LinkedGeoData and GeoKnow

Claus Stadler    Jens Lehmann

2013-09-19
Outline

1. Motivation for LinkedGeoData
2. OpenStreetMap
3. LinkedGeoData Architecture
4. Sparqlify and Tag Mappings
5. Access and Statistics
6. Use Cases
7. GeoKnow Project Overview
8. GeoKnow Achievements
Outline

1. Motivation for LinkedGeoData
2. OpenStreetMap
3. LinkedGeoData Architecture
4. Sparqlify and Tag Mappings
5. Access and Statistics
6. Use Cases
7. GeoKnow Project Overview
8. GeoKnow Achievements
Motivation

- Ease **information integration** tasks that require **spatial knowledge**, such as
  - Offerings of bakeries next door
  - Map of distributed branches of a company
  - Historical sights along a bicycle track

- **LOD cloud** contains **data sets** with spatial features
  - e.g. Geonames, DBpedia, US census, EuroStat
  - **But:** they are **restricted to popular or large entities** like countries, famous places etc. or specific regions
    - Therefore **they lack** buildings, roads, mailboxes, etc.
Outline

1 Motivation for LinkedGeoData
2 OpenStreetMap
3 LinkedGeoData Architecture
4 Sparqlify and Tag Mappings
5 Access and Statistics
6 Use Cases
7 GeoKnow Project Overview
8 GeoKnow Achievements
Basic entities are:

- **Nodes** Latitude, Longitude.
- **Ways** Sequence of nodes.
- **Relations** Associations between any number of nodes, ways and relations. Every member in a relation plays a certain **role**.

Each entity may be described with **tags** (≡ key-value pairs)

- A way is **closed** if the ID of the last referenced node equals that of the first one.
- Whether a closed way denotes a linear ring or a polygon (i.e. whether the enclosed area is part of the respective OSM entity) depends on the tags.
Example: Leipzig’s Zoo
Comparison: Leipzig’s Zoo (OpenStreetMap)
Comparison: Leipzig’s Zoo (GoogleMaps)

Zoo, near Leipzig, Germany

Zoo Leipzig
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933805, zoo-leipzig.de
  4.6 ⭐️ 116 reviews
  "One of the best zoos in all of Europe, DEFINITELY!"

Gondwanaland
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933805, zoo-leipzig.de
  Category: Zoo
  3 reviews
  "Die größte Tropenwelt Europas in einem der schönsten und modernsten Zoos ...
  "

Café Zoo-Konzertgarten
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933805, zoo-leipzig.de

Zoologischer Garten Leipzig
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933805, zoo-leipzig.de

Frisöre am Zoo
- Pfaffendörfer Straße 14, 04105 Leipzig, Germany
  +49 341 9801002

Teichcafe im Zoo Leipzig
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 59562866, teichcafe.de
  1 review

Café Zoo-Konzertgarten
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933 ext. 500

Parkhaus Leipziger Zoo
- Parthenstraße, 04105 Leipzig, Germany

Zoo-Restaurant
- Pfaffendörfer Straße 29, 04105 Leipzig, Germany
  +49 341 5933405
Outline

1 Motivation for LinkedGeoData
2 OpenStreetMap
3 LinkedGeoData Architecture
4 Sparqlify and Tag Mappings
5 Access and Statistics
6 Use Cases
7 GeoKnow Project Overview
8 GeoKnow Achievements
LGD Architecture

OpenStreetMap.org
  \[
  \text{Changesets} \quad \text{Full Dumps}
  \]

Osmosis

Postgis

Sparqlify Platform

Sparqlify Server
  \[
  \text{Sparqlify Core}
  \]

Sparql Endpoint

Linked Data via Pubby

Sparql Web-Interface via Snorql

LGD Tag Mappings

LGD View Definitions in Sparqlify-ML

Downloadable Dumps

Virtuoso

Vicbit

SemMap
  \[
  \text{"LGD browser"}
  \]

LGD REST API
Tag Mappings

- Key-value pairs will be assigned to RDF resources
- Each pair \((k, v)\) can be annotated with datatypes, language tags, classes
- Mappings are themselves tables
- Example table:

<table>
<thead>
<tr>
<th>lgd_map_literal</th>
<th>property</th>
<th>lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>property</td>
<td>lang</td>
</tr>
<tr>
<td>name</td>
<td>rdfs:label</td>
<td></td>
</tr>
<tr>
<td>name:en</td>
<td>rdfs:label</td>
<td>en</td>
</tr>
<tr>
<td>alt_label</td>
<td>skos:altLabel</td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>rdfs:comment</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**View Definition**

- RDF mapping of the data from a PostgreSQL database

Create View `lgd_nodes` As

Construct {
  ?n a lgdm:Node .
  ?n geom:geometry ?g .
  ?g ogc:asWKT ?o .
}

With

?n = uri(lgd:node, ?id)
?g = uri(lgd-geom:node, ?id)
?o = typedLiteral(?geom, ogc:wktLiteral)

From

nodes
SPARQL-SQL Rewriter
- Rewrites SPARQL Queries according to the view definition
- Platform module offers SPARQL Endpoint and Linked Data interface

https://github.com/AKSW/Sparqlify
- Offers REST methods for frequent queries
- Based on SPARQL (Virtuoso) endpoint
**Downloads**

- RDF dataset for download
  - Generated using
    \[ \text{Construct} \{ \ ?s \ ?p \ ?o \ \} \]
  - \url{http://downloads.linkedgeodata.org}

- Downloadable Dumps
- Virtuoso
Outline

1. Motivation for LinkedGeoData
2. OpenStreetMap
3. LinkedGeoData Architecture
4. Sparqlify and Tag Mappings
5. Access and Statistics
6. Use Cases
7. GeoKnow Project Overview
8. GeoKnow Achievements
The following slides demonstrate how to map relational data to RDF with the Sparqlification Mapping Language (SML).

Thereby, these prefixes are used:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>IRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
</tr>
<tr>
<td>ogc</td>
<td><a href="http://www.opengis.net/ont/geosparql#">http://www.opengis.net/ont/geosparql#</a></td>
</tr>
<tr>
<td>geom</td>
<td><a href="http://geovocab.org/geometry#">http://geovocab.org/geometry#</a></td>
</tr>
<tr>
<td>lgd</td>
<td><a href="http://linkedgeodata.org/triplify/">http://linkedgeodata.org/triplify/</a></td>
</tr>
<tr>
<td>lgd-geom</td>
<td><a href="http://linkedgeodata.org/geometry/">http://linkedgeodata.org/geometry/</a></td>
</tr>
</tbody>
</table>
Input Table

<table>
<thead>
<tr>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>

How to map tables to RDF?

How to introduce the commonly used distinction in GIS between feature and geometry?

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

... 

lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
Input Table

<table>
<thead>
<tr>
<th>nodes</th>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>geom</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
<td></td>
</tr>
</tbody>
</table>

Create View myNodesView As
Construct {
    ...

With
    ...

From
    ...

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

...

lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
Create View myNodesView As Construct {
    ?n geom:geometry ?g .
    ?g ogc:asWKT ?o
}

With...

From

nodes

<table>
<thead>
<tr>
<th>id</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POINT(0 0)</td>
</tr>
<tr>
<td>2</td>
<td>POINT(1 1)</td>
</tr>
</tbody>
</table>

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
...

lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
Input Table

<table>
<thead>
<tr>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Create View myNodesView As
Construct {
    ?n geom:geometry ?g .
    ?g ogc:asWKT ?o
}
With
    ?n = uri(lgd:node, ?id)
    ?g = uri(lgd-geom:node, ?id)
    ?o = typedLiteral(?geom,
                      ogc:wktLiteral)
From
    nodes

Aimed for RDF Output

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
...
lgd:node1 geom:geometry lgd-geom:node1 .
lgd:node2 geom:geometry lgd-geom:node2 .

lgd-geom:node1 ogc:asWKT "POINT(0 0)"^^ogc:wktLiteral .
lgd-geom:node2 ogc:asWKT "POINT(1 1)"^^ogc:wktLiteral .
A more complex example, which demonstrates the use of an SQL mapping table and an SQL helper view.
### Input Table

<table>
<thead>
<tr>
<th>id</th>
<th>k</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>Universitaet Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>name:en</td>
<td>University of Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>amenity</td>
<td>university</td>
</tr>
<tr>
<td>1</td>
<td>addr:street</td>
<td>Augustusplatz</td>
</tr>
<tr>
<td>1</td>
<td>addr:city</td>
<td>Leipzig</td>
</tr>
</tbody>
</table>

### Aimed for RDF Output

```sql
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix lgd: <http://linkedgeodata.org/triplify/> .

lgd:node1 rdfs:label "Universitaet Leipzig" .
lgd:node1 rdfs:label "University of Leipzig"@en .
```
### OSM Table

<table>
<thead>
<tr>
<th>iD</th>
<th>k</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>Universitaet Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>name:en</td>
<td>University of Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>amenity</td>
<td>university</td>
</tr>
<tr>
<td>1</td>
<td>addr:street</td>
<td>Augustusplatz</td>
</tr>
<tr>
<td>1</td>
<td>addr:city</td>
<td>Leipzig</td>
</tr>
</tbody>
</table>
### OSM Table

<table>
<thead>
<tr>
<th>id</th>
<th>k</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>Universitaet Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>name:en</td>
<td>University of Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>amenity</td>
<td>university</td>
</tr>
<tr>
<td>1</td>
<td>addr:street</td>
<td>Augustusplatz</td>
</tr>
<tr>
<td>1</td>
<td>addr:city</td>
<td>Leipzig</td>
</tr>
</tbody>
</table>

### RDF Mapping Table

<table>
<thead>
<tr>
<th>k</th>
<th>property</th>
<th>lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>rdfs:label</td>
<td></td>
</tr>
<tr>
<td>name:en</td>
<td>rdfs:label</td>
<td>en</td>
</tr>
<tr>
<td>alt_label</td>
<td>skos:altLabel</td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>rdfs:comment</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
OSM Table

<table>
<thead>
<tr>
<th>node_tags</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>k</td>
<td>v</td>
</tr>
<tr>
<td>1</td>
<td>name</td>
<td>Universitaet Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>name:en</td>
<td>University of Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>amenity</td>
<td>university</td>
</tr>
<tr>
<td>1</td>
<td>addr:street</td>
<td>Augustusplatz</td>
</tr>
<tr>
<td>1</td>
<td>addr:city</td>
<td>Leipzig</td>
</tr>
</tbody>
</table>

RDF Mapping Table

<table>
<thead>
<tr>
<th>lgd_map_literal</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>property</td>
<td>lang</td>
</tr>
<tr>
<td>name</td>
<td>rdfs:label</td>
<td></td>
</tr>
<tr>
<td>name:en</td>
<td>rdfs:label</td>
<td>en</td>
</tr>
<tr>
<td>alt_label</td>
<td>skos:altLabel</td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>rdfs:comment</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Helper View

<table>
<thead>
<tr>
<th>lgd_node_tags_literal</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>property</td>
<td>v</td>
</tr>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>Universitaet Leipzig</td>
</tr>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>University of Leipzig</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

SELECT id, property, v, lang FROM node_tags, lgd_map_literal
WHERE node_tags.k = lgd_map_literal.k
### Logical Table

<table>
<thead>
<tr>
<th>id</th>
<th>property</th>
<th>v</th>
<th>lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>Univ. L.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>Univ. of L.</td>
<td>en</td>
</tr>
</tbody>
</table>

### SML View

Create View `lgd_node_tags_text` As
Construct {

...
Create View `lgd_node_tags_text` As
Construct {
}

With

...
Create View lgd_node_tags_text As
  Construct {
  }
With
  ?s = uri(lgd:node, ?id)
  ?p = uri(?property)
  ?o = plainLiteral(?v, ?lang)
From
  lgd_node_tags_literal
Logical Table

<table>
<thead>
<tr>
<th>id</th>
<th>property</th>
<th>v</th>
<th>lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>Univ. L.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>rdfs:label</td>
<td>Univ. of L.</td>
<td>en</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

SML View

Create View lgd_node_tags_text As
Construct {
}
With
  ?s = uri(lgd:node, ?id)
  ?p = uri(?property)
  ?o = plainLiteral(?v, ?lang)
From
  lgd_node_tags_literal

Resulting RDF

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix lgd: <http://linkedgeodata.org/triplify/> .

lgd:node1 rdfs:label "Universitaet Leipzig" .
lgd:node1 rdfs:label "University of Leipzig"@en .
### Further Tag Mappings

**decide datatype based on key:**

<table>
<thead>
<tr>
<th>lgd_map_datatype</th>
<th>k</th>
<th>datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>seats</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>unisex</td>
<td>boolean</td>
</tr>
</tbody>
</table>

**decide property: based on key:**

<table>
<thead>
<tr>
<th>lgd_map_property</th>
<th>k</th>
<th>property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>website</td>
<td>foaf:homepage</td>
</tr>
</tbody>
</table>

**decide predicate and object based on key (e.g. class assertion):**

<table>
<thead>
<tr>
<th>lgd_map_resource_k</th>
<th>k</th>
<th>property</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>highway</td>
<td>rdf:type</td>
<td>lgdo:HighwayThing</td>
</tr>
</tbody>
</table>

**decide predicate and object based on key and value:**

<table>
<thead>
<tr>
<th>lgd_map_resource_kv</th>
<th>k</th>
<th>v</th>
<th>property</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>waterway</td>
<td>river</td>
<td>rdf:type</td>
<td>lgdo:River</td>
</tr>
</tbody>
</table>
**LGD Edit Tool**

- Automatic heuristics for mappings
- Multi user tag mapping web application for manual refinements

---

**List of all K-Mappings**

<table>
<thead>
<tr>
<th>k</th>
<th>property</th>
<th>object</th>
<th>affected Entities</th>
<th>edit</th>
<th>delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>railway</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td>ldg:Railway</td>
<td>1</td>
<td>Hide</td>
<td>Delete</td>
</tr>
<tr>
<td>shop</td>
<td>w3#type</td>
<td><a href="http://linkedgeodata.org/ontology/Railway">http://linkedgeodata.org/ontology/Railway</a></td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>emergency</td>
<td>w3#type</td>
<td>LGD:EmergencyThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>route</td>
<td>w3#type</td>
<td>LGD:Route</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>barrier</td>
<td>w3#type</td>
<td>LGD:BarrierThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>boundary</td>
<td>w3#type</td>
<td>LGD:Boundary</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>waterway</td>
<td>w3#type</td>
<td>LGD:WaterwayThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>highway</td>
<td>w3#type</td>
<td>LGD:HighwayThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>power</td>
<td>w3#type</td>
<td>LGD:PowerThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>leisure</td>
<td>w3#type</td>
<td>LGD:Leisure</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>military</td>
<td>w3#type</td>
<td>LGD:MilitaryThing</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>building</td>
<td>w3#type</td>
<td>LGD:Building</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
<tr>
<td>landuse</td>
<td>w3#type</td>
<td>LGD:Landuse</td>
<td>1</td>
<td>Edit</td>
<td>Delete</td>
</tr>
</tbody>
</table>
Ontology

- Lightweight: taxonomy + domain/range axioms
- 1200+ classes
- Enriched classes and properties with multilingual labels from TranslateWiki
  - http://translatewiki.net
- Imported icons for 90 classes from the freely available icon collection from the SJJB Management
  - http://www.sjjb.co.uk/mapicons/
Outline

1 Motivation for LinkedGeoData
2 OpenStreetMap
3 LinkedGeoData Architecture
4 Sparqlify and Tag Mappings
5 Access and Statistics
6 Use Cases
7 GeoKnow Project Overview
8 GeoKnow Achievements
Resources

- **Sparqlify**
  - [http://sparqlify.org](http://sparqlify.org)

- **LinkedGeoData**
  - [http://linkedgeodata.org](http://linkedgeodata.org)

- **Tag Mappings**

- **SML View Definitions**
Complete OSM planet file corresponds to \( \sim 20.000.000.000 \) triples

- Virtual access via Sparqlify
- Full download: ca. 120 GB bz2 compressed (ca. 2.5 TB uncompressed)
- 2.7 MB ontology

Downloads for important fragments available (no roads etc.):

- 292.780.188 Triples
  - 153.613.243 triples of Nodes
  - 139.166.945 triples of Ways
  - Relations not yet available for download

Among them

- 532.812 PlaceOfWorship
- 82.788 RailwayStation
- 72.091 Toilets
- 71.613 Town
- 19.937 City
Access

- **Materialized Sparql Endpoint** (based on Virtuoso DB, download datasets loaded)
  - http://linkedgeodata.org/sparql
  - http://linkedgeodata.org/snorql

- **Virtual Sparql Endpoint** (based on Sparqlify, access to 20B triples, limited SPARQL 1.0 support)
  - http://linkedgeodata.org/vsparql
  - http://linkedgeodata.org/vsnorql

- **Rest Interface** (based on the Virtual Sparql Endpoint)
  - Supports limited queries (e.g. circular/rectangular area, filtering by labels)

- **Downloads**
  - http://downloads.linkedgeodata.org
  - Monthly updates on the above datasets envisioned
Outline

1. Motivation for LinkedGeoData
2. OpenStreetMap
3. LinkedGeoData Architecture
4. Sparqlify and Tag Mappings
5. Access and Statistics
6. Use Cases
7. GeoKnow Project Overview
8. GeoKnow Achievements
Use Cases Generic Browsing

Stadler, Lehmann (Univ. Leipzig)  
LinkedGeoData and GeoKnow  
2013-09-19
Use Cases Generic Browsing
### App Generation

#### ExhibitYourVicinity

**Generator: Settings**

- **What**
  - **Interests:**
    - Pub, fastFood, Bakery
    
  
  (description: a comma-separated list of class names from the [http://linkedgodata.org/ontology](http://linkedgodata.org/ontology)
  
  i.e. Pub, fastFood, Bakery

- **Where**
  - **Latitude:** 51.310228
  - **Longitude:** 12.372966
  - **Max range:** 1.5
    
  
  i.e. 51.310228 i.e. 12.372966 i.e. 1.5

- **Options**
  - **Google Maps Key:**
  - **Language:** en
    
  description: get your google maps key from [http://code.google.com/p/gps/mapp/signup.html](http://code.google.com/p/gps/mapp/signup.html) to generate the source code from the previewed exhibit map view

**Generate**

#### Generator: Preview

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bakery</td>
<td></td>
</tr>
<tr>
<td>Fast Food</td>
<td></td>
</tr>
<tr>
<td>Pub</td>
<td></td>
</tr>
</tbody>
</table>

13 Items filtered from 46 originally (Reset All Filters)

[Map Image]
Event Management

![Map interface for Event Management](image-url)
Outline

1 Motivation for LinkedGeoData
2 OpenStreetMap
3 LinkedGeoData Architecture
4 Sparqlify and Tag Mappings
5 Access and Statistics
6 Use Cases
7 GeoKnow Project Overview
8 GeoKnow Achievements
Project Overview

GeoKnow Generator Interface & API

Use Case Specific Applications
access components via API

LinkedGeoData and GeoKnow

Technology Stack

Quality Assurance Metrics
Fusing
Faceted Browsing
Interlinking
Spatial Widgets
Aggregation
Spatial Linked Data Processing
Adaptive Authoring
Spatial Authoring/Browsing Tools

Open Data/LOD Cloud

E-Commerce Data
Supply Chain Management Data
3rd Party Use Case Data

WP1: Provide interfaces and APIs
WP2: RDF Views, SPARQL-\rightarrow SQL rewriter, Query Optimization
WP3: Linked Data
WP4: GeoSPARQL queries
WP5: (Geo)SPARQL queries
WP6: E-Commerce Data

Triples Stores
Spatial Data Clustering
GIS Databases

use
maintain data

export/wrap/convert to RDF
publish/sync/load/stream
Motivation for LinkedGeoData

OpenStreetMap

LinkedGeoData Architecture

Sparqlify and Tag Mappings

Access and Statistics

Use Cases

GeoKnow Project Overview

GeoKnow Achievements
Many SW tools support one or more life-cycle stages

**Linked Data Stack** ([http://stack.linkeddata.org](http://stack.linkeddata.org)) provides a consolidated repository of such tools

Each tool is a **Debian** package

Lightweight integration between tools via common vocabularies and SPARQL

Demonstrator interfaces for showing tools in combination

Developed by LOD2 and GeoKnow

---

*Stadler, Lehmann (Univ. Leipzig) | LinkedGeoData and GeoKnow*
Intensive work on benchmarking geospatial systems

“Slippy map benchmark” for standard web map applications and other benchmarks to follow

State of the Art Analysis for Triple Stores performed (Virtuoso, uSeekM, Parliament, AllegroGraph, OWLIM-SE, Strabon + Oracle Spatial 11g, PostGIS as reference) using fragments of OSM and Ordnance Survey data

High deviation in performance and supported functionality between geospatial triple stores

GeoSPARQL compliance not achieved yet

More specific results to be published throughout the project ...
**Achievements - Sparqlify**

- SPARQL-SQL rewriter
- Rewrites a SPARQL query to a **single** SQL query allowing the underlying database to perform optimizations
- Improved extensibility of Sparqlify’s model of the underlying database: Custom SQL functions can now be programatically declared.
- Web interface with syntax highlighting and live data generation for easy mapping creation
- Many bug fixes, new optimizations (SQL, LEFT JOIN, ORDER BY)
Achievements - LinkedGeoData

- LinkedGeoData conversion simplified to a set of SQL files and Sparqlify Mapping Definitions
- Monthly dumps now working and automatically scheduled
- LinkedGeoData and Sparqlify are available as Debian Packages!¹ ²

¹ https://github.com/AKSW/Sparqlify
² https://github.com/GeoKnow/LinkedGeoData
Achievements - TripleGeo

- Converts shapefiles/spatial DBMS output to RDF
Achievements - Linking

- Orthodromic distance only supported by LIMES and SILK
- Linking algorithm ORCHID for geospatial data
- Reduces number of comparisons
- For DBpedia (threshold in kilometres):

![Graph showing runtime (ms) for ORCHID and SILK with thresholds 0.1, 0.2, 0.5, and 1 kilometre.]

- Linking papers at ESWC’13 (linking + cloud, best paper award) and ISWC’13 (ORCHID algorithm)
Achievements - Facete

- Generic faceted browser for HTTP SPARQL endpoints
- Nested facets and client side pagination without pre-processing
- Tested with up 10K spatial objects → Goal is to work with DBpedia/LGD
Claus Stadler
cstadler@ informatik.uni-leipzig.de
AKSW/Uni Leipzig

Jens Lehmann
lehmann@informatik.uni-leipzig.de
AKSW/Uni Leipzig

http://geoknow.eu