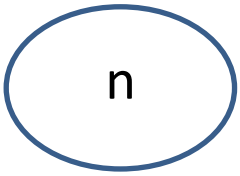
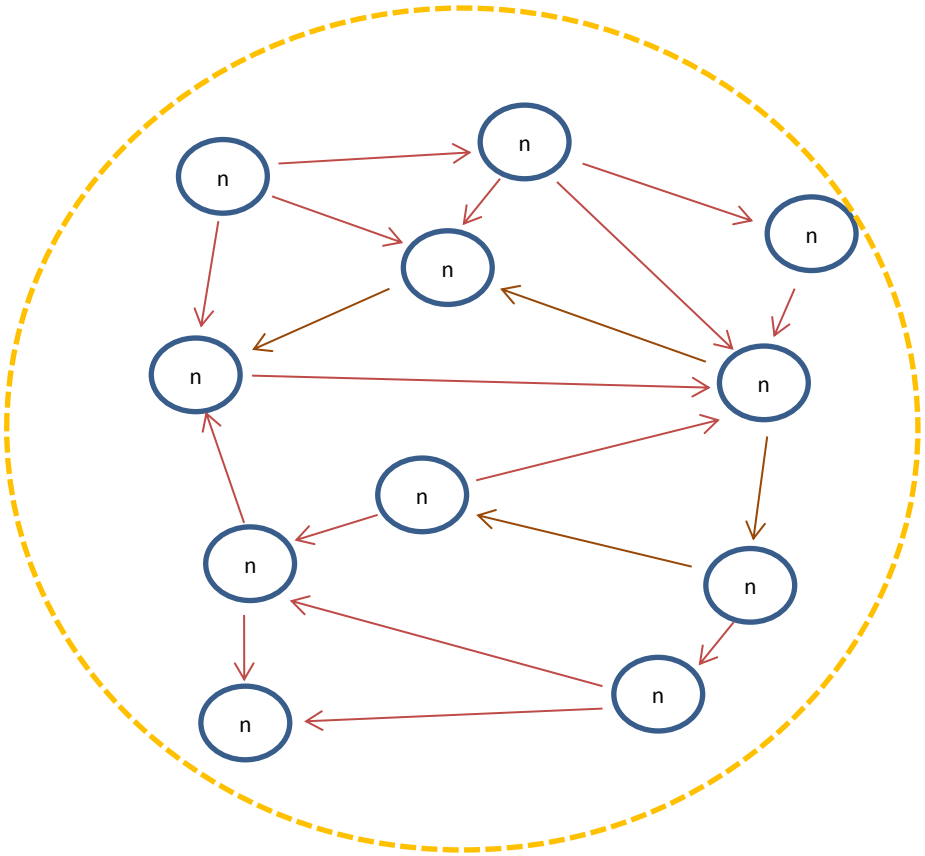
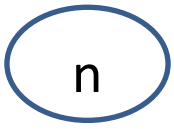
*A **Path** is series of connected nodes and relationships. Paths can be matched by a pattern.*



Pattern ‘n’ means visit every node to see if it matches and assign the result to variable ‘n’. So every node will match!



Every node will match this pattern!

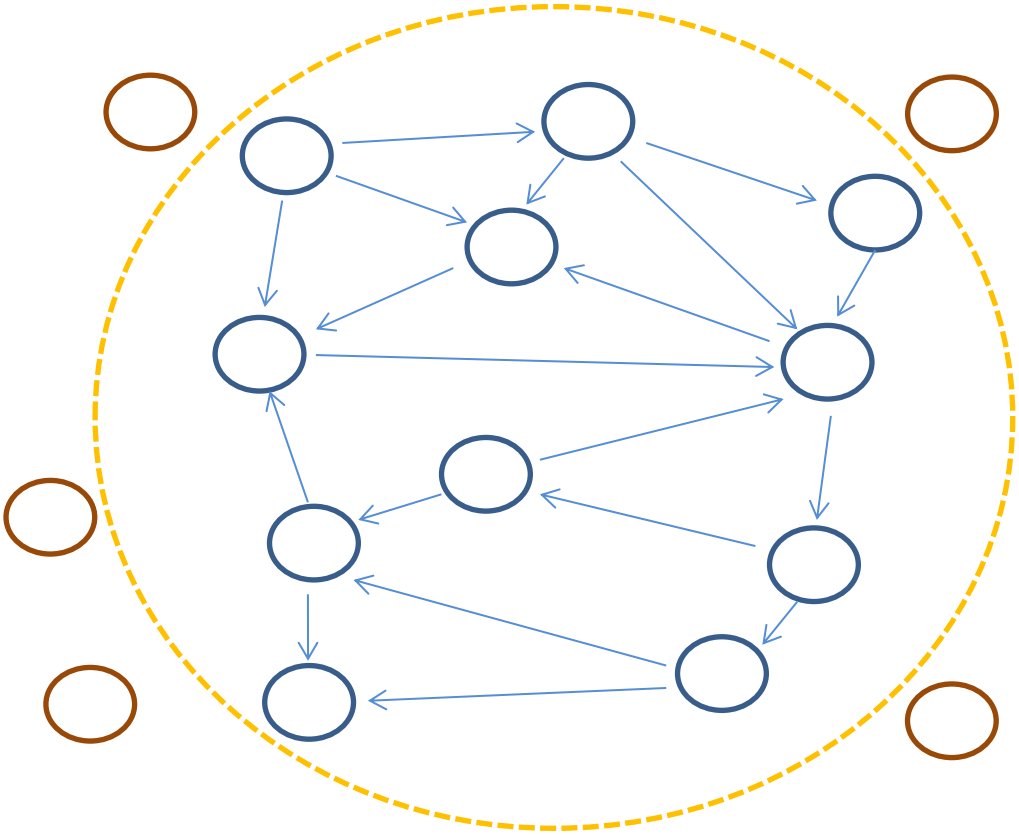


MATCH (n) -Will return all nodes in variable ‘n’ as the pattern is just ( ) for node.

(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))

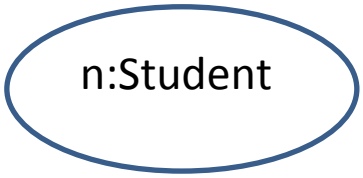
Pattern will match every single node that has a relationship with another node. Looks like:

MATCH (a) - - >(b)



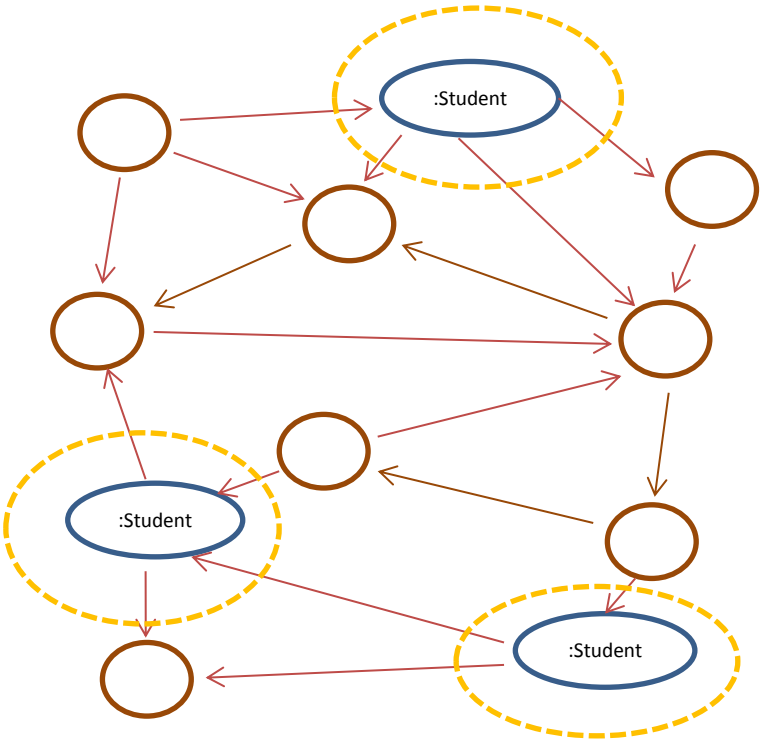
Every node with a Relationship!

(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))



Pattern ':Student' means visit every node to see if it matches pattern node with label 'Student' and assign the result to variable 'n'. So every *Student* labelled node will match! Looks like:

MATCH (n :Student)

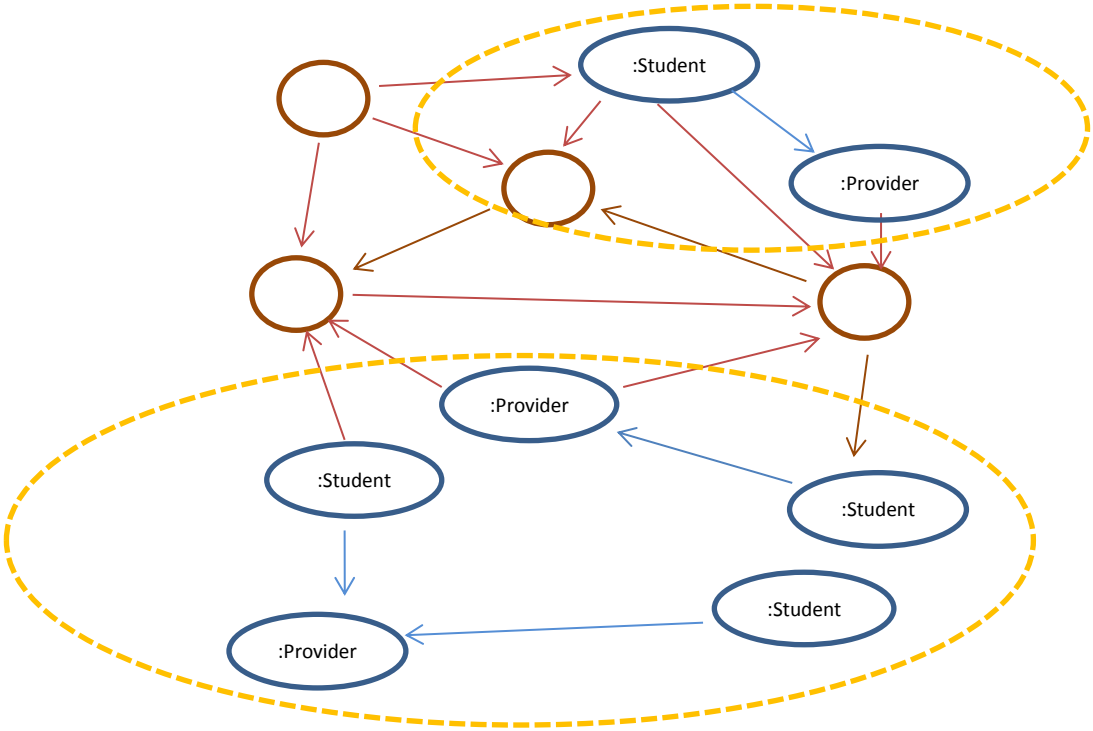


Every 'Student' node will match this pattern!

(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))

Pattern will match every single node with a label of 'Student' that has any kind of relationship with another node with a label of 'Provider'.

Looks like: MATCH (a :Student) - - >(b :Provider)

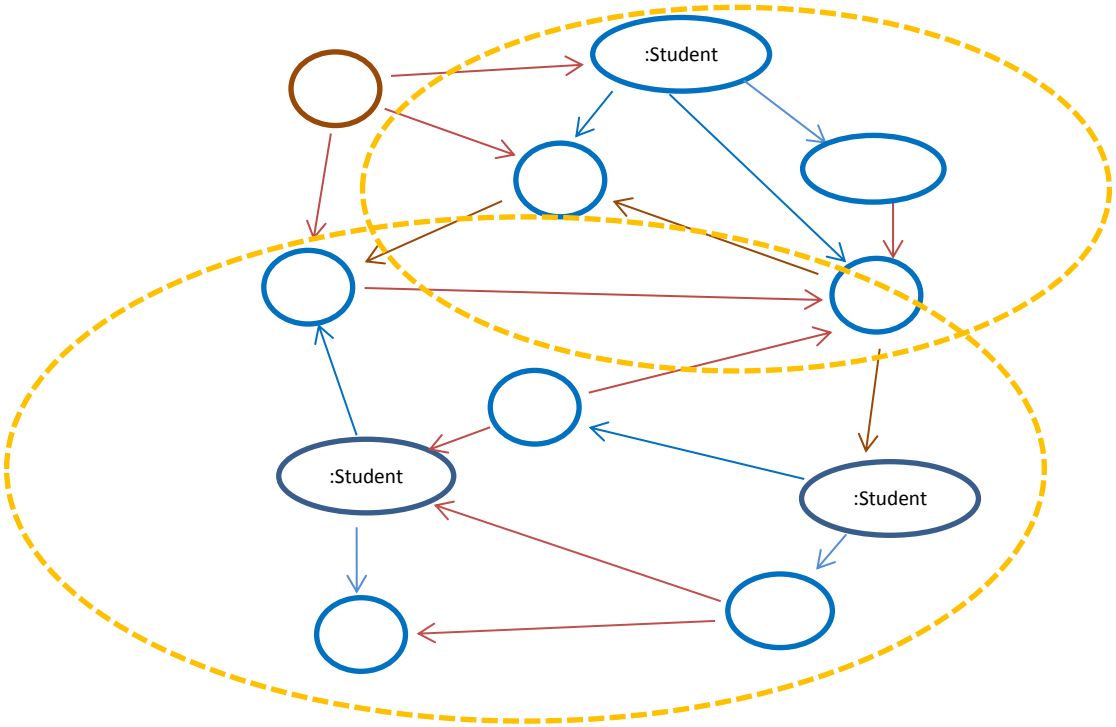


Every 'Student' node, has a relationship with any node with 'Provider' label.

(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))

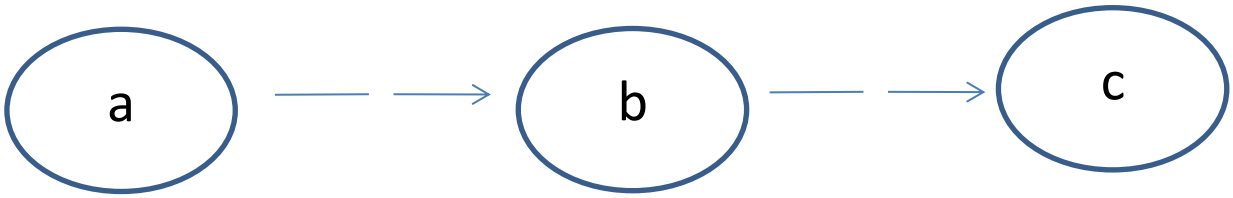
Pattern will match every single node with a label of 'Student' that has any kind of relationship with another node.

Looks like: MATCH (a :Student) - - >(b)

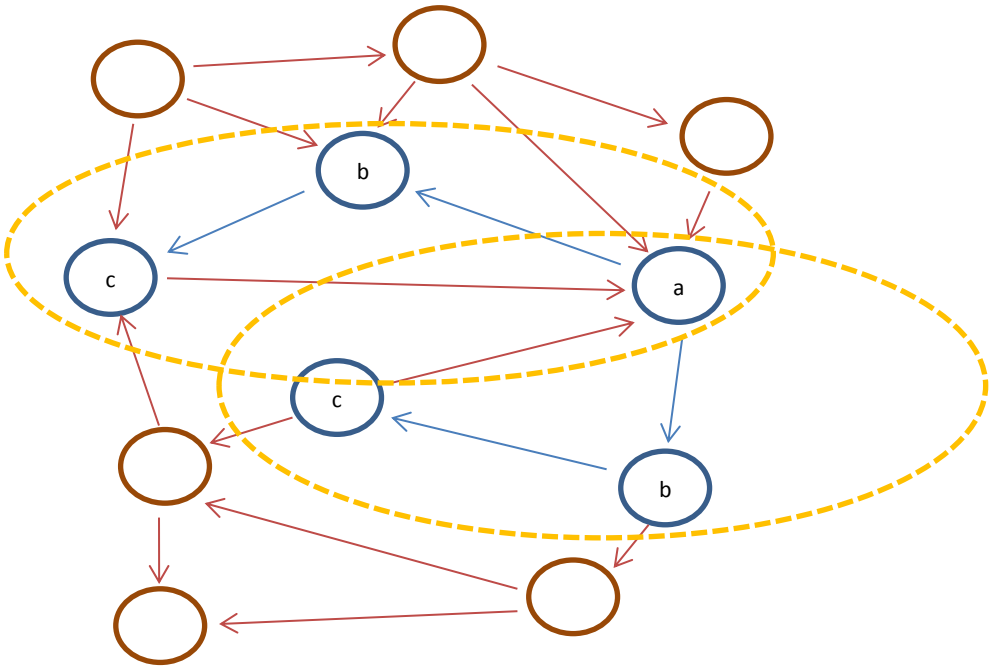


Every 'Student' node, that has a relationship with any node.

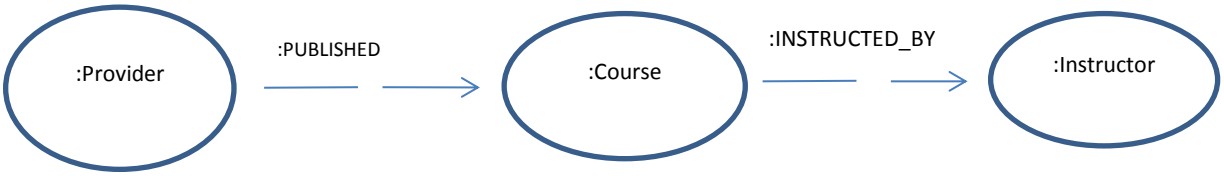
(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))



'a' is connected to 'b' which is connected to 'c'

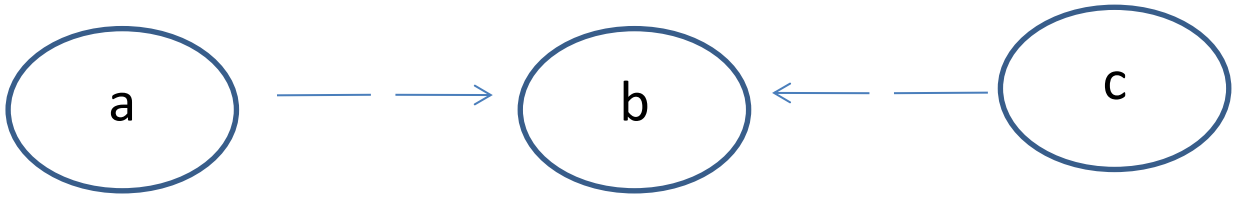


The 'paths' returned by the Cyphers Matching the Pattern (a)→(b)→(c)

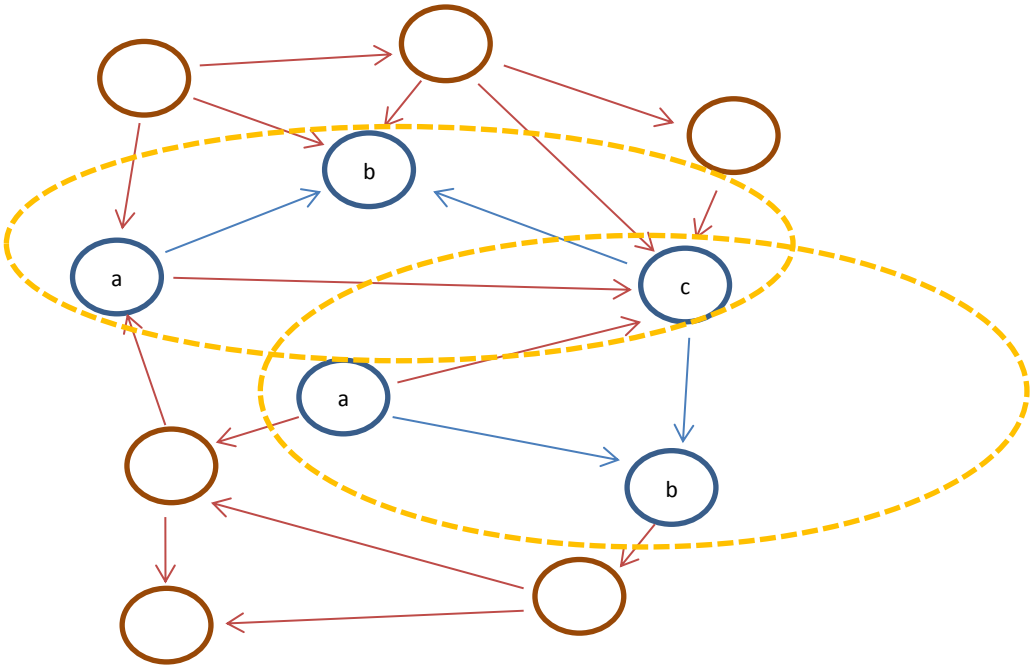


'Provider' node connected to 'Course' nodes and 'Course' nodes connected to 'Instructor'

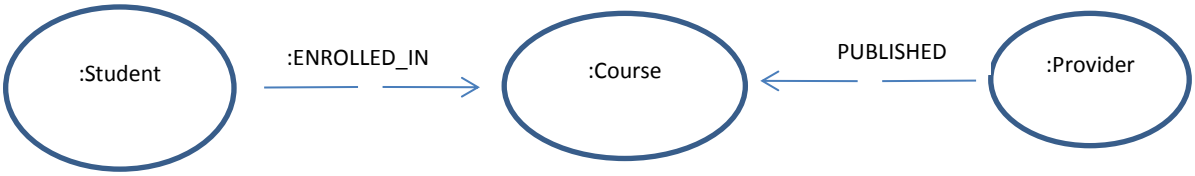
(HINT: Always consider the direction of the Relationship, the arrow shows the direction, also NO arrow = bi-directional (both directions))



Both ‘a’ and ‘c’ are directed towards ‘b’



The ‘paths’ returned by the Cyphers Matching the Pattern (a)→(b)← (c)



‘Student’ nodes connected to ‘Course’ nodes and ‘Provider’ nodes connected to ‘Course’ nodes