Foundations of Databases

1st Tutorial

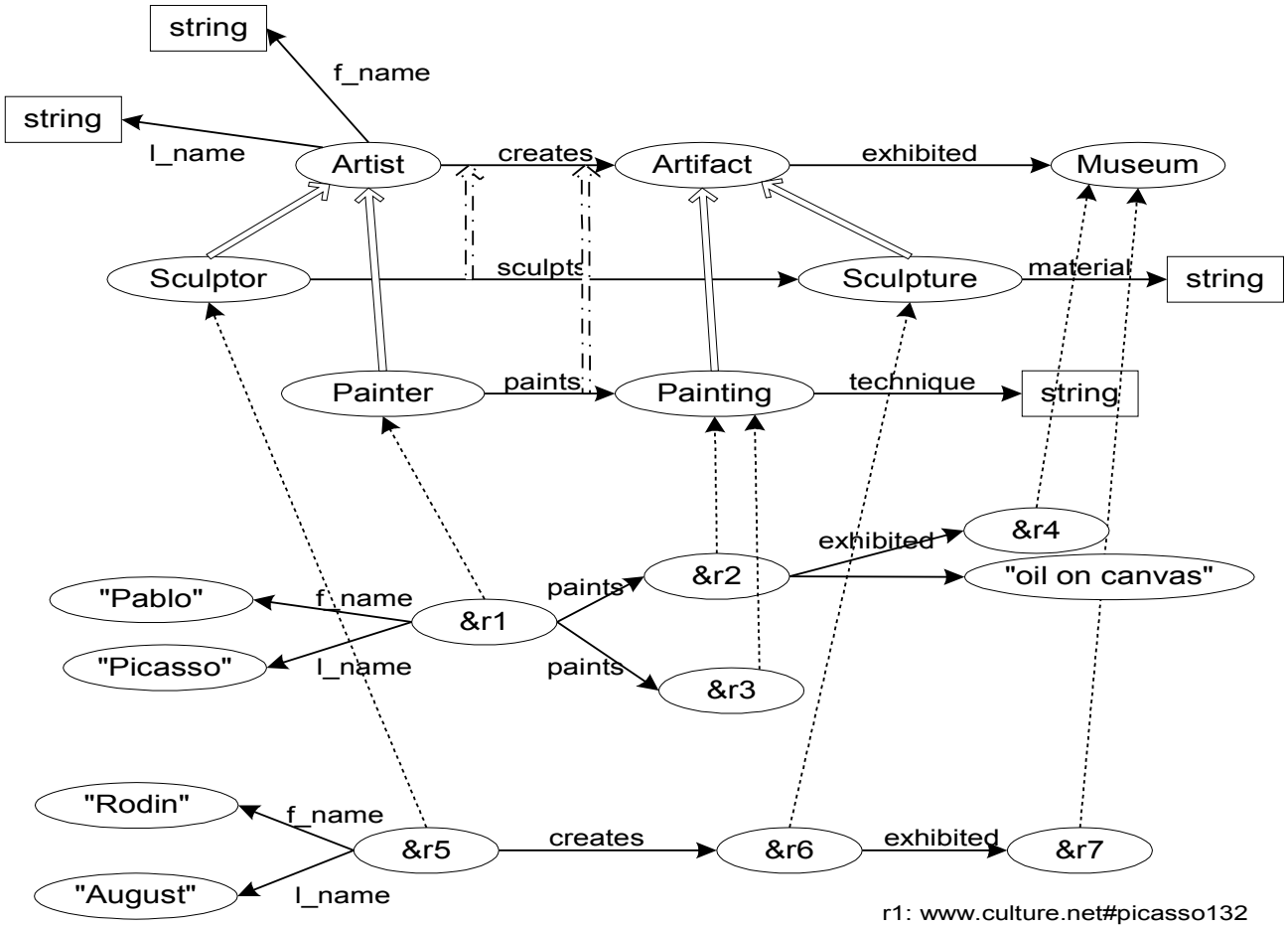
Kostis Kyzirakos

15/3/2010

RDF(S): basic elements

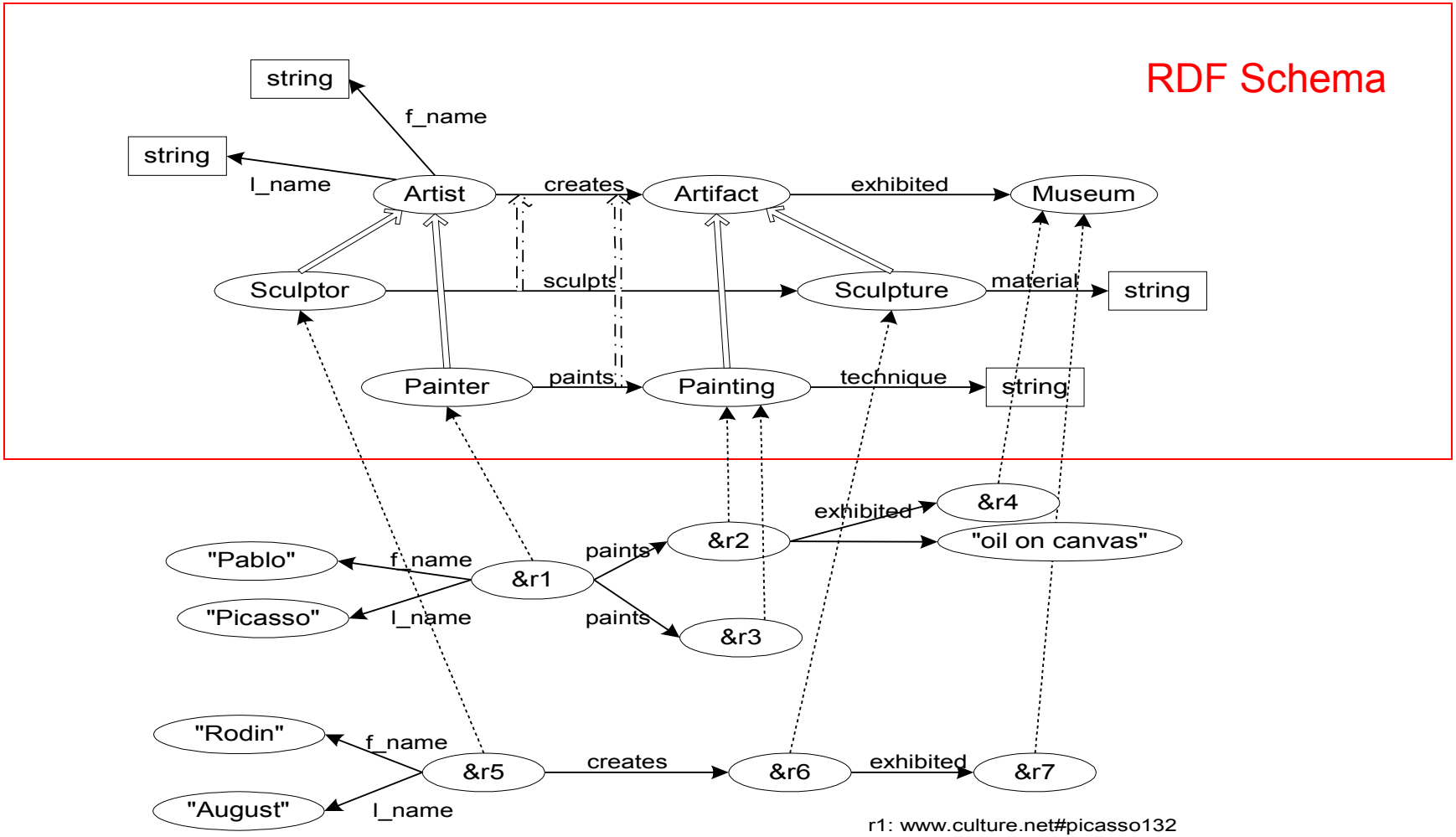
- Classes
  - Concepts of our world
  - Hierarchical description of concepts
- Properties
  - Relationships between concepts
  - Attributes of concepts
- Instances
  - Specific individuals of concepts
- Literals
  - Specific values such as numbers and dates

An RDF(S) example

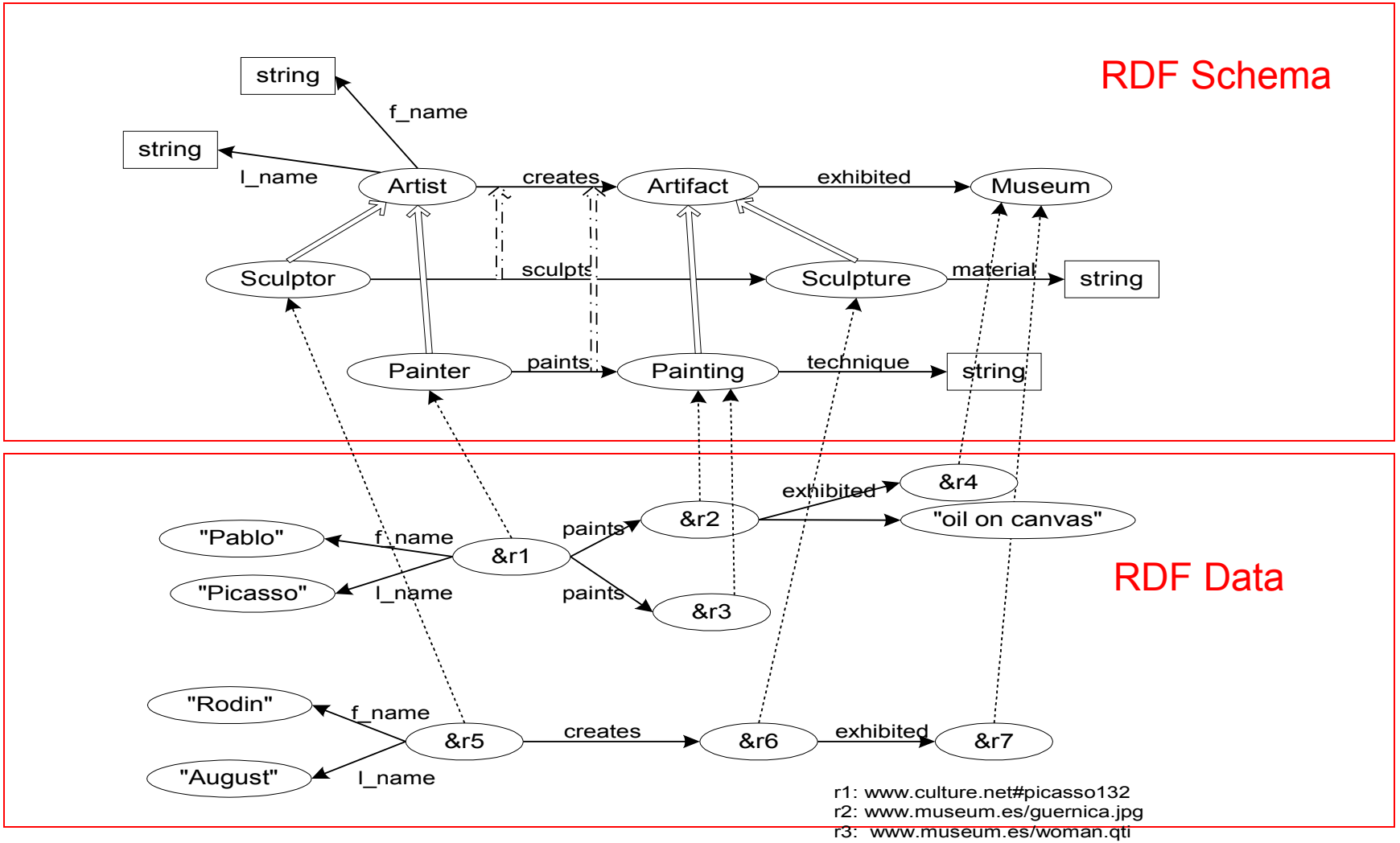


r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

An RDF(S) example

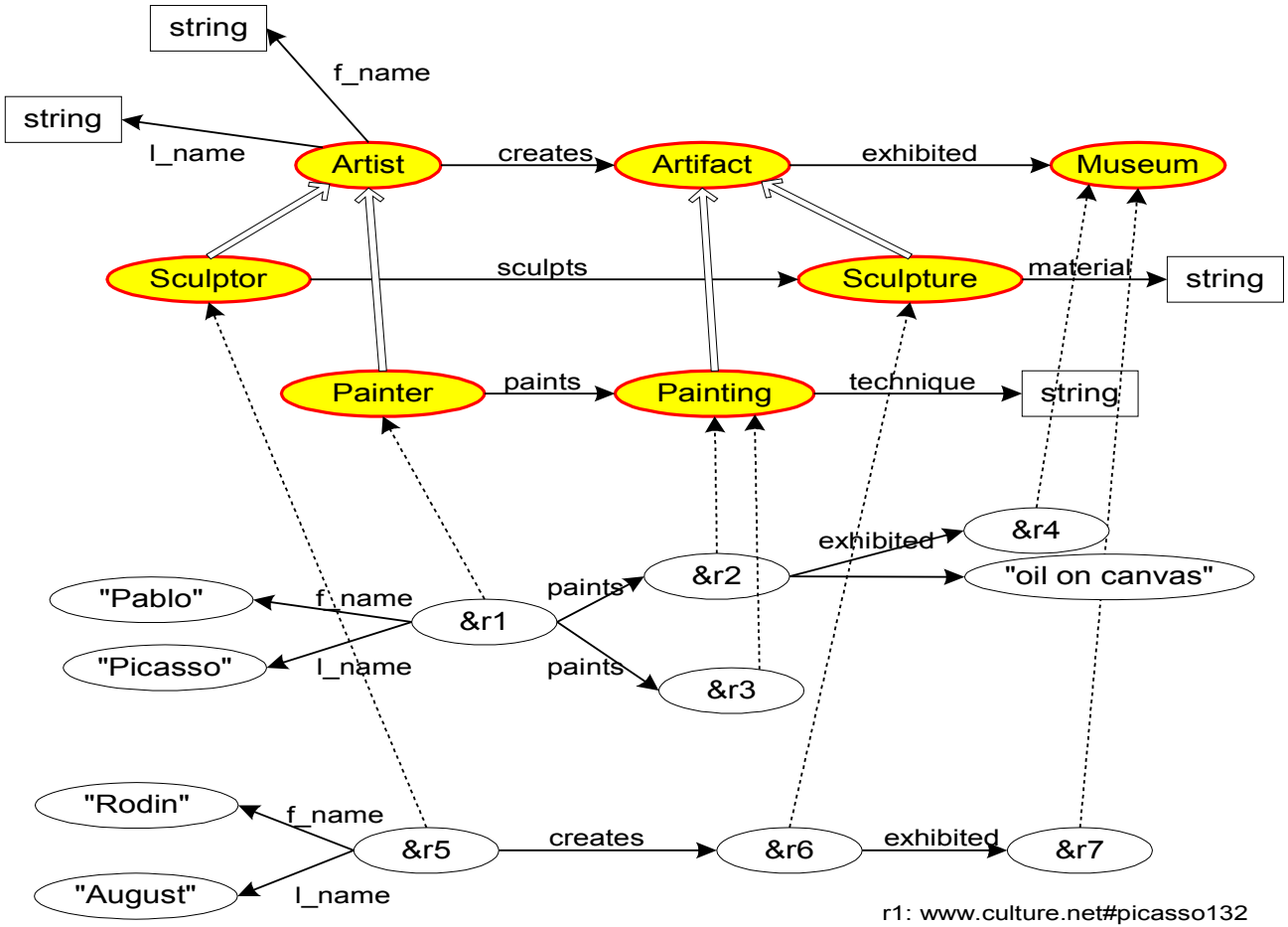


An RDF(S) example



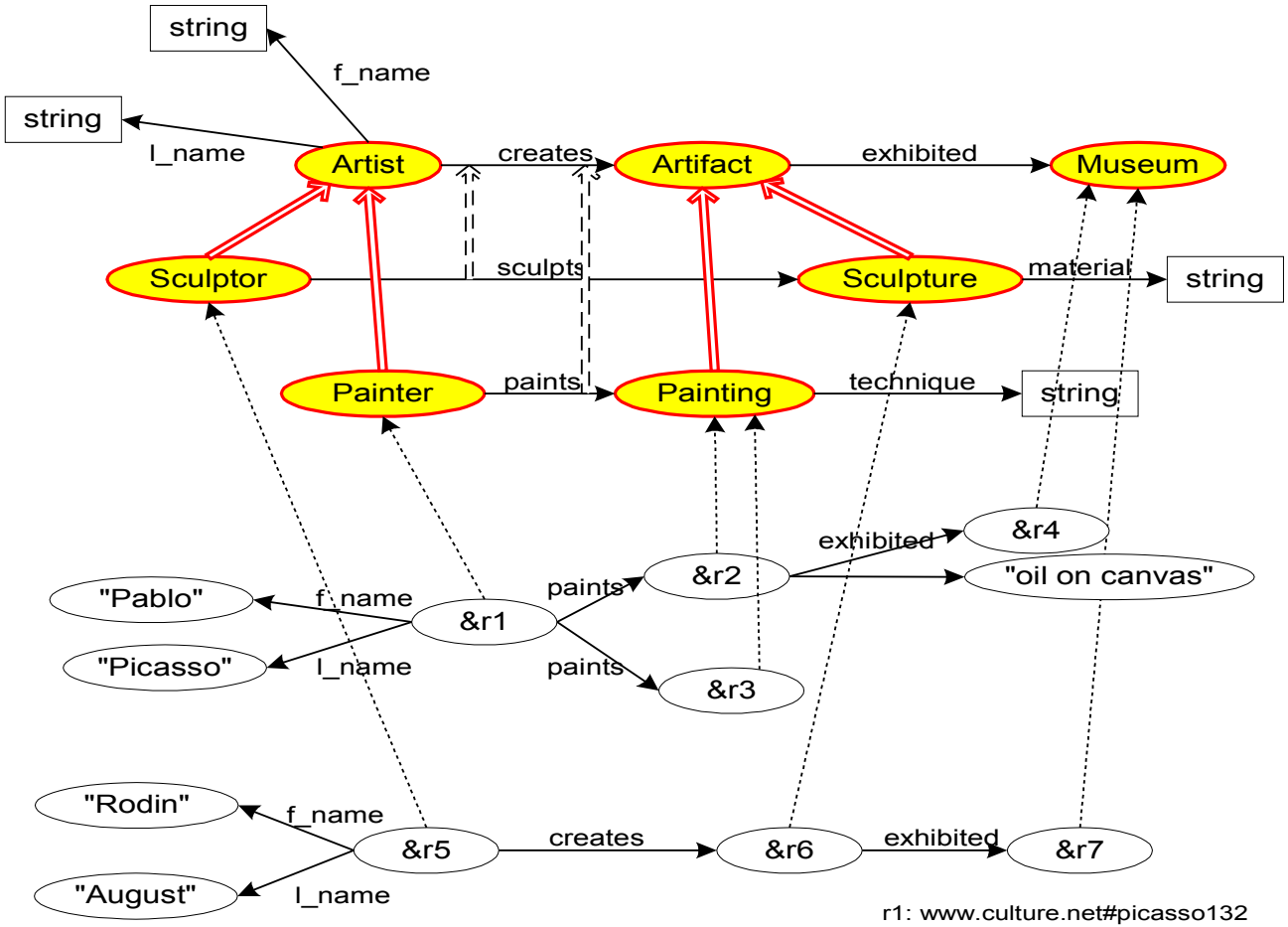
r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Classes



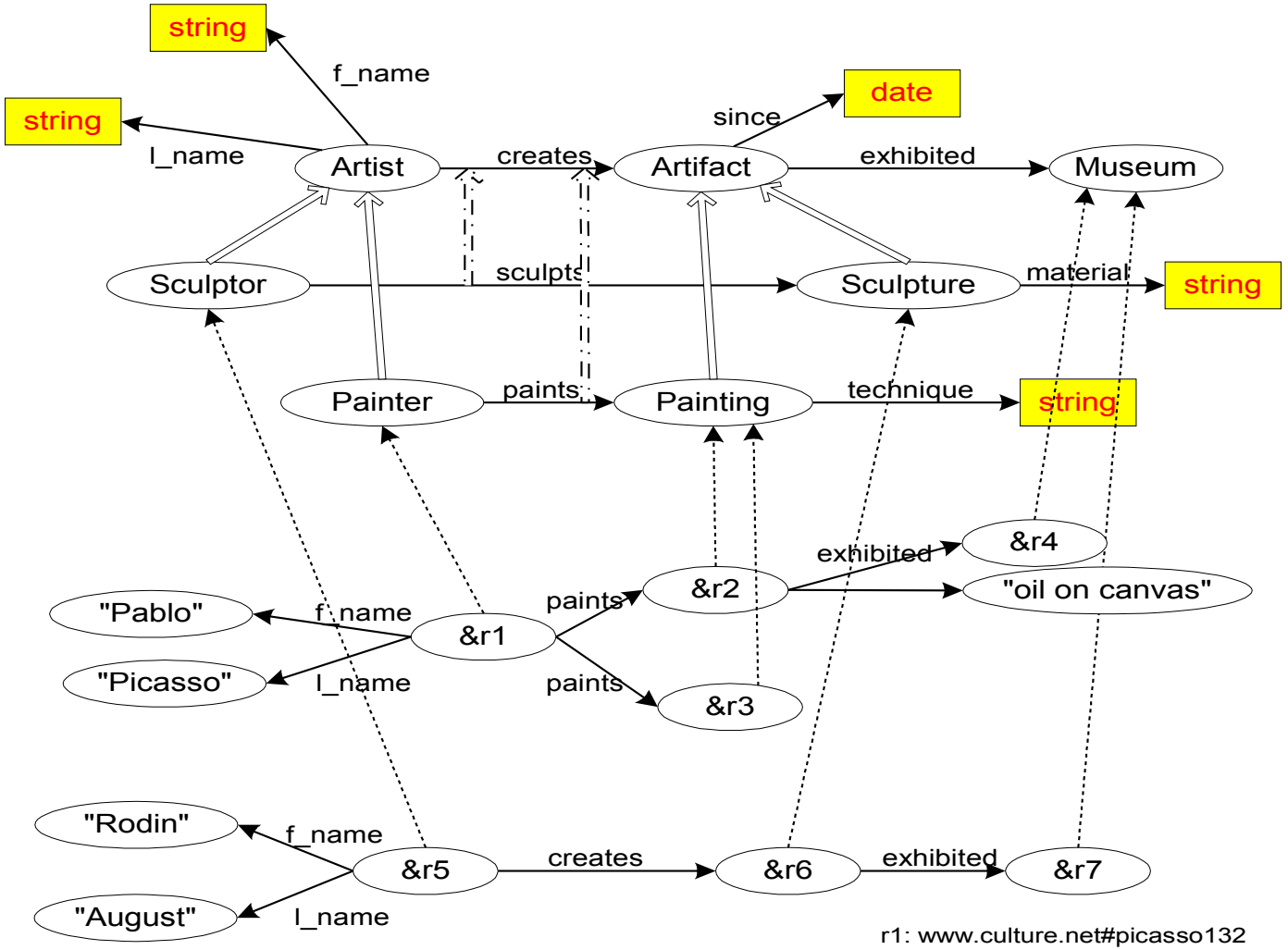
r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Subclass relation



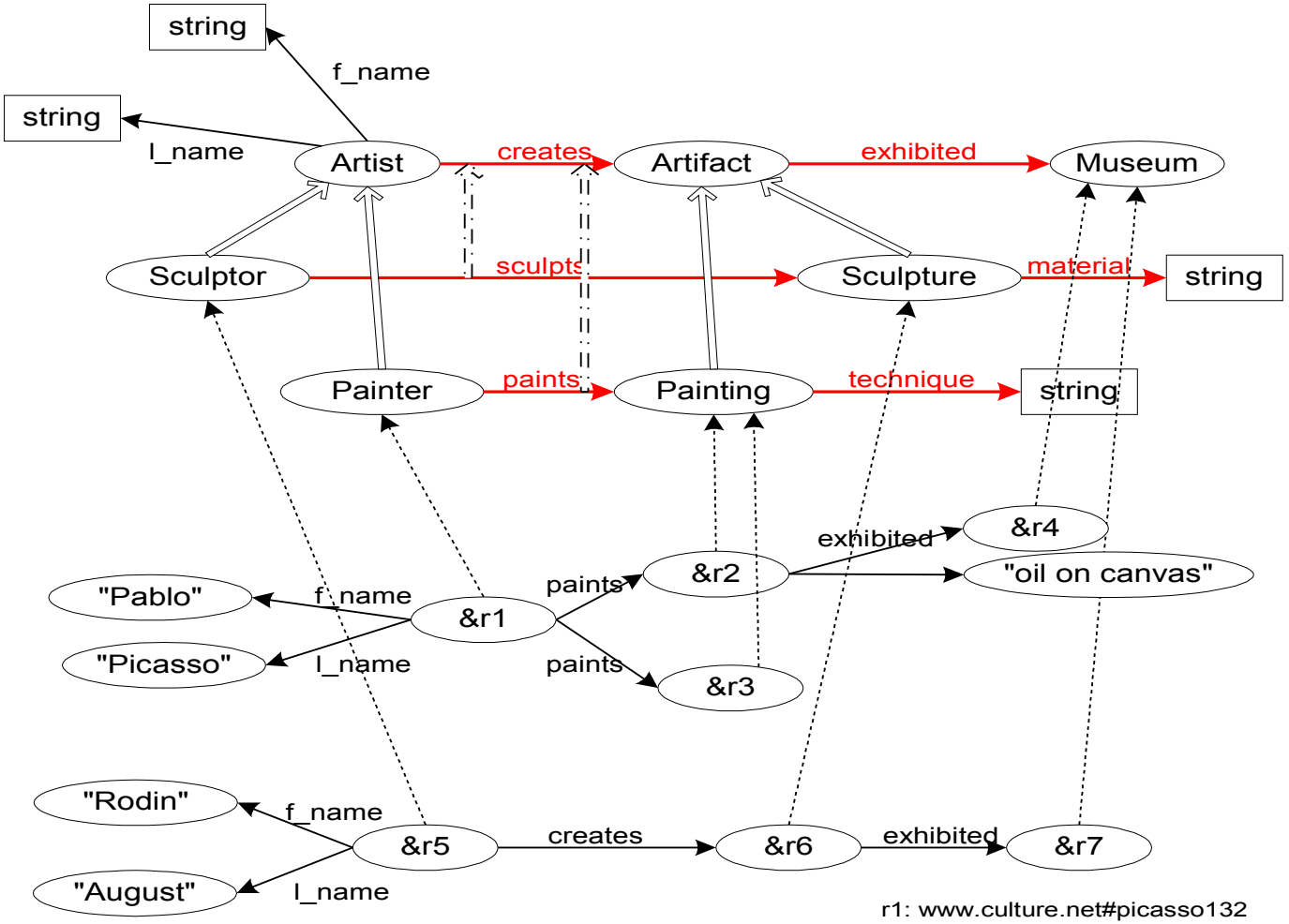
r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Literals



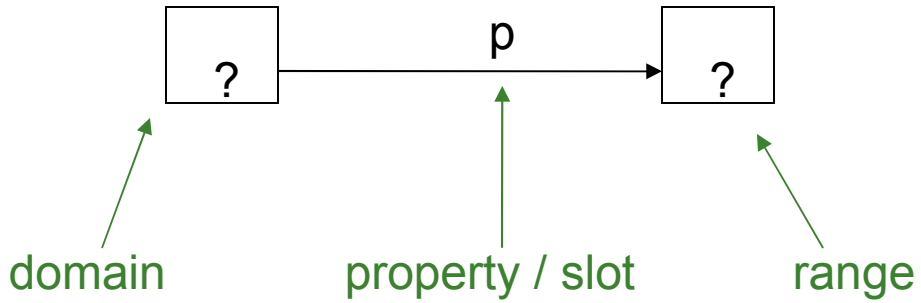
r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Properties or Slots

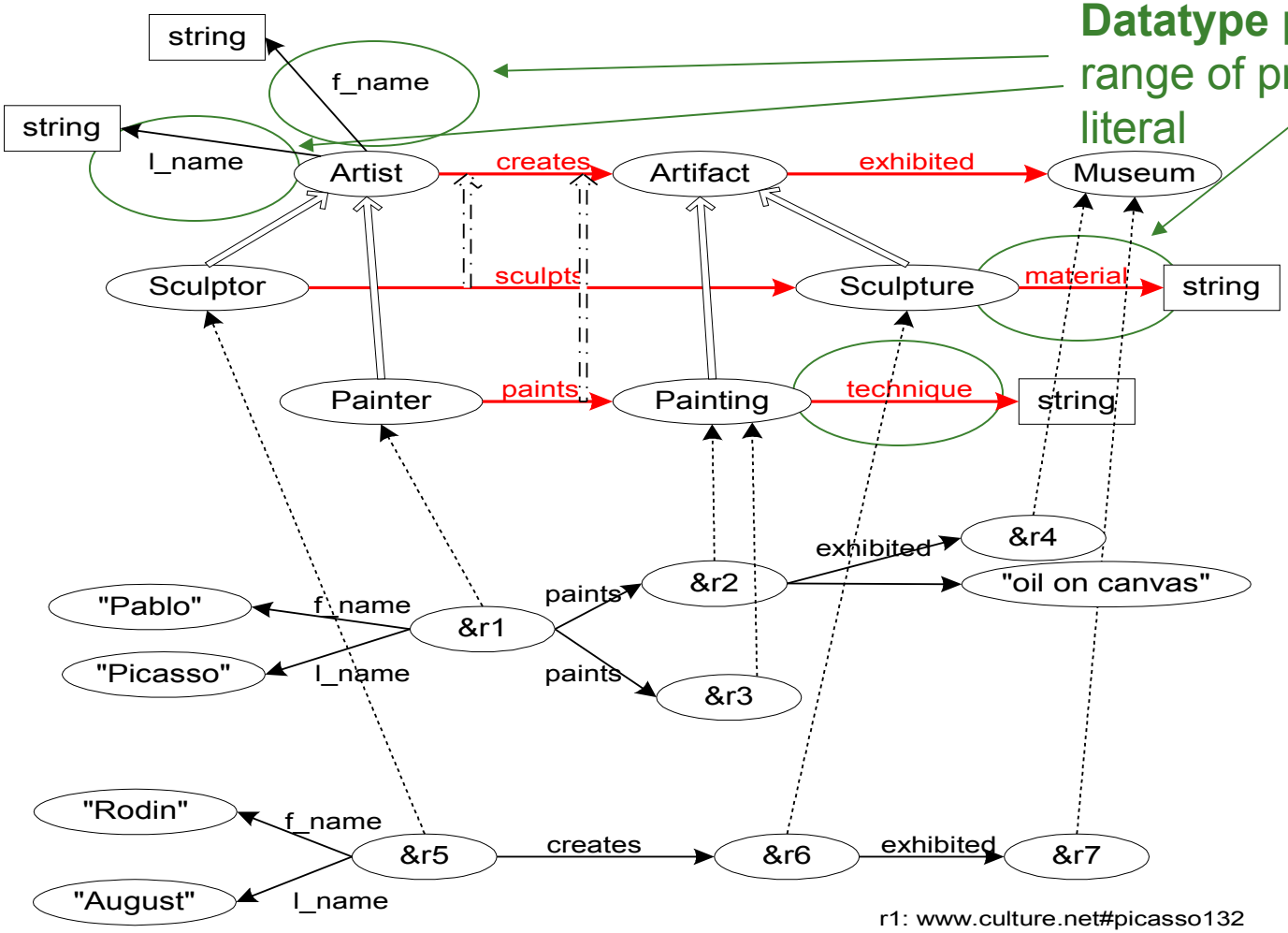


r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Property domain and range



Properties

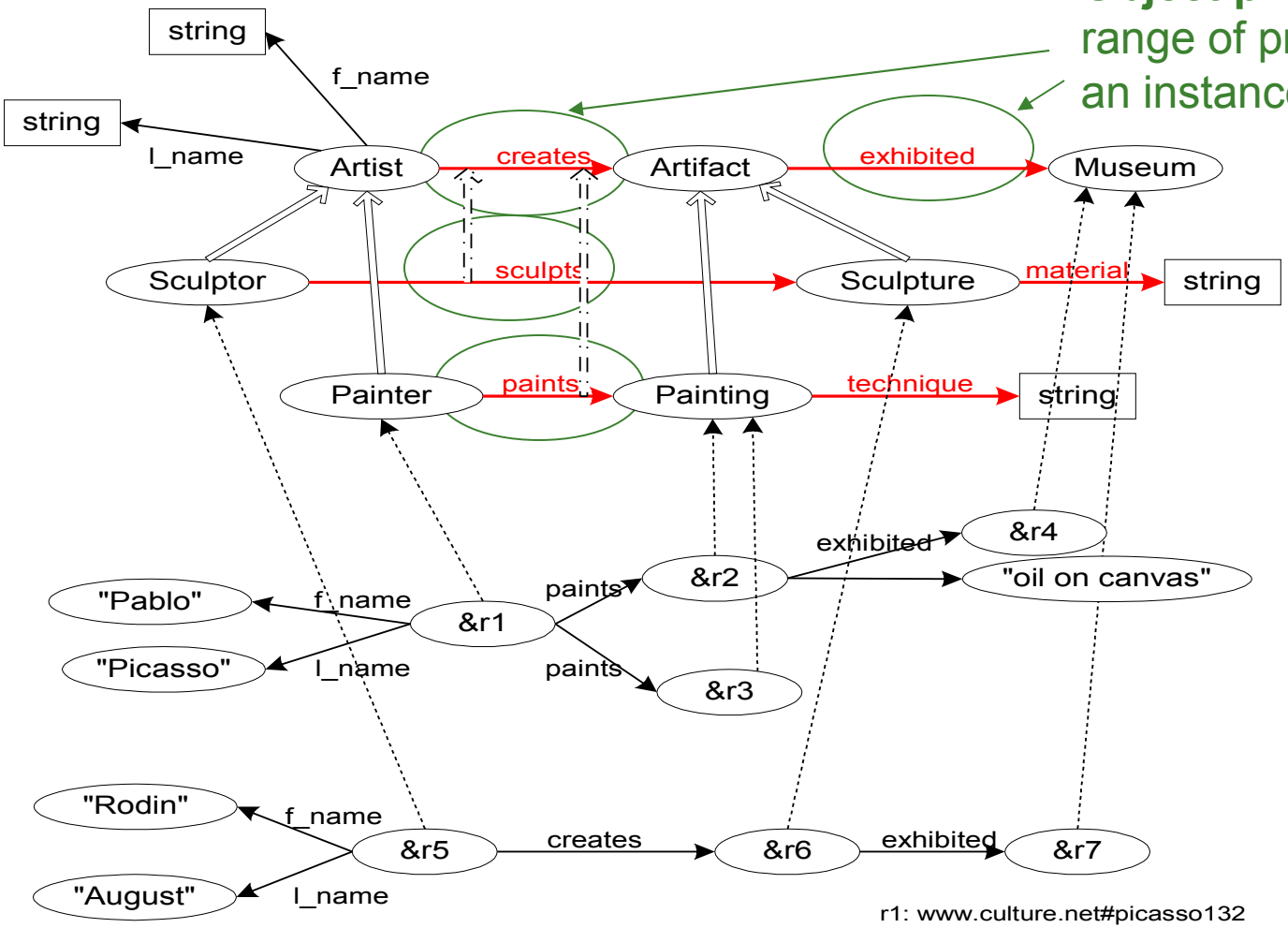


Datatype properties: range of properties is a literal

r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

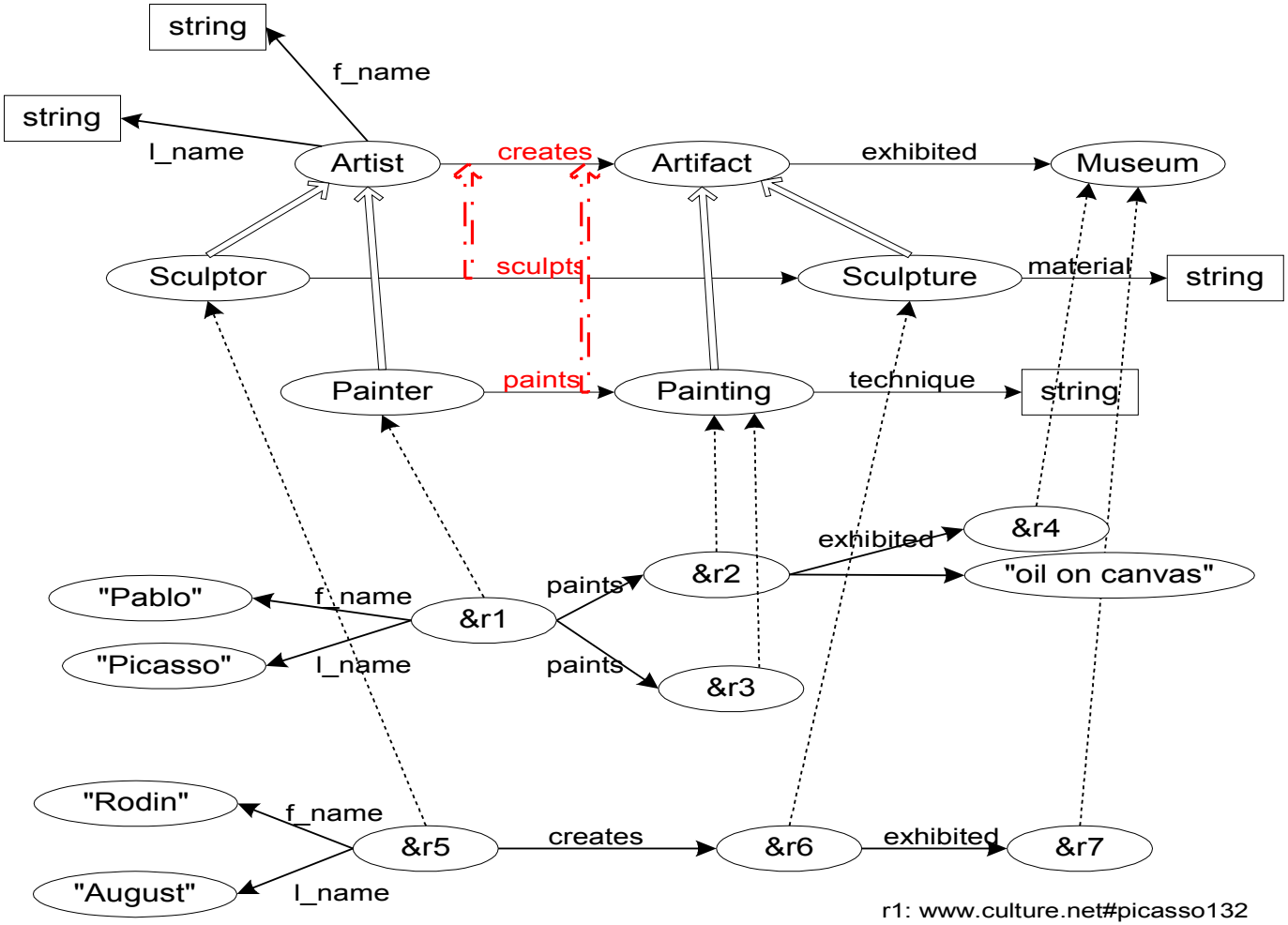
Properties

Object properties: range of properties is an instance



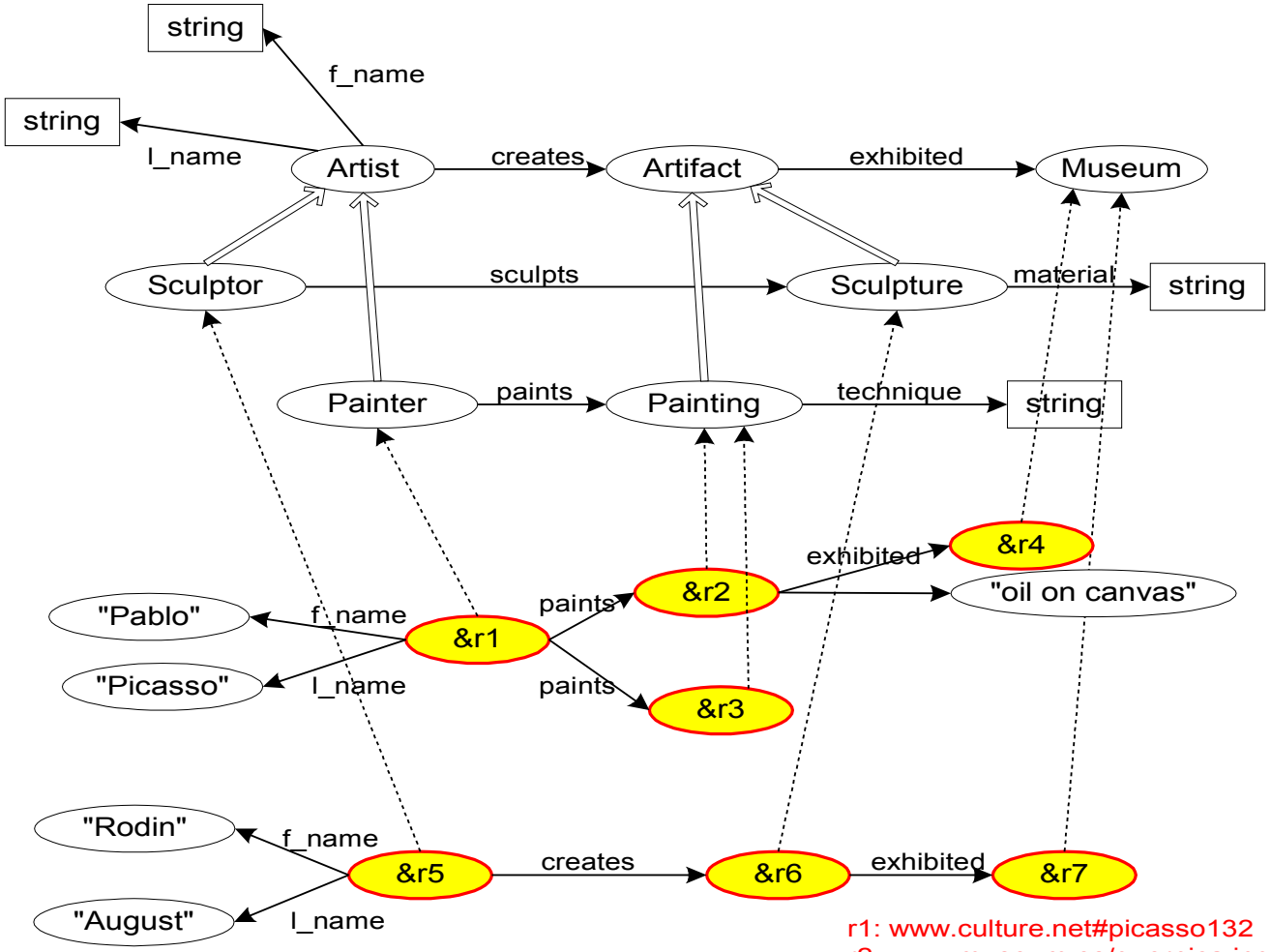
r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

SubProperty Relation



r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

Instances



r1: www.culture.net#picasso132
r2: www.museum.es/guernica.jpg
r3: www.museum.es/woman.qti
r4: www.museum.es
r5: www.culture.net#rodin424
r6: www.artchve.com/crucifixion.jpg
r7: www. rodin.fr

RDF(S) + SPARQL

- How can we handle (store and query) RDF(S) data?
- There are various RDF(S) stores similar to the database management systems
  - Jena2, Sesame, 3store, Oracle, etc.

Sesame

- An open source Java framework for storing, querying and reasoning with RDF and RDF Schema (http://www.openrdf.org/)
- Different storage supports:
  - Main memory
  - Native store
  - Database
  - Server
- The central concept is the ***repository***
  - Add RDF data to a repository
  - Query a particular repository
- Sesame supports RDFS inference in a ***forward chaining*** approach:
  - It adds all implicit information to the repository when data is being added

Prerequisites

- RDF(S)
- SPARQL
- Java!

Sesame – creating a repository

```
//Create a new main memory repository
MemoryStore store = new MemoryStore();
Repository myrepository = new SailRepository(store);
myrepository.initialize();

//store RDF from a file
File file = new File(inputDataFileName);
String fileBaseURI = "http://example.org/namespace#"; //namespace
RDFFormat fileRDFFormat = RDFFormat.RDFXML;
//open connection
RepositoryConnection con = myrepository.getConnection();
//add file to the repository
con.add(file, fileBaseURI, fileRDFFormat);

//store RDF from a URL
URL url = new URL("http://cgi.di.uoa.gr/~zoi/rdf/example.rdf");
String urlBaseURI = " http://example.org/namespace#";
RDFFormat urlRDFFormat = RDFFormat.RDFXML;
//open connection
RepositoryConnection con = myrepository.getConnection();
//add file to the repository
con.add(url, urlBaseURI, urlRDFFormat);
```

Sesame – querying a repository

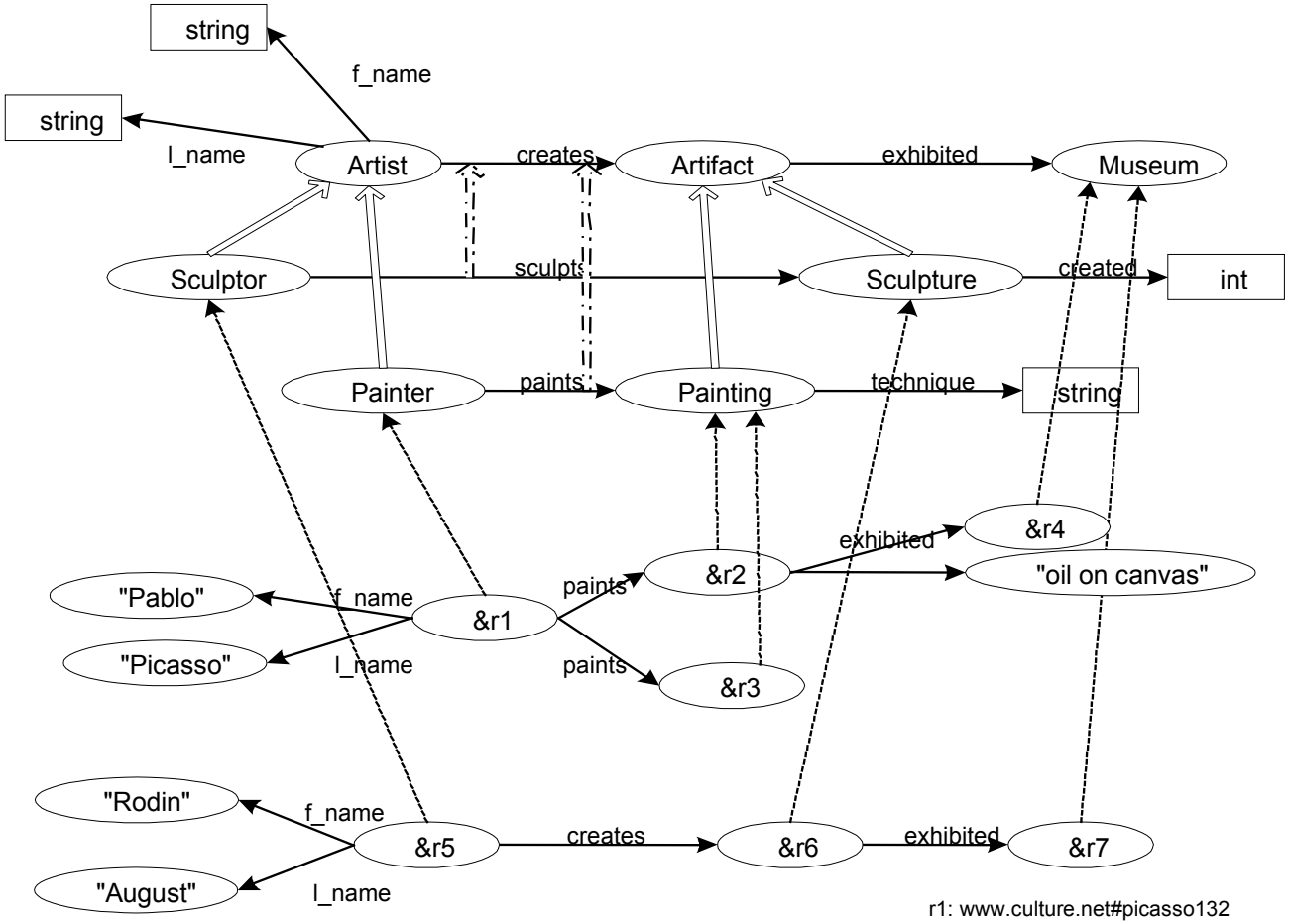
```
// open connection
RepositoryConnection con = myrepository.getConnection();
// create query
TupleQuery tupleQuery = con.prepareTupleQuery(QueryLanguage.SPARQL, queryString);
TupleQueryResult result = tupleQuery.evaluate();

// 1st way to iterate the results
while (result.hasNext()) {
    BindingSet bindingSet = result.next();
    Value valueOfX = bindingSet.getValue("x");
    Value valueOfY = bindingSet.getValue("y");
    System.out.println("?x="  + valueOfX + " ?y=" + valueOfY);
}

// 2nd way to iterate the results
List<String> bindingNames = result.getBindingNames();
while (result.hasNext()) {
    BindingSet bindingSet = result.next();
    Value firstValue = bindingSet.getValue(bindingNames.get(0));
    Value secondValue = bindingSet.getValue(bindingNames.get(1));
    System.out.println("?x=" + firstValue + ", ?y=" + secondValue);
}
```

Sesame – querying with inference

```
// Create a new main memory repository
MemoryStore store = new MemoryStore();
// create an inferencer
ForwardChainingRDFSInferencer inferencer = new ForwardChainingRDFSInferencer(store);
// include the inferencer in the repository
Repository myrepository = new SailRepository(inferencer);
myrepository.initialize();

File file = new File(inputFileName);
String fileBaseURI = "http://example.org/example#"; //namespace
RDFFormat fileRDFFormat = RDFFormat.RDFXML;

// open connection
RepositoryConnection con = myrepository.getConnection();
con.add(file, fileBaseURI, fileRDFFormat);

// add file to the repository
con.add(file, fileBaseURI, fileRDFFormat);
```

Sesame demonstration

An RDF(S) example



Sesame + named graphs

- A SPARQL query is executed against an RDF Dataset which represents a collection of graphs.
- Sesame uses the notion for ***context*** to group a set of RDF triples.
- This is the same with the notion of named graphs as you know from SPARQL.

Sesame + named graphs

```
//Create a new main memory repository
MemoryStore store = new MemoryStore();
Repository myrepository = new SailRepository(store);
myrepository.initialize();

//store RDF from a file
File file = new File(inputDataFileName);
String fileBaseURI = "http://example.org/namespace#"; //namespace
RDFFormat fileRDFFormat = RDFFormat.RDFXML;

// adding context
ValueFactory f = myrepository.getValueFactory();
URI context = f.createURI(“http://example.org/namedgraph1#”);

//open connection
RepositoryConnection con = myrepository.getConnection();
//add file to the repository
con.add(file, fileBaseURI, fileRDFFormat, context);
```

Sesame + querying namedgraphs

- Queries are executed as before using the SPARQL syntax for named graphs.

Sesame + construct queries

```
//Create a new main memory repository
….
// load file
….

GraphQuery gQuery = con.prepareGraphQuery(QueryLanguage.SPARQL, queryString);

GraphQueryResult graphResult = gQuery.evaluate();
while (graphResult.hasNext()) {
   Statement st = graphResult.next();
   System.out.println(st.toString());
}
```

OR

```
RDFXMLWriter rdfxmlWriter = new RDFXMLWriter(System.out);
gQuery.evaluate(rdfxmlWriter);
```

Useful links

- Sesame http://www.openrdf.org/
- Sesame download 2.2.4 http://sourceforge.net/projects/sesame/files/Sesame%20
- Sesame user guide http://www.openrdf.org/doc/sesame2/users/
- Sesame Chapter 8. The Repository API http://www.openrdf.org/doc/sesame2/users/ch08.html
- Sesame: “Using context” http://www.openrdf.org/doc/sesame2/users/ch08.html#d

Sesame demonstration