

# Exam

## Web Data Management

### Master Parisien de Recherche en Informatique

March 4th, 2020

This is the final exam for the Web Data Management class, which will determine 50% of your grade for this class (the other 50% being given by the project). The exam consists of two independent exercises. **You must write your answer to each exercise on a separate sheet of paper.** You can choose to answer the questions in English or in French, as you prefer.

Write your name clearly on the top right of every sheet used for your exam answers, and number every page.

You are provided with an SPARQL “cheat sheet” which gives you a summary about the syntax of SPARQL. You are additionally allowed **one A4 sheet** (i.e., two pages, one on each side) with the content of your choice. You may not use any other written material.

The exam is **strictly personal**: any communication or influence between students, or use of outside help, is prohibited. No electronic devices such as calculators, computers, or mobile phones, are permitted. Any violation of the rules may result in a grade of 0 and/or disciplinary action.

### Exercise 1: Provenance for SPARQL queries (10 points)

We assume that we have a namespace `m`, and we consider the following RDF document  $D_0$  in N-Triples notation, seen as a set of RDF facts (ignore for now the comments preceded by `#`):

```
<m:joe> <m:likes> <m:sparql> . #l1
<m:jane> <m:likes> <m:sparql> . #l2
<m:jane> <m:likes> <m:xpath> . #l3
<m:joe> <m:follows> <m:jane> . #f1
<m:jane> <m:follows> <m:joe> . #f2
<m:jack> <m:follows> <m:joe> . #f3
<m:jack> <m:follows> <m:jane> . #f4
```

We consider the SPARQL query  $Q_0$  with two free variables `?x` and `?y` and returning all answer pairs  $(?x, ?y)$  such that `?x` follows someone who likes `?y`.

**Question 0.** Write  $Q_0$  in SPARQL and write the output  $Q_0(D_0)$  of  $Q_0$  on  $D_0$ , i.e., the answers returned by  $Q_0$  on  $D_0$ .

In the rest of the exercise, we will study the *Boolean provenance* of queries. Remember that a *valuation* of  $X_0$  is a function  $\nu : X_0 \rightarrow \{0, 1\}$ , and a *Boolean function*  $\phi$  on  $X_0$  is a mapping from valuations of  $X_0$  to  $\{0, 1\}$ : given a valuation  $\nu$  and a Boolean function  $\phi$ , we say that  $\phi$  *evaluates to true on*  $\nu$  if  $\phi(\nu) = 1$ , and otherwise that it *evaluates to false on*  $\nu$ . We will write Boolean functions as Boolean formulas. For instance,  $l_1 \wedge (l_2 \vee f_2)$  is a Boolean function. It evaluates to true under the valuation  $\nu_0$  mapping  $l_1, l_2, f_1, f_4$  to 1 and  $l_3, f_2, f_3$  to 0.

We annotate each fact of the RDF document  $D_0$  above with a Boolean variable in  $X_0 = \{l_1, l_2, l_3, f_1, f_2, f_3, f_4\}$  in the order indicated. Given a valuation  $\nu$  of  $X_0$ , we write  $\nu(D_0)$  to be the RDF document consisting of precisely the facts of  $D_0$  that  $\nu$  maps to 1, formally  $\nu(D_0) := \{f \in D_0 \mid \nu(f) = 1\}$ .

**Question 1.** Write  $\nu_0(D_0)$  for the example valuation  $\nu_0$  above, and write the output of  $Q_0$  on  $\nu_0(D_0)$ .

Given a SPARQL query  $Q$  and an answer  $\mathbf{a}$  for  $Q$  on  $D_0$ , the *provenance* of  $\mathbf{a}$  for  $Q$  on  $D_0$  is the Boolean function  $\phi$  on the variables  $X_0$  such that, for any valuation  $\nu$  of  $X_0$ , we have that  $\phi$  evaluates to true under  $\nu$  if and only if  $\mathbf{a}$  is an answer to  $Q$  on  $\nu(D_0)$ . For instance, the provenance of the answer (*jane*) for the following SPARQL query on  $D_0$ :

```
SELECT ?x WHERE {  
  ?x <m:likes> ?y  
}
```

is  $l_2 \vee l_3$ .

**Question 2.** Write again the output of  $Q_0$  and  $D_0$  like in Question 0, and write next to each answer of the output its provenance for  $Q_0$  on  $D_0$ .

**Question 3.** We say that two SPARQL queries  $Q$  and  $Q'$  are *equivalent* if, for every RDF document  $D$ , we have  $Q(D) = Q'(D)$ , i.e.,  $Q$  and  $Q'$  have the same output on  $D$ .

Give an example of two SPARQL queries  $Q$  and  $Q'$  that are not equivalent but have the same output on  $D_0$  and for which every tuple  $\mathbf{a}$  of their common output has the same provenance for  $Q$  and  $Q'$ .

**Question 4.** A SPARQL query  $Q$  is *monotone* if, given two RDF documents  $D$  and  $D'$ , if  $D \subseteq D'$  (meaning that every fact of  $D$  is in  $D'$ ), then  $Q(D) \subseteq Q(D')$  (meaning that every tuple in the output of  $Q$  on  $D$  is also in the output of  $Q$  on  $D'$ ). Is the query  $Q_0$  monotone? Give a short proof or a counterexample.

**Question 5.** Given two valuations  $\nu$  and  $\nu'$  on  $X_0$ , we write  $\nu \subseteq \nu'$  if for all  $x \in X_0$  such that  $\nu(x) = 1$ , we have  $\nu'(x) = 1$ . A Boolean function  $\phi$  on  $X_0$  is *monotone* if, for every valuations  $\nu \subseteq \nu'$ , if  $\phi$  evaluates to true on  $\nu$ , then  $\phi$  evaluates to true on  $\nu'$ .

Show that, for any monotone SPARQL query  $Q$ , for any answer tuple  $\mathbf{a}$  of  $Q$ , the provenance of  $\mathbf{a}$  for  $Q$  on  $D_0$  is a monotone Boolean function.

**Question 6.** Give an example of a SPARQL query  $Q_6$  having an answer  $\mathbf{a}_6$  such that the provenance  $\phi$  for  $Q_6$  on  $D_0$  is a Boolean function  $\phi$  which evaluates to true under *exactly* one valuation.

**Question 7.** We say that a Boolean function  $\phi$  on  $X_0$  *implies* a Boolean function  $\psi$  on  $X_0$  if, for every valuation  $\nu$  of  $X_0$ , if  $\phi$  evaluates to true under  $\nu$  then  $\psi$  also does.

Can there be a SPARQL query  $Q_7$  having two answer tuples  $\mathbf{a}$  and  $\mathbf{b}$  whose respective provenances  $\phi$  and  $\psi$  are different Boolean functions such that  $\phi$  implies  $\psi$ ? Give an example, or prove that this is impossible.

**Question 8.** We consider the problem of computing the provenance of a SPARQL query, defined as follows. We fix a SPARQL query  $Q$  of the form `SELECT ?x1 ... ?xn WHERE { ... }` where the variables of the `SELECT` clause do not use any operator, and where the body of the `WHERE` clause only consists of RDF facts where the predicate of each fact is a constant and where the subject and object of the facts can be constant or variables (without any operator or other feature of the SPARQL syntax). We want, given an RDF document  $D$  with facts annotated by variables, to compute all answers of  $Q$  on  $D$  and their provenance. We study the *data complexity* of this problem, with the input being the annotated RDF document  $D$ .

Design an algorithm to solve this problem with polynomial data complexity, and argue that it is correct.

**Question 9.** Open question: Some informations usually tracked by semiring provenance are not tracked in the definition of provenance proposed above, in particular the *number of uses of a given fact* and the *number of ways to use a set of facts*.

Explain these shortcomings by showing examples of queries and documents where some answer tuples has the same provenance in the sense above, but should have a different provenance if we accounted for these additional informations.

Propose ways in which we could define an extended notion of provenance for SPARQL (or some fragment thereof) to capture these additional informations, and explain how they could be computed.

Which SPARQL features would be problematic for this extended provenance notion?

*Please write your answer to the second exercise on a separate sheet of paper.*

## Exercise 2: Web search and Big Data management (10 points)

**Question 1.** Explain the difference between *coverage* and *freshness*, as alternative objectives in a crawler strategy, and how they may impact that strategy.

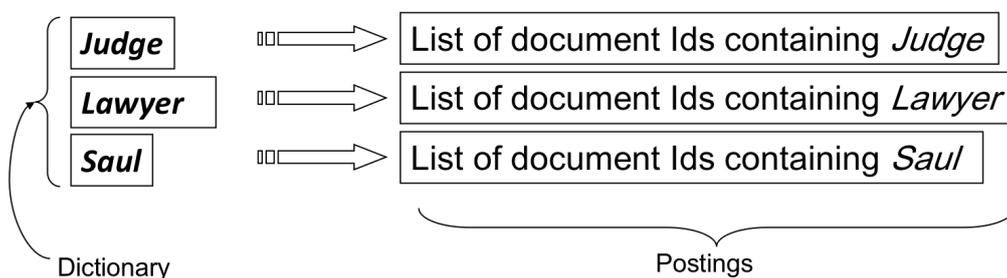
**Question 2.** Explain *binary freshness* in the context of a crawler, and name one drawback of this freshness indicator.

**Question 3.** Explain *link pollution* and how a search engine could avoid it.

**Question 4.** Explain why Web graphs must be compressed, and name two specificities of adjacency lists (of the Web graph) that can help in this regard.

**Question 5.** Explain why inverted indexes must be compressed, and explain one technique that can help in this regard.

**Question 6.** Boolean queries are particularly helpful/frequent for text retrieval in law research services. In a simple version of an inverted index designed to answer Boolean queries, we would keep, for each word (in the dictionary), the list of documents in which it appears (its *posting list*). Let us assume only document Ids are kept in posting lists, as in the following example:



1. Posting lists may be very large. Which condition is necessary in order to be able to process queries such as *Judge AND Saul* efficiently (say, in linear time in the size of the posting lists)?
2. What would be a good logic to process a query such as *Judge AND Saul AND Lawyer*?

3. Assuming that we only keep the dictionary in main memory, but not the posting lists, which additional information do we need to keep in the dictionary for the previous logic to work?
4. How could that logic be extended to handle OR queries, such as *Judge OR Lawyer*?

**Question 7.** Open question: We want to build a search engine for *sports* only. How would you do that (lay down the main changes with respect to a general search engine design)?

**Question 8.** To build a search engine for *sports*, how could we compute a topic-specific PageRank? Is the sports PageRank score of a page always smaller than the general PageRank score of that page?

**Question 9.** Explain the fault tolerance and crash recovery logic of a Big Data Management System.

**Question 10.** Name one reason why Spark can be considered as “Hadoop compatible”, so that it can be appealing as an alternative data processing engine for Hadoop users.