Open Information Systems

Project Specification: Second Session 2021-2022

VRIJE UNIVERSITEIT BRUSSEL

Professor

Prof. Dr. Bas Ketsman <bas.ketsman@vub.be>

Teaching Assistants

Tim Baccaert <tim.jan.baccaert@vub.be>
Samuel Ngugi Ndung'u <samuel.ngugi@vub.be>

Introduction

The goal of this project is to design and implement an information system using techniques seen during the lectures. You will individually develop one aspect of a large system, there is no group aspect to the second session project.

For this project, you will design a **ticket booking system** for Brussels Airlines. Due to the lingering COVID crisis, the company had to end its contract with a consultancy firm who built and maintained its software systems. Instead, the upper management has decided to invest in hiring in-house developers to build a completely new system, such that the maintenance cost can be controlled more easily.

The company has a number of **minimal requirements** for this subsystem:

1. Potential passengers should be able to **book a flight**.
    - A flight is from one airport to another airport[1], potentially with layovers (connecting flights) via a different airport[2]. Keep in mind that each of these flights starts and ends at a given date and time.
    - A flight can be intercontinental (i.e., from Europe to North America) or continental (i.e., within Europe).
    - Remember that there are several price classes based on the size of the plane. The cheapest tickets are usually *economy*, followed by *business class*, and *first class*. Smaller planes will typically lack first class seats.
    - Some passengers want to check in their luggage, each suitcase should add a flat fee to the ticket price.
    - Furthermore, some passengers want an extra insurance so that they can cancel or reschedule their flight without loss of money.
2. Handling agents working on behalf of the airline should be able to view the **contact information of passengers** in order to inform them of changes to their booking. Additionally, they can **upgrade or change the flight bookings** of their passengers.
    - Contact information includes first and last name, an email address or a phone number.
    - Passengers could have a frequent flyer number, once they pass a certain treshold of distance traveled they are eligable to be upgraded to business or first class by a handling agent.
3. Agents of the airline should be able to **(re-)assign planes to certain flights**.

[1] Limit yourself to about 10 airports.
[2] You may limit yourself to a flights with a single layover for the purposes of this project.

- The system keeps track of capacities of planes such that the agent can verify if the passengers of a given flight can fit on a specific plane type.
- Store the fuel type required by a plane, that is, kerosene (for turbine engines and diesel engines) or gasoline (for piston engines), and the fuel capacity of the plane. This allows the agent to estimate fuel costs.

Note that these are the minimal requirements, you can always add more features as you see fit. The precise modelling of the data system, and the actual data you use are entirely up to you. You could base yourself on the data of existing platforms, you can find open data, you can scrape data, or you could use a dummy data generator such as Mockaroo[3]. But, be sure to **mention your data sources** in the report.

Part 1: ER Diagram

Based on the requirements outlined in the introduction, you can start modelling your data system using an **Entity Relationship (ER) diagram**. The techniques for this have been handled in the first bachelor's Database course here at the VUB. Since this course also has students from other universities, these students might want to look at online refreshers[4]. In any case, please keep the following guidelines in mind:

- Make sure to annotate relationships with **cardinalities**, use numeric notation (i.e. 0..*, 1..*, 0..1, etc.), and place the relationship name in a diamond.
- Try to use **natural keys** as much as possible (i.e. keys that consist of one or more attributes). Only use an artificial identifier (i.e. UUID, etc.) when it's unavoidable, and motivate why (on the diagram itself).
- You are free to use any tools[5] for the ER diagram, but make sure the diagram is in a `.pdf` format.

Part 2: Database System

With your ER Diagram finished, you can now start creating your relational database. We require that you install and use PostgreSQL 13[6], as seen in the first lab. Please limit yourself to the usage of **standard SQL** features. That means that you should not use PostgreSQL's extended types such as JSON.

Based on your ER Diagram you should create the necessary **database, tables; primary and foreign key constraints**. You are allowed to use a graphical user interface for this, it is included in the distribution of PostgreSQL that you installed, under the name of 'pgAdmin4'. As seen in the first lab, you should create a `.sql` dump of your database as a deliverable. Use the naming convention `lastname-firstname-database.sql`

Finally, you should **demonstrate the functionality** of your database to us by writing a number of queries. The idea is that you provide us with at least 5 queries (i.e., SELECT-statement, inserts and updates are not queries) expressing usecases described in the introduction. The idea is that you write these queries into a `.sql` file. Adding extra comments with further explanation is appreciated.

[3] https://mockaroo.com/
[4] The wikipedia article on this is quite alright for our purposes, https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model.
[5] https://app.diagrams.net/
[6] https://www.postgresql.org/

Part 3: Information System

You will now augment your database with an ontology to turn it into a richer **information system**. To do this, we will be using the virtual Knowledge Graph system Ontop[7]. For information on how Ontop works, you can look at the lab sessions for R2RML and SPARQL.

The first task is to build an ontology. The idea is to explicitly encode the meaning behind your database into an **ontology**. We expect an **OWL ontology** using **RDF/turtle** syntax. Keep the following guidelines in mind:

- Make sure to add **explanations** to your classes and properties using `rdfs:comment`.
- Try to base your definitions on other ontologies you find online, and mention these ontologies in your report if you do.
- You can generate a visual representation of your ontology using WebVOWL[8], but this is not mandatory.

After creating your ontology, you are ready to work on a **relational mapping**. We will use R2RML (i.e. RDB to RDF Mapping Language) in order to translate between our relational databases and RDF triples. You can test whether your mapping is done correctly by manually inspecting the triples generated by the `ontop materialize` command[9]. You should submit both the **R2RML mapping file** and the **data file** in the RDF/Turtle syntax, using the file naming conventions `lastname-firstname-mapping.ttl` and `lastname-firstname-data.ttl` respectively.

With a working mapping, you can create an endpoint using the `ontop endpoint` command[10], use this to come up with SPARQL queries that query across the multiple databases. Try to translate your **5 SQL queries** from part 2 into SPARQL queries, and include these queries in your report.

Lastly, you should write a report with the following structure:

- **Introduction**: Discuss the overall design of your ERD and how you decided to model the scenario in the introduction, and mention the source of the **data** you used. If anything is out of the ordinary, make sure to mention it here.
- **Ontology**: Discuss the design decisions made in your Ontology. Which constraints did you use and why? Did you use OWL2 QL, EL or RL, and why? Did you align your ontology with existing ontologies available on the Web, and why?
- **Mapping**: Discuss the design decisions you made for your R2RML mapping.
- **Queries**: Show and discuss the 5 SPARQL queries you created, and motivate which use-cases they solve.
- **Discussion**: Conclude the report, reflect on what went well and what went poorly. Properly address any shortcomings.

Please do not forget to **mention your enrollment number, name, email address, and the current academic year**. Make sure to structure your report properly, and pay attention to your writing. This is a good preparation for your thesis.

Deliverables and Deadline

Below is a short summary of all the deliverables that we expect for this project. You can submit this project to the appropriate section on Canvas as a single `lastname-firstname-ois.zip` archive.

[7] https://ontop-vkg.org/
[8] https://github.com/VisualDataWeb/WebVOWL
[9] https://ontop-vkg.org/guide/cli.html#ontop-materialize
[10] https://ontop-vkg.org/guide/cli.html#ontop-endpoint

- **ERD**: Your ERD file in a PDF format, with the filename `lastname-firstname-erd.pdf`.
- **Database**: All the necessary statements that **create your database** and **insert your data** into the database. Leave a comment with the sources of your data if applicable, with filename `lastname-firstname-database.sql`.
- **Queries**: Give at least 5 queries based on the description in the introduction, with filename `lastname-firstname-queries.sql`.
- **Ontology**: an RDF/Turtle file with the filename `lastname-firstname-ontology.ttl`.
- **Mapping**: an RDF/Turtle file with the filename `lastname-firstname-mapping.ttl`.
- **RDF Data**: an RDF/Turtle file with the filename `lastname-firstname-data.ttl`.
- **Report**: A pdf document with the filename `lastname-firstname-report.pdf`.

The deadline for this project is **Sunday, 28th of August** at **23:59**.

Evaluation

This project consists of **60%** of your final grade for this course. You need at least 8/20 on the project in order for your grade to count. When you submit late, **within 24 hours** after the deadline this will result in **-4/20** for the total grade of the project. Subsequently, **after 24 hours** you will receive **0/20** for the project. We will use the following grade distribution for each part of the project:

Task	Grading
Part 1: ER Diagram	10%
Part 2: Database System	10%
Part 3: Information System	50%
Project Defense	30%

We would like to emphasize that plagiarism **of any kind** has a **zero-tolerance** under any circumstances, it will be dealt with and reported to the dean of the faculty (c.f. OER). This includes copying database layouts from the web or from other students, copying ER diagrams, not citing sources in the report, and not citing the sources of your data.