Course code : **CSE3009**
Course title : **No SQL Data Bases**
Module : **6**
Topic : **4**

Neo4j CQL

Objectives

This session will give the knowledge about

- Neo4j Cypher Query Language (CQL) – Examples
- Neo4j CQL Write Clauses
- Neo4j CQL Read Clause
- All the examples commands are executed to implement Student alumni database

Neo4j – Create Command

- Create a single node
- Create multiple nodes
- Create a node with a label
- Create a node with multiple labels
- Create a node with properties
- Returning the created node

Create Command

Create Single node:

Syntax: CREATE (node_name);

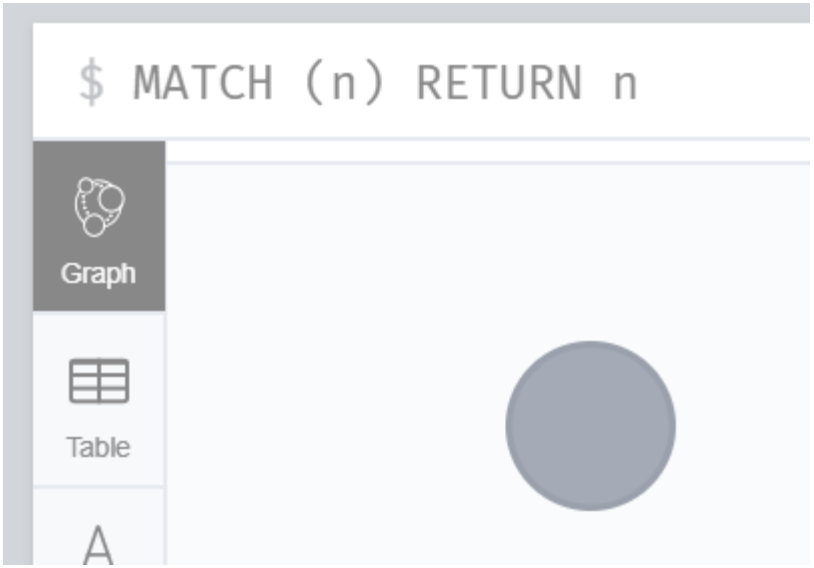
Example: CREATE (John);

Output:

```
CYPHER CREATE (John);
```

✔ Created 1 node, returned 0 rows in 537 ms

Verification: MATCH (n) RETURN n



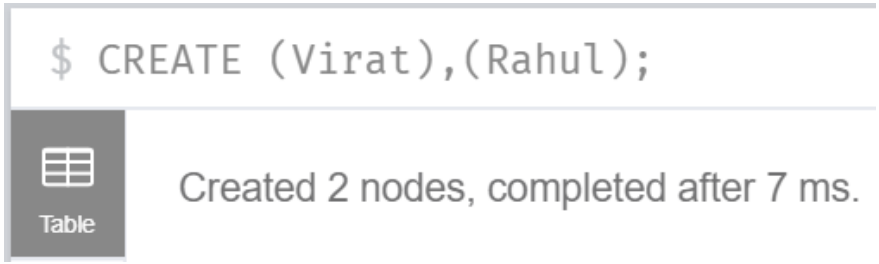
Create Command

Create Multiple node:

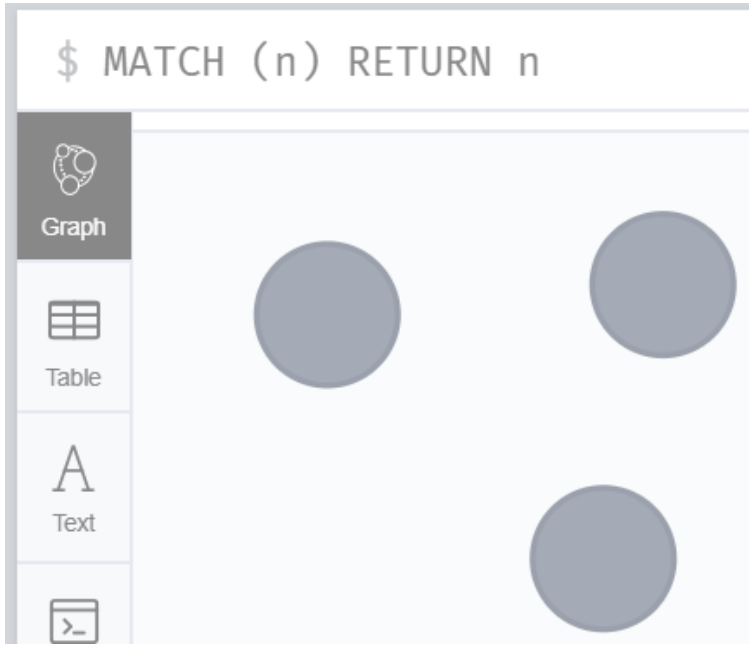
Syntax: CREATE (node_name1), (node_name2);

Example: CREATE (Virat),(Rahul);

Output:



Verification: MATCH (n) RETURN n



Delete Command

Deleting All Nodes and Relationships:

Syntax: MATCH (n) DETACH DELETE n;

Example: MATCH (n) DETACH DELETE n;

Deleting a Particular Node:

Syntax: MATCH (node:label {properties . . . . . }) DETACH DELETE node

Example:

CREATE (John:student {name: “John Smith”, YOP: 2015, DEPT: “CSE”})

MATCH (John:student {name: “John Smith”, YOP: 2015, DEPT: “CSE”}) DETACH DELETE John

Create Command

Creating a Node with a Label

Syntax: CREATE (node:label)

Example: CREATE (John:student), (Virat:student);

Creating a Node with Multiple Labels

Syntax: CREATE (node:label1:label2:. . . . labeln)

Example: CREATE (John:student:cse)

Create Command

Create Node with Properties

Syntax: CREATE (node:label { key1: value, key2: value, ... })

Example:

CREATE (John:student:cse {name: "John Smith", YOP: 2015, roll: 101}), (Virat:student:ece {name: “Virat Kohli", YOP: 2015, roll: 101})

Returning the Created Node: instead MATCH (n) RETURN n

Syntax: CREATE (Node:Label{properties. . . . }) RETURN Node

Example:

CREATE (John:student:cse {name: "John Smith", YOP: 2015, roll: 101}) RETURN John

Creating Relationships

Creating Relationships

Syntax: CREATE (node1)-[:RelationshipType]->(node2)

Example:

CREATE (John:student:cse {name: "John Smith", YOP: 2015, roll: 101}), (Virat:student:ece {name: "Virat Kohli", YOP: 2015, roll: 101})

CREATE (VIT:College {name: “VIT-Vellore"})

CREATE (John)-[r:student_of]->(VIT)

RETURN John, Virat, VIT

Creating Relationships

Creating a Relationship Between the Existing Nodes

Syntax:

MATCH (a:LabeofNode1), (b:LabeofNode2) WHERE a.name = "nameofnode1" AND b.name = " nameofnode2"

CREATE (a)-[: Relation]->(b)

RETURN a,b

Example:

MATCH (a:student), (b:College) WHERE a.name = "Virat Kohli" AND b.name = "VIT-Vellore"

CREATE (a)-[r: student_of]->(b)

RETURN a,b

Creating Relationships

Creating a Relationship with Label and Properties

Syntax

CREATE (node1)-[label:Rel_Type {key1:value1, key2:value2, . . . n}]-> (node2)

Example

MATCH (a:student), (b:College) WHERE a.name = "Virat Kohli" AND b.name = "VIT-Vellore"

CREATE (a)-[r: student_of {placed_in: "infy",salary:40000}]->(b)

RETURN a,b

Creating a Complete Path

Creating a Complete Path

Syntax

```
CREATE p = (Node1 {properties})-[:Relationship_Type]->
   (Node2 {properties})-[:Relationship_Type]->(Node3 {properties})
RETURN p
```

Example

```
CREATE path= (Virat{name:"Virat Kohli"}) –[:student_of]->(VIT{name:"VIT-Vellore"})-
[:friend_of]->(John{name:"John Smith"})
Return Virat,VIT,John;
```

Merge Command

- MERGE command is a combination of CREATE command and MATCH command.
- Neo4j CQL MERGE command searches for a given pattern in the graph. If it exists, then it returns the results.
- If it does NOT exist in the graph, then it creates a new node/relationship and returns the results.

In this session, we are going to discuss:

- Merge a node with label
- Merge a node with properties
- OnCreate and OnMatch
- Merge a relationship

Merge Command

Syntax

MERGE (node: label {properties . . . . . })

Before proceeding to the examples in this section

CREATE (John:student:cse {name: "John Smith", YOP: 2015, roll: 101})

CREATE (VIT:College {name: "VIT-Vellore"})

CREATE (John)-[r:student_of]->(VIT)

RETURN John, VIT

Merging a Node with a Label

Merging a Node with a Label

Syntax: MERGE (node:label) RETURN node

Example:

MERGE (John:student:cse {name: "John Smith", YOP: 2015, roll: 101}) RETURN John

MERGE (Venkat:student:cse{name: "Venkat Reddy", YOP: 2016, roll: 101}) RETURN Venkat;

OnCreate and OnMatch

OnCreate and OnMatch

Syntax:

```
MERGE (node:label {properties . . . . . .})
ON CREATE SET property.isCreated ="true"
ON MATCH SET property.isFound ="true"
```

Example

```
MERGE (Venkat:student:cse{name: "Venkat Reddy", YOP: 2016, roll: 101})
ON CREATE SET Venkat.isCreated = "true"
ON MATCH SET Venkat.isFound = "true"
RETURN Venkat
```

Merge a Relationship

Just like nodes, you can also merge the relationships using the MERGE clause.

Example

MATCH (a:student), (b:College) WHERE a.name = "John Smith" AND b.name = "VIT-Vellore" MERGE (a)-[r: student_of]->(b) RETURN a,b

MATCH (a:student), (b:student) WHERE a.name = "Venkat Reddy" AND b.name = "John Smith" MERGE (a)-[r: friends_of]->(b) RETURN a,b

Set Clause

Using Set clause, you can add new properties to an existing Node or Relationship, and also add or update existing Properties values.

In this session:

- Set a property
- Remove a property
- Set multiple properties
- Set a label on a node
- Set multiple labels on a node

Setting a Property

Syntax

MATCH (node:label{properties . . . . . }) SET node.property = value

RETURN node

Example

CREATE (Jenny:student:cse {name: "Jenny Rose", YOP: 2015, roll: 102})

MATCH (Jenny) SET Jenny.job = "infy" RETURN Jenny

Removing a Property

Syntax

MATCH (node:label {properties})

SET node.property = NULL

RETURN node

Example

```
MERGE (Jenny:student:cse {name: "Jenny Rose", YOP: 2015, roll: 102})
MATCH (Jenny) SET Jenny.YOP = NULL RETURN Jenny
```

Setting Multiple Property

Syntax

MATCH (node:label {properties})

SET node.property1 = value, node.property2 = value

RETURN node

Example

CREATE (Jenny:student:cse {name: "Jenny Rose", YOP: 2015, roll: 102})

MATCH (Jenny) SET Jenny.job = "infy", Jenny.salary=50000 RETURN Jenny

Setting a Label on a Node

Syntax

MATCH (n {properties . . . . . })

SET n :label SET n :label1:label2

RETURN n

Example

CREATE (Mark {name: "Mark Red", YOP: 2016, roll: 101})

CREATE (Jack {name: "Jack Sparrow", YOP: 2016, roll: 101})

MATCH (Mark {name: "Mark Red", YOP: 2016, roll: 101}) SET Mark:student:cse RETURN Mark

MATCH (Jack) SET Jack:student RETURN Jack

Remove Clause

The REMOVE clause is used to remove properties and labels from graph elements (Nodes or Relationships). The main difference between Neo4j CQL DELETE and REMOVE commands is:

- DELETE operation is used to delete nodes and associated relationships.
- REMOVE operation is used to remove labels and properties.

Syntax

```
MATCH (node:label{properties . . . . . })
REMOVE node.property
RETURN node
```

Remove Clause

CREATE (Gautham:student:mech {name: "Gautham", YOP: 2015, DEPT: "CSE"})

Example:

MATCH (Gautham:student:mech {name: "Gautham"}) REMOVE Gautham.YOP
RETURN Gautham

MATCH (Gautham:student:mech {name: "Gautham"}) REMOVE Gautham:mech
RETURN Gautham

MATCH (Gautham:student {name: "Gautham"}) REMOVE Gautham:student
RETURN Gautham

MATCH (Gautham {name: "Gautham"}) SET Gautham:student:ece RETURN Gautham

Foreach Clause

The FOREACH clause is used to update data within a list whether components of a path, or result of aggregation.

Syntax

MATCH p = (start node)-[*]->(end node)

WHERE start.node = "node_name" AND end.node = "node_name"

FOREACH (n IN nodes(p)| SET n.marked = TRUE)

Foreach Clause

CREATE path = (Mark:student:cse {name: "Mark Red", YOP: 2016, roll: 101}) –[: works_with] -> (Jack:student:ece {name: "Jack Sparrow", YOP: 2016, roll: 101}) –[: works_with] -> (Gautham:student:mech {name: "Gautham", YOP: 2015, DEPT: "CSE"})

RETURN path

MATCH path=(Mark) -[*]-> (Gautham) WHERE Mark.name = "Mark Red" AND Gautham.name="Gautham“ FOREACH (n IN nodes(path)| SET n.college = "VIT")

Match Clause

```
CREATE
(John:student:cse {name: "John Smith", YOP: 2015, roll: 101}),
(Sona:student:ece {name: "Sona Vihar", YOP: 2015, roll: 101}),
(Shyam:student:mech {name: "Shyam Prakash", YOP: 2016, roll: 101}),
(Virat:student:ece {name: "Virat Kohli", YOP: 2016, roll: 101}),
(Priya:student:ece {name: “Priya Mohan", YOP: 2015, roll: 103}),
(VIT:college{name: "VIT-AP"}), (SRM:college{name: "SRM"})

MATCH (a:student), (b:college) WHERE a.name = "John Smith" AND b.name = "VIT-AP"
MERGE (a)-[r:placed_in{company: "infy", salary:40000}]->(b)
```

Match Clause

```
MATCH (a:student), (b:college) WHERE a.name = "Sona Vihar" AND b.name = "VIT-AP"
MERGE (a)-[r:placed_in{company: "tcs", salary:50000}]->(b)

MATCH (a:student), (b:college) WHERE a.name = "Priya Mohan" AND b.name = "VIT-AP"
MERGE (a)-[r:placed_in{company: "infy", salary:55000}]->(b)

MATCH (a:student), (b:college) WHERE a.name = "Shyam Prakash" AND b.name = "SRM"
MERGE (a)-[r:placed_in{company: "infy", salary:50000}]->(b)

MATCH (a:student), (b:college) WHERE a.name = "Virat Kohli" AND b.name = "SRM"
MERGE (a)-[r:placed_in{company: "wipro", salary:45000}]->(b)
```

Match Clause

Getting All Nodes Under a Specific Label

Syntax

```
MATCH (node:label)
RETURN node
```

Example

```
MATCH (n:college)
RETURN n
```

Getting all nodes that matches the condition

```
MATCH (VIT:college{name: "VIT-AP"})<-[r:placed_in{company: "infy"}]-(n)
RETURN n.name
```

Optional Match Clause

The OPTIONAL MATCH clause is used to search for the pattern described in it, while using nulls for missing parts of the pattern.

OPTIONAL MATCH is similar to the match clause, the only difference being it returns null as a result of the missing parts of the pattern.

Syntax

```
MATCH (node:label {properties. . . . . .}) OPTIONAL MATCH (node)-->(x)
RETURN x
```

Example

```
MATCH (VIT:college{name: "VIT-AP"}) OPTIONAL MATCH (VIT)-->(x)
RETURN x.name
```

Where Clause

Like SQL, Neo4j CQL has provided WHERE clause in CQL MATCH command to filter the results of a MATCH Query.

Syntax

MATCH (label) WHERE label.property_name = "property_value"

RETURN label

Example

MATCH (label) WHERE label.roll = 101

RETURN label.name

Where Clause with Multiple Conditions

Syntax

```
MATCH (emp:Employee)

WHERE emp.name = 'Abc' AND emp.name = 'Xyz'

RETURN emp
```

Example

```
MATCH (label)

WHERE label.roll = 101 AND label.YOP=2016

RETURN label.name
```

Using Relationship with Where Clause

You can also use Where clause to filter the nodes using the relationships.

Example

```
MATCH (n)

WHERE (n)-[: placed_in{company: "infy"}]->({name: "VIT-AP"})

RETURN n
```

Count Function

The count() function is used to count the number of rows.

Syntax

```
MATCH (n { name: 'A' })-->(x)
RETURN n, count(*)
```

Example

```
Match(n{roll: 101})--(x)
RETURN n, count(*)
```

```
Match(n{roll: 101})-[r]-(x)
RETURN type (r), count(*)
```

Summary

This session will give the knowledge about

- Neo4j Cypher Query Language (CQL) – Examples
- Neo4j CQL Write Clauses
- Neo4j CQL Read Clause
- All the examples commands are executed to implement Student alumni database