

**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:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU28873	24UECS0627	K.Lahari

## 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:'BID01',Name:'Chennai Cricket Board', Address:'Chennai',  
Phone:9988776699}) return cb
```

### **Create Team Nodes:**

```
create(t1:Team{teamID:'CCB01',BoardID:'BID01',name:'ABS EXPRESS',  
Coach:'G.D.RAMESH', Captain:'SAMPATH KUMAR'}) return t1
```

```
create(t2:Team{teamID:'CCB02',BoardID:'BID01',name:'AVG EXPRESS',Coach:  
'T.KARTHIKH', 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.comm'}) 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 Browser interface with the following details:

- Database Information:** Shows 8 nodes and 7 relationships. Nodes include CricketBoard, Player (4), and Team (2). Relationships include has and playfor.
- Query Panel:** The query entered is `match(n) return n`. The results are displayed in three formats: Graph, Table, and RAW.
- Graph View:** A visualization of the nodes and their connections. Nodes are represented by colored circles: yellow for CricketBoard, pink for Player, and purple for Team.
- Table View:** Shows the following data:

Node Type	Count
CricketBoard	1
Player	4
Team	2
- RAW View:** Displays the raw Cypher results as a list of node IDs.
- Results Overview:** Summary of the query execution.
- System Status:** Shows system information like "Last update: 9:31:11 pm" and a Windows activation message.

## OUTPUT:

The screenshot shows the Neo4j Aura workspace interface. On the left, there's a sidebar with 'Database Information' sections for Nodes (8), Relationships (7), and Property keys. The main area displays a query result for 'match(n) return n'. The results are shown in a graph view where nodes are represented by colored circles (yellow, pink, purple, blue) and relationships by lines. To the right of the graph, a 'Node Details' panel is open for a yellow node labeled 'Team'. The node details table includes columns for <id>, BoardID, teamID, name, Captain, and Coach. Below the graph, a status bar shows 'Last update: 9:31:11 pm'.

## Retrieve particular player details:

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

This screenshot shows the same Neo4j Aura workspace after running the query 'match(p:Player{PlayerID:'33'}) return p'. The results are displayed in a graph view with a single pink node labeled 'Anand'. A 'Node Details' panel to the right shows the properties for this node, including <id>, PlayerID, PlayingRole, DateOfBirth, TeamID, email, Age, and Name. The status bar 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 Aura workspace interface. On the left, there's a sidebar with 'Database Information' sections for 'Nodes (8)', 'Relationships (7)', and 'Property keys'. The 'Nodes (8)' section lists 'CricketBoard', 'Player', and 'Team'. The 'Player' node is highlighted in yellow. In the main query editor area, the command `match(p:Player{PlayerID:'1'}) set p.age=27 return p` is run. Below the query, a node detail card for 'Player' shows a circular node with the name 'Raj'. The node properties table includes:

<id>	8
PlayerID	"1"
PlayingRole	"Bowler"
DateofBirth	"29-JUN-1996"
TeamID	"CCB01"
age	27
email	"rajn@gmail.com"
Age	23
Name	"Raj" Settings to activate Windows.

## Delete particular player from the team:

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

The screenshot shows the Neo4j Aura workspace interface. The 'Player' node (highlighted in pink) is selected. In the query editor, the command `match(p:Player{PlayerID:'33'}) delete p` is run. A red error message appears: **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.