OSW Semestral work CP2

Public transport use in the capital and its effect on park+ride occupancy and traffic events

Tadeáš Binder

Motivation

It is in any city’s best interest to decrease congestion and road accidents as much as possible. One way to do so is to create parking facilities near major public transport centers on the edge of the city (commonly referred to as “park+ride”) - so that travelers can conveniently park and use public transport, thereby easing on traffic and pollution in the centre.

Studying data concerning park+ride occupancy, traffic events near the park+rides and the usage of public transport can reveal just how significant the link between the three is. This can be useful for the city’s public transport and road network authorities, since they can determine whether building park+rides is worth it at all, whether they add choke points/traffic accidents to the roads in the city, or if there are other locations where they ought to be built.

Possible questions

- Are P+Rs effective at reducing traffic in the centre?
- Are there any measurable impacts of P+Rs on public transport pass sales?
- Are there any traffic hot-spots where building new P+Rs would be beneficial?
- Are there any P+Rs which are in need of expansion, reduction or outright removal?
- Does the existence of P+Rs exacerbate traffic/accidents at roads near the P+Rs?

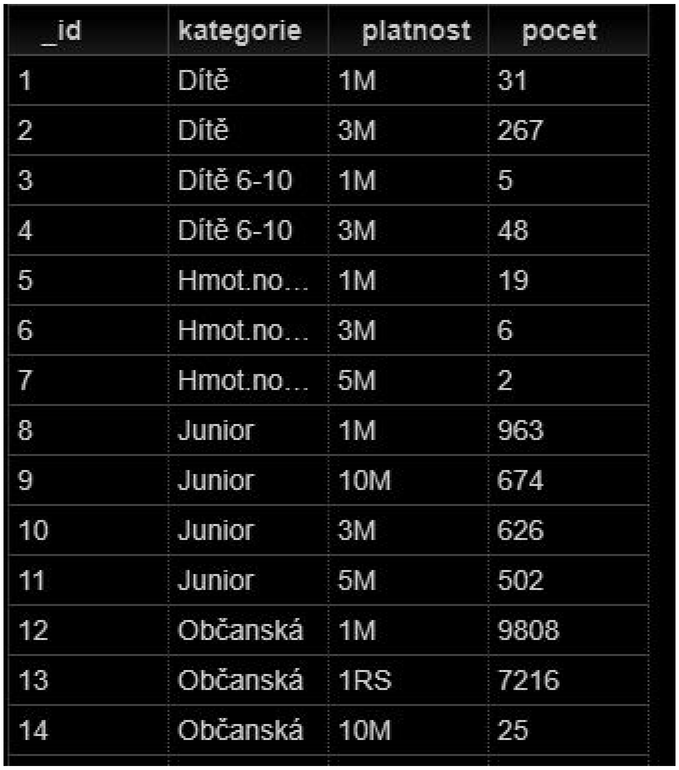
Data sources

We are going to use the following 4 data sources for this project:

Public transport coupon sales

Provider: Dopravní podnik hl. m. Prahy
Source: http://opendata.praha.eu/dataset/dpp-statistiky-prodanych-kuponu
Datatype: CSV

DPP offer other ways of tracking usage of its services, but this one is the most granular out of them all, as the data is updated each day. Some “stitching” will be required, as the data is separated into different files by type of sale (e-shop, card or paper) and by month. Park+ride users are very likely to commute, and as such use coupons rather than tickets.

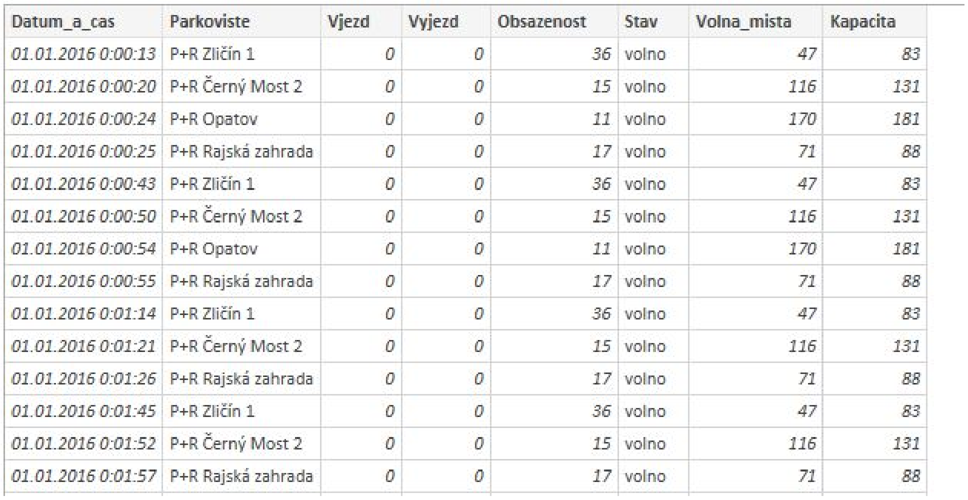


_id	kategorie	platnost	pocet
1	Dítě	1M	31
2	Dítě	3M	267
3	Dítě 6-10	1M	5
4	Dítě 6-10	3M	48
5	Hmot.no…	1M	19
6	Hmot.no…	3M	6
7	Hmot.no…	5M	2
8	Junior	1M	963
9	Junior	10M	674
10	Junior	3M	626
11	Junior	5M	502
12	Občanská	1M	9808
13	Občanská	1RS	7216
14	Občanská	10M	25

Park+ride occupancy data

Provider: Operátor ICT, a.s.
Source: http://opendata.praha.eu/dataset/parkovani_pr
Datatype: CSV

Provides information about the usage of park+rides every few minutes (how many spots are full/available at each location). Such detail is unnecessary for our purposes; therefore the data will be condensed.

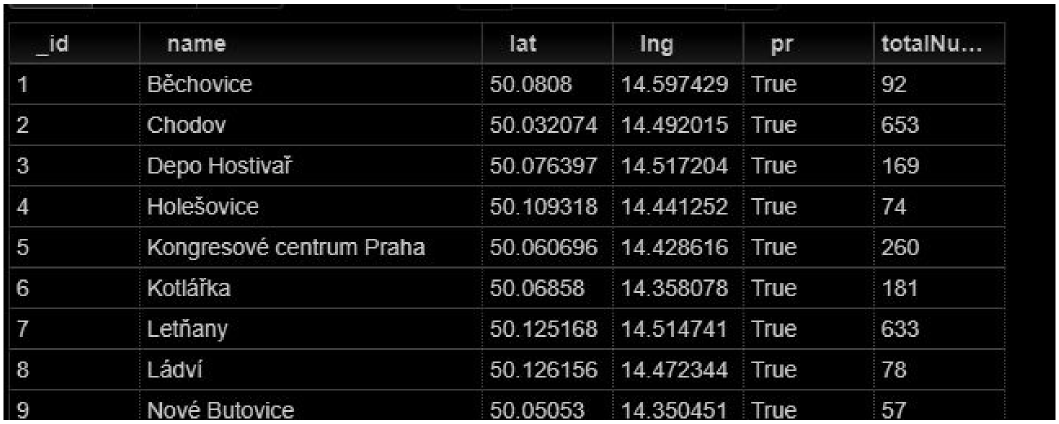


Datum_a_cas	Parkoviste	Vjezd	Vyjezd	Obsazenost	Stav	Volna_mista	Kapacita
01.01.2016 0:00:13	P+R Zličín 1	0	0	36	volno	47	83
01.01.2016 0:00:20	P+R Černý Most 2	0	0	15	volno	116	131
01.01.2016 0:00:24	P+R Opatov	0	0	11	volno	170	181
01.01.2016 0:00:25	P+R Rajská zahrada	0	0	17	volno	71	88
01.01.2016 0:00:43	P+R Zličín 1	0	0	36	volno	47	83
01.01.2016 0:00:50	P+R Černý Most 2	0	0	15	volno	116	131
01.01.2016 0:00:54	P+R Opatov	0	0	11	volno	170	181
01.01.2016 0:00:55	P+R Rajská zahrada	0	0	17	volno	71	88
01.01.2016 0:01:14	P+R Zličín 1	0	0	36	volno	47	83
01.01.2016 0:01:21	P+R Černý Most 2	0	0	15	volno	116	131
01.01.2016 0:01:26	P+R Rajská zahrada	0	0	17	volno	71	88
01.01.2016 0:01:45	P+R Zličín 1	0	0	36	volno	47	83
01.01.2016 0:01:52	P+R Černý Most 2	0	0	15	volno	116	131
01.01.2016 0:01:57	P+R Rajská zahrada	0	0	17	volno	71	88

Parking locations

Provider: Technická správa komunikací
Source: http://opendata.praha.eu/dataset/parkoviste
Datatype: CSV

This is a short file that connects parking facilities operated by the TSK authority with their geographical locations. We will use the locations of just the park+rides.

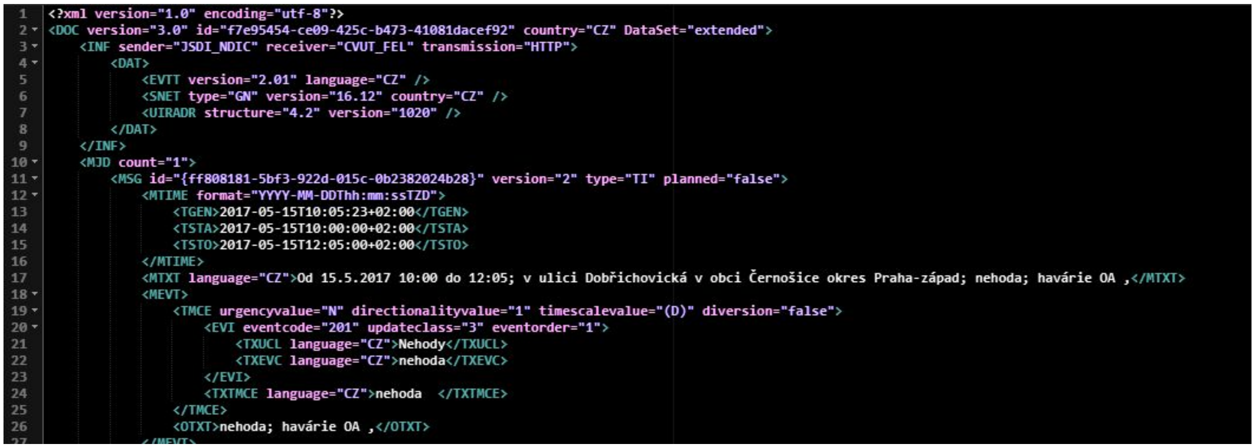


_id	name	lat	lng	pr	totalNu...
1	Běchovice	50.0808	14.597429	True	92
2	Chodov	50.032074	14.492015	True	653
3	Depo Hostivař	50.076397	14.517204	True	169
4	Holešovice	50.109318	14.441252	True	74
5	Kongresové centrum Praha	50.060696	14.428616	True	260
6	Kotlářka	50.06858	14.358078	True	181
7	Letňany	50.125168	14.514741	True	633
8	Ládví	50.126156	14.472344	True	78
9	Nové Butovice	50.05053	14.350451	True	57

List of traffic events

Provider: Ředitelství silnic a dálnic
Source: http://kbss.felk.cvut.cz/dopravni-info.zip
Datatype: XML

Details all traffic events for the entire Czech Republic, including type, location, and date. Only accidents and traffic jams in and around Prague will be of our interest, so we will focus solely on those.



```
<?xml version="1.0" encoding="utf-8"?>
<DOC version="3.0" id="f7e95454-ce09-425c-b473-41081dacef92" country="CZ" DataSet="extended">
    <INF sender="JSDI_NDIC" receiver="CVUT_FEL" transmission="HTTP">
        <DAT>
            <EVTT version="2.01" language="CZ" />
            <SNET type="GN" version="16.12" country="CZ" />
            <UIRADR structure="4.2" version="1020" />
        </DAT>
    </INF>
    <MJD count="1">
        <MSG id="{ff808181-5bf3-922d-015c-0b2382024b28}" version="2" type="TI" planned="false">
            <MTIME format="YYYY-MM-DDThh:mm:ssTZD">
                <TGEN>2017-05-15T10:05:23+02:00</TGEN>
                <TSTA>2017-05-15T10:00:00+02:00</TSTA>
                <TSTO>2017-05-15T12:05:00+02:00</TSTO>
            </MTIME>
            <MTXT language="CZ">Od 15.5.2017 10:00 do 12:05; v ulici Dobřichovická v obci Černošice okres Praha-západ; nehoda; havárie OA ,</MTXT>
            <MEVT>
                <TMCE urgencyvalue="N" directionalityvalue="1" timescalevalue="(D)" diversion="false">
                    <EVI eventcode="201" updateclass="3" eventorder="1">
                        <TXUCL language="CZ">Nehody</TXUCL>
                        <TXEVC language="CZ">nehoda</TXEVC>
                    </EVI>
                    <TXTMCE language="CZ">nehoda  </TXTMCE>
                </TMCE>
                <OTXT>nehoda; havárie OA ,</OTXT>
```

Data integration

All of the pipeline is coded in Python. I could have used GraphDB's OntoRefine tool for some data sources, but ultimately decided not to since I am quite familiar with working with RDF graphs in code already and can therefore control the output in ways that are important for this project.

I have used the RDFlib Python package for this project[1]. This package implements RDF graphs with lots of features, the most important of which is adding data into a graph and then serializing it into Turtle format. This bypasses the necessity to make custom serialization algorithms. The description of the scripts themselves are described in comments within the code.

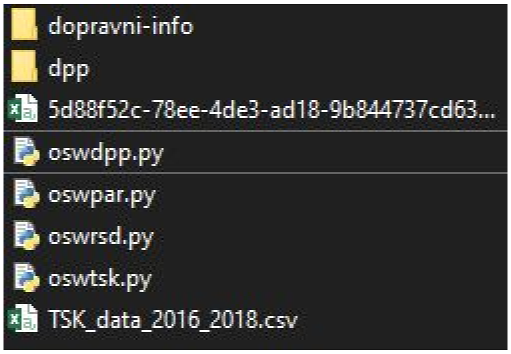
Pipeline instructions

To start, you must have Python installed. (The code has been developed on version 3.6.5; I can confirm that it won't work on version 2.7.4.) To install RDFlib, run "pip install rdflib".

The scripts require that the files are present in the same place as the scripts themselves. To be more specific:

- Public transport coupon sales should be in a subfolder called "dpp". Make sure to download all files from the source.
- P+R occupancy data and Parking locations should be in the same folder as the scripts.
- Traffic events data should be in a subfolder called "dopravni-info".

If you did everything correctly, your working folder should look like this:

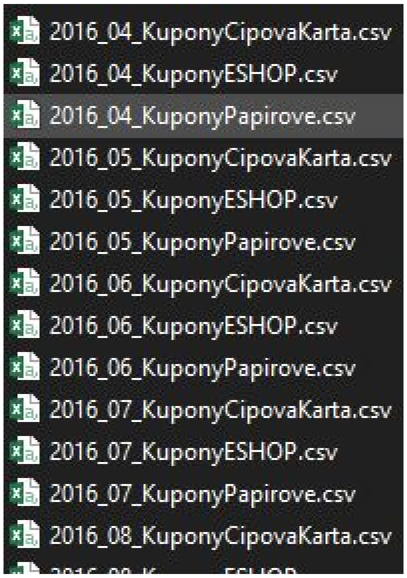


```
dopravni-info
dpp
5d88f52c-78ee-4de3-ad18-9b844737cd63...
oswdpp.py
oswpar.py
oswrsd.py
oswtsk.py
TSK_data_2016_2018.csv
```

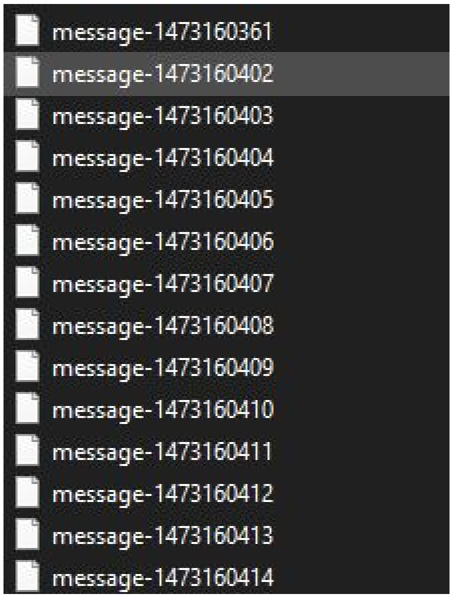
---

[1] https://rdflib.readthedocs.io/en/stable/index.html

Here’s what the /dpp/ and /dopravni-info/ folders should look like, respectively:



2016_04_KuponyCipovaKarta.csv
2016_04_KuponyESHOP.csv
2016_04_KuponyPapirove.csv
2016_05_KuponyCipovaKarta.csv
2016_05_KuponyESHOP.csv
2016_05_KuponyPapirove.csv
2016_06_KuponyCipovaKarta.csv
2016_06_KuponyESHOP.csv
2016_06_KuponyPapirove.csv
2016_07_KuponyCipovaKarta.csv
2016_07_KuponyESHOP.csv
2016_07_KuponyPapirove.csv
2016_08_KuponyCipovaKarta.csv



message-1473160361
message-1473160402
message-1473160403
message-1473160404
message-1473160405
message-1473160406
message-1473160407
message-1473160408
message-1473160409
message-1473160410
message-1473160411
message-1473160412
message-1473160413
message-1473160414

To run the scripts, simply run “python *script*.py”, where *script* is the filename of the script. The scripts can take a while to complete (as the files, especially the occupancy data, are very large), but should not take more than 10-15 minutes on an ordinary PC (with an SSD and a i7-4790k CPU, the process is about five minutes long and does not take more than 100MB of system memory). The code is single threaded, so it should not lock up a modern PC.

After running, the code generates TTL files in the same folder, so make sure that file permissions are in order.

Public transport coupon sales

- Script: oswdpp.py
- Output: dpp.ttl

The sales data is separated by year/month and by the type of sale (of electronic coupon, paper coupon or eshop coupon). The eshop data understandably does not record the place of sale. Therefore, before the data can even be converted, the files have to be connected together. Afterwards, the data has to be trimmed down, as a file containing all the details of the original would be too large (over 20MB). To do so, the type of coupon (for example Student or Citizen) as well as the validity of the coupon (such as a year or six months) is omitted during the conversion.

Schema: PassSalesRecord

Subject IRI: http://bindetad.com/dppsales/%year%/%month%/%boughtAt%, where %boughtAt%, %year%, %month% is the same as boughtAt, year and month in the schema respectively.

Property	**Datatype**	**Description**
boughtAt	String	At which location the pass was bought. Contains “Eshop” if the pass was bought online.
count	Integer	How many passes were bought during the month.
month	Integer	The month of the record.
year	Integer	The year of the record.
passType	String	The type of the pass bought; can be either Paper or Electronic.

Park+ride occupancy data

- Script: oswtsk.py
- Output: tsk.ttl

As the file contains detailed records of changes of occupancy about every 30 minutes for each P+R, a graph containing all the data would be far too large. To mitigate this, the records are reduced to entries of each day for every P+R, and the occupancy data is computed as an average for each day.

Schema: ParkAndRideAttendanceRecord

Subject IRI: http://bindetad.com/pprattendace/%parkandride%/%date%, where %parkandride% is the name of the P+R and %date% is the date of the record.

Property	**Datatype**	**Description**
capacity	String	How many spots does the P+R have.
date	xsd:date	The day of the record.
enter	Integer	How many cars entered during the day.
exit	Integer	How many cars left during the day.
freespace	Integer	How many spaces were available on average during the day.
occupancy	Integer	How many spaces were occupied on average during the day.
parking	String	The name of the P+R.
status	Float	A percentage representation of how often the P+R was full. For example, 0.12 means that the P+R was full approximately 12% of the time during the day.

Parking locations

- Script: oswpar.py
- Output: par.ttl

The pipeline for this is relatively straightforward. The CSV file contains parking lots that are not designated as P+Rs. After these records are skipped, the file is processed normally.

Schema: ParkAndRide

Subject IRI: http://bindetad.com/ParkAndRide/%parkandride% where %parkandride% is the name of the P+R.

Property	**Datatype**	**Description**
rdfs:label	String	Name of the P+R.
capacity	Integer	The capacity of the P+R.
geo:lat	String	WGS84 latitude of the P+R.
geo:long	String	WGS84 longitude of the P+R.

List of traffic events

- Script: oswrsd.py
- Output: rsd.ttl

This is the only dataset that is in XML and has a proprietary standard as defined by Národní dopravní informační centrum under the Jednotný systém dopravních informací initiative. Fortunately, the standard is clearly described in a document available online[2], so parsing the data is only a matter of understanding the aforementioned document. Any events that are not under region code 19 or 27 (Prague or Středočeský kraj) are ignored.

Schema: TrafficEvent

Subject IRI: http://bindetad.com/trafficevents/%id%, where %id% is the ID of the traffic event (as described by <MSG id=... attribute).

Property	**Datatype**	**Description**	**Element in XML**
description	String	Description of the event.	MTXT
eventCode	Integer	Event code(s) as described by the ALERT-C standard.[3]	EVI
geo:lat	String	WGS84 latitude of the event.	COORD
geo:long	String	WGS84 longitude of the event.	COORD

---

[2] http://portal.dopravniinfo.cz/public/files/userfiles/Rozhrani_DDR_v3.2.6.pdf

[3] https://wiki.openstreetmap.org/wiki/TMC/Event_Code_List

regionCode	Integer	Code of the region where the event occured.	DEST
eventDescription	String	Description of the event code(s) as described by the ALERT-C standard.	TXTMCE
timeIssued	xsd:datetime	Datetime of the issue of the event.	TGEN
timeFrom	xsd:datetime	Datetime of the event start.	TSTA
timeTo	xsd:datetime	Datetime of the event end.	TSTO

Ontology design

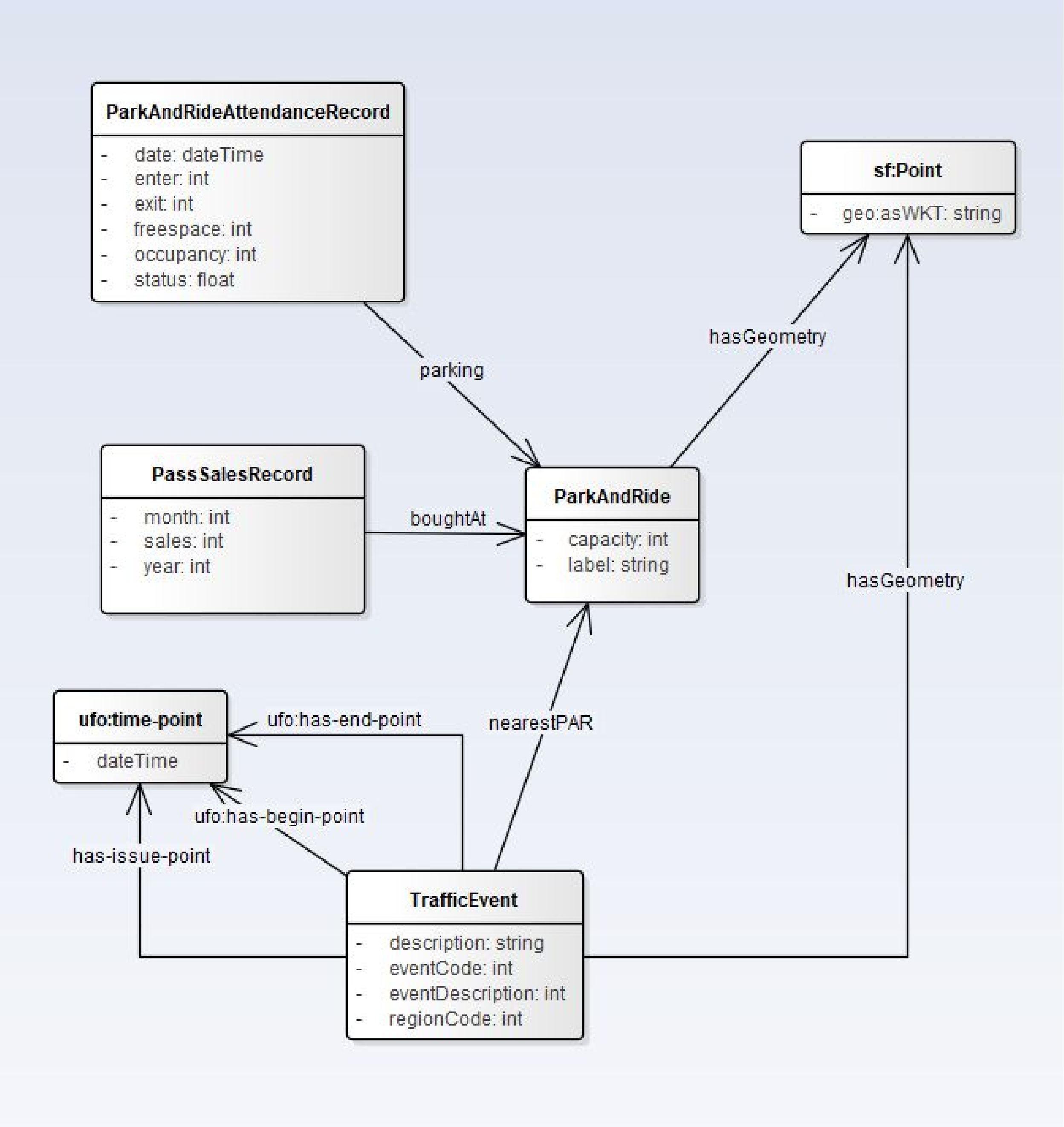
In order to make the most out of integrating the datasets, I have made the following changes to the datasets from CP1:

- ParkAndRideAttendanceRecord as well as PassSalesRecord have been transformed into RDF Data cubes.
- Instead of datetime annotations, traffic events are instead connected to Time Points to better conform to the UFO ontology.
- Instead of geo:lat and geo:long, all geographical data is represented with geometries and WKT Literals to make GeoSPARQL querying available.

The datasets were connected together in these ways:

- PassSalesRecord’s boughtAt’s range is no longer a literal, but instead a ParkAndRide.
- ParkAndRideAttendanceRecord’s parking’s range is no longer a literal, but instead a ParkAndRide.
- TrafficEvents have been modified with a new nearestPAR property, whose range is a ParkAndRide.

The relations between the various classes are represented in the following diagram:



Lessons learned

While the development of this project has certainly been confirming the fact that ontologies and RDF graphs are very powerful tools that allow interpretation of realities published by unrelated organizations in such a way which can be connected together, I am not sure whether I have provided the best example of it - not in the sense that anything is outright wrongly implemented, per se (at least I hope) but rather because fundamentally, the datasets were not ideal for the demonstration of the features of ontologies. Nothing here would require much thought to convert it to a traditional relational database. As such, if I were to do the project again, I would make sure to choose datasets that are more interconnected or from which more meaningful inferences can be made. Perhaps something that can take advantage of ontologies' inferences more thoroughly. For instance, what if you were to plan an ontology of medical conditions? You could infer diagnoses from certain symptoms (which could be automatically categorized by length, color, intensity or location on the body), and from that you could suggest treatment. With that, you could monitor patients' conditions and aid doctors' judgements with this ontology. This is undoubtedly outside the subject's scope; my point is simply that the project could have taken on a more of a "spirit of the *semantic* web" form.

It is also probable that my datasets were capable of much more and I have not been able to see it. For example, if I added a dataset containing all points of sale of coupons, I could have compared the P+R's sales to other coupon sale locations. Of course, all of these ideas come to you when it is too late to change, which is why early and thorough planning is very important.