CSE 4410

DATABASE MANAGEMENT SYSTEMS II LAB

Notes On: Neo4J

PREPARED BY :

DR. ABU RAIHAN MOSTOFA KAMAL || PROFESSOR
raihan.kamal@iut-dhaka.edu

ZANNATUN NAIM SRISTY || LECTURER
zannatunnaim@iut-dhaka.edu

MD. RAFID HAQUE || LECTURER
rafidhaque@iut-dhaka.edu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ISLAMIC UNIVERSITY OF TECHNOLOGY

MARCH 28, 2023

1 Getting Started with Neo4J

1.1 Create

- Creating a node

```
> create (variable:Label{property:'value',...,property:'value'})
```

- Creating multiple nodes

```
> create (variable: Label{property:'value'}), (variable: Label{
property:'value'})
```

- Creating a single node with multiple Labels

```
> create (variable:Label:...:Label{property:'value'})
```

- Create a relationship between nodes

```
> match (variable_1:Label_1), (variable_2:Label_2)
where variable_1.property='value' and variable_2.property='value'
create (variable_1)-
    [variable_3:Relationship_name {property:'value'}]->
    (variable_2)
```

- Create nodes with a relationship

```
> create (variable_1:Label_1{property:'value'}) -
    [variable_3:Relationship_name {property:'value'}] ->
    (variable_2:Label_2{property:'value'})
```

1.2 Delete

Deletion in Neo4J, is almost similar to SQL language. Because it also follows the same rule as SQL. One has to delete the relationships that are adjacent to a node in order to delete that node.

- To delete all the nodes (when there is no relationship)

```
> match (n) delete (n)
```

- To delete the full graph (with relation)

```
> match (n) detach delete (n)
```

- To delete a certain node

```
> match (variable {property: "value"}) delete variable
```

- To delete a certain node with an adjacent relationship

```
> match (variable {property: "value"}) detach delete variable
```

- To delete a relationship

```
> match (variable {property: "value"}) -
    [variable_3:Relationship_name] ->
    (variable {property: "value"})
    delete variable
```

1.3 Simple Query

- To see the whole graph

```
> match (variable) return variable
```

- To see the nodes of a specific label

```
> match (variable: Label) return variable
```

- To see the properties of a specific label

```
> match (variable: Label) return variable.property
```

- To show node based on property

```
> match (variable: Label)
    where variable.property='value'
    return variable

> match (variable: Label {property='value'})
    return variable
```

- And/ or operator

```
> match (variable: Label)
    where variable.property='value' and variable.property='value'
    return variable
```

- Limiting the nodes

```
> match (variable: Label)
    where variable.property='value'
    return variable
    limit value
```

- Skipping node

```
> match (variable: Label)
    where variable.property='value'
    return variable
    skip value
```

- Sorting the nodes

```
> match (variable: Label)
    where variable.property='value'
    return variable
    order by variable.property Desc
```

- Showing nodes of multiple labels

```
> match (variable_1: Label_1), (variable_2: Label_2)
    return variable_1, variable_2
```

- Showing all nodes that have a certain relationship

```
> match (variable_1: Label_1) -
    [variable_3: Relationship_name] ->
    (variable_2: Label_2)
    return variable_1, variable_2
```

- Nested search

```
> match (variable_1: Label_1{property:'value'}) -
    [variable_3: Relationship_name] ->
    (variable_2: Label_2)
        match (variable_2)-
            [variable_5: Relationship_name] ->
            (variable_4: Label_4)
            where variable_5.property='value'
            return variable_2
```

- Aggregate functions

```
> match (variable_1: Label_1) -
    [variable_3: Relationship_name] ->
    (variable_2: Label_2)
    return variable_1.property, count(variable_3)

> match (variable_1: Label_1) -
    [variable_3: Relationship_name] ->
    (variable_2: Label_2)
    return variable_1.property, avg(variable_3.property)
```