

# OWL or SHACL?

## *A Beginner's Guide to Making the Right Choice*

**ISWC 2025**  
**November, 3rd, 2025**

**Tara Raafat**, Head of Metadata and Knowledge Graph Strategy, Office of the CTO  
**Davide D'Amico**, Ontologist, Educator, and Software Engineer, Bloomberg Engineering

# The Ultimate Showdown: Game 1



*Disclaimer: Created by ChatGPT*

# What are we learning today?

- This will be an “Example-Based” hands-on tutorial
  - Together, we will make a choice of OWL or SHACL (or both) to solve problems and understand the implications of the choice
- Main Focus: Comparing the strengths and weaknesses of OWL and SHACL
- We will talk about
  - Open World Closed world
  - Reasoning
  - Classification
  - Transitivity and Property chains
  - And more.....
- This will NOT be an in-depth tutorial of everything OWL or everything SHACL... if only we had more time :)

# Some Logistics

- Download Protégé Desktop (NOT WebProtégé): <https://protege.stanford.edu/>
- Open SHACL playground: <https://shacl-playground.zazuko.com/>
- Open workshop page for all examples and code: <https://tinyurl.com/owlshacl>

# When you see these icons



Go to Protégé

{  
For modeling in owl  
Adding data instances  
Using a reasoner  
Running SPARQL



Go to SHACL playground

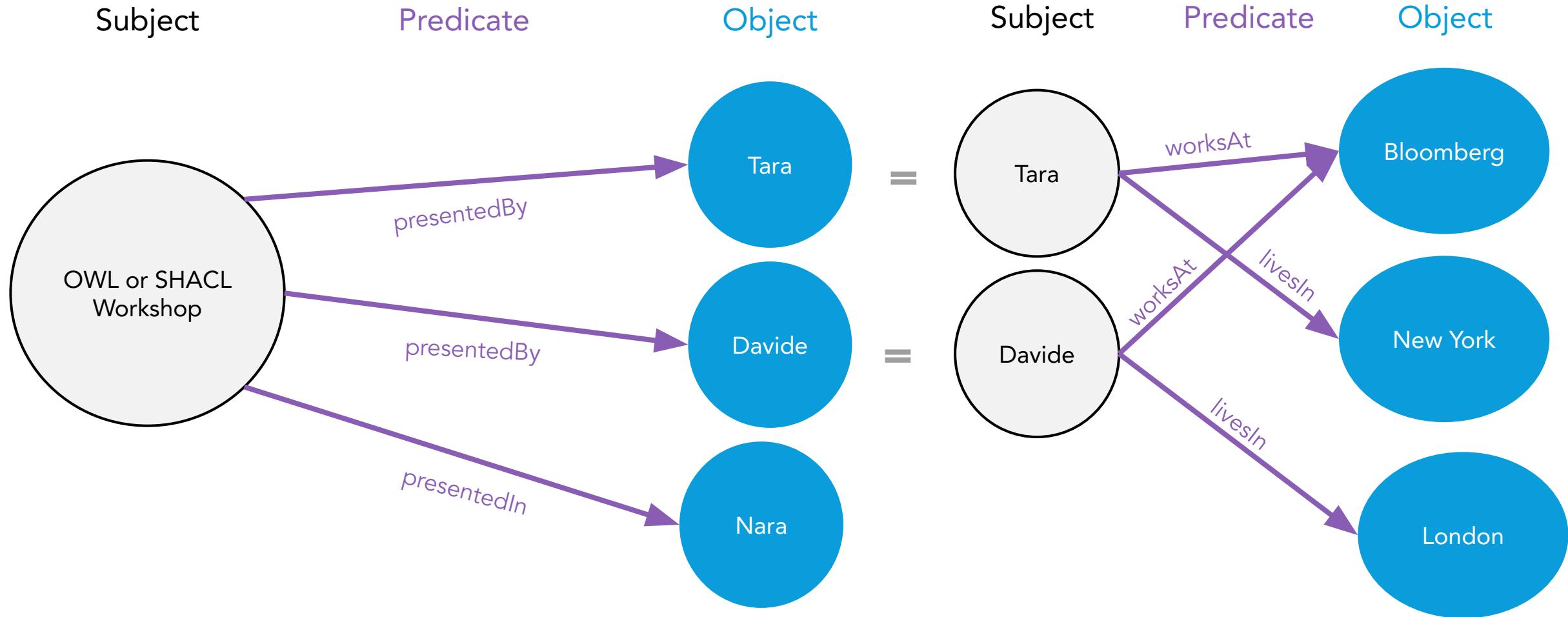
{  
For writing SHACL shapes  
For validating data graphs against SHACL shapes



Follow the instructions

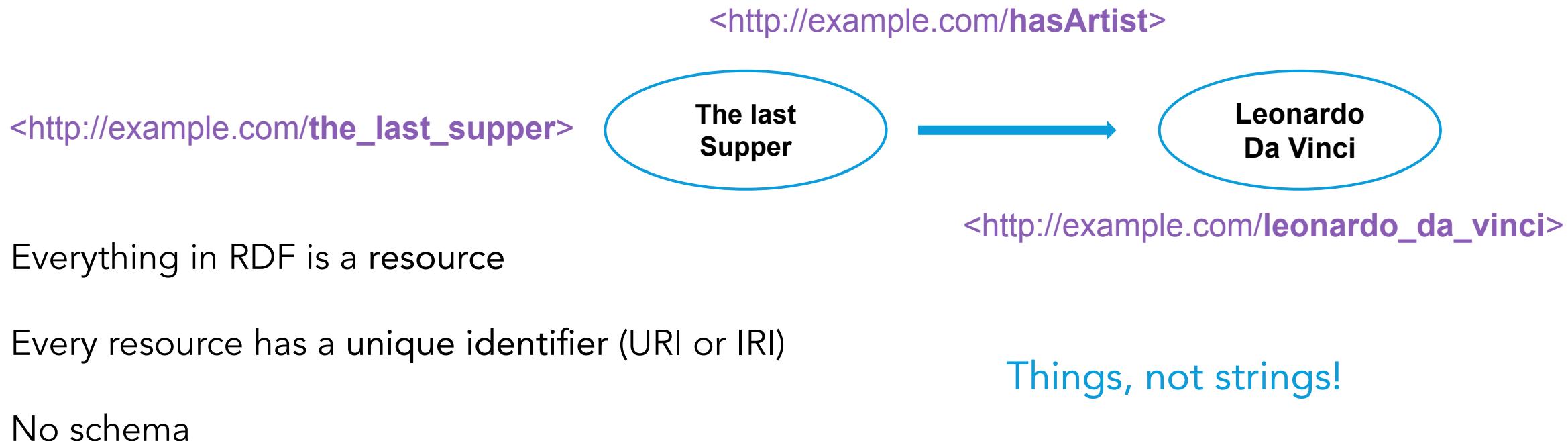
# A Little Background

# The World In Triples



# RDF (Resource Description Framework)

- W3C de facto standard for data interchange that is used for representing highly interconnected data
- Simple triple-based model: <subject> <predicate> <object>



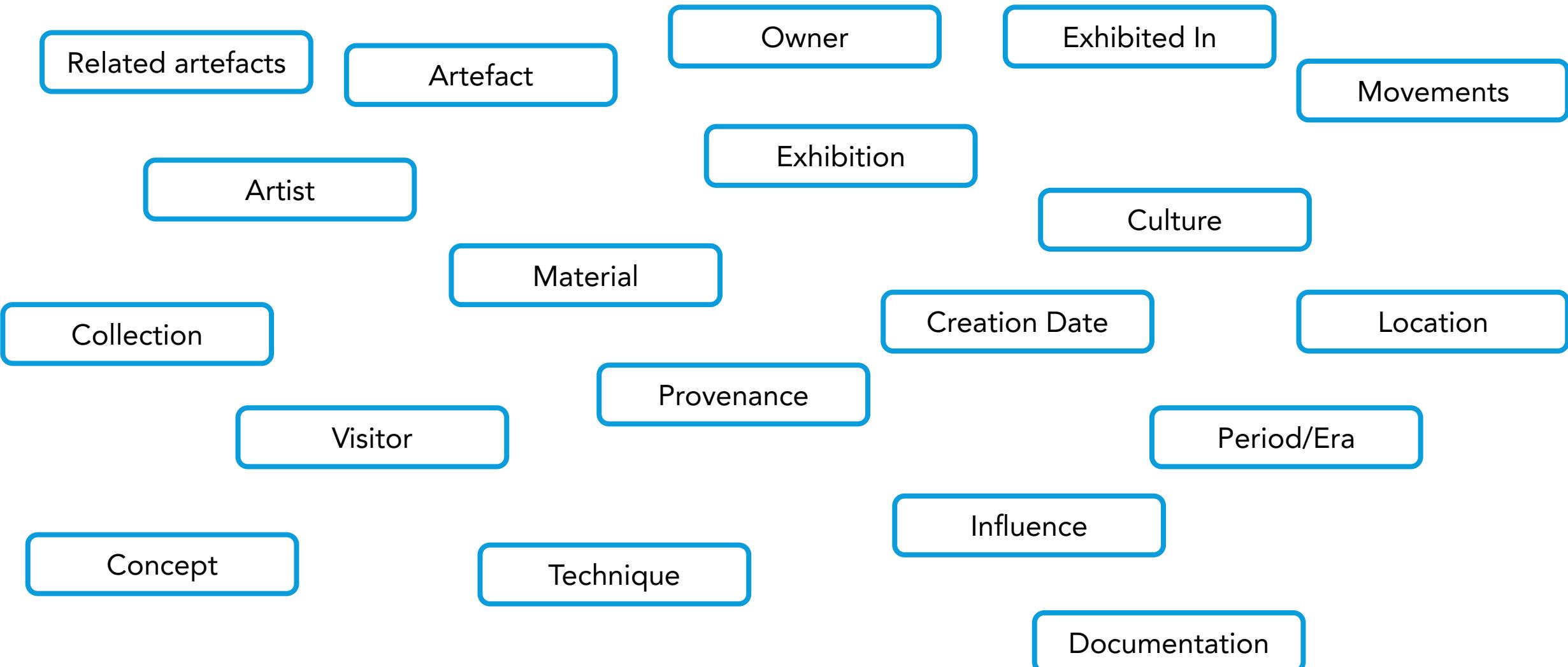
# A day at the museum ...



# Walking through an exhibition .... A million questions !

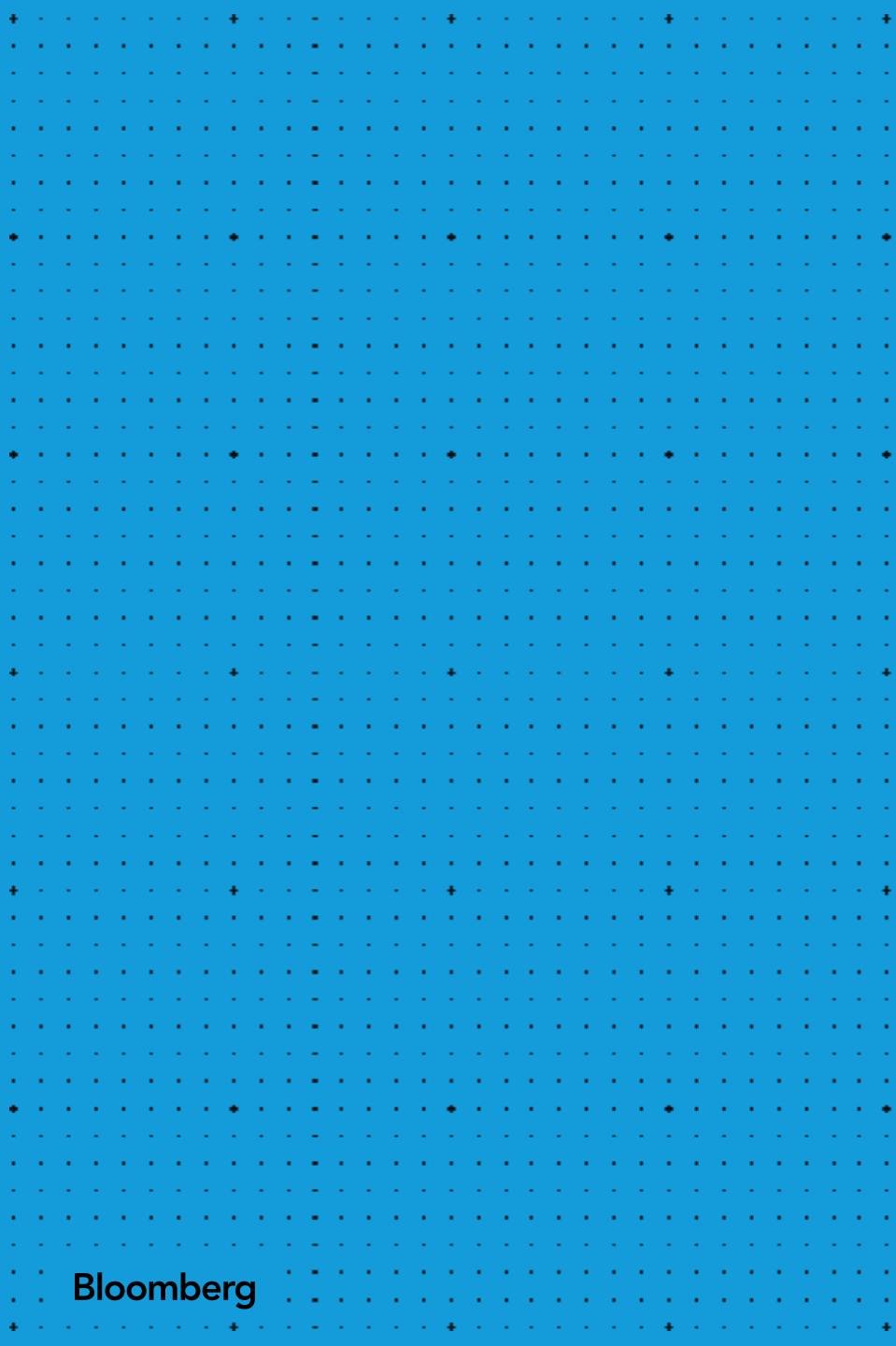
- What artefacts do you have?
- All the information about them in terms of artist, date of creation, materials, etc.?
- When you create the exhibitions how do you decide what artefacts go together?
- Can you know which city or county an artefact is being exhibited?
- Based on the art movement era of the piece can you tell the artists of those eras?
- Is it possible for you to know whether a copy of an original art pieces you have exists somewhere else?
- Which piece was influenced by whom?

# How do they organize their data?



# ROUND 1: Simple queries





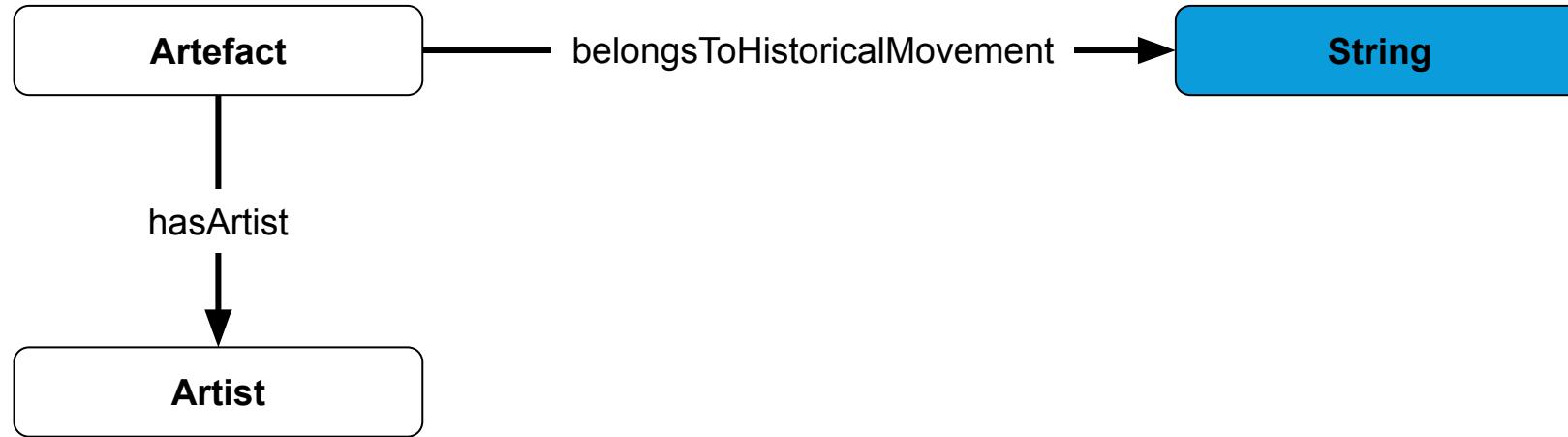
Let's create the  
ontology !

# How do you decide on the design of the ontology model?

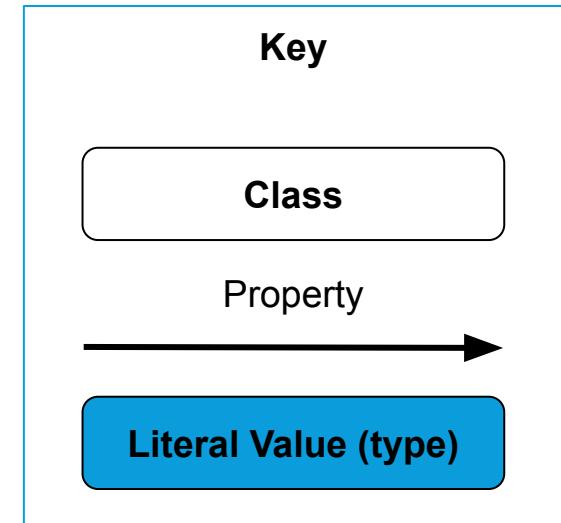
Start with the competency questions...

- What **artefacts** are attributed to a specific **artist**?
- What **artefacts** are associated with a specific **historical movement**?
- What **artefacts** are attributed to a specific **artist** and associated with a specific **historical movement**?

# The Simple Ontology . Let's code that in OWL

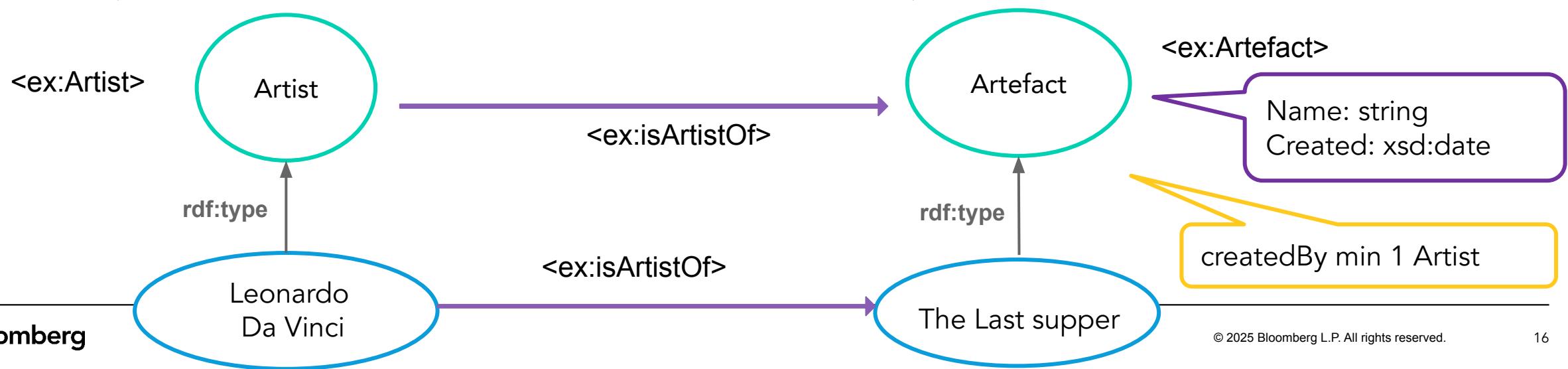


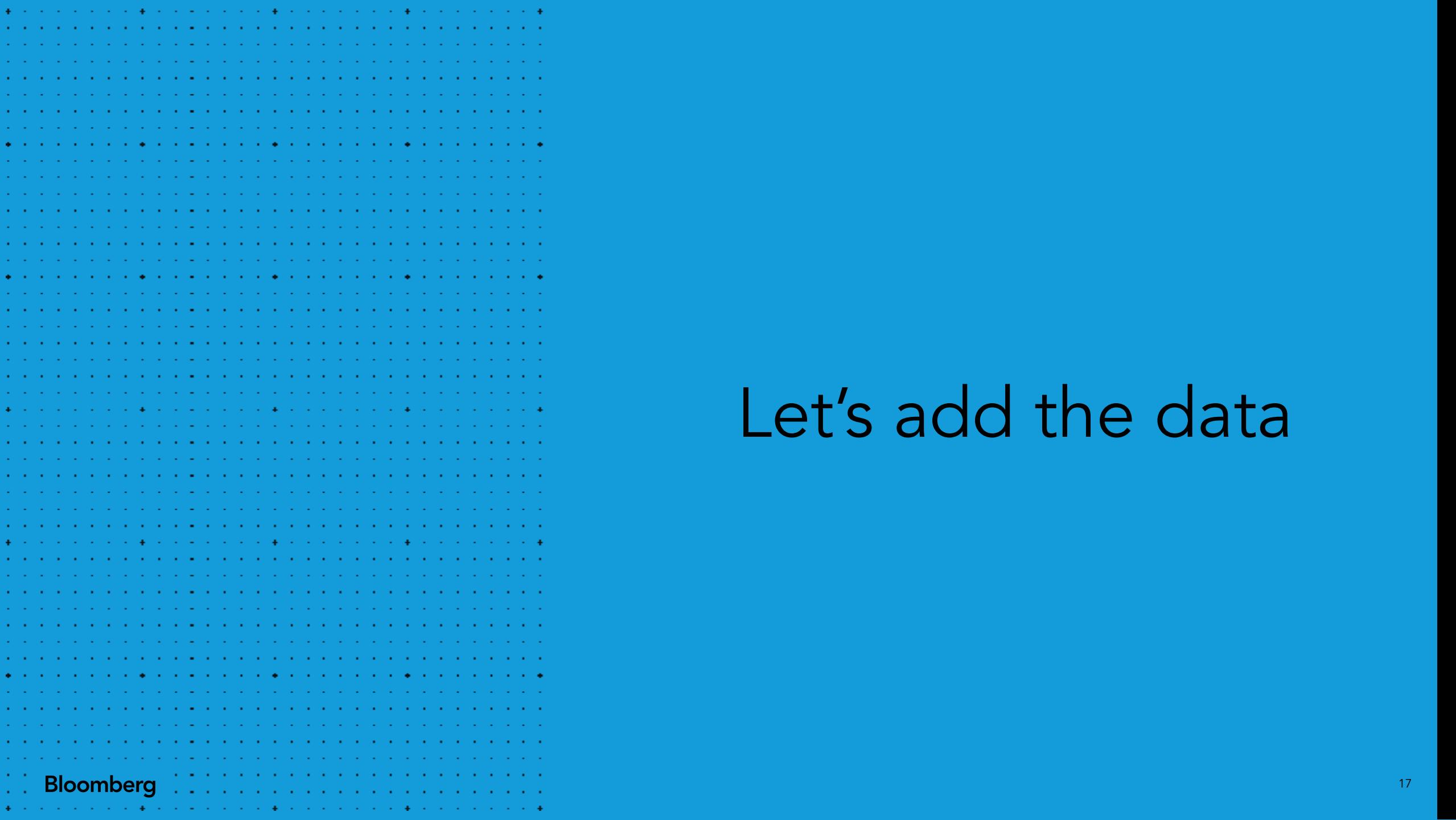
- Open [model\\_00.ttl](#)
- Add the above classes and properties  
Or
- Open [model\\_01.ttl](#)



# OWL (Web Ontology Language)

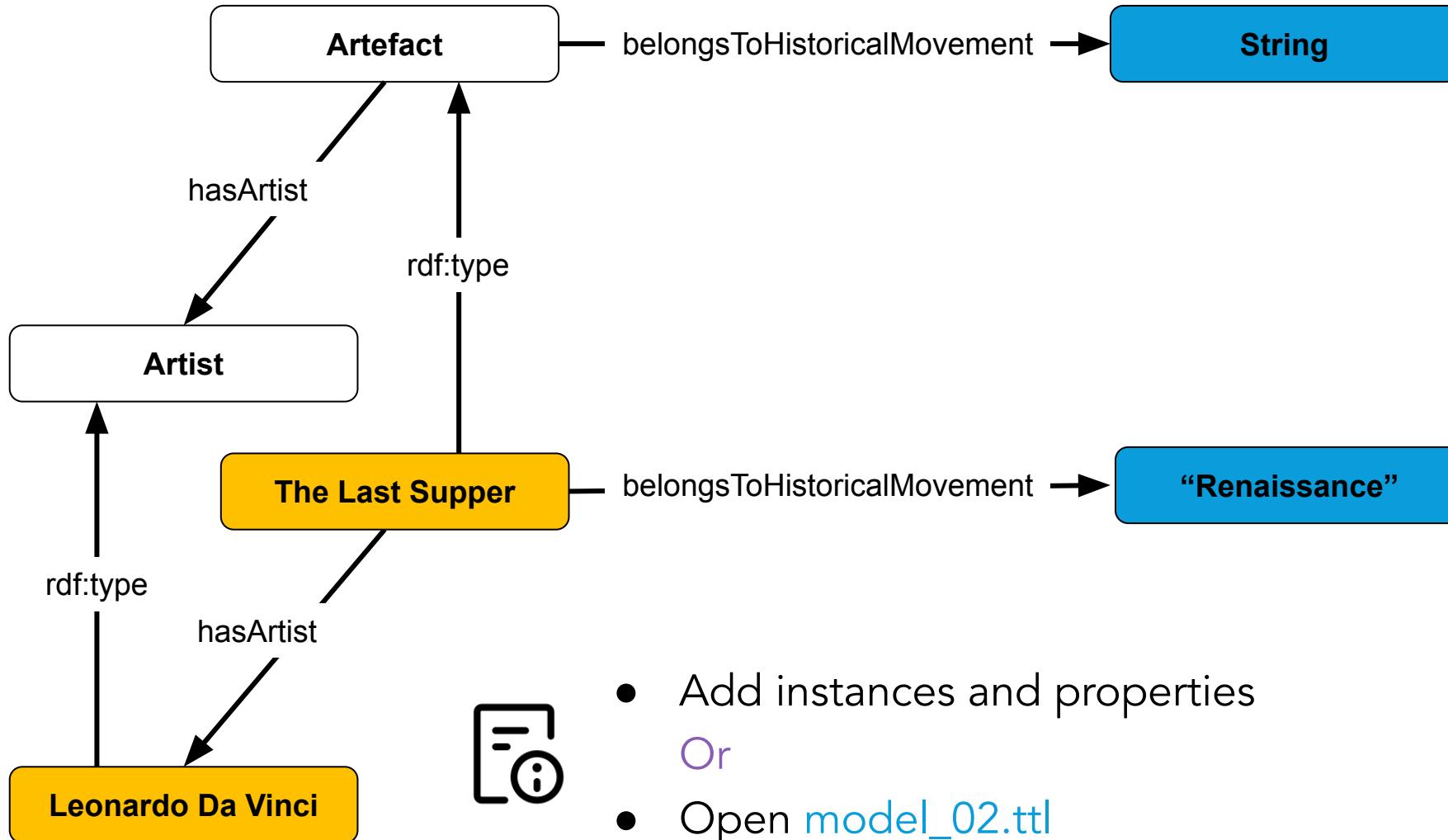
- An RDF-based de facto standard for ontology development
- Main components include
  - Classes: define concepts in a domain
  - Properties:
    - Object properties: define relationships between concepts
    - Datatype properties: define relationships between a concept and a literal
  - Individuals: instances of classes
  - Restrictions: allow definition of cardinality restrictions, as well as existential & universal quantifications  
(some rules can be defined in the context of a restriction)





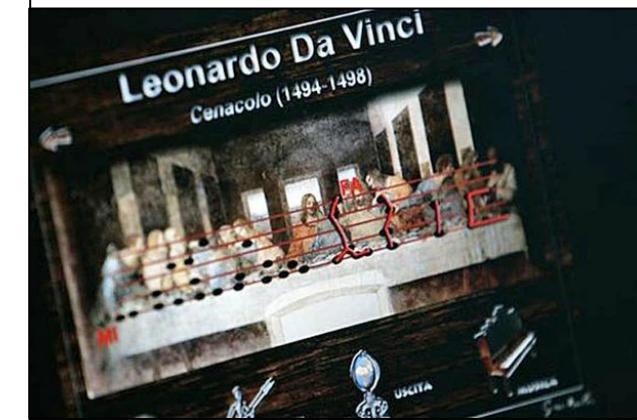
Let's add the data

# The Knowledge Graph



## Fun Fact:

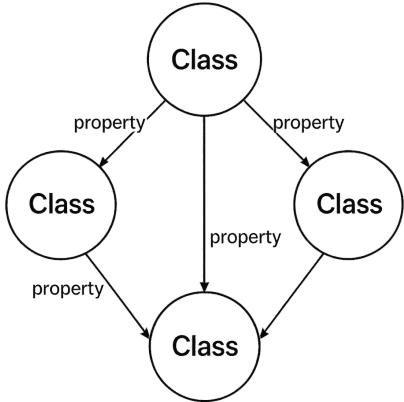
In 2007, an Italian musician claimed to have found musical notes in *The Last Supper* that result in a tuneful 40-second musical composition. The notes, which are 'hidden' in the bread rolls and hands of the apostles in the painting, is read from right to left, following Da Vinci's own writing style.



Source: <https://www.livescience.com/2039-da-vinci-musical-code-supper.html>

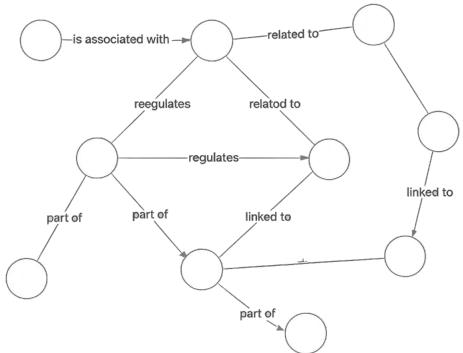
# What is a knowledge Graph ?

# The Two Layers of the Knowledge Graph



## Semantic Layer

Defines what things mean and how they relate to each other in a machine-understandable format



## Data Layer

Real-world facts and observations mapped onto the semantic structure/layer

# What is a Knowledge Graph?

*"A knowledge graph is the DNA of information —  
It encodes not just facts, but the structure, relationships,  
and rules that bring raw data to life and allow it to grow,  
adapt, and evolve*

*Tara Raafat*



# Let's test the queries

1. What artefacts are attributed to Leonardo da Vinci?

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:hasArtist ex:leonardo_da_vinci .
}
```

2. What artefacts are associated with the Renaissance movement?

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:belongsToHistoricalMovement "Renaissance"^^xsd:string .}
```

3. What artefacts are attributed to Leonardo da Vinci and associated with the Renaissance movement?

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:hasArtist ex:leonardo_da_vinci ;
  ex:belongsToHistoricalMovement "Renaissance"^^xsd:string .}
```



- Open [model\\_02.ttl](#)
- Open '[SPARQL](#)' tab in the website
- Run the queries in [Section 1](#)

Can we answer  
those questions  
using SHACL?



# Could SHACL be used to achieve this?

Yes!

Shape 1 in SHACL tab of website

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;

  sh:property [
    sh:path ex:belongsToHistoricalMovement ;
    sh:datatype xsd:string ;
  ] ;

  sh:property [
    sh:path ex:hasArtist ;
    sh:class ex:Artist ;
  ] .
```

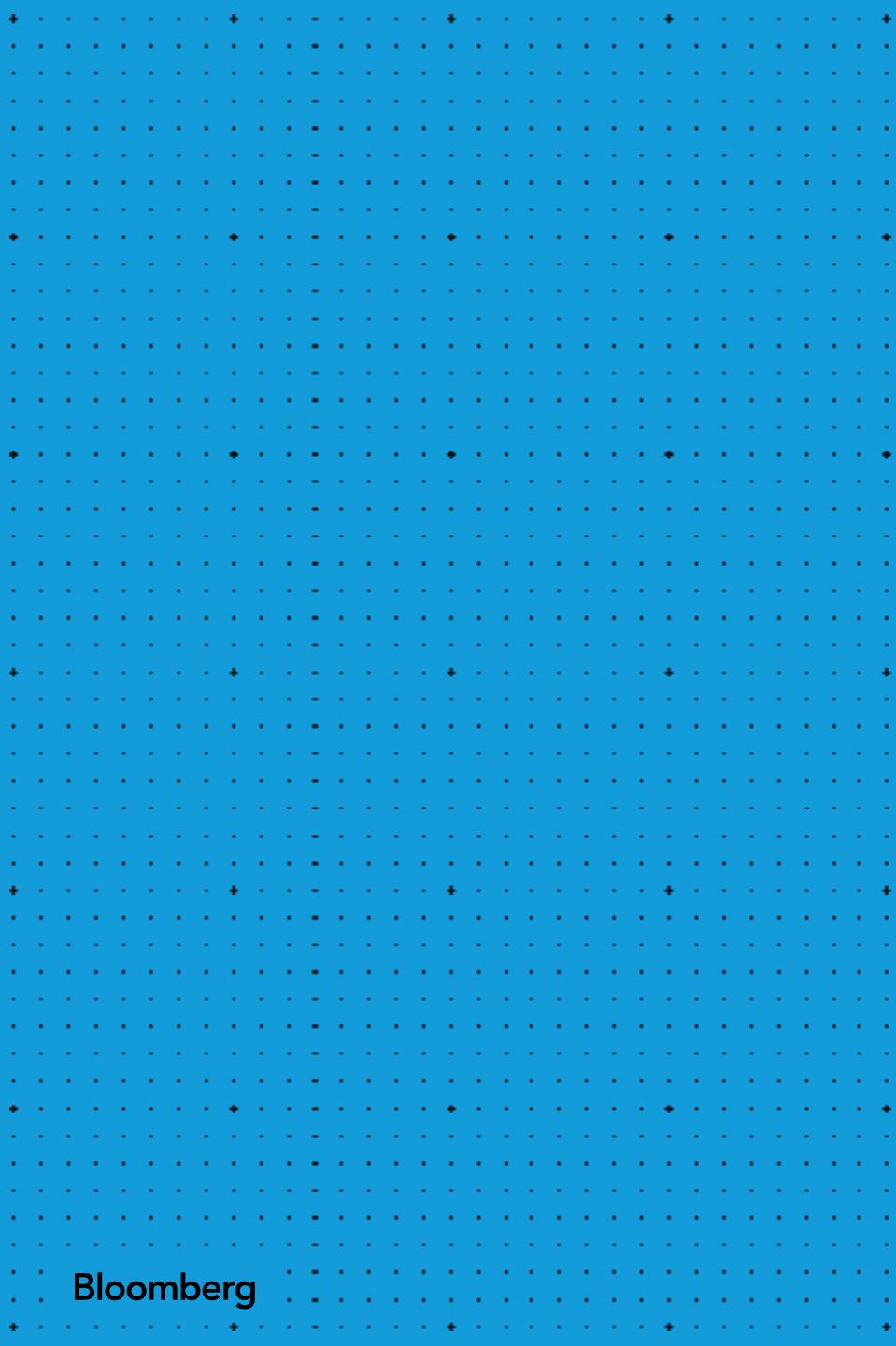
Since you are querying the data , same SPARQL will provide the same answer

If you want to query the model then you have to write different SPARQL queries one on owl construct and one on shacl constructs

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
SELECT *
WHERE {
  ?s a sh:NodeShape .
```

However, if you want to use the same SPARQL query written in OWL syntax for both, you'll need tools that translate SHACL to OWL

But what if you want to do more?



# What if data is missing ?

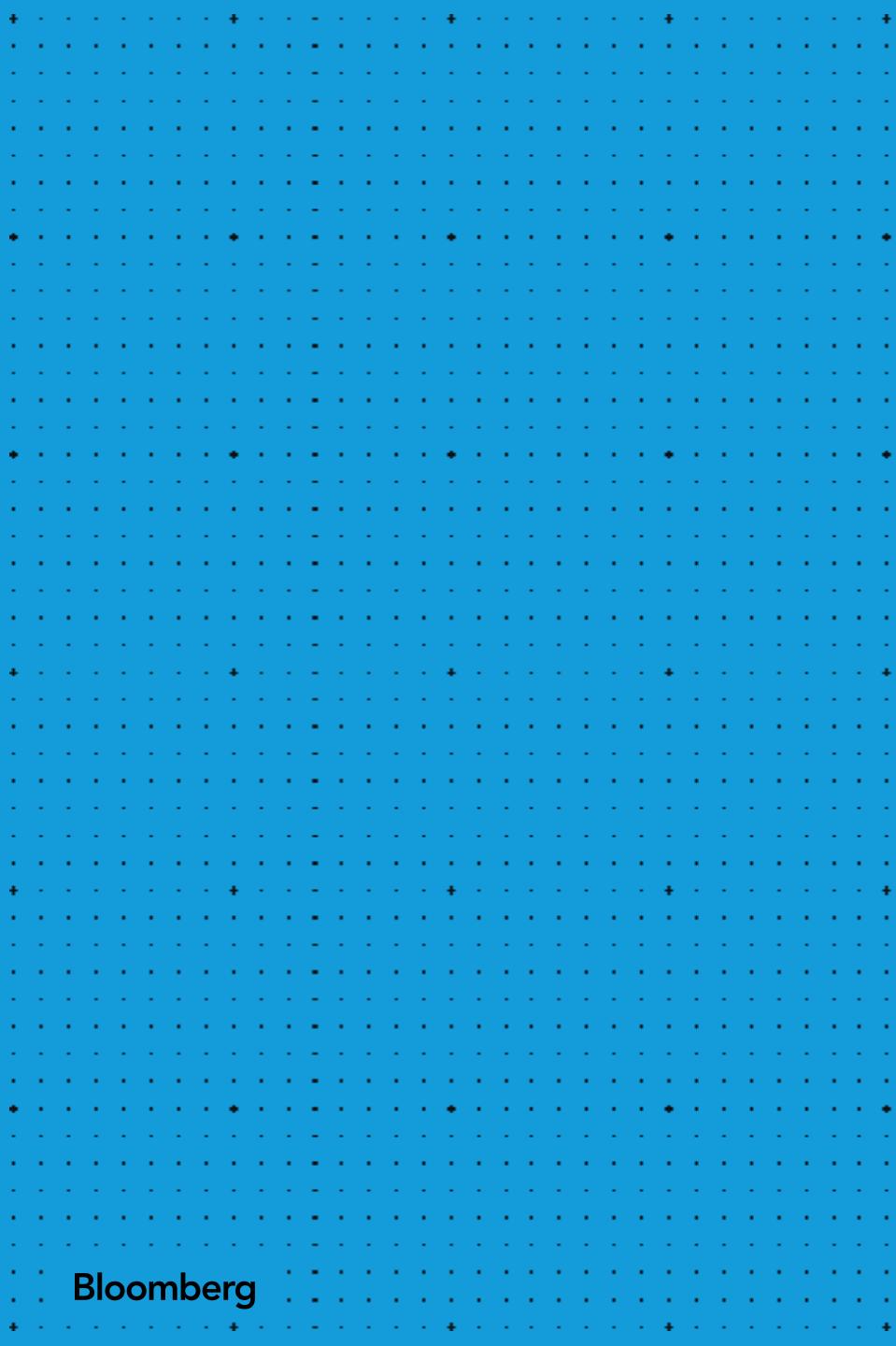
# ROUND 2: Dealing with Incomplete Information



# What about Missing (*Incomplete*) information?

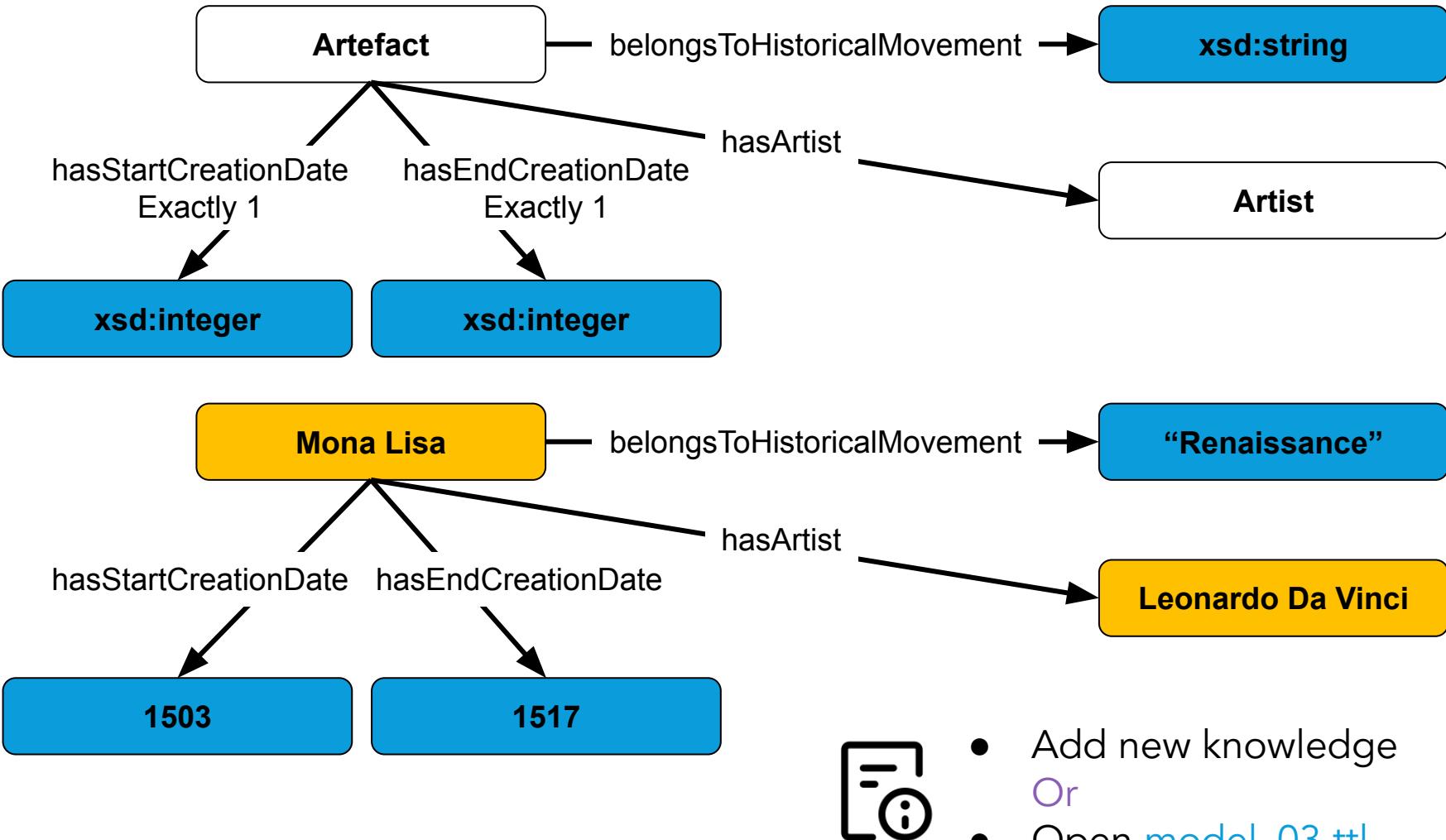
Some artworks are acquired without complete information at the time of acquisition.

- How do museum systems typically handle **missing** data?
- Does the absence of certain metadata trigger an **error**?
- Should incomplete records be treated as **errors**?

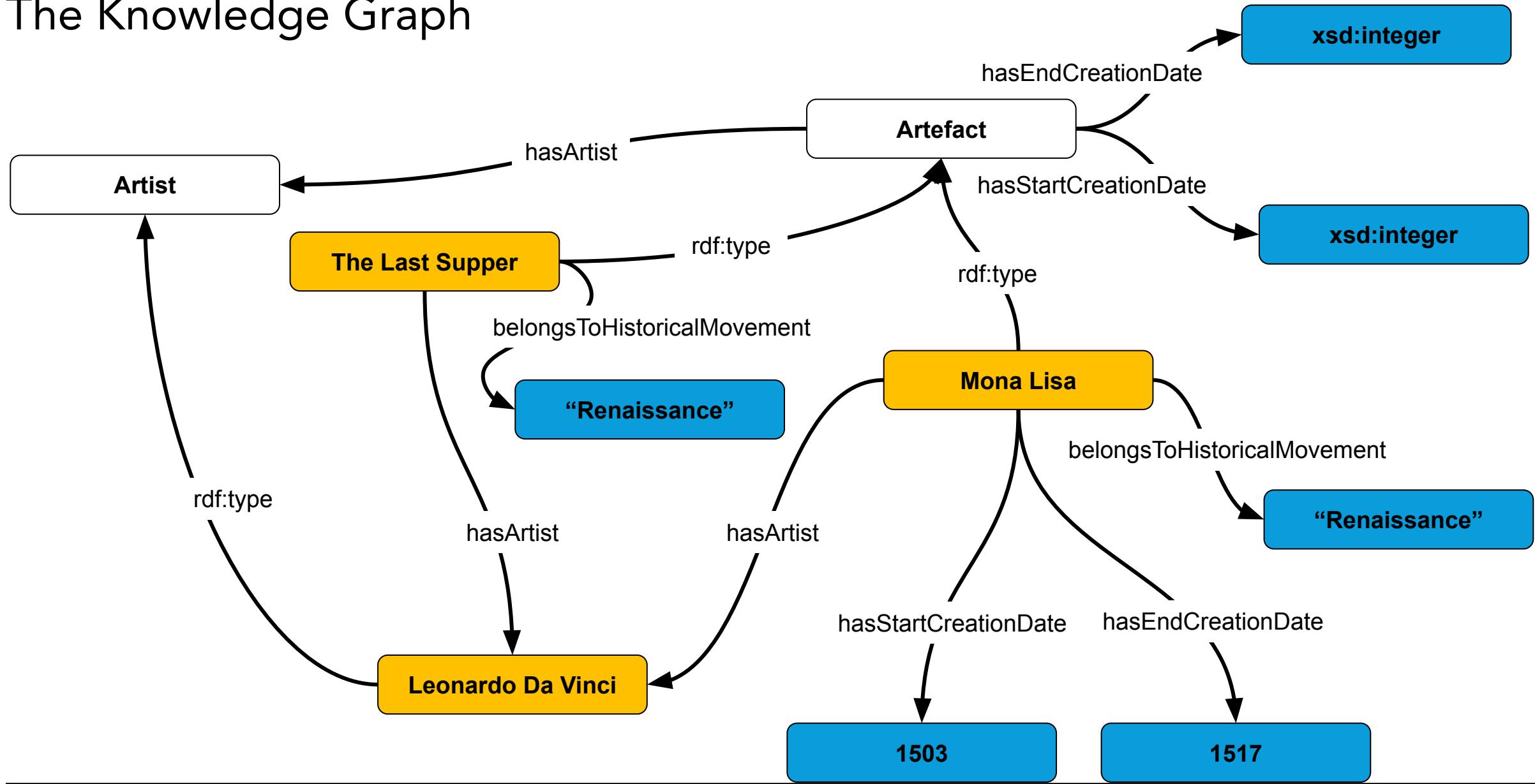


Let's add more  
Data

# The Knowledge Graph extended



# The Knowledge Graph



# Would OWL restrictions enforce data's existence & cause an error?

Artifact := hasStartCreationdate exactly 1 xsd:integer

Artifact := hasEndCreationdate exactly 1 xsd:integer



- Load [model\\_03.ttl](#)
- Run reasoner

The reasoner **doesn't** throw an error.



Why? Because OWL operates on an Open World Assumption ([OWA](#)) and assumes that the absence of information does not imply its absence in reality.

# Open World Assumption (OWA)



If certain information **does not exist** in the data, it **cannot** be assumed to be **false**, it's simply **incomplete**.

In such cases, an OWL reasoner **does not return an error**; it simply does not infer anything additional from the **missing information**.

When is the **OWA useful**?

The OWA is particularly valuable in scenarios where not all knowledge is **known** or **available** at a given time. In such contexts, the use of an ontology under OWA supports **ongoing discovery** by remaining flexible, adaptable, and collaborative.

New information can be **added** over time without invalidating existing assertions, enabling systems to **grow** and evolve without requiring full knowledge upfront.

What if you wanted  
to ensure the data  
existed at the time  
of acquisition?

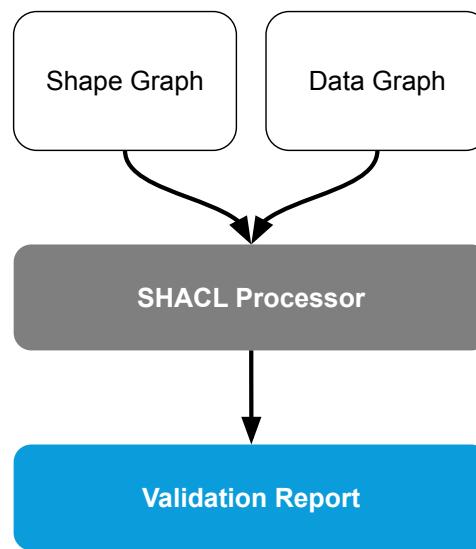
# ROUND 3: Validation



# Let's test SHACL

We need to define the **exact shape** of our data, including the required properties and any constraints on their values.

SHACL shapes are required to validate RDF data against specific structural and value constraints.



## Shape 2

```
ex:ArtefactShape a sh:NodeShape ;  
sh:targetClass ex:Artefact ;  
  
sh:property [  
    sh:path ex:belongsToHistoricalMovement ;  
    sh:datatype xsd:string ;  
]  
  
sh:property [  
    sh:path ex:hasArtist ;  
    sh:class ex:Artist ;  
]  
  
sh:property [  
    sh:path ex:hasEndCreationDate ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
]  
  
sh:property [  
    sh:path ex:hasStartCreationDate ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
].
```



# Validity Conditions

```
sh:property [  
    sh:path ex:hasEndCreationDate ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
] ;  
  
sh:property [  
    sh:path ex:hasStartCreationDate ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
] .
```

The validation shows **error**.



**Why?** Because SHACL operates on an Closed World Assumption (**CWA**) and assumes that the absence of information is an error in reality.



# Closed World Assumption (CWA)

- The system assumes it **has** all the information it needs about its domain.
- Anything not explicitly known or stated as true is assumed to be **false** => Missing information is assumed to be **false**, the data graph is flagged as **invalid**.
- If a data node does not meet the defined restrictions, such as a wrong value type for a property, it results in a validation **error**.

When is the **CWA** useful?

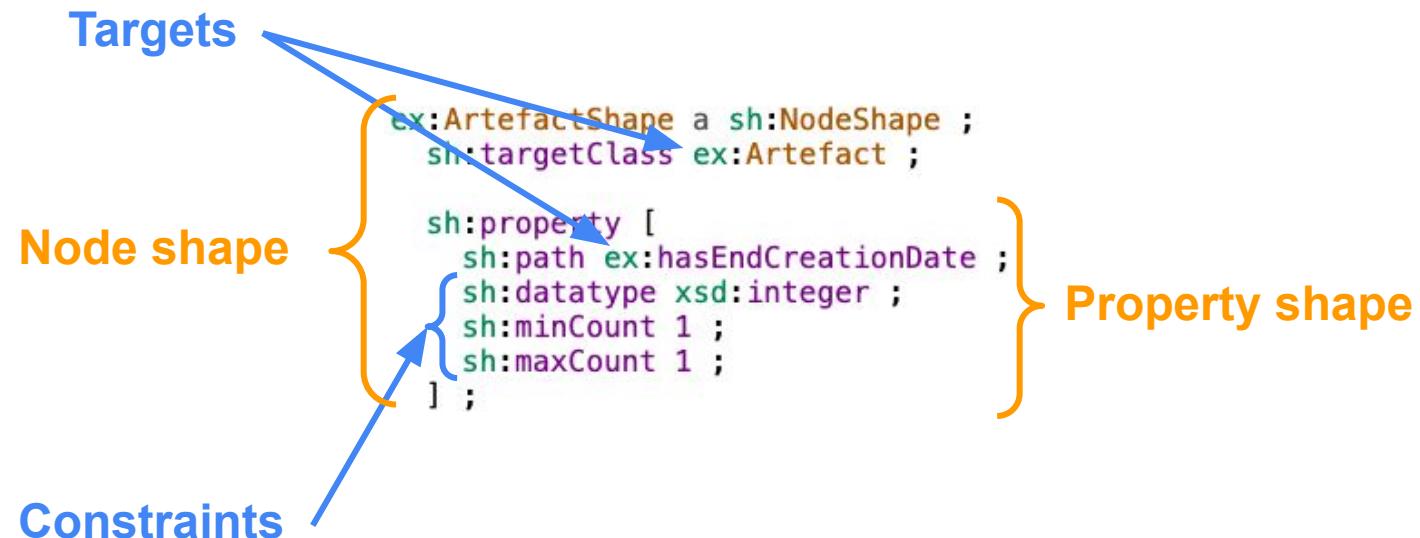
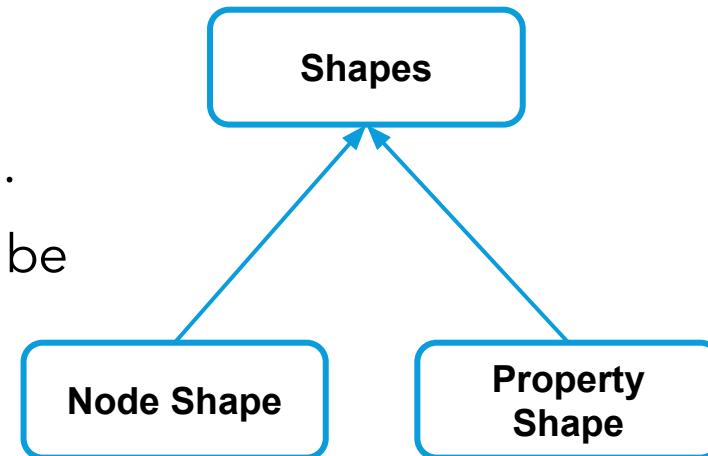
The CWA is particularly valuable in scenarios where data integrity is crucial and error reports need to be generated if information is missing or invalid

# The SHACL Shape

In **SHACL**, a shape is composed of targets and constraints.

- **Targets** identify the nodes in the data graph that the shape applies to.
- **Constraints** specify the rules that those nodes must satisfy in order to be considered valid.

*declare constraints directly on a node*



# Targets

**sh:targetClass**

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;
  sh:property [
    sh:path ex:belongsToHistoricalMovement ;
  ] ;
```

**sh:targetNode**

```
ex:ArtistShape a sh:NodeShape ;
  sh:targetNode ex:Leonardo_Davinci;
  sh:nodeKind sh:IRI;
  sh:property [ sh:path ex:hasDateOfBirth ;
    sh:range xsd:string ].
```

**sh:targetSubjectOf**

```
ex:SculptureShape a sh:NodeShape ;
  sh:targetSubjectOf ex:hasArtist ;
  sh:property [
    sh:path ex:hasEndCreationDate ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
```

**sh:targetObjectOf**

```
ex:SculptureShape a sh:NodeShape ;
  sh:ObjectOf ex:hasArtist ;
  sh:property [
    sh:path ex:hasName ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ; [] ;
```

# Recap so far...

Open world: missing information does not cause an error **OWL**

Closed world: missing information causes an error **SHACL**

Data Integrity: **SHACL**

Data Validation: **SHACL**

# ROUND 4: Classification



# What about Classification?

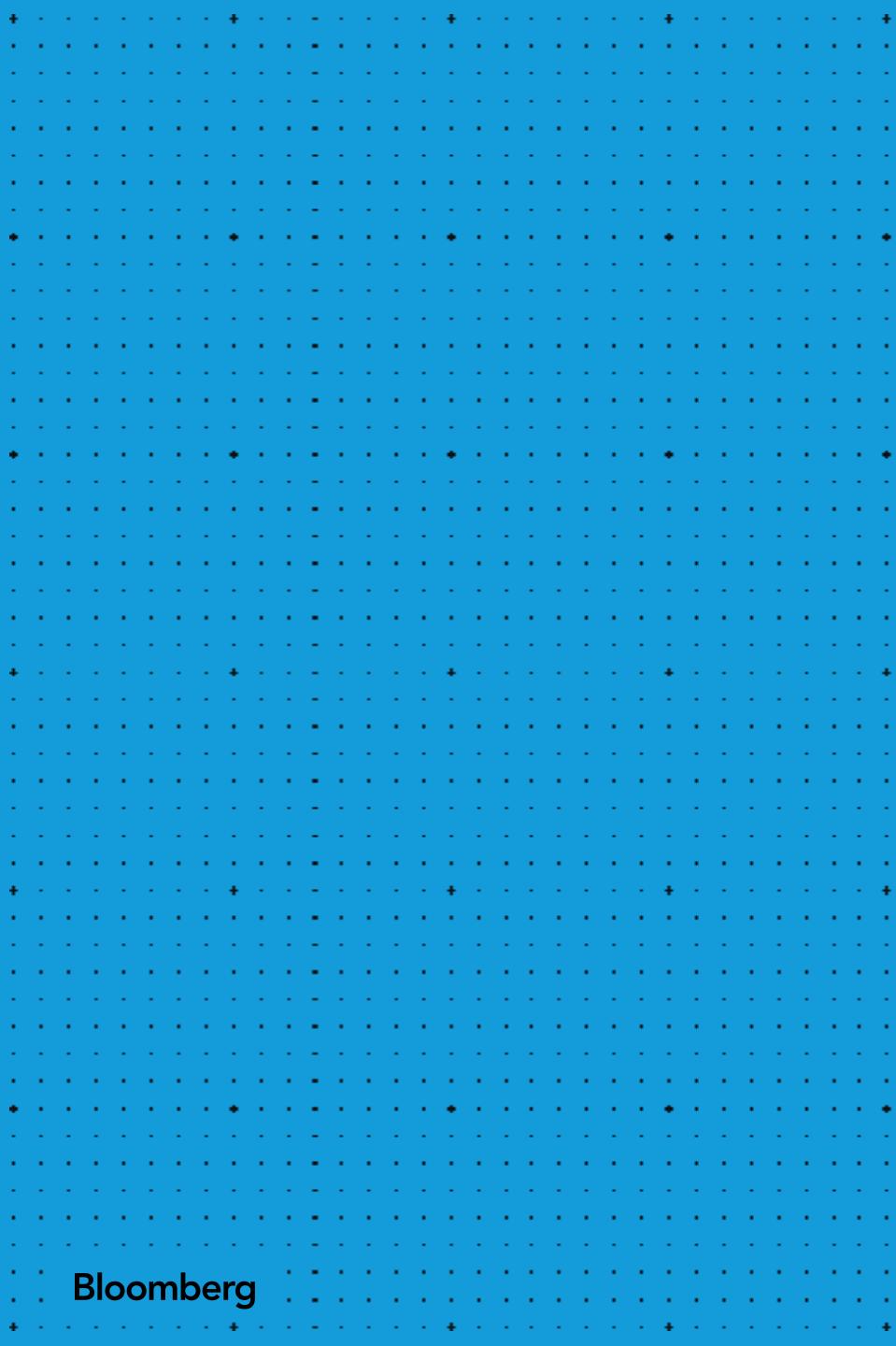
Museums regularly acquire new artefacts and classify them according to internal categorization systems.

Can this classification be automated? For instance, could a system infer whether an artefact is a painting or a sculpture based on metadata available at the time of acquisition?



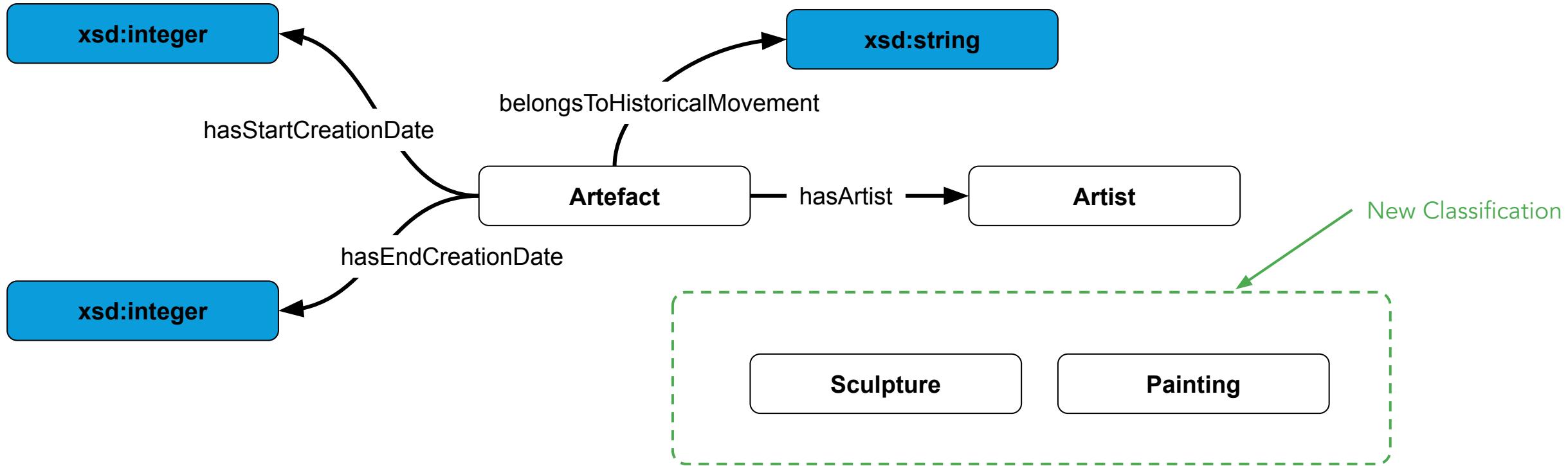
- On [model\\_03.ttl](#)
- Add start and end creation dates for 'The Last Supper' (1494-1498)
- Add the data about David on the right  
*Or*
- Open [model\\_04.ttl](#)

```
ex:david
  ex:belongsToHistoricalMovement "Renaissance" ;
  ex:hasArtist ex:michelangelo ;
  ex:hasEndCreationDate 1504 ;
  ex:hasStartCreationDate 1501 ;
  ex:hasSculptureMaterial "Marble" ;
  rdfs:label "David" .
```



# Let's Extend Our Ontology

# Ontology extension



- Add two new classes Sculpture and Painting
- Or
- Open [model\\_05.ttl](#)
  - Run the reasoner



# Classification Query

Using the new model and the data we had about David sculpture;  
Can we query the data to see what's sculptures we have?

Let's query the data for Sculpture ([SPARQL query 2.1](#)):

```
SELECT ?x  
  
WHERE {  
    ?x a ex:Sculpture .  
}
```

Nothing!

**Why?** There is neither a definition for the class sculpture nor a definition of the property hasSculptureMaterial.



## Fun Facts:

The statue was carved from a single block of marble.

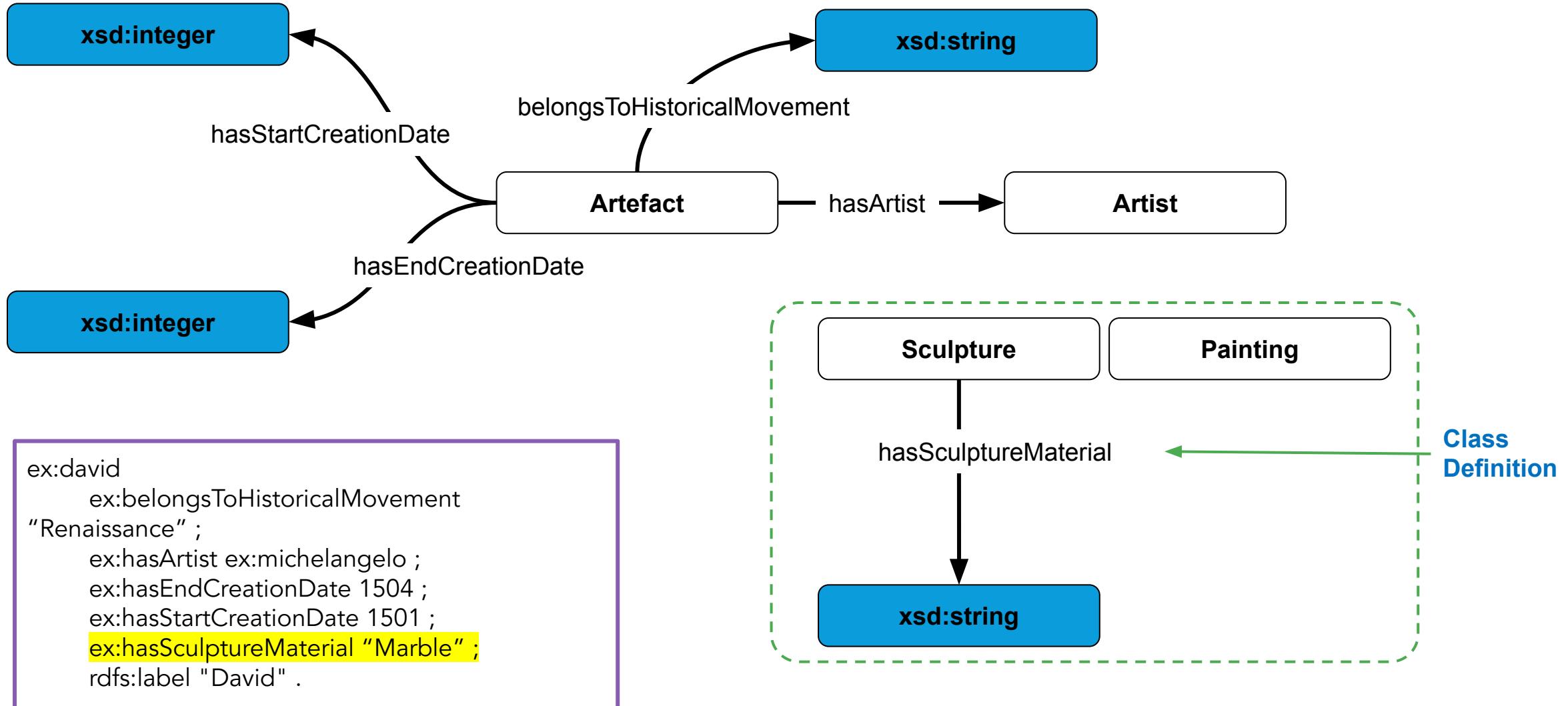
David's right hand is disproportionately large compared to his body because in the Middle Ages, David was said to be "manu fortis", meaning strong of hand.

There are at least 30 full-size replicas of this statue around the world.



Source: <https://www.contexttravel.com/blog/articles/ten-facts-about-the-statue-of-david>

# Class Definition



# Classification by Class Definition

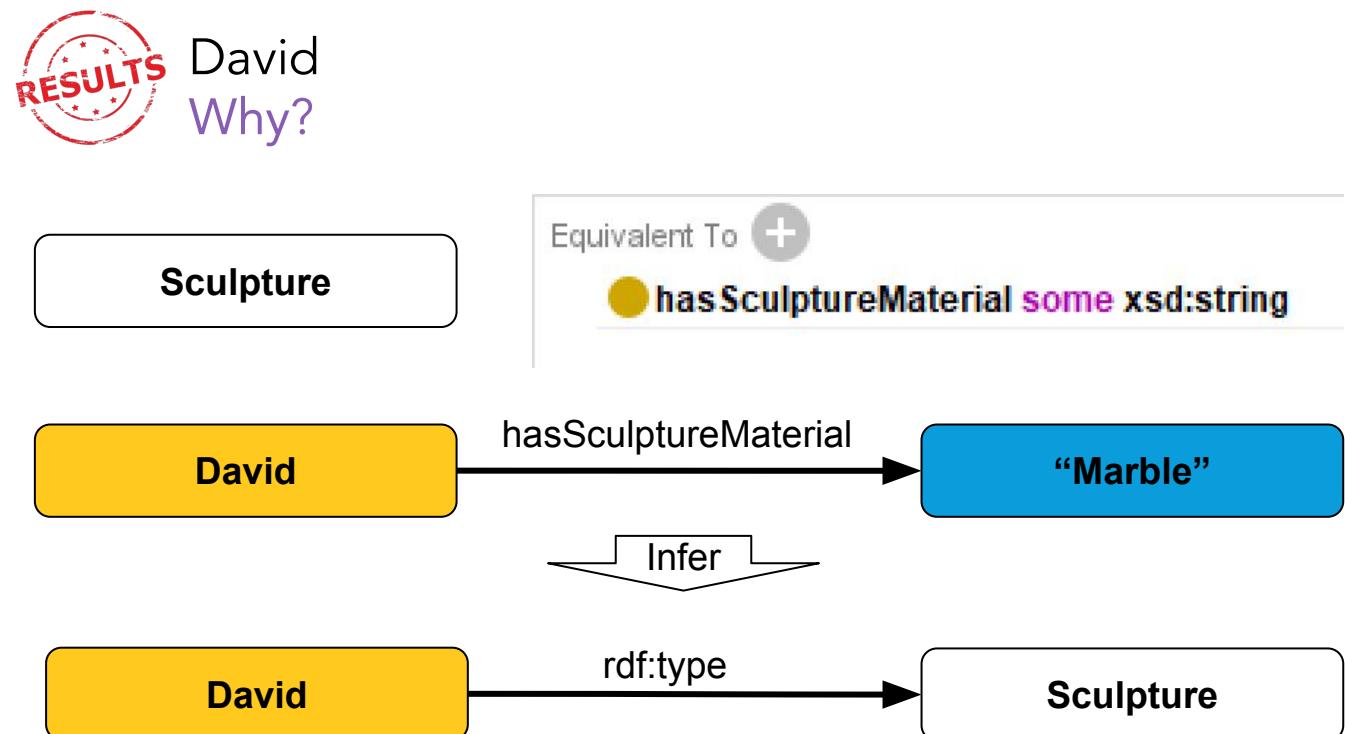
- Add the class definition  
Or
- load [model\\_06.ttl](#)
- Run the [reasoner](#)
- Add the changes suggested by the reasoner  
Or
- upload [model\\_06\\_inferred.ttl](#)
- Run the query 2.1 again



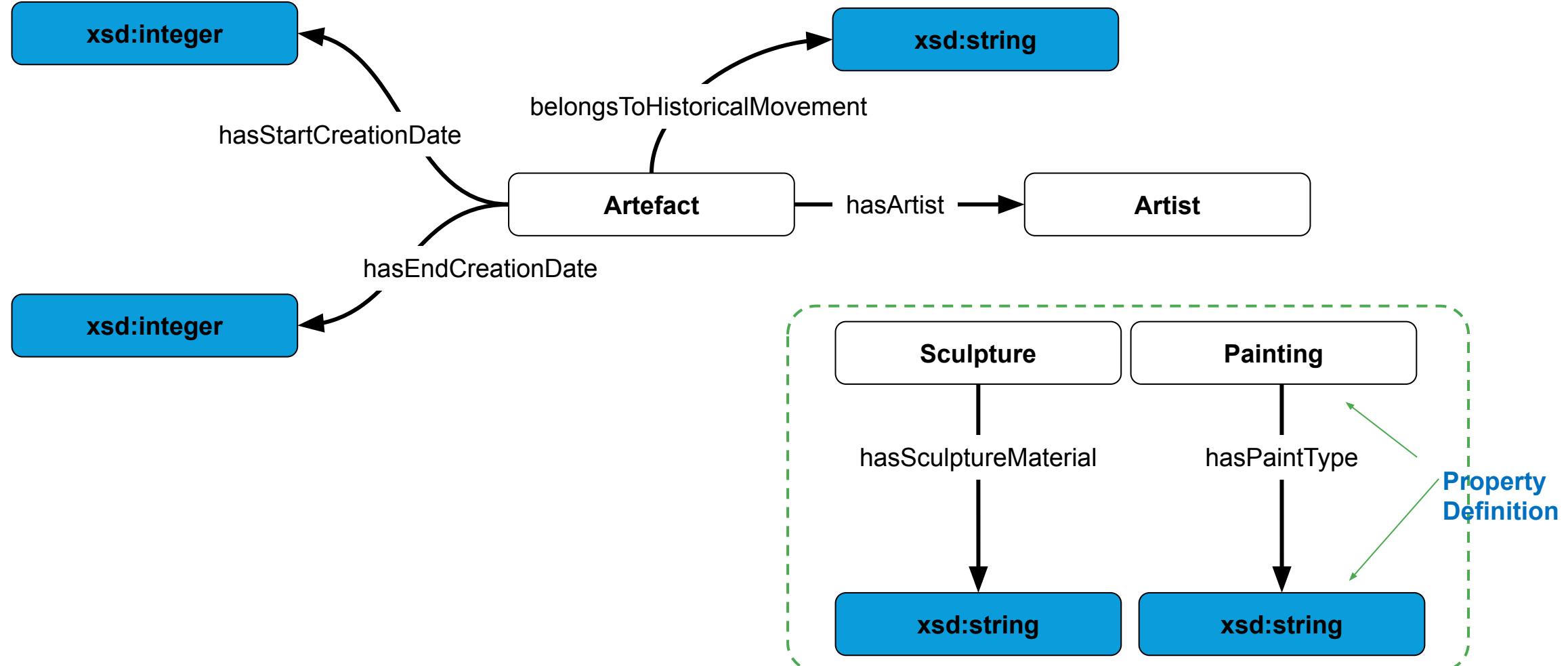
```
SELECT ?x
WHERE {
  ?x a ex:Sculpture .
}
```

## Class Definition

A sculpture has some sculpture material



# Property Definition



# Classification by Property definition

Property definition:

ex:hasPaintType owl:domain Painting

ex:hasPaintType owl:range xsd:string

At the data level:

ex mona\_lisa ex:hasPaintType "Oil paint"

ex:the\_last\_supper ex:hasPaintType "Mural Paint"



- Add the new knowledge  
Or
- load [model\\_07.ttl](#)
- Run the reasoner
- Add the changes suggested by the reasoner  
Or
- upload [model\\_07\\_inferred.ttl](#)
- Run query 2.2



# Classification by Property definition

Query 2.2

```
SELECT ?x
```

```
WHERE {
```

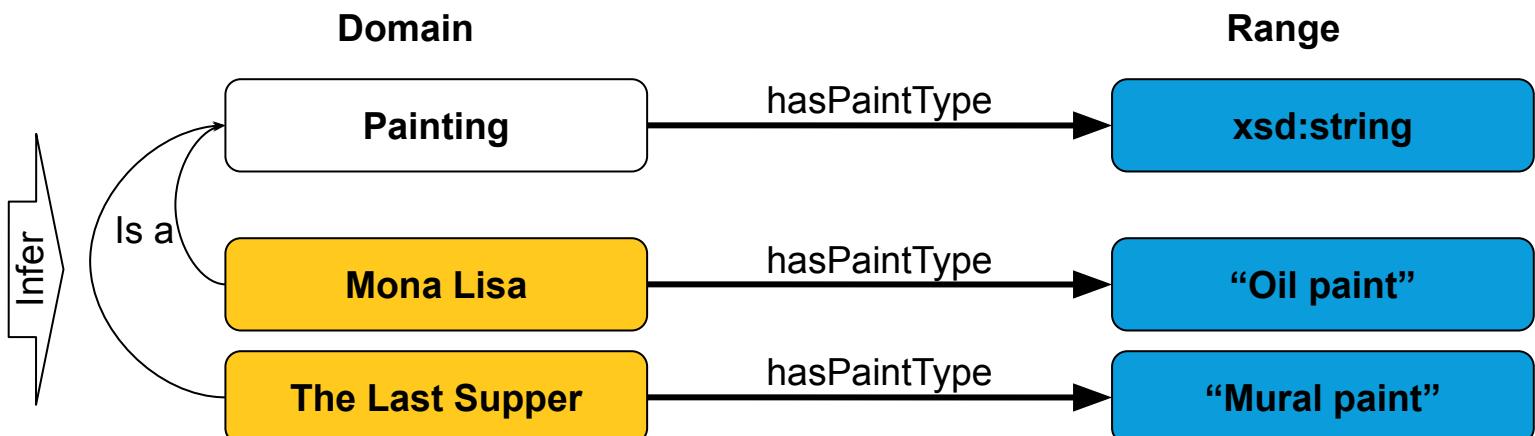
```
    ?x a ex:Painting .
```

```
}
```



Mona Lisa & Last Supper

Why? Based on the "hasPaintType" definition



# Classification Query

With all the data we have, can we query to see what are all the artefacts we have?

Query 2.3

```
SELECT ?x  
  
WHERE {  
    ?x a ex:Artefact .  
}
```

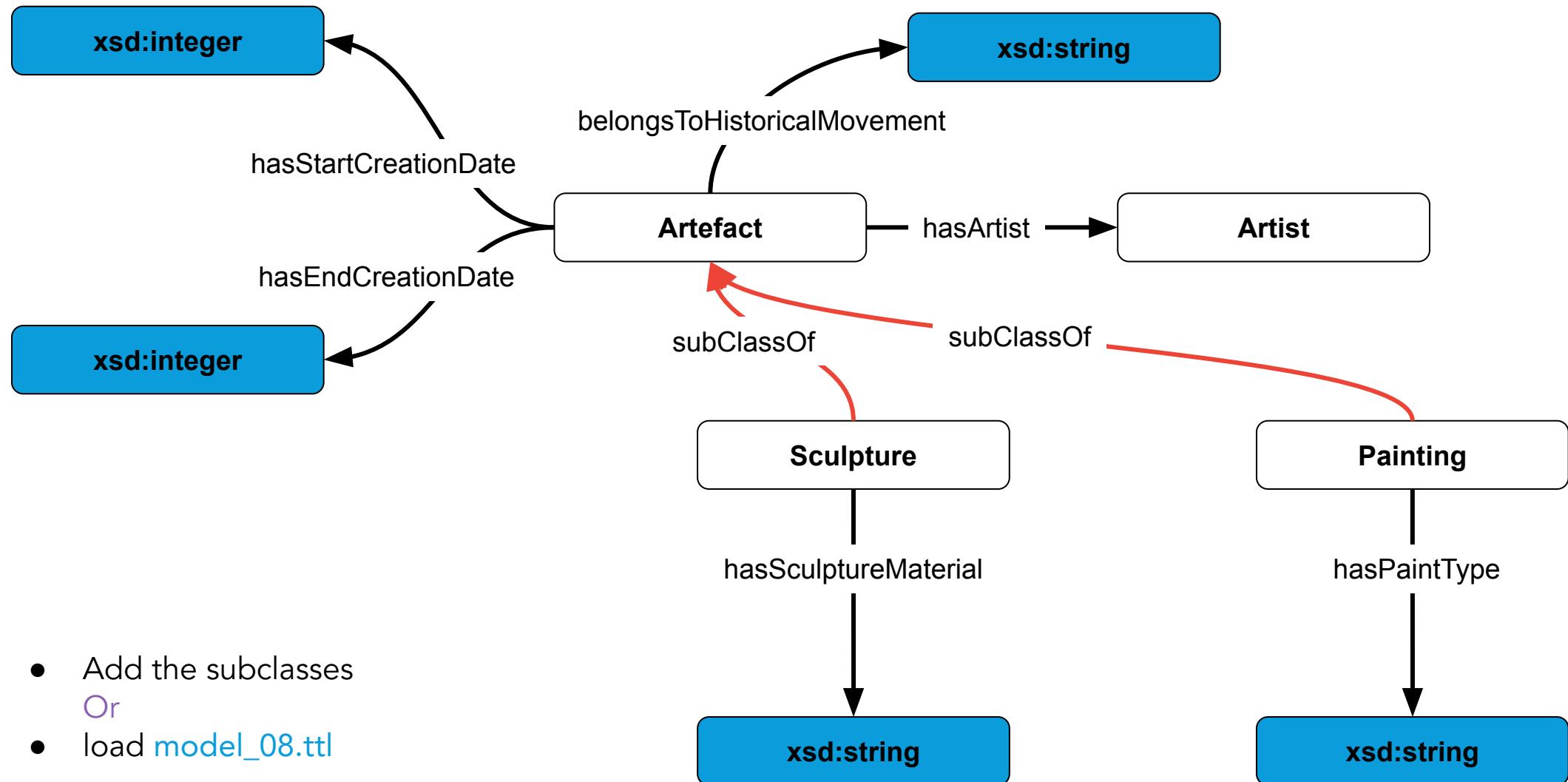
x
ex:mona_lisa
ex:the_last_supper



We did not get "David"!

Why? There's no subclass relationship defined between the sculpture class and the Artefact class.

# Extend the Model



- Add the subclasses  
Or
- load [model\\_08.ttl](#)



# The Subclass inference



- Run [query 2.4](#)
- Or
- load [model\\_09.ttl](#)
- Run [query 2.5](#)

RESULTS

x
ex:mona_lisa
ex:the_last_supper
ex:david

## Query 2.4

```
SELECT DISTINCT ?x
WHERE {
  ?x rdf:type/rdfs:subClassOf*
  ex:Artefact .
}
```

## Query 2.5

```
SELECT DISTINCT ?x
WHERE {
  ?x rdf:type ex:Artefact .
```

Inference/reasoning logic in Query (inference on read)

Inference/reasoning logic materialised in the knowledge graph (inference on write)



# Inference for Classification

One of the main reasons to use OWL is its capability of making [inferences](#)

Inference can happen based on:

- Definition of the class through its properties and restrictions
- Specifying the domain and range of a property
- Through the subclass/superclass relationship

On its own, the W3C standard SHACL-Core [does not](#) support inference

NOTE: For inference, please see SHACL Advance Features (W3C draft document)

So, how do  
Restrictions in  
OWL & SHACL  
differ?

# OWL & SHACL Restrictions Compared

In **OWL**, the restriction purpose is **Classification** of resources based on the values they have:

- If something has these properties and restriction, then it is of this class type, as we saw in the Sculpture example- ( in the implementation you have to define these restrictions as necessary and sufficient .
- OWL is **property-oriented**, meaning that the definition of a class is based on its properties and the restrictions on those properties

In **SHACL**, the restriction purpose is for **Validation**, and the logic is reverse meaning:

- If something is of a class/target type, then it has these properties
- And, if it doesn't have those properties, it will fail the **validation**

# Recap so far...

Open world: missing information does not cause an error OWL

Inference for Classification: OWL

- Class definition

- Property domain and range

- Subclass relationship

Closed world: missing information causes an error SHACL

- Data Integrity: SHACL

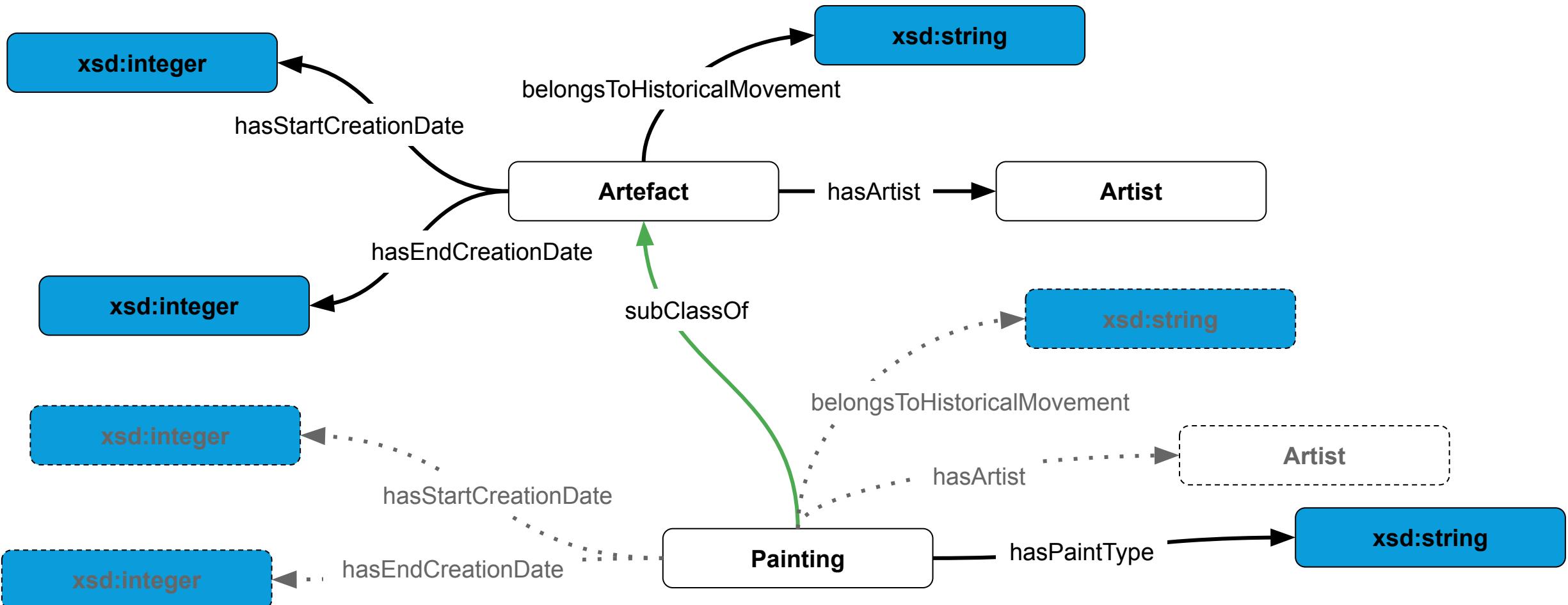
- Data Validation: SHACL

# ROUND 5: Inheritance



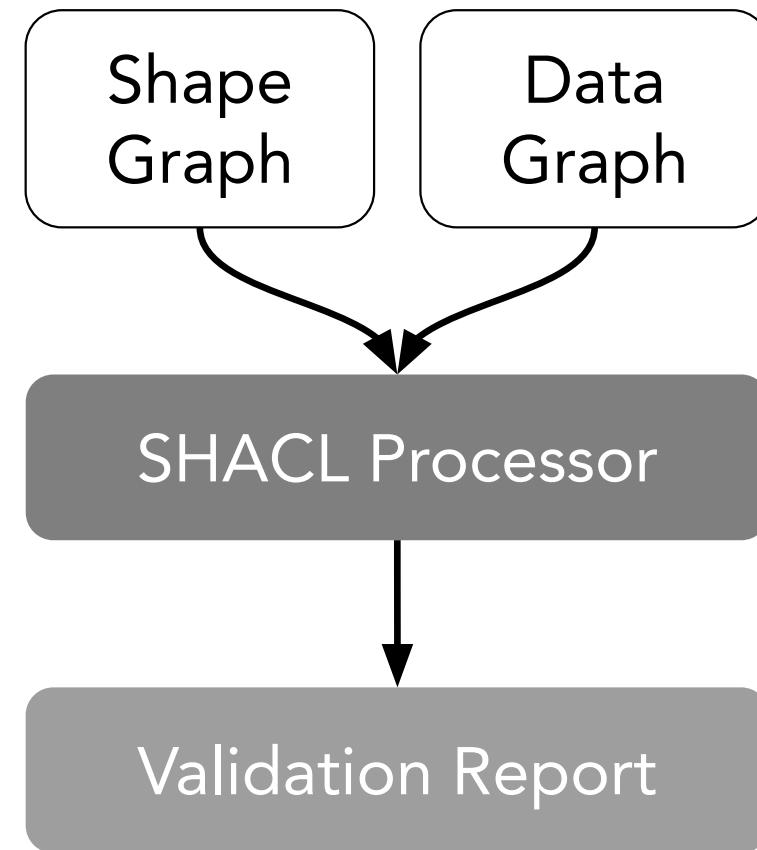
# Inheritance in OWL

Subclass superclass inheritance is native to owl what it means :



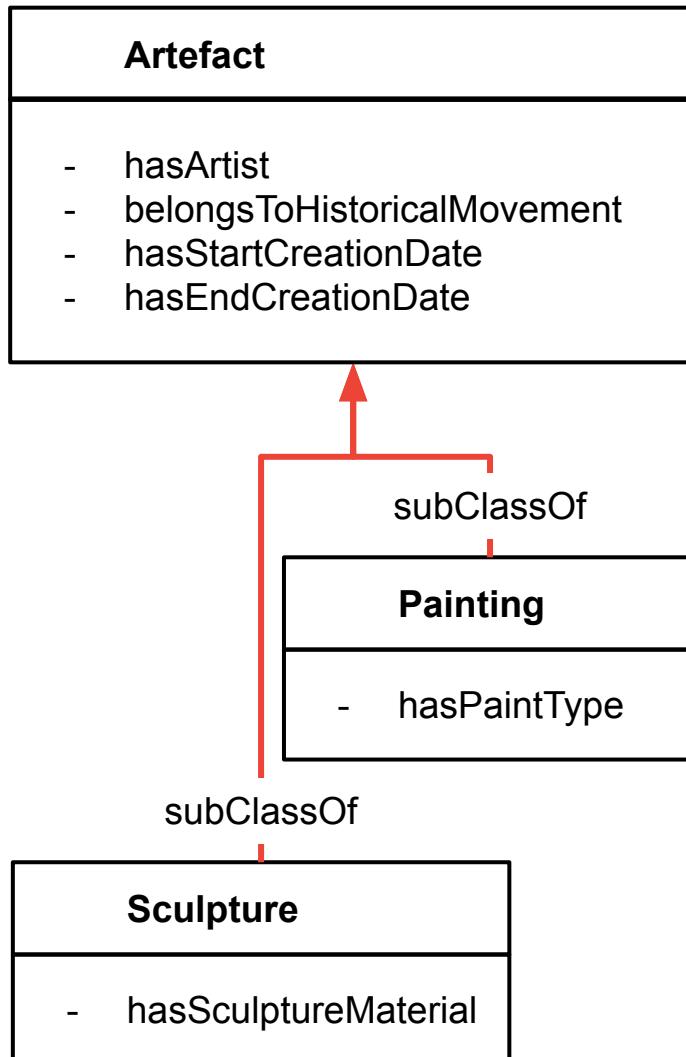
# Validation using inheritance in SHACL

# Validating with SHACL



Let's test our shape with the data in SHACL Playground

# subClass inheritance



- Run [Shape 3](#)
- The data graph conforms to the shape, hence, no error
- If we change one of the date from integer to string

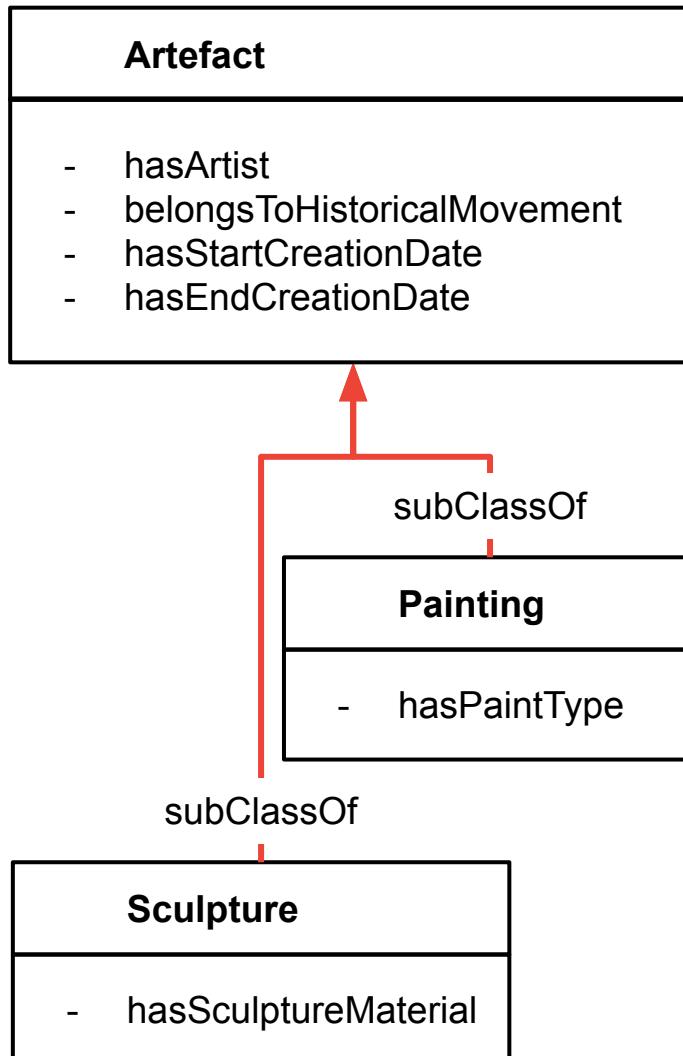


Still no Error!

The expectation is to inherit all properties of the super class Artefact -> but that didn't happen!

**WHY?** Defining the subclass relationship in SHACL is different from the one in OWL, let's dive deeper into this

# Making subClass inheritance work in SHACL



There are three ways to make it work:

1. Adding all the inference materialised in the data graph - [Shape 4](#)
2. Adding the semantic/knowledge level to the data graph - [Shape 5](#)
3. Adding `sh:node` on the SHACL side - [Shape 6](#)

If OWL subclassing says "*Every instance of A is also an instance of B*", the SHACL `sh:node` says "*Every A must have data that conforms to shape B*"

# SHACL: Closed Shape

By default, SHACL shapes are [open](#), which means:

- A node can have additional properties [beyond](#) those specified in the shape.
- SHACL [only](#) checks constraints on the properties you define; it [ignores](#) others.

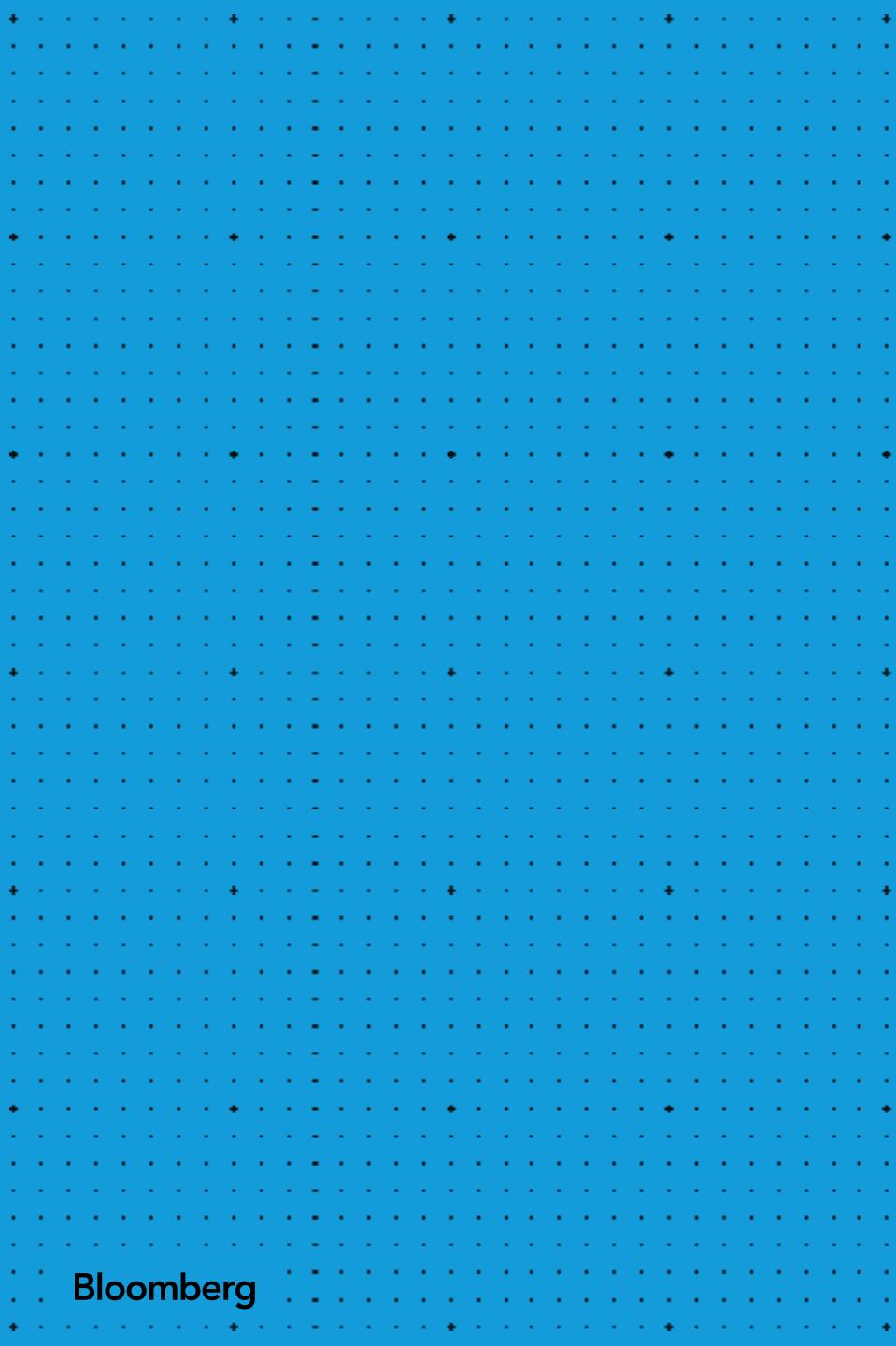
However, SHACL also allows you to define [closed shapes](#)

- A node must [only](#) have the properties I explicitly permit here, any other properties are [invalid](#).

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;
  sh:closed true ;
  sh:ignoredProperties (rdf:type rdfs:label) ;
```

But when you close the shapes then...

- With semantic/knowledge level + [sh:node](#) | [Shape 7](#)
- With materialized data + [sh:node](#) | [Shape 8](#)
- With everything | [Shape 9](#)



What if we need  
different data  
restrictions for  
different  
applications?

# ROUND 6: Multiple Shapes of Data



# One model, multiple shapes

Imagine another museum using the same ontology, but with an additional requirement: they need to ensure that the current owner or seller of an artefact is captured at the time of acquisition.

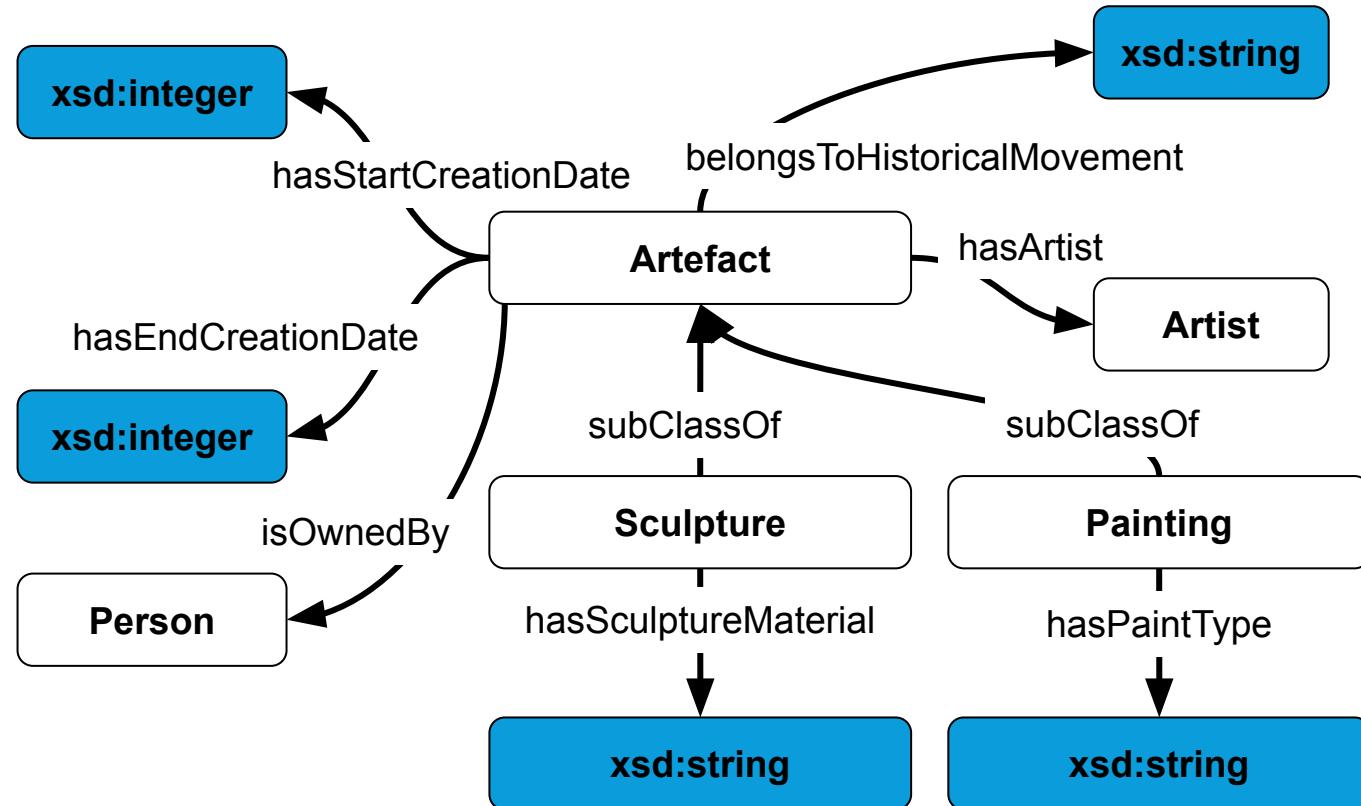
Does this mean they must create a completely new ontology? [Not necessarily](#)

- Extend the current ontology: Add properties such as `isOwnedBy` and classes like `Person`.
- Define a new shape: Define a different SHACL shape that validates their specific data needs for their use case

```
ex:Artefact  
  ex:isOwnedBy ex:Person .
```

```
sh:property [  
  sh:path ex:isOwnedBy ;  
  sh:class ex:Owner ;  
  sh:minCount 1 ;  
  sh:maxCount 1 ;  
] ;
```

# One model, multiple shapes

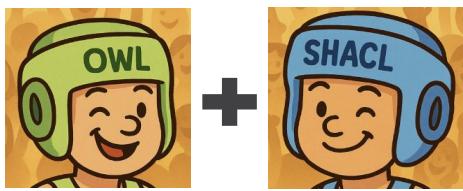


SHACL shape museum 1

[Shape 10](#)

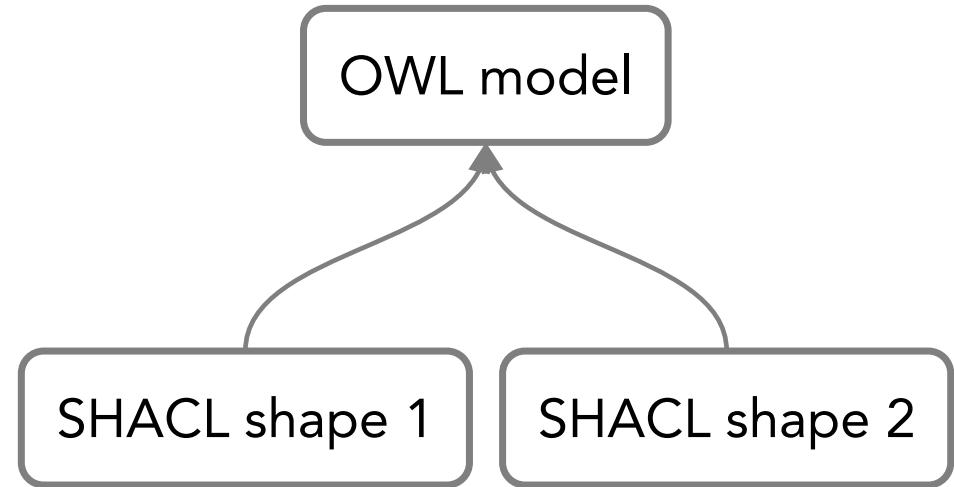
SHACL shape museum 2

[Shape 11](#)



# One model, multiple shapes

This ability to define **multiple SHACL shapes over the same ontology** supports **flexibility and interoperability**, allowing different institutions to share and validate data according to their own requirements while still adhering to a **common semantic foundation**.



# Recap so far...

Open world: missing information does not cause an error OWL

Inference for Classification: OWL

Class definition

Property domain and range

Subclass relationship

Closed world: missing information causes an error SHACL

Data Integrity: SHACL

Data Validation: SHACL

Class composition using `sh:node: SHACL`

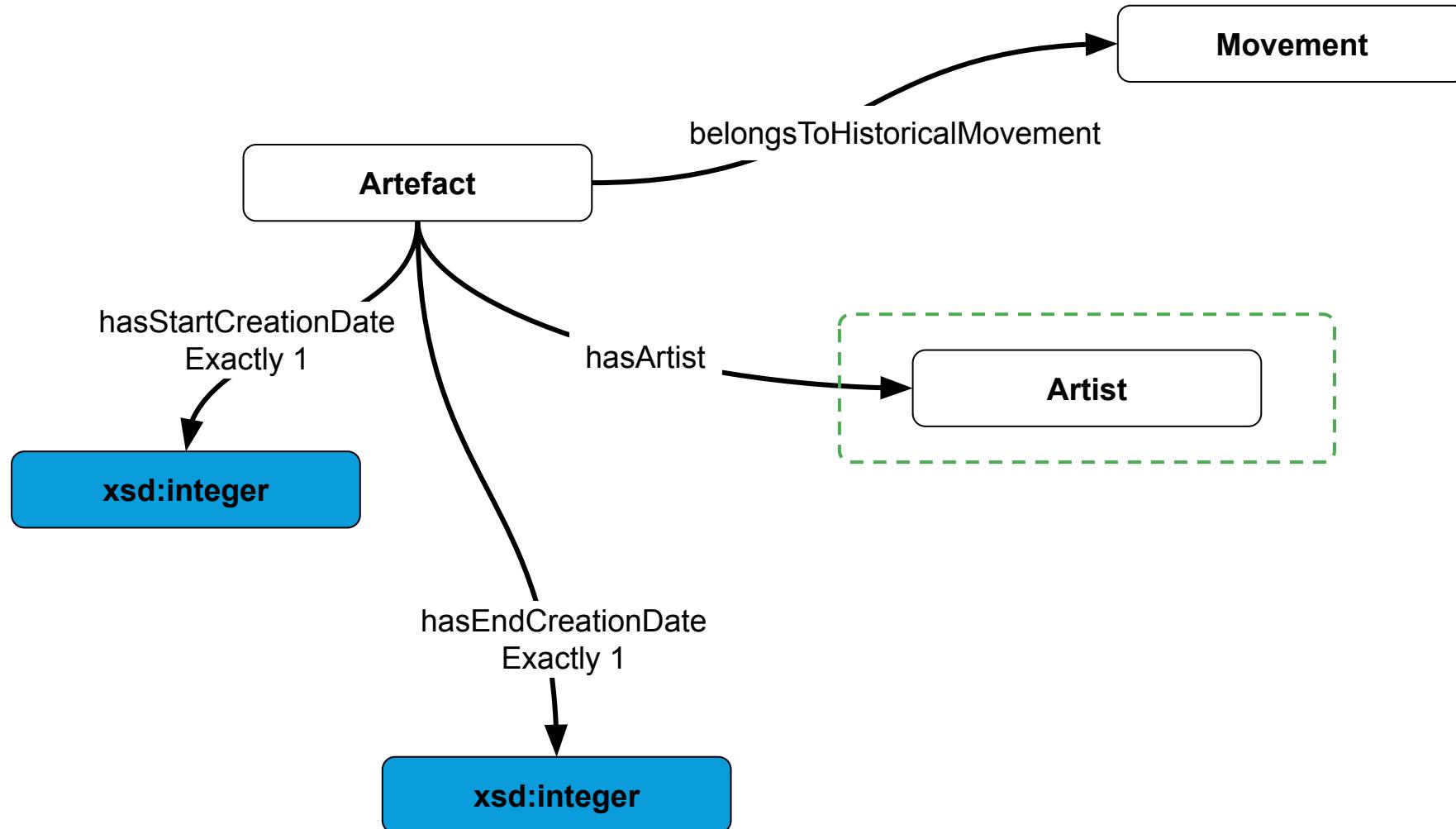
Multiple shapes for the same model: `OWL + SHACL`

What if want to  
ensure my art  
pieces are  
definitely created  
by an artist?

# ROUND 7: Constraints



I want to ensure every art piece is only created by an artist



# Deep dive in the property definition

Can I use domain and range of ex:hasArtist?



- Open [model\\_10.ttl](#)
  - Adding wrong artist (`hasArtist` Cubism)  
Or
  - open [model\\_10\\_wrong\\_artist.ttl](#)
  - Run the reasoner
- 
- No error even though Cubism is wrong
  - Cubism classified as an Artist! (remember inference?)



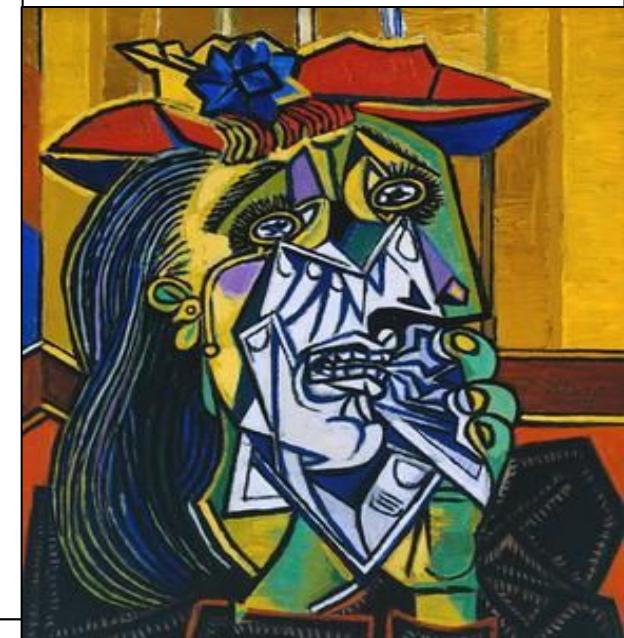
WHY? 'Violating' a domain or range constraint **does not** necessarily mean that the ontology is inconsistent or contains errors.

So what should we do ?

OWL solution: Define the two classes, Movement and Artist, as [disjoint](#)

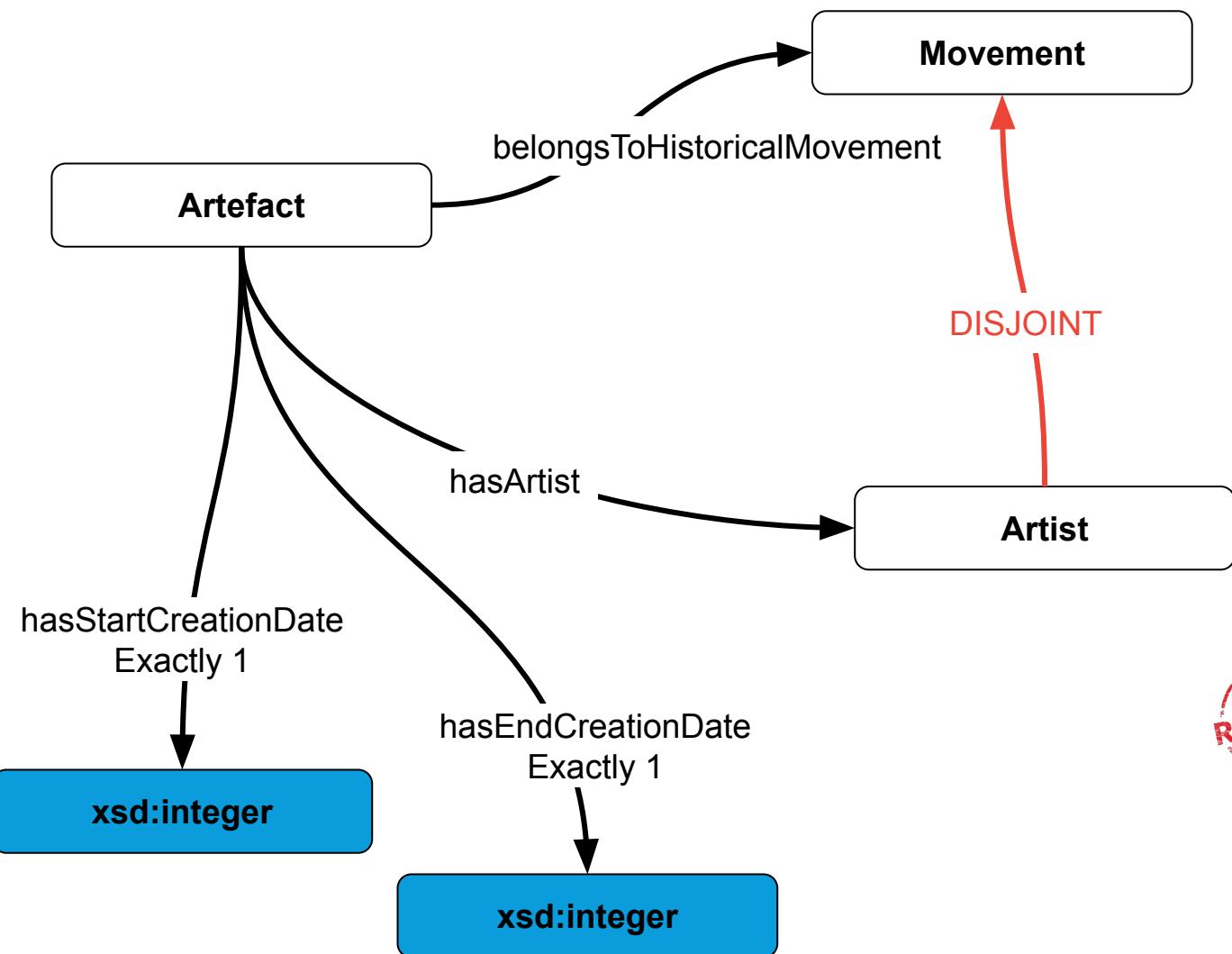
## Fun Fact:

*The Weeping Woman*, valued at A\$1.6M in 1986. It was stolen from the National Gallery of Victoria by a group calling themselves the Australian Cultural Terrorists (ACT), who demanded funding for the arts and the creation of a \$25,000 annual art prize (The Picasso Ransom) or they would destroy the painting. The government refused to negotiate, and police were informed three weeks later that *The Weeping Woman* was found, stowed in a locker, unharmed.

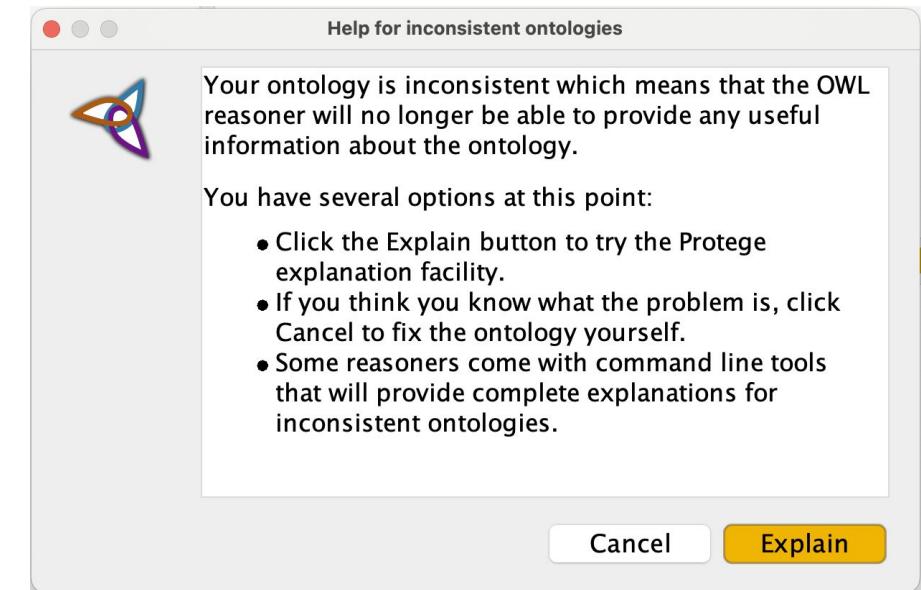


Source: [wikipedia](#)

# Disjoint Classes



- Make Artist and Movement disjoint  
Or
- open [model\\_10\\_disjoint.ttl](#)
- Run the reasoner





# Defining Constraints on your data with SHACL

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;

  sh:property [
    sh:path ex:belongsToHistoricalMovement ;
    sh:datatype xsd:string ;
  ] ;

  sh:property [
    sh:path ex:hasArtist ;
    sh:class ex:Artist ;
  ] ;

  sh:property [
    sh:path ex:hasEndCreationDate ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;

  sh:property [
    sh:path ex:hasStartCreationDate ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] .
```



- Open [Shape 12](#)
- Change the value of the scream  
ex:hasArtist to cubism

This restricts the values of ex:hasArtist to Class Artist



Error ! restricting the range of ex:hasArtist to the class Artist throws an error when any other value is defined as the range in the data graph

This ensure that every Painting has exactly 1 Start and End Creation Date

# SHACL Constraints

## Constraints

### Shape Based

- Node Shape
- Property Shape

### String Based

- sh:minLength
- sh:maxLength
- sh:pattern
- sh:languageIn
- sh:uniqueLang

### Value Type

- sh:Class
- sh:datatype
- sh:nodeKind

### Value Range

- sh:minExclusive
- sh:maxExclusive
- sh:minInclusive
- sh:MaxExclusive

### Logical

- sh:not
- sh:and
- sh:or
- sh:xone

### Property Pair

- sh>equals
- sh:disjoing
- sh:lessThan
- sh:lessThanOrEauals

### Cardinality

- sh:minCount
- sh:maxCount

# Recap so far...

Open world: missing information does not cause an error OWL

Inference for Classification: OWL

Class definition

Property domain and range

Subclass relationship

Closed world: missing information causes an error SHACL

Data Integrity: SHACL

Data Validation: SHACL

Restrictions/Constraints for Validation: [SHACL](#)

Class composition using sh:node: SHACL

Multiple shapes for the same model: OWL + SHACL

# ROUND 8: Inference



# Inference for New (implicit) Knowledge

# Inferring New Knowledge

You have the historical movement for your artefacts, does that allow you to also know which artist belongs to what movement?



- Open [model\\_11.ttl](#)
- Run the [query 3.1](#)

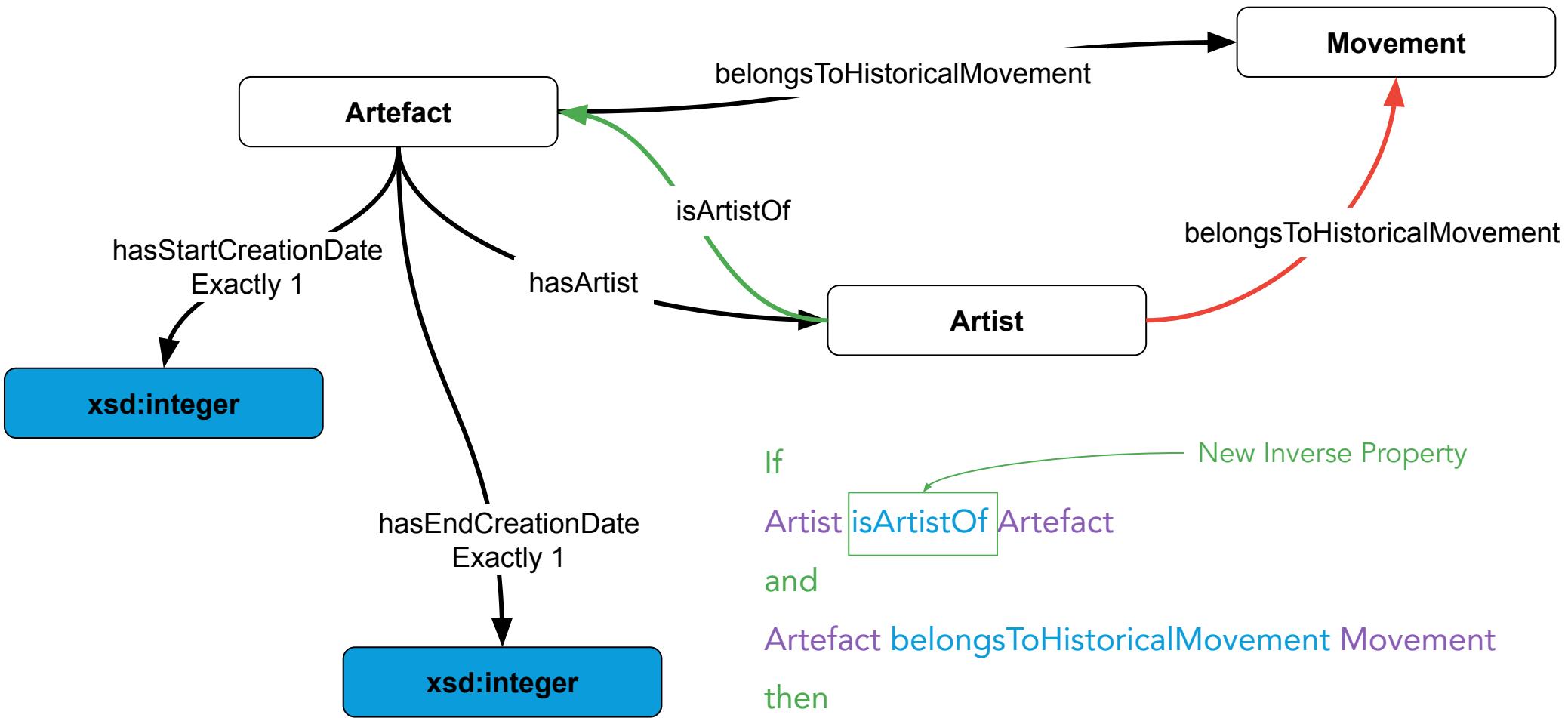


Nothing!

```
SELECT DISTINCT ?Artist ?Movement  
  
WHERE {  
    ?Artist a ex:Artist ;  
    ex:belongsToHistoricalMovement ?Movement .  
}
```

Possible solution: Property Chain

# The Model extended even more!



# Inverse Properties in OWL



- First, add `isArtistOf` inverse property
- Implement the property chain  
Or
- load `model_12.ttl`
- Run the reasoner



Reasoner infers the implicit Knowledge

Property assertions: Leonardo Da Vinci

Object property assertions +

<code>belongsToHistoricalMovement</code>	Renaissance	?	@
<code>isArtistOf</code>	'Mona Lisa'	?	@
<code>isArtistOf</code>	'The Last Supper'	?	@

# Property chain in OWL (owl:propertyChainAxiom)



- Save the new inferred data  
Or
- upload [model\\_12\\_inferred.ttl](#)
- Then run the query 3.1 again

```
SELECT DISTINCT ?Artist ?Movement
WHERE {
    ?Artist a ex:Artist ;
    ex:belongsToHistoricalMovement ?Movement .
}
```



Artist	Movement
Leonardo Da Vinci	Renaissance
Edvard Munch	Expressionism
Pablo Picasso	Cubism
Sandro Botticelli	Renaissance
Michelangelo	Renaissance

# Inferring New Knowledge

Considering that you know the museum in which the piece is exhibited, can you answer questions like:

Give me all the pieces that are located in a country (i.e., Italy)?

In order to answer to questions like this, we need to add further knowledge into our KG:

If

Artefact *isExhibitedIn* Premises

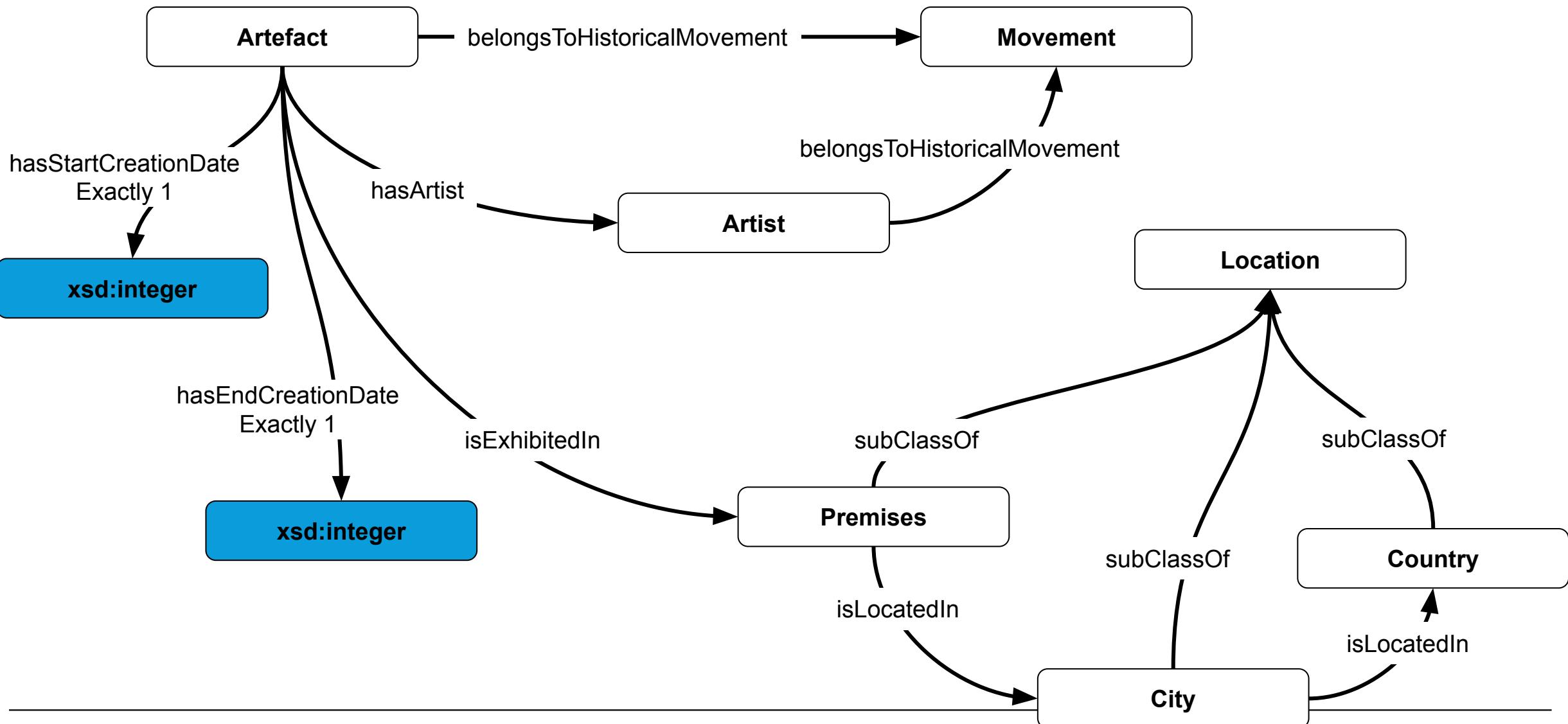
and

Premises *isLocatedIn* City

then

Artefact *isLocatedIn* City

# The model extended even more!



# Inferring New Knowledge



- Add new classes and the property chain  
Or
- Load [model\\_13.ttl](#)
- Run reasoner



Reasoner infers the implicit Knowledge based on property chain definition



- Save the new inferred data  
Or
- Upload [model\\_13\\_inferred.ttl](#)
- Run query 3.2



Nothing!  
Why? We are still missing the country

Property assertions: David

Object property assertions [+](#)

<b>isExposedIn</b>	'Gallery of the Academy of Florence'	<a href="#">?</a>	<a href="#">@</a>	<a href="#">x</a>	<a href="#">o</a>
<b>hasArtist</b>	Michelangelo	<a href="#">?</a>	<a href="#">@</a>	<a href="#">x</a>	<a href="#">o</a>
<b>belongsToHistoricalMovement</b>	Renaissance	<a href="#">?</a>	<a href="#">@</a>	<a href="#">x</a>	<a href="#">o</a>
<b>isLocatedIn</b>	Florence	<a href="#">?</a>	<a href="#">@</a>		

**SELECT DISTINCT** ?Artefact

**WHERE** {

?Artefact a ex:Artefact ;

ex:isLocatedIn ex:italy .

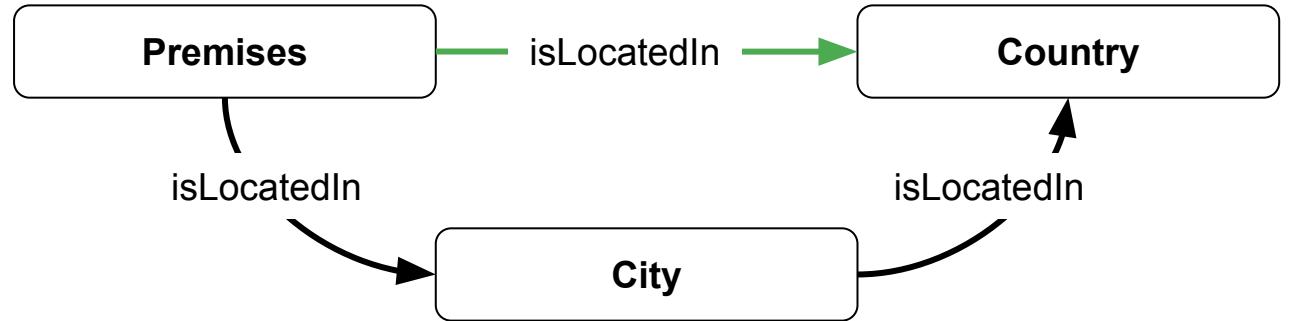
}



# Transitivity in properties for inference of New Knowledge

How can we ensure that if something is located in a city then that something is located in the country of that city?

**Transitivity** automatically infers that if a property links  $A \rightarrow B$  and  $B \rightarrow C$ , then  $A \rightarrow C$  for that same property so we can say



`owl:propertyChainAxiom` allows defining a new property that results from a sequence of different properties (e.g.,  $A \rightarrow B$  via `isExhibitedIn` and  $B \rightarrow C$  via `isLocatedIn`  $\Rightarrow A \rightarrow C$  via `isLocatedIn`).

`isLocatedIn` is transitive property. Then Artefact `isLocatedIn` Country

# Transitivity in properties for Inference of New Knowledge



- Make `isLocatedIn` transitive  
Or
- Open `model_14.ttl`
- Run reasoner

Reasoner infers the implicit Knowledge based on property chain definition and transitivity

- Save the new inferred data  
Or
- upload `model_14_inferred.ttl`
- Run query 3.2 again



You can now see all artefacts located in Italy



Property assertions: David

Object property assertions +

isExhibitedIn 'Gallery of the Academy of Florence'	? @ ✘ ○
isLocatedIn Florence	? @ ✘ ○
hasArtist Michelangelo	? @ ✘ ○
belongsToHistoricalMovement Renaissance	? @ ✘ ○
isLocatedIn Italy	? @

```
SELECT DISTINCT ?Artefact
WHERE {
?Artefact a ex:Artefact ;
ex:isLocatedIn ex:italy .
}
```

# Recap so far...

Open world: missing information does not cause an error OWL

Closed world: missing information causes an error SHACL

- Data Integrity: SHACL
- Data Validation: SHACL
  - Constraints for Validation: SHACL

Inference for Classification: OWL

- Class definition
- Property domain and range
- Subclass relationship

Inference for new/implicit knowledge: [OWL](#)

Multiple shapes / data behaviors for the same model: OWL + SHACL

# The Material Example

# ROUND 9: Range Restrictions



# The Material Example

What are the materials used to make this artefact?

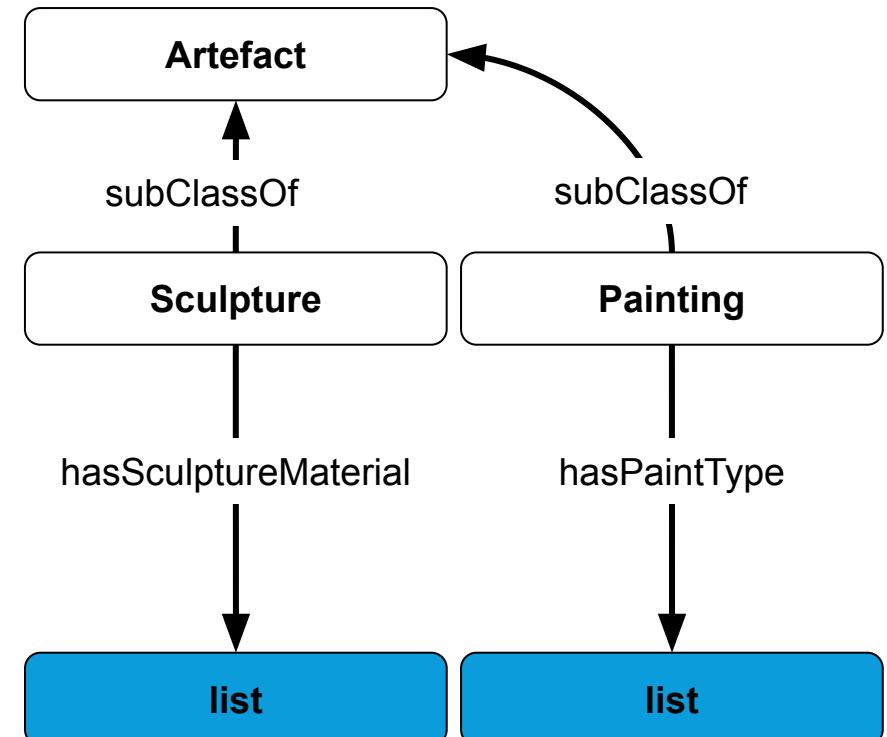
You also want to ensure that the range of allowed materials is previously captured.

Let's start with the model: [model\\_15.ttl](#)

**Choice 1:** use owl dataproperty and a list as the range

**Challenge:**

- What if you want to say something more about each material? Like its origin or consistency?



# The Material Example

What are the materials used to make this artefact?

**Choice 2:** Use OWL Enumeration class



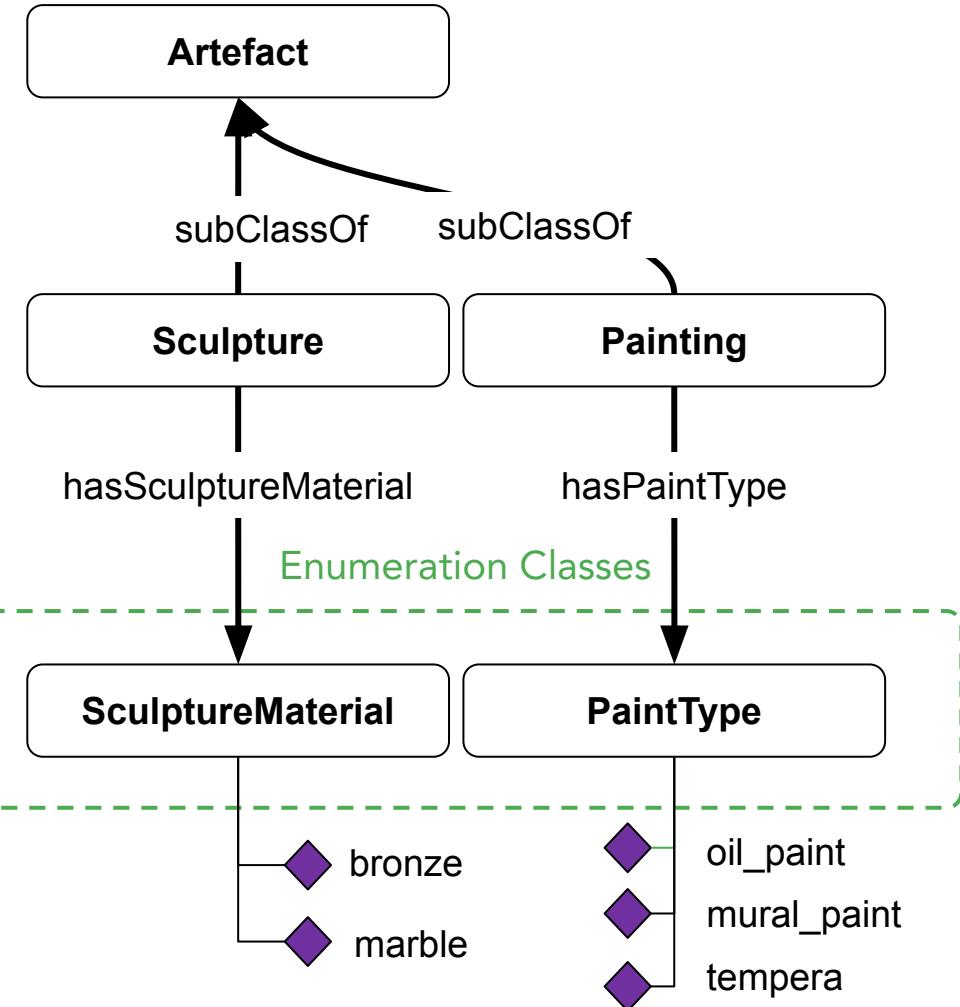
- Load [model\\_16.ttl](#)
- Add the following wrong data  
ex:weepingWoman ex:hasPaintType ex:florence
- Run reasoner



No Error!

Reasoner infers that florence is a member of *PaintType* class

**WHY?** Because of OWL's OWA



# The Material Example

What are the materials used to make this artefact?

**Choice 2:** Use OWL Enumeration class



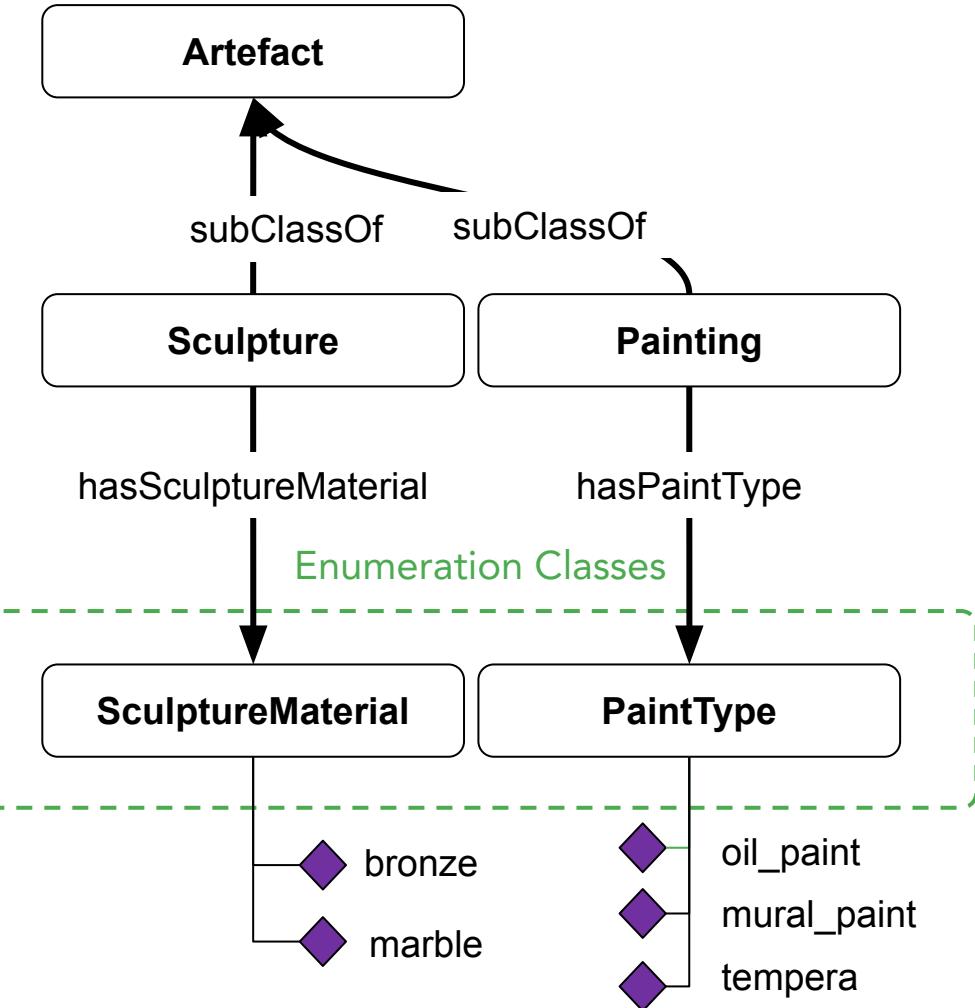
- What if we limit the range of *hasPaintType* to only one *PaintType*?
- Making the property *hasPaintType* Functional



No Error!

Reasoner infers that the instances are the same!

**WHY?** OWL does not support UNA



# Unique Name Assumption

In logics with UNA, different names always refer to different entities in the world

OWL does **NOT** support UNA

Consequences:

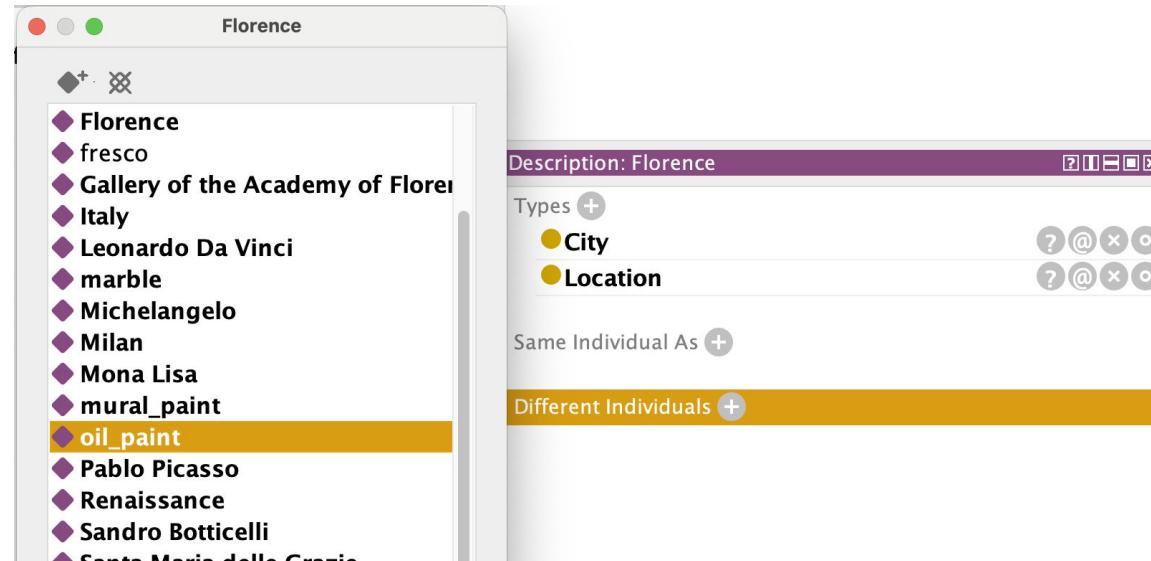
- Different entities have to be **declared to be different** (otherwise, they are potentially identical) - owl:differentFrom, owl:disjointWith
- Identical entities also have to be **declared to be identical** (otherwise, they are potentially different) - owl:sameAs, owl:equivalentClass

# Can we still solve this with OWL?

Yes, but...

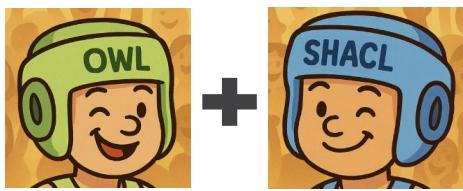
The property hasPaintType needs to be [Functional](#)

The instances need to all be defined as [owl:differentFrom](#)



Too much work!

There's a better solution :)



# The Material Example

What are the materials used to make this artefact?

## Choice 3:

- Use OWL enumeration classes & SHACL shapes to restrict the range of artefacts [Shape 13](#)

### Pros:

- no diff definition
- no wrong class inference
- easy expansion of material types
- Easy addition of information about material

```

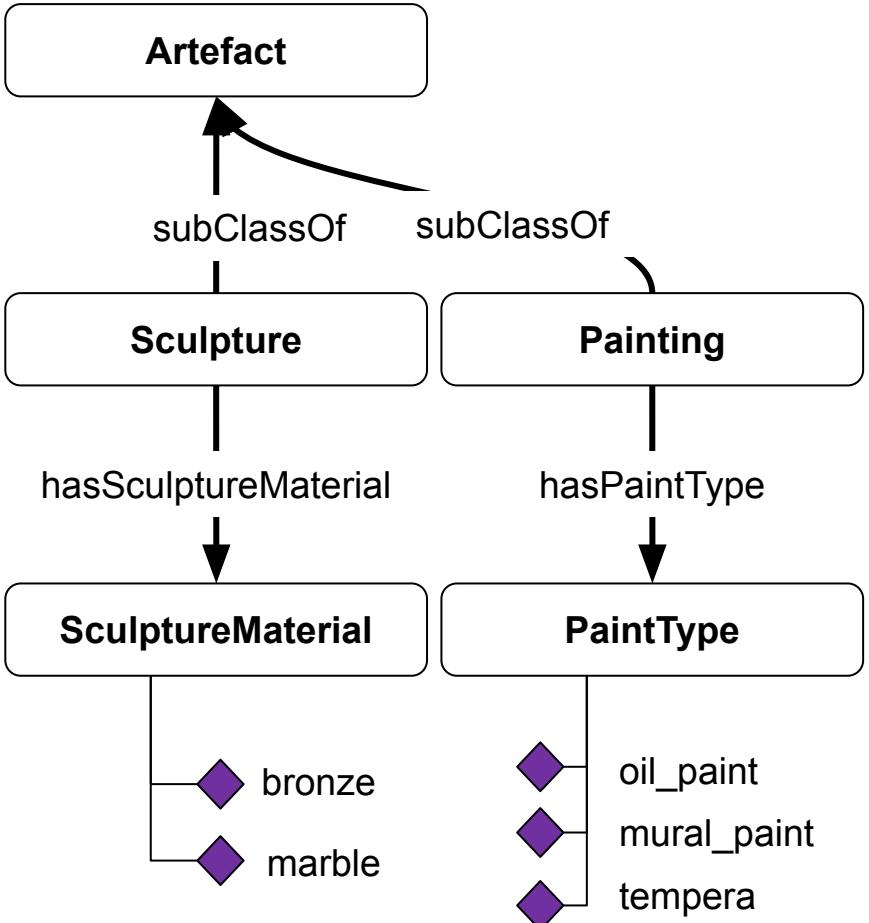
ex:PaintingShape a sh:NodeShape ;
sh:targetClass ex:Painting ;

sh:property [
  sh:path ex:hasPaintType ;
  sh:class ex:PaintType ;
  sh:in (ex:oil_paint ex:mural_paint ex:tempera) ;
] .

ex:SculptureShape a sh:NodeShape ;
sh:targetClass ex:Sculpture ;

sh:property [
  sh:path ex:hasSculptureMaterial ;
  sh:class ex:SculptureMaterial ;
  sh:in (ex:marble ex:bronze) ;
] .

```



# Full Recap

Open world: missing information does not cause an error **OWL**

Inference for Classification: **OWL**

- Class definition

- Property domain and range

- Subclass relationship

Inference for new/implicit knowledge: **OWL**

Closed world: missing information causes an error **SHACL**

- Validation for Data Integrity: **SHACL**

- Constraints for Validation: **SHACL**

Multiple shapes for the same model: **OWL + SHACL**

Mix of Inference and Validation: **OWL + SHACL**

# Final Round: Just a fun fact



# The Latest Louvre Heist !

## Speed-Run Art Theft

The crew pulled off the whole job in 7 minutes — with only 4 minutes inside the Louvre.

## Bob the Builder: Heist Edition

They showed up in **hi-vis construction vests**, carrying power tools, and scaled the museum using a moving lift before zipping away on scooters.

Which then the moving lift company used to advertise !

## Zero Bubble-Wrap

They stole French crown jewels from Empress Eugénie and other royals One piece was even dropped and damaged during the escape.



*Disclaimer: Created by ChatGPT*

And the winner  
is.....

# WINNER :YOU



*Disclaimer: Created by ChatGPT*

- Identify and understand the true problem you are trying to solve
- Understand the strengths and weaknesses of each technology and why and what it's built for
- Choose the right tooling and understand the strengths of your tool
- Based on the above **KNOWLEDGE** infer the **RIGHT CHOICE** !

# Questions ?



# Thank you!

## Let's discuss more

Connect with us on LinkedIn



Tara Raafat  
<https://www.linkedin.com/in/tara-raafat>



Davide D'Amico  
<https://www.linkedin.com/in/davide~damico>