

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

School of Computing

B.Tech. - Computer Science and Engineering



VTR UGE2021- (CBCS)



Academic Year: 2025-2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name : Database Management Systems

Slot No : S1L4

DBMS TASK - 9 REPORT

Title: CRUD operations in Graph databases

Submitted by:

VTU NO	REGISTER NUMBER	STUDENT NAME
VTU28953	24UECS0493	CH. KISHORE

TASK 9

CRUD operations in Graph databases

AIM:

To perform CRUD operations like creating, inserting, querying, finding, deleting operations on graph spaces.

The steps to get started with Neo4j's Aura Graph Database:

Step1: Copy and paste the following link into your web browser:

<https://neo4j.com/cloud/platform/aura-graph-database/?ref=docs-get-started-dropdown>

Step2: Click on "Start Free."

Step3: Choose the option to "Continue with Google."

Step4: Click the "Open" button.

Step5: After clicking "Open," a text file will be automatically downloaded. This file contains your user ID and password details.

Step6: Copy the password from the downloaded text file and paste it where required.

Step7: Close the "Get started with Neo4j with beginner guides" if it's open.

Step8: You're now ready to begin practicing with the Graph Database.

Create Node with Properties

Properties are the key-value pairs using which a node stores data. Create a node with properties using the CREATE clause and need to specify these properties separated by commas within the flower braces "{}".

Syntax

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

To verify the creation of the node, type and execute the following query in the dollar prompt.

Syntax:

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

Creating Relationships

To create a relationship using the CREATE clause and specify relationship within the square braces “[]” depending on the direction of the relationship it is placed between hyphen “ - ” and arrow “ → ” as shown in the following syntax.

Syntax:

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

Syntax:

```
MATCH (a:LabeofNode1), (b:LabeofNode2)  
WHERE a.name = "nameofnode1" AND b.name = " nameofnode2"  
CREATE (a)-[: Relation]->(b) RETURN a,b
```

Deleting a Particular Node

To delete a particular node and need to specify the details of the node in the place of “n” in the above query.

Syntax:

```
MATCH (node:label {properties ..... }) DELETE node
```

Create a graph database for student course registration, create student and dept node and insert values of properties.

Create a CricketBoard Node:

```
create(cb:CricketBoard{BoardID:'B1D01',Name:'Chennai Cricket Board', Address:'Chennai',  
Phone:9988776699}) return cb
```

Create Team Nodes:

```
create(t1:Team{teamID:'CCB01',BoardID:'B1D01',name:'ABS EXPRESS', Coach:'G.D.RAMESH',  
Captain:'SAMPATH KUMAR'}) return t1
```

```
create(t2:Team{teamID:'CCB02',BoardID:'B1D01',name:'AVG EXPRESS',Coach: 'T.KARTHIK',  
Captain:'Y.JOHN'}) return t2
```

Create Player Nodes:

```
create(p1:Player{PlayerID:'1',TeamID:'CCB01',Name:'Raj',Age:23,DateofBirth:'29-JUN-1996',  
PlayingRole:'Bowler',email:'rajn@gmail.com'}) return p1
```

```
create(p2:Player{PlayerID:'33',TeamID:'CCB01',Name:'Anand',Age:23,DateofBirth:'02-JAN-1999',  
PlayingRole:'Batsman',email:'balajid@gmail.com'}) return p2
```

```
create(p3:Player{PlayerID:'65',TeamID:'CCB02',Name:'Suresh',Age:27,DateofBirth:'02-JUN-1996',  
PlayingRole:'Batsman',email:'sureshd@gmail.comm'}) return p3
```

```
create(p4:Player{PlayerID:'75',TeamID:'CCB02',Name:'Rohit',Age:33,DateofBirth:'02-JUN-1991',  
PlayingRole:'Batsman',email:'srohit@gmail.comm'}) return p4
```

Creating Relationship among CricketBoard and Teams:

```
match(cb:CricketBoard{BoardID:'BID01'}),(t1:Team{teamID:'CCB01'}) create(cb)-[r:has]->(t1) return  
cb,r,t1
```

```
match(cb:CricketBoard{BoardID:'BID01'}),(t2:Team{teamID:'CCB02'}) create(cb)-[r:has]->(t2) return  
cb,r,t2
```

Creating Relationship among Players and Teams:

```
match(p1:Player{PlayerID:'1'}),(t1:Team{teamID:'CCB01'}) create(p1)-[r1:playfor]->(t1) return p1,r1,t1
```

```
match(p2:Player{PlayerID:'33'}),(t1:Team{teamID:'CCB01'}) create(p2)-[r2:playfor]->(t1) return  
p2,r2,t1
```

```
match(p3:Player{PlayerID:'65'}),(t2:Team{teamID:'CCB02'}) create(p3)-[r3:playfor]->(t2) return  
p3,r3,t2
```

```
match(p4:Player{PlayerID:'75'}),(t2:Team{teamID:'CCB02'}) create(p4)-[r4:playfor]->(t2) return  
p4,r4,t2
```

Display All nodes: match(n) return n

Output:

The screenshot shows the Neo4j Web Console interface. On the left, there's a sidebar with 'Database Information' sections for 'Nodes (8)', 'Relationships (7)', and 'Property keys'. The 'Nodes (8)' section highlights 'CricketBoard', 'Player', and 'Team'. The 'Relationships (7)' section highlights 'has' and 'playfor'. The 'Property keys' section lists various keys like Address, Age, BoardID, Captain, Coach, data, DateofBirth, did, dname, eid, email, ename, id, name, Name, nodes, Phone, PlayerID, PlayingRole, and relationships. Below this, a 'Show all (5 more)' link is visible.

In the main area, a terminal window displays the command: `neo4j $ match(n) return n`. The results are shown in a graph view, where nodes are represented by colored circles (yellow for CricketBoard, pink for Player, and purple for Team). Relationships are shown as lines connecting the nodes. A 'Graph' tab is selected, and a 'Table' and 'RAW' tab are also present.

On the right, a 'Results Overview' panel shows summary statistics: Nodes (8), CricketBoard (1), Player (4), and Team (2).

At the bottom, a status bar shows 'Activate Windows Go to Settings to activate Windows.' and system information like '35°C Partly sunny' and '22:05 06-10-2023'.

OUTPUT:

The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar titled "Database Information" with sections for "Nodes (8)" (CricketBoard, Player, Team), "Relationships (7)" (has, playfor), and "Property keys". A query terminal at the top right displays the result of the query: "neo4j \$ match(n) return n". The main area shows a graph visualization with several nodes (yellow, red, blue) and their connections. To the right of the graph, a "Node Details" panel is open for a "Team" node, showing properties like <id>, BoardID, teamID, name, Captain, and Coach. The operating system taskbar at the bottom indicates it's running on Windows 10.

Retrieve particular player details:

```
match(p:Player{PlayerID:'33'}) return p
```

The screenshot shows the Neo4j Workspace interface. On the left, the Database Information panel lists 8 nodes and 7 relationships. A search bar at the top right contains the text "neo4j\$". Below it, a graph view shows a single pink circular node labeled "Anand". To the right, the Node Details panel is open for the "Player" node with ID 1, showing properties like PlayerID, PlayingRole, DateofBirth, TeamID, email, Age, and Name. The system status at the bottom indicates "Started streaming 1 record after 44ms and completed after 46ms".

Update particular player details:

```
match(p:Player{PlayerID:'1'}) set p.age=27 return p
```

Output:

The screenshot shows the Neo4j Workspace interface after running the update query. The Node Details panel now displays a "Player" node with ID 8, showing updated properties: PlayerID, PlayingRole, DateofBirth, TeamID, age, email, Age, and Name. The name has been changed to "Raj". The system status at the bottom indicates "Started streaming 1 record after 44ms and completed after 46ms".

Delete particular player from the team:

```
match(p:Player{PlayerID:'33'}) delete p
```

The screenshot shows the Neo4j web interface. On the left, there's a sidebar titled "Database Information" with sections for "Nodes (8)" and "Relationships (7)". The "Nodes" section lists three nodes: "CricketBoard", "Player", and "Team". The "Relationships" section lists two relationships: "has" and "playfor". The main area is a query editor with the URL <https://workspace-preview.neo4j.io/workspace/query>. The query entered is `match(p:Player{PlayerID:'33'}) delete p`. Below the query, an error message is displayed: **Neo.ClientError.Schema.ConstraintValidationFailed**. The message states: "Cannot delete node<1>, because it still has relationships. To delete this node, you must first delete its relationships.".

Result:

Thus the CRUD operations like creating, inserting, querying, finding, deleting operations on graph spaces were executed successfully.