Introduction to Semantic Systems
- *ANSWERED* Possible Exam Questions-

This collection of question aims to support students during the preparation for the exam. Students are expected to provide answers based on the material discussed during the lectures and presented in the lecture slides. Additional information collected from third-party materials is encouraged but not mandatory. Please note that questions listed here might appear in a slightly modified form in the exam. Good luck with the preparation!

1. **What is the "Semantic Web"? Refer to the definition given by Tim Berners-Lee in the Scientific American journal.**

"an extension of the current web (1) in which information is given well-defined meaning (2), better enabling computers and people to work in cooperation (3)."

2. **Provide two of the definitions of a knowledge graph discussed during the lecture.**

"Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities."

"A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge."

3. **Give examples of two companies that use knowledge graphs and explain how they make use of these structures as part of their use cases.**

Google: Entity search results
BBC Music: search, recommendation etc. of music

4. **What are the key enabling technologies for the semantic web?**

- one or more standard vocabularies (ontologies) capturing the semantics
- a standard syntax
- lots of resources with meta-data attached

5. **Define an ontology (use the definition of Studer from 1998).**

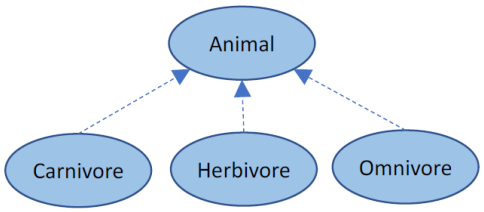
"Formal, explicit specification of a shared conceptualization."

6. **Which are the three main categories of ontologies based on their expressivity?**

- Lightweight ontologies (expressivity low)
- Taxonomies (expressivity medium)
- heavyweight ontologies (expressivity high)

7. **What is a taxonomy? Provide a definition and an example.**

Taxonomy: a controlled vocabulary organized into a hierarchical structure. Example:

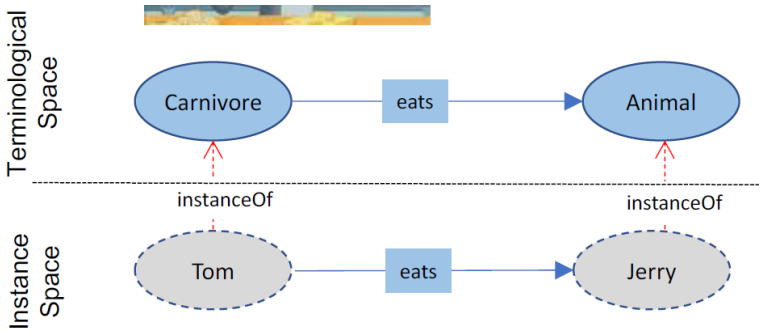


8. **What are the main elements of an ontology? Give a few examples for a domain of your choice.**

Ontology: A taxonomy extended with other relations and further constraints.

- Concepts: Denote the main concepts of the domain; E.g.: Carnivore, Animal
- Concept hierarchy: Denotes specialization/generalizations; E.g.: Carnivore is a Animal
- Relations between classes; E.g.: Carnivore eats Animal
- Restrictions on relations (type, cardinality); E.g.: any Animal has at least one BodyPart

-Instances: Denote concrete entities in the domain; E.g.: Tom, Jerry



9. **What are the main stages of the ontology engineering methodology proposed by Noy & McGuinness? Enumerate the stages and explain each of them briefly.**

- Determine Scope
- Consider Reuse
- Enumerate Terms
- Define Classes and Taxonomy
- Define Properties
- Define Constraints
- Create Instances
- Check for Anomalies

10. **What is the "AAA Slogan"? Which two important assumptions for the Semantic Web Stem from this slogan? Explain each briefly.**

- AAA Slogan: "On the Web anyone can say Anything about any topic"
- Nonunique Naming Assumption: The same entity could be known by more than one name;
  E.g., PersonA can be the same instance as PersonB
- Open World Assumption: "Missing information is not evaluated as negative information!"
  E.g., likes(PersonA, DrinkB) -> PersonA may also like other drinks…

11. **Define and give an example for a universal restriction.**

"If a property is declared for an instance, all its values must be of a certain type"
E.g.: Wines can only be made in Wineries

12. **Define and give an example for an existential restriction.**

"There must be at least one property with that value"
E.g.: Wines are located in regions so there has to be at least one region property given

13. **Name and give examples of at least three property characteristics that can be specified in ontologies.**

- Symmetry: If P(x,y) =>P(y,x)
- Transitivity: If P(x,y) and P(y,z) => P(x, z)
- Inverse properties: If P(x,y) => InvP(y,x) holds
  E.g.: hasParent(John, Bob)=> hasChild(Bob, John)

14. **What are three types of reasoning tasks that can be performed on ontologies?**

- Consistency Checking: Are there logical errors in the ontology?
- Satisfiability Checking: Are there classes that cannot possibly have any instances?
- Class Inference: Discover new knowledge based on the given knowledge.

15. **Define the notion of *Blank Nodes* in RDF and give one modelling example where they are useful.**

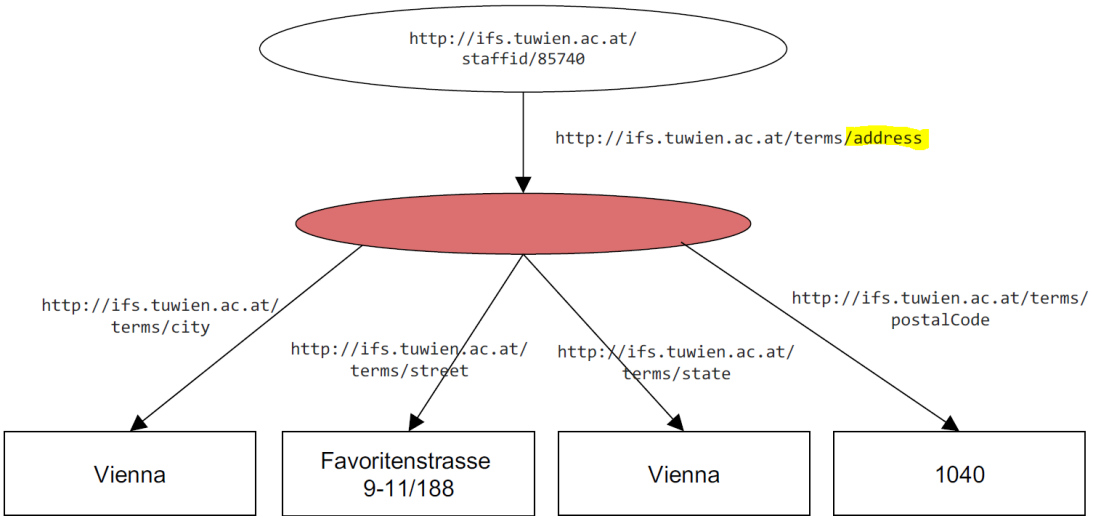
Blank Nodes represent unnamed resources, used widely, with varying intentions:

– Make statements about resources that do not haveURIs (but that are described in terms of relationships with other resources that do)

– group related information

– represent n-ary relationships

Example for useful blank node:



16. **What does the notion of *Reification* mean? How can reification be realized in RDF? Give an example.**

Reification is used for making statements about statements to

– model data origin

– formalize statements about reliability and trust

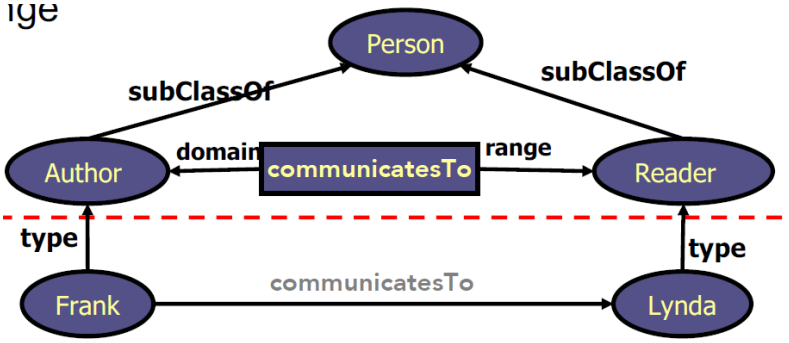
– define metadata about statements

17. **What are the three main knowledge representation languages used in the Semantic Web? Discuss how they differ in terms of their knowledge representation capabilities.**

- RDF:

- Triples consting of Subject, Predicate, Object
- Graph-based model
- Formal semantics → machine-readable
- Knowledge expressed as a list of statements

- RDFS:
  - Defines vocabulary for RDF
  - Organizes this vocabulary in a typed hierarchy
    - Class, subClassOf, type
    - Property, subPropertyOf
    - domain, range

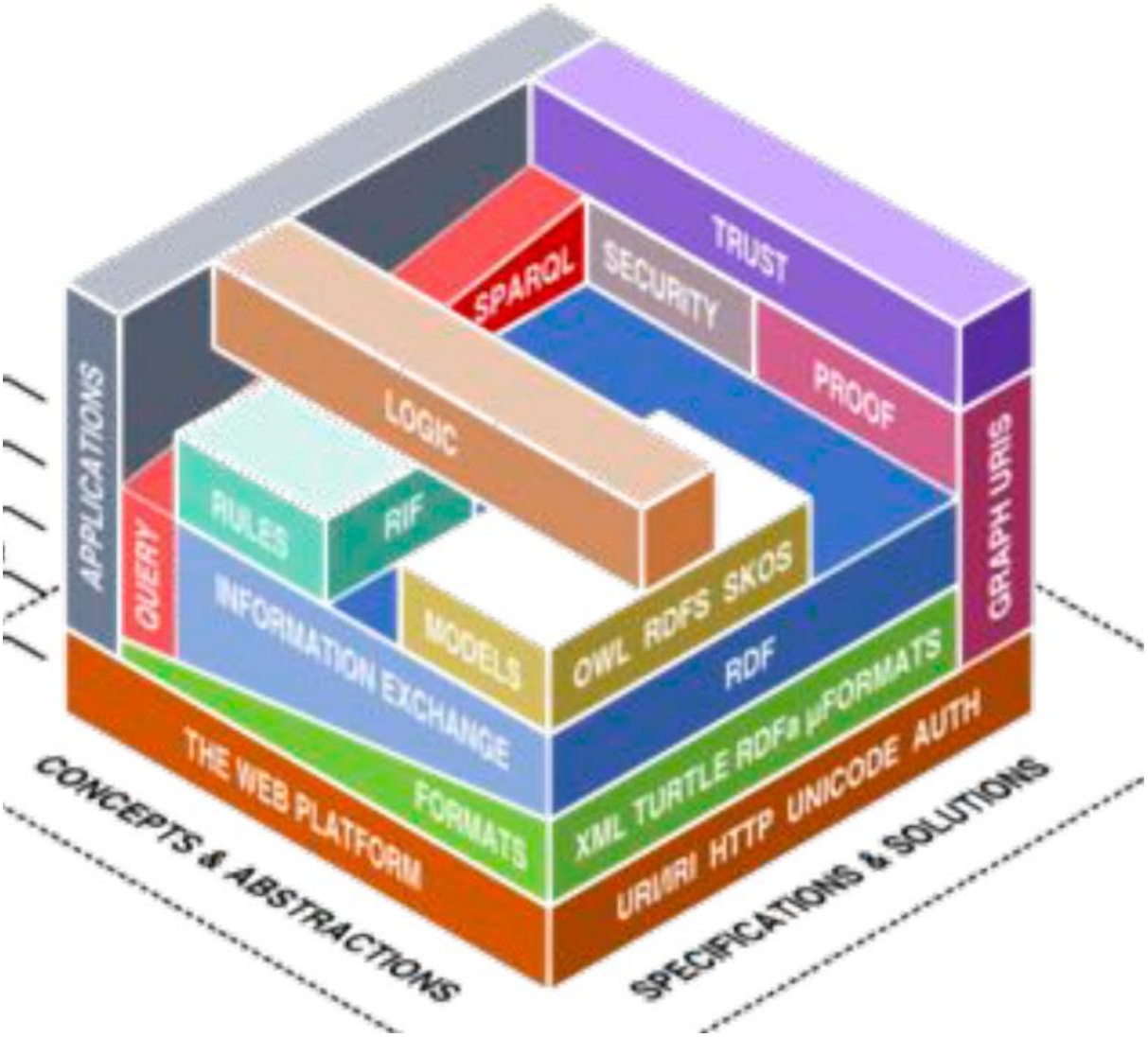


RDF(S) does not allow for
- Localized range and domain constraints
- Existence and cardinality constraints
- Transitive, inverse or symmetrical properties

- OWL:

Provides a wider range of ontological constructs and avoids some
of the potential confusion in RDF-S -> OWL extends RDF Schema to a full-fledged
knowledge representation language for the Web.
3 basic building blocks in OWL: Classes, Individuals, Properties

18. **Describe the Semantic Web Technology Stack by (1) identifying the main concepts and abstractions covered by the stack as well as (2) naming the concrete specifications that correspond to each of these abstractions. (It is sufficient to focus on those abstractions and solutions that were covered in the lecture).**



Abstraction -> Specification:
Web Platform -> URIs, HTTP
Formats -> XML, Turtle, RDF
Information Exchange -> RDF
Model -> OWL RDFS
Query -> Sparql

19. **Describe the typical stages followed with the (Google) OpenRefine tool for creating RDF based data from tabular data. Name and briefly describe each stage.**

1. Import messy input data, transform it into a table, and clean it.
2. Apply entity reconciliation to interlink with existing data sets.
3. Define the structure of the RDF output. (RDF skeleton + mapping from data to RDF)
4. Export the data into some RDF syntax.

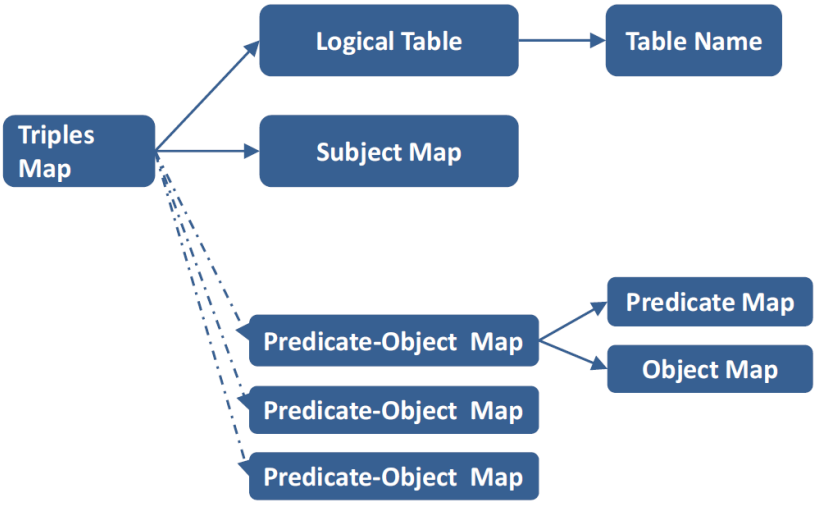
20. **Describe the process of transforming a database into an RDF dataset by using R2RML. Additionally, describe the mandatory and optional components of an R2RML mapping definition.**

Main idea: transform data from a database into RDF data.

R2RML, the RDB to RDF mapping language:

-Allows vocabulary mapping (subject, predicate and object maps with class options)

- Allows interlinking – URIs can be constructed



Note: Elements with dotted lines are optional

21. **Define the term "Semantic Annotation". Provide also the formal model of a semantic annotation.**

- The term "**annotation**" implies to **attach data to some other piece of data**. It establishes a **(typed) relation between the annotated data and the annotating data**
- The term "**annotation**" can denote both the **process of annotating** and the **result of that process**
- **Formal Model:**
  - An **annotation** A is a tuple **(a_s, a_p, a_o, a_c)**, where
    - **a_s** is the **subject** of the annotation (the annotated data)
    - **a_o** is the **object** of the annotation (the annotating data)
    - **a_p** is the **predicate** (the annotation relation) that defines the type of relationship between a_s and a_o, and
    - **a_c** is the **context** in which the annotation is made.

22. **What is the typical structure of a SPARQL query? Name and briefly describe the role of each part of a SPARQL query.**

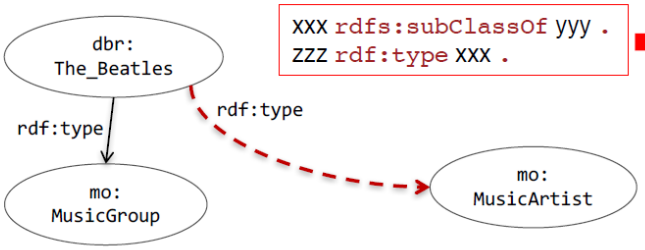
```
# Prefix declarations, for abbreviating URIs
PREFIX dbpedia: http://dbpedia.org/resource/
...

# Result clause, identifying what information to return from the query
SELECT/ASK/CONSTRUCT ...

# Dataset definitions, stating what RDF graph(s) are being queried
FROM  <http://musicbrainz.org/20130302>

# Query pattern, specifying what to query for in the underlying dataset
WHERE {
    ...
}

# Query modifiers, slicing, ordering, rearranging query results
ORDER BY ...
```

23. **What are the four query forms in SPARQL and what is their result type?**

- **SELECT**
  returns variables and their bindings directly
- **ASK**
  tests whether or not a query pattern has a solution.
  Returns yes/no
- **DESCRIBE**
  returns a single RDF graph containing RDF data about resource
- **CONSTRUCT**
  returns a single RDF graph specified by a graph template

24. **For two RDF entailment patterns of your choice (1) describe the pattern and (2) exemplify how querying is affected when reasoning in terms of that pattern in comparison to the case when no reasoning is enabled.**

Reasoning with – rdfs:subClassOf




```
XXX rdfs:subClassOf yyy .
ZZZ rdf:type XXX .
```

→

```
ZZZ rdf:type yyy .
```

Schema:

```
mo:MusicGroup rdfs:subClassOf mo:MusicArtist .
```

Query:

```
SELECT ?x
WHERE {?x a mo:MusicArtist.}
```

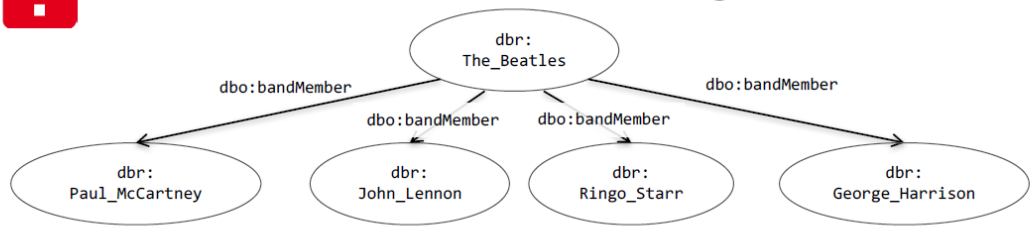
Result set with inference:

?x
dbr:The_Beatles …

Result set without inference:

?x

Reasoning with– rdfs:range



Schema:

```
dbo:bandMember rdfs:range foaf:Agent .
```

Query:

```
SELECT ?x
WHERE {?x a foaf:Agent.}
```

Result set with inference:

?x
dbr:Paul_McCartney
dbr:John_Lennon
dbr:Ringo_Starr
dbr:George_Harrison …

Result set without inference:

?x

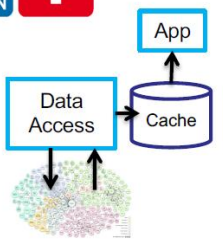
```
aaa rdfs:range XXX .
yyy aaa zzz .
```

→

```
zzz rdf:type XXX .
```

25. **Describe at least two architectural patterns for KG applications.**

Architectural Patterns

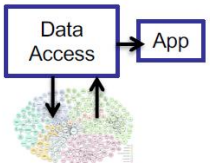


1. Crawling:

Crawl or load data in advance

\+ Data is managed in one triple store → efficient access

– Disadvantage: data might not be up to date



2. On-The-Fly Dereferencing:

URIs are dereferenced at the moment the app requires the data

\+ Retrieves up to date data

– Performance is affected when the app must dereference many URIs

26. **What is a re-publication component in the KG architecture? Why is it needed and what are the available options?**

Data Tier: Data Re-Publication

Exposes integrated dataset as Linked Data

Various options:

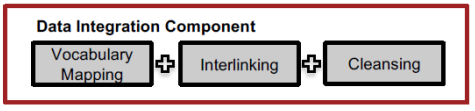
1. Via SPARQL endpoints
   (e.g., GraphDB, RDF4J Endpoint, OpenLink Virtuoso, Apache Jena Fuseki etc.)
2. As RDF dumps
3. As dereferenceable Linked Data
4. As Linked Data Fragments
   (cf. http://linkeddatafragments.org)
5. Via APIs
   (e.g., Linked Data API)
6. With built-in mechanisms of CMSs
   (e.g., Drupal, Information Workbench, …)

27. **How to perform data integration at schema and instance level in a KG application?**

Data Tier: Data Integration

Consolidates data retrieved from heterogeneous sources

- **Schema level**
  - Performs vocabulary mappings in order to translate data into a single unified schema
  - Links correspond to RDFS properties or OWL property and class axioms
- **Instance level**
  - Performs entity resolution via `owl:sameAs` links
  - Tools like Silk or OpenRefine can be used to integrate the data in case the data sources do not provide the links



28. **What are conceptual differences between RDF4J and Apache Jena? In what circumstances would you use each of them?**

Discussion

rdf4j vs jena

- **Conceptual differences:**
  - Actions against repositories (RDF4J) vs. handling models directly (Jena)
  - RDF4J completely abstracts the storage layer; Jena requires somewhat different models for SDB, TDB,..
- **Triple stores:**
  - Both Jena TDB and RDF4J's `NativeStore` perform reasonably well for moderate data set sizes
  - ... and not so well for very large graphs with billions of triples (can use other triple stores)
- **Query features:** Similar in terms of SPARQL features; RDF4J supports SeRQL

Discussion

rdf4j vs jena

- **Inference:**
  - RDF4J supports RDF-S reasoning
  - Jena additionally has built-in support for OWL, OWL Mini, OWL Micro
  - Jena has a flexible subsystem to plug in a range of reasoners
- **Misc:**
  - Both provide good Serialization/Deserialization for RDF I/O
  - RDF4J has some nice data management tools (console, workbench)
    - The latest version of Fuseki is trying to catch up
  - Jena's documentation is a little more comprehensive
  - Some differences in features and performance characteristics

29. **What are the differences/mismatch between Object-Oriented Programming and Ontology Modeling?**

Graph/Object mapping

Impedance mismatch

- OO: design decisions based on the operational properties of a class
- RDF-S/OWL: decisions based on the structural properties of a class

→ OWL is property-focused; Java is class-focused
→ a class structure and relations among classes in an ontology are different from the structure for a similar domain in an OO program
→ 1:1 class mapping between Objects and OWL is problematic (e.g. multiple inheritance)

"Java-first" Options:

- RDF4J Alibaba API ("Java-first" graph/object-mapping using Java Annotations)
- Jena itself doesn't have persistence annotations
  (there is JenaBean, but it's not actively maintained)

"RDFS first" options: code generators (RDFReactor, Kazui, Owl2Java..)
BUT: often not straightforward (e.g., `xsd:Literal` → ?)