

Semantic Technologies for Data Access and Integration: Part 4. Demo of Integrating Web APIs

Diego Calvanese, **Guohui Xiao**

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy



The 27th ACM International Conference on Information and Knowledge Management (CIKM'18)
Torino, Italy, 26 October 2018

Integrating Web APIs using Ontop

Goal: Integrating the Station and Sensor Web APIs of the South Tyrol Open Data Portal



- Weather Station API:
`http://daten.buergernetz.bz.it/services/meteo/v1/stations`
- Sensor API:
`http://daten.buergernetz.bz.it/services/meteo/v1/sensors`

Weather Stations

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

name: "station"
type: "FeatureCollection"
crs:
  type: "name"
  properties:
    name: "EPSG:25832"
features:
  0:
    type: "Feature"
    geometry:
      type: "Point"
      coordinates:
        0: 669803.015640121
        1: 5123442.04315501
    properties:
      SCODE: "89940PG"
      NAME_D: "ETSCH BEI SALURN"
      NAME_I: "ADIGE A SALORNO"
      NAME_L: "ETSCH BEI SALURN"
      NAME_E: "ETSCH BEI SALURN"
      ALT: 210
      LONG: 11.20262
      LAT: 46.243333
  1:
    type: "Feature"
```

Sensors

▼ 0:

SCODE: "82500WS"
TYPE: "LT"
DESC_D: "Lufttemperatur"
DESC_I: "Temperatura dell'aria"
DESC_L: "Temperatura dl'aria"
UNIT: "°C"
DATE: "2018-10-25T18:40:00CEST"
VALUE: 8.8

▼ 1:

SCODE: "82500WS"

TYPE: "WD"

Federation Engine



(<https://www.denodo.com>) is a platform for

- virtualizing heterogeneous datasources as a relational database, and
- performing federated queries using SQL queries

Denodo Express can be evaluated for free.

NB: Other data integration tools/federation engines (e.g. Spark, Presto, Teiid) might also be used, but we haven't tested yet.

Federation Engine



(<https://www.denodo.com>) is a platform for

- virtualizing heterogeneous datasources as a relational database, and
- performing federated queries using SQL queries

Denodo Express can be evaluated for free.

NB: Other data integration tools/federation engines (e.g. Spark, Presto, Teiid) might also be used, but we haven't tested yet.

Denodo Platform Control Center

Denodo Platform Control Center
192.168.0.94 (192.168.0.94)

Virtual DataPort

Scheduler

ITPilot

SERVERS

Virtual DataPort Server
Port 9999 Running Stop

WEB TOOLS

Data Catalog
<http://localhost:9090/denodo-data-catalog> Stopped Start

Diagnostic & Monitoring Tool
<http://localhost:9090/diagnostic-monitoring-tool> Stopped Start

TOOLS

Virtual DataPort Administration Tool LAUNCH

STOP ALL SERVERS

Virtual DataPort v7.0

Help

Configure

Update

Uninstall

Denodo Virtual DataPort Administration Tool

File Administration Tools Help

Server Explorer Database Management * bzopendata.stations bzopendata.f_sensors

Configuration VQL Save Discard Create base view Export Drop

Conn Edit HTTP Connection

Name: Configuration Pagination Proxy Authentication Filters

HTTP method: GET

URL: http://daten.buergernetz.bz.it/services/meteo/v1/stations

HTTP headers

Check certificates Autodetect encoding

Charset encoding: Adobe-Standard-Encoding

Test connection Ok Cancel

The screenshot shows the Denodo Virtual DataPort Administration Tool interface. On the left, the Server Explorer pane displays a tree structure of databases and tables, including 'admin', 'bzopendata' (with 'sensors' and 'stations' sub-tables), 'f_sensors', 'f_stations', and 'itpilot'. In the center, the Database Management tab is active, showing the 'bzopendata.stations' table. A modal dialog titled 'Edit HTTP Connection' is open, allowing configuration of an HTTP connection. The 'Configuration' tab is selected, showing the following settings:

- HTTP method: GET
- URL: http://daten.buergernetz.bz.it/services/meteo/v1/stations
- HTTP headers:
 - Check certificates
 - Autodetect encoding
- Charset encoding: Adobe-Standard-Encoding

At the bottom of the dialog are 'Test connection', 'Ok', and 'Cancel' buttons.

Denodo Virtual DataPort Administration Tool

File Administration Tools Help

Server Explorer

Search

- admin
- bzopendata
 - sensors
 - sensors
 - jsonarray
 - stations
 - stations
 - f_sensors
 - scode
 - type
 - desc_d
 - desc_i
 - desc_l
 - unit
 - date
 - value
 - f_stations
 - sensors
 - jsonarray
 - stations
 - itpilot

* bzopendata.stations x bzopendata.f_sensors x » 2

Summary Edit Options VQL

Model Where Conditions Group By Output Metadata

View parameters: none.

sensors

scode	text
type	text
desc_d	text
desc_i	text
desc_l	text
unit	text
date	text
value	double

Flatten Preview

scode	text
type	text
desc_d	text
desc_i	text
desc_l	text
unit	text
date	text
value	double

The screenshot shows the Denodo Virtual DataPort Administration Tool interface. On the left, the Server Explorer pane displays a hierarchical list of database objects, including 'bzopendata' and its sub-tables 'sensors' and 'stations', along with their respective sub-tables 'f_sensors' and 'f_stations'. On the right, the main workspace shows a 'bzopendata.f_sensors' table definition. The table has columns: scode (text), type (text), desc_d (text), desc_i (text), desc_l (text), unit (text), date (text), and value (double). A 'Flatten Preview' section on the right shows the same columns with their types. A dashed arrow points from the 'sensors' table in the workspace to the 'Flatten Preview' section, indicating the flattening process.

Denodo Virtual DataPort Administration Tool

File Administration Tools Help

Server Explorer

Search

admin
bzopendata
 sensors
 stations
 f_sensors
 scode
 type
 desc_d
 desc_i
 desc_l
 unit
 date
 value
 f_stations
 name
 type_0
 crs
 type
 geometry
 properties
 scode
 name_d
 name_i
 name_l
 name_e
 long
 lat
 coordinates
 sensors
 jsonarray

* bzopendata.stations x bzopendata.f_sensors x bzopendata.f_stations x

Summary Edit Options VQL

Model Where Conditions Group By Output Metadata

DISTINCT clause: ORDER BY:

View name: f_stations

	PK	Field Name	Field Type	Description
		name	text	
		type_0	text	
		crs	stations_crs	
		type	text	
		geometry	stations_geometry	
		properties	stations_properties	
		scode	text	
		name_d	text	
		name_i	text	
		name_l	text	
		name_e	text	
		alt	double	
		long	double	
		lat	double	
		scode	text	
		name_d	text	
		name_i	text	
		name_l	text	
		name_e	text	
		long	double	
		lat	double	
		coordinates	stations_geometry_coordinates	

unibz

DbVisualizer

DbVisualizer Free 10.0.15 - denodo

The screenshot shows the DbVisualizer interface with a connection named 'denodo' successfully established. The connection details are as follows:

- Connection**: Name: denodo, Notes: (empty)
- Database**: Settings Format: Database URL, Database Type: Generic, Driver (JDBC): Denodo, Database URL: jdbc:vdb://localhost:9999/bzopendata
- Authentication**: Database Userid: admin, Database Password: (redacted)
- Options**: Auto Commit: checked, Save Database Password: Save Between Sessions, Permission Mode: Development

At the bottom, there are buttons for **Reconnect**, **Disconnect**, and **Ping Server**. A **Connection Message** section indicates the Virtual DataPort version (7.0) and the driver used (com.denodo.vdp.idbc.Driver).

DbVisualizer

DbVisualizer Free 10.0.15 - denodo/bzopendata/VIEW/f_stations

Databases Scripts Favorites

Connections denodo

- admin
- bzopendata (Default)
 - TABLE
 - VIEW
 - f_sensors
 - f_stations**
- Procedures
- iptilot
- localopendata

denodo f_stations

Table: f_stations

In DbVisualizer Pro this feature includes data editing functionality and management of binary/BLOB data.

Data Row Count Primary Key Indexes Grants Row Id References

*	name	type_0	crs	type	geometry	properties
1	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 669803.015640121, "y": 5123442.04315501}]} ("89940PG", "E	"08155PG", "E
2	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 626295.144332811, "y": 5164467.60475602}]} ("08155PG", "E	"45200OS", "G
3	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 744723.350800056, "y": 5192575.70046406}]} ("45200OS", "G	"82500WS", "R
4	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 688387.444866793, "y": 5165389.11604176}]} ("45100WS", "C	"164288.600532982, "S
5	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 743964.273827689, "y": 5194147.3777763}]} ("45100WS", "C	"00700WS", "C
6	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 613580.465752357, "y": 5181424.77411351}]} ("00700WS", "C	"00390SF", "G
7	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 614288.600532982, "y": 5184739.67995313}]} ("00390SF", "G	"06040WS", "S
8	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 624958.049830706, "y": 5151214.04624925}]} ("06040WS", "S	"06090SF", "St
9	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 623889.208467509, "y": 5150180.53266798}]} ("06090SF", "St	"35100WS", "J
10	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 676779.811441324, "y": 5190004.87549571}]} ("35100WS", "J	"50400WS", "P
11	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 737957.242620869, "y": 5211689.61840654}]} ("50400WS", "P	"722527.967736351, "A
12	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 737190.866142645, "y": 5213918.23307904}]} ("50360SF", "Pi	"61690SF", "Ai
13	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 722527.967736351, "y": 5160922.17043332}]} ("61690SF", "Ai	"61720WS", "A
14	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 716638.174107094, "y": 5156970.01492438}]} ("61720WS", "A	"59450PG", "A
15	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 659066.440860789, "y": 5170818.30570573}]} ("19850PG", "E	"59450PG", "A
16	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 676827.854603636, "y": 5150310.85783781}]} ("29850PG", "E	"719835.823424396, "S
17	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 685247.143615374, "y": 5195138.09867744}]} ("36750PG", "N	"5184310.62400162, "S
18	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 723868.161612044, "y": 5189015.11477246}]} ("59450PG", "A	"706592.119483754, "S
19	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 719835.823424396, "y": 5184310.62400162}]} ("64550PG", "C	"5188228.51362349, "S
20	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 706592.119483754, "y": 5188228.51362349}]} ("67350PG", "R	"5176076.44850128, "S
21	station	FeatureCollection	{"name": "EPSG:25832"}	Feature	{"Point", [{"x": 703111.183748393, "y": 5176076.44850128}]} ("69790PG", "E	"51M of 512M

Max Rows: 1000 Max Chars: -1 Format: <Select a Cell> 0.221/0.068 sec 119/14 1-22

Protege

The screenshot shows the Protege interface for editing the 'weather' ontology. The top navigation bar includes tabs for Active Ontology, Entities, Individuals by class, DL Query, Ontop Mappings, and Ontop SPARQL. The 'Ontop Mappings' tab is active.

The main workspace displays the 'Datasource editor' tab, which is used for managing database connections. The connection parameters are set as follows:

- Connection URL: `jdbc:vdb://localhost:9999/bzopendata`
- Database Username: `admin`
- Database Password: `*****`
- Driver class: `com.denodo.vdp.jdbc.Driver`

Below the connection parameters, there is a 'Test Connection' button. On the left side of the interface, there are tabs for Class hierarchy, Annotation property hierarchy, Data property hierarchy, Object property hierarchy, and Object property hierarchy (repeated). The 'Object property hierarchy' tab is currently selected. It shows the hierarchy starting from `owl:topObjectProperty`.

Reasoner state out of sync with active ontology Show Inferences

Protege

The screenshot shows the Protege interface with the following details:

- Class hierarchy:** On the left, under the "owl:Thing" node, various RDF classes are listed: foaf:Agent, geo:SpatialObject, rdf>List, skos:Collection, skos:Concept, skos:ConceptScheme, sosa:ActuableProperty, sosa:Actuation, sosa:Actuator, sosa:FeatureOfInterest, sosa:ObservableProperty, sosa:Observation, sosa:Platform, sosa:Procedure, sosa:Result, sosa:Sample, and enca:Sampler.
- Mapping manager:** This pane contains several mapping rules:
 - WeatherStation**:
`:data/station/{scode} a WeatherStation ; rdfs:label {name_i} ; :longitude {long} ; :latitude {lat} ; geo:asWKT "POINT ({long} {lat})"^^geo:wktLiteral .`
`SELECT * FROM f_stations`
 - hasSensor**:
`:data/station/{scode} :hasSensor :data/sensor/{scode}/{type} .`
`SELECT * FROM f_sensors`
 - TemperatureSensor**:
`:data/sensor/{scode}/{type} a TemperatureSensor .`
`SELECT * FROM f_sensors WHERE type = 'LT'`
 - madeObservation**:
`:data/sensor/{scode}/{type} sosa:madeObservation :data/oberservation/{scode}/{type}/{date} .`
`SELECT * FROM f_sensors`
 - observation**:
`:data/oberservation/{scode}/{type}/{date} sosa:hasSimpleResult {value} ; sosa:resultTime {date}^^xsd:date .`
`SELECT * FROM f_sensors`
- Search:** At the bottom of the mapping manager pane, there is a search bar with the placeholder "pred:Sensor".
- Annotations:** At the bottom left, there are buttons for "Annotation property hierarchy", "Data property hierarchy", and "Object property hierarchy".
- Buttons:** Along the bottom edge are buttons for "Asserted", "Unasserted", and "Inferred".
- Status:** At the bottom right, it says "Reasoner state out of sync with active ontology" and has a checkbox for "Show Inferences".

Ontop SPARQL endpoint

The screenshot shows the RDF4J Workbench interface running in a web browser. The title bar reads "RDF4J Workbench - New Repo". The address bar shows the URL "localhost:8080/rdf4j-workben...". The main header features the "rdf4j / workbench" logo. On the left, a sidebar menu includes "RDF4J Server", "Repositories" (selected), "Explore", "Modify", and "System". The "Repositories" section has options for "New repository" and "Delete repository". The "Explore" section lists "Summary", "Namespaces", "Contexts", "Types", "Explore", "Query", "Saved Queries", and "Export". The "Modify" section lists "SPARQL Update", "Add", "Remove", and "Clear". The "System" section lists "Information". The central content area is titled "New Repository". It contains fields for "Type" (set to "Ontop Virtual RDF Store"), "ID" (set to "bz"), "Title" (set to "bz"), "OWL ontology file" (set to "/Users/xiao/Dropbox/talks/ontop-slides/2018-10-22"), "Mapping file" (set to "/Users/xiao/Dropbox/talks/ontop-slides/2018-10-22"), and "Ontop properties file" (set to "/Users/xiao/Dropbox/talks/ontop-slides/2018-10-22"). To the right of the "OWL ontology file" field is the note "Provide the full path to the Ontop Owl file". To the right of the "Mapping file" field is the note "Provide the full path to the Mapping file (e.g. .obda, .ttl)". To the right of the "Ontop properties file" field is the note "Provide the full path to the Ontop properties file (e.g. storing connection credentials)". At the bottom are "Create" and "Cancel" buttons.

Visualization of Query Results

The screenshot shows the YASGUI web interface running on a local host. The browser title bar reads "YASGUI" and the address bar shows "Not Secure | yasgui.org" with the URL "http://localhost:8080/rdf4j-server/repositories/bz".

The main area displays a SPARQL query:

```
1 PREFIX geo: <http://www.opengis.net/ont/geosparql#>
2 PREFIX sosa: <http://www.w3.org/ns/sosa/>
3 PREFIX : <http://example.org/weather#>
4 SELECT *
5 {
6   ?station a :WeatherStation ; geo:asWKT ?wkt; :hasSensor ?sensor ; :longitude ?lon; :latitude ?lat .
7   ?sensor a :TemperatureSensor .
8   ?sensor sosa:madeObservation ?o .
9   ?o sosa:hasSimpleResult ?wktLabel .
10  ?o sosa:resultTime ?t .
11 }
12
13
```

Below the query, there are several tabs: Table (selected), Response, Pivot Table, Google Chart, Geo, and a download icon. A search bar and a "Show 50 entries" button are also present.

The results table has columns: station, wkt, sensor, lon, lat, and o. The first row of data is:

	station	wkt	sensor	lon	lat	o
1	http://example.org/weather#data/station/45200SF	"POINT (12.2096 46.8417)"^^< http://www.opengis.net/ont/geosparql#ktLiteral >	http://example.org/weather#data/sensor/45200SF/LT	12.2096E1 [^] _^ xsd:double	4.68417E1 [^] _^ xsd:double	018-10-26T06%

Visualization of Query Results

Screenshot of the YASGUI interface showing a query result visualization.

The top navigation bar shows multiple tabs, including "daten.buergernetz.bz.it/service", "Yahoo Search - Ricerca n...", "daten.buergernetz.bz.it/service", "gzip - is there any way to...", and "YASGUI". The address bar displays "yasgui.org".

The main area is divided into two sections:

- Query:** A code editor containing an SPARQL query:

```
1 PREFIX geo: <http://www.opengis.net/ont/geosparql#>
2 PREFIX sosa: <http://www.w3.org/ns/sosa/>
3 PREFIX : <http://example.org/weather#>
4 SELECT *
5 {
6   ?station a :WeatherStation ; geo:asWKT ?wkt ; :hasSensor ?sensor ; :longitude ?lon; :latitude ?lat .
7   ?sensor a :TemperatureSensor .
8   ?sensor sosa:madeObservation ?o .
9   ?o sosa:hasSimpleResult ?wktLabel .
10  ?o sosa:resultTime ?t .
11 }
```
- Geo:** A map of the Alpine region (Austria/South Tyrol) showing the locations of weather stations. Blue pins mark the coordinates returned by the query. A callout bubble highlights a pin near Innsbruck with the text "9.4E0".

Below the map, a legend indicates the following icons:

- Table
- Response
- Pivot Table
- Google Chart
- Geo

A small "unibz" logo is located in the bottom right corner.

References I