**RDF (Resource Description Framework)**: common framework to express information about resources (documents, people, physical objects, abstract concepts,...)

Statements about resources with the format: *subject predicate object*

## RDF triples

**RDF triple**: statement in RDF framework that uses the triple format to describe resources

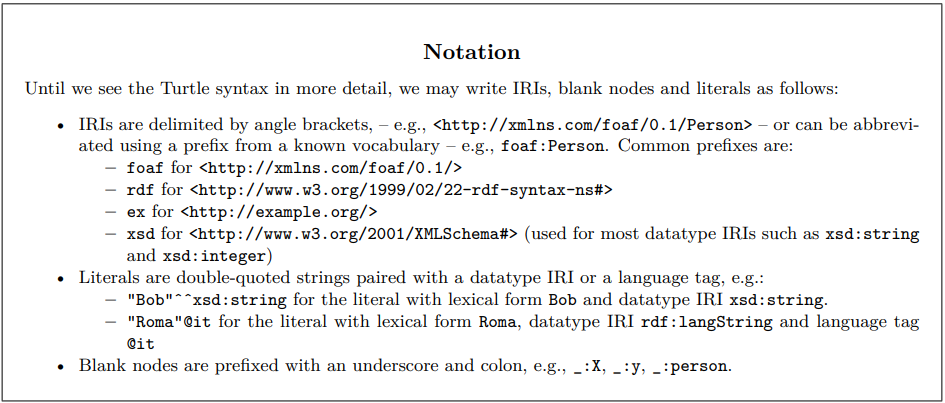
**RDF graph**: set of RDF triples

**IRI (International Resource Identifiers)**: used to identify resources, they are global identifiers. People can reuse IRIs to identify the same things. Appear in all three positions of a RDF triple and they can be abbreviated using known prefixes

**Blank nodes**: allows to talk about resources without using a global identifier. Are like simple variables, represent some things without saying what their value is. Can appear in the subject and object position of a triple. Blank nodes always have local scope. Even if the same blank node (\_:author) appears in two different RDF graphs this does not mean that it denotes the same resource in both graphs. Blank nodes should be renamed when working with RDF triples from multiple sources to avoid unwanted clashes

**Literals**: basic values that are not IRIs. Are strings “Leonardo da Vinci”, dates “the 4th of July”, numbers “3.14”. Can only appear in the object position

|  | **SUBJECT** | **PREDICATE** | **OBJECT** |
| --- | --- | --- | --- |
| IRI | yes | yes | yes |
| Blank nodes | yes | no | yes |
| Literals | no | no | yes |



## Union and merge of RDF graphs

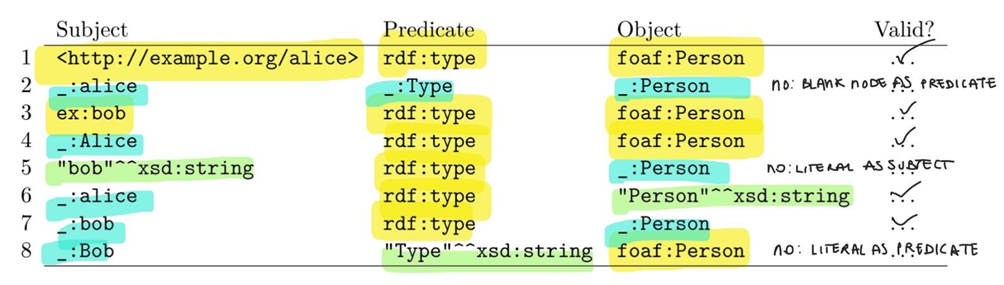
* **Union**: make the union of the two graphs, if we have blank nodes of different graphs with the same name, then they represent the same individual. Treat the blank nodes like constants.
* **Merge**: before doing the union of the two graphs, rename the blank nodes, in such a way that the two graphs do not share the same name for their blank nodes.

### Ex 1.1

IRI

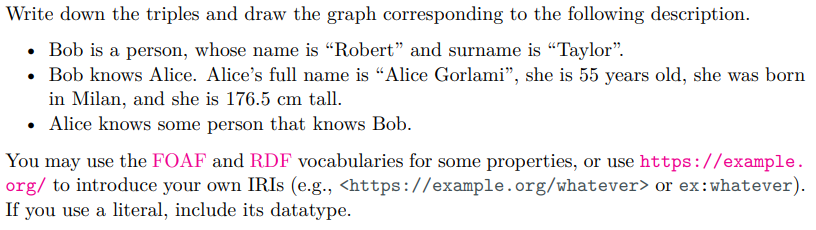
literal

blank node



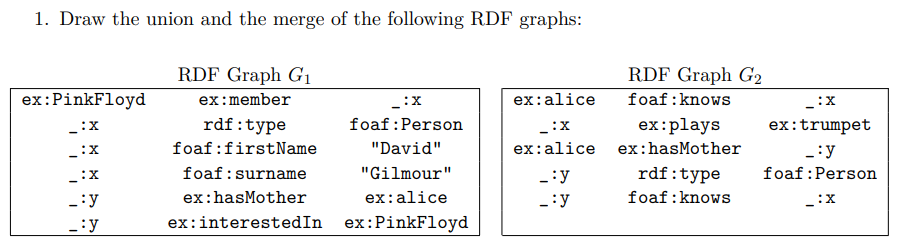
1. Yes
2. No: a blank node can be only in the subject or on the object, not in the predicate like \_:Type
3. Yes
4. Yes
5. No: a literal can only be in the object, "bob"^^xsd:string is in the subject
6. Yes
7. Yes
8. No: a literal like "Type"^^xsd:string can not be use as predicate

### Ex 1.2

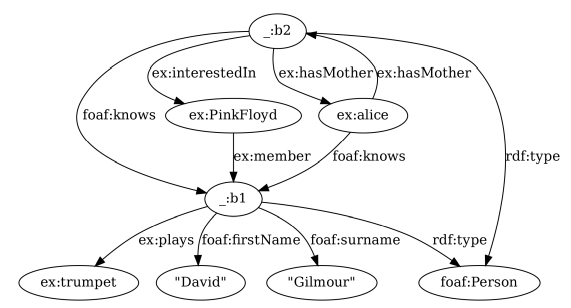


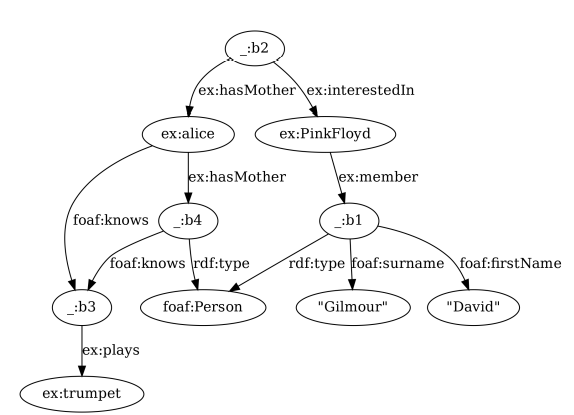
| **DESCRIPTION** | **TRIPLES** |
| --- | --- |
| Bob is a person | ex:bob rdf:type foaf:Person . |
| whose name is “Robert” | ex:bob foaf:firstName “Robert” . |
| and surname is “Taylor” | ex:bob foaf:surname “Taylor” . |
| Bob knows Alice | ex:bob foaf:knows ex:alice . |
| Alice’s full name is “Alice Gorlami” | ex:alice foaf:name ex:”Alice Gorlami” . |
| she is 55 years old | ex:alice foaf:age 55 . |
| she was born in Milan | ex:alice foaf:bornIn “Milan”@en . |
| and she is 176.5 cm tall | ex:alice hasHeight 176.5 |
| Alice knows some person | ex:alice foaf:knows \_:someone . |
| that knows Bob | \_:someone foaf:knows ex:bob . |

### Ex.2



Union (same blank node represent same individuals):

\_: x = b1 and \_: y= b2

Merge (same blank node in two graphs represent different individuals, need to rename it first): 

\_: x’ = b1 and \_: y’= b2

\_: x’’ = b3 and \_: y”= b4

## 

## 

## 

## Simple interpretations

U: set of IRIs

L: set of literals

B: set of blank nodes (assumed infinite)

**Vocabulary** (V) is a subset of U ∪ L

**Simple interpretation** *I* of a vocabulary V is a tuple *I = (IR, IP, IEXT, IS, IL*) where:

* IR: set of resources (called domainor universe of *I*)
* IP: set of properties (IP and IR may overlap, or IP subset of IR)
* IEXT: the actual extension of a property is given by a function IEXT: IP → P (IR x IR). It maps property to a binary relation on IR

The denotations of IRIs and literals in I are given by:

* IS: is a function IS: U → (IR ∪ IP). It maps an IRI to a resource or a property
* IL: partial function from literals to IR

For blank nodes you need an assignment α from the set B of blank nodes to the set IR (like variable assignments in other languages).

An interpretation *I* satisfies an RDF triple (s,p,o) without blank nodes if

* p^i € IP ⟹ I interprets p as a property
* (s^i, o^i) € IEXT(p^i) ⟹ I interprets s and o as members of the extension of p^i

Example:

Consider the triple: ex:a ex:b \_:c

where ex:a and ex:b are IRIs and \_:c is a blank node. If we want to satisfy this triple, we can build an interpretation like the following:

– The set of resources is IR = {a, b, c}

– The set of properties is IP = {b}.

– The extension of b is IEXT(b) = {(a, c)}.

– IS is such that IS(ex:a) = a and IS(ex:b) = b.

– We don’t care about IL.

– We can take an assignment α such that α(\_:c) = c.