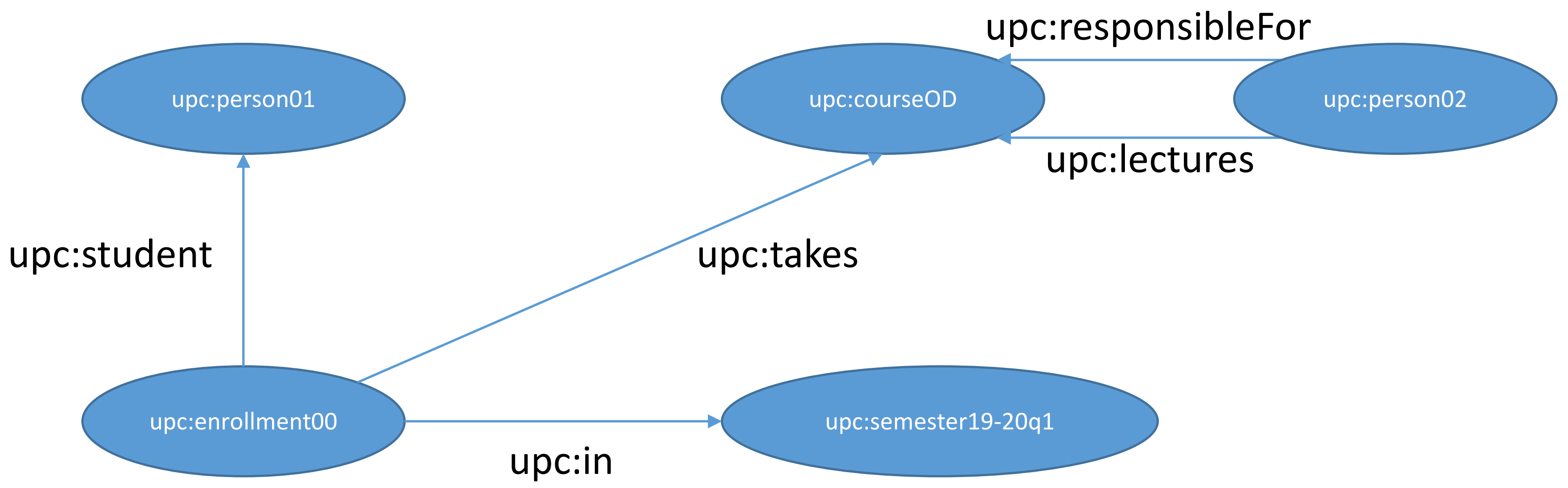
Namespace:
upc =“http://www.upc.edu/uris/”



Note: each of these URIs could participate in other triples to add relevant information. For example:

```
upc:person02 upc:name “Oscar Romero”
upc:courseOD upc:acronym “OD”
upc:courseOD upc:name “Open Data”, etc.
```

These triples would be equivalent to the concept of attributes in property graphs. Note though that edges cannot have attributes and we rely on the concept of reification to add attributes to relationships. Note that the concept of label does not exist in RDF either but you can simulate it with a triple.

# Observations

- In RDF, similar to property graphs, edges are binary (i.e., hyperedges are not allowed). This means an edge can only relate two concepts. Thus, n-ary relationships (like enrollment in this example) need of reification: i.e., the creation of an artificial concept that bridges the n concepts of the relationship.
  - Reification is needed whenever you need n-ary relationships.
- Everything is a URI (in this case, the URL is that of UPC, and the URN are the constants after the : in the URI inside the bubble).
  - A URI represents a concept. It cannot have neither attributes nor labels.
- As it is defined, RDF is very limited:
  - It cannot distinguish instances from schema (in the previous example, the URIs look like schema, but we cannot be sure without further information).
  - It cannot enforce constraints (e.g., cardinalities).
  - All URIs are created by the user. Therefore, we can understand what is it by dereferencing it through the HTTP protocol.
  - We may have used already existing vocabularies. If we use URIs from well-known vocabularies, it facilitates understanding the meaning of our graph and also reusing and sharing it.

# RDFS

- RDFS is a language that allows to distinguish instances from schema and add constraints into RDF.
- The difference and benefit is that the rdfs URIs are predefined and everybody understand their meaning. For example:

```
      ex:uri1 rdf:type ex:uri2,
```

which states that uri1 is an instance of the class uri2. See the RDFS slides to understand what are the constraints that can be expressed with the RDFS vocabulary.

I propose you redesign the exercise in these slides using RDFS and expressing as much constraints as possible.