# 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:

1. Step1: Copy and paste the following link into your web browser: https://neo4j.com/cloud/platform/aura-graph-database/?ref=docs-get-started-dropdown
2. Step2: Click on "Start Free."
3. Step3: Choose the option to "Continue with Google."
4. Step4: Click the "Open" button.
5. Step5: After clicking "Open," a text file will be automatically downloaded. This file contains your user ID and password details.
6. Step6: Copy the password from the downloaded text file and paste it where required.
7. Step7: Close the "Get started with Neo4j with beginner guides" if it's open.
8. 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 department nodes and insert values of properties.

Create a University Node:  
create(u:University{UniversityID:'UID01',Name:'ABC University', Address:'Chennai', Phone:9876543210}) return u

Create Department Nodes:  
create(d1:Department{DeptID:'DPT01',UniversityID:'UID01',name:'Computer Science', HOD:'Dr.S.Ravi'}) return d1  
create(d2:Department{DeptID:'DPT02',UniversityID:'UID01',name:'Information Technology', HOD:'Dr.K.Kumar'}) return d2

Create Student Nodes:  
create(s1:Student{StudentID:'S01',DeptID:'DPT01',Name:'Anjali',Age:20,DateofBirth:'12-MAY-2004', Course:'B.Tech',email:'anjali@gmail.com'}) return s1  
create(s2:Student{StudentID:'S02',DeptID:'DPT01',Name:'Rahul',Age:21,DateofBirth:'18-JUL-2003', Course:'B.Tech',email:'rahul@gmail.com'}) return s2  
create(s3:Student{StudentID:'S03',DeptID:'DPT02',Name:'Divya',Age:22,DateofBirth:'09-MAR-2002', Course:'B.Tech',email:'divya@gmail.com'}) return s3  
create(s4:Student{StudentID:'S04',DeptID:'DPT02',Name:'Arun',Age:23,DateofBirth:'01-JAN-2001', Course:'B.Tech',email:'arun@gmail.com'}) return s4

Creating Relationship among University and Departments:  
match(u:University{UniversityID:'UID01'}),(d1:Department{DeptID:'DPT01'}) create(u)-[r:has]->(d1) return u,r,d1  
match(u:University{UniversityID:'UID01'}),(d2:Department{DeptID:'DPT02'}) create(u)-[r:has]->(d2) return u,r,d2

Creating Relationship among Students and Departments:  
match(s1:Student{StudentID:'S01'}),(d1:Department{DeptID:'DPT01'}) create(s1)-[r1:enrolled\_in]->(d1) return s1,r1,d1  
match(s2:Student{StudentID:'S02'}),(d1:Department{DeptID:'DPT01'}) create(s2)-[r2:enrolled\_in]->(d1) return s2,r2,d1  
match(s3:Student{StudentID:'S03'}),(d2:Department{DeptID:'DPT02'}) create(s3)-[r3:enrolled\_in]->(d2) return s3,r3,d2  
match(s4:Student{StudentID:'S04'}),(d2:Department{DeptID:'DPT02'}) create(s4)-[r4:enrolled\_in]->(d2) return s4,r4,d2

Display All nodes: match(n) return n  
  
  
  
  
Retrieve particular student details:  
match(s:Student{StudentID:'S02'}) return s

Update particular student details:  
match(s:Student{StudentID:'S01'}) set s.age=21 return s  
Output:

Delete particular student from the department:  
match(s:Student{StudentID:'S02'}) delete s

## Result:

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