A Planet Scale Spatial-Temporal Knowledge Graph Based On OpenStreetMap And H3 Grid

Martin Böckling[1,*], Heiko Paulheim[1] and Sarah Detzler[2]

[1]Data and Web Science Group University of Mannheim, B6 26, Mannheim, 68159, Germany
[2]Corporate State University of Mannheim, Colblitzallee 1-9, Mannheim, 68163, Germany

Abstract

Geospatial data plays a central role in modeling our world, for which OpenStreetMap (OSM) provides a rich source of such data. While often spatial data is represented in a tabular format, a graph based representation provides the possibility to interconnect entities which would have been separated in a tabular representation. We propose in our paper a framework which supports a planet scale transformation of OpenStreetMap data into a Spatial Temporal Knowledge Graph. In addition to OpenStreetMap data, we align the different OpenStreetMap geometries on individual h3 grid cells. We compare our constructed spatial knowledge graph to other spatial knowledge graphs and outline our contribution in this paper. As a basis for our computation, we use Apache Sedona as a computational framework for our Spatial Temporal Knowledge Graph construction.

Keywords

Spatial Temporal Knowledge Graph, OpenStreetMap, Apache Sedona

1. Introduction

Within the spatial domain OpenStreetMap (OSM) provides a large extent of open-source spatial data that is modeled by different contributors. It provides data annotated by different contributors and official data providers, which represents data over the entire planet [1]. Often spatial data is represented in a tabular format. Graphs, but in particular Knowledge Graphs (KGs), provide a good foundation to interconnect related entities in the spatial domain. They can model entities and events in a multi-faceted way, where distance in the graph can express both geographic as well as semantic distances. When embedding such a graph using knowledge graph embedding methods, it is possible to create latent spaces where both facets, geographic and semantic proximity, are jointly reflected.

For our research paper, we focus on the spatial data source provided by OSM. We use the h3 Discrete Global Grid (DGG) to provide a regularization for the different OSM geometries, so that each individual cell tessellates the earth uniquely. This does not only involve data from a one-time snapshot, but we aim at providing a KG that is modeled over a temporal dimension. In the following sections, we outline our approach for a scalable Spatial-Temporal Knowledge

GeoLD2024: 6th Geospatial Linked Data Workshop, May 26, 2024, Hersonissos, Greece
*Corresponding author.
martin.boeckling.gast@uni-mannheim.de (M. Böckling); heiko@informatik.uni-mannheim.de (H. Paulheim); sarah.detzler@dhbw-mannheim.de (S. Detzler)
0000-0002-1143-4686 (M. Böckling); 0000-0003-4386-8195 (H. Paulheim); 0000-0002-7504-8856 (S. Detzler)

Graph (STKG) over the entire planet. Furthermore, we conclude our research by comparing it conceptually to other STKGs.

2. Related Work

Within the spatial data domain, various data representations for graph-based datasets are feasible. One representation involves static nodes and edge pairs. Static graphs, exemplified by road networks or electricity grids [2], are a specific instance of this data representation. The capability to establish connections between spatial information through a graph enables the modeling of links between individual geometries or multiple geometries, providing a means to represent complex systems [3].

In the realm of KGs within the spatial domain, our focus is directed towards a curated selection of diverse KGs. In this section we will highlight a selection of KGs, but we will not provide an extensive list of different KGs. One notable KG, WorldKG, is specifically designed to encapsulate OSM data. The structural organization of WorldKG involves the transposition of features extracted from OSM, establishing relationships between various categories. In this KG, tags derived from OSM function as separate entities, framing the hierarchical structure of WorldKG. Each individual feature extracted from OSM is represented as a distinct node within the WorldKG. Moreover, the geometries associated with each OSM entity are also exposed as point geometry as nodes within the WorldKG. However, no grid is used to represent geographical areas. In total, the published KG is based on OSM data from June 06, 2021. In total, the published KG contains over 828 million triples and over 113 million entities, categorized into 33 different top-level classes. [4]

In comparison to WorldKG, the KnowWhereGraph framework is constructed upon a diverse array of datasets encompassing hazard information, climate data, soil properties, crop and land cover types, as well as demographic and human health data. To facilitate the integration of these heterogeneous datasets, the framework employs the S2 discrete hierarchical grid, thereby harmonizing location data from various sources. The S2 grid uses squared shapes as grid cells, and the hierarchy allows larger and smaller granularity. Each discrete hierarchical grid cell functions as a unique identifier for the corresponding region. In conjunction to the grid cell Identifier (ID), various other regional attributes, such as ZIP codes, administrative regions, or Climate Division Boundaries, are systematically mapped to the respective areas. During the time of the publication, the Knowledge Graph consisted of 4.9 billion triples. [5]

The current approaches for constructing spatial Knowledge Graphs come with certain shortcomings. For instance, WorldKG currently considers only Point geometry types as an input. While providing a semantically enriched representation for OSM metadata, geographic near elements are not necessarily close in the graph. In comparison, KnowWhereGraph uses other data sources limited to selected datasets from the United States. It provides in addition relations between geometries and the DGG.

In this paper, we propose a framework that allows us to transform OSM data into a KG. Core considerations for our approach are the involvement of a DGG together with a modeling of the relations between the geometry and an individual grid cell. Furthermore, the KG should provide a temporal dimension to the dataset. In the following sections, we provide an overview

of the theoretical background of spatial data, as well as the transformation into a KG.

3. Theoretical Background of Spatial Data and Knowledge Graphs

For the following sections we introduce the related concepts for our KG with concepts from the spatial domain. Furthermore, we will provide an introduction to the data structure of OSM.

3.1. OpenStreetMap

OSM provides a data foundation that allows to map geographical entities. The complete project is open source and therefore provides a unique opportunity in the spatial domain for data analysis. OSM provides two different types of data categories. The first data category involves the so-called OSM map features. Map features provide additional information to OSM entities and represent geographical attributes. Map features are modeled by using key-value pairs, in which the key models the primary feature and the value further specifies the primary feature. We define OSM map features as followed [6]:

Definition 1. Let K be the set of all possible keys and V the set of all possible values. We therefore denote all tags as T. Therefore, OSM map features are represented as $T : K \rightarrow V$.

For instance, the key *highway* can be specified with the value *motorway*. The map feature *highway=motorway* describes therefore a divided highway with more than two lanes. OSM does not restrict the variety of map features that can be defined, which also provides the possibility to have multilingual map features defined by users. Nevertheless, OSM provides commonly accepted map features [6].

The second data category is described as OSM elements. They are divided into Nodes, Ways and Relations. The Node element represents a point within a geographic space. Nodes represent its geographic location as a pair of longitude and latitude. Nodes can represent, for instance, coffee shops, trees, or park benches. We use for the Node element the following definition:

Definition 2. Let A_{node} be the set of OSM Node attribute names and let W_{node} be the possible values for the attributes A_{node}. We define properties P_{node} as $P_{node} : A_{node} \rightarrow W_{node}$. For node objects N_{node} we define the following attributes $A_{node} = \{"id", "version", "changeset", "lat", "lon", "user", "uid", "visible", "timestamp"\}$. A node O_{node} consists of the following tuple $O_{node} = (P_{node}, T)$.

Ways represent in the context of OSM linear geometries. In general, Ways can be translated into Line or Polygon geometries. A Way within OSM consists of a set of Node elements. The combination of the Point location from nodes constructs the Line or Polygon geometries. Objects that use Line geometries are highways or power lines. Polygons encode, for example, buildings or land-use areas. The data structure of a OSM Way is defined as followed:

Definition 3. Let A_{way} be the set of OSM way attribute names and let W_{way} be the possible values for the attributes A_{way}. We define P_{way} as $P_{way} : A_{way} \rightarrow W_{way}$. We define N to be the set of of node references in a way. Each element in N references a

node identified by an ID. For way objects O_{way} we define the following attributes $A_{way} = \{"id", "version", "changeset", "user", "uid", "timestamp"\}$. A way object O_{way} consists of the following tuple $O_{way} = (P_{way}, N, T)$.

The third element of the OSM data structure is a relation. Relations consist of Nodes, Ways, or other Relations to define logical geographical relations to the different elements. Each element within a relation specifies a role element, which represents the function of the element within the relation. Those generally represent administrative boundaries and routes. The definition of an OSM relation is defined as followed:

Definition 4. Let $A_{relation}$ be the set of OSM relation attribute names and let $W_{relation}$ be the possible values for the attributes $A_{relation}$. We define the relation property $P_{relation}$ as $P_{relation} : A_{relation} \rightarrow W_{relation}$. Within a relation we define members M that either represent Nodes or Ways. Each member is consisting of a tuple of type, reference and role. For $O_{relation}$ we define following attributes $A_{relation} = \{"id", "version", "changeset", "user", "uid", "timestamp"\}$. We define a relation object $O_{relation}$ as a tuple consisting of relation properties $P_{relation}$, members M and tags T: $R_{relation} = (P_{relation}, M, T)$.

Based on the data structure of OSM, a geometry for an OSM entity cannot be retrieved directly. Instead, the geometry for each of the three different data structures needs to be constructed explicitly. In the following subsection 3.2 the theoretical foundation for the used spatial methods are outlined.

3.2. Spatial Foundation

The geometries that are extracted from OSM can range from simple point geometries to more complex geometry representations like Polygons or Geometry Collections. To build up the relationships between the different geographic relationships, the topological model Dimensionally Extended 9-Intersection Model (DE-9IM) is used. Similarly to our methodology, KnowWhereGraph also uses the DE-9IM methodology to model the geometry-grid relationship [5]. Furthermore, different scientific papers use the DE-9IM method to model the spatial relationship between spatial geometries [7].

The DE-9IM models the relationships between geometries by using a 3x3-dimensional matrix. It models the relationship between the different geometries by intersecting the interior I, exterior E and boundary B of two geometries. Assuming two geometries a and b, the dimension of the intersection between geometric objects a and b can be calculated with the function dim. The dimension function dim of a general set of geometry S returns for relation determination the highest value, and is defined as followed [8]:

$$\dim(S) = \begin{cases} -1 & \text{if } S \text{ is empty} \\ 0 & \text{if } S \text{ contains a point and no lines or areas} \\ 1 & \text{if } S \text{ contains a line and no areas} \\ 2 & \text{if } S \text{ contains an area} \end{cases}. \tag{1}$$

The complete intersection matrix on which the DE-9IM builds up is given in Equation 2. The DE-9IM method intersects the two geometric objects a and b for each of the topological

properties of I, B and E. Based on the intersection of both geometries, the dimension function *dim* is applied. [8]

$$\text{DE9IM}(a,b) = \begin{bmatrix} dim(I(a) \cap I(b)) & dim(I(a) \cap B(b)) & dim(I(a) \cap E(b)) \\ dim(B(a) \cap I(b)) & dim(B(a) \cap B(b)) & dim(B(a) \cap E(b)) \\ dim(E(a) \cap I(b)) & dim(E(a) \cap B(b)) & dim(E(a) \cap E(b)) \end{bmatrix} \tag{2}$$

When applying the DE-9IM model to two geometries, the 3×3 matrix is filled with the respective result of the *dim* function. By concatenating the output of the DE-9IM associated spatial predicates can be determined. For the determination of the spatial predicates, the value range from the function dim displayed in Equation 1 can be masked with the symbol set T, F *. Element T masks dim values 0, 1, 2, element F masks the return value -1, and element * masks the values -1, 0, 1, 2. When applying the DE-9IM method, the results can be merged into a string by concatenating element-wise the results from the method. In Table 1 a selection of the DE-9IM patterns are shown to predicates [9].

Table 1
Spatial predicate and associated DE-9IM pattern [9]

Spatial predicate	Spatial geometry combination	DE-9IM pattern
Equals	All	T*F**FFF*
Disjoint	All	FF*FF****
Touches	All except Point ∩ Point	FT******* ∨ F**T***** ∨ F***T****
Crosses	Point ∩ Line	T*T******
Crosses	Line ∩ Line	0********
Within	All	T*F**F***
Overlaps	Point ∩ Point, Area ∩ Area	T*T***T**
Overlaps	Line ∩ Line	1*T***T**
Intersects	All	a.intersects(b) =¬b.disjoint(a)
Contains	All	a.contains(b) = b.within(a)

As a basis for our STKG, we use a DGG to harmonize our geometries that are extracted from OSM similar to KnowWhereGraph [5]. With DGGs we have the possibility to extract global unique IDs per individual grid cell which allows for extensibility of the KG. Instead of using the S2 DGG, we will base our geometry relation on the h3 DGG. The most important difference is that the S2 DGG uses a square-based grid cell geometry, whereas h3 uses a hexagonal-based grid cell geometry [10]. The main reason for our decision is that in a hexagonal grid, the distance to all neighboring cells is uniform, which is not the case for square- or triangular-based grids.

4. Outline of Knowledge Graph representation

For our KG we provide a representation where the geometries of a geographic entity are aligned on a DGG and modeled over time. Besides the geographic relation also the tag information provided by OSM is encoded within the KG. Overall, the representation for the KG ontology should respect the outlined rules:

- Represent spatial entities from OSM in quadruple structure
- Include hierarchical relations of common OSM tags in KG
- Provide spatial relationship between OSM geometries and DGG
- Provide relationships between individual grid cells in DGG

4.1. Overview of Classes and Properties

Starting with the commonly used OSM tags, we expose the OSM entity in our ontology using the OSM ID. Between the common tags, we build a subclass relation based on the respective hierarchy in the OSM tags. For the spatial relationships, we adapt the GeoSPARQL classes and properties to align the created KG to the respective standard. The individual object of OSM has for the respective value the relation *rdf:type*. If an OSM tag is not within the commonly addressed tags, we use the key value pair within our KG. The OSM ID is the subject, the key acts as the predicate and the value is used as the object.

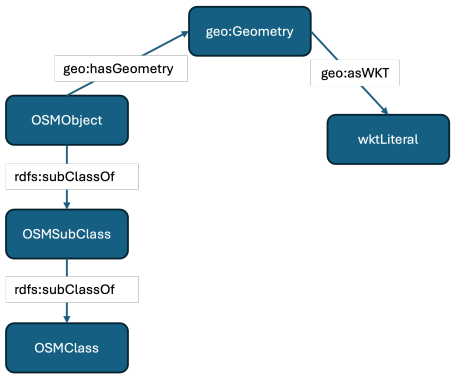

Figure 1: Ontology of our Spatial-Temporal Knowledge Graph for OpenStreetMap objects

For the different individual grid cells, we use the related DE-9IM GeoSPARQL properties to model the relation to OSM. This is based on the conditions outlined in table 1. This involves the following properties: *geo:sfContains*, *geo:sfCrosses*, *geo:sfEquals*, *geo:sfOverlaps*, *geo:sfTouches*, *geo:sfWithin*, *geo:ehCovers*, *geo:ehCoveredBy*, and *geo:sfIntersects*. Due to the hierarchical relationships between the DE-9IM methodology, multiple relationships can be directed from the grid cell to the individual OSM geometry. An example of how for one specific OSM ID the STKG is structured is outlined in subsection 4.2.

Within our STKG, we use a DGG, on which we align the OSM geometries for the KG. Specifically for DGG, an ontology has already been proposed. Similar to our ontology, for the relation to other geometries not from the grid, the GeoSPARQL classes have been adopted using, for instance, the DE-9IM methodology. However, for the selected h3 grid, the ontology can only be partially used, as the only relations for the grid cells are *hcf:isAdjacentTo* and *hcf:contains*, which are based on the DE-9IM contains predicate [11]. While this works for square-based grid systems, as grid cells on different hierarchy levels always contain the respective smaller grid cell, hexagonal grid cells or triangular grid cells on different resolutions do not fulfill this

argument. We therefore expand the hierarchical relationships properties to *isParentCellOf* and *isChildCellOf* to capture the hierarchical relationship between grid cells.

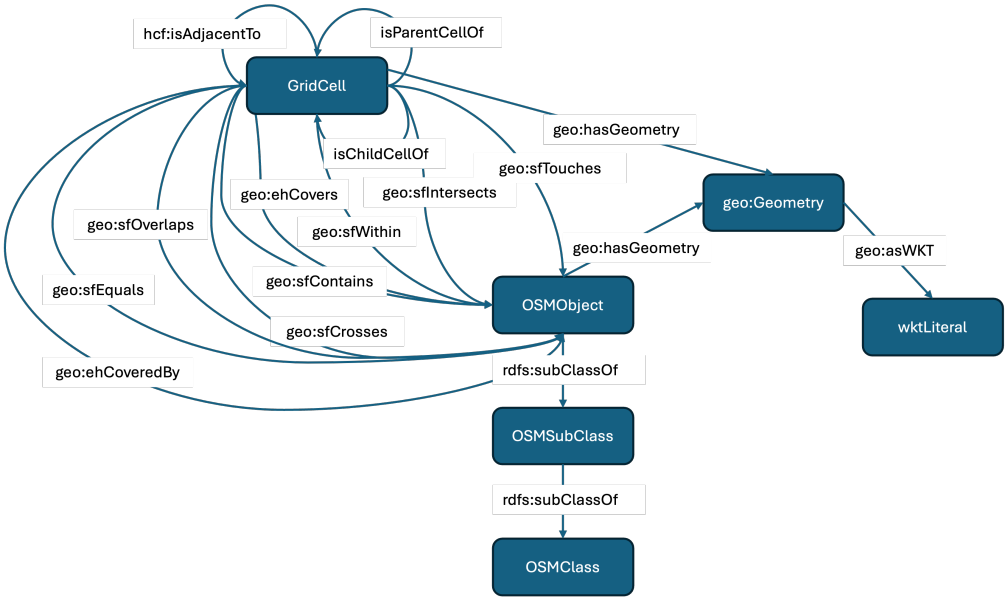

Figure 2: Ontology of our Spatial-Temporal Knowledge Graph for grid cell ontology

4.2. Example of Knowledge Graph Entity

To display how the structure of the STKG looks, we use a sample OSM object and display the respective part of the STKG. In addition to the OSM object, we also display the related h3 grid cell to the OSM object. As an example entity for OSM, we use OSM Way 240974013. In total, we transform the single element of the OSM ID and the related h3 grid cell with its neighbors into 51 quadruples. As the complete list of quadruples would be too long for the paper, we publish a full example under the following link of our Github. In table 2, we provide an excerpt from the complete example.

5. Knowledge Graph Data Preparation

For the data preparation to construct our STKG, we used the OSM files from Geofabrik. To provide an overview of the implemented approach, we have divided our data preparation into three different phases: (1) the specific OSM data preparation, (2) the h3 DGG data preparation, and (3) the STKG construction. Throughout the data preparation phase, we use Parquet files as the main data storage format. It provides the best trade-off between data storage costs and possibilities to push down queries to the file itself [12]. The capability of predicate push downs to Parquet files allows during the data preparation phase to only scan the relevant data for the necessary transformation [13, 12].

Table 2
Example of our Spatio-Temporal Knowledge Graph for the University building of Mannheim, represented in Way 240974013

subject	predicate	object	date
240974013	rdf:type	university	2024-01-01
university	rdfs:subClassOf	amenity	2024-01-01
240974013	addr:city	Mannheim	2024-01-01
240974013	geo:hasGeometry	geo240974013	2024-01-01
geo240974013	geo:asWKT	POLYGON (([...]))	2024-01-01
881fae61b9fffff	geo:sfContains	240974013	2024-01-01
881fae61b9fffff	geo:ehCovers	240974013	2024-01-01
881fae61b9fffff	geo:sfIntersects	240974013	2024-01-01

5.1. Preparation of OpenStreetMap data

We use OSM as a data foundation for our STKG. Specifically, the *.osm.pbf* files are used, which can be derived from Geofabrik[1]. Each OSM file represents a specified time stamp for a specific region. For the three different main OSM data structures, we need to convert the XML-based structure into a table-based structure for the future processing of the data. For the Node data structure, the coordinates are given, for all other data structures, the geometries need to be explicitly constructed. In order to achieve that, we use a gdal-based method called ogr2ogr, which allows us to natively read .osm.pbf files and convert them to other formats while constructing the respective geometries for the different formats. In our case, we use ogr2ogr to convert the .osm.pbf files to Parquet files to further process them for our STKG. For that, each.osm.pbf is split into five different files, which address the different layers gdal assigns to OSM files.

After the initial conversion of the .osm.pbf file, we use the five resulting files to optimize them further for the future STKG construction. For that, we use Apache Sedona as the main processing method for the STKG. In comparison to other spatial data frameworks, Apache Sedona has shown a better speed and memory efficiency when using it in computational extensive workloads [14, 15]. Therefore, we decided to use Apache Sedona as our processing engine. With Apache Sedona, the possibility exists to write Parquet files that support predicate pushdown for geometric operations. To support the spatial predicate pushdown, the files need to contain, for each geometry, a geohash and then be written back to a Parquet file. From the file name we further extract the specific date the OSM file represents and add the date as a column to the written file. Based on the transformed geometry files, we build up our grid, where we will use the h3 grid system. In subsection 5.2, we outline the approach to grid creation for our STKG.

5.2. h3 Grid Data Preparation

Based on the requirements outlined in section 4, we generate the h3 grid, which is used to align geometries on it. For the respective grid cells, we use a world map as an input to retrieve

[1]Geofabrik provides OSM data extracts in different formats under the following web page

the h3 grid cells using the h3 Python package [16]. Depending on the parameter, we use the Nominatim API to create a bounding box and clip the individual geometries of the world map. This allows to not only construct a world-wide h3 grid, but also region-specific DGG.

For the h3 grid we allow to specify grid-specific parameters. This includes the definition of the resolution for each individual grid cell. The values that could be used are in the range of $[0, 15] \subseteq \mathbb{N}$, where 0 represents the resolution of the biggest cell area with an average size of 4,357,449.41 km^2. For the resolution of 15, an individual h3 grid cell has an average area size of 0.895 m^2, which is the smallest grid cell area captured in the h3 grid. Additionally, the h3 grid system allows the compacting of all grid cells based on population density. This allows, especially for sparsely populated regions, to represent the data in fewer grid cells while still representing the same area. After receiving all the respective grid cell IDs, we retrieve for each individual grid cell the respective geometry associated with the individual grid cell. After retrieving for all specified regions all grid cells, we store the grid data in a Parquet file. Similar to OSM, we use Apache Sedona to store the geohash per individual grid cell to optimize the predicate push down to the Parquet file. After finishing the grid creation, we outline in the last step of our data preparation step, the STKG construction.

5.3. Spatio-Temporal Knowledge Graph Construction

For the creation of our STKG we use Apache Sedona, which is an optimized data transformation engine for spatial data analysis. Overall, we base our STKG creation on the defined ontology we outlined in section 4. All aspects of the STKG construction are based on Apache Sedona functionality and use it as a backbone to provide a salable transformation engine. Our STKG creation is divided into four main parts: *OSM data tags*, *OSM geometry triples*, *h3 grid cell triples*, *h3 grid and OSM geometry relation*.

As outlined in section 4, we separate our OSM tags in two different categories. In the case where it is defined as a commonly used tag, we expose the key and value of the OSM tags to individual entities. For all tags that do not fall into the category, we expose the key as a predicate and the value as the object. In addition, for all the respective we assign the date from the respective OSM entity to our STKG, marking the respective date for the individual quadruple.

For the different parts, we follow the outlined approach from section 4 where we expose for each geometry of an object in separate entities, storing them in the WKT format for our STKG. The storage of the geometry in the WKT format instead of the WKB format allows users that query the graph to be able to directly interpret the geometry of an object. Based on the respective elements for the grid, we model the neighborhood relationships for the grid cells with the same resolution. For grid cells that do not have the same resolution, the parent-child relationship is modeled and determined as outlined in section 4. For the relationships between the different OSM geometries and the grid cell geometries, we use the predicate functions from Apache Sedona to determine the DE-9IM predicates. With regard to the constructed STKG we outline in subsection 5.4 key statistics based on a selected set of osm.pbf files.

5.4. Characteristics of constructed Spatio-Temporal Knowledge Graph

For the creation of the STKG we provide a coding base for which we are able to scale spatial data analysis on large scale data representations. As a data base we use the yearly geofabrik data extracts from OSM, which involves the datasets from the year 2018 to 2024. We use the .osm.pbf files from all continents which are provided on geofabrik. This involves the regions Africa, Antarctica, Asia, Australia, Central America, Europe, North America and South America [17]. For the OSM dataset in total this results in 529,065,633 distinct OSM elements. For the h3 grid in total 3,675,984 individual grid cells are used in our STKG. In table 3 we outline the different key statistics for our STKG.

Table 3
Overview of Spatio-Temporal Knowledge Graph statistics

Metric name	Count
Total triples	27,042,753,856
Distinct entity count	1,841,912,579
Distinct predicate count	98,955

6. Conclusion and Outlook

We have provided in our research a framework for the creation of STKG over the entire planet based on OSM. Our data foundation builds up on a large representation of spatial data, representing various types of data. Representing those aspects in a STKG allows users to interact with changing spatial data over time. Similar to the outlined use cases of WorldKG or KnowWhereGraph, our STKG allows the possibility to explore connected entities based on shared metadata information or geographic relations.

In comparison to the presented KGs in section 2, our approach provides a holistic representation of spatial data over the planet Earth. Compared to WorldKG, we provide with our STKG a representation of all available geometries in OSM. Additionally, the usage of the DGG allows consumers to use the constructed STKG for various downstream tasks where the spatial grid is involved. Similar to KnowWhereGraph we allow consumers to make use of the individual DGG cell in their downstream tasks. Compared to both presented KGs our STKG has with over 27 billion triples and over 1,8 billion entities the largest coverage of spatial entities.

For our current implementation, we rely on third-party tooling provided by GDAL to transform the OSM data format. While the module ogr2ogr shows a great efficiency in the transformation process, the handling of multiple large .osm.pbf files provides a certain overhead in the transformation phase. In a future implementation, the native support of .osm.pbf files in frameworks like Apache Sedona might help to reduce current initial processing times that are outside the data preparation related to the STKG construction. While OSM provides a rich set of spatial data, it needs to be emphasized that OSM data does not reflect the exact spatial reality and is also subject to vandalism.

With our approach, we have showcased that for the complete planet, all relevant geometries

from OSM and metadata. However, our current approach poses some limitations when it comes to traditional KG specific standards. We currently produce as an output file format delta files for our STKG. It provides for our large resulting STKG a good trade-off between the compression size of our data and the fulfillment of the ACID transaction consistency. However, KG specific frameworks like (Geo)SPARQL or are not directly supported in our file structure out of the box. There has been research around the mapping of SPARK SQL to SPARQL [18, 19] or GeoSPARQL [20] to support efficient queries on large KGs. For future research, this could be evaluated compared to traditional KG frameworks that STKG. In addition, in future research a comparison of the different STKGs on downstream spatial benchmark datasets could be performed to compare the different STKG.

Acknowledgments

Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org.

References

[1] OpenStreetMap contributors, OpenStreetMap, 2024. URL: https://www.openstreetmap.org/about.

[2] M. Barthélemy, Spatial networks, Physics Reports 499 (2011) 1–101. doi:10.1016/j.physrep.2010.11.002.

[3] R. G. Morris, M. Barthelemy, Transport on Coupled Spatial Networks, Physical Review Letters 109 (2012) 128703. doi:10.1103/PhysRevLett.109.128703.

[4] A. Dsouza, N. Tempelmeier, R. Yu, S. Gottschalk, E. Demidova, WorldKG: A World-Scale Geographic Knowledge Graph, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, ACM, Virtual Event Queensland Australia, 2021, pp. 4475–4484. doi:10.1145/3459637.3482023.

[5] K. Janowicz, P. Hitzler, W. Li, D. Rehberger, M. Schildhauer, R. Zhu, C. Shimizu, C. K. Fisher, L. Cai, G. Mai, J. Zalewski, L. Zhou, S. Stephen, S. Gonzalez, B. Mecum, A. Lopez-Carr, A. Schroeder, D. Smith, D. Wright, S. Wang, Y. Tian, Z. Liu, M. Shi, A. D'Onofrio, Z. Gu, K. Currier, Know, Know Where, KnowWhereGraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence, AI Magazine 43 (2022) 30–39. doi:10.1002/aaai.12043.

[6] OpenStreetMap contributors, Map features - OpenStreetMap Wiki, 2023. URL: https://wiki.openstreetmap.org/wiki/Map_features.

[7] E. Romanschek, C. Clemen, W. Huhnt, A Novel Robust Approach for Computing DE-9IM Matrices Based on Space Partition and Integer Coordinates, ISPRS International Journal of Geo-Information 10 (2021) 715. doi:10.3390/ijgi10110715.

[8] E. Clementini, J. Sharma, M. J. Egenhofer, Modelling topological spatial relations: Strategies for query processing, Computers & Graphics 18 (1994) 815–822. doi:10.1016/0097-8493(94)90007-8.

[9] T. Feng, Z. Zeng, X. Wu, R. Liu, L. Gao, Discovery of Multi-Level Spatial Association Rules

Based on DE-9IM, in: 2010 International Conference on Management and Service Science, IEEE, Wuhan, China, 2010, pp. 1–5. doi:10.1109/ICMSS.2010.5577834.
[10] M. Li, E. Stefanakis, Geospatial Operations of Discrete Global Grid Systems—a Comparison with Traditional GIS, Journal of Geovisualization and Spatial Analysis 4 (2020) 26. doi:10.1007/s41651-020-00066-3.
[11] C. Shimizu, R. Zhu, G. Mai, C. Fisher, L. Cai, M. Schildhauer, K. Janowicz, P. Hitzler, L. Zhou, S. Stephen, A Pattern for Features on a Hierarchical Spatial Grid, in: Proceedings of the 10th International Joint Conference on Knowledge Graphs, ACM, Virtual Event Thailand, 2021, pp. 108–114. doi:10.1145/3502223.3502236.
[12] J. Hu, Y. Wang, F. Shi, C. Xu, Compared Analysis of Row-Based Storage and Column-Based Storage, in: 2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), IEEE, Harbin, China, 2018, pp. 168–173. doi:10.1109/IMCCC.2018.00043.
[13] Z. Luo, L. Niu, V. Korukanti, Y. Sun, M. Basmanova, Y. He, B. Wang, D. Agrawal, H. Luo, C. Tang, A. Singh, Y. Li, P. Du, G. Baliga, M. Fu, From Batch Processing to Real Time Analytics: Running Presto® at Scale, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, Kuala Lumpur, Malaysia, 2022, pp. 1598–1609. doi:10.1109/ICDE53745.2022.00165.
[14] R. Y. Tahboub, T. Rompf, Architecting a Query Compiler for Spatial Workloads, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, ACM, Portland OR USA, 2020, pp. 2103–2118. doi:10.1145/3318464.3389701.
[15] V. Pandey, A. Kipf, T. Neumann, A. Kemper, How good are modern spatial analytics systems?, Proceedings of the VLDB Endowment 11 (2018) 1661–1673. doi:10.14778/3236187.3236213.
[16] Uber Technologies, h3-py: Uber's H3 Hexagonal Hierarchical Geospatial Indexing System in Python, 2023. URL: https://uber.github.io/h3-py.
[17] M. Boeckling, Geofabrik data links, 2024. doi:10.5281/ZENODO.10798949.
[18] D. Graux, L. Jachiet, P. Genevès, N. Layaïda, SPARQLGX: Efficient Distributed Evaluation of SPARQL with Apache Spark, in: P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, Y. Gil (Eds.), The Semantic Web – ISWC 2016, volume 9982, Springer International Publishing, Cham, 2016, pp. 80–87. doi:10.1007/978-3-319-46547-0_9, series Title: Lecture Notes in Computer Science.
[19] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, G. Lausen, S2RDF: RDF Querying with SPARQL on Spark (2015). doi:10.48550/ARXIV.1512.07021.
[20] D. Bilidas, T. Ioannidis, N. Mamoulis, M. Koubarakis, Strabo 2: Distributed Management of Massive Geospatial RDF Datasets, in: U. Sattler, A. Hogan, M. Keet, V. Presutti, J. P. A. Almeida, H. Takeda, P. Monnin, G. Pirrò, C. d'Amato (Eds.), The Semantic Web – ISWC 2022, volume 13489, Springer International Publishing, Cham, 2022, pp. 411–427. doi:10.1007/978-3-031-19433-7_24, series Title: Lecture Notes in Computer Science.

A. Online Resources

The documentation as also the coding for our research paper can be found on GitHub