Knowledge Graphs

How to create a Knowledge Graph?

Outline

- Overview
- Design of an RDF Graph
- Design of a Property Graph
- Summary

Overview

- Creating a Knowledge Graph
  - Design of Schema
  - Populating the knowledge graph

Overview

- Creating a Knowledge Graph
  - Design of Schema
    - Strictly speaking not required to get started
    - Design improves the usefulness
  - Populating the knowledge graph

Overview

- Creating a Knowledge Graph
  - Design of Schema
    - Strictly speaking not required to get started
    - Design improves the usefulness
  - Populating the knowledge graph
    - Structured and semi-structured sources
    - Text
    - Curation

Outline

- Overview
- Design of an RDF Graph
- Design of a Property Graph
- Summary

Design of an RDF Graph

- Linked Data Principles
  - Use URIs as names of things
  - Use HTTP URIs so that people can look up those names
  - Whenever someone looks up a URI provide useful information using standards RDF and SPARQL
  - Include links to other things so that people can discover new things

Use URIs as name of things

- Identify the items of interest (People, Places, Product, Gene, …)
  - Referred to as resources
    - Information resources
      - http://www.Wikipedia.Org
    - Non information resources
      - Person: http://biglynx.co.uk/people/dave-smith

Use URIs as name of things

- Identify the items of interest (People, Places, Product, Gene, …)
  - Referred to as resources
    - Information resources
      - http://www.Wikipedia.Org
    - Non information resources
      - Person: http://biglynx.co.uk/people/dave-smith
- Keep them short and pneumonic
- Ensure persistence

Use HTTP URIs

- Use HTTP URIs so that people can lookup things
  - Do not use
    - Digital Object Identifier: doi:10.1038/nphys1170
  - Prefer:
    - https://doi.org/10.1038/nphys1170

Use HTTP URIs

- Use HTTP URIs so that people can lookup things
  - Looks up is also called Dereferencing
    - When we look up an information object
      - We get a representation of its current state
    - When we look up a non-information object
      - We get a set of RDF facts about it

Provide useful Information in RDF / SPARQL

- Dereferencing of a non-information object returns RDF data
  - Use standardized vocabularies
    - Catalogs, Organizations, Multi-dimensional data
    - Schema.Org

Provide useful Information in RDF / SPARQL

- Dereferencing of a non-information object returns RDF data

```
@prefix uk_cabinet: <http://reference.data.gov.uk/id/department/>
uk_cabinet:co rdf:type org:Organization
uk_cabinet:co skos:prefLabel "Cabinet Office"
uk_cabinet:co org:hasUnit uk_cabinet:cabinet-office-communications
uk_cabinet:cabinet-office-communications rdf:type org:OrganizationUnit
uk_cabinet:cabinet-office-communications skos:prefLabel "Cabinet Office Communications"
uk_cabinet:cabinet-office-communications org:hasPost uk_cabinet:post_246
uk_cabinet:post_246 skos:prefLabel "Deputy Director, Deputy Prime Minister's Spokesperson"
```

SKOS: Simple Knowledge Organization System
ORG: Organization

Provide useful Information in RDF / SPARQL

- Dereferencing of a non-information object returns RDF data

```
@prefix uk_cabinet: <http://reference.data.gov.uk/id/department/>
uk_cabinet:co rdf:type org:Organization
uk_cabinet:co skos:prefLabel "Cabinet Office"
uk_cabinet:co org:hasUnit uk_cabinet:cabinet-office-communications
uk_cabinet:cabinet-office-communications rdf:type org:OrganizationUnit
uk_cabinet:cabinet-office-communications skos:prefLabel "Cabinet Office Communications"
uk_cabinet:cabinet-office-communications org:hasPost uk_cabinet:post_246
uk_cabinet:post_246 skos:prefLabel "Deputy Director, Deputy Prime Minister's Spokesperson"
```

SKOS: Simple Knowledge Organization System
ORG: Organization

Provide useful Information in RDF / SPARQL

- Dereferencing of a non-information object returns RDF data

```
@prefix uk_cabinet: <http://reference.data.gov.uk/id/department/>
uk_cabinet:co rdf:type org:Organization
uk_cabinet:co skos:prefLabel "Cabinet Office"
uk_cabinet:co org:hasUnit uk_cabinet:cabinet-office-communications
uk_cabinet:cabinet-office-communications rdf:type org:OrganizationUnit
uk_cabinet:cabinet-office-communications skos:prefLabel "Cabinet Office Communications"
uk_cabinet:cabinet-office-communications org:hasPost uk_cabinet:post_246
uk_cabinet:post_246 skos:prefLabel "Deputy Director, Deputy Prime Minister's Spokesperson"
```

SKOS: Simple Knowledge Organization System
ORG: Organization

Provide useful Information in RDF / SPARQL

- If creating a new vocabulary becomes necessary, ensure that
  - it is documented
  - It is self-describing
  - versioning policy
  - it is defined in multiple languages
  - it is published by a trusted sources

Provide useful Information in RDF / SPARQL

- If creating a new vocabulary becomes necessary, ensure that
  - it is documented
  - It is self-describing
    - Information about the schema is available within the data itself
  - versioning policy
  - it is defined in multiple languages
  - it is published by a trusted sources

Include Links to Other URIs

- Relationship Links
- Identity Links
- Vocabulary Links

Include Links to Other URIs

- Relationship Links
  - Relate object in one dataset to an object in another

```
@prefix big: <http://biglynx.co.uk/people/>
@prefix dbpedia: <http://dbpedia.org/resource/>
big:dave-smith foaf:based_near dbpedia:Birmingham
```

Include Links to Other URIs

- Identity Links
  - Equate objects in one dataset to objects in another dataset

```
@prefix ds: <http://www.dave-smith.eg.uk>
@prefix owl: <http://www.w3.org/2002/07/owl>
@prefix big: <http://biglynx.co.uk/people/>
ds:me owl:sameAs big:dave-smith
```

Include Links to Other URIs

- Vocabulary Links
  - Links from the data to the definition of terms

@prefix big: <http://biglynx.co.uk/people/>

@prefix dbpedia: <http://dbpedia.org/ontology/>

big:SmallMediumEnterprise rdfs:subClassOf dbpedia:Company

Design of an RDF Graph

- Linked Data Principles
  - Use URIs as names of things
  - Use HTTP URIs so that people can look up those names
  - Whenever someone looks up a URI provide useful information using standards RDF and SPARQL
  - Include links to other things so that people can discover new things

Outline

- Overview
- Design of an RDF Graph
- Design of a Property Graph
- Summary

Design of a Property Graph

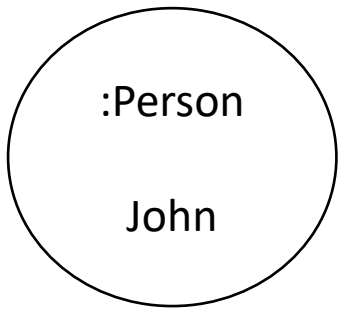
- Choosing nodes, labels and properties
- When to introduce relationships
- When to introduce relationship properties
- How to handle non-binary relationships

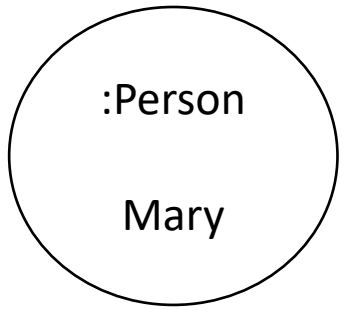
Choosing Nodes, Labels and Properties

- Nodes usually represent entities in a domain

Choosing Nodes, Labels and Properties

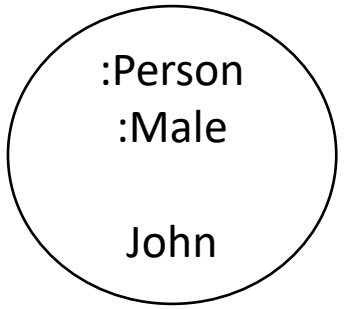
- Nodes usually represent entities in a domain
  - How to represent gender?

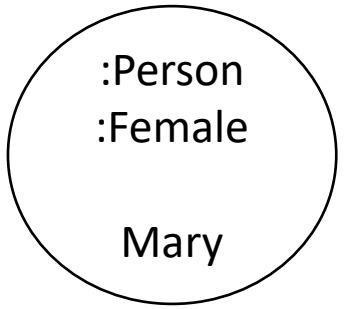




Choosing Nodes, Labels and Properties

- Nodes usually represent entities in a domain
  - How to represent gender?





Introduce a new Label

Labels are like classes

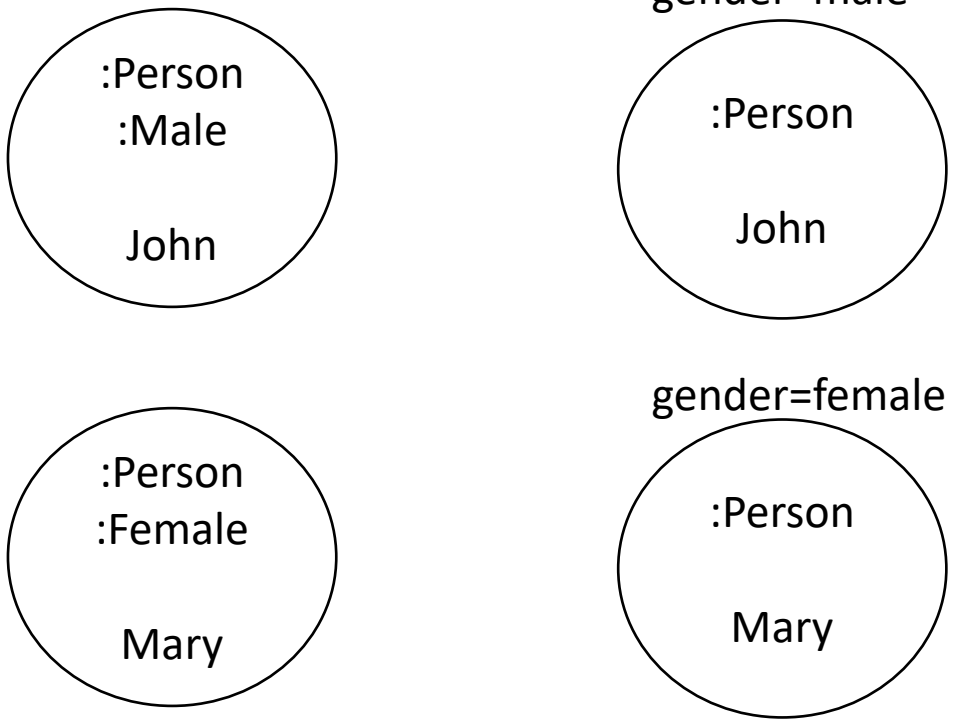
Labels should be natural

Labels should not change with time

Could benefit from indexing

Choosing Nodes, Labels and Properties

- Nodes usually represent entities in a domain
  - How to represent gender?

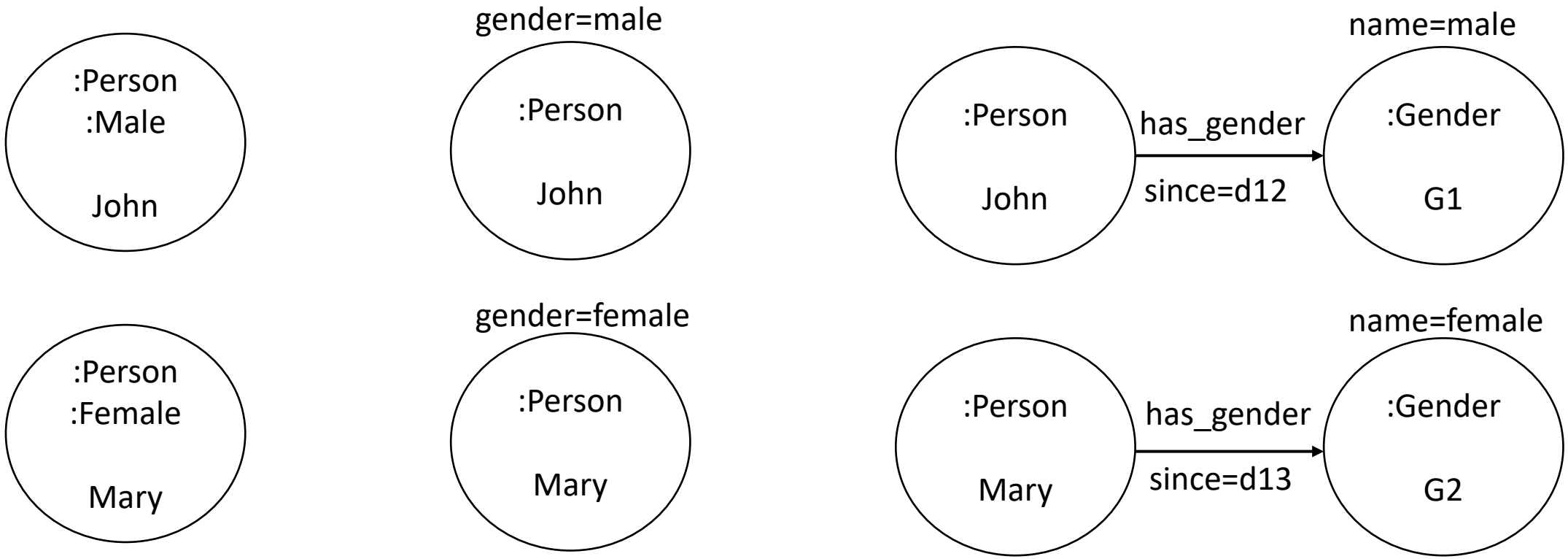


Introduce a new node property

Property should not change with time

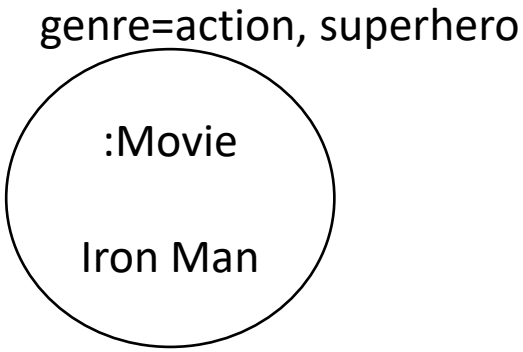
Choosing Nodes, Labels and Properties

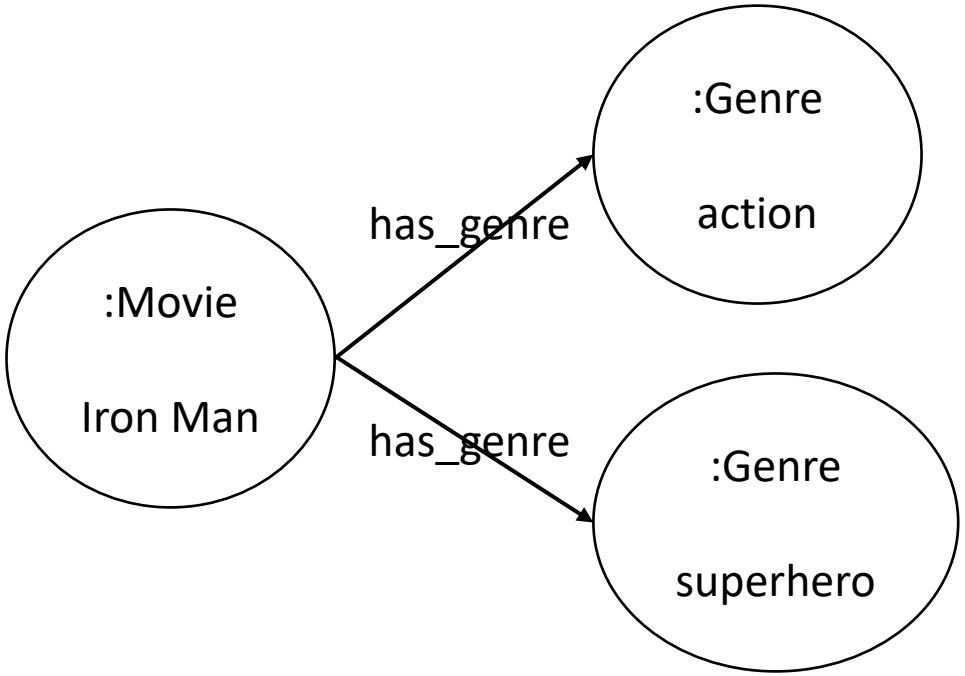
- Nodes usually represent entities in a domain
  - How to represent gender?



When to introduce a Relationship?

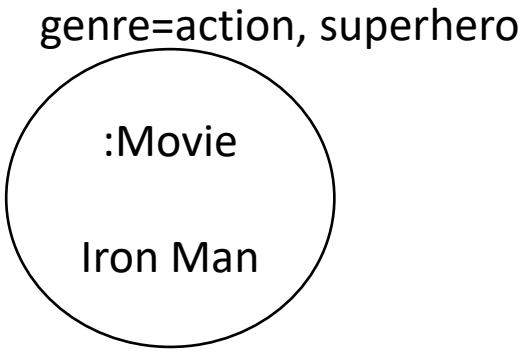
- When efficient access is required

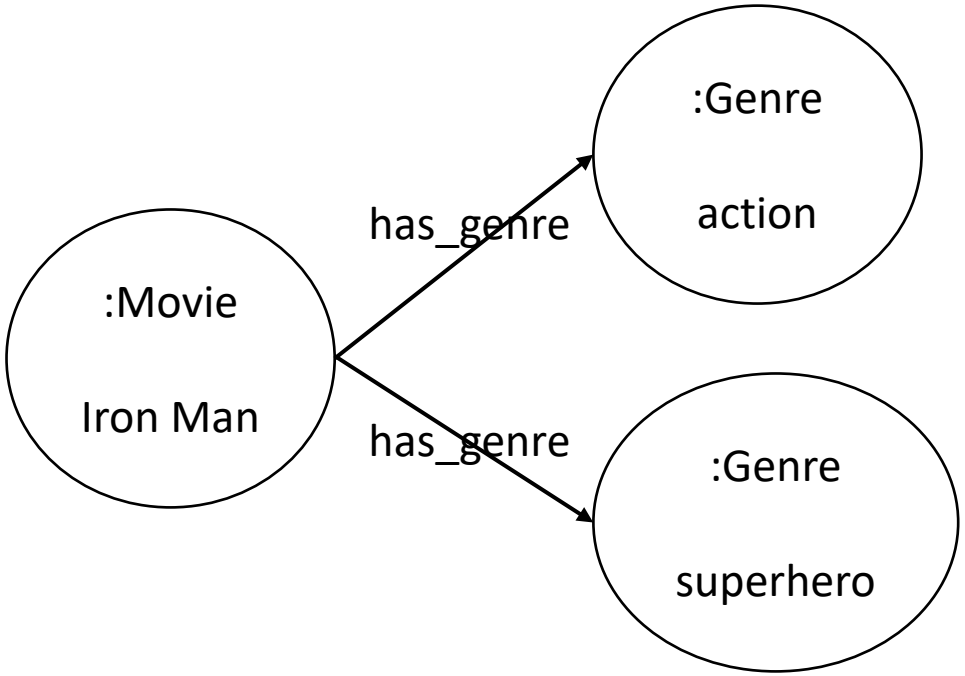




When to introduce a Relationship?

- When efficient access is required



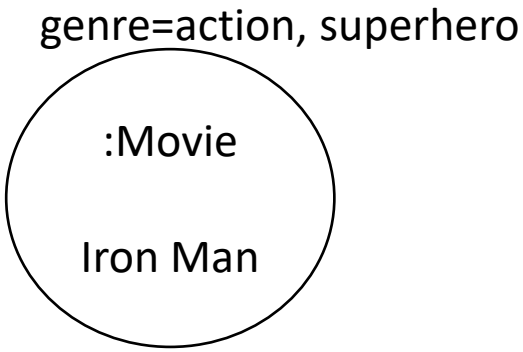


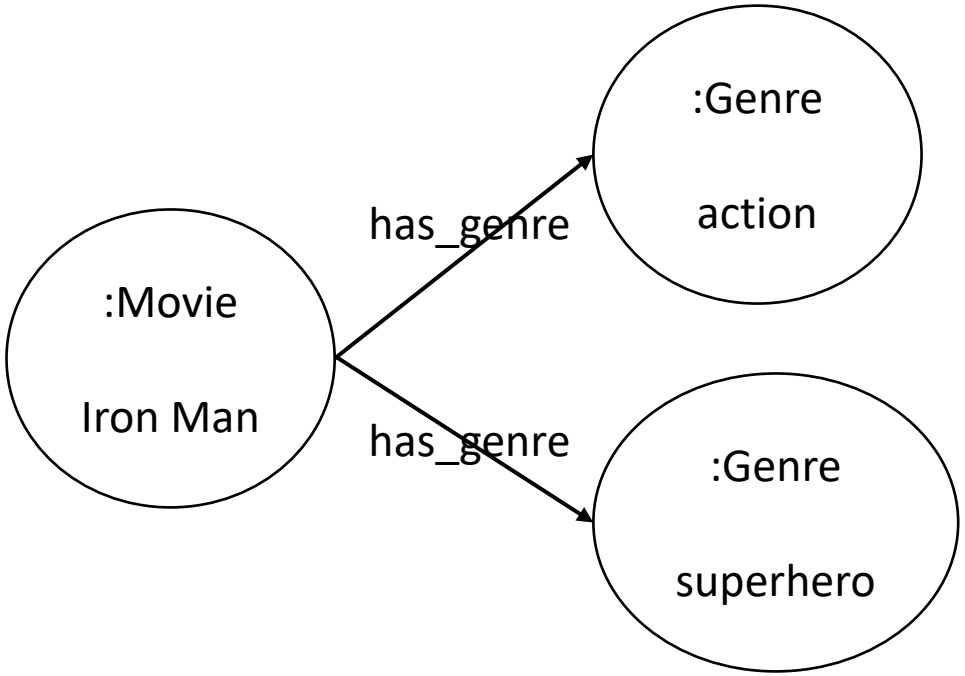
Find movies that have genres in common

```
MATCH (m1:Movie), (m2:Movie)
WHERE any(x IN m1.genre WHERE x IN m2.genre)
AND m1 <> m2
RETURN m1, m2
```

When to introduce a Relationship?

- When efficient access is required





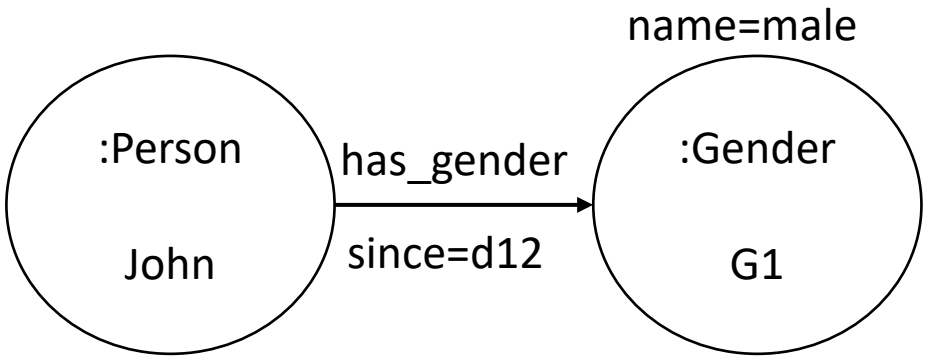
Find movies that have genres in common

```
MATCH (m1:Movie), (m2:Movie)
WHERE any(x IN m1.genre WHERE x IN m2.genre)
AND m1 <> m2
RETURN m1, m2
```

```
MATCH (m1:Movie)-[:has_genre]->(g:Genre),
      (m2:Movie)-[:has_genre]->(g)
WHERE m1 <> m2
RETURN m1, m2
```

When to Introduce a Relationship Property

- Common use cases for relationship properties
  - Time varying relationships
  - Provenance
  - Confidence

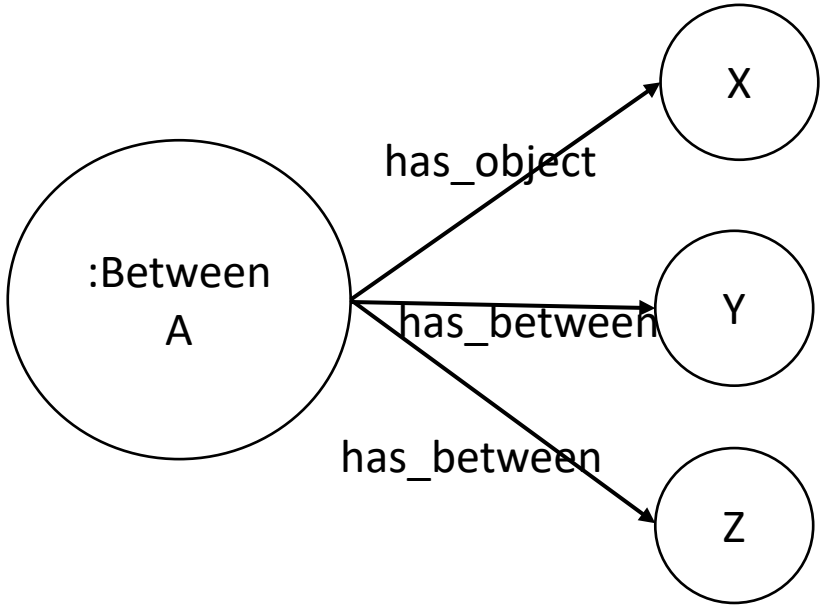


When to Introduce a Relationship Property

- Disadvantages
  - Many systems do not index on relationship properties
  - This may not be a problem if relationship properties are used in the last stage of query processing
  - For performance sensitive queries, it is better to reify the relationship

Handling non-binary relationships

- Reification is a common technique to handle relationships with arity higher than 2
  - Create an object representing the relationship
  - Create objects for each argument of the relationship
  - Introduce relationships to connect them



X is between Y and Z

Summary

- Common considerations
  - Choice of classes and relations
  - Reification and handling higher arity relationships
- RDF knowledge graph design driven by
  - Use over the WWW
  - Reuse of vocabularies
  - Linking data
- Property Graph design driven by
  - Optimizing query performance

Even though there are some guidelines to design, there are also equivalent good choices.