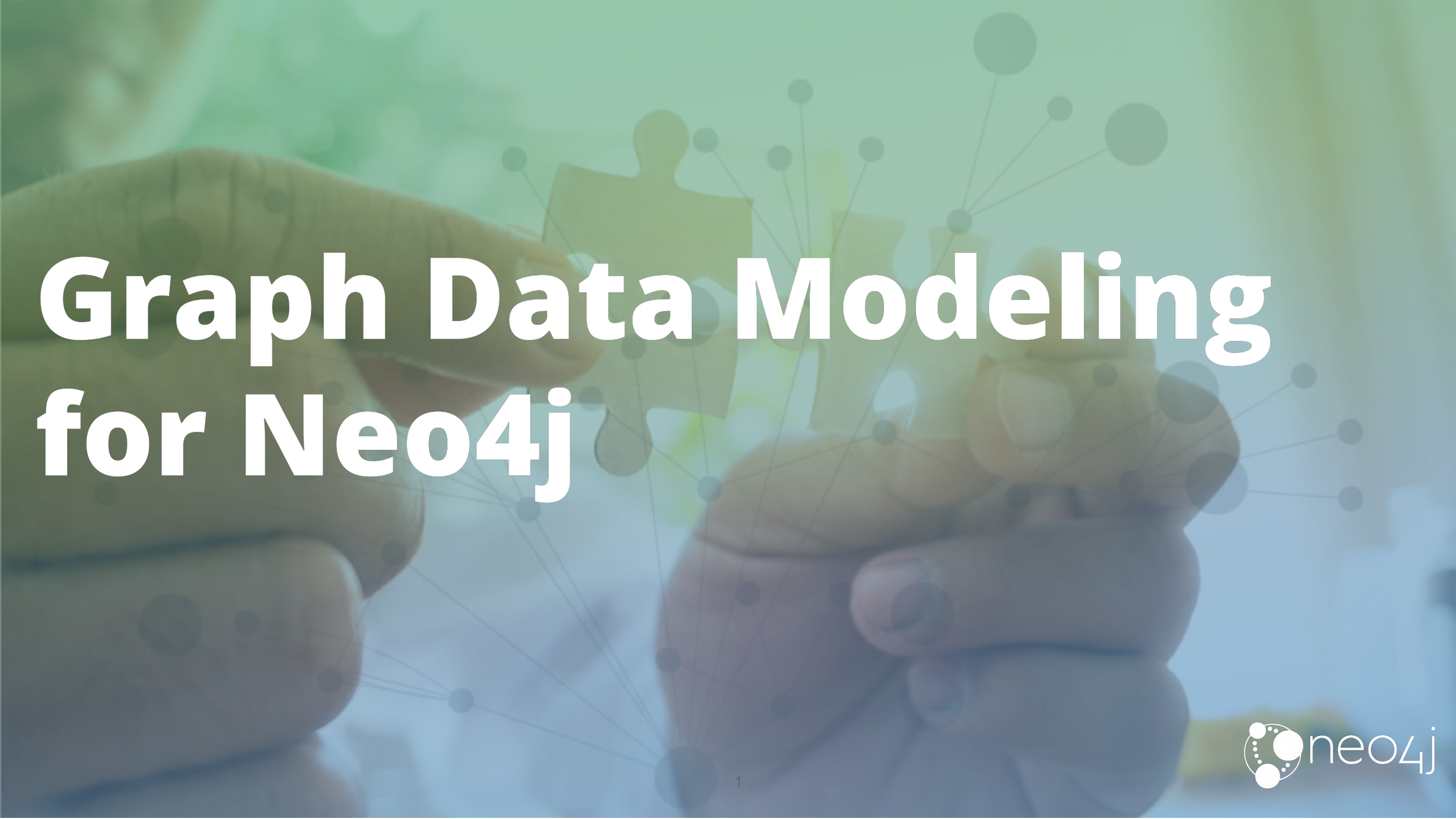
Graph Data Modeling for Neo4j



About this course

This course introduces you to how to develop a graph data model for an application using Neo4j best practices.

In this course, you complete hands-on exercises to gain experience with developing graph data models for use with Neo4j.

To complete the hands-on exercises for this course, you need a Web browser and Internet access.

This Course is published by Neo4j per this License for Use.

Lesson Overview

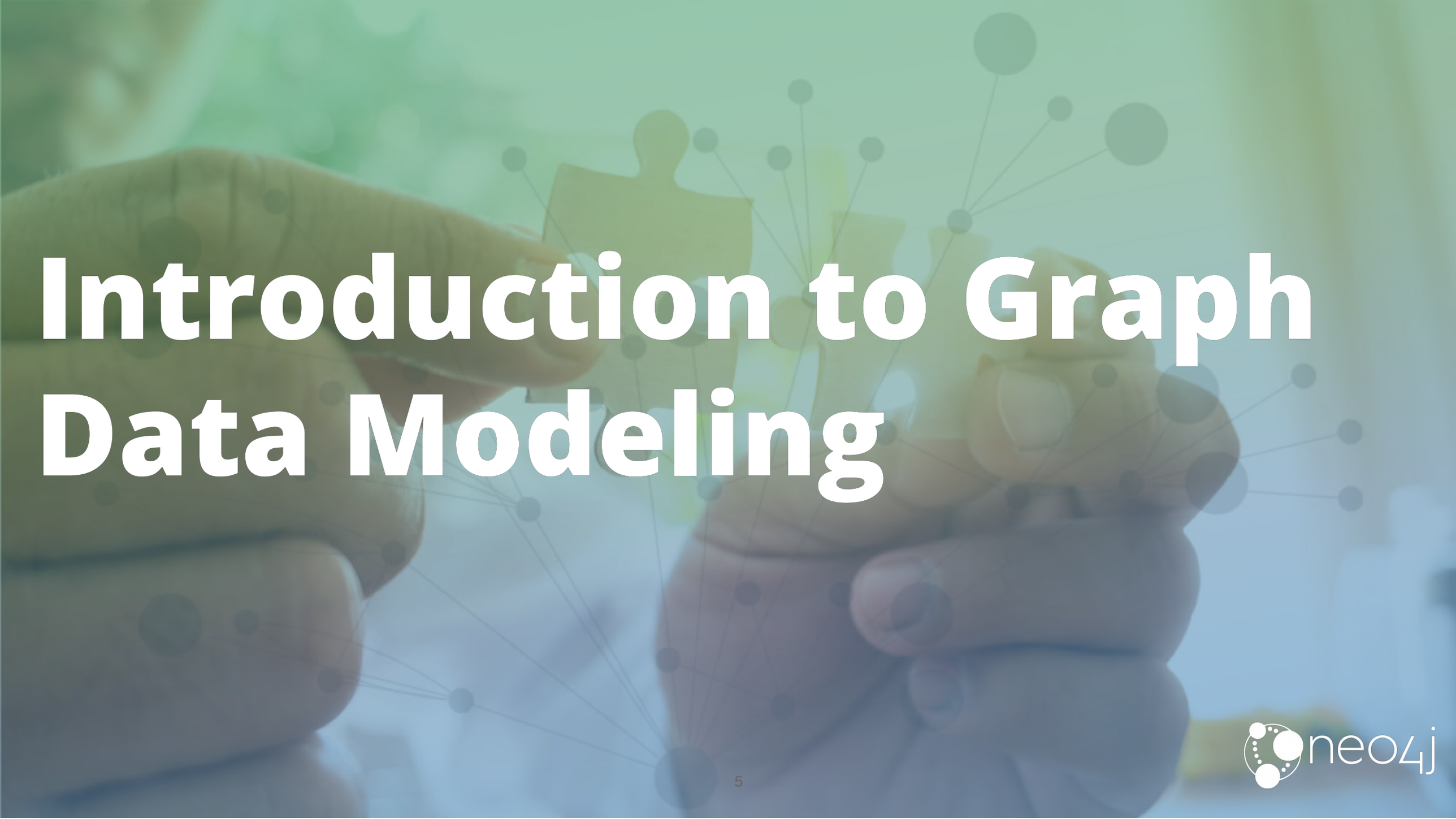
Here are the lessons of this course:

1. Introduction to Graph Data Modeling
2. Designing the Initial Graph Data Model
3. Graph Data Modeling Core Principles
4. Common Graph Structures
5. Refactoring and Evolving a Graph Data Model

Resources

Here are some resources you may use as you go through this course:

- Neo4j Cypher Manual
- Neo4j Developer Resources



Introduction to Graph Data Modeling

neo4j

About this module

At the end of this module, you should be able to:

- Describe what graph data modeling is.
- Describe how Neo4j supports graph data modeling.
- Describe the tools you can use for graph data modeling.
- Describe the workflow for graph data modeling.

What is graph data modeling?

Graph data modeling is a collaborative effort where the application domain is analyzed by stakeholders and developers to come up with the optimal model for use with Neo4j. Stakeholders must understand the domain and be prepared to ask detailed questions about how the business-at-hand operates.

Stakeholders are:

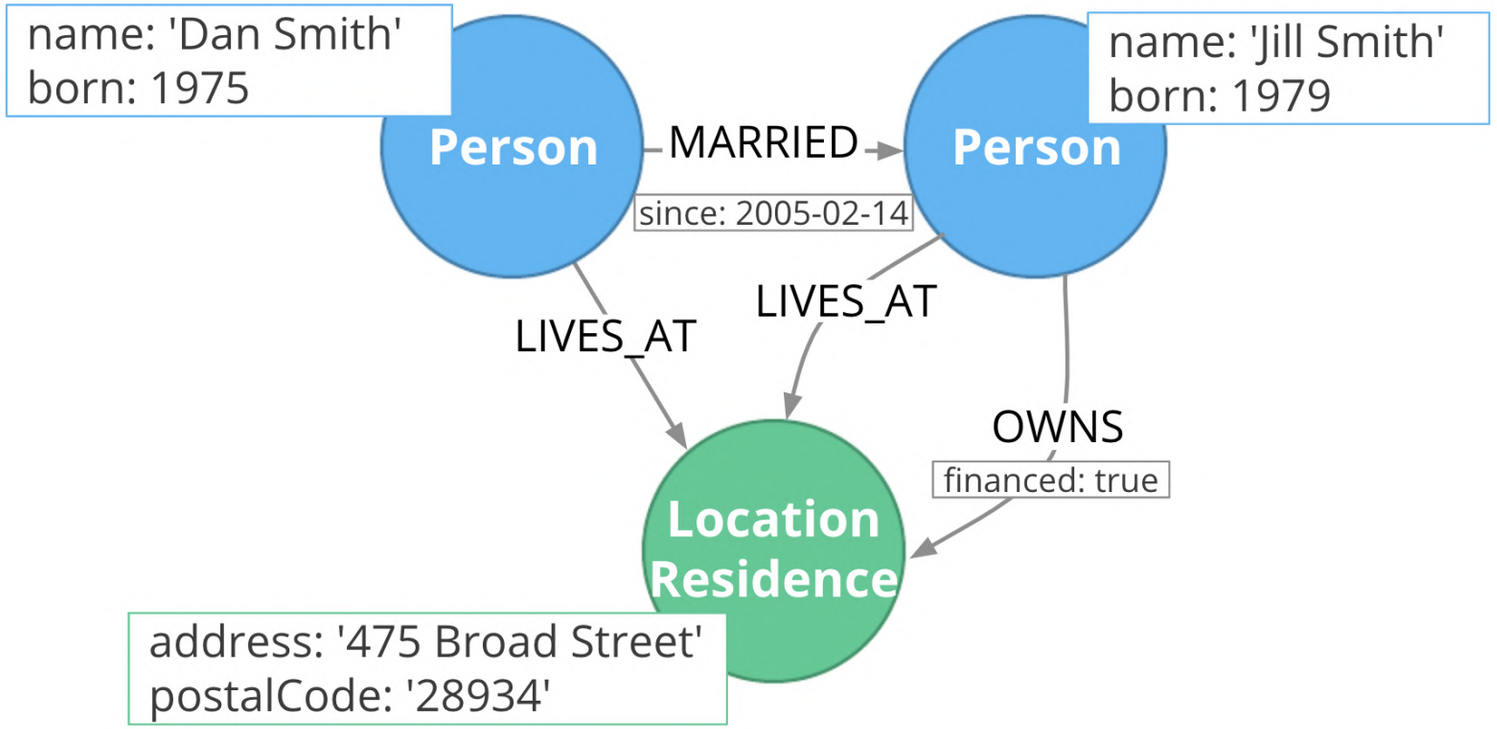
- Business analysts
- Architects
- Managers
- Project leaders

How does Neo4j support graph data modeling?

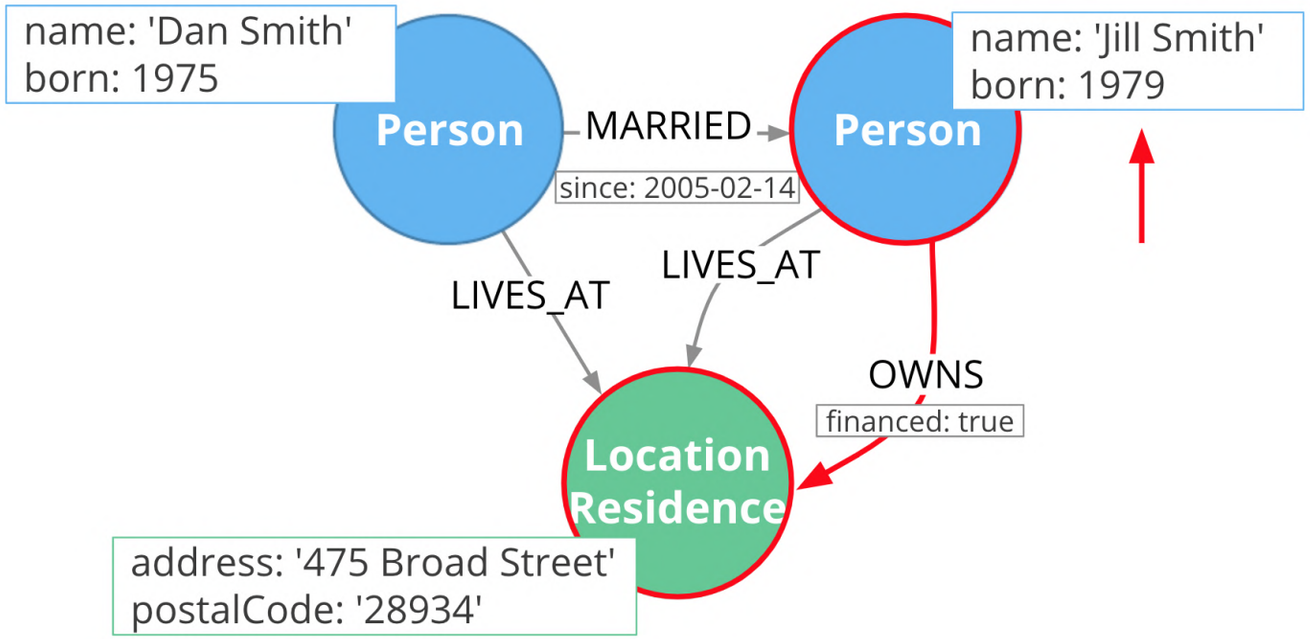
- Neo4j is a full-featured graph database that allows you to create **property graphs**.
- Applications retrieve data for business use cases by **traversing the graph**.

Neo4j Property Graph Model

- Nodes (Entities)
- Relationships
- Properties
- Labels

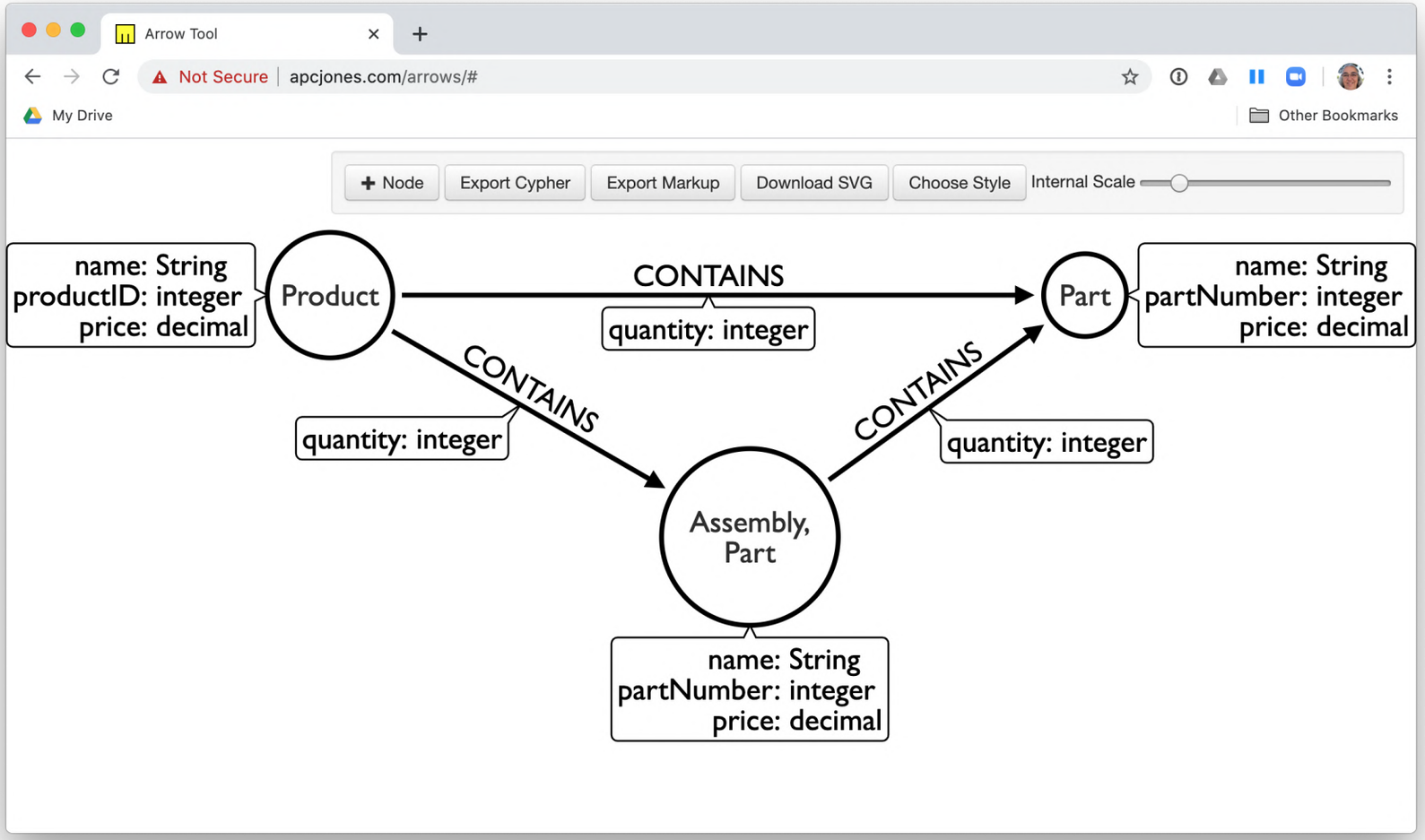


Traversal in the graph




```
MATCH (r:Residence)<-[:OWNS]-(p:Person)
WHERE r.address = '475 Broad Street'
RETURN p
```

Tools for graph data modeling



Guided Exercise: Using the Arrow tool

https://youtu.be/NB184T-S46w

Workflow for graph data modeling

Step	Description	Stakeholders	Developers
1.	Build the initial graph data model.	✔	✔
2.	Create and profile Cypher queries to support the model.		✔
3.	Create data in the database to support the model.		✔
4.	Identify additional questions for the application.	✔	✔
5.	Modify the graph data model to support new questions.		✔
6.	Refactor the database to support the revised graph data model.		✔
7.	Create and profile Cypher queries to support the revised model.		✔
	Repeat Steps 4-7.	✔	✔

Check your understanding

neo4j

Question 1

What elements of a Neo4j graph are used to categorize entities?

Select the correct answer.

- [ ] Relationship
- [ ] Property
- [ ] Node
- [ ] Label

Question 1

What elements of a Neo4j graph are used to categorize entities?

Select the correct answer.

- [ ] Relationship
- [ ] Property
- [ ] Node
- [x] **Label**

Question 2

Which statements below are true about Neo4j graph traversal?

Select the correct answers.

- [ ] Traversal during a query starts at a set of anchor nodes.
- [ ] Traversal during a query ends at an anchor node.
- [ ] A relationship is only traversed once during a query.
- [ ] A relationship can be traversed multiple times during a query.

Question 2

Which statements below are true about Neo4j graph traversal?

Select the correct answers.

- ✔ **Traversal during a query starts at a set of anchor nodes.**
- ☐ Traversal during a query ends at an anchor node.
- ✔ **A relationship is only traversed once during a query.**
- ☐ A relationship can be traversed multiple times during a query.

Question 3

What modeling tool is designed specifically for use with Neo4j?

Select the correct answer.

- [ ] Visio
- [ ] Sketch
- [ ] Arrow Tool
- [ ] Neo4j Modeler

Question 3

What modeling tool is designed specifically for use with Neo4j?

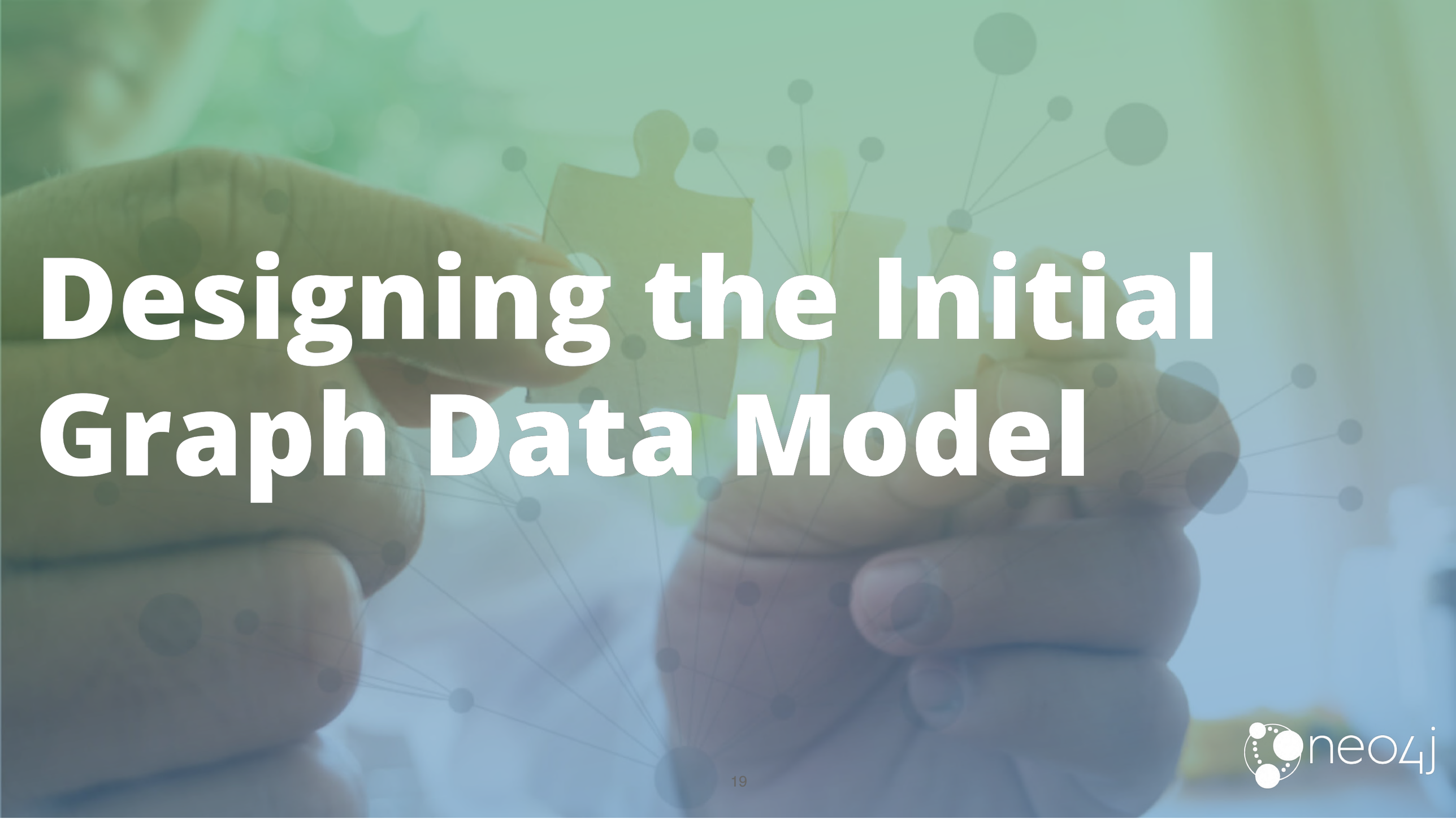
Select the correct answer.

- [ ] Visio
- [ ] Sketch
- [x] **Arrow Tool**
- [ ] Neo4j Modeler

Summary

You should now be able to:

- Describe what graph data modeling is.
- Describe how Neo4j supports graph data modeling.
- Describe the tools you can use for graph data modeling.
- Describe the workflow for graph data modeling.



Designing the Initial Graph Data Model

About this module

At the end of this module, you should be able to:

- Describe the domain for a model.
- Define the questions for the domain.
- Identify entities from the questions for the domain.
- Use the Arrow Tool to model the domain.
- Identify the connections between entities.
- Describe how to test the initial model.

Designing the initial data model

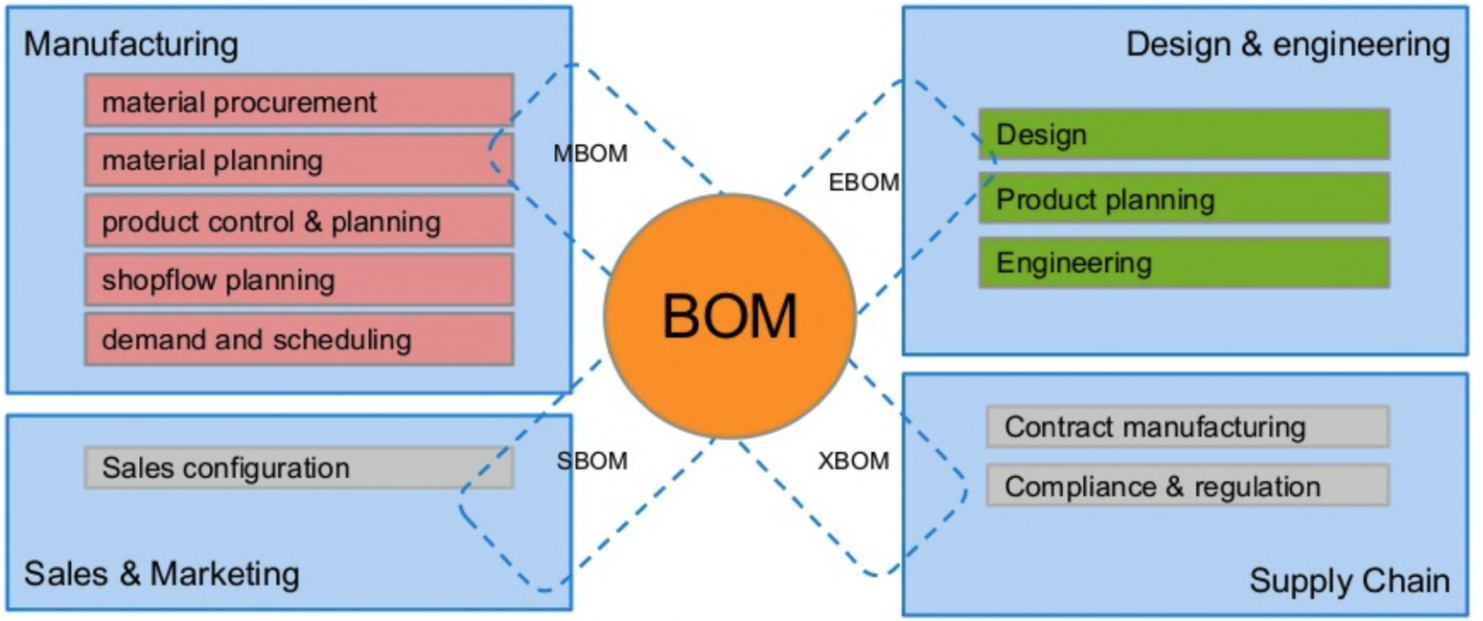
1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. Identify entities.
5. Identify connections between entities.
6. Test the questions against the model.
7. Test scalability.

Step 1: Understanding the domain

- Describe the application in detail.
- Identify the stakeholders and developers of the application.
- Identify the users of the application (people, systems).
- Enumerate the use cases that are agreed upon by all stakeholders where users are part of the use case.

Note No knowledge of the underlying implementation is required!

Example domain: Bill of Materials



BOM connects all engineering and non-engineering processes

Example BOM use cases

- System produces list of parts to make a product.
- System produces list of products that can be made with available parts.
- System produces list of parts that are made with other parts.
- User picks parts to make a product.
- System creates a price for a product based upon the part prices.
- System creates list of parts that need to be ordered.

Note A product or part can be made of multiple parts of the same type. Some parts are made from other parts (sub-assembly).

Step 2: Create high-level sample data

1. Understand the domain.
2. **Create high-level sample data.**
3. Define specific questions for the application.
4. Identify entities.
5. Identify connections between entities.
6. Test the questions against the model.
7. Test scalability.

BOM high-level sample data

Products	Parts	Assemblies	Notes
Wood table 40"	Wood top 40"	Leg assembly	Has 4 legs
Deluxe wood table 40"	Glass top 40"	Leg assembly	Has 4 legs
Wood table 60"	Wood top 60"	Leg assembly	Has 6 legs, table brace
Deluxe wood table 60"	Glass top 60"	Leg assembly	Has 6 legs, table brace
	Leg		
	Leg foot		
	M20 bolt		
	M20 nut		
	Leg plate		Uses 2 bolts/nuts per leg
	Table brace		

Step 3: Define specific questions for the application

1. Understand the domain.
2. Create high-level sample data.
3. **Define specific questions for the application.**
4. Identify entities.
5. Identify connections between entities.
6. Test the questions against the model.
7. Test scalability.

Sample questions for the BOM

1. What parts are needed to make Wood table 40"?
2. Do we have enough parts to make 100 Deluxe wood table 60"?
3. What products require a table brace?
4. How much will the parts cost to make product Wood table 60"?

Step 4 : Identify entities

1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. **Identify entities.**
5. Identify connections between entities.
6. Test the questions against the model.
7. Test scalability.

Identify entities from the questions

Entities are the nouns in your application questions:

1. What ingredients are used in a recipe?
2. Who is married to this person?

- The generic nouns will often become labels in your model.
- Proper nouns will often become values for properties.
- Use domain knowledge to decide if entities need to be further grouped or differentiated.
- In Neo4j Enterprise Edition, there is no limit to the number entities (nodes) in the graph. (Community Edition has a limit of 34B)

Define properties

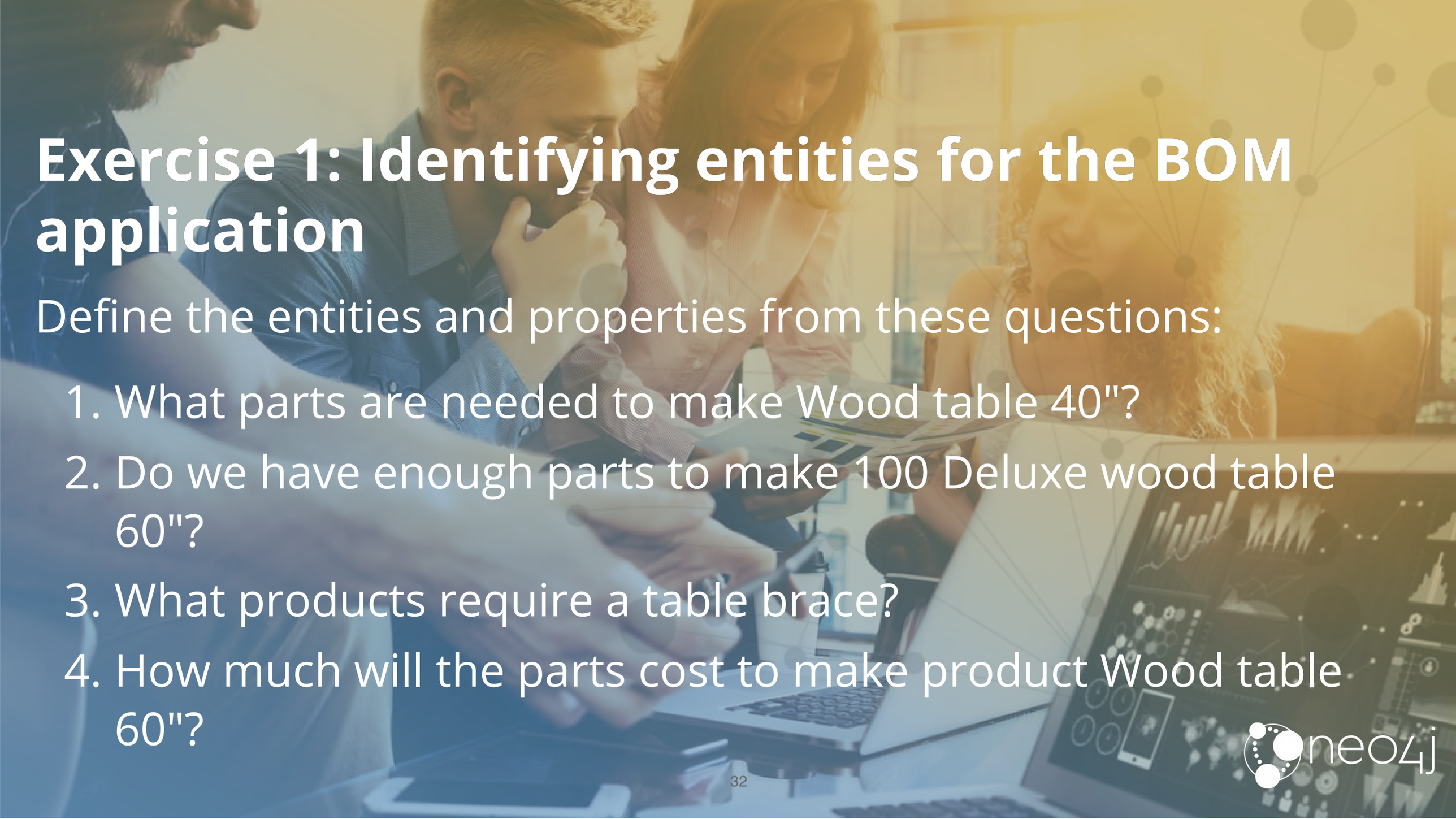
Properties serve one of two purposes:

1. Unique identification.
2. Answering one of the application questions.

Otherwise, they are merely "decoration".

Properties are used in a Cypher query for:

- Anchoring (where to begin the query).
- Traversing the graph (navigation).
- Returning data from the query.



Exercise 1: Identifying entities for the BOM application

Define the entities and properties from these questions:

1. What parts are needed to make Wood table 40"?
2. Do we have enough parts to make 100 Deluxe wood table 60"?
3. What products require a table brace?
4. How much will the parts cost to make product Wood table 60"?

Exercise 1 solution

Product

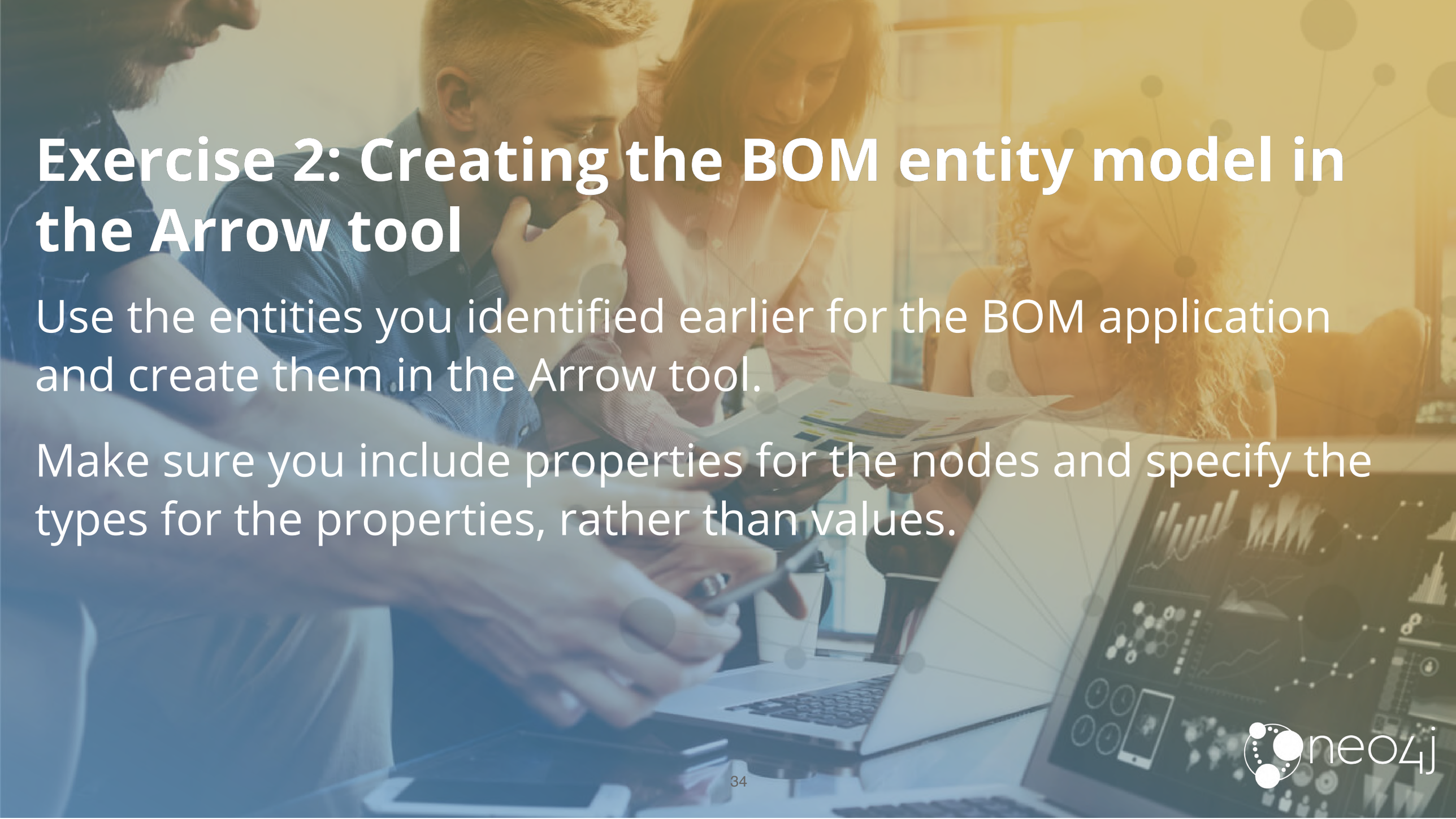
- name
- productId

Part

- name
- partNumber
- price

Part, Assembly

- name
- partNumber
- price



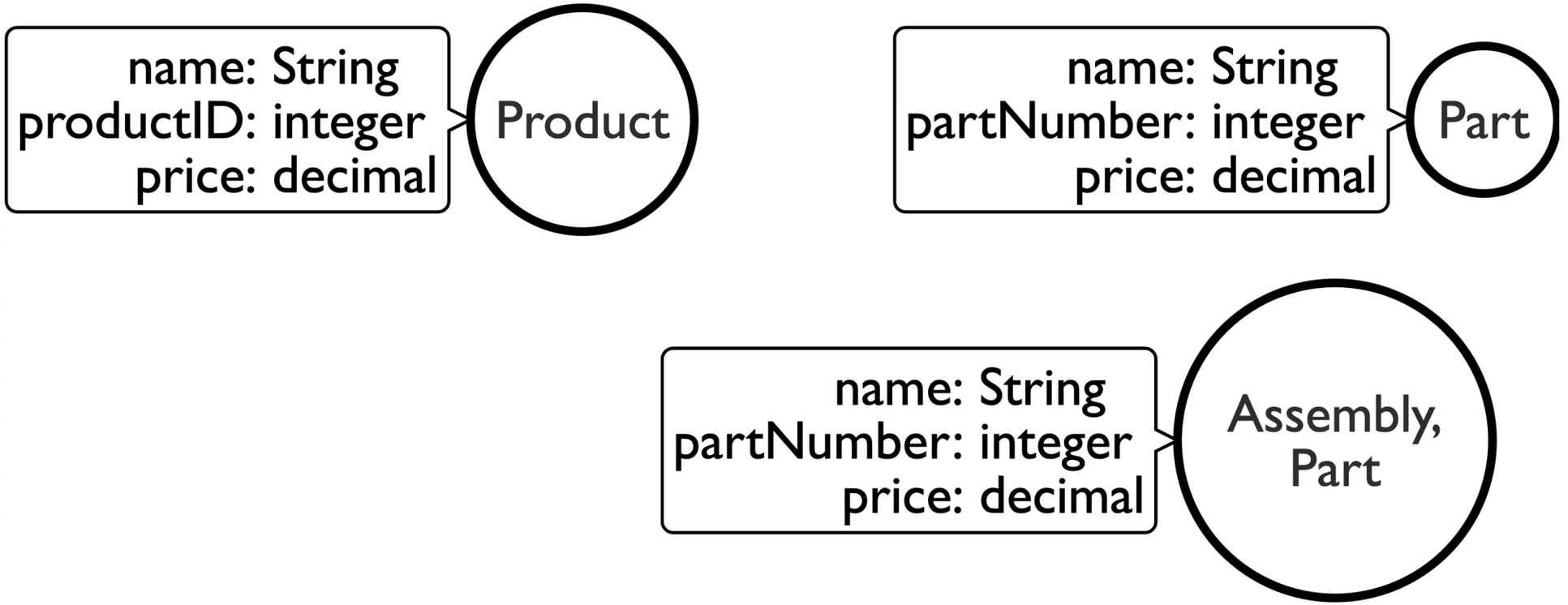
Exercise 2: Creating the BOM entity model in the Arrow tool

Use the entities you identified earlier for the BOM application and create them in the Arrow tool.

Make sure you include properties for the nodes and specify the types for the properties, rather than values.

Exercise 2 solution

Here is what it should look like in the UI of the Arrow tool:



Step 5: Identify connections between entities

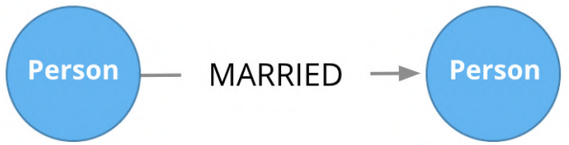
1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. Identify entities.
5. **Identify connections between entities.**
6. Test the questions against the model.
7. Test scalability.

Identify connections between entities

Connections are the verbs in your application questions:

1. What ingredients are used in a recipe?

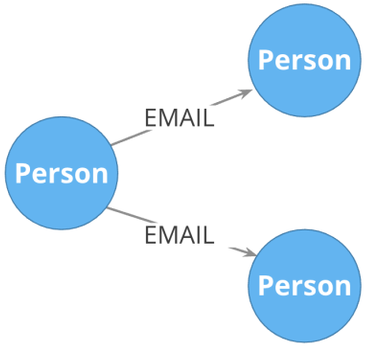
1. Who is married to this person?



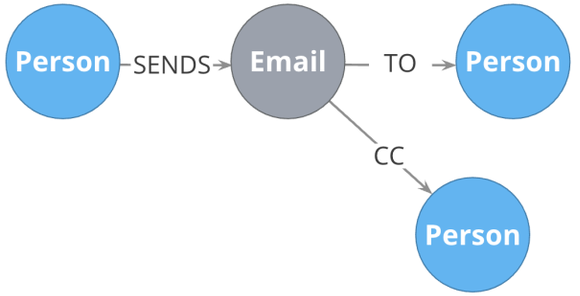
Naming relationships

- Stakeholders must agree upon name (type for the relationship).
- Avoid names that could be construed as nouns (for example email).
- Neo4j has a limit of 16M relationship types in Enterprise Edition (64K in Community Edition).

Do not do this:



Instead do this:

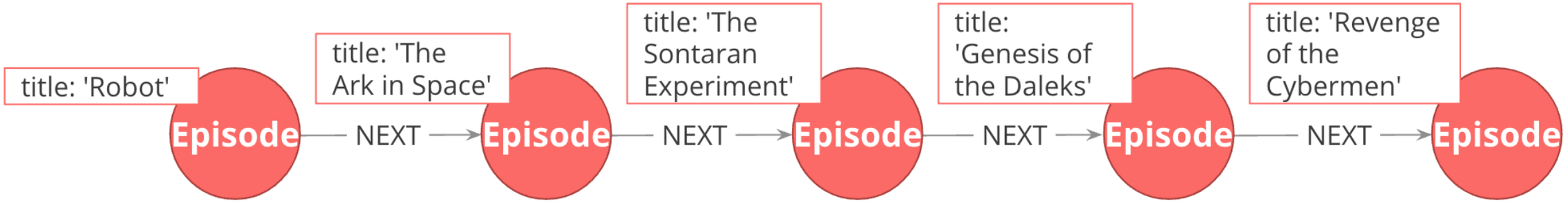


Direction and type

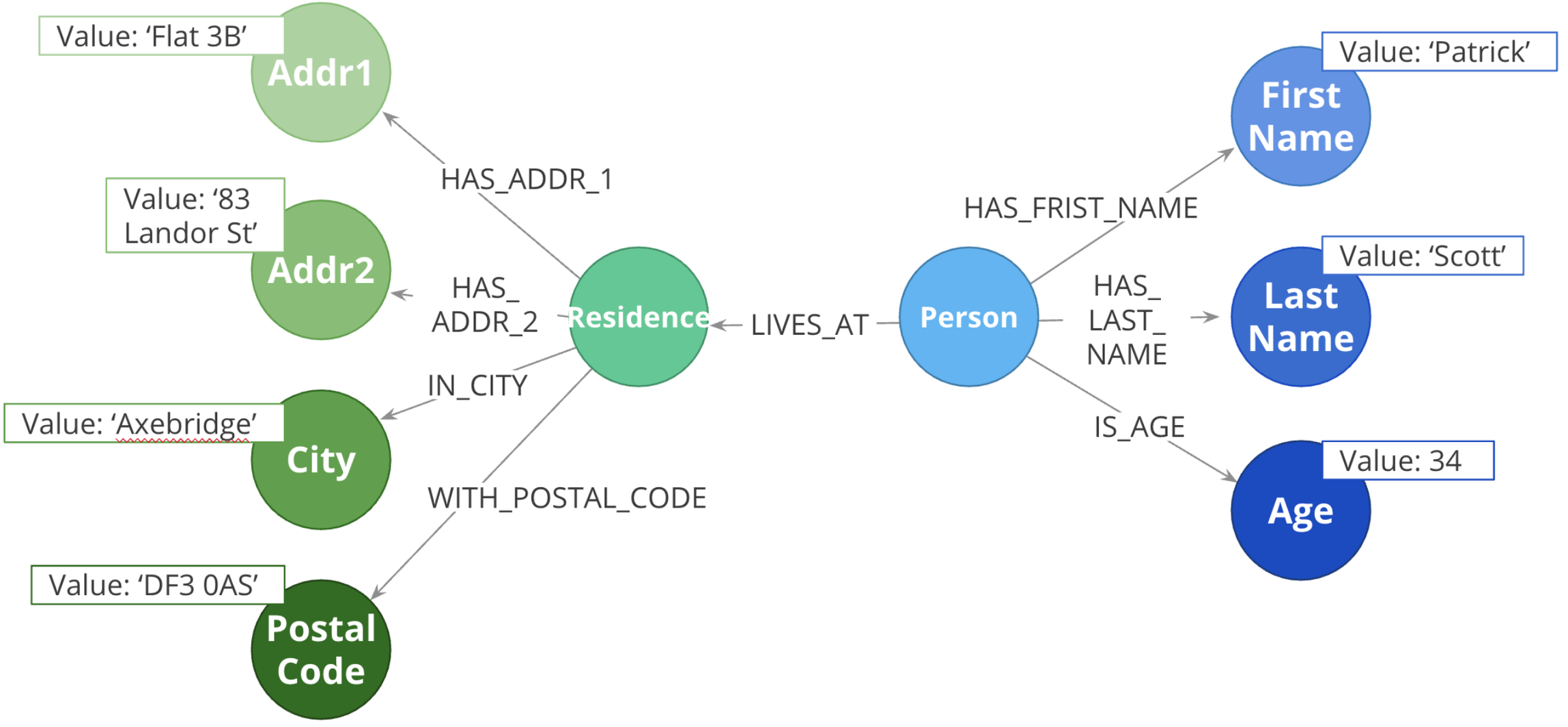
Direction and type are required in Neo4j.

Choose direction (and type) based on the expected questions:

1. What episode follows The Ark in Space? (NEXT)
2. What episode came before Genesis of the Daleks? (PREVIOUS)



How much fanout will a node have?



Exercise 3: Adding relationships to the model

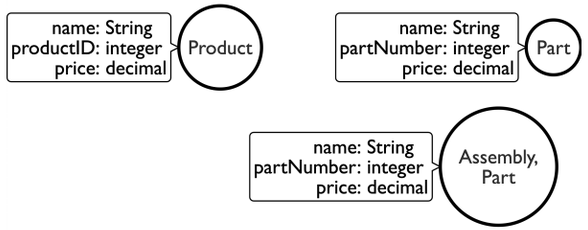
Follow the instructions on the next slide.

Exercise 3 instructions

Here are the questions we need to answer for our BOM application that you have already seen:

1. What parts are needed to make Wood table 40"?
2. Do we have enough parts to make 100 Deluxe wood table 60"?
3. What products require a table brace?
4. How much will the parts cost to make product Wood table 60"?

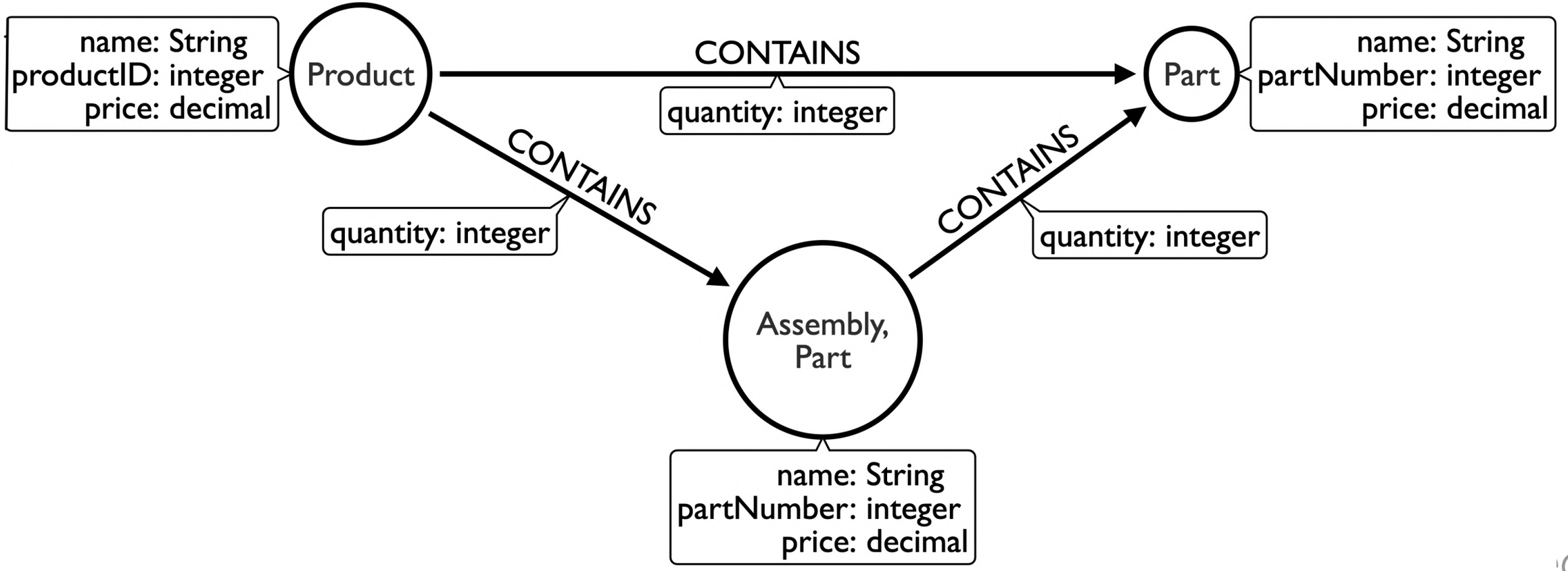
And here is the entity model:



Using the Arrow tool add the relationships between the entities.

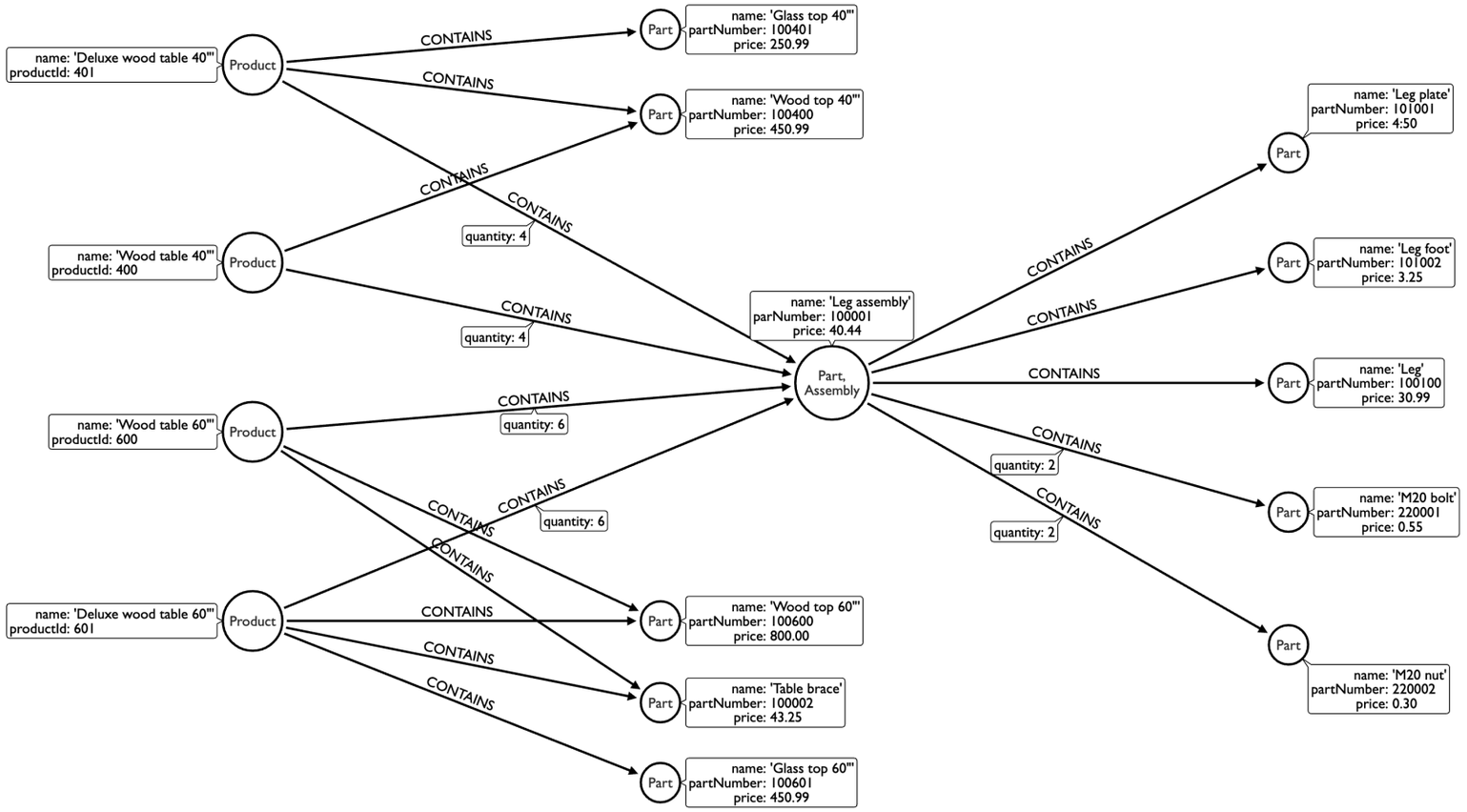
Exercise 3 solution

Here is what your graph data model might look like with relationships added:



Example: Detailed sample data for the BOM application

Here is what the sample data might look like in the UI of the Arrow tool:

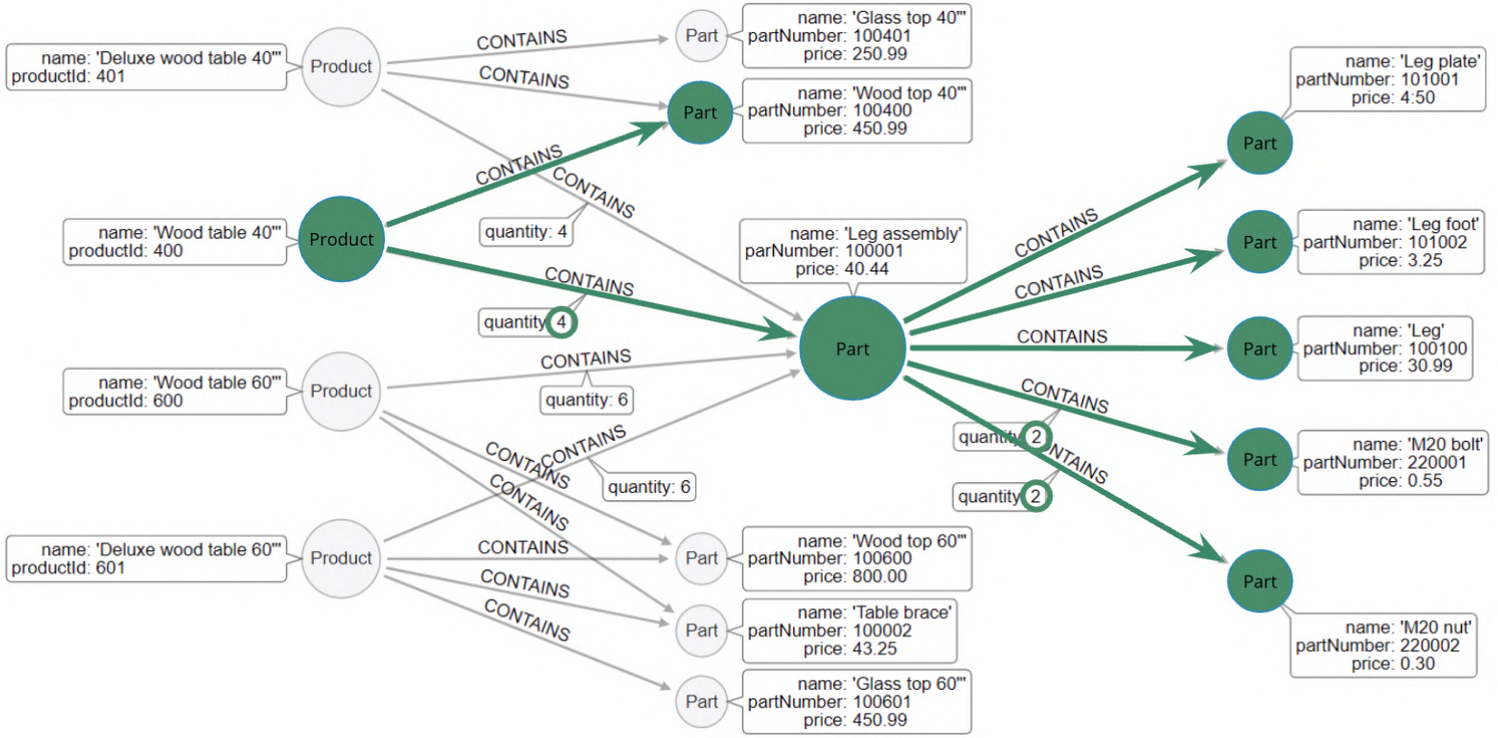


Step 6: Test the questions against the model

1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. Identify entities.
5. Identify connections between entities.
6. **Test the questions against the model.**
7. Test scalability.

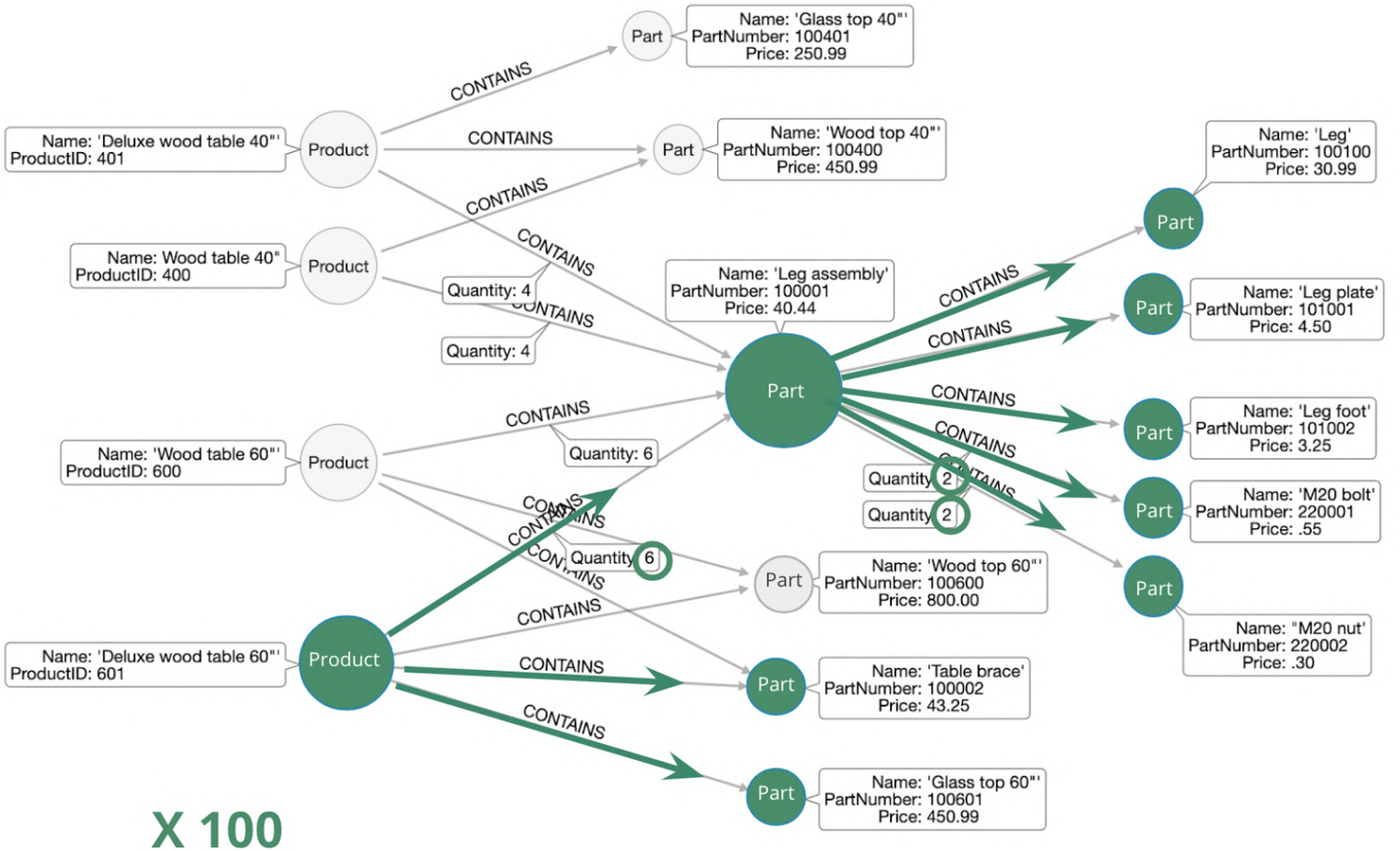
Testing the model - 1

What parts are needed to make Wood table 40"?



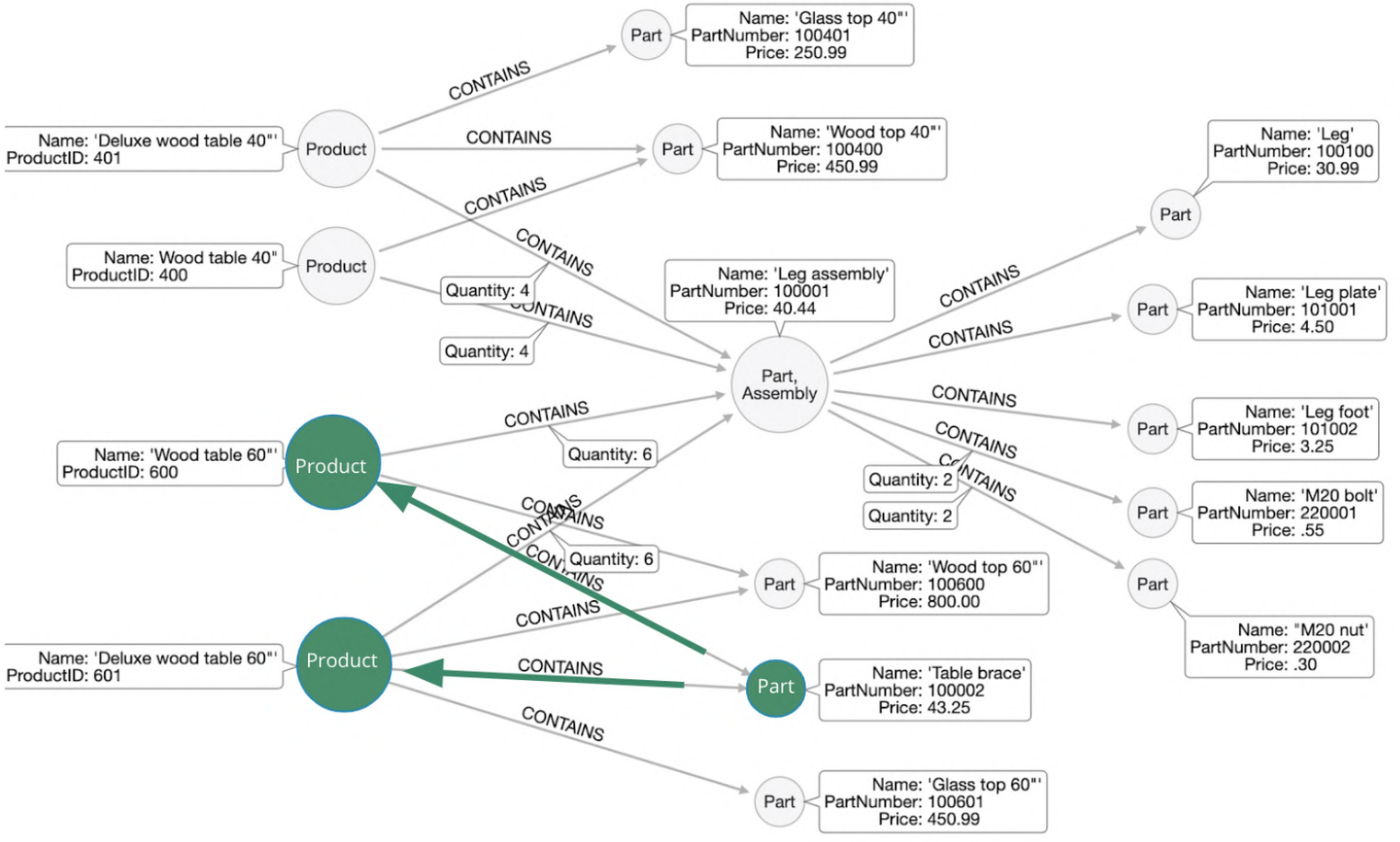
Testing the model - 2

Do we have enough parts to make 100x Deluxe wood table 60"?



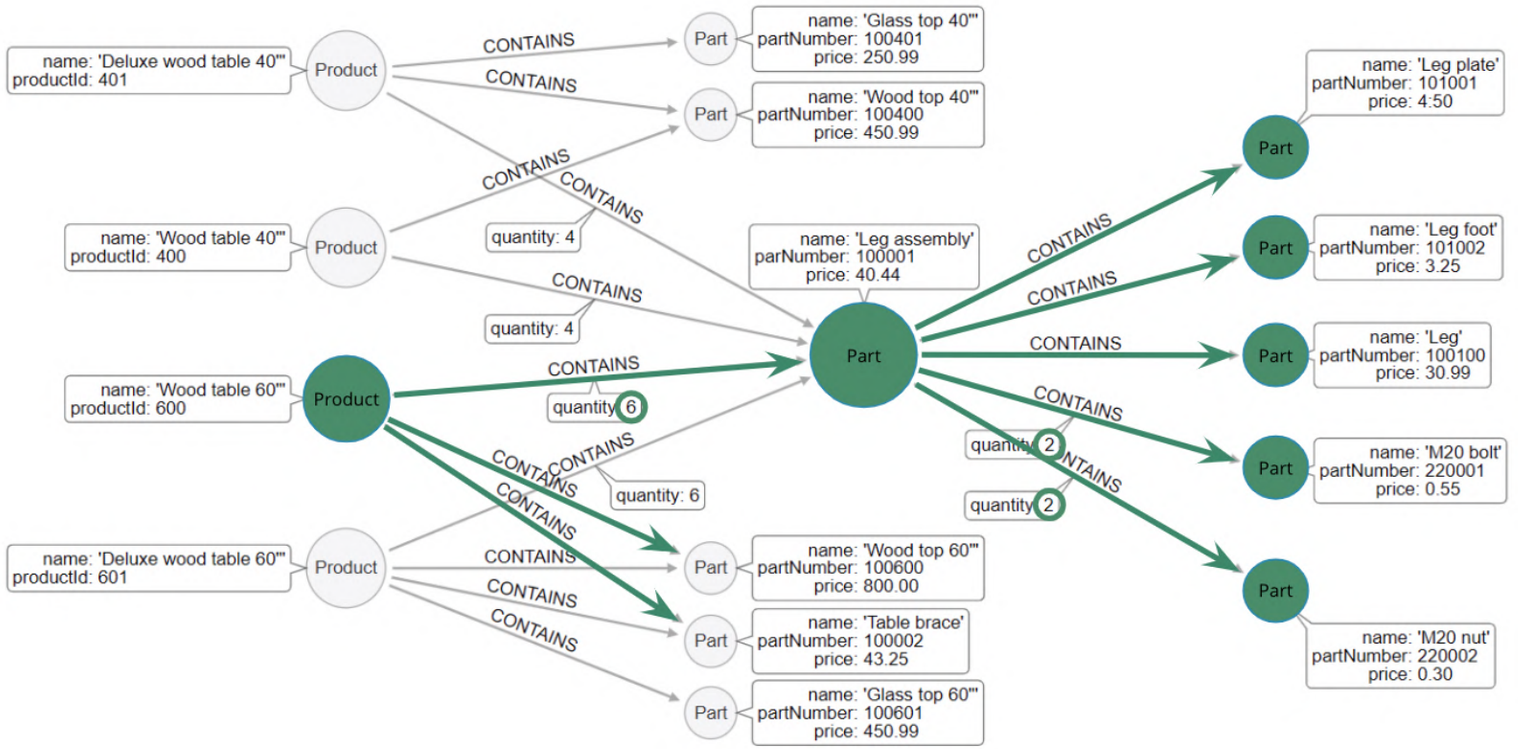
Testing the model - 3

What products require a table brace?



Testing the model - 4

How much will the parts cost to make Wood table 60"?



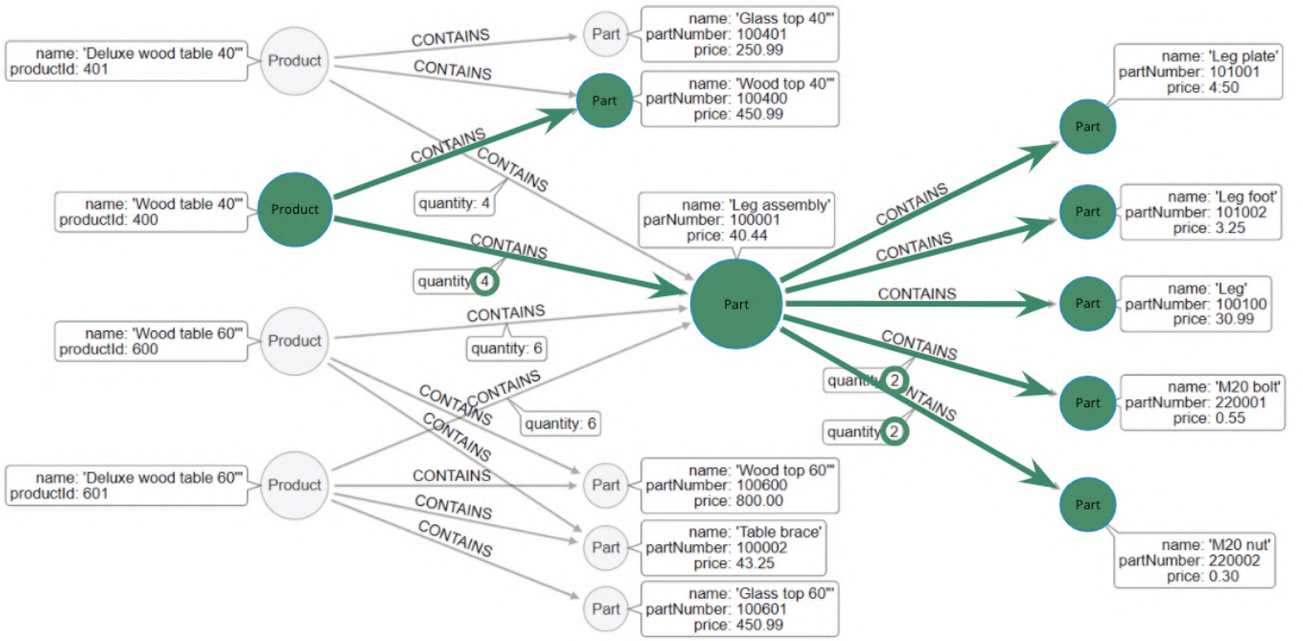
Step 7: Test scalability

1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. Identify entities.
5. Identify connections between entities.
6. Test the questions against the model.
7. **Test scalability.**

Testing scalability

Here are some questions you should answer:

- How many products?
- How many parts?
- How often are products added?
- How often do prices change?
- Are prices based upon time?
- Is inventory part of the model?



Check your understanding

Question 1

What component of a graph data model is used to model the nouns of the questions for the domain?

Select the correct answer.

- [ ] Relationship
- [ ] Property
- [ ] Entity
- [ ] Data source

Question 1

What component of a graph data model is used to model the nouns of the questions for the domain?

Select the correct answer.

- [ ] Relationship
- [ ] Property
- [x] **Entity**
- [ ] Data source

Question 2

What component of a graph data model is used to model the verbs of the questions for the domain?

Select the correct answer.

- [ ] Relationship
- [ ] Property
- [ ] Entity
- [ ] Behavior

Question 2

What component of a graph data model is used to model the verbs of the questions for the domain?

Select the correct answer.

- [x] **Relationship**
- [ ] Property
- [ ] Entity
- [ ] Behavior

Question 3

When you create the model for your application, at a minumum, what must you specify for a relationship?

Select the correct answers.

- [ ] Type
- [ ] Weight
- [ ] Direction
- [ ] From and to entities

Question 3

When you create the model for your application, at a minumum, what must you specify for a relationship?

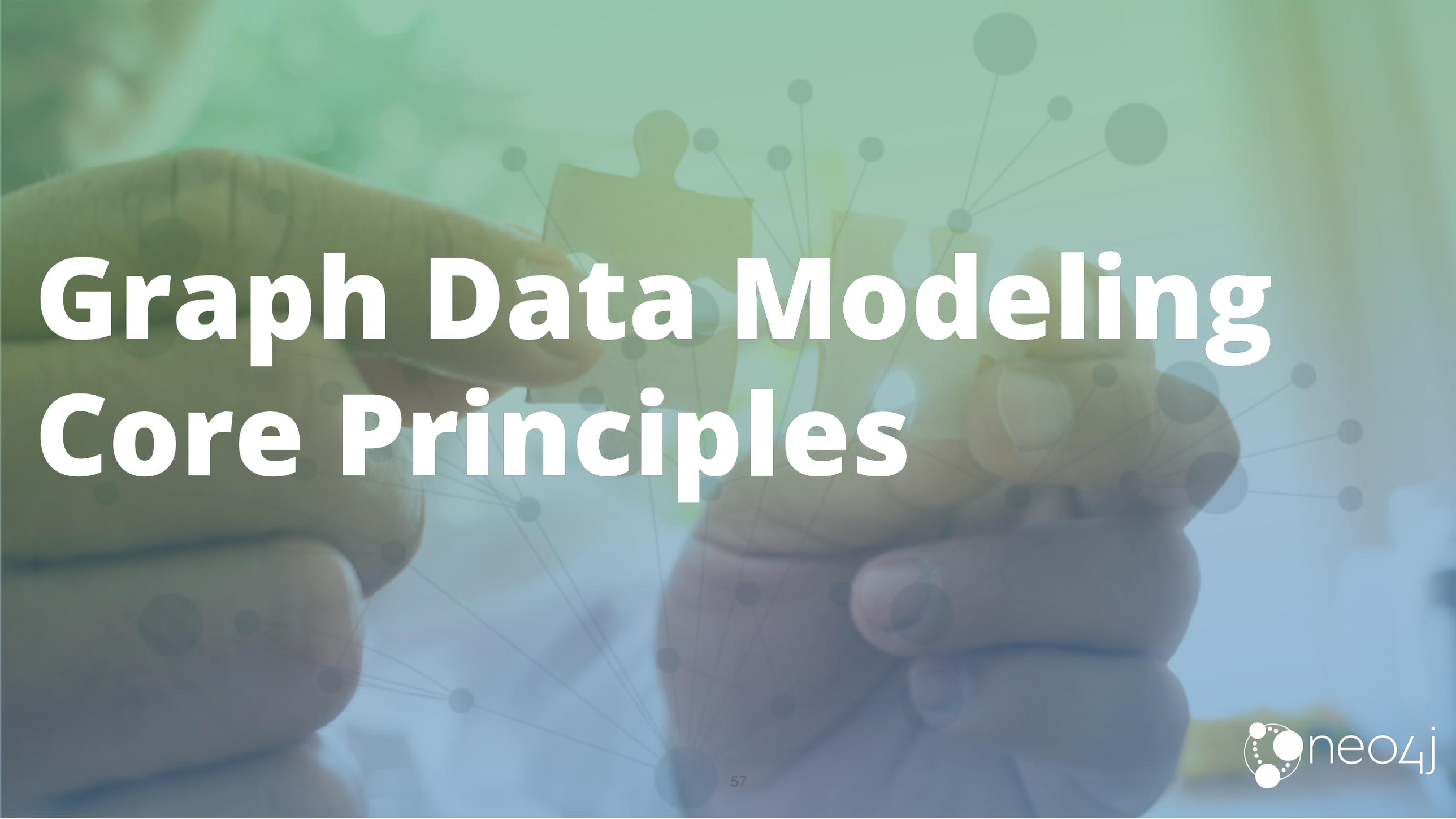
Select the correct answers.

- ☑ **Type**
- ☐ Weight
- ☑ **Direction**
- ☑ **From and to entities**

Summary

You should now be able to:

- Describe the domain for a model.
- Define the questions for the domain.
- Identify entities from the questions for the domain.
- Use the Arrows Tool to model the domain.
- Identify the connections between entities.
- Describe how to test the initial model.



Graph Data Modeling Core Principles

About this module

At the end of this module, you should be able to:

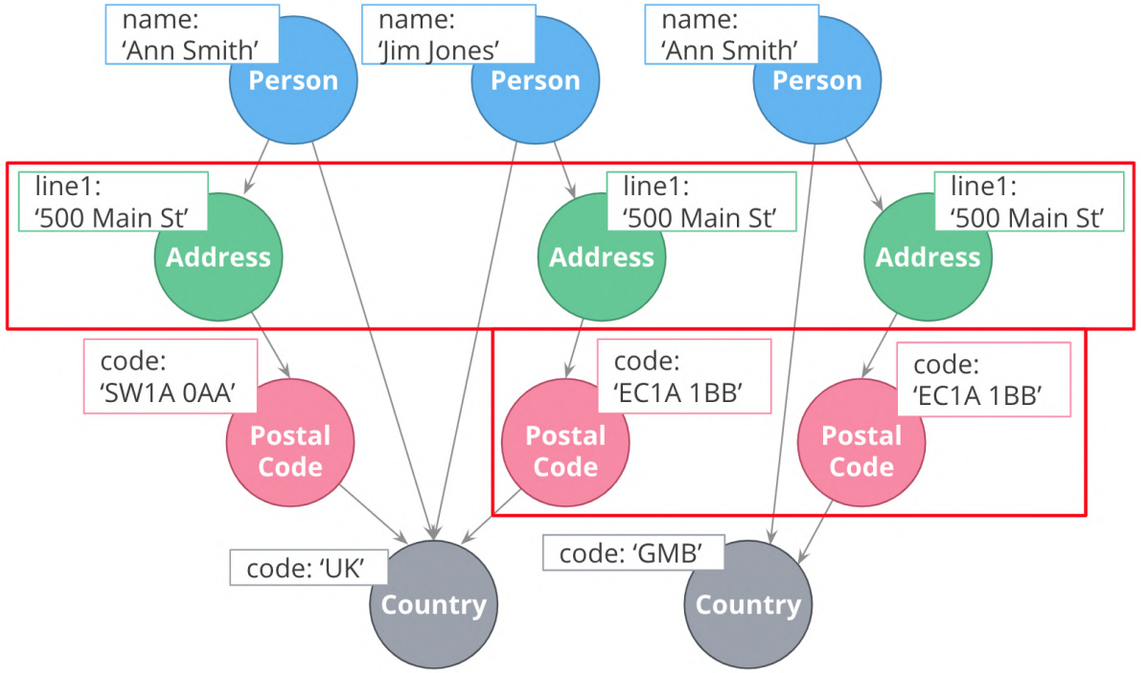
- Describe graph data modeling best practices for modeling:
  - Nodes (entities)
  - Relationships
  - Properties
- Describe data accessibility

Best practices for modeling nodes

Your model should address:

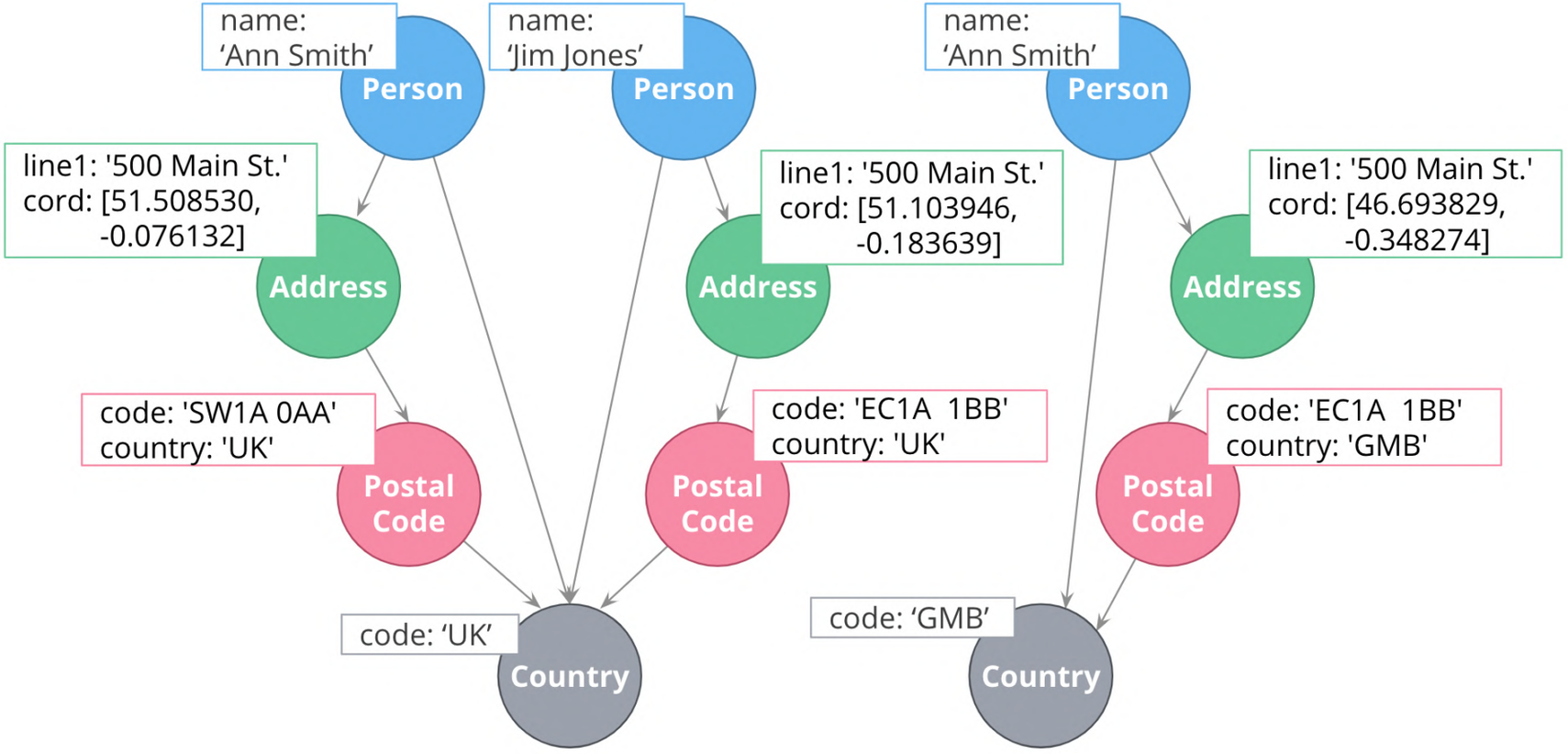
- Uniqueness of nodes
- Complex data

Uniqueness of nodes: before



Note The Country nodes are considered **super nodes** (a node with lots of fan-in or fan-out). You need to be careful about using them in your design and developers in particular need to be mindful of queries that might select all paths in or out of the super node.

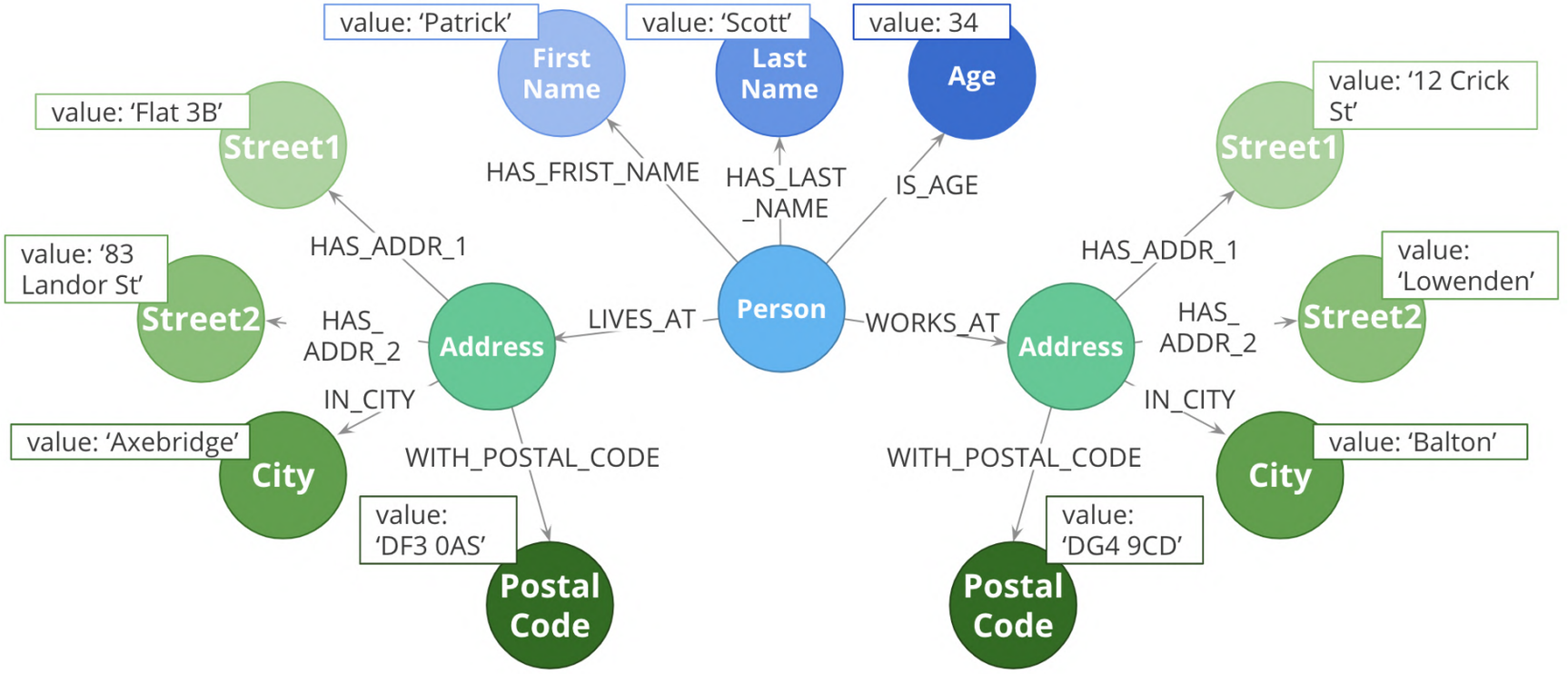
Uniqueness of nodes: after



Complex data

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

Use fanout judiciously for complex data



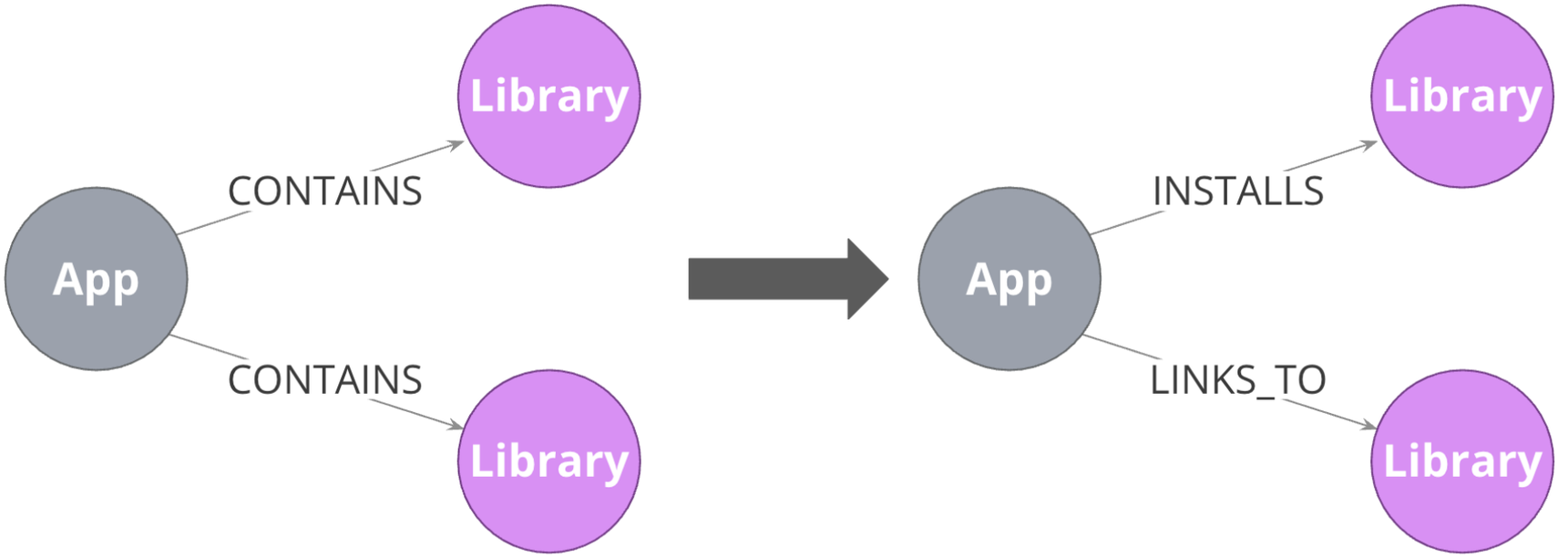
- Reduce property duplication.
- Reduce gather-and-inspect.

Best practices for modeling relationships

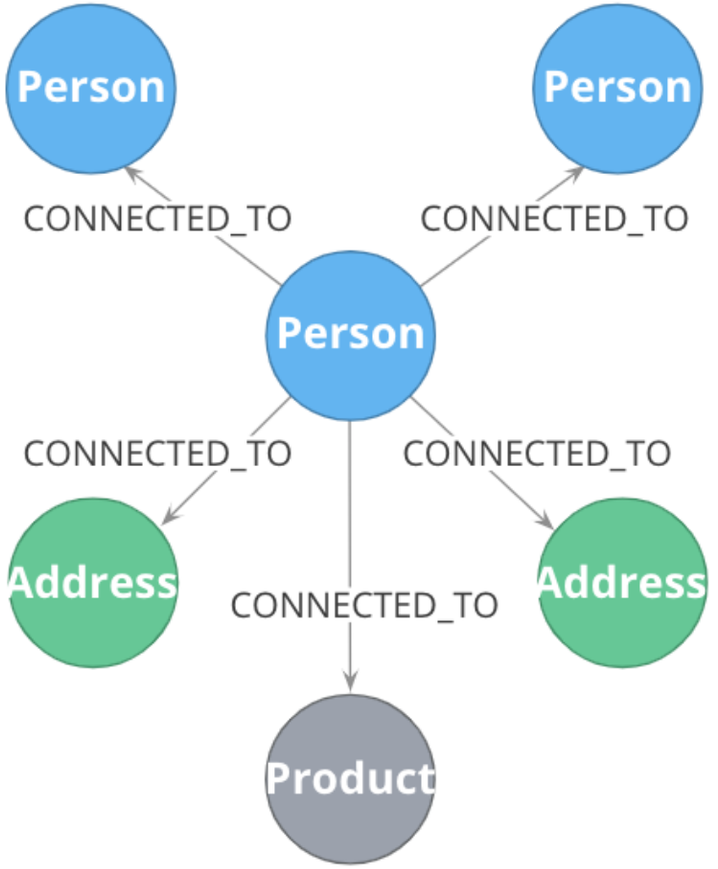
Your model should address:

- Using specific relationship types.
- Reducing symmetric relationships.
- Using types vs. properties.

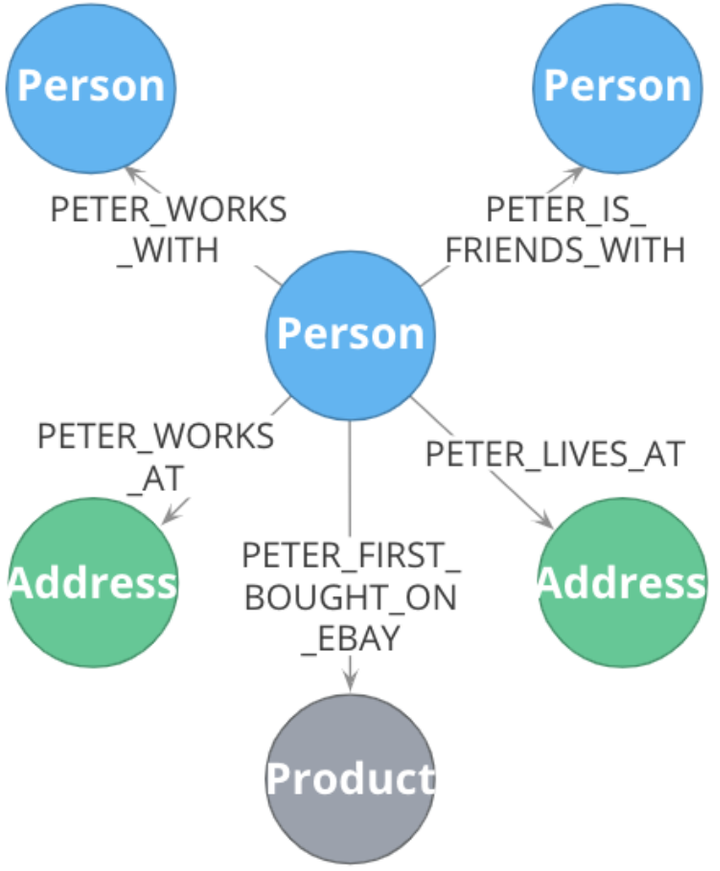
Using specific relationship types



But… not too specific



Not specific enough

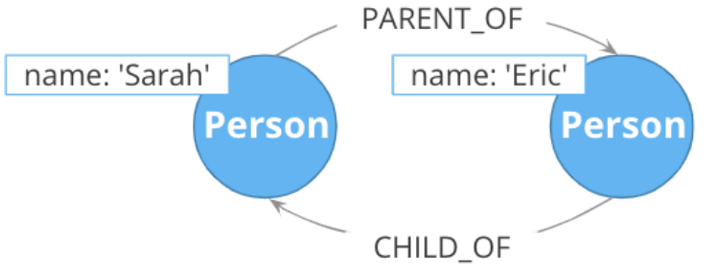


Too specific

neo4j

No symmetric relationships

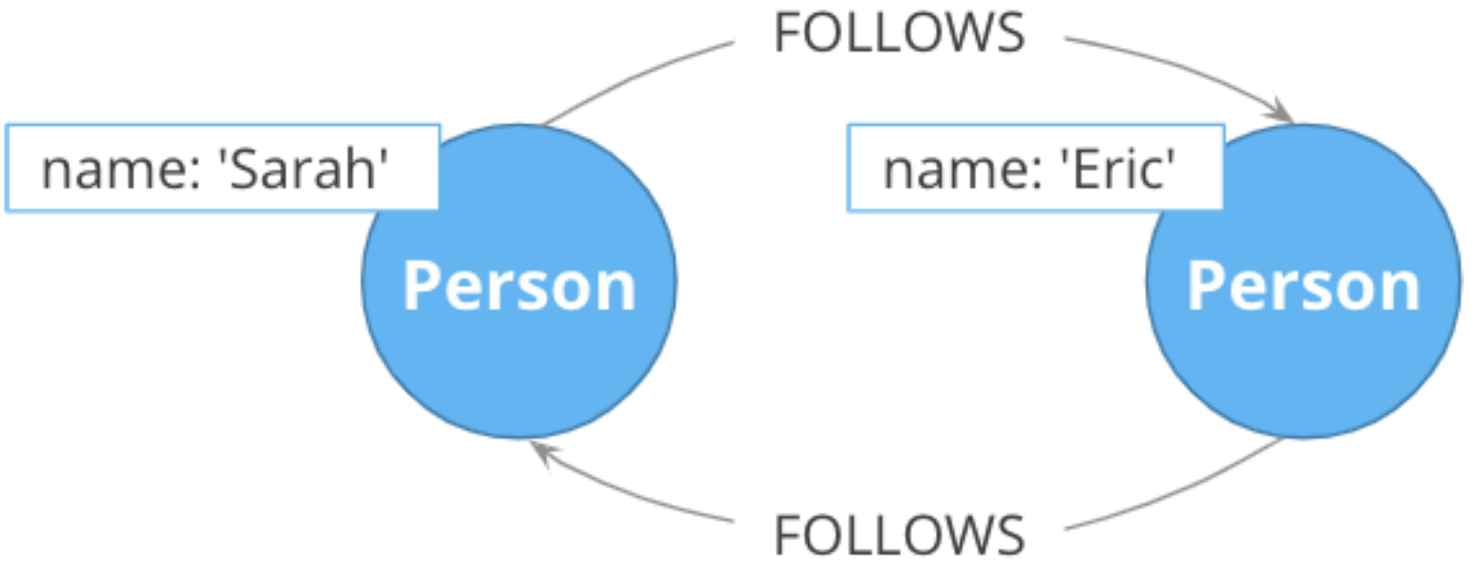
You should never do this:



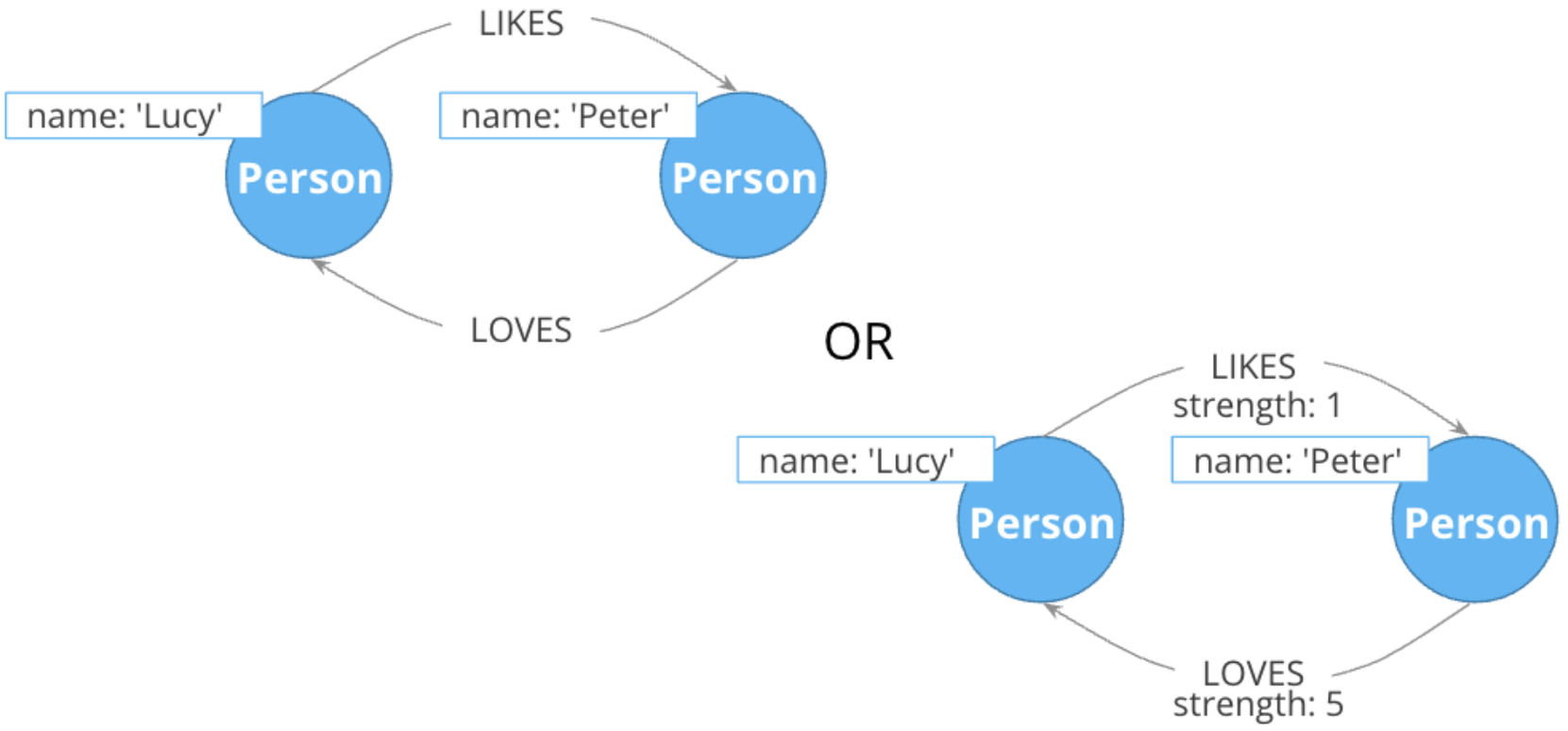
Do one of these:

or

Semantics of symmetry are important

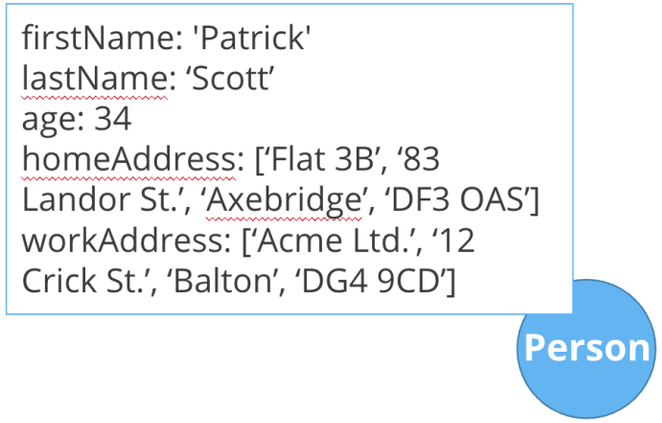


Using types vs. properties



Property best practices

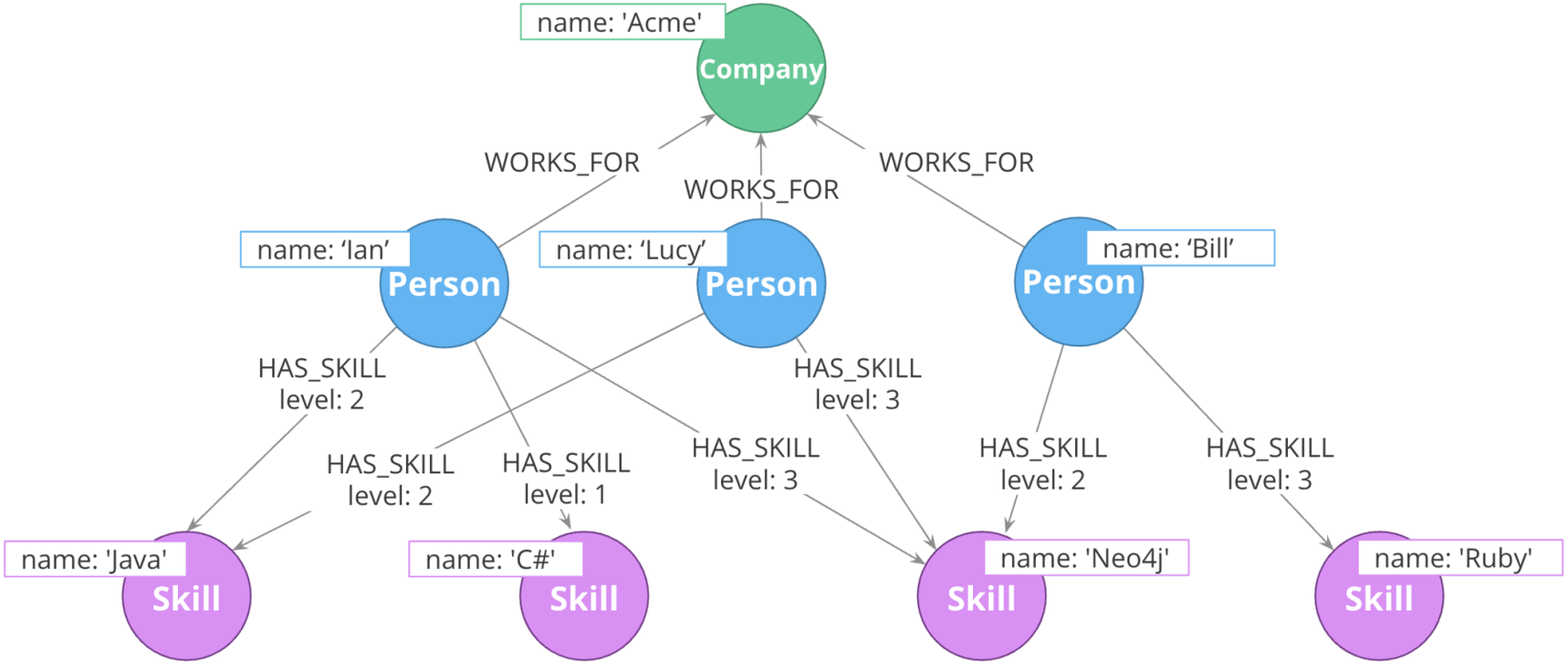
- Property lookups have a cost.
- Parsing a complex property adds more cost.



- Anchors and properties used for traversal should be as simple as possible.
- Identifiers, outputs, and decoration are OK as complex values.

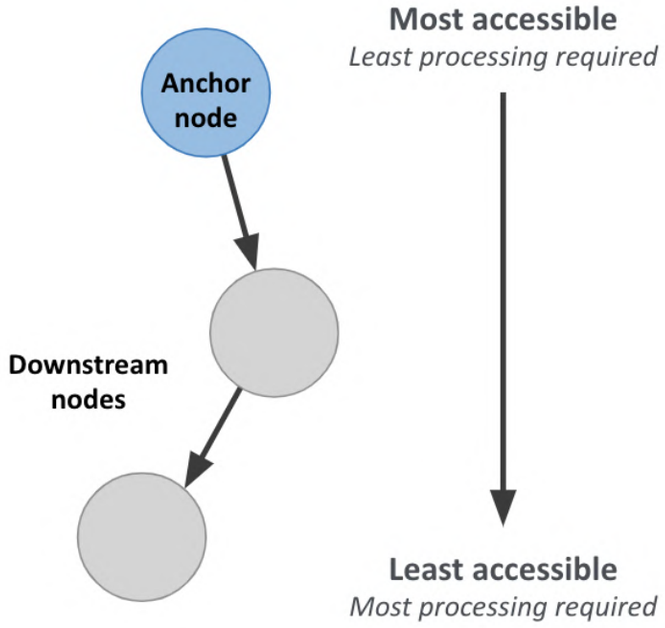
Data accessibility

For each query, how much work must Neo4j do to evaluate if the traversal represents a “good” path or a “bad” one?



Hierarchy of accessibility

For each data object, how much work must Neo4j do to evaluate if this is a "good" path or a "bad" one?



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

Check your understanding

Question 1

What are some benefits of using fanout for your nodes?

Select the correct answers.

- [ ] Reduces the number of nodes in the graph.
- [ ] Reduces duplication of property values.
- [ ] Reduces the number of relationships defined in the graph.
- [ ] Reduces gather-and-inspect traversals during a query.

Question 1

What are some benefits of using fanout for your nodes?

Select the correct answers.

- ☐ Reduces the number of nodes in the graph.
- ✓ **Reduces duplication of property values.**
- ☐ Reduces the number of relationships defined in the graph.
- ✓ **Reduces gather-and-inspect traversals during a query.**

Question 2

Why is naming relationship types to be as specific as possible a benefit?

Select the correct answers.

- [ ] Reduces the number of relationships in the graph.
- [ ] Reduces traversals through nodes that are not necessary for the query.
- [ ] Reduces gather-and-inspect traversals during a query.
- [ ] Reduces the number of nodes in the graph.

Question 2

Why is naming relationship types to be as specific as possible a benefit?

Select the correct answers.

- ☐ Reduces the number of relationships in the graph.
- ✓ **Reduces traversals through nodes that are not necessary for the query.**
- ✓ **Reduces gather-and-inspect traversals during a query.**
- ☐ Reduces the number of nodes in the graph.

Question 3

Which data should be most accessible for your queries?

Select the correct answers.

- [ ] Anchor node label
- [ ] Anchor node property that has an index
- [ ] Node property downstream that has an index
- [ ] Relationship properties

Question 3

Which data should be most accessible for your queries?

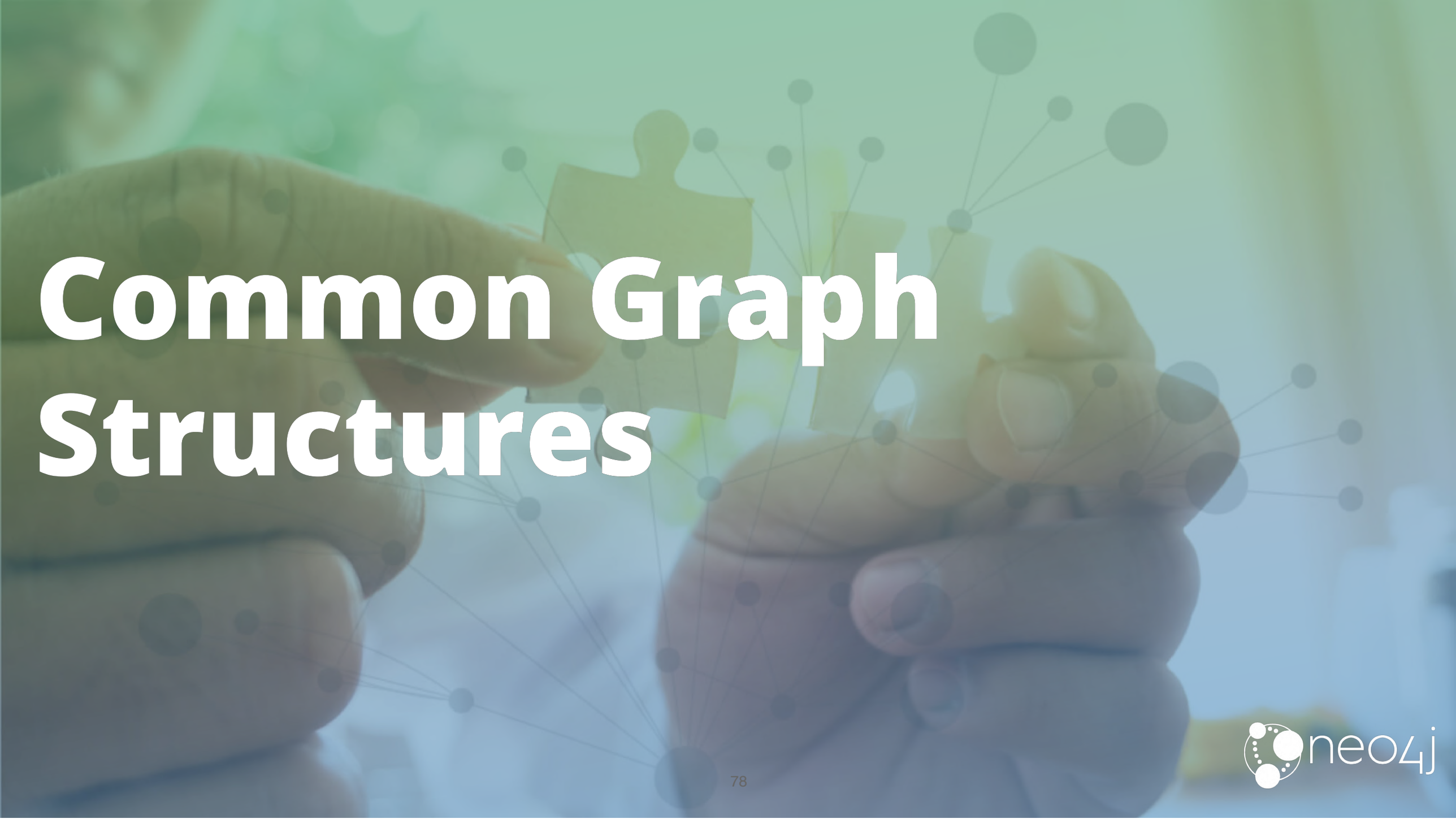
Select the correct answers.

- [x] **Anchor node label**
- [x] **Anchor node property that has an index**
- [ ] Node property downstream that has an index
- [ ] Relationship properties

Summary

You should now be able to:

- Describe graph data modeling best practices for modeling:
  - Nodes (entities)
  - Relationships
  - Properties
- Describe data accessibility



Common Graph Structures

neo4j

About this module

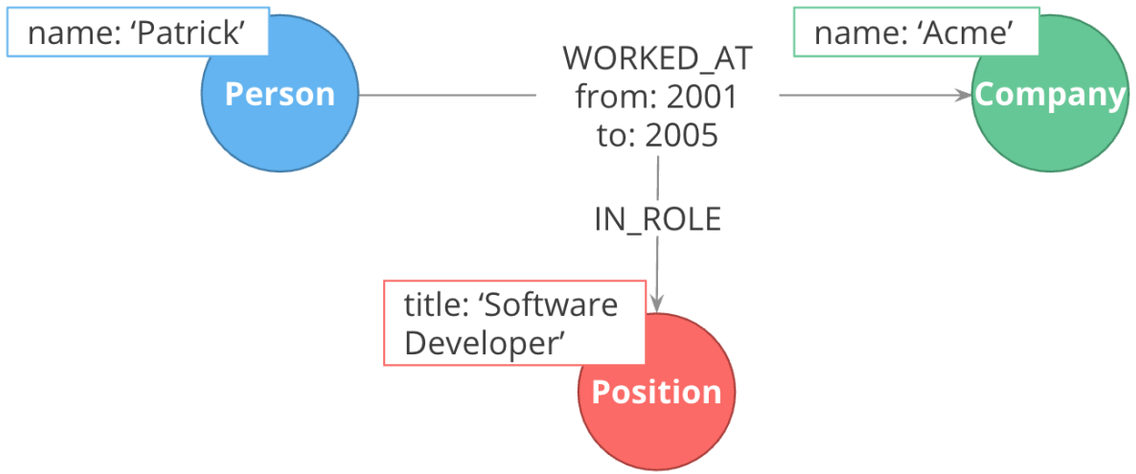
At the end of this module, you should be able to:

- Describe common graph structures used in modeling:
  - Intermediate nodes
  - Linked lists
  - Timeline trees
  - Multiple structures in a single graph

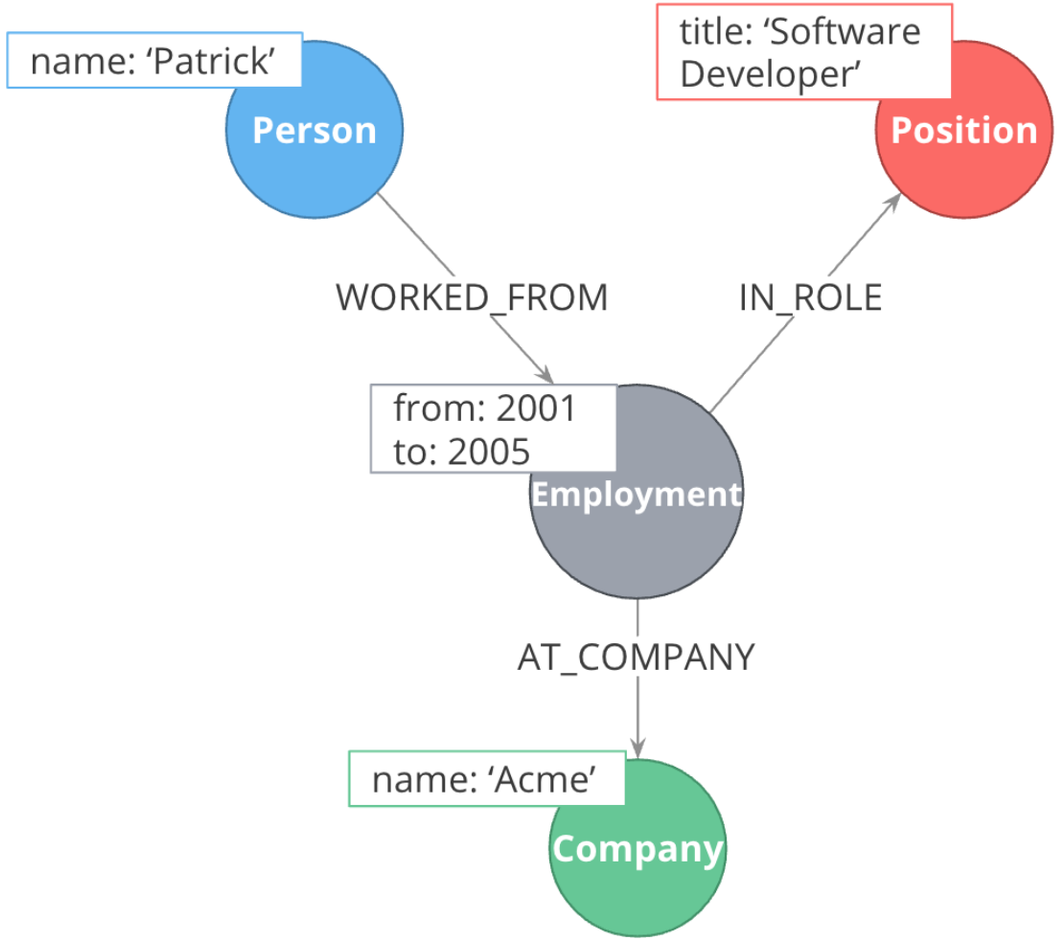
Intermediate nodes

You create intermediate nodes when you need to:

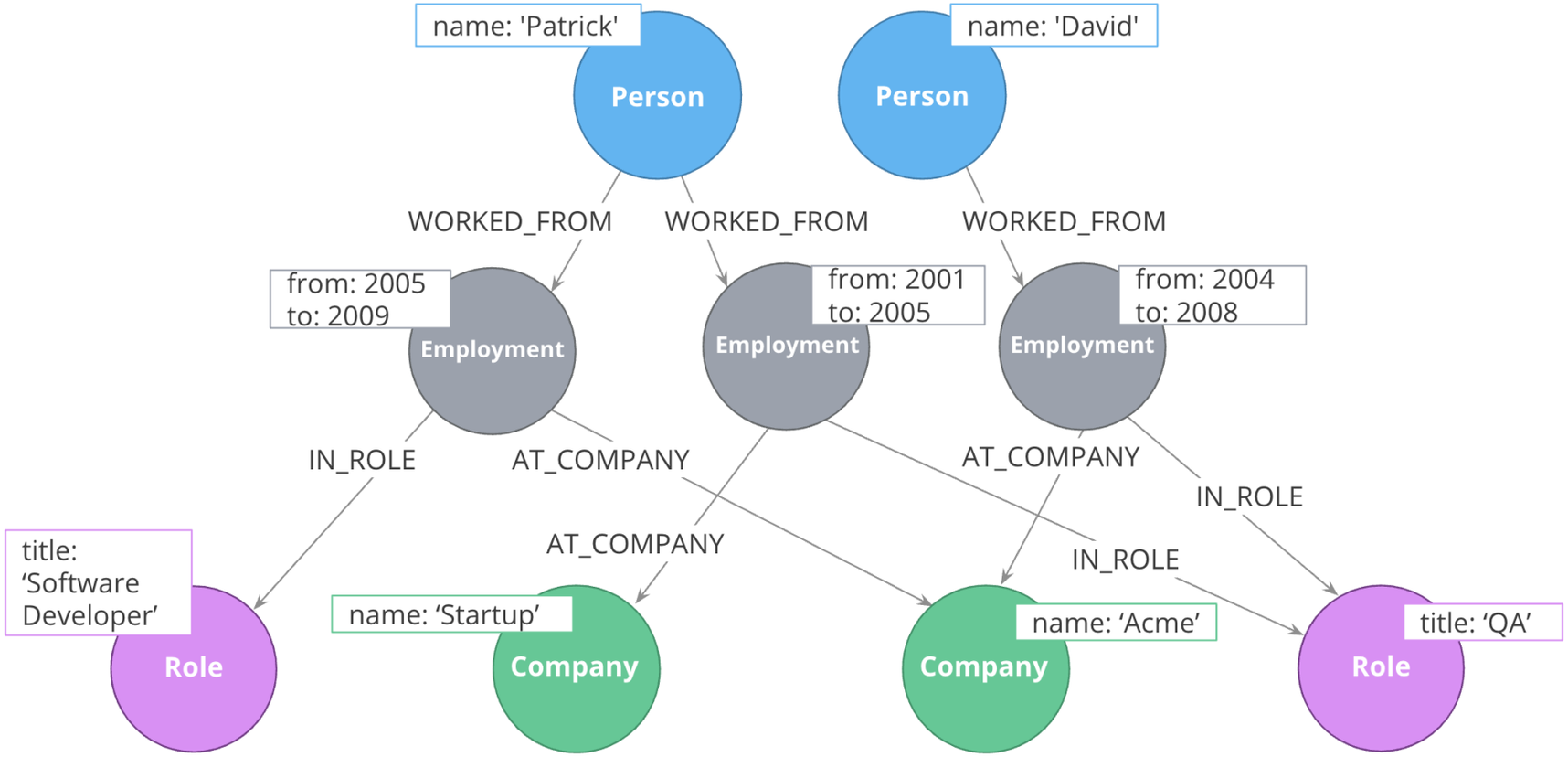
- Connect more than two nodes in a single context.
  - Hyperedges (n-ary relationships)
- Relate something to a relationship.



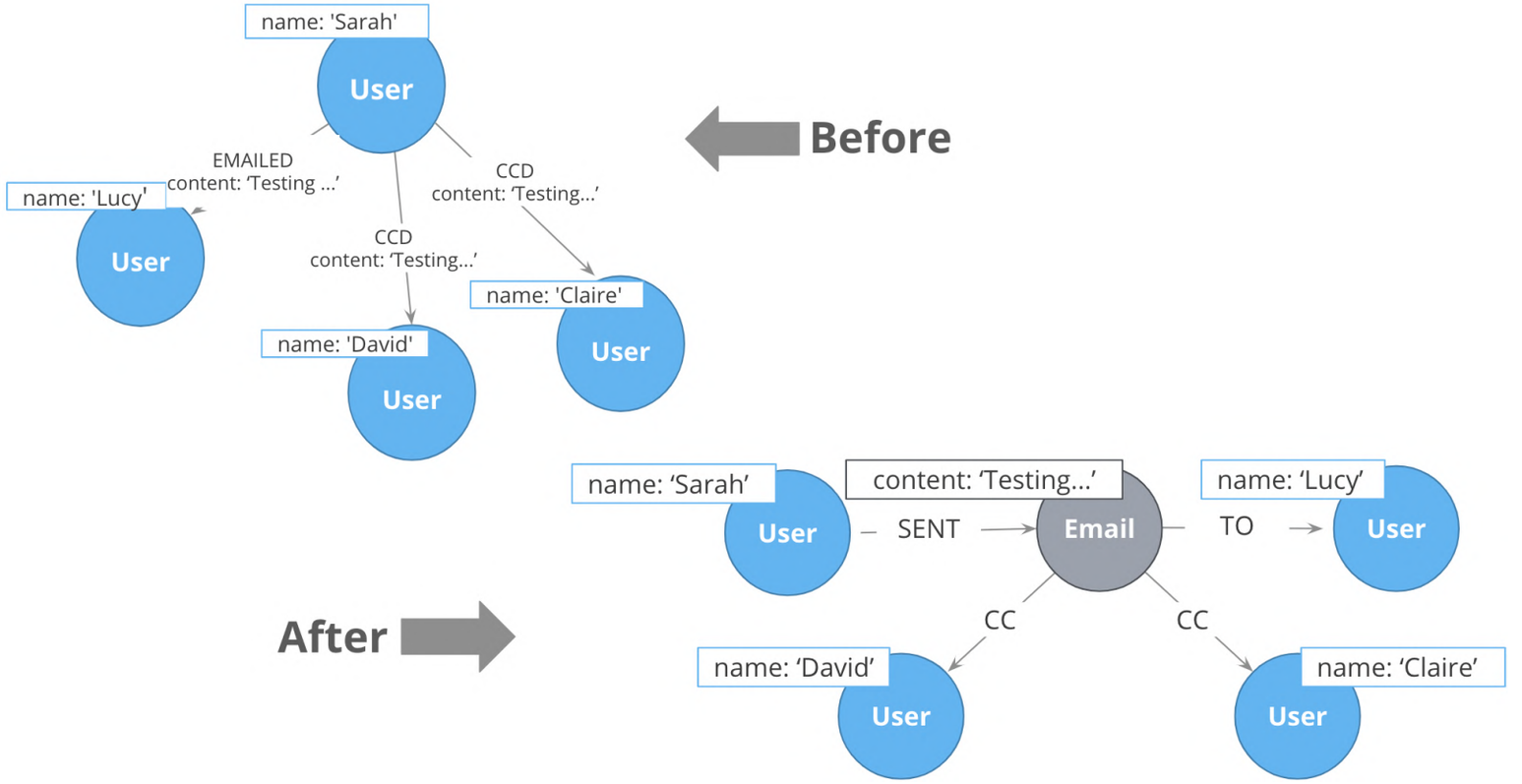
Using intermediate nodes



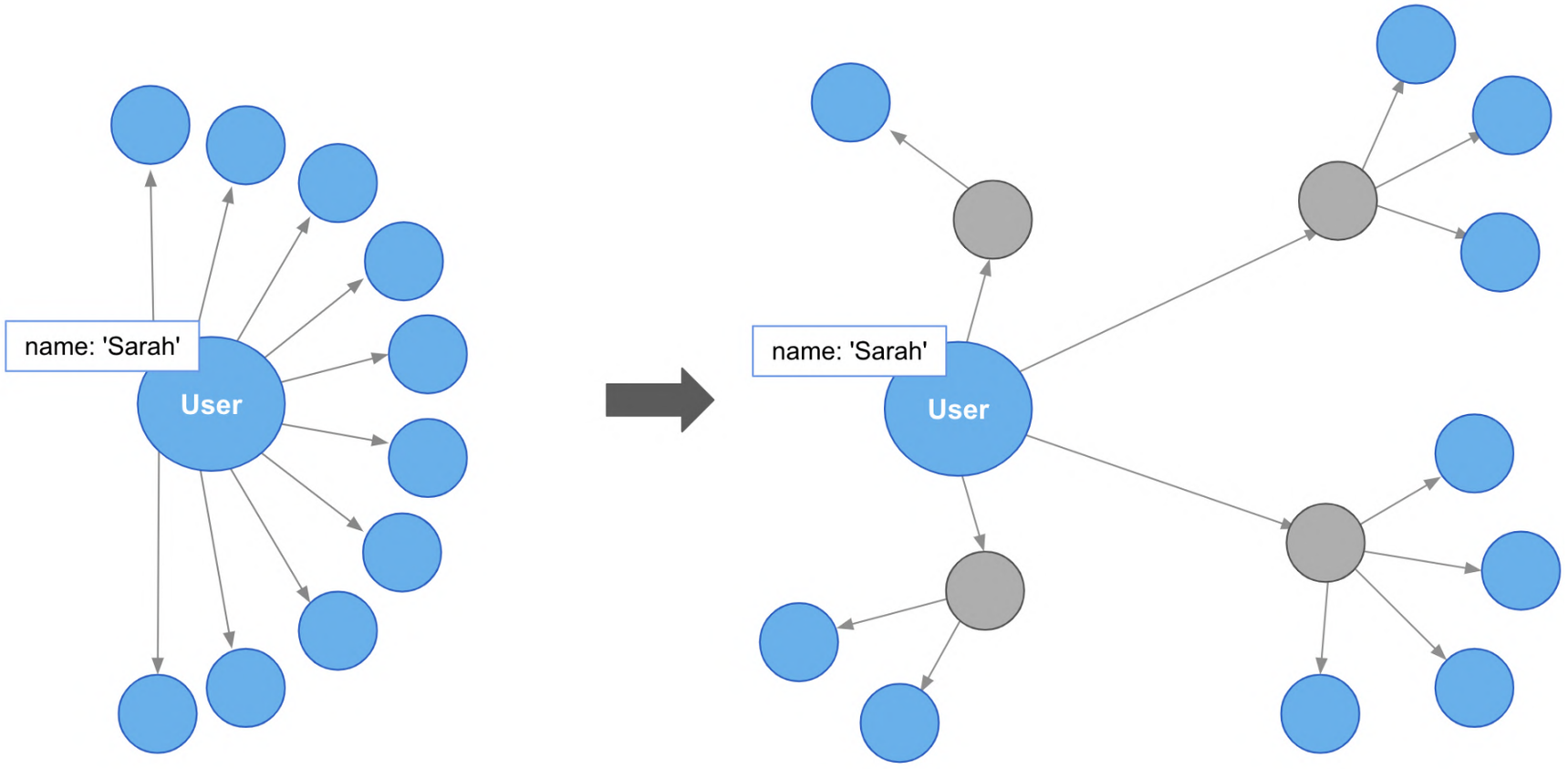
Intermediate nodes: sharing context



Intermediate nodes: sharing data

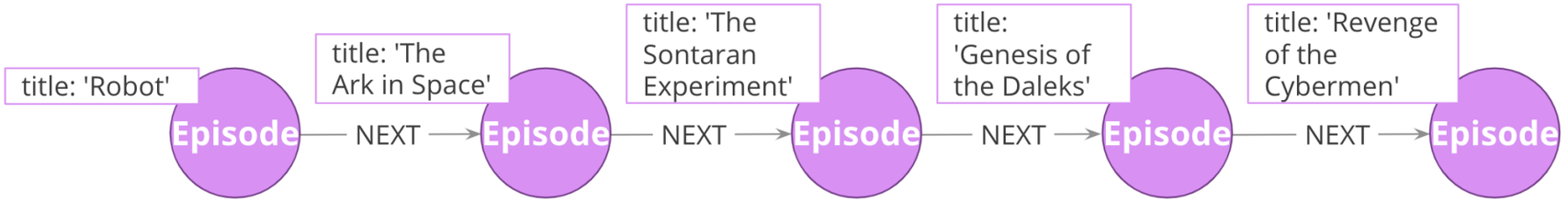


Intermediate nodes: organizing data

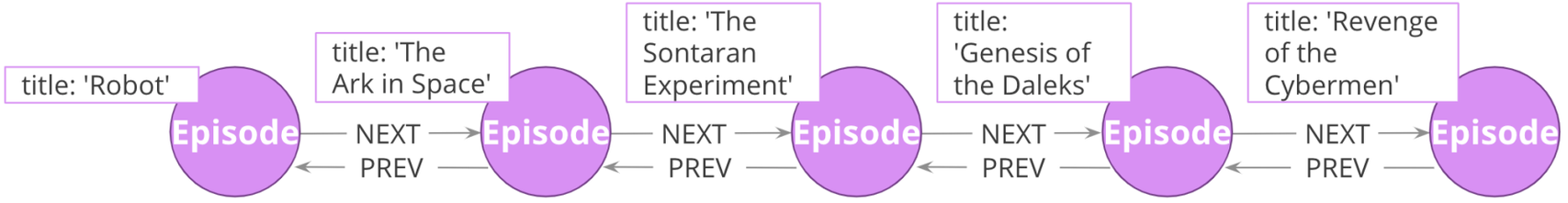


Linked lists

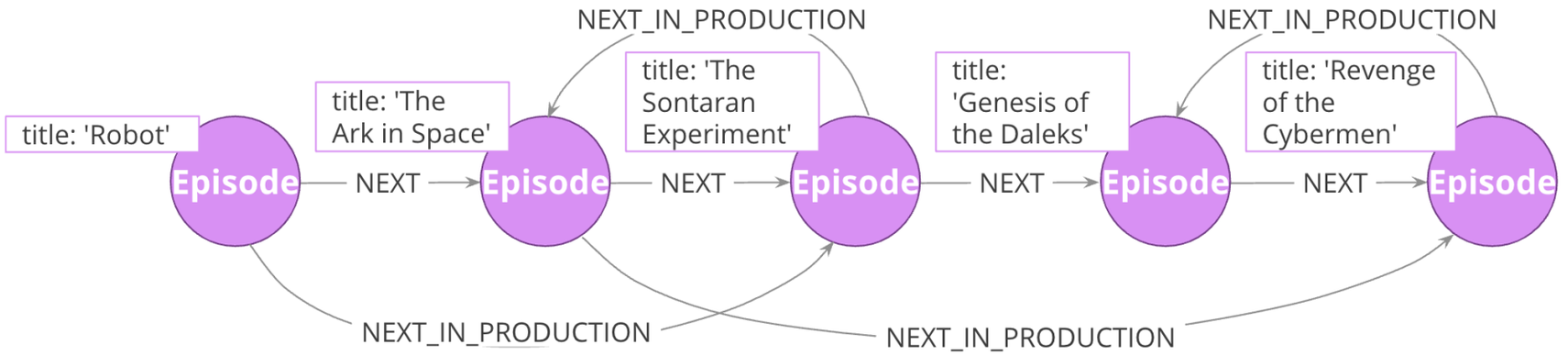
Episodes of the Dr.Who series:



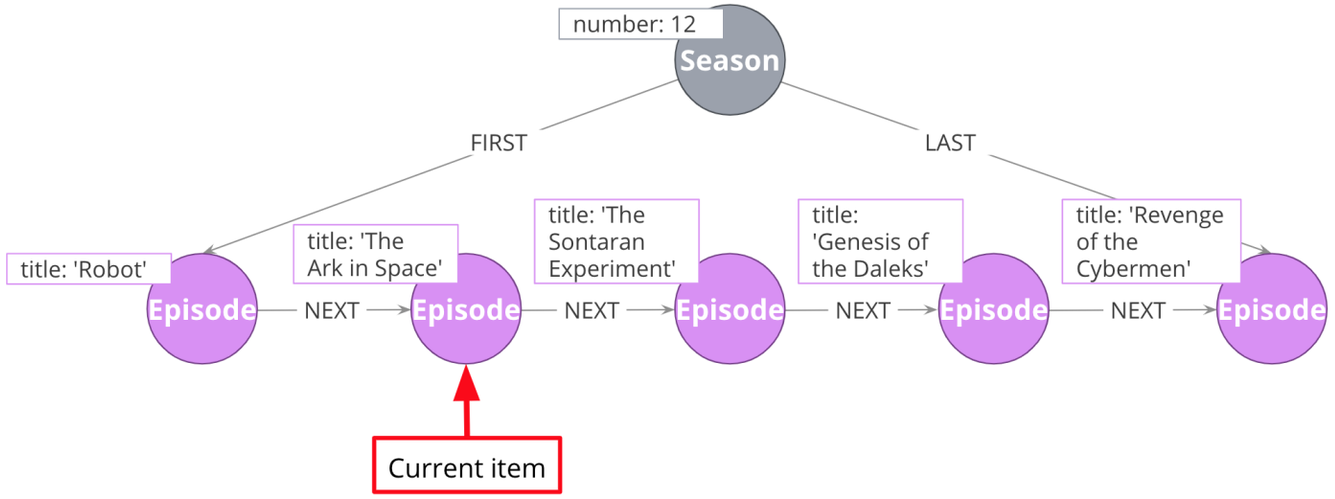
Do NOT do this (doubly-linked list):



Interleaved linked list



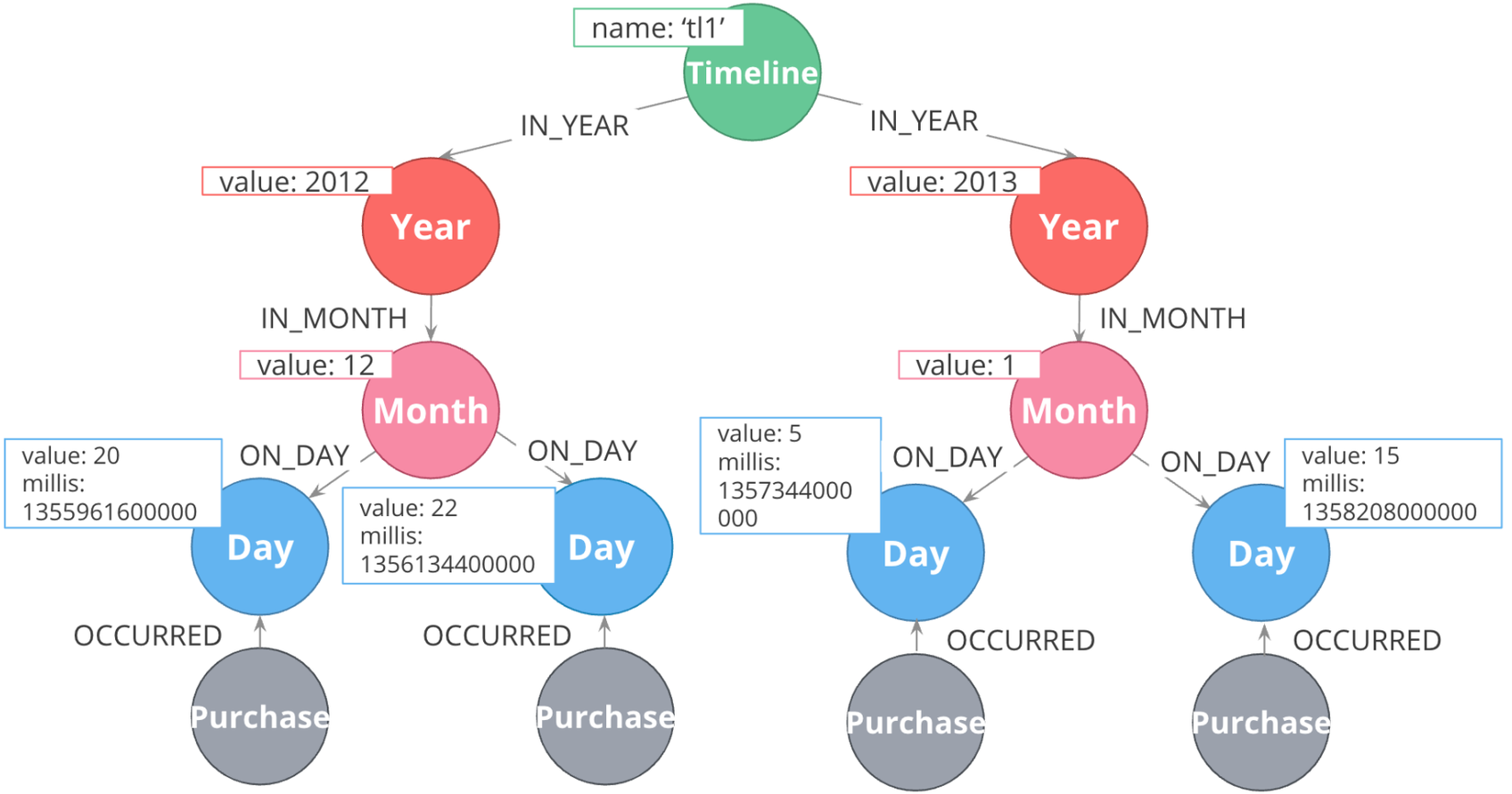
Head and tail of linked list



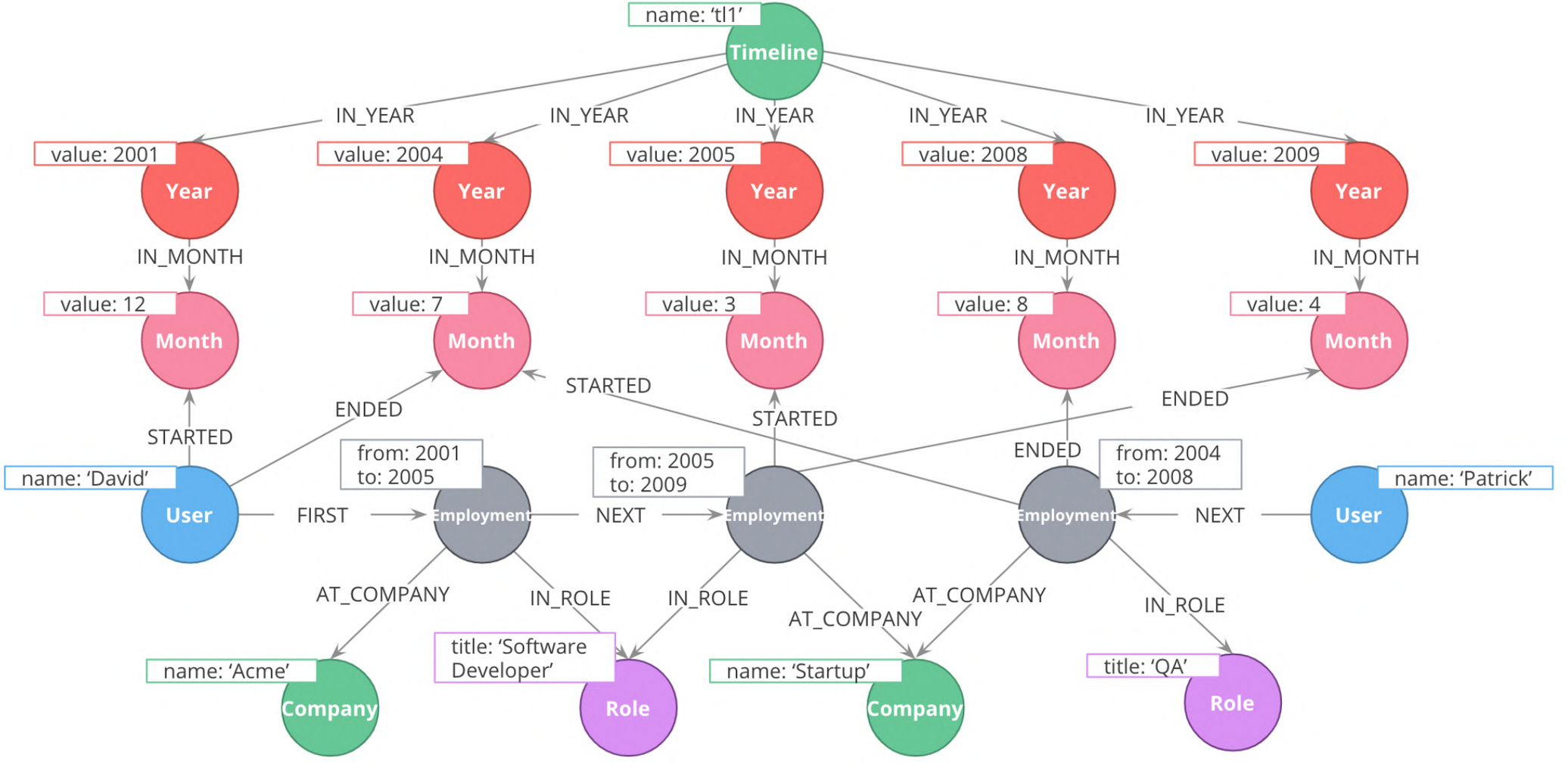
Some possible use cases:

- Add episodes as they are broadcast.
- Maintain pointer to first and last episodes.
- Find all episodes broadcast so far.
- Find latest episode broadcast so far.

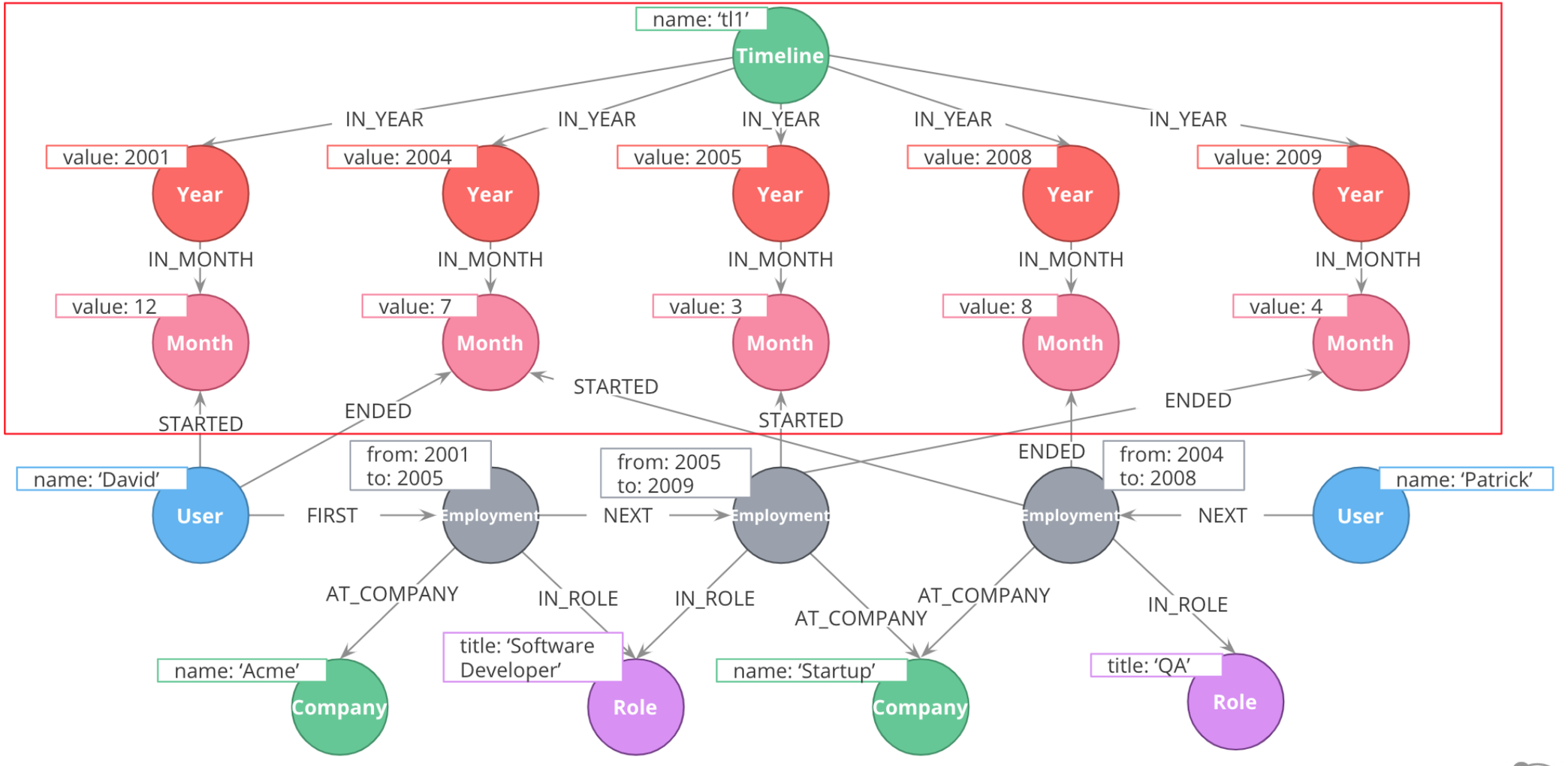
Timeline tree



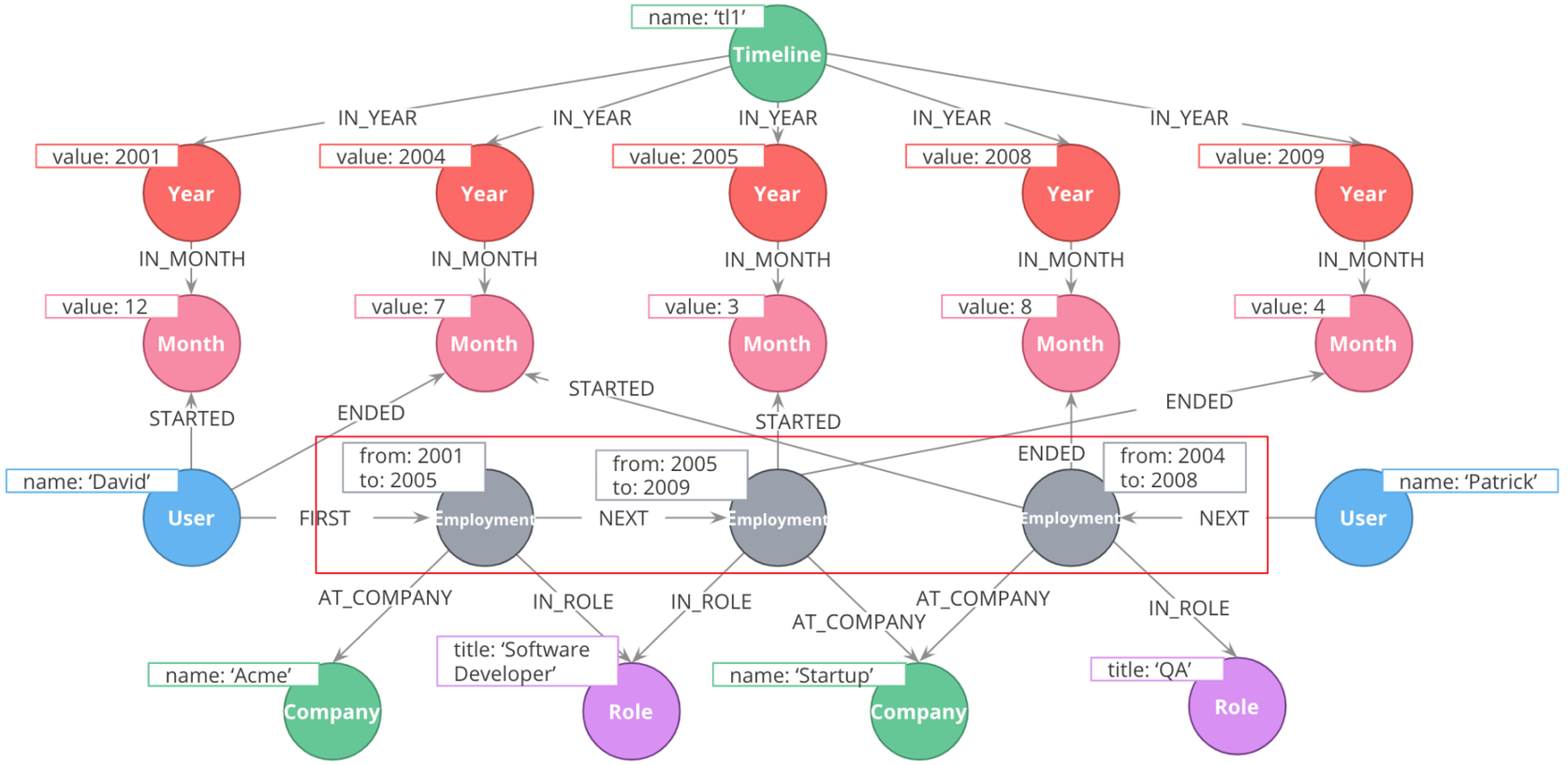
Using multiple structures



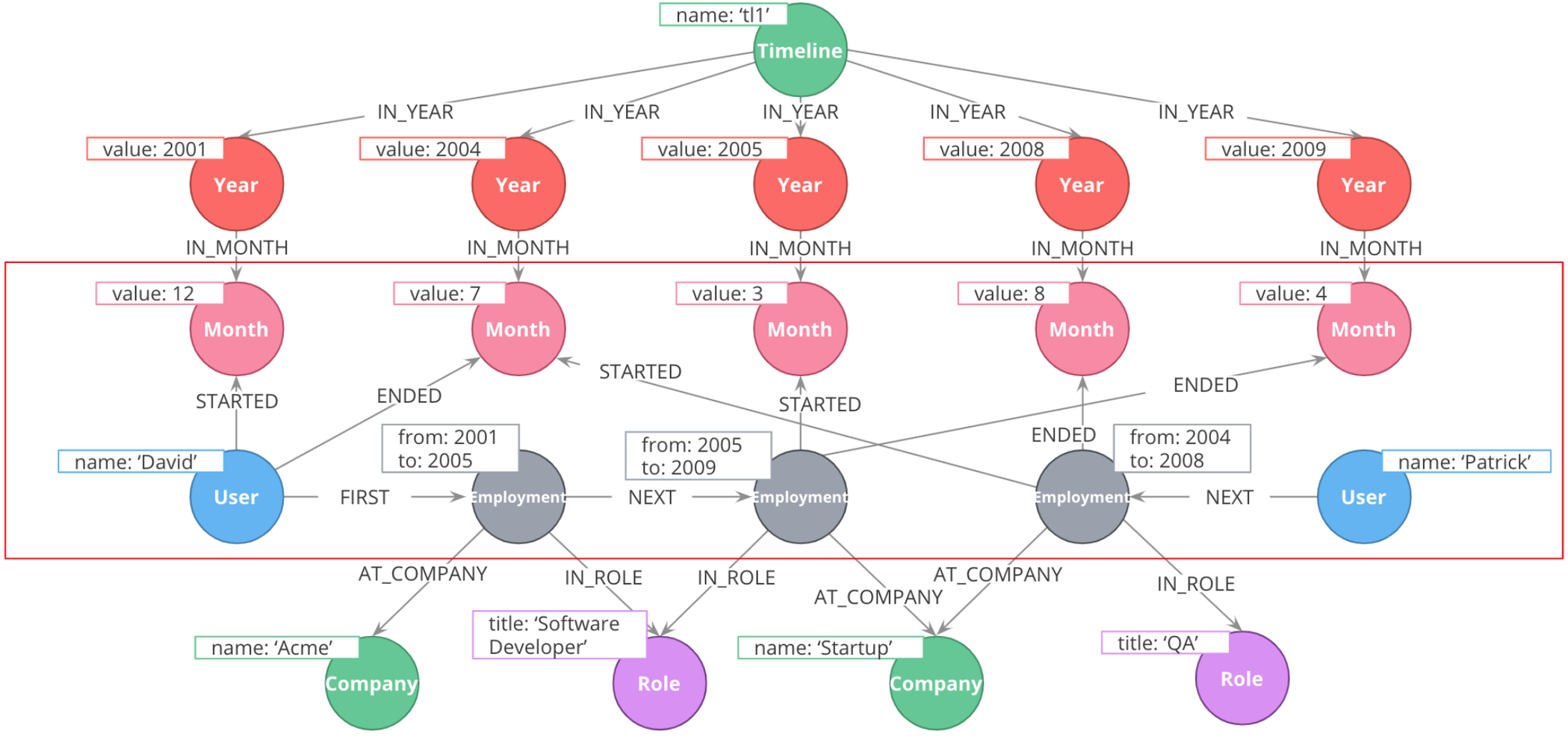
Using the timeline tree

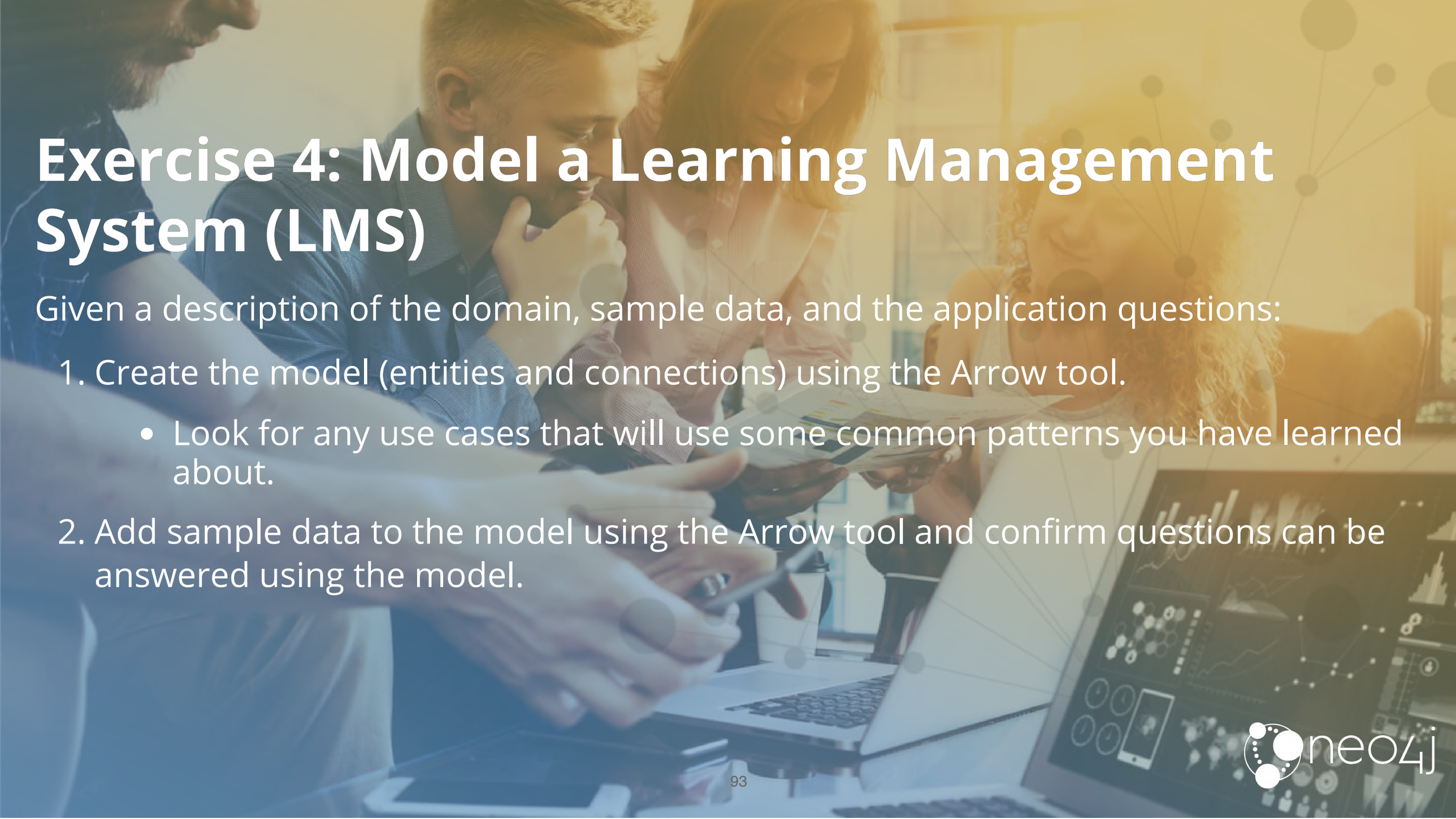


Using intermediate nodes



Using linked list





Exercise 4: Model a Learning Management System (LMS)

Given a description of the domain, sample data, and the application questions:

1. Create the model (entities and connections) using the Arrow tool.
    - Look for any use cases that will use some common patterns you have learned about.
2. Add sample data to the model using the Arrow tool and confirm questions can be answered using the model.

Exercise 4 instructions: The domain

- There are many courses in the LMS, each of which contains a number of lessons that must be completed in a specific order.
- Every course grants a certificate upon completion.
- This certificate has a term of validity. When it expires, students must take the course again.
- Students can enroll in as many simultaneous courses as they want to.
- When a student logs in and chooses a course, the LMS must send them to their latest unfinished lesson.

Exercise 4 instructions: Sample data

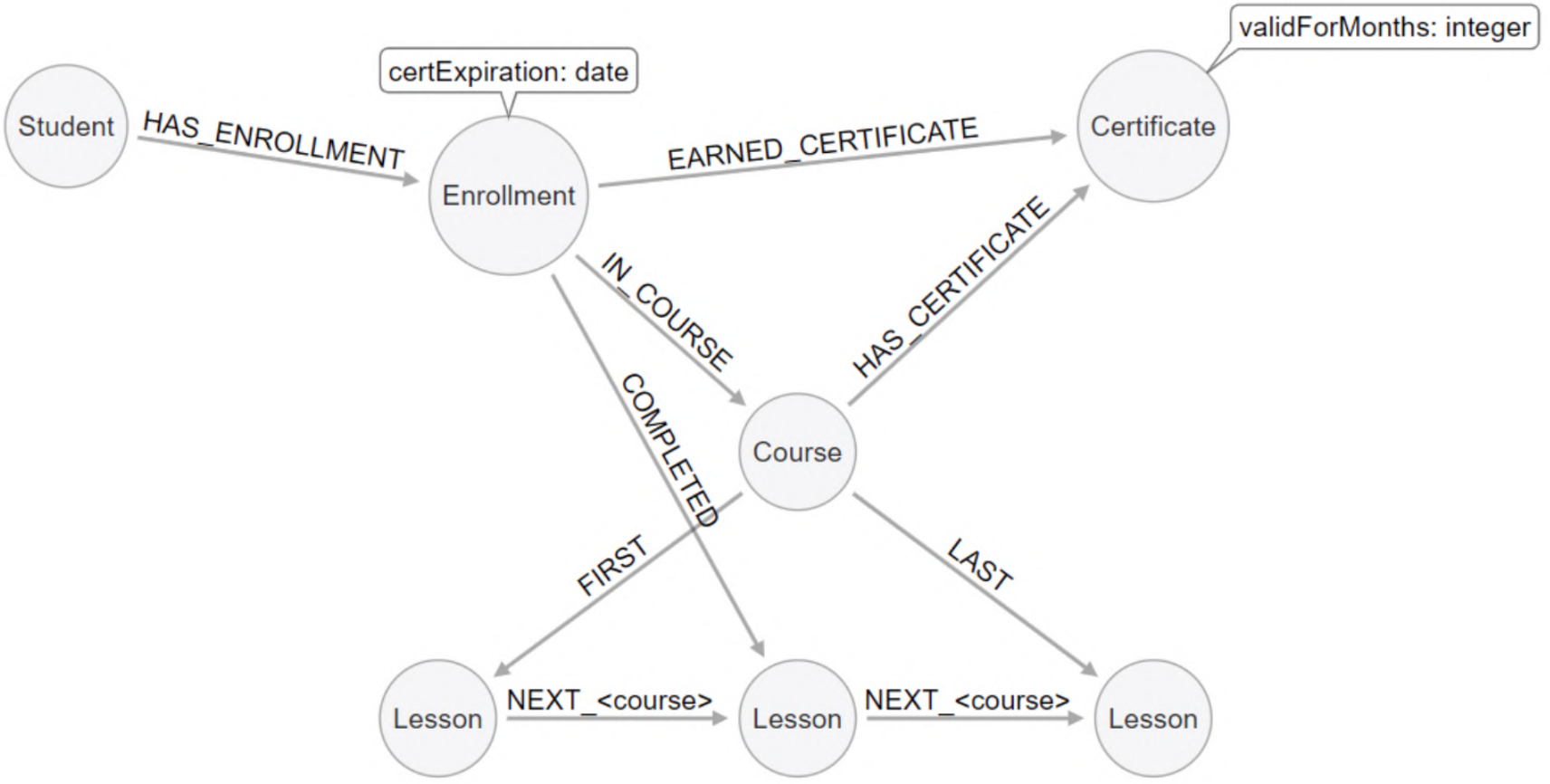
Courses	Lessons	Certificate
Introduction to Neo4j	Graph Theory, Graph Databases, Basic Cypher	2-year validity
Neo4j for Developers	Graph Theory, Property Graph, Graph Databases	6-month validity

Students	Completed Courses	In-Progress Courses
Alice	Introduction to Neo4j (2016), Introduction to Neo4j (2018)	Introduction to Neo4j (lesson 1)
Dan		Introduction to Neo4j (lesson 3), Neo4j for Developers (lesson 2)

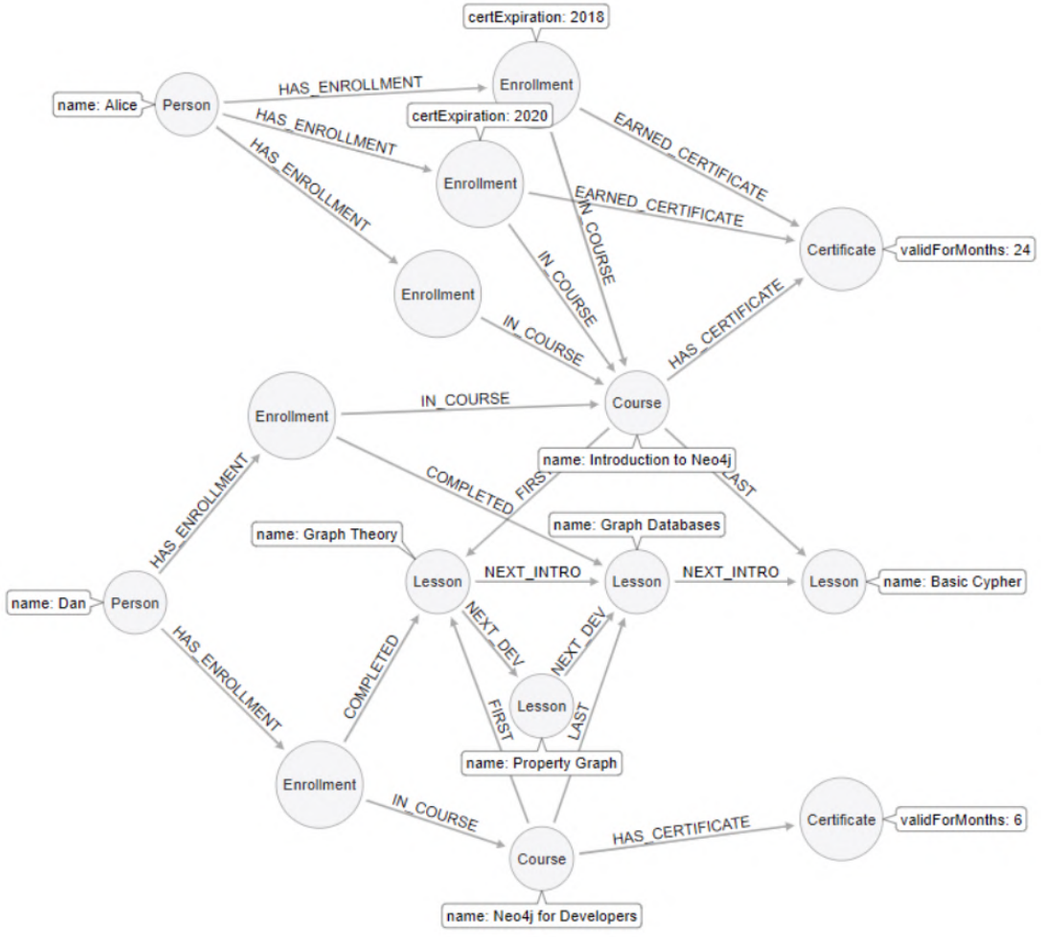
Exercise 4 instructions: Application questions

1. Which lesson(s) is Dan currently working on?
2. What are Alice's current certifications?
3. Which lessons are in the Neo4j for Developers course?
4. What is the last lesson in the Introduction to Neo4j course?
5. Which lesson follows Graph Theory in the Neo4j for Developers course?
6. Who has completed Introduction to Neo4j?

Exercise 4 solution: Application model



Exercise 4 solution: Sample data



1. Which lesson(s) is Dan currently working on?
2. What are Alice's current certifications?
3. Which lessons are in the **Neo4j for Developers** course?
4. What is the last lesson in the **Introduction to Neo4j** course?
5. Which lesson follows **Graph Theory** in the **Neo4j for Developers** course?
6. Who has completed **Introduction to Neo4j**?

Check your understanding

Question 1

What graph modeling pattern can you use to add more contextual and organizational information to relationships that is not easily modeled with simple relationship properties?

Select the correct answer.

- [ ] Sub-relationships.
- [ ] Linked list.
- [ ] Timeline tree.
- [ ] Intermediate nodes.

Question 1

What graph modeling pattern can you use to add more contextual and organizational information to relationships that is not easily modeled with simple relationship properties?

Select the correct answer.

- ☐ Sub-relationships.
- ☐ Linked list.
- ☐ Timeline tree.
- ☑ **Intermediate nodes.**

Question 2

In Neo4j, which structure is not recommended as a best practice for representing linked lists?

Select the correct answer.

- [ ] Interleaved linked lists with multiple starting points for navigation.
- [ ] Doubly linked lists.
- [ ] Single linked list.
- [ ] Linked list with nodes that point to the head and tail of the list.

Question 2

In Neo4j, which structure is not recommended as a best practice for representing linked lists?

Select the correct answer.

- [ ] Interleaved linked lists with multiple starting points for navigation.
- [x] **Doubly linked lists.**
- [ ] Single linked list.
- [ ] Linked list with nodes that point to the head and tail of the list.

Question 3

What common structure is used to model data where you need to know when an event occurred in the application and find other events that occurred in the same time-frame?

Select the correct answer.

- [ ] Linked list where each node has a timestamp.
- [ ] Intermediate nodes where the intermediate nodes have time-related data.
- [ ] Timeline tree
- [ ] Doubly-linked list where the symmetric relationbship is the timestamp relationship.

Question 3

What common structure is used to model data where you need to know when an event occurred in the application and find other events that occurred in the same time-frame?

Select the correct answer.

- [ ] Linked list where each node has a timestamp.
- [ ] Intermediate nodes where the intermediate nodes have time-related data.
- [x] **Timeline tree**
- [ ] Doubly-linked list where the symmetric relationbship is the timestamp relationship.

Summary

You should now be able to:

- Describe common graph structures used in modeling:
  - Intermediate nodes
  - Linked lists
  - Timeline trees
  - Multiple structures in a single graph

Refactoring and Evolving a Model

neo4j

About this module

At the end of this module, you should be able to:

- Describe why you would refactor a graph data model.
- Refactor a model to:
  - Eliminate duplicate data in nodes.
  - Use node labels rather than properties.
  - Extract property values to create nodes.

What is refactoring?

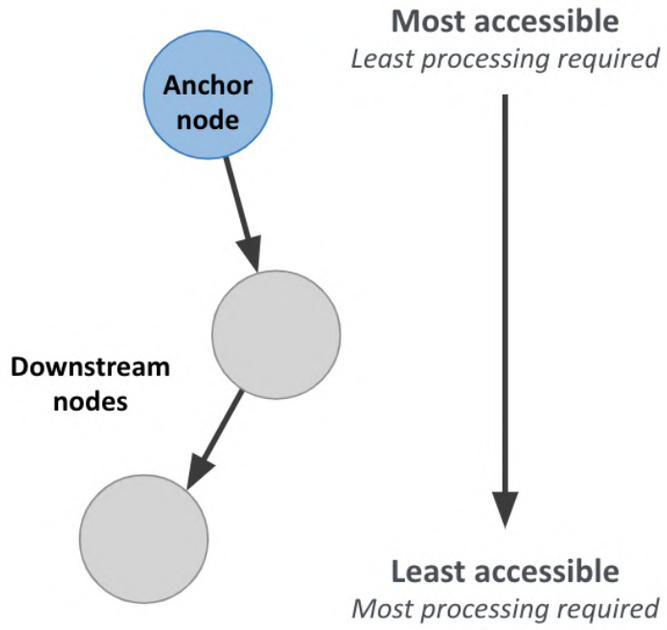
Refactoring is the process of changing the data structure without altering its semantic meaning.

- Most of the time, refactoring simply involves moving data from one structure to another within the graph.
- In some cases, refactoring involves adding more data from other sources.

The most common type of refactoring of a graph is to use a property value to restructure the graph. That is, a property value is used to create a label, a node, or a relationship.

Review: Hierarchy of Accessibility

For each data object, how much work must Neo4j do to evaluate if this is a "good" path or a "bad" one?



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

neo4j

Why refactor?

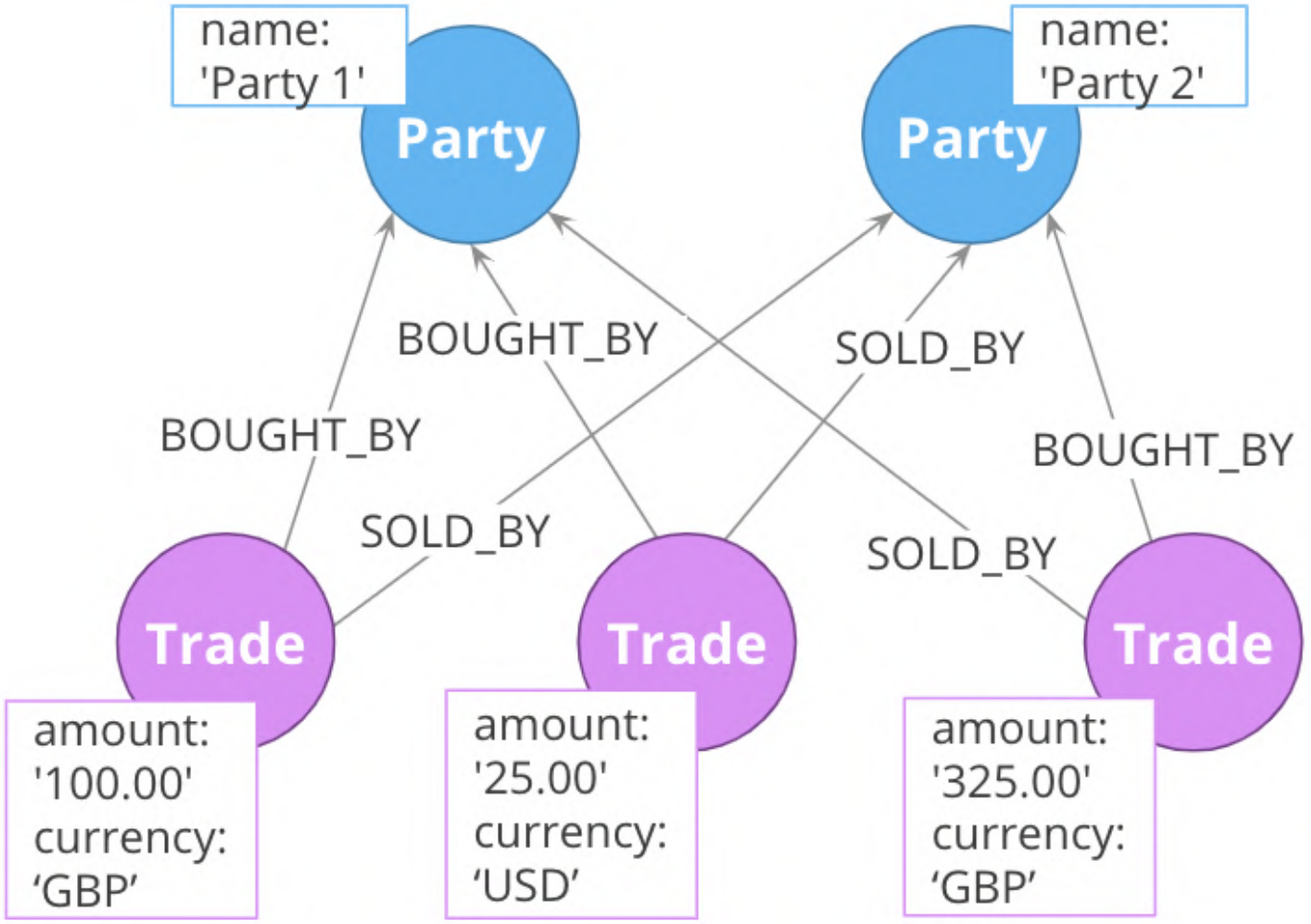
Data models can be optimized for one of four things:

- Query performance
- Model simplicity & intuitiveness
- Query simplicity (i.e., simpler Cypher strings)
- Easy data updates

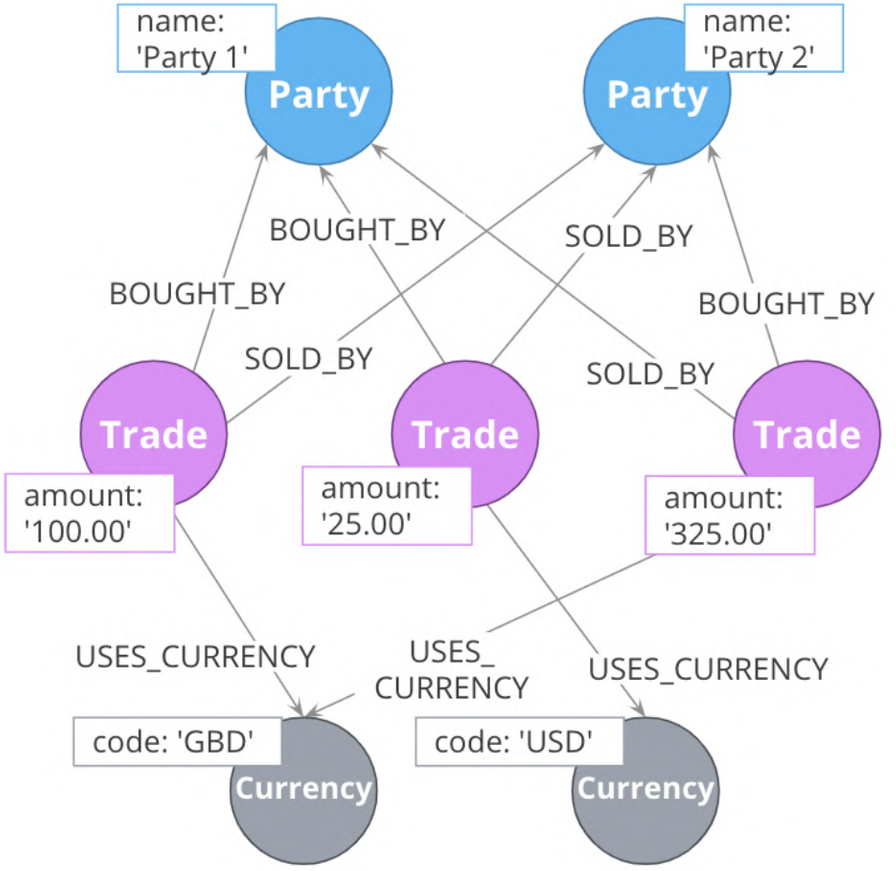
> Note Improving behavior in one of these areas frequently involves sacrifices in others.

Another **important** reason to refactor is to accomodate new application questions in the same model.

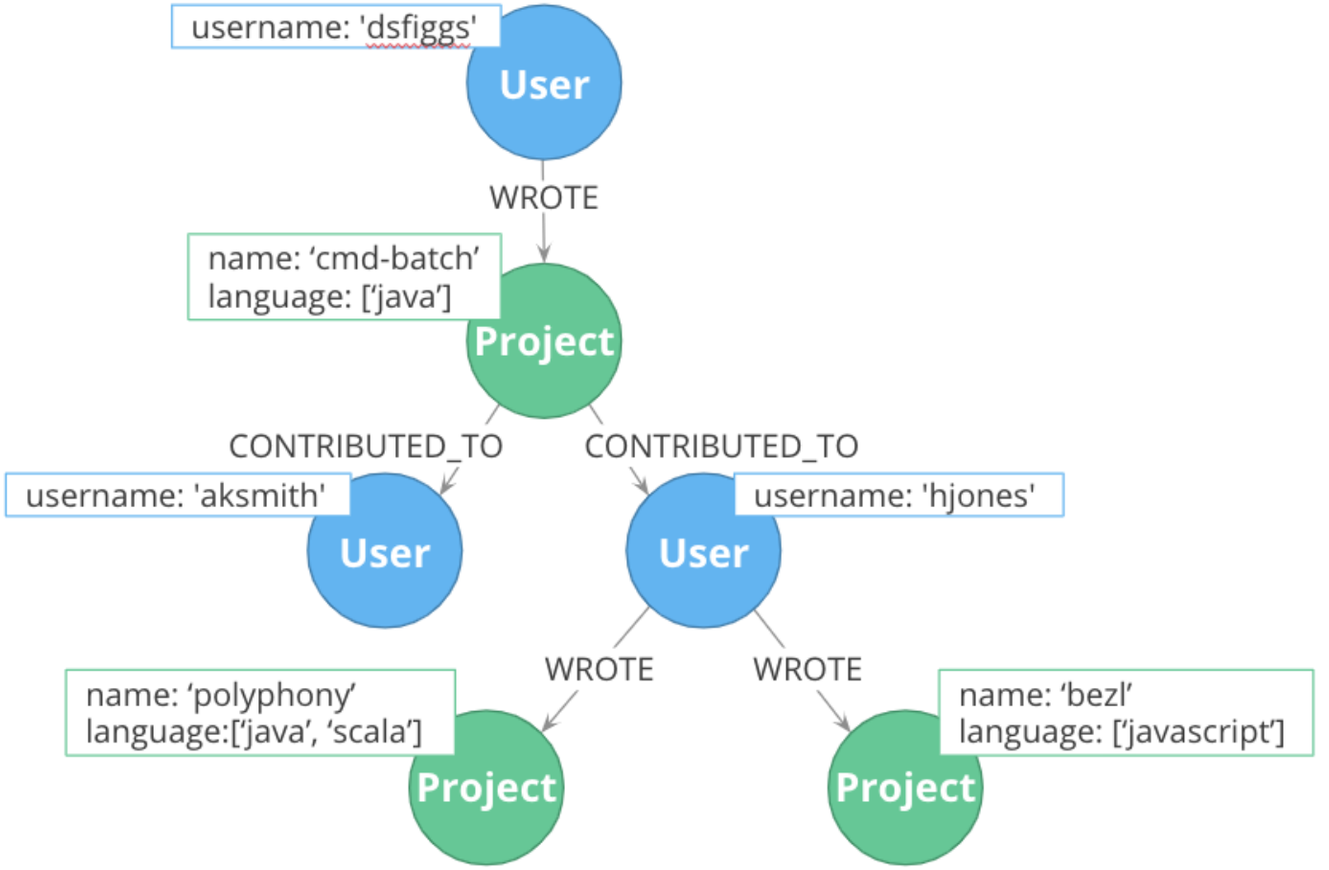
Goal: Eliminate duplicate data in properties



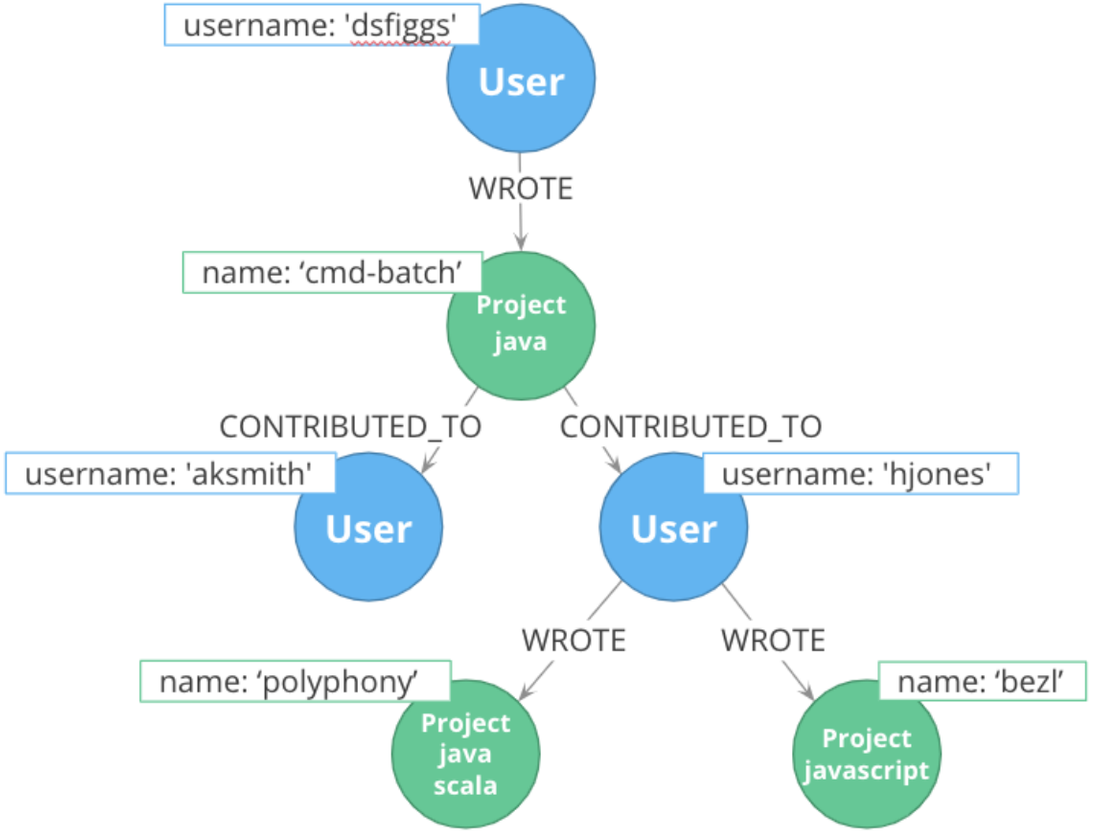
Refactor example: Extracting nodes from properties



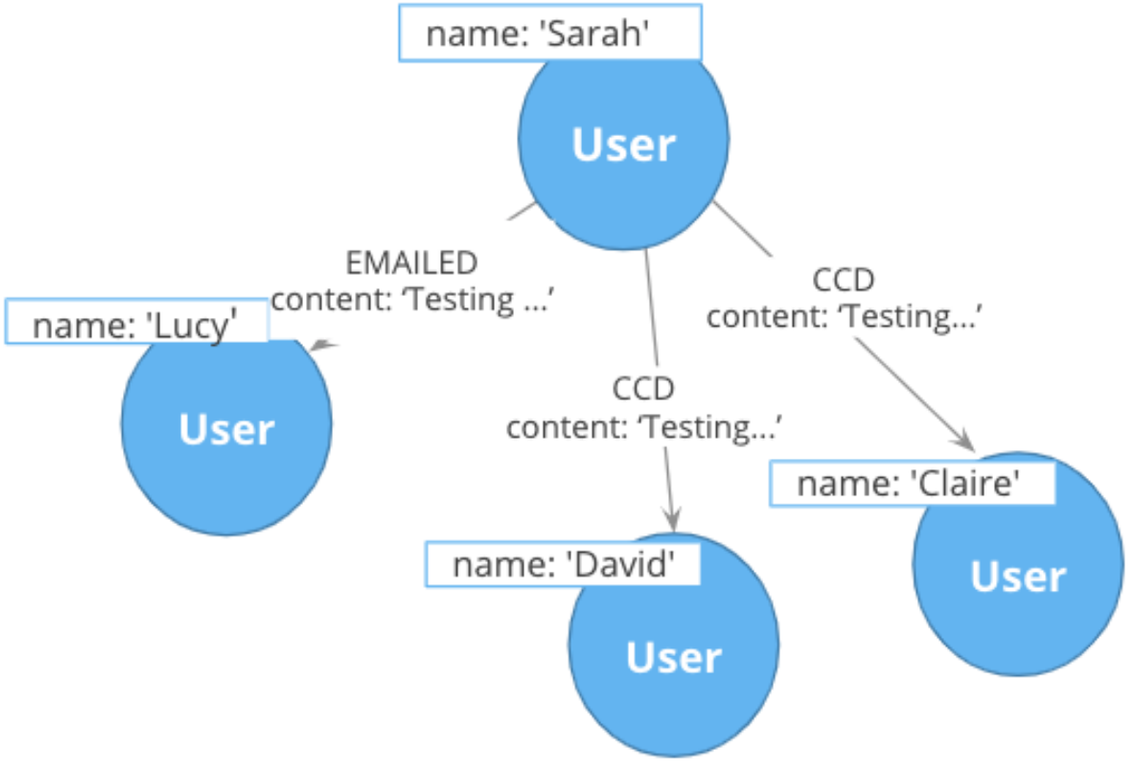
Goal: Use labels instead of property values



Refactor example: Turn property values into labels for nodes

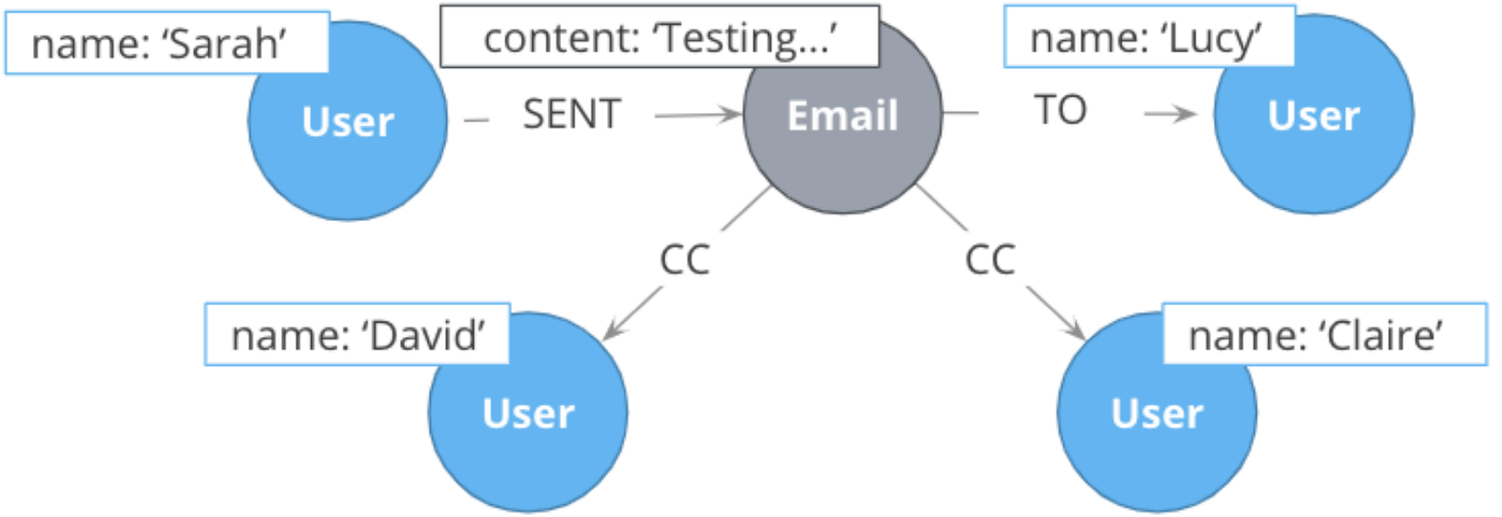


Goal: Use nodes instead of properties for relationships

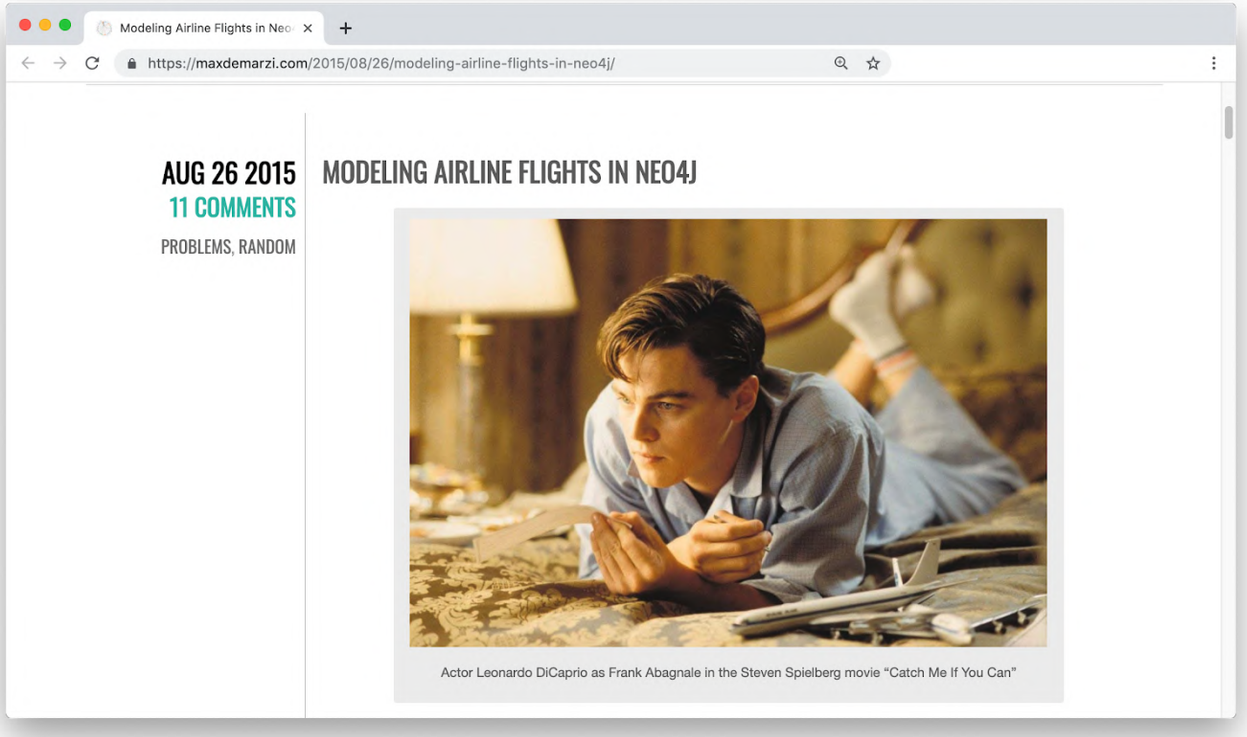


neo4j

Refactor: Extract nodes from relationship properties



Refactoring example: Modeling airline flights



Credit: Max De Marzi https://maxdemarzi.com/2015/08/26/modeling-airline-flights-in-neo4j/

Initial question for our model

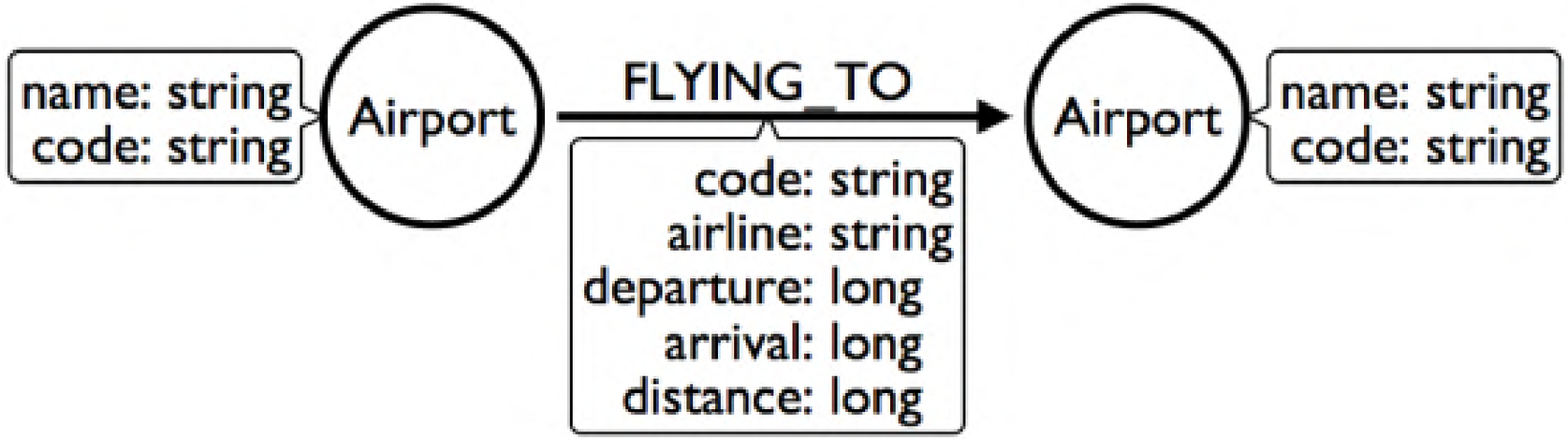
Question: What flights will take me from Malmo to New York on Friday?

Ask yourself:

- What are the entities?
- What are the connections between the entities?
- What properties do we need?

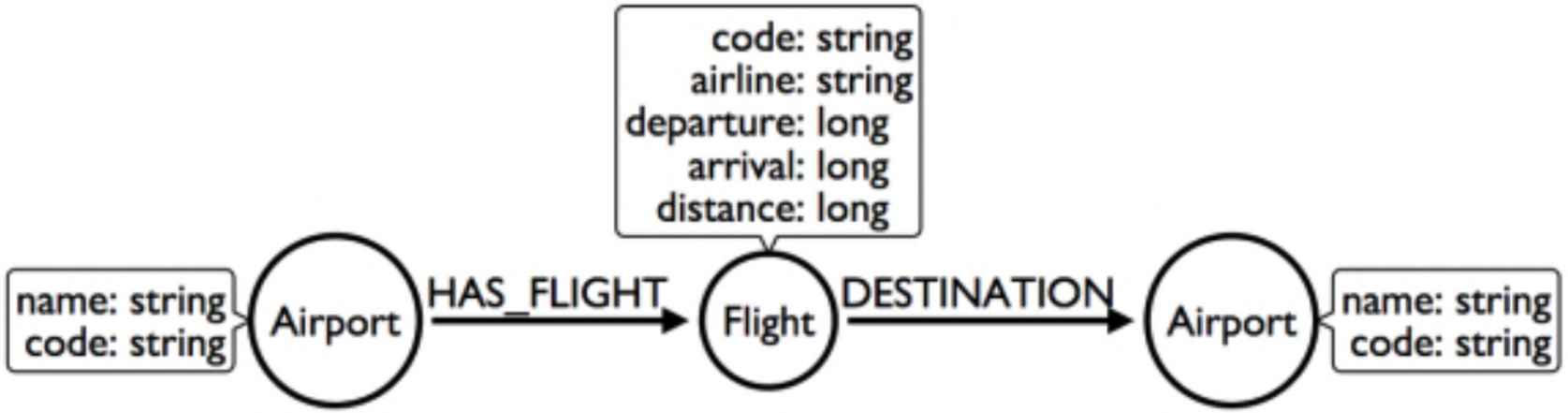
Initial model

Question: What flights will take me from Malmo to New York on Friday?



New Question: Mom is on flight AY189. When will she land?

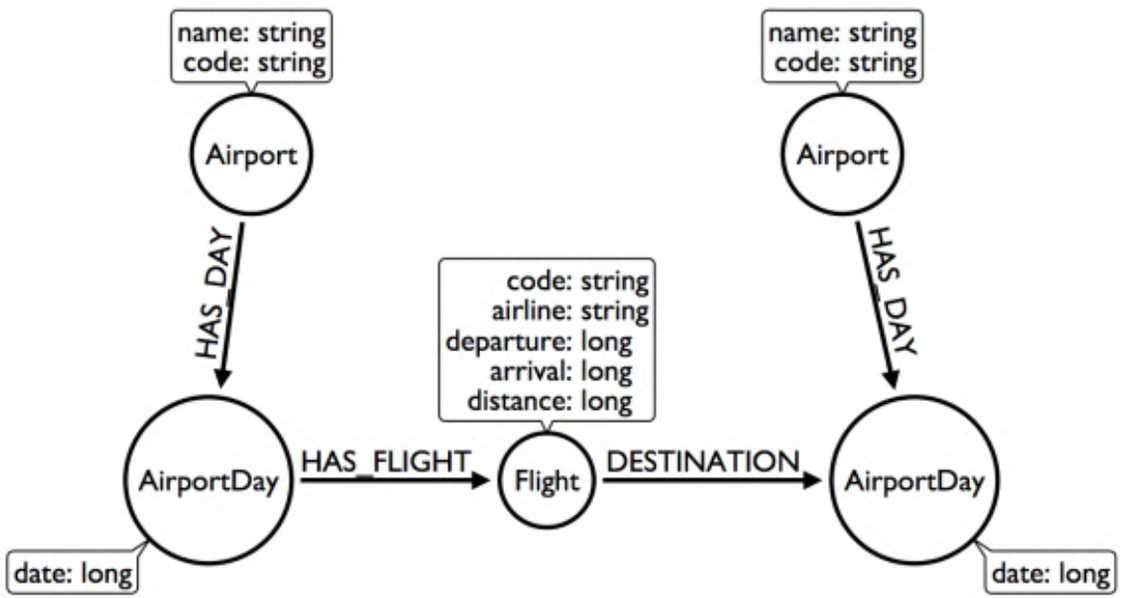
Refactor: Create intermediate Flight nodes



Question 1: What flights will take me from Malmo to New York on Friday?

Question 2: Mom is on flight AY189. When will she land?

Refactor: Create AirportDay intermediate nodes

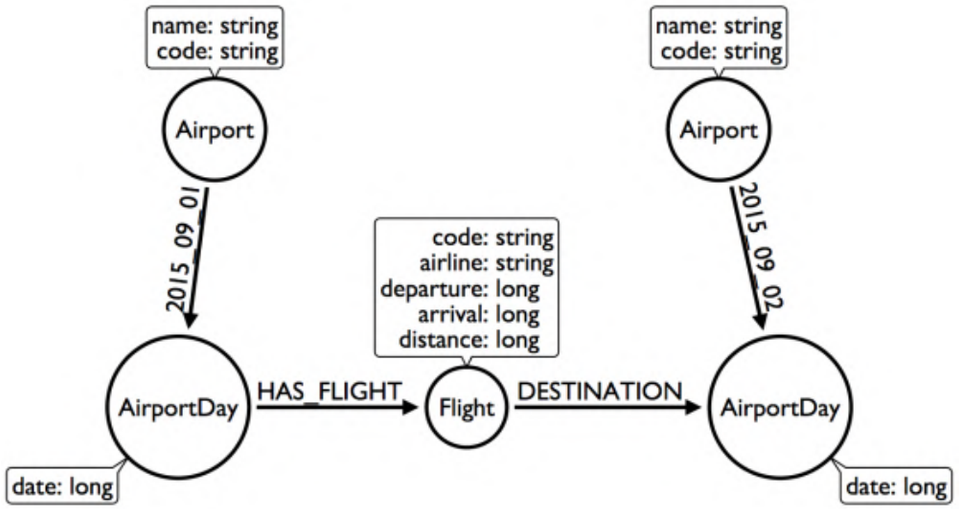


Question 1: What flights will take me from Malmo to New York on Friday?

Question 2: Mom is on flight AY189. When will she land?

neo4j

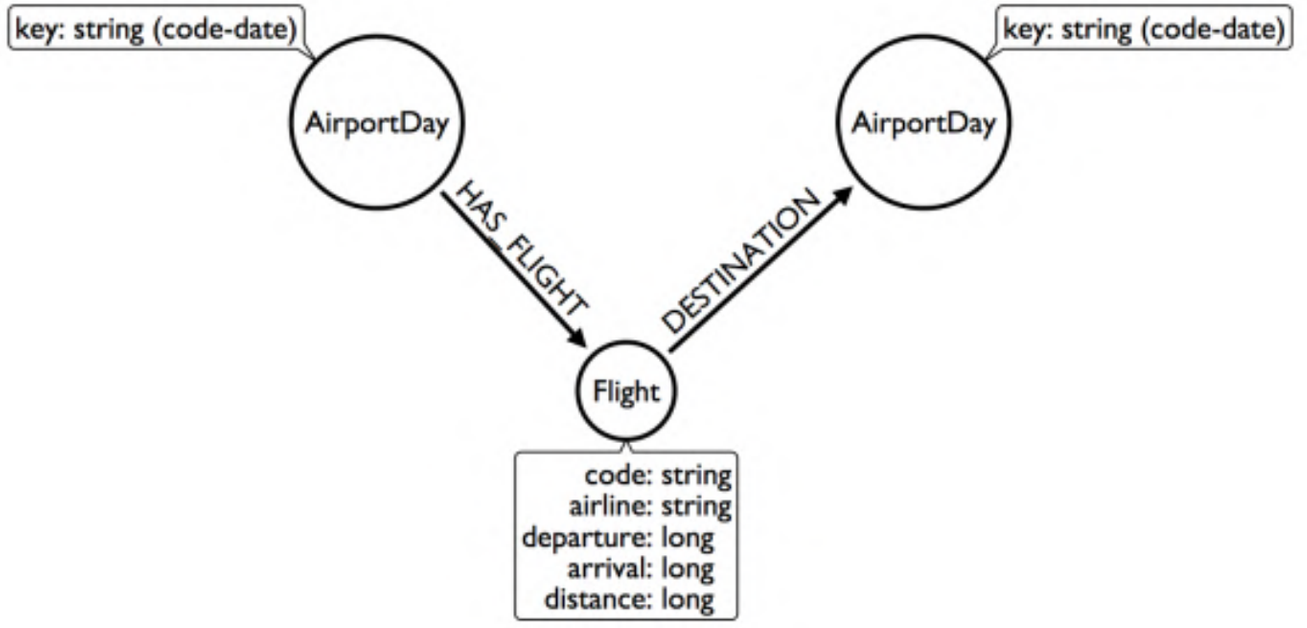
Possible refactor: Change relationship type to date



Question 1: What flights will take me from Malmo to New York on Friday?

Question 2: Mom is on flight AY189. When will she land?

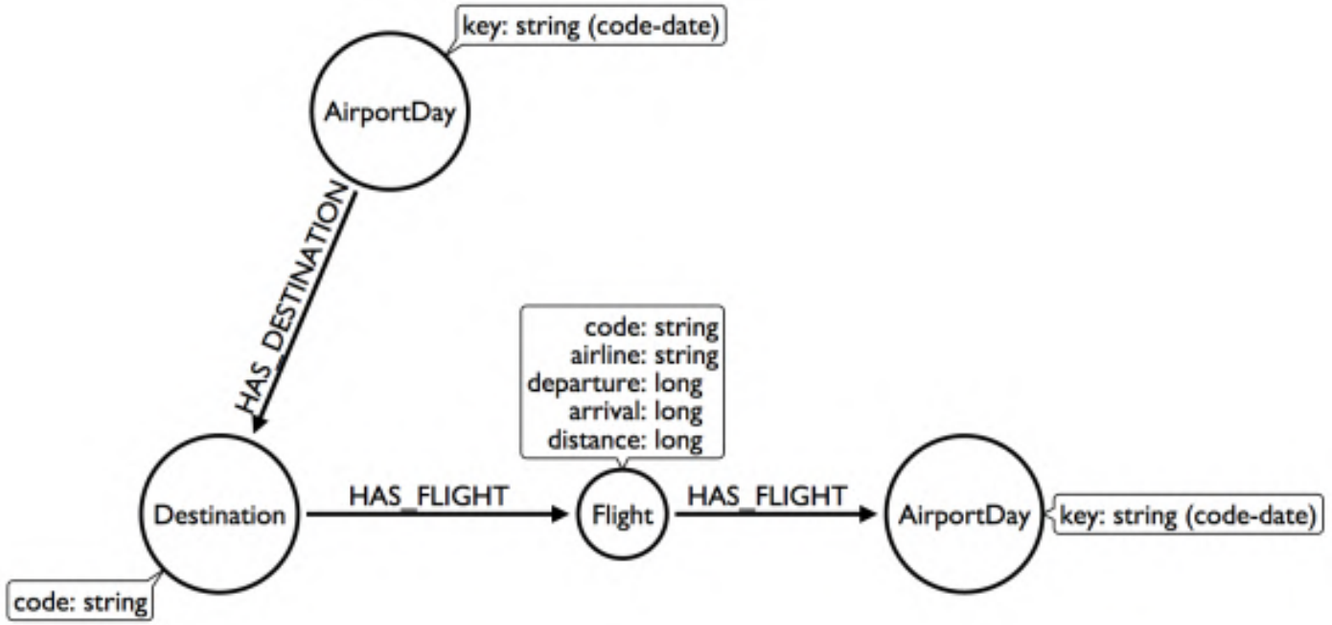
Possible refactor: Remove Airport nodes



Question 1: What flights will take me from Malmo to New York on Friday?

Question 2: Mom is on flight AY189. When will she land?

Refactor: Add Destination intermediate nodes



Question 1: What flights will take me from Malmo to New York on Friday?

Question 2: Mom is on flight AY189. When will she land?

Check your understanding

Question 1

What tasks can be done during the refactoring of a graph data model?

Select the correct answers.

- [ ] Data is moved from one structure in the existing graph to another.
- [ ] A new graph is created from an existing graph.
- [ ] Statistics are collected about the numbers of nodes, properties, and relationships.
- [ ] Data may be added to the graph from other sources.

Question 1

What tasks can be done during the refactoring of a graph data model?

Select the correct answers.

- ✔ **Data is moved from one structure in the existing graph to another.**
- ☐ A new graph is created from an existing graph.
- ☐ Statistics are collected about the numbers of nodes, properties, and relationships.
- ✔ **Data may be added to the graph from other sources.**

Question 2

Why do you refactor a graph data model?

Select the correct answers.

- [ ] Improve query performance.
- [ ] Simplify the model to make it more intuitive.
- [ ] Allow for simpler Cypher queries.
- [ ] Make updates to the data in the graph easier.

Question 2

Why do you refactor a graph data model?

Select the correct answers.

- ✔ **Improve query performance.**
- ✔ **Simplify the model to make it more intuitive.**
- ✔ **Allow for simpler Cypher queries.**
- ✔ **Make updates to the data in the graph easier.**

Question 3

When thinking about refactoring a graph data model. What is the most common type of refactoring you typically do?

Select the correct answer.

- [ ] Rename node labels.
- [ ] Duplicate property values where they will be queried most.
- [ ] Extract property values to change the structure of the graph.
- [ ] Create indexes that will speed up queries for the most important questions.

Question 3

When thinking about refactoring a graph data model. What is the most common type of refactoring you typically do?

Select the correct answer.

- ☐ Rename node labels.
- ☐ Duplicate property values where they will be queried most.
- ☑ **Extract property values to change the structure of the graph.**
- ☐ Create indexes that will speed up queries for the most important questions.

Summary

You should now be able to:

- Describe why you would refactor a graph data model.
- Refactor a model to:
  - Eliminate duplicate data in nodes.
  - Use node labels rather than properties.
  - Extract property values to create nodes.

Graph Data Modeling for Neo4j: Summary

Course Summary

In this course, you have learned how to:

- Describe what a graph data model is for a Neo4j database.
- Design an initial graph data model using Neo4j best practices.
- Describe the core principles used for Neo4j graph data modeling.
- Describe the common structures used in a Neo4j graph data model.
- Refactor and evolve a graph data model.

Resources - 1

There are many resources available to you for learning more about Neo4j and Cypher.

https://neo4j.com/developer/resources/

Neo4j Community Site where you can ask or answer questions about Neo4j and discuss with other users:

https://community.neo4j.com

Neo4j documentation:

https://neo4j.com/docs/

Resources - 2

Neo4j Sandboxes for experimenting with graphs:

https://sandbox.neo4j.com

Videos on the Neo4j YouTube channel:

https://www.youtube.com/channel/UCvze3hU6OZBkB1vkhH2lH9Q

Neo4j online and classroom training:

https://neo4j.com/graphacademy/

Resources - 3

Become a Neo4j certified developer:

https://neo4j.com/graphacademy/neo4j-certification/

> Note You can take the certification exam multiple times until you pass!

GitHub repository:

https://github.com/neo4j-contrib

Neo4j events all over the world:

https://neo4j.com/events/world/all/

Resources - 4

Graph Gists for learning more use cases for Neo4j:

https://neo4j.com/graphgists/

Attend a Neo4j meetup:

https://www.meetup.com/topics/neo4j/

View questions/answers raised about Neo4j:

https://stackoverflow.com/tags/neo4j/hot