Transforming geospatial data into RDF
Transforming to RDF

- **Direct mapping** approach became a W3C recommendation in 2012
  - tables becomes classes
  - tables’ attributes are mapped to RDF properties that represent the relation between subject and object
  - Identifiers becomes the subjects

Mapping Languages

R2RML

- A language for expressing customized mappings from relational databases to RDF graphs
- Became W3C recommendation in 2012.
- Express transformation of existing relational data into the RDF data model.

RML

- The RDF Mapping language (RML) is a generic mapping language.
- It can express rules that map data with heterogeneous structures to RDF graphs.
- RML is defined as a superset of R2RML and allows the expression of rules that map relational and semi-structured data (e.g., XML, JSON) into RDF graphs.
RML Example

@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix qt: <http://semweb.mmlab.be/ns/qt#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>.
@base <http://example.com/ns#>.

<#AirportMapping>
  rml:logicalSource [
    rml:source "Airport.csv" ;
    rml:referenceFormulation qt:CSV
  ];
  rr:subjectMap [
    rr:template "http://airport.example.com/{id}";
    rr:class transit:Stop
  ];
  rr:predicateObjectMap [
    rr:predicate transit:route;
    rr:objectMap [
      rml:reference "stop";
      rr:datatype xsd:int
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate wgs84_pos:lat;
    rr:objectMap [
      rml:reference "latitude"
    ]
  ];
  rr:predicateObjectMap [
    rr:predicate wgs84_pos:long;
    rr:objectMap [
      rml:reference "longitude"
    ]
  ].

Input Data

id,stop,latitude,longitude
6523,25,50.901389,4.484444

Output Triples

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix transit: <http://vocab.org/transit/terms/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>.

<http://airport.example.com/6523> wgs84_pos:lat "50.901389".
<http://airport.example.com/6523> wgs84_pos:long "4.484444".

https://rml.io/specs/rml/#string-template
RML/R2RML Term Maps

- A **Term Map** is a function that generates an RDF term from a logical reference.
- Term maps are used to generate the subjects, predicates and objects of the RDF triples
  - subject maps
  - predicate maps
  - object maps

- A **Term Map** must be exactly one of the following:
  - a constant-valued term map,
  - a reference-valued term map,
  - a template-valued term map.
RML/R2RML Term Maps

```
rr:.predicateObjectMap [  
  rr:.predicateMap [ rr:constant exp:hasType];  
  rr:.objectMap [    
    rr:datatype xsd:string;    
    rr:constant "Point";  
  ];  
];

?x exp:hasType "Point"^^<http://www.w3.org/2001/XMLSchema#string>

rr:.predicateObjectMap [  
  rr:.predicateMap [ rr:constant ogc:hasName ];  
  rr:.objectMap [    
    rr:datatype xsd:string;    
    rml:reference "name";  
  ];  
]  

?x exp:hasName "George"^^<http://www.w3.org/2001/XMLSchema#string>

rr:subjectMap [  
  rr:template "http://trans.example.com/stop/{@id}" ];

http://trans.example.com/stop/645
```
Tools/Methods

- The Geometry2RDF was one of the first tools that enabled users to transform spatially enabled RDB systems into RDF graphs.
- K. Chentout et al presented how R2RML can be combined with a spatially-enabled relational database in order to transform geospatial data into RDF.
- TripleGeo is a tool for transforming geospatial features from various sources into RDF graphs.
  - Supports complex formats such as OpenStreetMap data and certain INSPIRE.
  - Parallelized executions based on multi-threading and Apache Spark.


Kostas Patroumpas et al. “Exposing Points of Interest as Linked Geospatial Data”. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD2019

TripleGeo Repo: https://github.com/SLIPO-EU/TripleGeo
The tool GeoTriples

- GeoTriples is a semi-automated tool that enables the automatic transformation of geospatial data into RDF graphs
- The transformation process comprises three steps.
  1. GeoTriples generates automatically extended R2RML or RML mappings
  2. As an optional second step, the user may revise these mappings according to her needs
  3. Finally, GeoTriples processes these mappings and produces an RDF graph.
- GeoTriples supports the transformation of Spatially-enabled relational databases, CSV, GeoJSON, ESRI Shapefiles, KML and XML documents

Kostis Kyzirakos et al. “GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings”. In: J. Web Semant. 52-53 (2018)

Repo: https://github.com/LinkedEOData/GeoTriples
We have extended RML/R2RML with a **Transformation-valued** term map, that generates an RDF term by applying SPARQL extension function on one or more term maps.

A transformation-valued term map has exactly one `rrx:function` property and one `rrx:argumentMap` property.

The `rrx:argumentMap` property has as range an `rdf:List` of term maps that define the arguments to be passed to the transformation function.
RML/R2RML Extension

- We can assert topological relations using the topology vocabulary of GeoSPARQL
- A **referencing object map** is a map that allows a predicate–object map to generate as objects the subjects of another triples map.

```xml
rr:objectMap [  
  rr:joinCondition [  
    rrx:function geof:sfOverlaps;  
    rrx:argumentMap (  
      [ rr:column "Geom" ]  
      [ rr:column "Geom" ; rr:triplesMap <MapB>  
    )  
  ]  
];
```
GeoTriples System Architecture

GeoTriples comprises of three main components: the **Mapping Generator**, the **Mapping Processor** and the **stSPARQL/GeoSPARQL Evaluator**.

**Mapping Generator**
The mapping generator is given as input a data source and creates automatically an R2RML/RML mapping document, depending on the type of the input. The user may edit the generated mapping document to make it comply with her requirements.

**Mapping Processor**
The mapping processor receives as input the mapping document. Based on the term maps, the Mapping Processor generates the final RDF graph, which can be manifested in any of the popular RDF syntaxes such as Turtle, RDF/XML, Notation3 or N-Triples.

**stSPARQL/GeoSPARQL Evaluator**
This component evaluates an stSPARQL/GeoSPARQL query over a relational database given an R2RML mapping.
Mappings Generation

- The mappings produced by GeoTriples consists of two logical sources:
  - non-geometric (thematic) data,
  - geospatial information
- The subjects are defined by combining a URI template with a unique identifier
- For each field of the input data source, GeoTriples generates an RDF predicate according to the name of the field and a predicate-object map
- The triples map that handles geospatial information contains a serialization of the geometric information according to the WKT format
- The generated RDF Graph will be compliant with the GeoSPARQL vocabulary
Mappings Generation

```xml
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix ql: <http://semweb.mmlab.be/ns/ql#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rrx: <http://www.w3.org/ns/r2rml-ext#>.
@prefix rrxf: <http://www.w3.org/ns/r2rml-ext/functions/def/>.
@prefix ogc: <http://www.opengis.net/ont/geosparql#>.
@prefix onto: <http://example.di.uoa.gr/ontology#>.

<#shp_example_Geometry>
 rml:logicalSource {
  rml:source "/path/to/shapefile/file.shp";
  rml:referenceFormulation ql:SHP;
  rml:iterator "shp_iterator";
};
 rr:subjectMap [
  rr:template "http://example.di.uoa.gr/Geometry/{GeoTriplesID}";
  rr:class ogc:Geometry;
];
 rr:predicateObjectMap [
  rr:predicateMap [ rr:constant ogc:asWKT ];
  rr:objectMap [
    rr:datatype ogc:wktLiteral;
    rrx:function rrxf:asWKT;
    rrx:argumentMap ([ rml:reference "geometry" ];)
  ];
];

<#shp_example>
 rml:logicalSource {
  rml:source "/path/to/shapefile/file.shp";
  rml:referenceFormulation ql:SHP;
  rml:iterator "gis_osm_natural_free_1";
};
 rr:subjectMap [
  rr:template "http://example.di.uoa.gr/id/{GeoTriplesID}";
  rr:class onto:gis_osm_natural_free_1;
];
 rr:predicateObjectMap [
  rr:predicateMap [ rr:constant onto:hasName ];
  rr:objectMap [
    rr:datatype xsd:string;
    rml:reference "name";
  ];
];
 rr:predicateObjectMap [
  rr:predicateMap [ rr:constant onto:hasGeometry ];
  rr:objectMap [
    rr:template "http://example.di.uoa.gr/Geometry/{GeoTriplesID}";
  ];
];
```

Mapping Processor

- If the input mapping is an R2RML GeoTriples uses an extended version of D2RQ
- If the input mapping is an RML, GeoTriples uses an extended version of the iMinds processor.
- The transformation follows three main steps

1. Extracts the content of the logical source (e.g. using a SELECT query)
2. Defines Subject based on the Term map of the Subject map
3. Iterate the entities and forms the triples based on the Predicate object maps
GeoTriples System Architecture
GeoTriples-Spark

- We extended GeoTriples to run on top of Apache Spark
- It can run in standalone or in any cluster that supports Apache Spark
- GeoTriples-Spark is capable of transforming large amount of data into RDF graphs
- RML Mappings
- It supports the transformation of CSV, GeoJSON and ESRI Shapefiles
Apache Spark

- Apache Spark is an open-source, distributed, general-purpose, cluster-computing framework.
- Spark uses a master/worker architecture.
- There is a Driver (JVM process) that talks to a single coordinator called Master which manages Workers in which Executors (JVM processes) run.

RDD

- **Resilient Distributed Dataset (RDD)** is an immutable distributed collection of elements of data.
- The data is partitioned across machines in the cluster, which can be operated in parallel using transformations and actions.
- **Dataframes** and **Datasets** are immutable collections of data in which is organized into named columns.
GeoTriples-Spark

Architecture
GeoTriples-Spark

- The Input data are loaded as multiple partitions, distributed across the cluster
- The RML mappings are extracted from the RML file and broadcasted to all servers
- Each partition is transformed into RDF triples by an RML processor
- The number of concurrent tasks is defined by the number of partitions and the number of the Executors (and their cores)
- There is no need for intermediate caching
- Except the broadcasting of the RML mappings, no further data shuffling occurs.
- Each partition is transformed independently from the rest.
CSV files are considered text files

The geometry feature in CSV files is expected to be in Well Known Text (WKT)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Records</th>
<th>#Produced Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1GB.csv</td>
<td>1GB</td>
<td>5M</td>
<td>58M</td>
</tr>
<tr>
<td>10GB.csv</td>
<td>10GB</td>
<td>52M</td>
<td>540M</td>
</tr>
<tr>
<td>100GB.csv</td>
<td>100GB</td>
<td>0.5B</td>
<td>5B</td>
</tr>
<tr>
<td>250GB.csv</td>
<td>250GB</td>
<td>1.3B</td>
<td>13B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Times loaded</th>
<th>Input Size</th>
<th>#Executors</th>
<th>Output Size</th>
<th>#Execution time (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1GB.csv</td>
<td>1</td>
<td>1GB</td>
<td>2</td>
<td>7.7GB</td>
<td>1</td>
</tr>
<tr>
<td>10GB.csv</td>
<td>1</td>
<td>10GB</td>
<td>21</td>
<td>83.4GB</td>
<td>1.6</td>
</tr>
<tr>
<td>100GB.csv</td>
<td>1</td>
<td>100GB</td>
<td>41</td>
<td>840.1GB</td>
<td>3.3</td>
</tr>
<tr>
<td>250GB.csv</td>
<td>1</td>
<td>250GB</td>
<td>60</td>
<td>2.1TB</td>
<td>6.6</td>
</tr>
<tr>
<td>250GB.csv</td>
<td>2</td>
<td>500GB</td>
<td>65</td>
<td>4.1TB</td>
<td>13</td>
</tr>
<tr>
<td>250GB.csv</td>
<td>4</td>
<td>1TB</td>
<td>70</td>
<td>8.3TB</td>
<td>26</td>
</tr>
<tr>
<td>250GB.csv</td>
<td>8</td>
<td>2TB</td>
<td>80</td>
<td>16.6 TB</td>
<td>50</td>
</tr>
</tbody>
</table>
Scalability Experiments

Weak Scaling

Strong Scaling
ESRI Shapefile: Transformation and Evaluation

- ESRI Shapefile is a file format for storing spatial data, and consist of multiple file (.shp, .shx, .dbf)
- **GeoSpark** is used for loading the input Shapefile into a Spark Dataset
- GeoSpark is an in-memory cluster computing framework developed by the Data Systems Lab, in order to support spatial data types, indexes, and processing of large-scale spatial data.
- It is important to mention that **GeoSpark always loads the input shapefile into a single partition** as it merges all the related component files of shapefile into one.
- Therefore, each Task have to transform a whole shapefile into RDF.

*Jia Yu, Jinxuan Wu, Mohamed Sarwat: GeoSpark: a cluster computing framework for processing large-scale spatial data. SIGSPATIAL/GIS 2015*
GeoTriples-Spark

ESRI Shapefile: Transformation and Evaluation

- There is a 2 GB size limit for any of the component files of shapefile, which translates to a maximum of roughly 70 million point features.
- Therefore GeoTriples-Spark provides the option to load multiple shapefiles and transform them at once, by specifying a folder containing shapefile folders.
- GeoTriples-Spark loads each shapefile into an individual Spark Dataset and in the end it unites them into one.
- Otherwise, if the user want to transform a single big shapefile, it can repartition it into multiple partitions which will be transformed in parallel.
# GeoTriples-Spark

## ESRI Shapefile: Transformation and Evaluation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>.shp size</th>
<th>.dbf size</th>
<th>Total size</th>
<th>#Records</th>
<th>#Produced Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoadSystem_AUS</td>
<td>381.1MB</td>
<td>278.9MB</td>
<td>672.4 MB</td>
<td>1615868</td>
<td>17165376</td>
</tr>
<tr>
<td>RoadSystem_GER</td>
<td>1.7GB</td>
<td>1.9GB</td>
<td>3.7 GB</td>
<td>11107532</td>
<td>115146843</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Times</th>
<th>Input Size</th>
<th>#Executor</th>
<th>Output Size</th>
<th>Execution Time (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoadSystem_AUS</td>
<td>2</td>
<td>1.3GB</td>
<td>1</td>
<td>5.5 GB</td>
<td>1.2</td>
</tr>
<tr>
<td>RoadSystem_AUS</td>
<td>16</td>
<td>9.8GB</td>
<td>3</td>
<td>41.6 GB</td>
<td>2.5</td>
</tr>
<tr>
<td>RoadSystem_AUS</td>
<td>153</td>
<td>100.5 GB</td>
<td>20</td>
<td>427.7 GB</td>
<td>4.3</td>
</tr>
<tr>
<td>RoadSystem_AUS</td>
<td>381</td>
<td>250.2 GB</td>
<td>30</td>
<td>1068.6 GB</td>
<td>9.9</td>
</tr>
<tr>
<td>RoadSystem_GER</td>
<td>136</td>
<td>502.9 GB</td>
<td>15</td>
<td>2.5 TB</td>
<td>17</td>
</tr>
<tr>
<td>RoadSystem_GER</td>
<td>258</td>
<td>1TB</td>
<td>27</td>
<td>5.1 TB</td>
<td>34</td>
</tr>
</tbody>
</table>
GeoTriples-Spark and TripleGeo-Spark

Execution of SHP

GeoTriples-Spark  |  TripleGeo-Spark  |  GeoTriples

Execution time (s)

- 440MB
- 764MB
- 1.7GB
- 3.7GB

GeoTriples-Spark Performance

Execution Time (min)

- Input Size (GB): 1, 2, 5, 10, 20, 30, 50
- GeoTriples-Spark
- GeoTriples
- TripleGeo-Spark

K Patroumpas, D Skoutas, et al., Exposing Points of Interest as Linked Geospatial Data. SSTD 2019
Geospatial Interlinking
Entity Resolution

- The task of identifying entities that correspond to the same real-world objects, but are located in different data collections.

- An essential task for
  - data cleaning
  - data integration
  - document clustering

- **Similarity Join**: Find all pairs of records from a given set, that have similarity score greater than a predefined similarity threshold, under a given similarity function.
Entity Resolution

- The comparisons of two entities is expensive

- If we were up to compare all the entities with each other (in a dataset with $n$ entities) it would correspond to a complexity of $n^2$ (or between two datasets of $m$, $n$ sizes it would result to an $nm$ complexity)

- This is too expensive!
- Contains unnecessary comparisons.
Entity Resolution

- We use **Blocking**

- The idea is to divide the entities into blocks, and to compare only the entities within a block

- **Block Building** clusters very similar entities into blocks so as to drastically reduce the candidate match space and to cut down on the running time

  **This may lead to missing matches**
Spatial Interlinking

- Match all the Geometries that relate
  - intersection
  - cross
  - contains, etc

- Comparing Geometries is very expensive
Spatial Interlinking

Ideally we have three comparisons:

- $g_1 - g_3$
- $g_1 - g_2$
- $g_3 - g_4$
Similarly with Entity resolution, we apply something like Blocking

- We divide the space into blocks
- Using constant lat/long granularity
TestMBR

- $g_1$ and $g_4$ co-exist in the same block $b_{11}$, therefore will be compared.

- **They don’t relate, the comparison is redundant**

- The **Minimum Bounding Box (MBR)** of a Geometry, is defined as
  
  \[
  \text{MBR}(G) = (G.\text{min}X, G.\text{max}X, G.\text{min}Y, G.\text{max}Y)
  \]

- Comparing MBRs, is cheap

Contains($G_1$, $G_2$) =

\[
G_1.\text{min}X \leq G_2.\text{min}X \&\& G_1.\text{max}X \geq G_2.\text{max}X \&\& G_1.\text{min}Y \leq G_2.\text{min}Y \&\& G_1.\text{max}Y \geq G_2.\text{max}Y
\]
TestMBR

- **TestMBRs:**
  - If the MBRs of two geometries intersects then the geometries may relate
  - If the MBRs don’t intersect, then the geometries will not relate

- The MBRs of $g_1$ and $g_4$ don’t intersect, hence can’t relate
  
  We omit the comparison

- Using **TestMBRs** we can avoid some redundant comparisons
Reference Point
Reference Point

- \textbf{g1} belongs in: b00, b01, b10, b11, b20, \textbf{b21}, b30, \textbf{b31}
- \textbf{g2} belongs in: \textbf{b21}, b22, \textbf{b31}, b32
- Their common blocks are \textbf{b21} and \textbf{b31}, hence the comparison will be performed \textit{twice}

- In order to avoid duplicate comparisons, we apply the comparison only in the block that contains a \textbf{Reference Point}
Reference Point
In case we want to find all the relations between two geometries we can use the DE9IM model.

The **Dimensionally Extended 9-Intersection Model (DE-9IM)** is a topological model and a standard used to describe the spatial relations of two regions.

Using DE9IM we can express all the possible relations between two geometries.

\[
\text{DE9IM}(a, b) = \begin{bmatrix}
\dim(I(a) \cap I(b)) & \dim(I(a) \cap B(b)) & \dim(I(a) \cap E(b)) \\
\dim(B(a) \cap I(b)) & \dim(B(a) \cap B(b)) & \dim(B(a) \cap E(b)) \\
\dim(E(a) \cap I(b)) & \dim(E(a) \cap B(b)) & \dim(E(a) \cap E(b))
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>dim: dimensions</th>
<th>B: Boundaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>E: Exterior</td>
<td>I: Interior</td>
</tr>
</tbody>
</table>
Parallel Spatial Interlinking

- Partition the input geometries using a Grid Algorithm
  - KDBTrees
  - QUADTrees
  - EQUIGRID
- Assign each process with a partition and perform the mentioned techniques in parallel
Thank you