Building Virtual Knowledge Graphs from sensor data using Ontop and the PostgreSQL ecosystem

Benjamin Cogrel, CTO and co-founder of Ontopic
Core developer of Ontop

Knowledge Graph Forum, Zurich, 29 September 2022

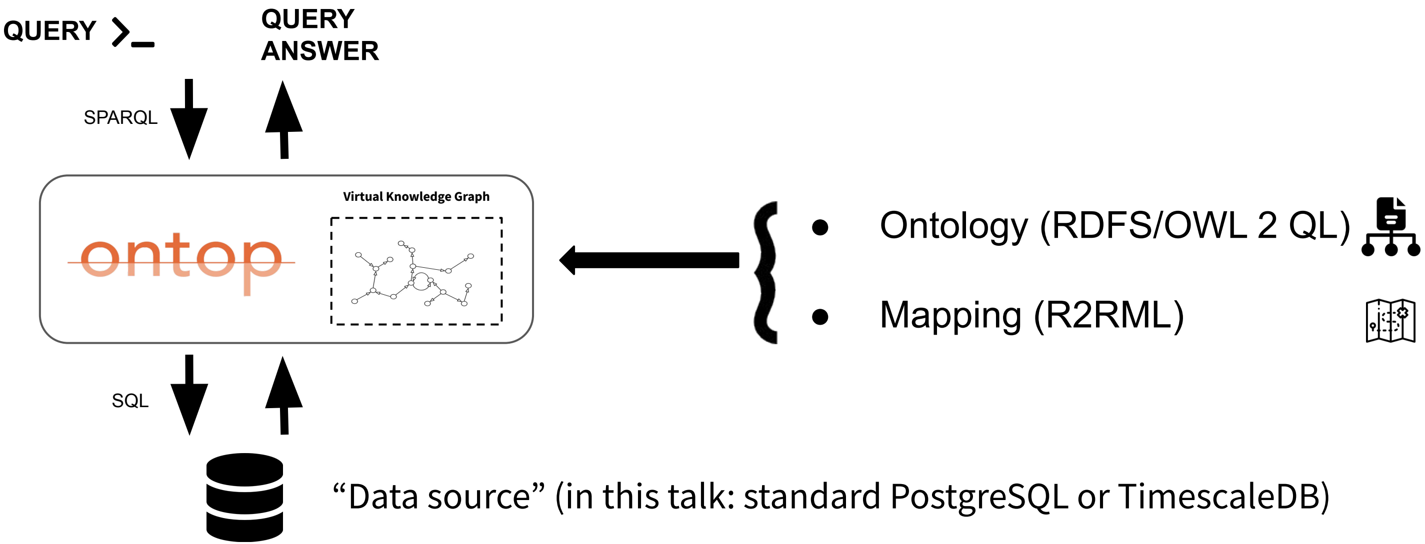
Our experience with sensor data

1. From the South Tyrolean Open Data Hub
   - Weather (air/water temperature, wind speed, pollution, etc.)
   - Traffic (highway, streets)
   - Parking lot occupation
   - Car/bike sharing availability
   - E-charging stations
   - etc.
2. From the Swiss Federal Office for the Environment (BAFU)
   - Hydrological data
   - Proof of Concept done with Zazuko

All the technologies mentioned in this talk are open-source

Virtual Knowledge Graphs (VKGs) with Ontop

No RDF materialization: data stays in the "data source"



What makes (V)KGs attractive for sensor data?

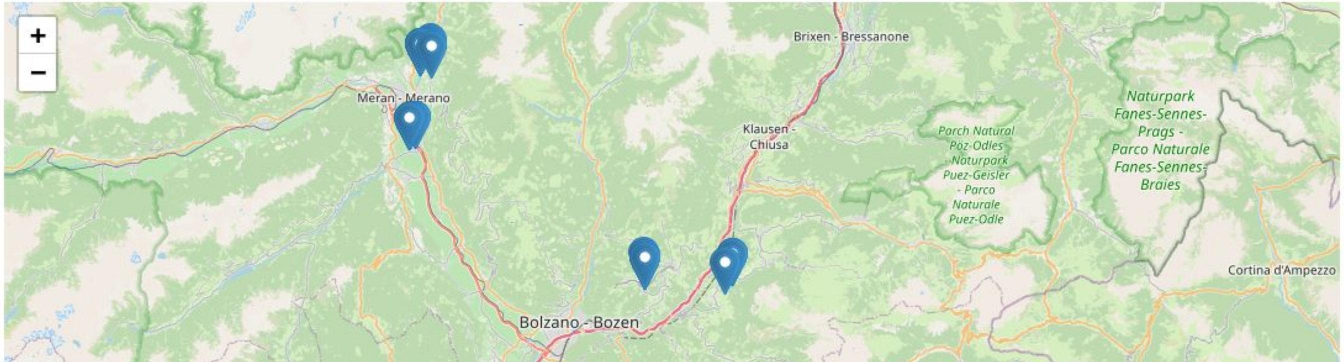
- Knowledge Graphs (in general)
  - Richer representation of the observation
  - Context made explicit
- Virtual KGs
  - Don’t have to pay the price for having an expanded RDF representation
  - Keep the storage efficient (critical for sensor data)
  - Can work with time-series databases

Open Data Hub South Tyrol

Nearby Restaurants and Weather Stations | Nearby Restaurants and Weather Stations, and observations | Restaurants nearby water above 18°C

```
SELECT ?hPos ?hPosLabel ?pPos ?pPosLabel ?v ?t
WHERE {
  # Restaurants
  ?h a schema:Restaurant ; geo:defaultGeometry/geo:asWKT ?hPos ; schema:name ?hPosLabel .
  FILTER(LANG(?hPosLabel)='it')
  BIND('red' AS ?hPosColor)
  # Platforms
  ?p a sosa:Platform ; schema:name ?pPosLabel .
  BIND('blue' AS ?sPosColor)
  # Observations
  ?p sosa:hosts/sosa:madeObservation ?o .
  ?o a :LatestObservation .
  ?o sosa:hasResult ?r ; sosa:resultTime ?t ; sosa:hasFeatureOfInterest ?f ; sosa:observedProperty [ a :Temperature ] .
  ?r qudt:numericValue ?v ; qudt:unit qudt-unit:DegreeCelsius .
```

Table | Response | Pivot Table | Google Chart | Geo | </>



Challenges - Open Data Hub South Tyrol

1. Data synchronization with large volumes and frequent inserts

2. Modelling: mapping to a user-friendly Knowledge Graph

Data synchronization - Setting

A few numbers:

- 500 GB of historical data
- 40k observations at different frequencies (from 1 minute to 3 days periods)

PostgreSQL instance dedicated to the VKG

- Merging data from 2 production PG clusters (one for sensor data, one for tourism data)
- No unpredictable load on the production databases
- Can run geospatial queries joining the 2 datasets

Data synchronization - Solutions

First solution: using the PostgreSQL logical replication

- Failed every two months
- Too much pressure on the WAL (Write-Ahead-Log)
- No reliable fix found, even with the help of a PG expert

Failback solution: custom synchronization script

- Using timestamps to incrementally update the historical tables
- Full copy of the other tables (small enough)
- Every 15 min -> latency

Modelling - Source relational schema

- 4 main tables related to sensor data
    - About latest and historical measurements, measurement types and stations
    - With integrity constraints (unique and foreign keys)

- Factorization of the data diversity
    - Good: low number of tables
    - Insufficient: **no reuse** of measurement types among providers
        - Many similar types with different ids and sometimes different units

Modelling - Mapping

Target ontology: Semantic Sensor Network (https://www.w3.org/TR/vocab-ssn/)

- Similar notions:
  - measurement -> *sosa:Observation*
  - type of measurement -> *sosa:ObservableProperty*
  - station -> *sosa:Platform*
- Not directly matched notions:
  - *sosa:Sensor*
    - Connects observations to platforms
    - Easy to map (one sensor per station/measurement type pair)
  - *sosa:FeatureOfInterest*
    - Describe the “entity” being observed
    - Modelling effort as it brings new knowledge (and added-value!)

New querying experience

Getting the outdoor temperature in a given area over time

- Before, with the original data model:
    1. Look for a station of type “MeteoStation”
    2. Get its GPS coordinates
    3. Get its corresponding measurements and keep the ones associated to the type ids 8 or 2151 (which are described as “air temperature”)
- Now, with the KG:
    1. Get a feature of interest that is instance of the class *OutdoorAir*
    2. Get its GPS coordinates
    3. Get its observations and keep the ones for which the observable property is an instance of the class *Temperature*

New querying experience

Getting the outdoor temperature in a given area over time

Imagine now we add a dataset about air temperature measured by **buildings**

- With the previous approach:
  - Need to add a new subquery
  - New station type, new measurement type ids
- With the KG:
  - No need to change the query
  - Robust to the introduction of indoor air temperature measurements

Proof of Concept with TimescaleDB

Done with Zazuko

- TimescaleDB
  - Time-series extension of PostgreSQL
  - New storage mechanism (with time-based partitioning)
  - Scalable architecture

- Data provided by the Swiss Federal Office for the Environment
  - 2M hydrological observations, started in 1978, every 10 min
  - We duplicated 50 times to run our experiments over 100M observations

- Using Zazuko's RDF cube model: https://cube.link/

Proof of Concept with TimescaleDB

- Objective:
  - Validating that Ontop + TimescaleDB can process the typical SPARQL queries over Zazuko RDF cubes efficiently

- Results:
  - Good performance: all the queries answered efficiently
    - Aggregation on the full timeline on ad-hoc queries with interactive answer time
  - From Ontop’s perspective, TimescaleDB is plain PostgreSQL
  - Very promising

Take home messages

- There is a good match between VKGs and sensor data
- Sensor data can be challenging for regular databases so time-series databases should be seriously considered
- The combination of Ontop and TimescaleDB is promising

Slides

ONTOPIC