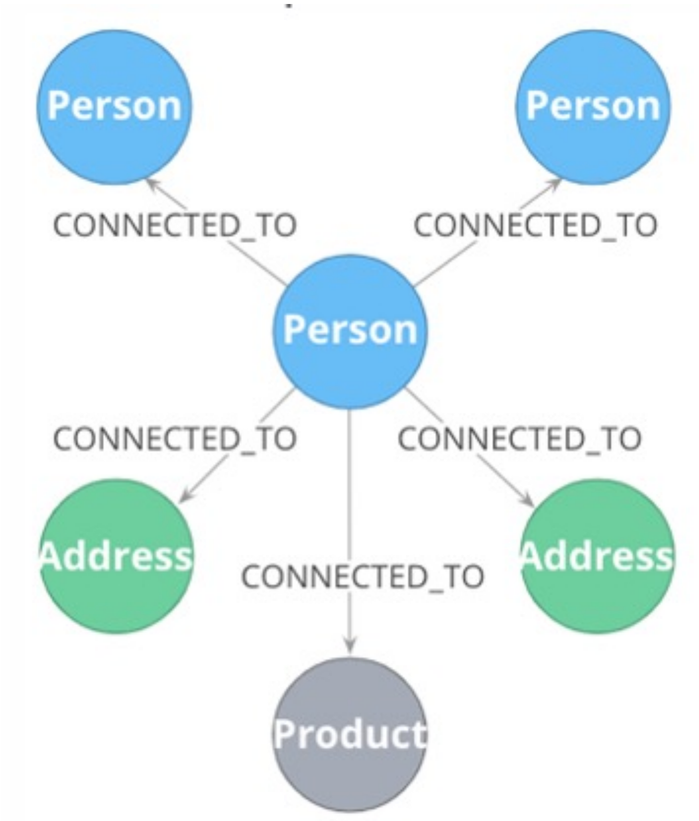
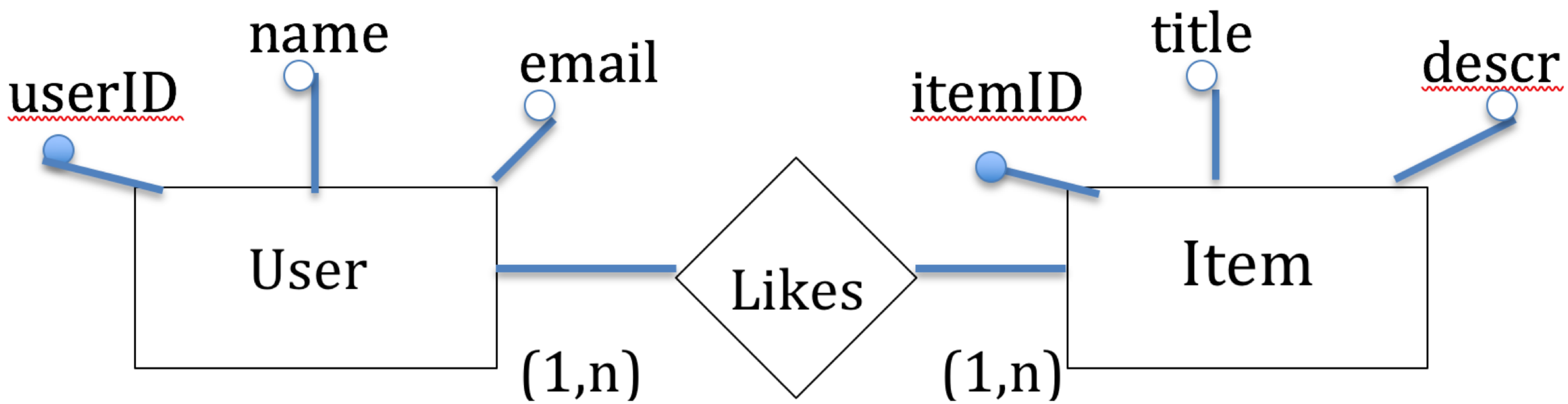
Graph Data Modelling
Wooclap

which problem (if any) do you see with this graph?



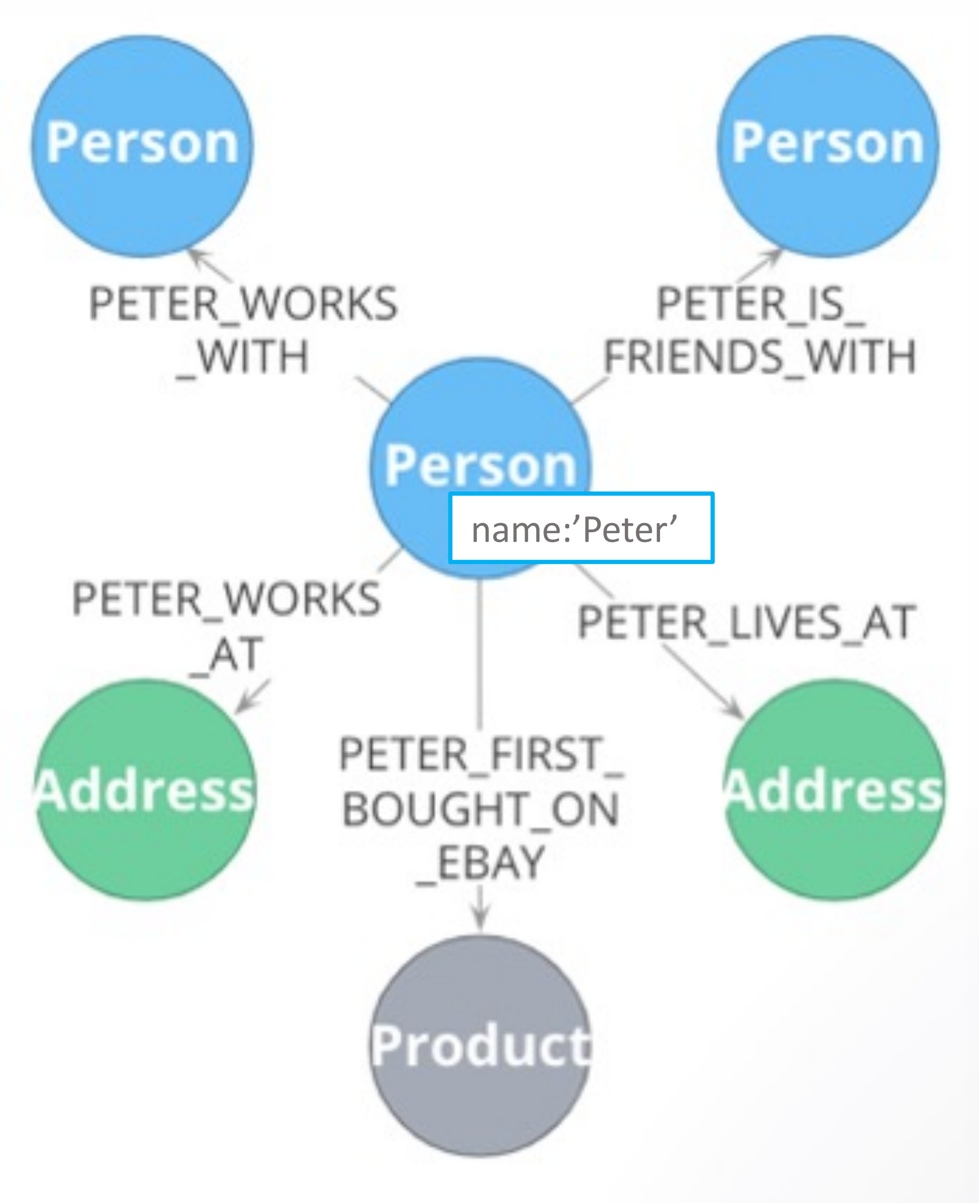
- The type of the relationship is too generic and thus useless
- Different kinds (living at, being friends, …) of relationships are modelled in the same way
- This also means that the relationship relates heterogeneous pairs of nodes, this is not per se a problem, if the link represented by it is semantically the same

is a single type for relationships a problem?



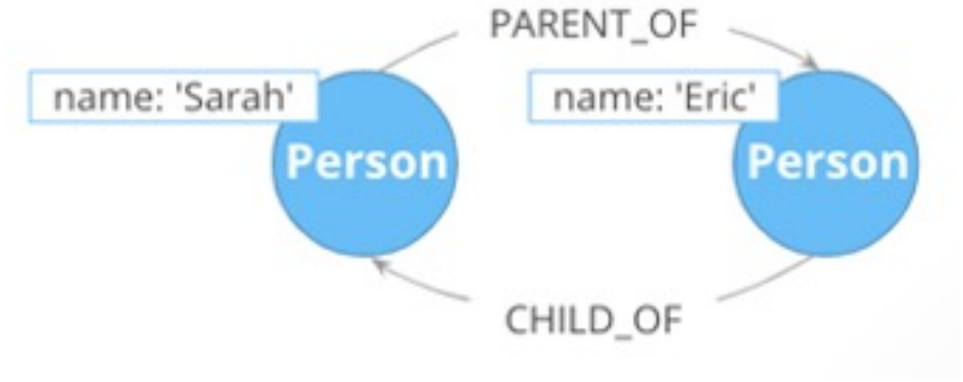
- this conceptual schema results in a graph with a single type for relationships, but the link represented by all of them is semantically the same (*liking*)

which problem (if any) do you see with this graph?



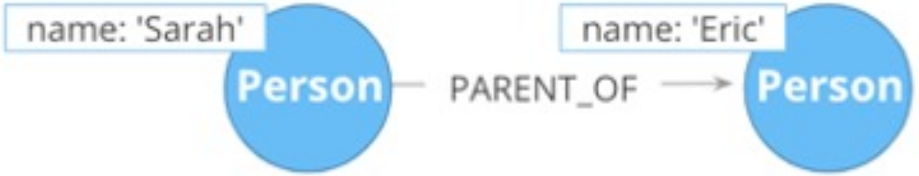
- The type of the relationship is too specific and refers to the value of a property of a connected node («peter»)
- This makes traversal over links of a semantically identical kind referring to another person (e.g., PAUL_WORKS_WITH) difficult
- Peter-centric modelling of relationships

which problem (if any) do you see with this graph?

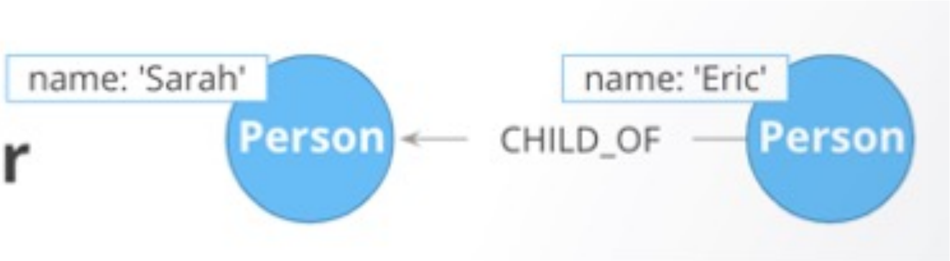


- The two relationships provide the same information
- This redundancy has a cost and provides no advantage
- The parent_of/child_of relations are one the inverse of the other

- One of these alternative graphs is better

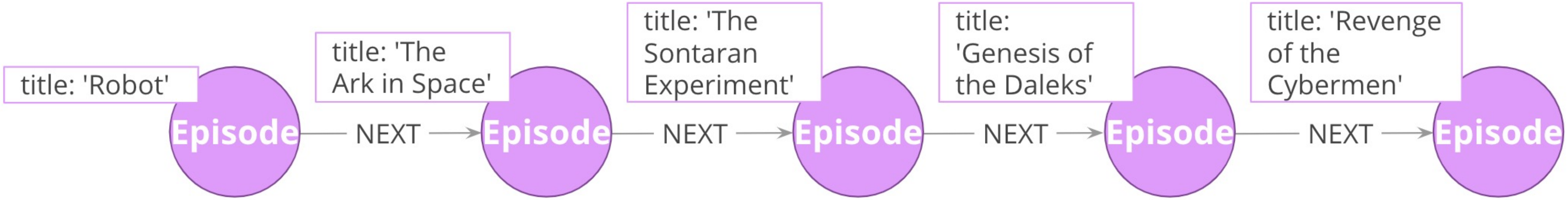


or

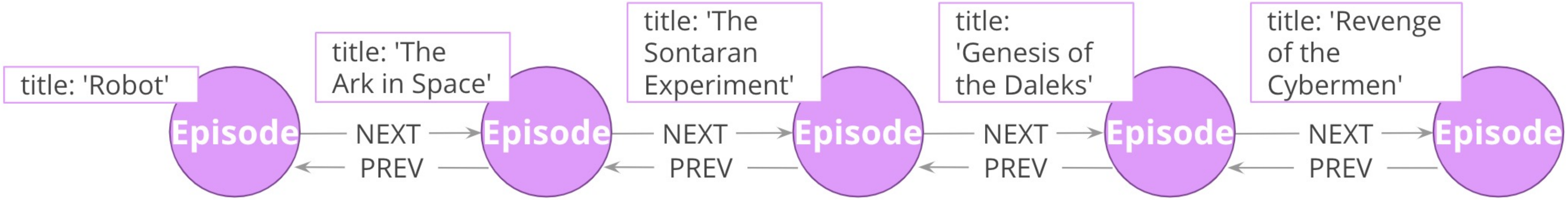


inverse relationships - a similar case

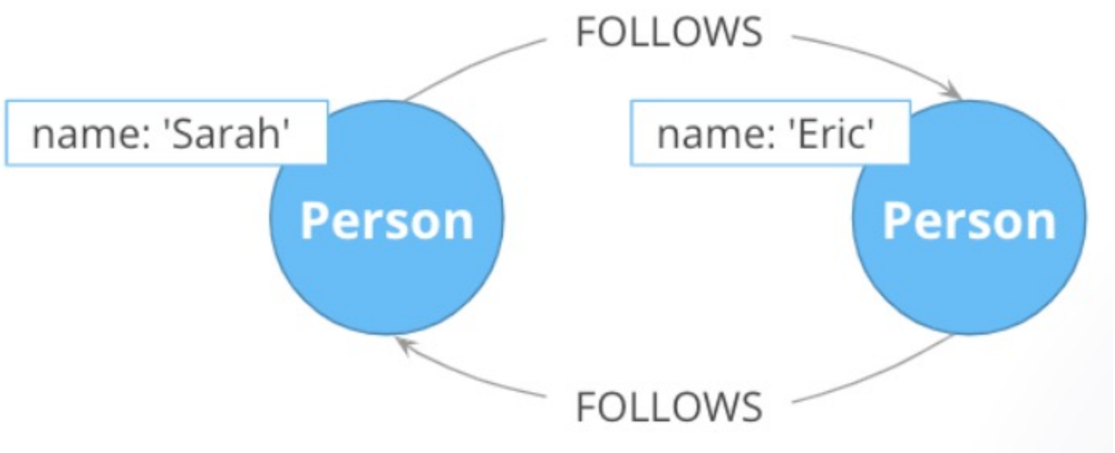
Episodes of the Dr.Who series:


title: 'Robot'
title: 'The Ark in Space'
title: 'The Sontaran Experiment'
title: 'Genesis of the Daleks'
title: 'Revenge of the Cybermen'
Episode
NEXT
Episode
NEXT
Episode
NEXT
Episode
NEXT
Episode

Do NOT do this (doubly-linked list):

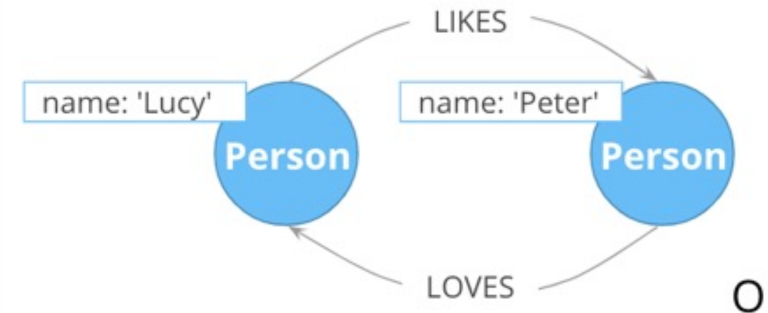

title: 'Robot'
title: 'The Ark in Space'
title: 'The Sontaran Experiment'
title: 'Genesis of the Daleks'
title: 'Revenge of the Cybermen'
Episode
NEXT
PREV
Episode
NEXT
PREV
Episode
NEXT
PREV
Episode
NEXT
PREV
Episode

which problem (if any) do you see with this graph?

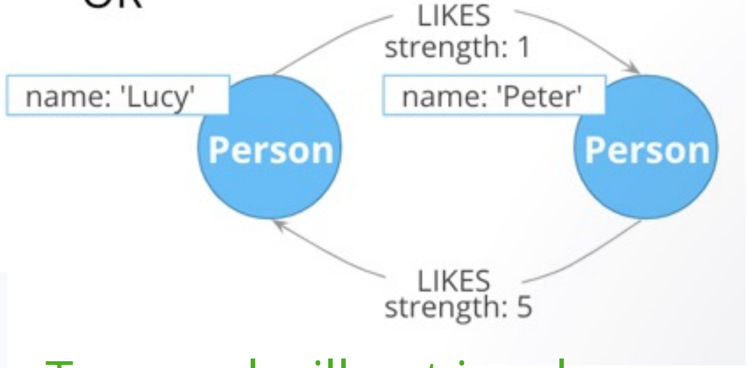


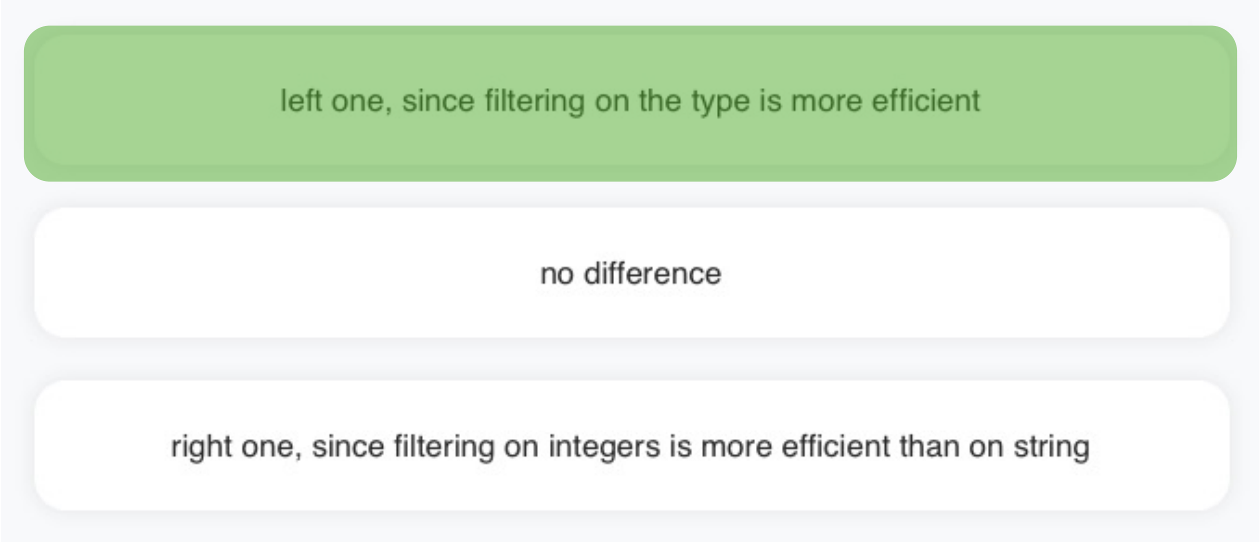
- The graph refers to a case like the ones in twitter or instagram or tiktok in which the fact that Sarah follows Eric guarantees that Eric is followed by Sarah but Eric does not necessarily follow Sarah back
- The relationship is not symmetric (unlike for intance the facebook friendship relationship)
- Thus, both the edges are needed here, since they represent different information

which of the two alternatives is better if we want to find all the strong relationships and discard the weaker ones?



OR



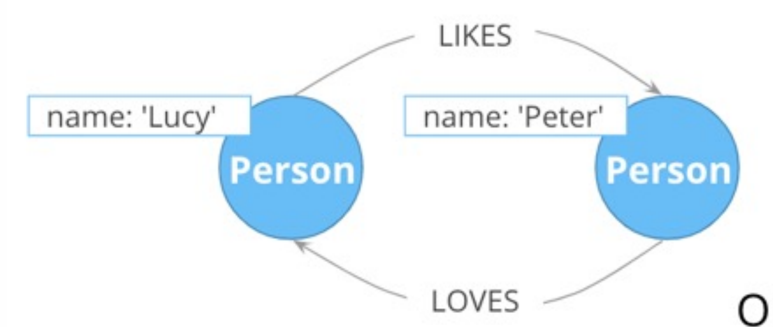


[✓ left one, since filtering on the type is more efficient]

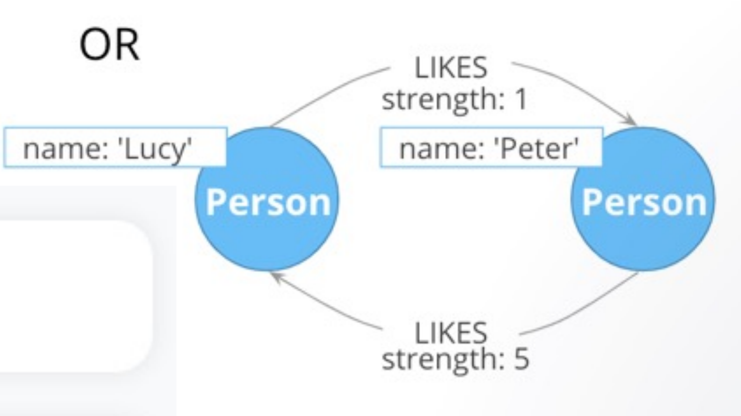
no difference

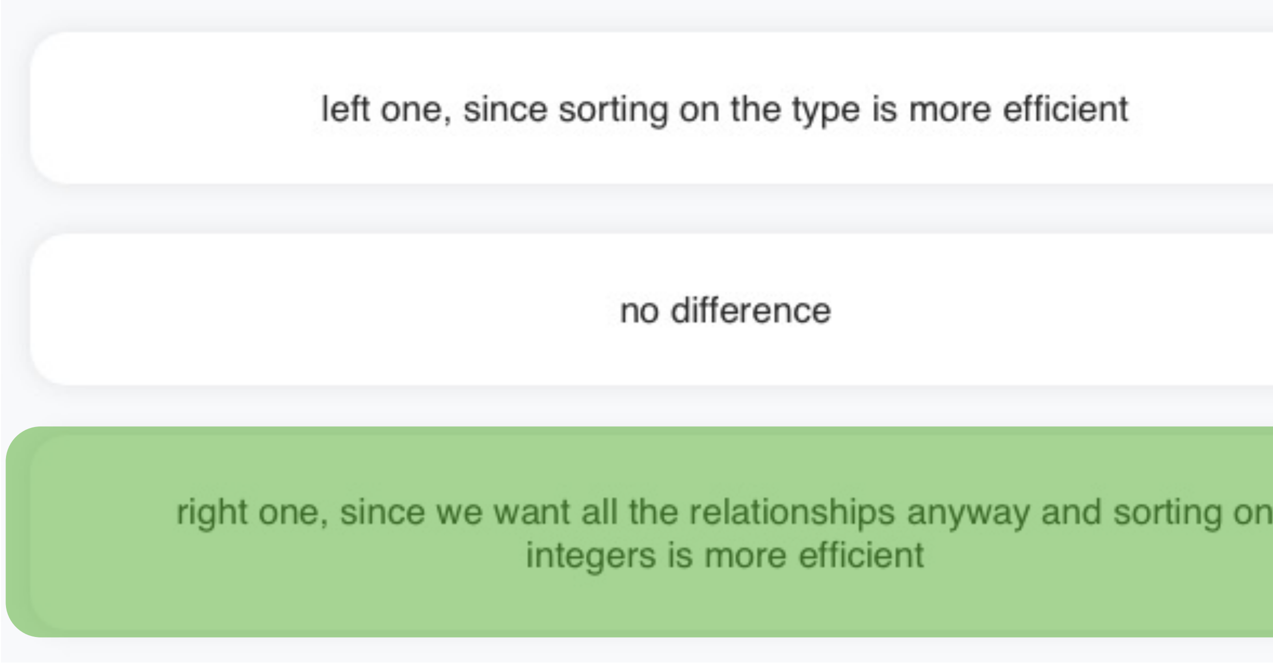
right one, since filtering on integers is more efficient than on string

Traversal will not involve any gather-and-inspect which of the two alternatives is better if we want to find all the relationshinps (both weak and strong), ranked by strength (stronger first)?



OR





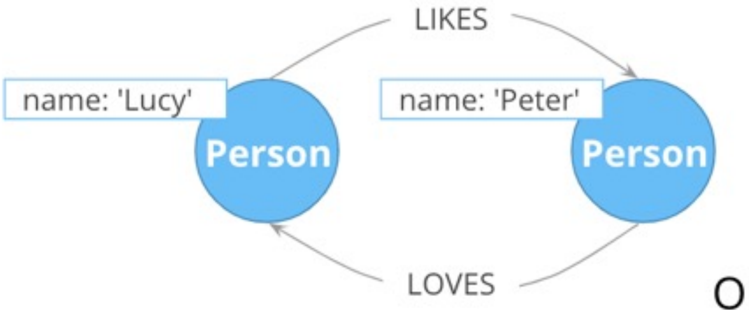
left one, since sorting on the type is more efficient

no difference

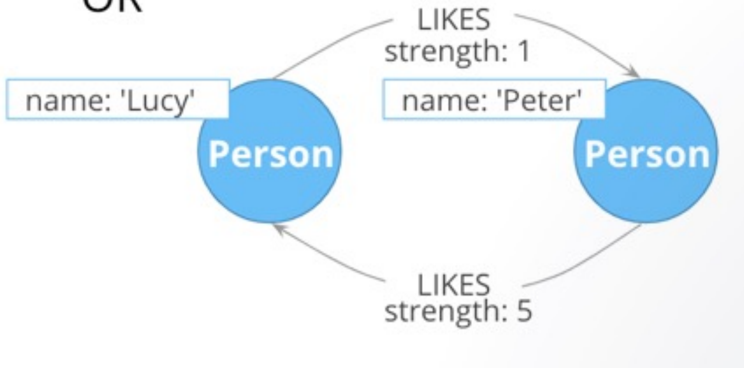
right one, since we want all the relationships anyway and sorting on integers is more efficient

it is also simpler to express in Cypher, and there will not be any gather-and-inspect discards because we want everything anyway

Any other difference among the two?



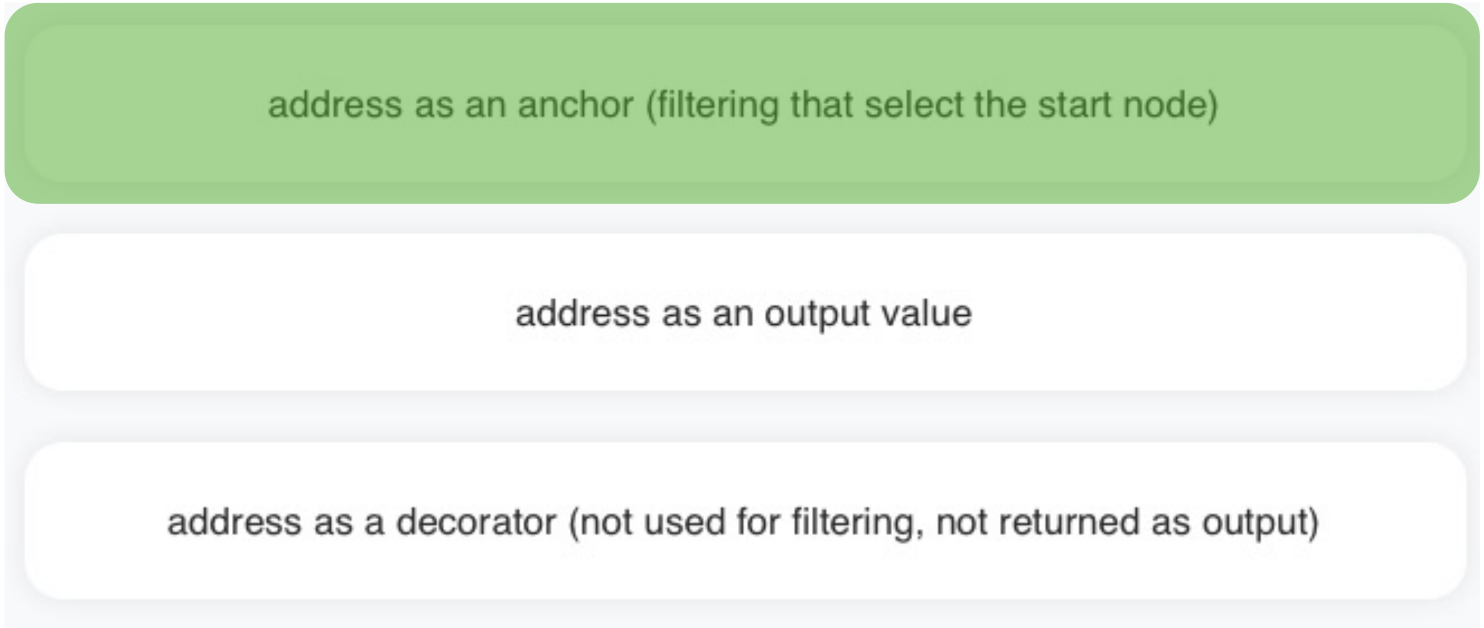
OR



- The right one is more flexible if we consider all the values for strength in the range 1..5 rather than just 1 and 5
- It allows to represent a wider set of nuances

for which of the following cases do you see an issue with the complex/multivalued address property?

firstName: 'Patrick'
lastName: ‘Scott’
age: 34
homeAddress: [‘Flat 3B’, ‘83 Landor St.’, ‘Axebridge’, ‘DF3 OAS’]
workAddress: [‘Acme Ltd.’, ‘12 Crick St.’, ‘Balton’, ‘DG4 9CD’]



[✓ address as an anchor (filtering that select the start node)]

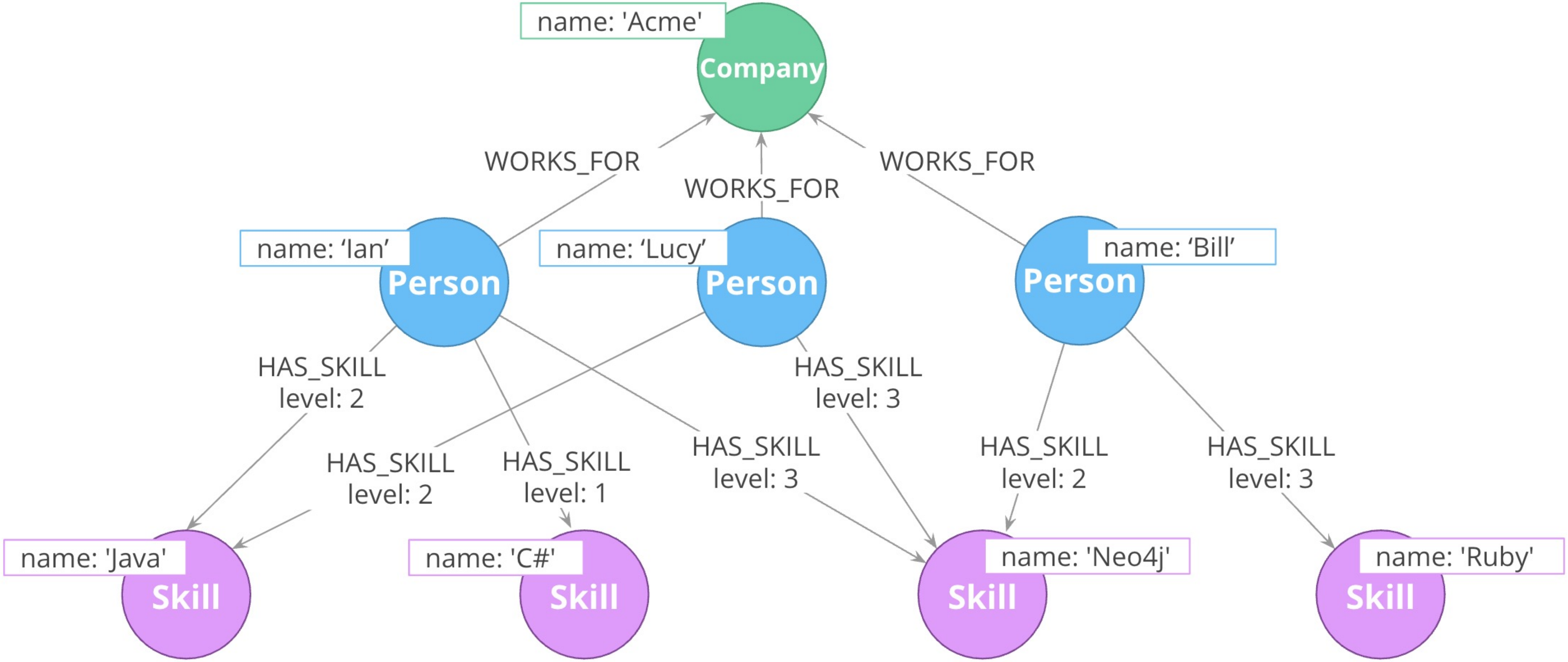
address as an output value

address as a decorator (not used for filtering, not returned as output)

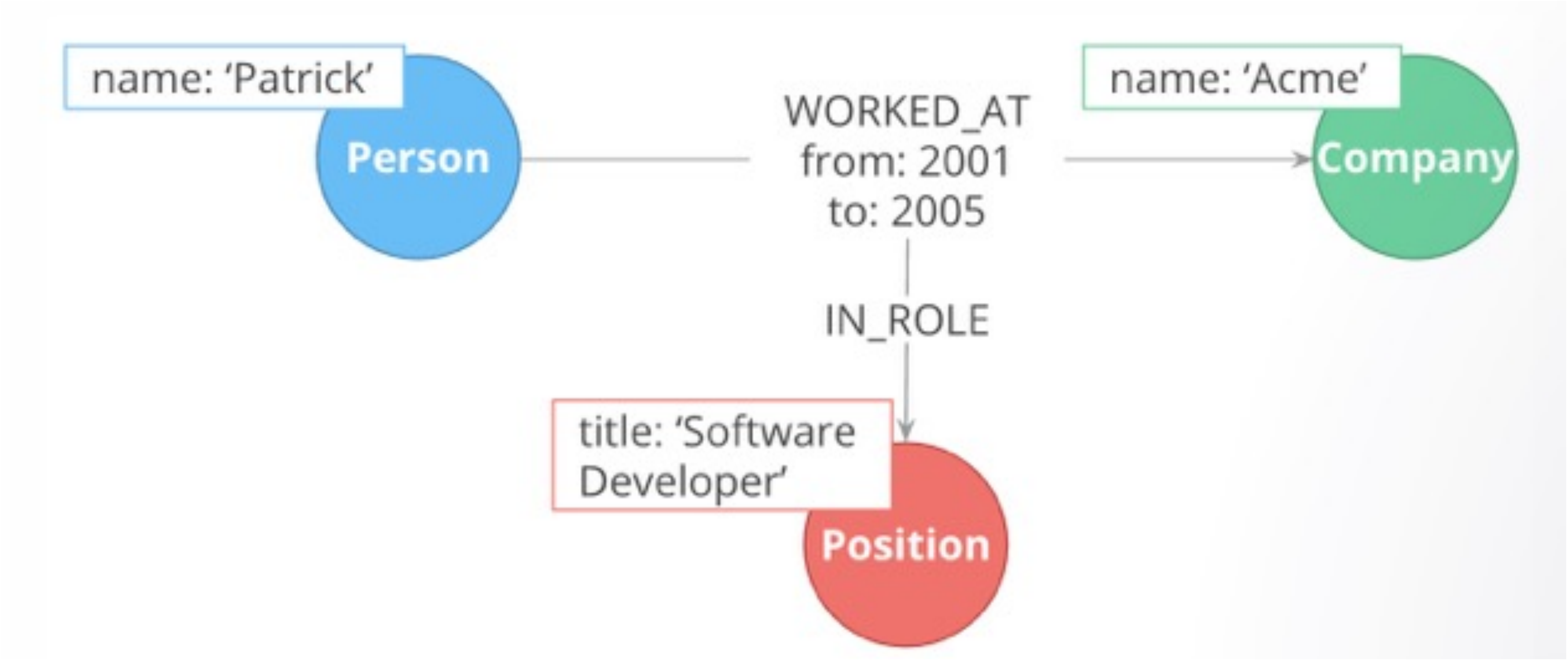
properties used for anchoring should be as simple as possible

Graph Traversals – Access Costs

1. Anchor node label, indexed anchor node properties
2. Relationship types
3. Non-indexed anchor node properties
4. Downstream node labels
5. Relationship properties, downstream node properties

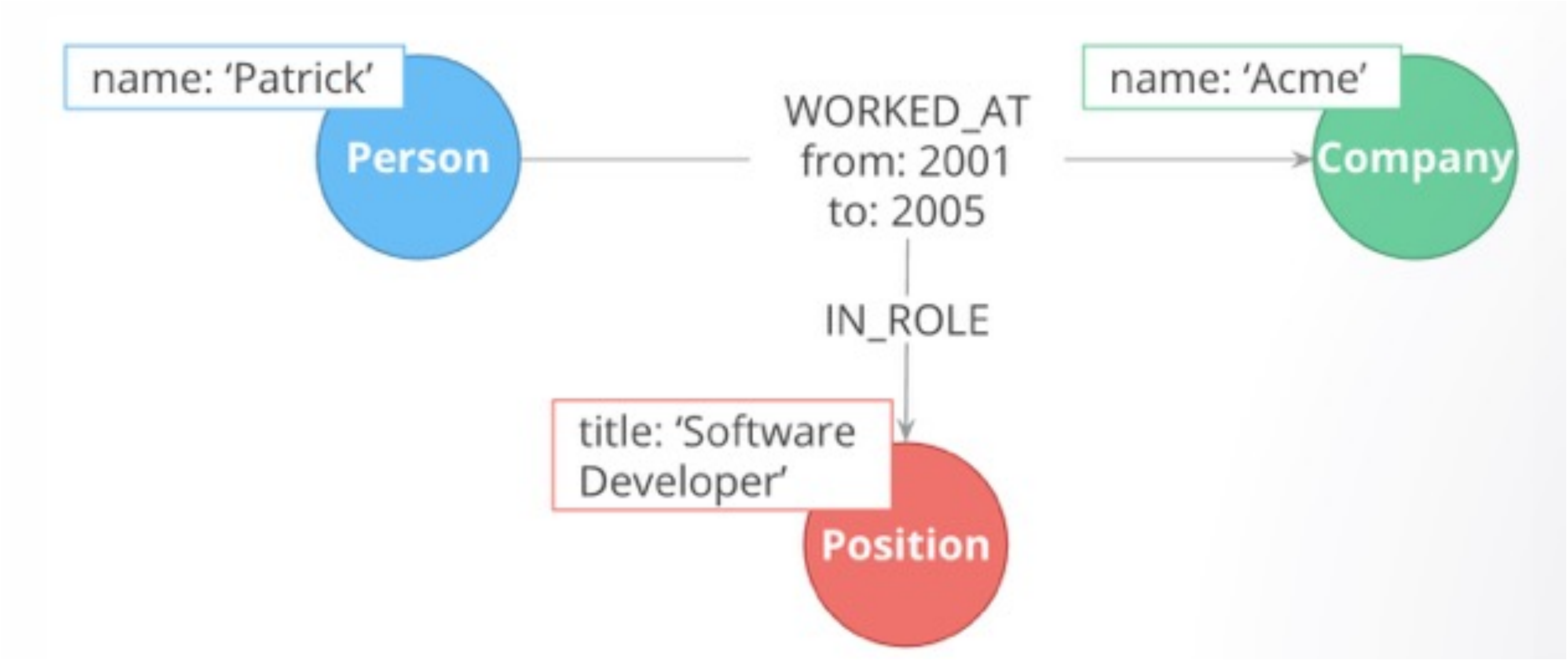


which problem (if any) do you see here?



This is not a graph!
Relationships cannot connect three nodes (an hyperedge would be needed, not supported by the property graph model)

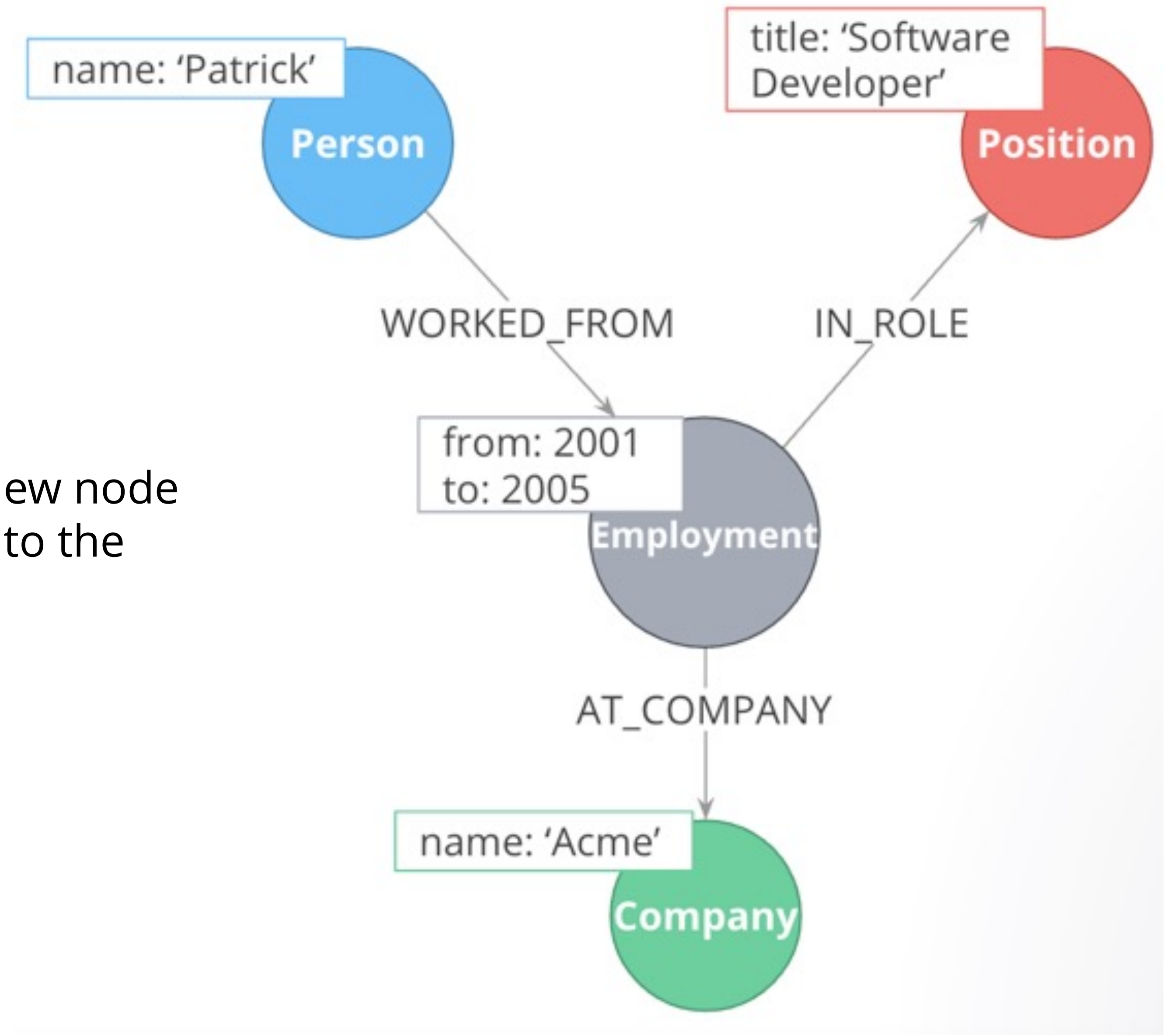
how can we represent it? (1)



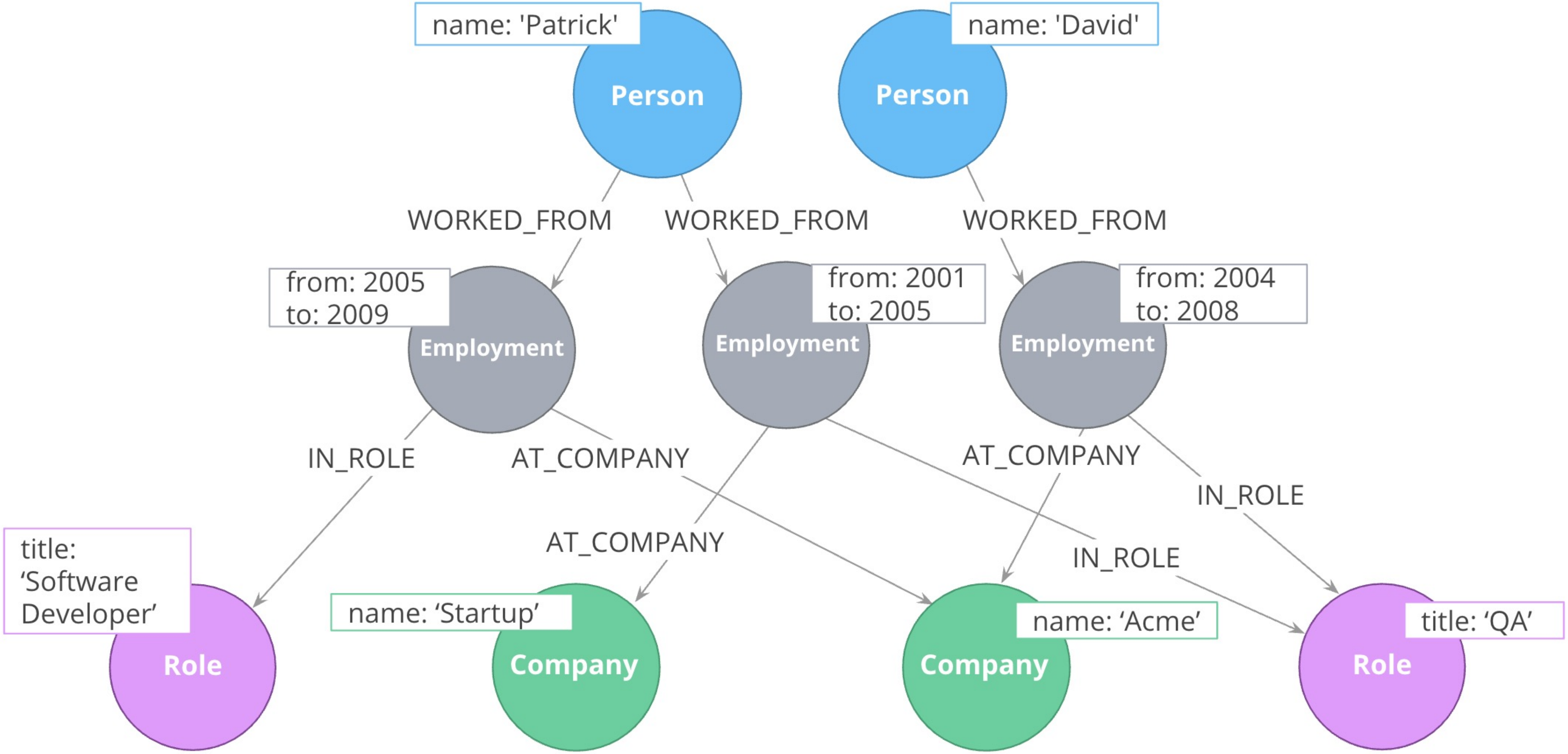
We could simply give up the position node and model roles as properties of the relationship

This makes queries like «find the employees that worked as Software Developer» or «find the companies with Software Engineering positions» less efficient

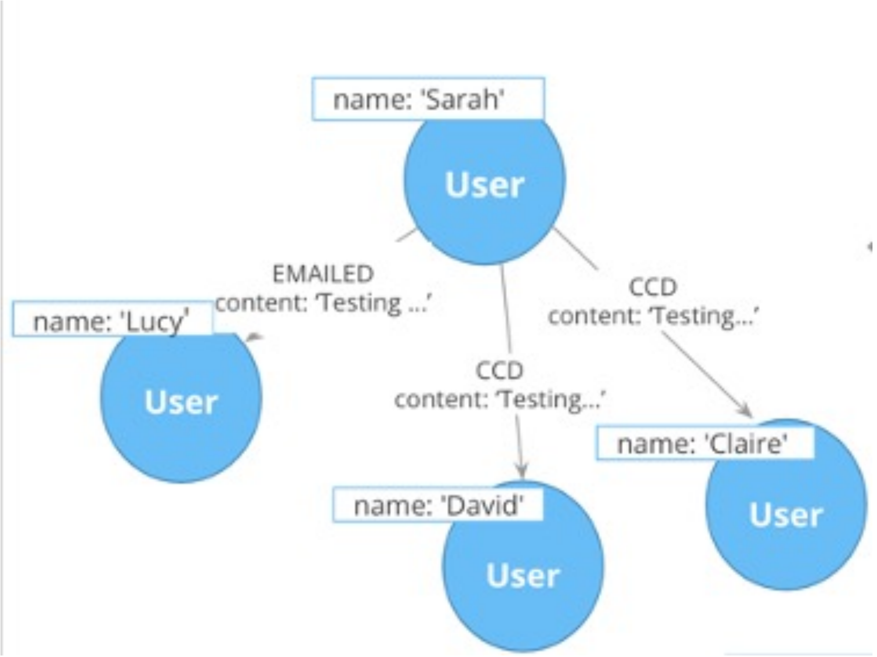
how can we represent it [if we want to keep position nodes and capture a ternary relationship]?



Introducing a new node corresponding to the relationship

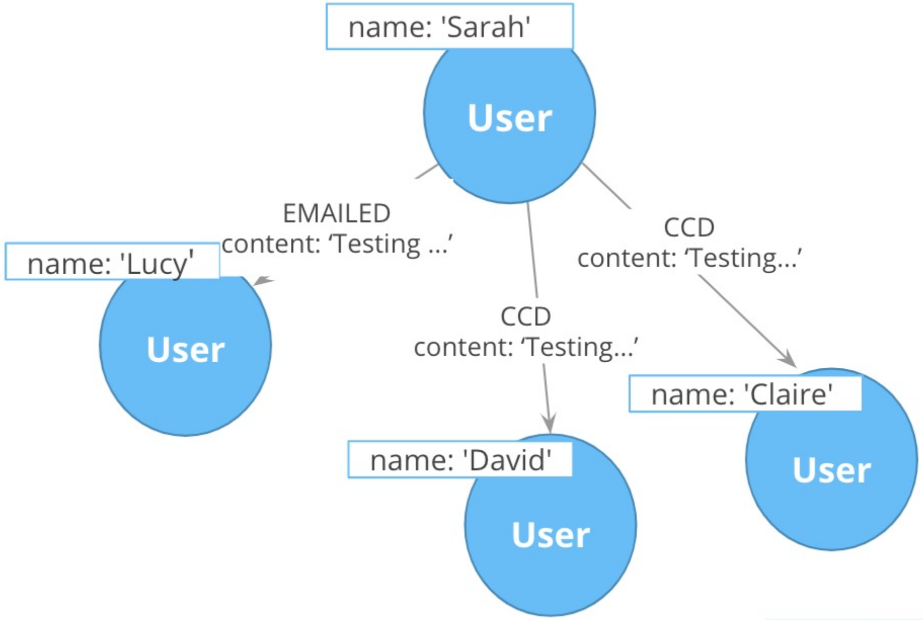

name: 'Patrick'
Person
name: 'David'
Person
WORKED_FROM
WORKED_FROM
WORKED_FROM
from: 2005
to: 2009
Employment
from: 2001
to: 2005
Employment
from: 2004
to: 2008
Employment
IN_ROLE
AT_COMPANY
AT_COMPANY
AT_COMPANY
IN_ROLE
IN_ROLE
title:
'Software
Developer'
Role
name: 'Startup'
Company
name: 'Acme'
Company
title: 'QA'
Role

which problem (if any) do you see with this graph?



- Duplicate email text
- We cannot distinguish persons that are ccopied in the same email without looking at the content property of relationships
- The content of the email itself could not be enough to determine whether two emails are the same
- User nodes likely become dense since they will be connected to all the recipients and ccopied users of all the emails they send

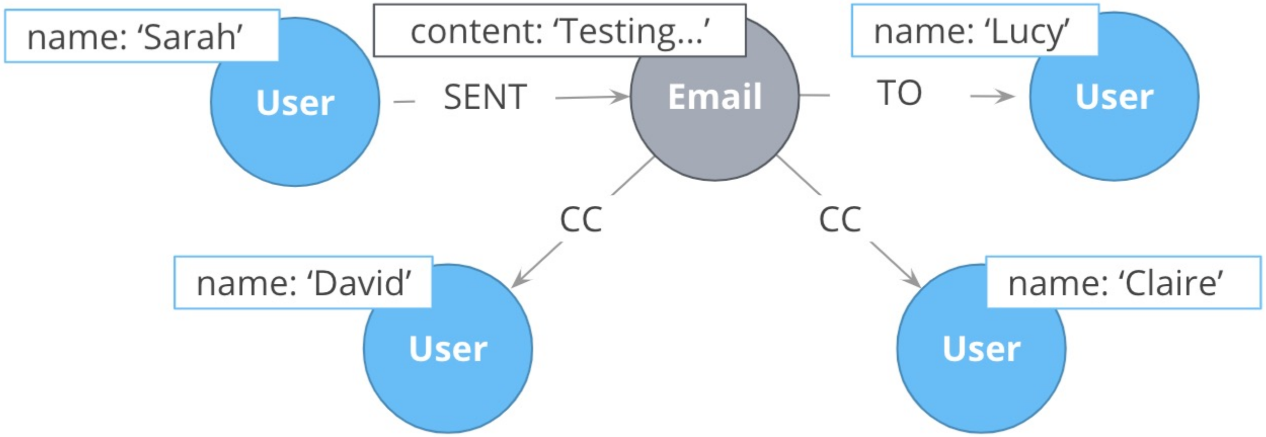
a better modelling



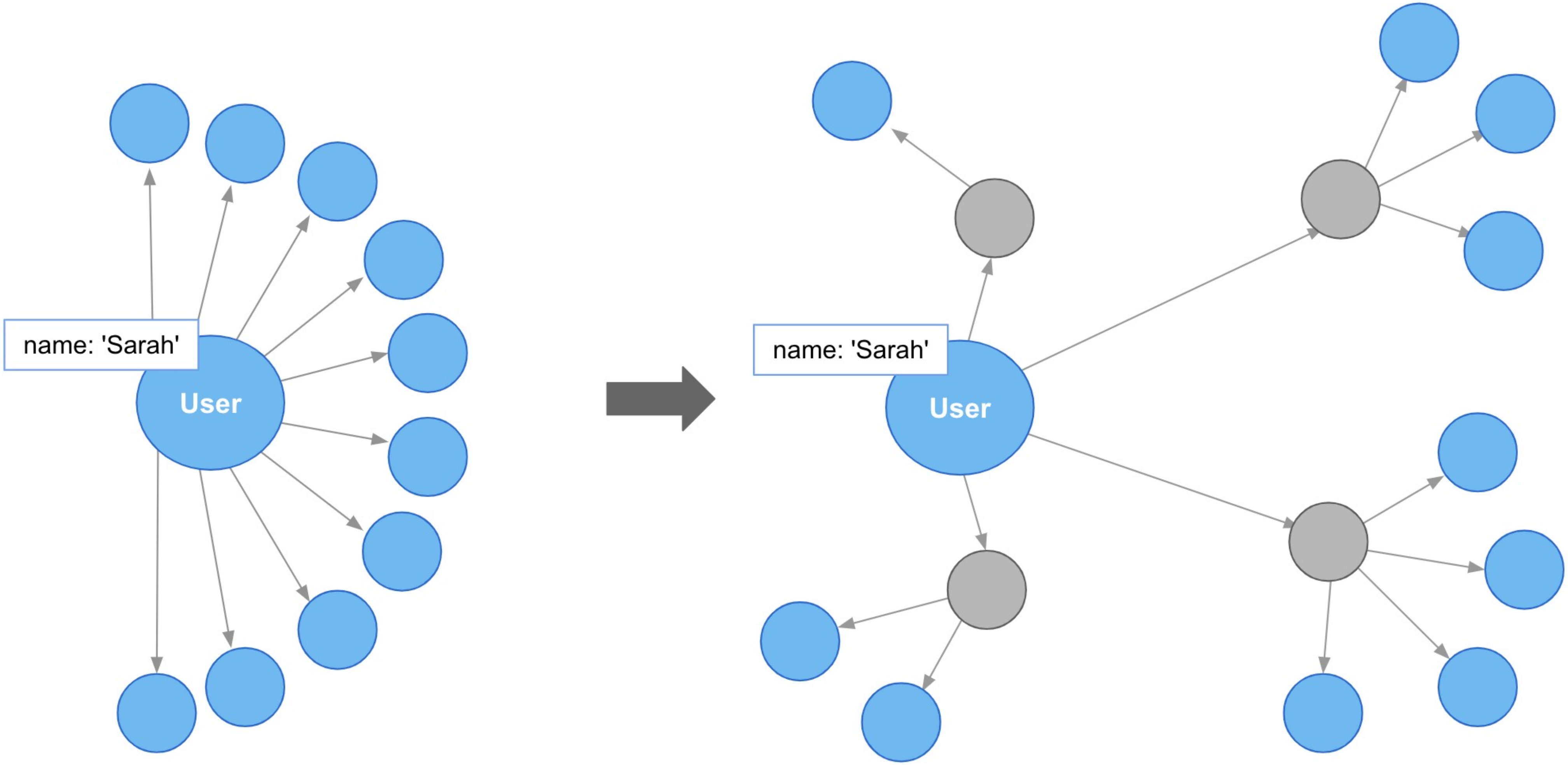
By introducing the email node we can fix the issues

Now it is much simpler and more efficient to determine

All recipients of an email about «ADM project assignment»



a better modelling – node density



With the email node, each user is connected to the emails she wrote