Weekly exercises IN3020&4020 (2025)
Week 13: Graph Databases

1 Graph Databases/Knowledge Graphs Intro

1. What is a knowledge graph?

   Solution: The definition of a knowledge graph is not universally agreed upon, so you will find some variations. We can say that a knowledge graph is a knowledge base represented using a graphs structure. With a knowledge base one typically means data (e.a., entities or facts) together with semantic information so that the computer can "interpret" and "understand" the data.

2. Give examples of knowledge graphs technologies?

   Solution:

   - Google knowledge graph
   - Linked data semantic knowledge graphs (RDF)
   - Neo4j property path graphs

2 RDF graphs

1. What are some of the W3C certified standards for semantic technologies?

   Solution:

   - data format: RDF
   - ontologies: RDFS/OWL/OWL 2
   - queries: SPARQL
   - (rules: SWRL)
   - (shape constraints: SHACL)
   - ...

2. What is RDF?

   Solution: The Resource Description Framework(RDF) is a framework for data interchange on the Web

3. What is an RDF triple?

   Solution: An atomic data entity in the RDF datamodel consisting of a subject, predicate and object. E.g., <ex:john, ex:father-of, ex:bill>.

4. What is an RDF graph?

   Solution: A set of RDF triples.

5. An RDF graph is often visualized as a directed graph. It is, however, not really a graph. Why?

   Solution: A graph typically consists of two disjoint sets. The edges and the nodes. There is nothing in the definition of an RDF graph that says we cannot express things about a predicate. Consider <rdf:type, rdf:type, rdf:Property> .

6. Why do you think we use URIs to represent concepts in RDF?

   Solution: In relational databases, we need a key to identify a row in a table, RDF need an identifier that is globally unique. URIs are unique across the Web. URIs are also addresses so it would be easy to follow them to find data.

7. What is a literal in an RDF graph?

   Solution: Literals represent constants, typed or untyped.

8. Why do you think we need blank nodes?

   Solution: Blank nodes are used for several purposes. First, they can be used for anonymous (but distinguishable) URIs. Second, they can be used for adding more structure: containers, collections (lists), reification, etc.

9. How would you express the following statements in RDF (using any syntax, for example our toy syntax in lectures)?

   (a) IN3020 is a university course

   Solution: There can be many possible solutions here, one example: `(ex:IN3020, rdf:type, ex:UniversityCourse)`

   (b) IN3020 has a lecturer

   Solution: `(ex:IN3020, ex:hasLecturer, _:x)`

(c) IN3020 has a lecturer with the name "Egor", and he is a person.

Solution:

```
(ex:IN3020, ex:hasLecturer, _:x)

(_:x foaf:name, "Egor)
```

10. Draw a graph representing the following RDF graph in turtle syntax:

```
_:x rdf:type foaf:Person ;
    foaf:name ‘‘Bob’’ ;
    foaf:age   38 .
```

Solution:
This graph will have three edges, connecting a blank node to nodes foaf:Person, "Bob" and 38 by means of the corresponding edges.

11. What is the idea of reification?

Solution: *Dictionary.com:* "The act of treating something abstract [...] as if it were a concrete object." In RDF it means to write statements about statements by means of introducing a new blank node that "represents" the statement and connected to the original triple elements by means of three triples. This way we can overcome the limitation of binary relations and model more complex statements.

3 RDF/RDFS/OWL

1. What is the purpose of RDF vocabularies?

Solution: Reuse and interoperability. No need to reinvent when someone already has put a lot of effort into this.

2. What do we mean when we say that there are several RDF file formats (serializations)?

Solution: RDF is conceptually a graph (a set of triples), but there are several ways of reading/writing this information to a file. Some examples are RDF+XML, Turtle, N-triples.

3. What is RDFS?

Solution: A vocabulary for RDF where some URIs have special meaning (e.g., `rdfs:Class`, `rdfs:subClassOf`, `rdfs:domain`). The purpose of RDFS is to describe concepts in the domain on a semantic level (schema). It comes with a set of inference rules that lets us derive new facts mechanically (RDFS reasoning).

4. Use RDF/RDFS vocabulary to describe:

(a) `:Person`, `:Toyota`, `:Car` and `:Vehicle` have the type `rdfs:Class`

(b) `:drives` has the type `rdf:Property`.

(c) `:john` is a `:Person` (individual)

(d) `:my_car` is a `:Toyota` (individual)

(e) every `:Toyota` is a `:Car` (subclass)

(f) every `:Car` is a `:Vehicle` (subclass)

(g) `:john :drives :my_car`

(h) `:drives` has the domain `:Person`

(i) `:drives` has the range `:Vehicle`

Solution:

(a)
```
:Person rdf:type rdfs:Class
:Toyota rdf:type rdfs:Class
:Car rdf:type rdfs:Class
:Vehicle rdf:type rdfs:Class
```

(b) `:drives rdf:type rdf:Property`

(c) `:john rdf:type :Person`

(d) `my_car rdf:type :Toyota`

(e) `:Toyota rdfs:subClassOf :Car`

(f) `:Car rdfs:subClassOf :Vehicle`

(g) `:john :drives :my_car`

(h) `:drives rdfs:domain :Person`

(i) `:drives rdfs:range :Vehicle`

5. What is OWL?

   Solution: Web ontology language. The W3C recommended ontology language. Allows for much more detailed modelling (and reasoning) than RDFS.

6. What is an ontology?

   Solution: A formal description of the vocabulary (used in a domain of interest). Typically we speak about classes (categories), properties and relationship between concepts and data.

7. OWL uses the so called Open-World assumption, while relational databases use a Closed-World assumption. You can find some information about this here: `https://en.wikipedia.org/wiki/Open-world_assumption`

   (a) What do we mean by open/closed world assumption.

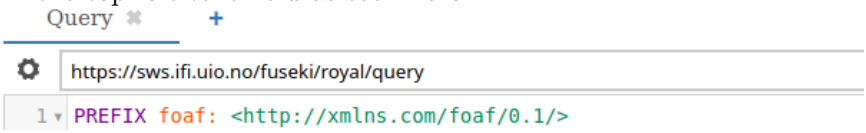
   (b) What practical implications can this have?

   (c) Why do you think it makes sense that OWL assumes an open world?

   Solution: With the open-world assumption (OWA), we mean that we assume that there is information about the world that are, for some reason, not in the database. This means that if we search for something that's not in the database, we can only conclude that "we don't know" if it exists or not. With the closed-world assumption, we assume that what's in the database is what is. So if something is not in the database, we conclude that it doesn't exist. This assumption also makes aggregation queries somewhat nonsensical. E.g., if we count the number of persons in the dataset and find 100, we can only conclude that "according to our data, we know that there are at least 100 persons. But we don't really know how many". It makes sense that OWL makes uses OWA because its role is to describe an evolving semantic web, which we cannot assume to have complete knowledge.

4 SPARQL

1. Go to: `https://sws.ifi.uio.no/yasgui/`.
2. Enter `https://sws.ifi.uio.no/fuseki/royal/query` in the top left text field as seen here:



Now you should be able to query a small RDF dataset with some information about the norwegian royal family. The graph is described in more detail in Appendix A on page 11. You will need the following prefixes for these queries:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

4.1 URI of Haakon Magnus

Find the URI of a person (a `foaf:Person`) with the name (`foaf:name`) `"Haakon Magnus"`.

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?uri
WHERE {
  ?uri a foaf:Person ;
       foaf:name "Haakon Magnus"
}
```

4.2 Sonja's age

Find the age (`foaf:age`) of a person (`foaf:Person`) with the name (`foaf:name`) "Sonja".

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?age
WHERE {
  ?sonja a foaf:Person ;
         foaf:name "Sonja" ;
         foaf:age ?age
}
```

4.3 Young family members

Find name and age of all persons that have age below 18. You need the datatype propery `foaf:age`, the class `foaf:Person`.

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?age
WHERE {
  ?person foaf:age ?age .
```

```
  FILTER (?age < 18)
}
```

4.4 The name of of Haakon Magnus' spouse

Find the name (`foaf:name`) of the spouse of (`fam:hasSpouse`) a person (a `foaf:Person`) with the name (`foaf:name`) "Haakon Magnus".

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?spouseName
WHERE {
  ?haakon a foaf:Person ;
          foaf:name "Haakon Magnus" ;
          fam:hasSpouse [ a foaf:Person ;
                          foaf:name ?spouseName ] .
}
```

or:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?spouseName
WHERE {
  ?haakon a foaf:Person ;
          foaf:name "Haakon Magnus" ;
          fam:hasSpouse ?spouse .

  ?spouse a foaf:Person ;
          foaf:name ?spouseName .
}
```

4.5 The parents of Olav V

Find the name of both the parents of Olav V. E.g., A person (`foaf:Person`) with the name (`foaf:name`) "Olav V". Two different properties connect a person to his/her parents: `fam:hasFather` and `fam:hasMother`.

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?parentName
WHERE {
  ?olav a foaf:Person ;
        foaf:name "Olav V" .

  { ?olav fam:hasMother ?mother .
    ?mother foaf:name ?parentName }

  UNION

  { ?olav fam:hasFather ?father .
    ?father foaf:name ?parentName }
}
```

or

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?parentName
WHERE {
  ?olav a foaf:Person ;
        foaf:name "Olav V" ;
        fam:hasFather|fam:hasMother [ foaf:name ?parentName ]
}
```

4.6 URI, name and homepage

Find the URI, the name and the homepage (if they have one) of all persons in the dataset. You will need the class `foaf:Person` and the properties `foaf:name` and `foaf:homepage`.

Solution:

```
PREFIX fam: <http://www.ifi.uio.no/IN3020/family#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?person ?name ?homepage
WHERE {
?person a foaf:Person ;
```

```
        foaf:name ?name .
      OPTIONAL { ?person foaf:homepage ?homepage }
}
```

4.7 DBPedia

It is possible to query large knowledge graphs online with SPARQL. Examples of such datasets are Wikidata and DBPedia.

Change the SPARQL endpoint to `https://dbpedia.org/sparql`. You need the following prefixes:

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
```

Find the population of the capital of a country called (`rdfs:label`) "Norway"@en. Where `dbo:Country` is the class of countries. `dbp:capital` is the relation between a country and its capital. `dbp:populationTotal` is the total population.

Solution:

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?capitalPopulation
WHERE {
  ?norway a dbo:Country ;
          rdfs:label "Norway"@en ;
          dbp:capital [ dbp:populationTotal ?capitalPopulation ] .
}
```

5 Neo4j and Cypher

1. Describe the data model of Neo4j

   Solution:

   - Node
   - Label: a type to a node
   - Relationship: directed edge between two nodes
   - Relationship type: used to characterisze a relationship

- Property: Key-value pair. Both nodes and relations can have

2. Go to `https://neo4j.com/sandbox` and create an account. Choose the *Movie* dataset and work through the Movies Project Tutorial.

A Excerpt RDF graph

Below is a short excerpt from the royal family graph. It consists of persons with names, spouses, ages, homepages, fathers and mothers:

```
fam:Sonja    a  foaf:Person ;
    fam:hasSpouse  fam:haraldv ;
    foaf:age       83 ;
    foaf:name      "Sonja" ;
    foaf:homepage
        <https://www.kongehuset.no/seksjon.html?tid=27197&sek=26953> .

fam:haakonmagnus    a   foaf:Person ;
    fam:hasFather  fam:haraldv ;
    fam:hasMother  fam:Sonja ;
    fam:hasSpouse  fam:mettemarit ;
    foaf:age       47 ;
    foaf:name      "Haakon Magnus" ;
    foaf:homepage
        <https://www.kongehuset.no/seksjon.html?tid=27658&sek=27226> .
```