IIC3432 — Advanced topics in databases

Homework 2: explanation and projects

1. Administrative stuff

You should form groups of two people each. Your group will select one of the projects detailed below, investigate the topic, and present the findings in the class on 28th of September. Each group will have precisely 30 minutes for a presentation, including discussion and questions. **Recall that you have to change the group compared to the last homework assignment.**

2. FAQ

Do we need to hand in the work we do?
No, you will be evaluated based on the presentation.

What should be the structure of my presentation?
You have 30 minutes, so there is sufficient time to quickly explain the setting and background, and then detail the specific problem that is tackled, and how is this resolved. The idea is that all the students in the class should be able to follow your presentation, not just the lecturer.

We have a size 9 font in our presentation so that all the information fits on the slides. Is this OK?
No! An important part of the evaluation is your ability to summarize the work you did in a short presentation. This is no easy task. In particular, a good short presentation requires better understanding of the topic than a bad long presentation. In this course you will need to be able to isolate the important details of your work and present it in an accessible way.

I don't like any of the proposed topics.
Propose a new topic. For this, send me an email (dvrgoc@ing.puc.cl).

We're stuck, we don't know how to proceed, or can not decide what to do next.
That's what I'm here for. During next week, I'll be around during the lecture hours, so if you decide to come to the lecture room and work at that time, you can ask me questions directly (I'll be in my office, or in the lecture room). Also, you can send an email at other times. Some projects have a lot of work in them, so if you do not finish all of it you still might get the highest score if sufficient effort is put into it.

3. Projects

A representative of the group should send me an email ccing the other person in the group with two preferences for a project they would like to do. It is recommendable that you do this before Wednesday 20th of September at 15pm to get the topic you wish to work on. One topic can not be taken by two groups.

3.1. Benchmarking SPARQL systems - Berlin

There is a wide variety of different systems that implement the SPARQL query standard. To determine when an implementation handles the SPARQL standard correctly several benchmarks that model real-world use cases of SPARQL were developed. In this project your task is to explore the Berlin SPARQL benchmark: `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/`.

In particular, you should get familiar with the datasets and queries used here, and check how they run on different SPARQL engines. Some of the most popular engines running SPARQL standard are the BlazeGraph query engine `https://www.blazegraph.com`, Apache Jena `https://jena.apache.org/` and Open Link Virtuoso `http://vos.openlinksw.com/owiki/wiki/VOS/`. I strongly suggest you select those engines for your evaluation. In the class you should present both the benchmark (datasets and queries), and the results you obtained by running the benchmark on different implementations. (You select how many different engines you use).

3.2. Benchmarking SPARQL systems - LUBM

This is a variation of the previous project, but now you should consider the Lehigh University Benchmark (LUBM), available at `http://swat.cse.lehigh.edu/projects/lubm/`.

Similarly as for the Berlin benchmark, you should understand and present the structure of the test cases, and use the benchmark to run queries over different SPARQL engines.

If you are having problems with the data format I suggest reading `http://www.linkeddatatools.com/semantic-web-basics`, parts 1–5. Generally SPARQL engines know how to import ontologies, so you can also explore this a bit.

3.3. Can SPARQL beat SQL on Berlin?

In the last homework assignment we explored how the performance of Neo4j measures to that of a SQL engine. In this project you are to do the same, but now with SPARQL. In particular, you should select one SPARQL engine to install, as well as one SQL engine, and look into the specification of the Berlin SPARQL benchmark `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/`, which also generates SQL queries. You

should therefore create an RDF and a relational dataset, and test the two engines on the corresponding queries.

3.4. Can SPARQL beat SQL on gMark?

This is similar to the previous task, but this time we will concentrate on path queries. More precisely, you should explore the gMark graph database benchmark available at `https://github.com/graphMark/gmark`. This is a graph database benchmark generating both graphs and queries. It can generate queries in SPARQL and SQL. More details behind the project can be found in the paper `https://arxiv.org/pdf/1511.08386.pdf`. Clone the benchmark and see what types of queries it will generate. (Note: There is a small error when running the github demo; namely some XML files are missing the `size` parameter. The file `test.xml` has the correct configuration, so you should check that one and imitate it. Size is just the number of graphs that will be used. The important thing is to configure the parameters at the very end of the XML file. Also, the generator might use the SPARQL abbreviation syntax {,3} which is supported only in Virtuoso).

In this task, you should select one SPARQL engine to install, as well as one SQL engine, and test how they compare when using the queries and datasets generated by gMark.

3.5. Real world path benchmark

At this moment there is no benchmark for SPARQL property paths over real world data (tools such as gMark that we introduced in the previous block of projects generate their own data). In this project, your task is to construct one such benchmark over DBPedia. The benchmark should contain at least 10–15 queries that use property paths with a Kleene star, and whose meaning you will have to explain (i.e. arbitrary queries will not do). Your queries should also have answers over the DBPedia dataset. Test your queries on the DBpedia endpoint, and discover which ones run well, and which ones do not. Test how they perform if only a fixed number of iterations is given instead of a Kleene star. At which point does the query not execute? Try this both with {,3} for a repetition, or by writing the path under the Kleene star three times explicitly (i.e. a/a/a instead of {,3}).

3.6. SPARQL SERVICE

SPARQL has many operators that make it a true Web database system. One such operator is SERVICE, which allow SPARQL queries to connect to other endpoints, run a query there, and incorporate the data obtained from another server into its own query results.

In this project, you task is to explore how the SERVICE operator should work according to the SPARQL standard (see `https://www.w3.org/TR/sparql11-federated-query/`), and to give us several examples of queries showcasing how SERVICE can be used.

In particular, for this task you should explain what the SERVICE operator is, what are its syntax and semantics, and also design at least 10 queries that test how well this operator performs over the SPARQL endpoint infrastructure, and what are some of the issues it can run into.

For examples of SERVICE queries, see e.g. `http://marenas.sitios.ing.uc.cl/publications/jws12.pdf`. You **are** allowed to use some queries from other sources, but you should be clear about this. You should also design some of the queries on your own.

3.7. Implementation of SPARQL SERVICE

The Apach Jena project provides an open source implementation of a SPARQL engine available at `https://jena.apache.org/`. In this project your task is to explore how the SERVICE operator, described in the previous project, is implemented in the Jena database. For this you should familiarize yourself with how SERVICE should be implemented (see the standard `https://www.w3.org/TR/sparql11-federated-query/`), explore how the implementation is carried out at a high level, and then run some service queries through the Jena pipeline.

3.8. Implementation of SPARQL property paths

The Apach Jena project provides an open source implementation of a SPARQL engine available at `https://jena.apache.org/`. In this project your task is to explore how path queries are implemented in Jena. The specification of SPARQL property paths is available at `https://www.w3.org/TR/sparql11-query/#propertypaths`. The main objective is to see how Jena treats property paths at a high level, and see how it executes them in its pipeline.

3.9. Evaluating property paths over Linked Data

In class on Linked Data we saw one way to implement specific property path queries. The objective of this project is to generalize this and create an evaluator for arbitrary property path queries. For this, you should assume that as input you receive one string containing a query of the form $< IRI >$ pp ?x, where $< IRI >$ is your starting point in Linked Data where you start evaluating the property path, pp is a property path in the standard SPARQL syntax (i.e. it is of the form `a/b/d*` etc.), and ?x is the IRI you are searching. Your task is to build an evaluator that decomposes the property path pp into an automaton, and then evaluates this automaton using breadth first search or depth first search (your choice). You should try your implementation with 5 different property path queries running over Linked Data and report on the results. In the presentation you are supposed to give us an overview of your implementation and show us how you implemented the tokenizer (the thing decomposing the input string) and the search algorithm itself.

3.10. Which search algorithm is better over Linked Data?

In class we saw that property path queries can be evaluated rather efficiently over Linked Data using standard search algorithms. Your task here is to explore the behaviour of these algorithms in more depth. In particular, you are supposed to create a test suite of queries (you can reuse the ones from class, but should add some new ones), evaluate them using BFS and DFS, and test how well they perform when you want to retreive 10, 50, 100, and 500 query answers. You should measure the time needed to get the answers, as well as the number of triples received, the number of requests to the Linked Data service, and the amount of memory your program uses. Discuss when BFS/DFS is superior and why. You could also set up your own junk linked data repository to test some of your claims, but this is not a requirement.

3.11. Propose your own topic

Quite simple: send me an email (dvrgoc@ing.puc.cl) to confirm the topic you would like to explore.