Q.1

Parts (a) - (f) refer to the following data. In each of (a) - (f), write out the exact output of the

SPARQL query in the question.

@prefix ab: <http://learningsparql.com/ns/addressbook#> .

@prefix d: <http://learningsparql.com/ns/data#> .

d:i0432 ab:firstName "Richard" .

d:i0432 ab:lastName "Mutt" .

d:i0432 ab:homeTel "(229) 276-5135" .

d:i0432 ab:email "richard49@hotmail.com" .

d:i9771 ab:firstName "Cindy" .

d:i9771 ab:lastName "Marshall" .

d:i9771 ab:homeTel "(245) 646-5488" .

d:i9771 ab:email "cindym@gmail.com" .

d:i8301 ab:firstName "Craig" .

d:i8301 ab:lastName "Ellis" .

d:i8301 ab:email "craigellis@yahoo.com" .

d:i8301 ab:email "c.ellis@usairwaysgroup.com" .

(a)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?craigEmail

WHERE

{

?person ab:firstName "Craig" .

?person ab:email ?craigEmail .

}

|?craigEmail |

| |

|[craigellis@yahoo.com](mailto:craigellis@yahoo.com) |

|[c.ellis@usairwaysgroup.com](mailto:c.ellis@usairwaysgroup.com) |

(b)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?craigEmail

WHERE

{

?person ab:firstName "Craig" .

?person ab:lastName "Ellis" .

?person ab:email ?craigEmail .

}

|?craigEmail |

| |

|[craigellis@yahoo.com](mailto:craigellis@yahoo.com) |

|[c.ellis@usairwaysgroup.com](mailto:c.ellis@usairwaysgroup.com) |

Side 2 av 6(c)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?first ?last

WHERE

{

?person ab:homeTel "(229) 276-5135" .

?person ab:firstName ?first .

?person ab:lastName ?last .

} \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|?first | ?last |

|Richard| Mutt |

(d)

PREFIX a: <http://learningsparql.com/ns/addressbook#>

SELECT ?propertyName ?propertyValue

WHERE

{

?person a:firstName "Cindy" .

?person a:lastName "Marshall" .

?person ?propertyName ?propertyValue .

}

|?propertyName | ?propertValue |

a:firstName | Cindy

a:lastName | Marshall

a:homeTel | (245) 646-5488

| a:email | cindym@gmail.com|

(e)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT \*

WHERE

{

?s ?p ?o .

FILTER (regex(?o, "yahoo","i"))

}

|?s | ?p | ?o |

|d:i8301 | ab:email | craigellis@yahoo.com|

(f)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?craigEmail ?homeTel

WHERE

{

?person ab:firstName "Craig" .

?person ab:lastName "Ellis" .

?person ab:email ?craigEmail .

?person ab:homeTel ?homeTel . //Må oppfylle alt i queryen

}

| ?craigEmail | ?homeTel |

Side 3 av 6Q2.

Part (a) refers to the following data. Write out the exact output of the SPARQL query in the

question.

IMPORTANT: Assume that the SPARQL endpoint is able to perform OWL reasoning.

@prefix ab: <http://learningsparql.com/ns/addressbook#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

ab:i0432

ab:firstName "Richard" ;

ab:lastName "Mutt" ;

ab:spouse ab:i9771 .

ab:i8301

ab:firstName "Craig" ;

ab:lastName "Ellis" ;

ab:patient ab:i9771 .

ab:i9771

ab:firstName "Cindy" ;

ab:lastName "Marshall" .

ab:spouse

rdf:type owl:SymmetricProperty ;

rdfs:comment "Identifies someone's spouse" .

ab:patient

rdf:type rdf:Property ;

rdfs:comment "Identifies a doctor's patient" .

ab:doctor

rdf:type rdf:Property ;

rdfs:comment "Identifies a doctor treating the named resource" ;

owl:inverseOf ab:patient .

(a)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?doctorFirst ?doctorLast ?spouseFirst ?spouseLast

WHERE

{

?s ab:firstName "Cindy" ;

ab:lastName "Marshall" ;

ab:doctor ?doctor ;

ab:spouse ?spouse .

?doctor ab:firstName ?doctorFirst ;

ab:lastName ?doctorLast .

?spouse ab:firstName ?spouseFirst ;

ab:lastName ?spouseLast .

}

|  |  |  |  |
| --- | --- | --- | --- |
| ?doctorFirst | ?doctorLast | ?spouseFirst | ?spouseLast |
| Craig | Ellis | Richard | Mutt |
|  |  |  |  |
|  |  |  |  |

Side 4 av 6Parts (b) and (c) refer to the following data. In each of (b) and (c), write out the exact output of

the SPARQL query in the question.

@prefix ab: <http://learningsparql.com/ns/addressbook#> .

@prefix d: <http://learningsparql.com/ns/data#> .

d:i0432 ab:firstName "Richard" .

d:i0432 ab:lastName "Mutt" .

d:i0432 ab:homeTel "(229) 276-5135" .

d:i0432 ab:nick "Dick" .

d:i0432 ab:email "richard49@hotmail.com" .

d:i9771 ab:firstName "Cindy" .

d:i9771 ab:lastName "Marshall" .

d:i9771 ab:homeTel "(245) 646-5488" .

d:i9771 ab:email "cindym@gmail.com" .

d:i8301 ab:firstName "Craig" .

d:i8301 ab:lastName "Ellis" .

d:i8301 ab:workTel "(245) 315-5486" .

d:i8301 ab:email "craigellis@yahoo.com" .

d:i8301 ab:email "c.ellis@usairwaysgroup.com" .

(b)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?first ?last ?workTel

WHERE

{

?s ab:firstName ?first ;

ab:lastName ?last ;

ab:workTel ?workTel .

}

|  |  |  |
| --- | --- | --- |
| ?first | ?last | ?workTel |
| Craig | Ellis | (245) 315-5486 |

(c)

PREFIX ab: <http://learningsparql.com/ns/addressbook#>

SELECT ?first ?last ?workTel ?nick

WHERE

{

?s ab:firstName ?first ;

ab:lastName ?last .

OPTIONAL

{

?s ab:workTel ?workTel ;

ab:nick ?nick .

}

|  |  |  |  |
| --- | --- | --- | --- |
| ?first | ?last | ?workTel | ?nick |
| Richard | Mutt |  |  |
| Craig | Ellis |  |  |
| Cindy | Marshall |  |  |

}

Side 5 av 6Q3. For each of the HTML snippets in parts (a) and (b), write out the RDF triples embedded

into the snippets. You should write the RDF with the turtle notation.

(a)

<p vocab="http://schema.org/" prefix="ov: http://open.vocab.org/terms/" resource="#manu"

typeof="Person">

My name is

<span property="name">Manu Sporny</span>

and you can give me a ring via

<span property="telephone">1-800-555-0199</span>.

<img property="image" src="http://manu.sporny.org/images/manu.png" />

My favorite animal is the <span property="ov:preferredAnimal">Liger</span>.

</p>

**Turtle**

@prefix ns1:<<http://www.w3c.org/ns/rdfs#>>.

@prefix ns2:<<http://schema.org/>>.

@prefix ov:<<http://open.vocab.org/terms#>>

<>ns1:usesVocabulary ns2:

<#manu> a ns2:Person;

ns2:name “Manu Sporny”;

ns2:telephone “1-800-555-0199”;

ns2:image “<http://manu.sporny.org/images/manu.png>";

ov:preferredAnimal “Liger”.

**JSON-LD**

{

"@context" : {

"ns1": "http://www.schema.org/",

"ov": "http://open.vocab.org/terms/"

},

"@type": "Person",

"name":"Manu Sporny",

"telephone" : "1-800-555-0199",

"image" : "http://manu.sporny.org/images/manu.png",

"preferredAnimal":"Liger"

}

(b)

<div vocab="http://schema.org/" resource="#bbg" typeof="LocalBusiness">

<h1 property="name">Beachwalk Beachwear &amp; Giftware</h1>

<span property="description"> A superb collection of fine gifts and clothing

to accent your stay in Mexico Beach.</span>

<div property="address" resource="#bbg-address" typeof="PostalAddress">

<span property="streetAddress">3102 Highway 98</span>

<span property="addressLocality">Mexico Beach</span>,

<span property="addressRegion">FL</span>

</div>

Phone: <span property="telephone">850-648-4200</span>

</div>

@prefix ns1:<<http://ww.w3c.org/ns/rdfa#>>.

@prefix ns2:<<http://www.schema.org/>>

<>ns1:usesVocabulary ns2.

<#bbg> a ns2:LocalBusiness;

ns2:address “#bbg-address”;

ns2:name “Beachwalk Beachwear & Giftware”;

ns2:description “ A superb collection of fine gifts and clothing

to accent your stay in Mexico Beach.”;

ns2:telephone “850-648-4200”;

<#bbg> a ns:2PostalAddress;

ns2:streetAddress “3102 Highway 98”;

ns2:addressLocality “Mexico Beach”;

ns2:addressRegion “FL”;

**JSON-LD**

{

“@context”:”<http://schema.org.org/>”,

“@type”:”LocalBusiness”,

“address”:”#bbg-address”

“name”:”Beachwalk Beachwear & Giftware”,

“description”:” A superb collection of fine gifts and clothing

to accent your stay in Mexico Beach.”,

“telephone”:”850-648-4200”}

{

“@context”:”[http://schema.org/](http://schema.org.org/)”,

“@type”:”PostalAddress”,

“streetAddress”:”3102 Highway 98”,

“addressLocality”:”Mexico Beach”,

“adressRegion”:”FL”

}

**Q4. What do we mean by the word “semantics” in Information Systems? Why is “semantics”**

**useful (or even perhaps necessary) for the use of information in the world today?**

Semantics in Information Systems is the “meaning” of the data in question. “Meaning” in ths particular case is dependent on an outside interaction by, e.g. a query, from either directly from a person, or another computer. A semantically annoteated file (e.g. HTML file with RDF properties (RDFa)) by itself doesn’t explicitly have “more meaning” than a regular HTML, the difference lies in the context of how it’s utilised, various computer languages can access these embedded properties, which in return, returns the information associated with these properties. These properties are defined on external sites, where it’s listed pre-defien meanings of the said properties.

The need for semantic data is growing exponentially everydy, even as we speak. The amount ofg data generated is more than it is humanly possible to handle, therefore we need a way to access this data easier and more convenient, by giving the information on the documents in the world wide web, we create an easier and more precise tool to retrieve relevant information. This enables computers and humans to interact in a more convenient fashion, the computers, given semantic data to the information, will have an “understanding” of the information it has, and this understanding enables us humans to use more natural language to describe the *things* we would like get more information about.

**Q5. What is Implicit Semantics? Give an example of how a semantic application can make use of implicit semantics.**

Implicit Semantics are when the semantics in the *refers* to other data in a non-explicit manner. An example of this can be observed as following: “Craig has a patient named Cindy”. This statement is *explicitly* stated in the data. To us humans, it’s a simple task to *infer* who the doctor is in that statement, but to a computer this is not the case. To establish to the computer we’ll have to find a way to make it understand this relation goes *both* ways, a person has a patient, it is therefore a doctor. We would then have to state that if a person has the relation “has a patient”, it is then seen as a doctor, only then the computer, with the right tools, could “understand” that if a person has a patient, it is then a doctor. The advantage of implicit semantic in this fashion shown, it’s causes less redundant data.

e.g. instead of assigning each and every patient in hospital to a doctor, it’s creates less redundant data to assign a patient to a person, to then say that if *person* has a patient it is then an active doctor at the hospital.

**Q6. Suppose you saw the following comment on a social media web site, and you had to**

**automatically infer what the content of the message was: “Lily I loved your cheryl tweedy do … heart Amy.” What are the challenges to assigning meaning? How would you try and solve the**

**problem?**

The challenges to such a message is to know which of the words to concatenate to get the  *meaning* out of it, in other words knowing which words “belongs” to another. There is somewhat a noticeable lack of grammar in the sentence, which clutters the exact meaning of the message, burt looking at the composition of the word, it is still possible to see the “bigger picture” of it. Deconstructing the sentence and taking it part by part might get a better view on it: “Lily I loved your cheryl tweedy do…” Analyzing this further we see that it contains a subject and a predicate with a possesive adjective “yours” and a direct object “cheryl tweedy do” further on we see three punctuation marks, which, in most cases indicates a pause. “heart Amy” could be seen as a “regards”, send their best wishes etc.

We have now established that the message is from “Amy” to “Lily”, and that “Amy” loves “Lily” ‘s “cheryl tweedy do”, the message also indicates relation between “Lily” and “cheryl tweedy do” the possesive adjective “your” on strengthens this statement, meaning that “cheryl tweedy do” belongs, or created by “Lily”. Without a provided context of what these parts of the message means, there is no way to reach an exact conclusion.

**Q7. What is the idea of “Linked Data”? Why is linked data useful? What are the four rules of linked data?**

The idea about Linked Data is to link the available data on the web to each other, very much like documents are linked to each other on the web, by that, I mean websites written and made up of HTML. Usually the *data*  contained in these documents are not linked to each other.

Therefore the W3C (World Wide Web Consortium) devised a plan to identify and link these huge amounts of data together. They found out that the most intuitive way to was to

use the same principles as it’s used to link the aforementioned HTML-documents, by using a widespread protocol, HTTP , simplifying the entire process. The data had to be assigned identifiers, which would point to another place on the web, e.g. a site that contained the meaning of the data. Further on, the data on the web had to be serialized and formatted in a way that computers had a way to “read” it, therefore the use of the standards for semantic data (RDF, SPARQL) was created. RDF for formatting ad serializing the data in a proper manner, SPARQL for retrieving the linked data and outputting it.

The URI would have pointers to other data, so that it would become a web of data, references upon references.

Most of the things described earlier is also called “The four rules” createsd by Tim Berner-Lee.

1. Create URI to name and describe *things*
2. Use HTTP so that other people can discover it
3. When someone looks up an URI, provide information using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discoer other *things*.

**Q8. What is “provenance”, and why is it important? What are the major components needed to track provenance as used in the Provenance Ontology?**

Provenance is about validating the data retrieved, its trustworthiness.

The web contains a lot of information decentralized information, and by looking things, it will provide wqith more information than necessesary, therefore a need for something to check for it’s validity is required.

In semantic applications the need for this is even greater, because it’s difficult to exactly determine where the information came from, the keyword in provenance is *metadata*.

For semantic applications, a provenance ontology was made to determine whether the information can be trusted or not. Provenance assertions are largely based on the use of metadata in the information. E.g. a file’s creation date-metadata might be relevant to it’s provenance, in the context of *how* it was created, but it’s file size does not. Provenance is often represented as metadata, but not all metadata is necessarily provenance.

**Q9. What is a Cloud Platform as a Service? Explain the four types of semantic annotation for**

**cloud services (data, logic/process, non-functional, system), and explain how each one is relevant**

**for Platform as Service?**

Four levels in cloud computing semantics.

System Semantics

Semantics pertaining to the system

Deployment and load balancing

Data Semantics

Semantics pertaining to data

Data typing (type being a typing, not the verb) storage, formatting, access and manipulation restrictions.

Logic and process Semantics

Semantics pertaining to the core functions of the applications

Programming language and runtime, exception handling

Cloud Software as a Service (SaaS)

The consumer can use the providers applications running on a cloud infrastructure. Available from *thin client* *interface*, such as web-browsers (webmail, google docs/drive) or native application interface. The end-user is given little to no control or management capabilities regarding cloud infrastructure, servers, network, storage, OS. and could have some possibilities to edit configurations within the applications.

Cloud Platform as a Service (Paas)

The consumer is given capabilities to deploy applications onto the cloud infrastructure, and utilize prog. languages, libraries, services and tools supported by the provider, to create services and applications. The consumer does not have access or control to underlying infrastructure; such as network, servers, OS or storage, but has control over its own applications, some config. for the application environments.

Cloud Infrastructure as a Service (IaaS)

Consumer has the capability to oversee and manage processing, storage, OS and other fundamental computer resources. With this the consumer can deploy and run arbitrary software, which can include OS, and applications. What can *not* be controlled is the underlying infrastructure, servers, and has limited network control over such components (e.g. hosts, firewall).

Side 6 av 6