Graphe oriented databases : Neo4J by ACHRAF KHABAR

1 - General :

- Data in graphe databases is stored in somthing called `nodes`.
- Inside a node it can be anything : person, id, name, laptops, cities, world ...
- I need to store for exaples students and their schools, i stored *Ashraf khabar* in node and *ENSAT*, Ashraf khabar studies in ENSAT so i need to make an arrow from ashraf to ENSAT not the opposit : this is what we call `oriented graph`.
- If a have a social media, I have *ashraf khabar* in node, and *sami aouad* in the other node, ashraf and sami are both friends, so i need tom make two ligne of relationships with arrows between ashraf and sami, this is what we call `non oriented graph`.
- Each node has somthing called `label`, which is a name of the node, for example we have *Node={name = 'ashraf khabar', age = 22}* , this is labeled by `Person` .
- The realationship can have also *properties*, for example `lebron james` in node1, and `LA Lakers` in other node naed node2, the relationship `Plys for` goes from node1 to node2, with property named `salary` with value of **40M$**.

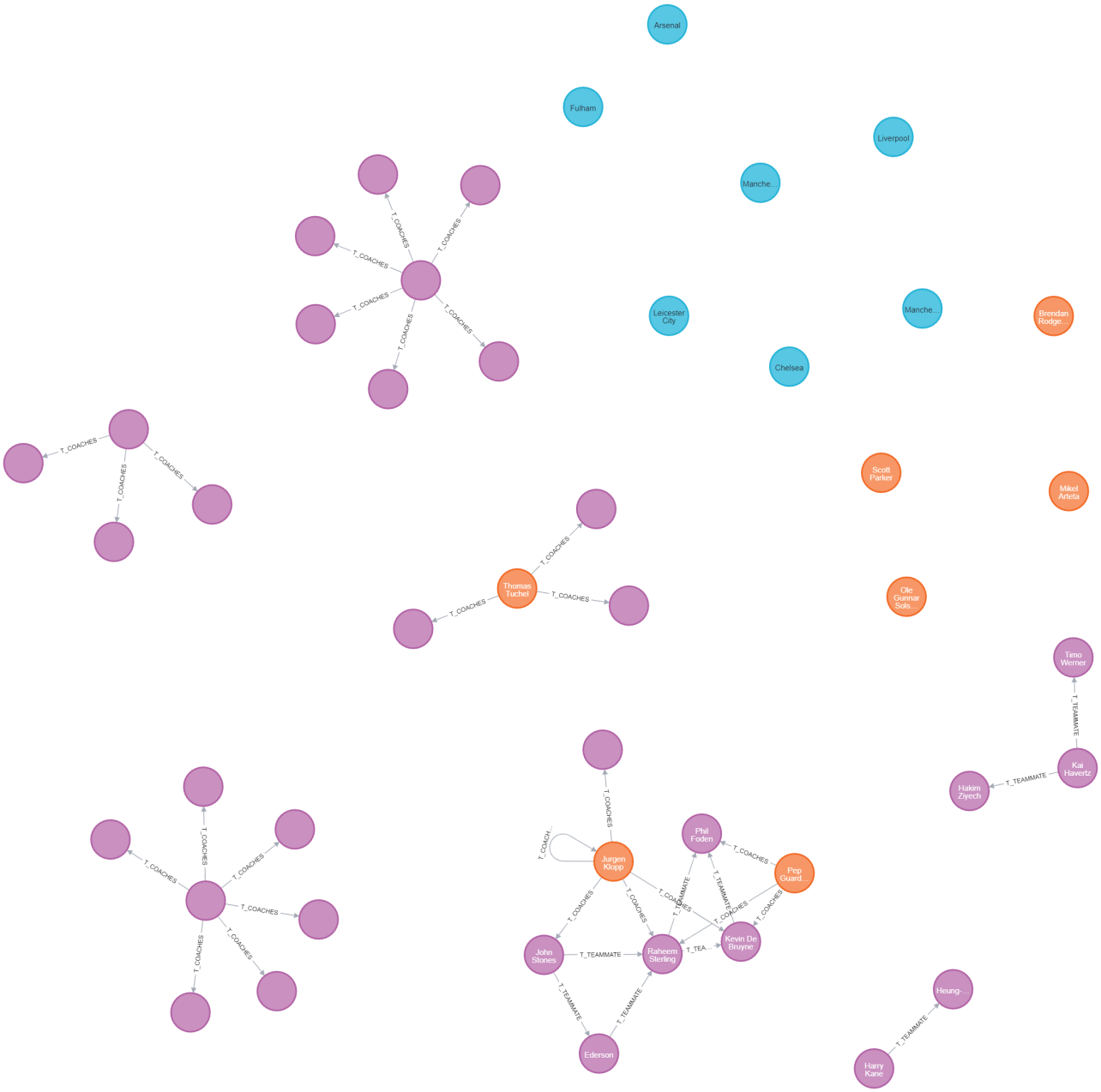
2 - Querying Data :

- First we need to create the nodes and the realtion , we gonna work for the eample of Premiere league players, coeaches and teams :

```
  CREATE
      (john:T_PLAYER {name:"John Stones", age: 27, number: 5, height: 1.88,
weight: 70}),
      (ederson:T_PLAYER {name:"Ederson", age: 28, number: 31, height: 1.88,
weight: 86}),
      (raheem:T_PLAYER {name:"Raheem Sterling", age: 27, number: 7, height:
1.70, weight: 69}),
      (kevin:T_PLAYER {name:"Kevin De Bruyne", age: 30, number: 17, height:
1.81, weight: 68}),
      (phil:T_PLAYER {name:"Phil Foden", age: 21, number: 47, height: 1.71,
weight: 68}),
      (harry:T_PLAYER {name:"Harry Kane", age: 28, number: 10, height: 1.88,
weight: 86}),
      (son:T_PLAYER {name:"Heung-Min Son", age: 29, number: 7, height: 1.83,
weight: 77}),
      (hakim:T_PLAYER {name:"Hakim Ziyech", age: 28, number: 22, height:
1.82, weight: 68}),
      (kai:T_PLAYER {name:"Kai Havertz", age: 22, number: 29, height: 1.89,
```

```
weight: 85}),
      (timo:T_PLAYER {name:"Timo Werner", age: 26, number: 11, height: 1.81,
weight: 75}),

      (jurgen:T_COACH {name: "Jurgen Klopp"}),
      (pep:T_COACH {name: "Pep Guardiola"}),
      (thomas:T_COACH {name: "Thomas Tuchel"}),
      (ole:T_COACH {name: "Ole Gunnar Solskjaer"}),
      (mikel:T_COACH {name: "Mikel Arteta"}),
      (brendan:T_COACH {name: "Brendan Rodgers"}),
      (scott:T_COACH {name: "Scott Parker"}),

      (man_city:T_TEAM {name:"Manchester City"}),
      (liverpool:T_TEAM {name:"Liverpool"}),
      (chelsea:T_TEAM {name:"Chelsea"}),
      (manchester_united:T_TEAM {name:"Manchester United"}),
      (arsenal:T_TEAM {name:"Arsenal"}),
      (leicester:T_TEAM {name:"Leicester City"}),
      (fulham:T_TEAM {name:"Fulham"}),

      (john)-[:T_TEAMMATE]->(ederson),
      (john)-[:T_TEAMMATE]->(raheem),
      (ederson)-[:T_TEAMMATE]->(raheem),
      (raheem)-[:T_TEAMMATE]->(kevin),
      (raheem)-[:T_TEAMMATE]->(phil),
      (kevin)-[:T_TEAMMATE]->(phil),
      (harry)-[:T_TEAMMATE]->(son),
      (kai)-[:T_TEAMMATE]->(hakim),
      (kai)-[:T_TEAMMATE]->(timo),

      (jurgen)-[:T_COACHES]->(john),
      (jurgen)-[:T_COACHES]->(kevin),
      (jurgen)-[:T_COACHES]->(raheem),
      (jurgen)-[:T_COACHES]->
      (jurgen)-[:T_COACHES]->(virgil),
      (steven)-[:T_COACHES]->(sadio),
      (steven)-[:T_COACHES]->(mo),
      (steven)-[:T_COACHES]->(jordan),
      (steven)-[:T_COACHES]->(andy),
      (steven)-[:T_COACHES]->(trent),
      (steven)-[:T_COACHES]->(alisson),
      (pep)-[:T_COACHES]->(kevin),
      (pep)-[:T_COACHES]->(raheem),
      (pep)-[:T_COACHES]->(phil),
      (zinedine)-[:T_COACHES]->(karim),
      (zinedine)-[:T_COACHES]->(toni),
      (zinedine)-[:T_COACHES]->(sergio),
      (thomas)-[:T_COACHES]->(robert),
      (thomas)-[:T_COACHES]->(manuel),
      (thomas)-[:T_COACHES]->(alphonso),
      (ronald)-[:T_COACHES]->(leo),
      (ronald)-[:T_COACHES]->(sergio_r),
      (ronald)-[:T_COACHES]->(ansu),
      (ronald)-[:T_COACHES]->(frenkie),
```

```
        (ronald)-[:T_COACHES]->(antoine),
        (ronald)-[:T_COACHES]->(pedri)
```



- In neo4j we can store `Nodes` or `Relations`.

3 - Fetching Data :

- We gonna use the `Cypher` language.
- We gonna use the word `MATCH` in order to querying nodes.
- In order to get all the `nodes` we have :

```
MATCH (n)  -- variable called n representing node
RETURN n
```

- How about getting only `Player` nodes :

```
MATCH (n:T_PLAYER) -- T_PLAER is the label
RETURN n
```

PS : n is a variable , you can call it whatever you want :

```
MATCH (players:T_PLAYER)
RETURN players
```

- How to fetch a specifique property :

If we hover on a Player, we gonna find some properties :

id	age	height	name	number	weight
7	28	1.82	Hakim Ziyech	22	68

We need to execute the commande :

```
MATCH (players:T_PLAYER)
RETURN players.name
```

The result is not a `graph`, but a `table` of `names` :

players.name
John Stones
Ederson
Raheem Sterling
Kevin De Bruyne
Phil Foden
Harry Kane
Heung-Min Son
Hakim Ziyech

We can get more `properties` :

```
MATCH (players:T_PLAYER)
RETURN players.name, players.age
```

players.name	players.age
John Stones	27
Ederson	28
Raheem Sterling	27
Kevin De Bruyne	30
Phil Foden	21
Harry Kane	28
Heung-Min Son	29
Hakim Ziyech	28

How about changing the `player.name` with `names` (creating alias) :

```
MATCH (players:T_PLAYER)
RETURN players.name AS names, players.age AS ages
```

names	ages
John Stones	27
Ederson	28
Raheem Sterling	27
Kevin De Bruyne	30
Phil Foden	21
Harry Kane	28
Heung-Min Son	29
Hakim Ziyech	28

4 - Feltering Data :

- For example if we want only `Hakim ziyech` node :

```
MATCH (n:T_PLAYER)
WHERE n.name = "Hakim Ziyech"
RETURN n
```

I filtered with `name` because we have only on hakim ziyech in the premiere league.

- We can make it in other way on `Cypher` language :

```
MATCH (n:T_PLAYER {name : "Hakim Ziyech"})
RETURN n
```

- We can filter based on multiple things :

```
MATCH (n:T_PLAYER {name: "Hakim Ziyech", age: 20})
RETURN n
```

PS : It's not like `sql` language in term of capital letters, so "*Hakim Ziyech*" is "*hakim ziyech*" .

- We can get the opposit of where *(<>)* :

```
MATCH (n:T_PLAYER)
WHERE n.name <> "Hakim Ziyech"
RETURN n
```

We gonna have all the player but "*Hakim Ziyech*" .

- How about *(>)* and *(<)* :

```
MATCH (n:T_PLAYER)
WHERE n.name >= "Hakim Ziyech"
RETURN n
```

PS : we can use : `<, >, <=, >=`.

- Arithmic opperations are also allowed here :

```
MATCH (n:T_PLAYER)
WHERE (n.age * (n.heigh / 20)) = 20
RETURN n
```

- We can do also the `AND` and `OR` :

```
MATCH (n:T_PLAYER)
WHERE n.age > 18 AND n.age < 29
RETURN n
```

- Limiting the feltring :

```
MATCH (n:T_PLAYER)
WHERE n.age > 18 AND n.age < 29
RETURN n
LIMIT 5
```

But how about if we want to *skip* the first 5 nodes and get the 5 next nodes :

```
MATCH (n:T_PLAYER)
WHERE n.age > 18 AND n.age < 29
RETURN n
SKIP 5
LIMIT 5
```

- *ORDER BY*, *DESC* and *ASC*:

```
MATCH (n:T_PLAYER)
WHERE n.age > 18 AND n.age < 29
RETURN n
ORDER BY n.heigh DESC
```

- What about if we want to get more than one `Node` :

```
MATCH (player:T_PLAYER), (coach:T_COACH)
RETURN player, coach
```

5 - Queryin nodes based on realtionships:

- We want to filter the data based on relationships, for example getting all the `Players` whom playe to `Chelsea` :

```
MATCH (player:T_PLAYER) -[:PLAYS_FOR]-> (team:T_TEAM)
WHERE team.name = "Chelsea"
RETURN player
```

The relation is in one side, that's why we use `->` , we can use in the other side .

```
MATCH (team:T_TEAM) <-[:PLAYS_FOR]- (player:T_PLAYER)
WHERE team.name = "Chelsea"
RETURN player
```

and we got the same result for both queries .

- Properties in realtionship : Every `Relationship` has a `Property` besides the *`ID`* :

```
MATCH (player:T_PLAYER) -[contract:PLAYS_FOR]-> (team:T_TEAM)
WHERE contract.salary > 20000
RETURN player
```

- Adding some filters :

```
MATCH (player:T_PLAYER {name : "Hakim Zieych"} ) -[:TEAMMATES]->
(teammate:T_PLAYER)
RETURN teammate
```

This query retunr all *teammates* of `Hakim Ziyech` .

6 - Aggregating data:

- We all know now hhow to extract the data from a node, feltring that data based on properties using only the close `MATCH`, `WHERE`, `RETURN` ...
- We can make more than one `MATCH` in same query, for example if we want to extract all the teammates of `Hakim Ziyech` whom has a salary higher than 200000 :

```
MATCH (hakim:T_PLAYER {name : "Hakim Ziyech"}) -[:TEAMMATE]->
(temmate:T_PLAYER)
MATCH (teammate) -[contract:PLAYS_FOR] -> (team:TEAM)
WHERE contract.salary >= 200000
RETURN teammate
```

- We can use the aggregation functions as `COUNT()` and `AVG()` :

```
MATCH (player:PLAYER) -> [gp:PLAYED_AGAINST] ->(:T_TEAM)
RETURN player.name, AVG(gp.points)
```

Aggregation functions	**Role**
`COUNT()`	Counting the number of elements
`AVG()`	Calculating the average
`MAX()`	Calculating the maximum
`MIN()`	Calculating the minimum

7 - Modifying data:

- Deleting data is not allowed no matter the conditions, we need to be sure that the node we want to delete has no relationships :

```
MATCH (player:T_PLAYER {id : 25})
DELETE player
```

This can raise an exception `Neo.ClientError.Schema.ConstraintValidationFailed` because that node still has some relationships , in order to solve this issue, we need to modify the previous query to :

```
MATCH (player:T_PLAYER {id : 25})
DETACH DELETE player
```

We detach the node from other nodes so we can delete it without any probleme.

- How about if `Hakim Ziyech` move to other club, so we neet to delete the realtionship of `Playes_for` between `Hakim Ziyech` and `Chelsea` :

```
MATCH (player:T_PLAYER {name : "Hakim Ziyech"}) - [rel:PLAYES_FOR] ->
(:T_TEAM)
DELETE rel
```

Here we don't need to use the word `DETACH` because we are deleting a realtionship not a node.

- How about creating a new Node :

```
CREATE (:T_PLAYER {name: "Ashraf khabar", age : 22})
```

We can make also multiple labels :

```
CREATE (:T_PLAYER:T_MANAGER {name: "Ashraf khabar", age : 22})
```

- We said that `Hakim Ziyech` moved to other club :

```
CREATE (:T_PLAYER{name : "Hakim Ziyech"}) -[:PLAYS_FOR {salry : 2500000}]->
(:T_TEAM {name : "Manchester United"})
```

- If we want to update a node :

```
MATCH (player {id : 23})
RETURN
```

This query retun nothing eventhough there is a player with `id` = 0 .

```
MATCH (player:T_PLAYER)
WHERE ID(player) = 23
RETURN player
```

Now it gonna work without problems . Now we can update our data with any problem using the query commande :

```
MATCH (player:T_PLAYER)
WHERE ID(player) = 23
SET player.name = "Ashraf KHABAR", player.age = 23
RETURN player
```

I can add other label for the node :

```
MATCH (player:T_PLAYER)
WHERE ID(player) = 23
SET player:FN9
RETURN player
```