Course code : **CSE3009**
Course title : **No SQL Data Bases**
Module : **6**
Topic : **5**

# Neo4j CQL

# Objectives

This session will give the knowledge about

- Neo4j CQL General Clauses
- Neo4j CQL Functions
- Neo4j CQL Admin

# Return Clause

The RETURN clause is used return nodes, relationships, and properties in Neo4j. In this session, we are going to learn how to:

- Return nodes
- Return multiple nodes
- Return relationships
- Return properties
- Return all elements
- Return a variable with column alias

# Return Clause

Return nodes

```
CREATE (Guru:student:cse {name: "Guru Nanal", YOP: 2016, roll: 101})
RETURN Guru
```

Return multiple nodes

```
CREATE (Infy:company {name: "Infosys"})
CREATE (TCS:company {name: "Tata"})
RETURN Infy,TCS
```

# Return Clause

Return relationships

```
MERGE (Guru:student:cse{name:"Guru Nanal"})-[r1:employee_of{exp:4}]->
(Infy:company{name:"Infosys"})
MERGE (Sona:student:ece{name:"Sona Vihar"})-[r2:employee_of{exp:3}]->
(TCS:company{name:"Tata"})
RETURN r1,r2
```

Return properties

```
MATCH (John:student:cse {name: "John Smith"}) RETURN John.roll, John.YOP
```

# Return Clause

Returning a Variable With a Column Alias

```
MATCH (John:student:cse {name: "John Smith"}) RETURN John.YOP AS
Year_of_Passing
```

Returning All Elements

```
Match p = (n {name: "VIT-AP"})-[r]-(x)
RETURN *
```

# Order By Clause

Order By Clause

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

Ordering Nodes by Multiple Properties

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

Ordering Nodes by Descending Order

```
MATCH (n)  RETURN n.name, n.roll
ORDER BY n.name DESC
```

# Limit Clause

Limit Clause

```
MATCH (n)  RETURN n.name, n.roll
ORDER BY n.name LIMIT 3
```

Limit Clause with expression

```
MATCH (n)  RETURN n.name, n.roll
ORDER BY n.name LIMIT toInt(3*rand())+1
```

# Skip Clause

Skip Clause

```
MATCH (n)  RETURN n.name, n.roll
ORDER BY n.name SKIP 3
```

Skip Clause with expression

```
MATCH (n)  RETURN n.name, n.roll
ORDER BY n.name SKIP toInt(2*rand())+1
```

# With Clause

Syntax

```
MATCH (n)
WITH n
ORDER BY n.property
RETURN collect(n.property)
```

Example

```
MATCH (n)
WITH n
ORDER BY n.name DESC LIMIT 3
RETURN collect(n.name)
```

# UNWIND Clause

With UNWIND, you can transform any list back into individual rows.

Unwinding a list

```
UNWIND [1, 2, 3, NULL ] AS x
RETURN x, 'val' AS y
```

Creating a distinct list

```
WITH [1, 1, 2, 2] AS coll
UNWIND coll AS x
WITH DISTINCT x
RETURN collect(x) AS setOfVals
```

# String Functions

String Functions List

UPPER - It is used to change all letters into upper case letters.

```
MATCH (n) RETURN UPPER(n.name)
```

LOWER - It is used to change all letters into lower case letters.

```
MATCH (n) RETURN LOWER(n.name)
```

SUBSTRING - It is used to get substring of a given String.

```
MATCH (n) RETURN SUBSTRING(n.name,0,2)
```

Replace - It is used to replace a substring with a given substring of a String

```
RETURN replace("hello", "l", "w")
```

# Aggregation Function

COUNT - It returns the number of rows returned by MATCH command.

```
MATCH (n:employee) WHERE n.sal>27000 RETURN COUNT(n)
```

MAX - It returns the maximum value from a set of rows returned by MATCH command.

```
MATCH (n:employee) RETURN MAX(n.salary)
```

MIN - It returns the minimum value from a set of rows returned by MATCH command.

```
MATCH (n:employee) RETURN MIN(n.salary)
```

SUM - It returns the summation value of all rows returned by MATCH command.

```
MATCH (n:employee) RETURN SUM(n.salary)
```

AVG - It returns the average value of all rows returned by MATCH command.

```
MATCH (n:employee) RETURN AVG(n.salary)
```

# String Functions

More Functions:

- left()
- lTrim()
- reverse()
- right()
- rTrim()
- split()
- toString()
- trim()

# Predicate functions

Create these nodes:

```
MERGE (Ajith:student:cse {name: "Ajith", roll:105, lang:['telugu','english','tamil']})
MERGE (Vijay:student:cse {name: "Vijay", roll:125, lang:['hindi','english','tamil']})
MERGE (Surya:student:cse {name: "Surya", roll:135, lang:['telugu','hindi']})
RETURN Ajith,Vijay,Surya
```

# Predicate functions

Predicates are boolean functions that return true or false for a given set of non-null input. They are most commonly used to filter out subgraphs in the WHERE part of a query.

Functions:

- all()
- any()
- exists()
- none()
- single()

# Predicate functions

Predicates are boolean functions that return true or false for a given set of non-null input. They are most commonly used to filter out subgraphs in the WHERE part of a query.

Functions:

all() :

```
MATCH (n) WHERE EXISTS (n.roll) AND ALL(x IN n.roll WHERE x>120) RETURN n.name AS name, n.roll AS roll
```

any() :

```
MATCH (n) WHERE EXISTS (n.lang) AND ANY(x IN n.lang WHERE x="english") RETURN n.name AS name, n.lang as lang
```

# Predicate functions

exists() :

MATCH (n) WHERE EXISTS (n.lang) RETURN n.name AS name

none() :

MATCH (n) WHERE EXISTS (n.lang) AND NONE(x IN n.lang WHERE x="telugu")
RETURN n.name AS name, n.lang as lang

single() :

MATCH (n) WHERE EXISTS (n.lang) AND SINGLE(x IN n.lang WHERE x="english")
RETURN n.name AS name, n.lang as lang

# Neo4j - Backup

Step 1 − Click the "Stop" button to shut down the server.

Step 2 − Open the command prompt.

Step 3 − Create a folder "Neo4jDbBackup-01" at C:\Neo4j (This may be any location in your file system).

```
mkdir C:\Neo4j\Neo4jDbBackup-01
```

Step 4 − Type the following command and press Enter key.

```
copy C:\Ne04j2.0db C:\Neo4j\Neo4jDbBackup-01
```

Step 5 − Use any Windows compression/decompression tool like WinZip, 7 Zip, or WinRAR to zip our Database folder.

# Neo4j Database Restore

Step 1 − Shutdown the database server. Please refer to the previous steps to shut down the server.

Step 2 − Empty the current database folder.

Step 3 − Use any Windows compression/decompression tool like WinZip, 7 Zip, or WinRar to unzip our backup folder.

Step 4 − Open the command prompt and execute the following command.

```
Copy C:\Neo4j\Neo4jDbBackup-01 C:\Ne04j2.0db
```

Step 5 - Now we can observe that our database folder contains working backup files

# Neo4j - Index

Neo4j SQL supports Indexes on node or relationship properties to improve the performance of the application. We can create indexes on properties for all nodes, which have the same label name.

We can use these indexed columns on MATCH or WHERE or IN operator to improve the execution of CQL command.

Creating an Index

Syntax: CREATE INDEX ON:label (node)

Example: CREATE (Mary:employee{name: “Mary Josep", exp: 5})

CREATE INDEX ON:employee(Mary)

# Neo4j - Index

Deleting an Index

Neo4j CQL provides a "DROP INDEX" command to drop an existing index of a Node or Relationshis property.

Syntax: DROP INDEX ON:label(node)

Example: DROP INDEX ON:employee(Mary)

# UNIQUE Constraint

Neo4j CQL provides "CREATE CONSTRAINT" command to create unique constraints on node or relationship properties.

Syntax

```
MATCH (root {name: "Dhawan"})
CREATE UNIQUE (root)-[:LOVES]-(someone)
RETURN someone
```

Example

```
CREATE(Sarad:student{sid:001, name: “Sarad Yadav", branch: “cse"})
CREATE(Sumanth:student{sid:002, name: “Sumanth Rao", branch: “cse"})
CREATE(Roshni:student{sid:003, name: “Roshni", branch: “cse"})

CREATE CONSTRAINT ON (n:student) ASSERT n.sid IS UNIQUE
```

# Neo4j - Drop Unique

Neo4j CQL provides "DROP CONSTRAINT" command to delete existing Unique constraint from a node or relationship property.

Syntax

```
DROP CONSTRAINT ON (node:label)
ASSERT node.id IS UNIQUE
```

Example

```
DROP CONSTRAINT ON (n:student)
ASSERT n.sid IS UNIQUE
```

# Neo4j - Overview

**Create queries**

Create a new node

```
CREATE (a:Person {name:"Théo Gauchoux"}) RETURN a
```

Create a new relationship (with 2 new nodes)

```
CREATE (a:Person)-[k:KNOWS]-(b:Person) RETURN a,k,b
```

**Match queries**

Match all nodes

```
MATCH (n) RETURN n
```

# Neo4j - Overview

Match nodes by label

```
MATCH (a:Person) RETURN a
```

Match nodes by label and property

```
MATCH (a:Person {name:"Théo Gauchoux"}) RETURN a
```

Match nodes according to relationships (undirected)

```
MATCH (a)-[:KNOWS]-(b) RETURN a,b
```

Match nodes according to relationships (directed)

```
MATCH (a)-[:MANAGES]->(b) RETURN a,b
```

# Neo4j - Overview

Match nodes with a WHERE clause

```
MATCH (p:Person {name:"Théo Gauchoux"})-[s:LIVES_IN]->(city:City) WHERE
s.since = 2015 RETURN p,state
```

You can use MATCH WHERE clause with CREATE clause

```
MATCH (a), (b) WHERE a.name = "Jacquie" AND b.name = "Michel" CREATE
(a)-[:KNOWS]-(b)
```

**Update queries**

Update a specific property of a node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" SET p.age = 23
```

# Neo4j - Overview

Replace all properties of a node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" SET p = {name:
"Michel", age: 23}
```

Add new property to a node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" SET p + =
{studies: "IT Engineering"}
```

Add a label to a node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" SET p:Internship
```

# Neo4j - Overview

**Delete queries**

Delete a specific node (linked relationships must be deleted before)

```
MATCH (p:Person)-[relationship]-() WHERE p.name = "Théo Gauchoux"
DELETE relationship, p
```

Remove a property in a specific node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" REMOVE p.age
```

Pay attention to the REMOVEkeyword, it’s not DELETE !

Remove a label from a specific node

```
MATCH (p:Person) WHERE p.name = "Théo Gauchoux" DELETE p:Person
```

# Neo4j - Overview

Delete entire database

```
MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n, r
OR
MATCH(n) DETACH DELETE(n)
```

# Summary

This session will give the knowledge about

- Neo4j CQL General Clauses
- Neo4j CQL Functions
- Neo4j CQL Admin