[Skip – skip first result 5](#_Toc28181190)

[Search with label 5](#_Toc28181191)

[Starts with – name starts with R 5](#_Toc28181192)

[Ends With – name ends with KY 5](#_Toc28181193)

[Contains – name contain ‘O’ 5](#_Toc28181194)

[Match with relationship 5](#_Toc28181195)

[Pattern – find friend of friend 6](#_Toc28181196)

[CASE – show msg when age > 20 6](#_Toc28181197)

[Mathematical operations 7](#_Toc28181198)

[Addition, Multiplication … 7](#_Toc28181199)

[Comparison operations:- =,<>,<,> 7](#_Toc28181200)

[Delete 7](#_Toc28181201)

[Delete with relationship 7](#_Toc28181202)

[Graph stylesheet modification 8](#_Toc28181203)

[Neo4j with python 8](#_Toc28181204)

[install neo4j 8](#_Toc28181205)

[Check connection manually 8](#_Toc28181206)

[Establish connection 8](#_Toc28181207)

[Run query 8](#_Toc28181208)

[Dynamic node property 8](#_Toc28181209)

[Regex – return name start with J 8](#_Toc28181210)

# Basics

## Create node

CREATE (n)

### SET – add property

MATCH (N) SET N:player

MATCH (n) return n

Output > one node have id 20,second one have id 0

### Create Node with property and values

CREATE (node1:player{name:"VIRAT"}),(node2:player{name:"ROHIT"}) # Create node with property and values

MATCH (n) return n # Return all nodes

Output > shows two node VIRAT & ROHIT

MATCH (a), (b) where a.name='VIRAT' and b.name='ROHIT' return a,b # select based on condition

### Directed relation – add relationship bwn two nodes

MATCH (a), (b) where a.name='VIRAT' and b.name='ROHIT' CREATE (a)-[:knows]->(b) # create directed relationship between a->b

MATCH (n) return n

### Foreach – create multiple nodes

foreach (i in RANGE(0,49)|CREATE(n)) # create 50 nodes

### Create node with label only

create (n:player) # create a node n with label player

### Create node with label and properties

create (n:player{name:"LEO MESSI", team:"Bacelona"})

# create node with label player and add two property name and team

### Create multiple nodes with label and property

create(n:player{name:"LEO"}), (p:player{name:"CR7"})

# create multiple node with label player and property name

### Add relationship to two nodes

match (n:player), (p:player) where n.name="CR7" and p.name="LEO" create (n)-[r:is\_friend\_of]->(p)

# Add relationship to two nodes

### Load csv

LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS fl

CREATE(N:BULKCSV{NAME:fl.NAME,COMPANY:fl.COMPANY})

# data.csv

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NAME,COMPANY

ROHIT,ABC

RAHUL,XYZ

ROHIT,PQR

VIRAT,BBCI

Homu,homu&co

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* The data.csv file should be in the following directory
* [file:///C:/Users/abhijith.m/.Neo4jDesktop/neo4jDatabases/database-2627cb3e-5cba-43e7-b089-7618fc32847c/installation-3.5.12/import/](file:///C:\Users\abhijith.m\.Neo4jDesktop\neo4jDatabases\database-2627cb3e-5cba-43e7-b089-7618fc32847c\installation-3.5.12\import\)

## Match

MATCH (anything) return anything

CREATE (N)

MATCH (n) return n

Output > two circles will generate

### OR – match two ids

MATCH (N) where id(N)=5 or id(N)=6 return (N) # Search node 5 or 6

### WHERE

MATCH (n) where id(n)=1 return n # return node having id==1

MATCH (n) where id(n)=1 SET n:player return n # set property player to node 1

MATCH (N) DETACH DELETE (N) # delete all nodes

### Order by – name ascending order

MATCH(N)

RETURN N.NAME ORDER BY N.NAME LIMIT 5

### Limit result

MATCH(N) return (N) LIMIT 1 # only shows one result

### Skip – skip first result

MATCH(N)

RETURN N.NAME ORDER BY N.NAME SKIP 1

### Search with label

CREATE (N:TEAM) # create node with label team

MATCH(N:TEAM) return (N) # search with labels

### Starts with – name starts with R

MATCH(N)

WHERE N.NAME STARTS WITH 'R'

RETURN N.NAME

### Ends With – name ends with KY

MATCH(N)

WHERE N.NAME ENDS WITH 'KY'

RETURN N.NAME

### Contains – name contain ‘O’

MATCH(N)

WHERE N.NAME CONTAINS 'O'

RETURN N.NAME

### Match with relationship

CREATE(RONI:PLAYER{NAME:'RONI'}),

(VICKY:PLAYER{NAME:'VICKY'}),

(PAUL:PLAYER{NAME:'PAUL'}),

(JOHN:PLAYER{NAME:'JOHN'}),

(RONI)-[:FRIENDS]->(VICKY),

(RONI)-[:FRIENDS]->(PAUL),

(PAUL)-[:FRIENDS]->(VICKY)

# match with relationship

MATCH (N{NAME:'RONI'})-[:FRIENDS]->(friend)

WITH N, friend,count(friend) as fcount

RETURN (friend.NAME), fcount

### Pattern – find friend of friend

CREATE (N:FRIEND{NAME:'RONI'}),

(M:FRIEND{NAME:'ROCKY'}),

(P:FRIEND{NAME:'JOHN'}),

(N)-[:FRIENDS]->(M),

(M)-[:FRIENDS]->(P)

# Match friend of friend relationship

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND{NAME:'RONI'})-[:FRIENDS]-(P) # find roni's friend - rocky

MATCH (P)-[:FRIENDS]->(O) # find rocky's friends - roni & john

RETURN O.NAME AS FRIEND\_OF\_FRIEND

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND{NAME:'RONI'})-[:FRIENDS]-(P), # find roni's friend -> rocky

(P)-[:FRIENDS]-(O) # find rocky's friend -> john

RETURN O.NAME AS FRIEND\_OF\_FRIEND

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### CASE – show msg when age > 20

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND{NAME:'RONI'})

SET N.AGE = 20

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND{NAME:'ROCKY'})

SET N.AGE = 25

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND{NAME:'JOHN'})

SET N.AGE = 28

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# CASE Example

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH (N:FRIEND)

RETURN N.NAME,

CASE

WHEN N.AGE > 25

THEN 'OK'

WHEN N.AGE > 20

THEN 'Not OK'

ELSE 'Not anything' END AS RESULT

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## PATTERN

### Find all paths

MATCH (A)-[]->(B)

RETURN A,B

### Pattern without realation

MATCH (a)<--(b)-->(c)

RETURN a,b,c

### Inbetween 2 nodes:- A \* \* B

MATCH (A)-[\*2]->(B )RETURN A,B # A & B are connected inbetween their are two nodes are available

MATCH (A)-[\*1..3]->(B )RETURN A,B # A & B are connected inbetween their are 1..3 nodes are available

MATCH (A{NAME:"PAUL"})-[\*1..3]->(B{NAME:'VICKY'})RETURN A,B # Check their is 2 relationships between paul and Vicky

### In degree

# Indegree – number of nodes comes from other nodes

MATCH (N:PLAYER)

with [(N)-->(P) WHERE P:PLAYER|P.NAME] AS NAMES

unwind NAMES as names

return names, count(names)

### Mutual friend

# Find mutual friend

MATCH (N:PLAYER{NAME:"RONI"}),

(M:PLAYER{NAME:"PAUL"})

with [(N)--(B)--(M) WHERE B:PLAYER|B.NAME] AS NAMES

unwind NAMES as NAME\_OF\_MUTUAL\_FRIEND

RETURN NAME\_OF\_MUTUAL\_FRIEND

### Available path between two nodes

# Find the number of available path between two nodes

MATCH(N:PLAYER{NAME:"RONI"}),

(M:PLAYER{NAME:"VICKY"})

RETURN [(N)-[:FRIENDS\*1..]->(M)|TRUE]

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MATCH(N:PLAYER{NAME:"RONI"}),

(M:PLAYER{NAME:"VICKY"})

WITH [(N)-[:FRIENDS\*1..]->(M)|TRUE] AS NUMBERS

unwind NUMBERS AS NO\_OF\_PATHS

RETURN count(NO\_OF\_PATHS) AS NUMBER\_OF\_AVAILABLE\_PATHS

## Constraints

### Unique

# constaint - unique property

create constraint on (a:COMPANY)

ASSERT a.id is unique

# testing

create (a:COMPANY{id:1,name:"google"})

create (a:COMPANY{id:1,name:"yahoo"})

* Output >> Neo.ClientError.Schema.ConstraintValidationFailed

Node(12) already exists with label `COMPANY` and property `id` = 1

### Delete constraint

# delete unique constriant

drop constraint on (a:COMPANY)

Assert a.id is unique

## RETURN

### Return All nodes

match(n)return (n)

### relationship

# Relaionship name

match(n:PLAYER)-[R]->(M:PLAYER)

return type(R)

match(n)-[R]->(M)

return type(R)

### property

# return property

match(N:PLAYER) return N.name

### check in list

WITH ["RONI", "MESSI"] AS a

unwind a as name

with name

return name in ["RONI", "RAHUL"]

### With Where

MATCH (N)

where N:PLAYER

return N.NAME

### Count

MATCH (N)

return count(N)

## Mathematical operations

### Addition, Multiplication …

with 2 as a, 3 as b

return a + b as result

### Comparison operations:- =,<>,<,>

with 2 as a, 3 as b

return a = b as result

### Boolean operations:- AND, OR, XOR

with 2 as a, 3 as b

return a > b OR a=b as result

## String operations

### Concatination

with "RANI" as a, "DAS" as b

return a +","+b as result

## List Operations

### Concatination

with [2,3] as a, [4,5] as b

return a +b as result

output >> <Record result=[2, 3, 4, 5]>

### IN operator

with [2,3] as a

unwind a as number

with number

where number in [1,3]

return number

### List Comprehension

RETURN [x in RANGE(0,10) WHERE x%2=0|x^2] AS result

* Output >> result=[0.0, 4.0, 16.0, 36.0, 64.0, 100.0]

## Delete

### Delete with relationship

MATCH (n) DETACH DELETE (n) # delete all nodes with relationship

CREATE (N), (n) # creating two nodes

MATCH (N) return N

## Graph stylesheet modification

:style

# modify the style of graph

# Neo4j with python

## install neo4j

!pip install neo4j

## Check connection manually

* Open neo4j desktop app – start graph
* Manage -> copy https port
* Brows it in browser

## Establish connection

from neo4j import GraphDatabase

graphdb = GraphDatabase.driver(uri="bolt://localhost:7687", auth=("neo4j", "initial#024"))

# auth - username, password

seesion = graphdb.session()

# create session

## Run query

q1 = "MATCH(N) return (N)"

nodes = seesion.run(q1)

# Display result

for node in nodes:

print (node)

## Dynamic node property

q1 = "CREATE (N:PERSON{NAME:$NAME, CITY:$CITY})"

x = {"NAME" : "RONITEST", "CITY":"MUMBAI"}

nodes = seesion.run(q1, x)

## Regex – return name start with J

x = {"regex": "J.\*"}

q1 = '''

MATCH(N)

WHERE N.NAME =~ $regex

RETURN N.CITY

'''

nodes = seesion.run(q1,x)

for node in nodes:

print (node)