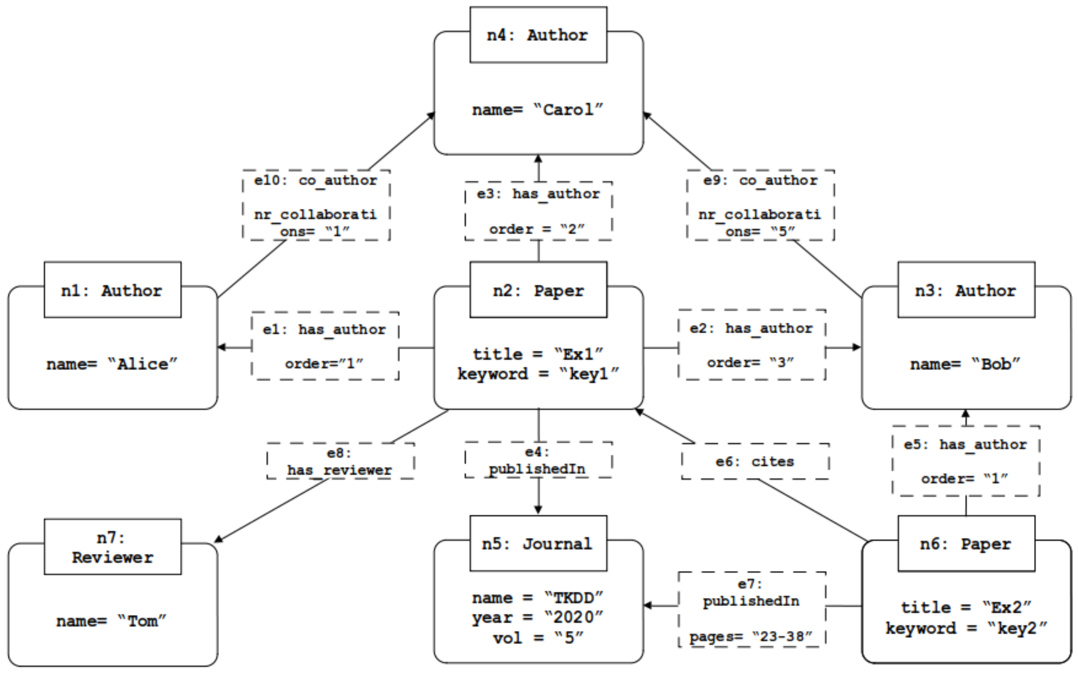
OPEN DATA EXAM

21st of June 2021. *The exam will take **2 hours**. Answer each question in the provided space.* <u>Answers out of such space will not be considered</u>. Further, clearly read the instructions how to answer. Answers not following the format set might not be considered.

Name: …….

QUESTION 1. PROPERTY GRAPHS [2p]

Given the property graph in Figure 1, answer the following questions.



a. Consider the property graph above to write a Cypher query whose result would be different under homomorphism semantics.

Justify your answer:

QUESTION 2. KNOWLEDGE GRAPHS AND DATA INTEGRATION [8p]
Note: read carefully the whole statement before solving this exercise

A typical data integration use case in Data Science is that of enriching internal data with external sources (e.g., Open Data). In this case, assume an online retail company that stores the following information about the purchases done in their website e-ACME.

"A customer buys a product at a given time from a certain location for a given price and a specific discount. Realise that every purchase, even if related to the same product, may have different prices and discounts (e.g., due to sales, offers, etc.). We enrich our own information with the following external data:

- *Using likedgeodata.org, we relate each location to the country it belongs to.*
- *By scrapping our competitors' websites, we relate each product with the price offered by each competitor."*

a. **[3p]** Use RDFS to represent the target schema of the integration system. **Draw your solution in the last page of the exam, in the target schema box**.

Important: When creating the solution to 2.a, you must also consider the necessary constraints so that your result can be reused for 2.c. Thus, carefully follow these instructions. To be reusable for 2.c, every concept must have an identifier related to it. Finally, distinguish concepts from features/attributes by representing concepts with regular circles and features/attributes with dashed circles. There is no need to specify domain / range constraints, just follow the notation used in the lectures for these exercises.

Hint: Draw your target schema in a tree-like shape (with concepts at top and attributes in the leafs). It will facilitate drawing the solution for 2.c.

b. **[2p]** Description Logics and OWL to add the following complex constraint. Write right below, first, the DL axiom and then their translation to OWL:

- *Define a new concept that represents all purchases that applied a discount (i.e., it contains all instances of purchase such that they have a discount >0).*

DL axiom:

OWL statements:

```
_:a rdfs:subClassOf :Purchase
_:a rdfs:subClassOf owl:Restriction
_:a owl:onProperty :discount
_:a owl:minCardinality 1
:DiscountedPurchase rdfs:subClassOf _:a
```

c. **[3p]** We aim at developing an integrated system to analyse data related to the e-ACME registered purchases. Since reacting fast is key, we aim at developing a **graph-based virtual data integration system** spanning several relevant data sources. Specifically, we choose to implement the **ontology-mediated querying system** seen in the lectures.
For this exercise, all instances are stored in other formats than graphs. Specifically, we have three sources:

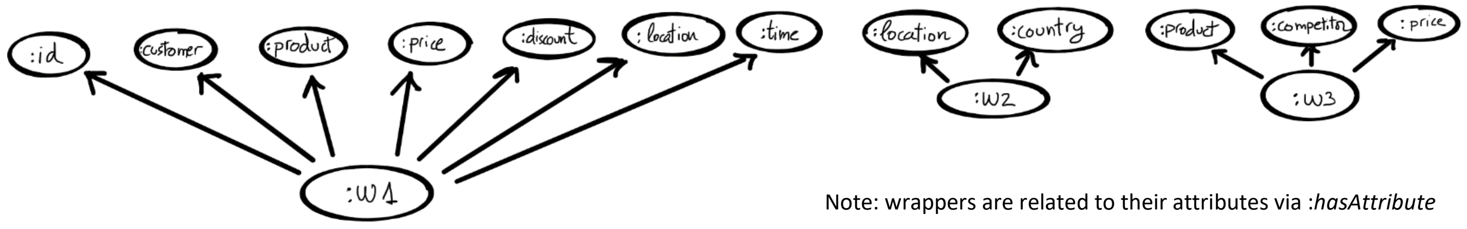
- A PostgreSQL database where e-ACME stores the information of its purchases:
  purchase (id, customer, product, price, discount, location, time)

- an API accessing *likedgeodata.org* data:
  place (location, country)
- and a MongoDB database storing the scrapped data from competitors:
  scrappedPrices (product, competitor, price)

In the next page, you can see the local graphs corresponding to the required wrappers (W1, W2 and W3, respectively, for each of the sources above), already created. In this exercise, you must create the mappings between the global and local levels required to make the data integration system work.

You must draw the mappings on top of your 2.a solution and the local level provided following the notation used in the lectures.

Global level: target schema (solution to 2.a)



Note: wrappers are related to their attributes via :*hasAttribute*

Source level: source graphs for each wrapper