Master's Thesis

Integrating Multi-Modal Spatial Data using Knowledge Graphs – a Case Study of Microflora Danica

Thomas Heede[1]
Mads Corfixen[1]

Supervised by
Thomas Dyhre Nielsen[1]
Katja Hose[2,1]

[1]Department of Computer Science
Aalborg University
Denmark

[2]Institute of Logic and Computation
Technische Universität Wien
Austria

Resumé

Afhandlingen bygger på artiklen "Integrating Multi-Modal Spatial Data using Knowledge Graphs – a Case Study of Microflora Danica" og et omfattende appendiks, der undersøger, hvordan semantisk relaterede, heterogene og multimodale datasæt kan integreres, når en af modaliteterne er spatial. Arbejdet tager udgangspunkt i et konkret case-studie med det danske Microflora Danica (MfD)-datasæt, der indeholder geografisk annoterede data omkring mikroorganismer, som er kombineret med miljødata i form af rasterfiler fra EcoDes-DK15 og jordbundskort. Målet er at skabe en fleksibel og præcis integration ved hjælp af Knowledge Graphs (KG'er), så data kan analyseres på tværs af modaliteter uden tab af væsentlig information.

Vi indleder med at præsentere de udfordringer, der opstår, når forskellige typer data skal kobles på tværs af formater, opløsninger, koordinatsystemer og semantiske strukturer. Traditionelle integrationsmetoder er ofte begrænset til tabulære eller vektorbaserede data, mens rasterdata er mindre udforsket i denne kontekst. Vi gennemgår relateret arbejde og identificerer et hul i forskningen: der findes ingen udbredt metode til at integrere rasterdata direkte i KG'er uden at miste væsentlige detaljer eller uden at være ressourcekrævende for større applikationer.

Det tekniske bidrag er en metode til at anvende Googles S2 Geometry som et fælles spatialt referencesystem. S2-geometrien opdeler Jorden hierarkisk i celler helt ned til centimeter niveau, hvilket muliggør præcis kobling af både vektor- og rasterdata uafhængigt af deres oprindelige koordinatsystemer, opløsninger eller transformationer til S2-geometrien. Lokationerne fra MfDs jordbundsprøver, repræsenteres som punkter med GPS-koordinater, kobles direkte til S2-celler, mens rasterceller fra miljødata omsættes til et tilsvarende sæt af S2-celler. Dette muliggør en ensartet kobling, hvor observationer for miljømålinger, habitattyper og mikroorganismer kan forbindes ved fælles spatiale enheder.

En central problemstilling er balancen mellem høj granularitet, som reducerer overdækning (områder, hvor S2-celler strækker sig ud over rastercellens grænser), og de øgede krav til lagringsplads, som høj opløsning medfører. Gennem evaluering af forskellige S2-niveauer viser vi, at niveau 24 giver et acceptabelt informationstab på blot 2,2%. Vi benytter en "majority rule"-strategi til at undgå, at en S2-celle kobles til flere rasterceller med forskellige værdier, hvilket ellers kunne skabe datakonflikter. Sammenligninger med op- og nedskalering af rasterdata demonstrerer, at S2-baseret integration markant reducerer informationstabet i forhold til andre metoder for at integrere forskellig data repræsenteret i gitre. Hvorvidt man er villig til at acceptere et større tab for at reducere mængden af data bør besluttes på baggrund af ens domæne.

Dertilhørende er et supplementerende appendiks, som uddybende detaljerer artiklen. Heri beskriver vi den tekniske baggrund, der ligger som fundamentet for artiklen og dele af appendikset. Desuden skematiserer vi sammenkoblingen mellem de forskellige datakilder og redegør for eventuelle uhensigtsmæssige designvalg fra det originale data. Hertil beskriver vi, hvordan det kan korrigeres. Desuden har vi givet eksempler på hver type af data og beregnet statistik for at give en dybere forståelse af tabellerne. Da artiklen havde en begrænsning i antallet af sider har vi uddybet de nævnte metoder til at repræsentere data spatialt, samt sammenlignet de nævnte metoder. Som følge af beskrivelsen af den tilgængelige data samt forståelsen af de forskellige måder at repræsentere data på, gennemgår vi, hvordan vi har transformeret dataen til RDF og giver eksempler herpå.

Slutvist beskriver vi, hvordan man kunne fortsætte arbejdet. Vi vil blandt andet kunne reducere antallet af S2-celler med 81%, hvis vi aggregerer S2-celler op til deres

forældre-celler, hvor muligt. Derudover overvejer vi også, hvordan designet af KG'en kunne ændres for at reducere mængden af nødvendig hukommelse når man skal lave forspørgsler på grafen. Dette opnås ved at opdele grafen og kun benytte sig af de dele, der er nødvendige, for at besvare forespørgslen. Vi inkluderer også, hvordan det kunne tænkes, at grafen kunne benyttes til at lave "machine learning"-modeller til yderligere analyse.

Introduction to the Thesis

For our 4+4 PhD studies, we are both part of the interdisciplinary DarkScience Project, "Illuminating Microbial Dark Matter through Data Science". The overall objective of this project is to develop new methods and approaches that enable access and analysis of microbial species and their interaction with the environment. As part of the project, we have developed a framework for integrating spatial heterogeneous data sources. The framework employs a knowledge graph to semantically unify point-based observations of microbial soil, sediment, and water samples from the Microflora Danica dataset with grid-based environmental measurements. By leveraging the hierarchical discrete global grid, S2 Geometry, to integrate all spatial data irrespective of format, the framework enables integration of both current and future data sources with minimal information loss.

This work, constituting our Master's Thesis, builds upon our publication presented at the 7th Workshop on Semantic Web Solutions for Large-scale Biomedical Data Analytics (SeWeBMeDA'24).

In addition to the paper, the thesis contains an appendix to support and complement the study, allowing readers to refer to specific sections in the appendix for a more in-depth understanding of selected topics, other perspectives not included in the published paper, as well as considerations for future work.

Throughout the main paper, we have denoted the relevant appendix sections in the margins to guide the reader to the supplementary material.

Integrating Multi-Modal Spatial Data using Knowledge Graphs – a Case Study of Microflora Danica

Mads Corfixen[1], Thomas Heede[1], Tomer Sagi[1], Mads Albertsen[2], Thomas D. Nielsen[1] and Katja Hose[3,1]

[1]Department of Computer Science, Aalborg University, Aalborg, Denmark
[2]Department of Chemistry and Bioscience, Aalborg University, Aalborg, Denmark
[3]Institute of Logic and Computation, TU Wien, Vienna, Austria

Abstract

Integrating semantically related, multi-modal, heterogeneous data sources is challenging, especially if one of the modalities includes spatial data, such as field measurements organized in geographical grids. Since geographical grids can have different rotations, be translated along one or more axes, or have different resolutions, a particular challenge when integrating such data is to reduce the information loss from projecting different grids into a common format. In this paper, we study this problem and sketch a method for integrating such spatial data using knowledge graphs. We discuss this solution in the context of a real-world use case, where we integrate geographically annotated microbial data (Microflora Danica) as well as environmental data to enable joint analysis. The first results of our experiments show that our method reduces the information loss compared to baseline methods.

Keywords

Spatial Data Integration, Knowledge Graph, S2 Geometry

1. Introduction

Integrating and linking semantically related heterogeneous and multi-modal data is an important task that is often very challenging [1, 2]. While many works have focused on integrating tabular data, defined over a fixed relational schema, recent years have seen a diversification into combining multi-modal data (such as semi-structured data from different modalities) [3] and data with a flexible schema [4].

The need for data integration appears in a multitude of domains. One such domain is the study of microbiomes, i.e., the interaction between microbes and the environment they inhabit. Understanding these interactions is essential for solving current and future environmental challenges [5]. Such data is inherently multi-modal because the DNA sequence data is of a different modality than the semantic relations between microbial species and the spatial

SeWebMeDA-2024: 7th International Workshop on Semantic Web Solutions for Large-scale Biomedical Data Analytics, May 26, 2024, Hersonissos, Greece

maco@cs.aau.dk (M. Corfixen); thhe@cs.aau.dk (T. Heede); tsagi@cs.aau.dk (T. Sagi); ma@bio.aau.dk (M. Albertsen); tdn@cs.aau.dk (T. D. Nielsen); katja.hose@tuwien.ac.at (K. Hose)
0009-0004-7237-9694 (M. Corfixen); 0009-0004-7934-1293 (T. Heede); 0000-0002-8916-0128 (T. Sagi); 0000-0002-6151-190X (M. Albertsen); 0000-0002-4823-6341 (T. D. Nielsen); 0000-0001-7025-8099 (K. Hose)

information about their environments. Spatial information in general often refers to vector or raster data. Vector data consists of (x, y, z)-coordinates associated with real-world measures of a specific area, where the area's shape is defined by vector coordinates and can take the form of points, lines, polylines, and polygons. On the other hand, raster data is a defined grid over an area where each grid cell has a measurement value. The raster file itself is associated with information such as the geographical location and the height and width of the grid.

One method to facilitate the integration of multi-modal data is through a knowledge graph (KG) [6]. An advantage of using KGs for data integration is that they can flexibly support multiple modalities [7]. KGs can cover strings, images, and audio, as well as links between them, effectively allowing heterogeneous data to be mapped directly without requiring any transformations to conform to strict schemas required by other data integration solutions. Another advantage is that KGs are associated with ontologies that clearly define the semantics of the concepts in the data sources. In contrast, table and column headers in a relational database are often only clearly defined in the mind of the designer [8].

In this paper, we present a case study for integrating spatial heterogeneous multi-modal data sources using a KG. The case study is based on the Microflora Danica[1] (MfD) dataset, an archive of microorganisms in Denmark. We present the challenges and possible solutions of enriching this dataset with data from EcoDes-DK15 [9], a high-resolution dataset of ecological descriptors, and soil maps of Denmark [10]. We focus on integrating the data along the spatial dimension, as the datasets are connected through their geographical properties, i.e., GPS coordinates for the MfD dataset and spatial raster maps for the EcoDes-DK15 and soil maps. The remainder of this paper is structured as follows. First, in Section 2, we provide an overview of existing approaches for integrating multi-modal data with a spatial component. Then, in Section 3, we present the different data sources of our case study and their different modalities. Next, in Section 4, we present our KG design for spatial integration and show how it can be extended with microbial data. In Section 5, we evaluate our spatial integration approach based on the information loss and compare it to a baseline method. Finally, Section 6 concludes the paper with a summary and an outlook to future work.

2. Related Work

Data integration using KGs has recently gained attention, mainly due to their flexibility, but also because of the advantages when dealing with heterogeneous multi-modal data [6, 11]. While many studies have explored integrating or transforming spatial data into KGs, they focus mostly on vector data [1, 12, 13, 14]. Integration of raster data has not been thoroughly explored.

GeoTriples [12] proposes a method for transforming spatial data in vector format into RDF. The method extends the mapping languages R2RML[2] and RML[3] to define rules for mapping structured and semi-structured, heterogeneous vector data to RDF graphs. They recognize that spatial data also exists in raster format but only provide a framework for transforming spatial vector data into RDF triples. TripleGeo [15] is a similar method, but the proposed framework also does not support raster data.

[1] https://www.bio.aau.dk/forskning/projekter/microflora-danica
[2] https://www.w3.org/TR/r2rml/
[3] https://rml.io/specs/rml/

Tran et al. [16] propose an ETL process for integrating files in raster format to build an RDF triple store over a designated area. They aggregate raster cells to extract a single value for a territorial area, which leads to a loss of fine-grained information.

Zhu et al. [17] model observations as aggregations on discrete global grid systems, albeit without providing an actual framework. Moreover, they describe the integration of singular event geometries, such as wildfires, rather than integrating raster data.

Several hierarchical discrete global grids (HDGG) have been proposed to represent spatial data, including H3[4], Bing Maps Tile System[5], and S2 Geometry[6]. Common among them is the hierarchical division of the Earth into subsets. H3 utilizes hexagons, but, since hexagons cannot be perfectly subdivided into smaller hexagons, child cells are only approximately contained within their parent cells. Bing Maps Tile System projects the Earth onto a map; however, this projection distorts the scales in proportion to the distance to the poles. The S2 Geometry decomposes Earth into a hierarchy of cells. Instead of mapping points to a plane, S2 maps them to a perfect sphere. Since Earth is closer to being a sphere than a plane, this creates less distortion. At the top-most level of the S2 hierarchy, Earth is represented by six cells, perfectly covering the Earth, with each lower level subdividing each cell into four children.

The KnowWhereGraph (KWG) [18, 19] is a KG using S2 Geometry as the spatial component. The KWG focuses on how to model spatial data as RDF triples, but without providing a framework for mapping between raster files and S2 Geometry. Additionally, they recognize the advantages and limitations of the S2 Geometry as the spatial component but do not quantify the error. Thus, our work is based upon the KWG, using it as a spatial layer to enable spatial data integration.

3. Data Sources and Use Case Description

In this paper, we consider a use case where we want to integrate several real-world heterogeneous data sources of microbes and ecological descriptors of their habitats, exploiting the capabilities of KGs for flexible multi-modal integration of spatially related data. In this section, we describe the available data sources and how they shape our case study.

The Microflora Danica Dataset. The Microflora Danica[7] (MfD) dataset comprises more than 10,000 measured samples collected from different sample sites in Denmark. The MfD dataset was created to provide a database of all microbes in Denmark, and contains several different modalities. Information on where each sample was taken is geographical vector data, and DNA reads can be considered as very long strings. Each sample in the MfD dataset is sequenced at least once, which is a process that constructs probable DNA reads from a sample. Through this process, each sample is associated with approximately 10 million DNA reads, R, each being a fragment of a full genome consisting of the four bases, Adenine (A), Thymine (T), Cytosine (C), and Guanine (G), found in DNA, $R \in \{A, T, C, G\}^+$. In total, the MfD contains 28 TB of reads.

The structure of the MfD dataset is shown in Figure 1. Each sample site is described in the **Fieldsample Metadata**, which contains information such as GPS location and habitat type,

[4] https://www.uber.com/en-SE/blog/h3/

[5] https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

[6] https://s2geometry.io

[7] https://www.bio.aau.dk/forskning/projekter/microflora-danica

with the habitat types linked to three external taxonomies. Each sample is sequenced multiple times, each sequencing being described in the **Sequencing Metadata**, linked to their related sample site through a *fieldsample_barcode*. The **Sequencing Metadata** describes how each sample was sequenced and handled in the laboratory. Each sequencing leads to a number of reads, collected in a single file we denote as **Read Data**, connected to their related sequencing through a *sequence_id*. In a **Read Data** file, there is, besides the reads themselves, a certainty measure for each base of the reads. Finally, each site was sampled as part of different projects, each project potentially collecting multiple samples. The projects are described in the **Project Metadata**, containing information on the parties responsible for the projects. Each sample site is linked to its related project through a *project_id*.

Appendix B.1

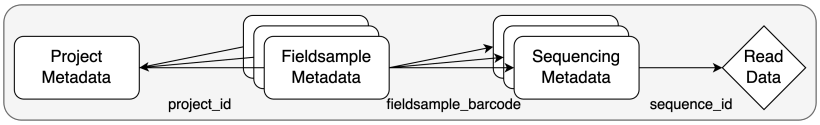


Figure 1: Overview of the Microflora Danica dataset

Besides information on the reads, sequences, and sample site, we are also provided a mapping from each sample to potential microbial species (taxons) present in that sample. Since reads are only fragments of the complete genome of a taxon, the mappings are uncertain. They are structured as shown in Table 1. Each Operational Taxonomic Unit (OTU) is a DNA sequence that encodes a specific, potentially undiscovered taxon. Each *MFD_X* column represents a *fieldsample_barcode* and shows how many reads from a sample site were mapped to that OTU.

Appendix B.2

Table 1
Mappings from sample sites to taxons

OTU	MFD_1	MFD_2	⋯	Kingdom	⋯	Species
OTU_1	0	7	⋯	Archaea	⋯	MFD_s_17257
OTU_2	633	482	⋯	Bacteria	⋯	Cornyebacterium
⋮	⋮	⋮	⋱	⋮	⋱	⋮

Environmental Raster Files. The EcoDes-DK15 [9] and soil maps [10] datasets contain measurements of ecological descriptors across Denmark, providing us a means to quantify the microbial habitats of the MfD dataset. Both datasets are in the form of raster files, adding a new modality that needs to be integrated besides the ones present in the MfD dataset.

A raster file is a matrix organized into a grid of rows and columns of raster cells, each cell representing a geographical area. Each raster cell of a raster file is furthermore associated with a value representing information about the geographical area, such as temperature or pH. In total, the environmental data contains approximately 200 GB across approximately 100 files, each containing a different type of measure.

A raster file contains the geo-location of the upper-left cell of its matrix and a transformation matrix containing information on what translations and rotations are needed to get the rest of the spatial locations of all the other cells. In our datasets, raster cells have a resolution of 10×10 meters, and the spatial location is given on the Universal Transverse Mercator (UTM)

Appendix B.3

coordinate system. In the UTM coordinate system, each location is defined by the pair (e, n) with e, *easting*, being the distance east of the Greenwich Meridian, and n, *northing*, being the distance north of the equator.

The S2 Geometry. In order to handle the integration of raster files with potentially different transformation matrices, we integrate them into a common grid, the S2 Geometry, as all raster cells in a raster file can be mapped to a set of corresponding cells in the S2 Geometry, regardless of the transformation matrix.

The S2 Geometry is a 31-level hierarchical grid that decomposes Earth into a hierarchy of cells. At the top-most level (level 0) of the hierarchy, Earth is divided into six cells perfectly covering it, while each higher level of granularity subdivides each cell into four children, such that there are 24 cells at level 1, 96 cells at level 2, and so on.

4. Spatial Integration Approach

In this section, we present our KG design for spatial integration, demonstrating how the S2 Geometry can be adopted to independently integrate the spatial aspect of each data source. Further, we show how we extend the KG design to integrate the rest of the MfD dataset. The code is available on GitHub[8].

Spatial Data. We build upon well-established ontologies to model the spatial dimension, as shown in Table 2. The *geo* ontology is used to model the spatial relations between raster cells, S2 cells, and MfD sample sites; the upper level ontology *oboe* is used to model the environmental observations and measurements; and the *kwg-ont* ontology is the ontology of the KnowWhereGraph.

Table 2
Ontologies for the spatial part of the KG design

Prefix	IRI
geo	<http://www.opengis.net/ont/geosparql#>
oboe	<http://ecoinformatics.org/oboe/oboe.1.2/oboe-core.owl#>
kwg-ont	<http://stko-kwg.geog.ucsb.edu/lod/ontology#>
rdf	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

We can consider a raster file as a collection of raster cells, where each raster cell is associated with at least one measurement. For raster files where their raster cells perfectly overlap one another, such as they do in our use case, each raster cell is associated with multiple measurements (pH value, soil salinity, etc.). This is modeled using the *oboe* concepts *ObservationCollection*, *Observation*, *Measurement*, and *MeasurementType*; and properties *hasMember*, *hasMeasurement*, *hasValue*, and *containsMeasurementsOfType* (see Figure 2).

To model the spatial relation between S2 Geometry and the MfD sample sites, we link each site to the S2 cell that covers it, using the GPS location obtained from the MfD dataset, described in Section 3. For the spatial relation between the raster cells and S2 Geometry, we associate

[8]https://github.com/MicrobialDarkMatter/MfD-spatial-integration

each raster cell with the set of S2 cells that covers it. To model these spatial relations between raster cells, S2 cells, and MfD sample sites, we utilize the *geo* ontology properties *coveredBy* and *covers*, see Figure 2, where the data from the raster files is colored in orange, the data from the MfD dataset is colored in blue, and the spatial integration layer, S2 Geometry, in bold.

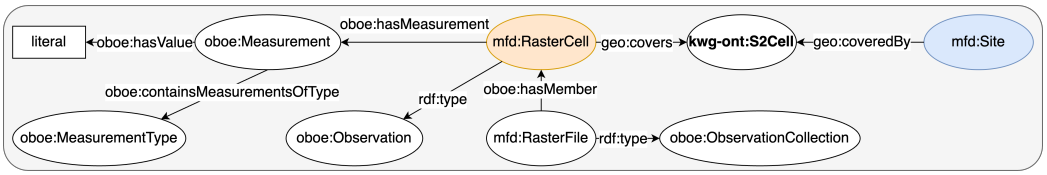


Figure 2: Central part of KG design regarding spatial data

Our current datasets do not have significant overlaps, so we do not have significant interoperability conflicts among them. However, in the future, we would like to integrate data from different sources; therefore, a semantic alignment pipeline will be needed.

Spatial Integration. The spatial aspect of each data source is mapped to the S2 Geometry. The integration can be split into two types: the vector-based GPS integration of the MfD sample sites and the area-based integration of raster files. We focus on the integration of raster files in this section, as the integration of the vector-based MfD sample sites simply requires us to integrate the given coordinates directly to the covering S2 cell.

As described in Section 3, the S2 Geometry is a hierarchy of levels determining the granularity of S2 cells. Increasing the granularity of the S2 level results in the area of each S2 cells being lowered by a factor of 4. Therefore, the S2 level affects the number of S2 cells required to cover each raster cell; a high granularity requires many S2 cells, whereas a low granularity requires fewer. Consequently, lower granularities of the S2 level increase the over-coverage, i.e, the area of S2 cells that covers an area outside their associated raster cells. The over-coverage is illustrated for two different granularities of the S2 level in Figure 3, where the S2 cells (red) are covering area outside the raster cell (green).

Ideally, a set of S2 cells covers each raster cell perfectly; however, this is not the case, and as such, choosing the highest granularity will yield the lowest possible over-coverage. The issue with this is that each level quadruples storage requirements, as each S2 cell has four child cells.

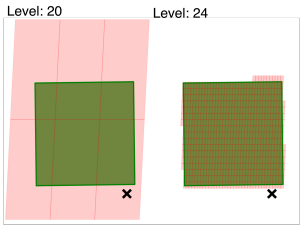


Figure 3: S2 cells covering raster cells at different levels

The EcoDes-DK15 and soil raster files have a resolution of 100 m^2; hence, to minimize over-coverage, an S2 level of 24 is chosen as S2 cells have an area of approximately 0.3 m^2 at that S2 level.

Irrespective of the S2 level, S2 cells over-covering a raster cell will overlap neighboring raster cells. This is not desirable since each S2 cell should only correspond to one value per feature, e.g., we cannot associate two pH values with the same location. To address this issue, we use a majority rule; however, one could also aggregate the values, as long as they are continuous. The majority rule disassociates S2 cells from any raster cells in which their centroids are not located. However, disassociating S2 cells from a raster will introduce under-coverage, i.e., the sub-area of a raster cell that is not covered by any S2 cells.

A potential error introduced by over- and under-coverage is integrating an MfD site to an S2 cell corresponding to an incorrect raster cell. For example, in the left of Figure 3, the sample site (cross) is linked to an S2 cell that covers the shown raster cell instead of an S2 cell covering the raster cell below it. S2 cells of higher granularity diminishes this problem.

In general, the integration of raster files is challenging due to different raster files possibly having different transformations, i.e., resolutions, rotations, and translations. However, utilizing the S2 Geometry, the integration of raster files is independent across different transformations. Thus, our approach affords the integration of arbitrary raster files.

Microbial Data. The MfD dataset contains sample-specific information such as metadata, DNA reads, sequencing metadata, sample-to-OTU mappings, and habitat information of the sample site. As the focus of this paper is on spatial integration, we omit details on this part; however, we outline the design for this integration in Figure 4. Note that each entity has properties, but due to limitations, we do not discuss metadata associated with the microbial data. Each sample site, marked in blue, has a habitat type, such as 'forests' or 'grasslands', which we map to a corresponding concept in the Environment Ontology, *ENVO* [20], marked in green. Furthermore, each sample has a mapping to an OTU, which encodes for a specific microbial taxon. These taxons are mapped to corresponding taxons in the the Taxonomy Database ontology, *NCBITaxon*[9], marked in green. Both the habitat and taxon mappings to external ontologies require some entity resolution methods to account for, e.g., spelling errors. Finally, since the DNA reads, marked in red, take up 28 TB of storage, we do not keep them directly in the graph, but instead store a reference to an external key-value store.

Appendix D.1 and Appendix D.2

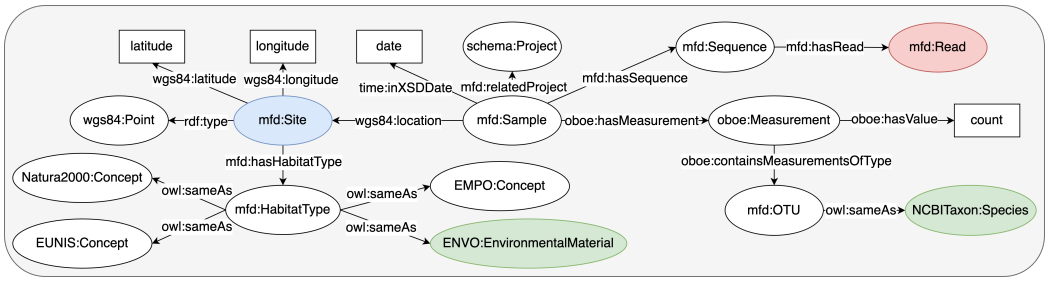


Figure 4: Extending the KG design with MfD microbial data

5. Evaluation

In this section, we evaluate the proposed framework in terms of the information loss of using the S2 Geometry as the integration layer and compare it to a baseline method. Furthermore, we highlight potential issues when using the S2 Geometry.

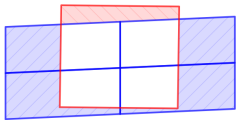


Figure 5: Over- and under-coverage

Information Loss. We observe two contributing factors to the information loss from our integration to the S2 Geometry. The first type of information loss is over-coverage (Equation 1), where a part of the area of the

[9] https://obofoundry.org/ontology/ncbitaxon.html

minimal set of covering S2 cells associated with a raster cell is outside the bounds of the raster cell. The over-coverage is calculated as the proportionate area of the set of S2 cells not contained within the raster cell. The second type of information loss results from the implementation of the majority rule, and is denoted under-coverage (Equation 2), where a raster cell is only partly covered by its associated minimal set of covering S2 cells. The under-coverage is calculated as the proportionate area of the raster cell not contained within the set of S2 cells. These two types of information loss are visualized in Figure 5, with the blue area being over-coverage and the red area being under-coverage.

$$\text{OC} = \frac{|S_2 \setminus \text{Raster}|}{|S_2|} \qquad (1) \qquad\qquad \text{UC} = \frac{|\text{Raster} \setminus S_2|}{|\text{Raster}|} \qquad (2)$$

Since the two contributing factors are equally unwanted, we define the information loss as the harmonic mean of the over- and under-coverage (Equation 3). Due to the nature of the harmonic mean, higher values are desirable; however, this is not the case for the over- and under-coverage calculations. Therefore, we use the complement of each coverage. This yields an information accuracy, which we subtract from 1 to get the loss.

$$\text{Information Loss} = 1 - 2\frac{(1-\text{OC})\cdot(1-\text{UC})}{(1-\text{OC})+(1-\text{UC})}. \qquad (3)$$

The mean information loss for integrating with and without the majority rule is reported in Figure 6 for different S2 levels, based on random sample of 100 raster cells. Low granularity S2 levels have a mean information loss approaching 1, whereas higher granularities of the S2 level reduce the information loss. The advantage of using the majority rule is visible for higher granularities, starting from S2 level 20. However, for low granularity S2 cells, the majority rule introduces high under-coverage, as many raster cells become associated with no S2 cells, due to the centroid of large S2 cells that covers many raster cells being located only within a single raster cell.

The S2 levels 1 through 15 and 27 through 30 are not shown, as these approach an information loss of 1 and 0, respectively. At S2 level 24, where the information loss is 0.022, the information loss begins to stagnate with higher granularity of the S2 levels, which is an indication that this is a suitable level to integrate the use case raster files into.

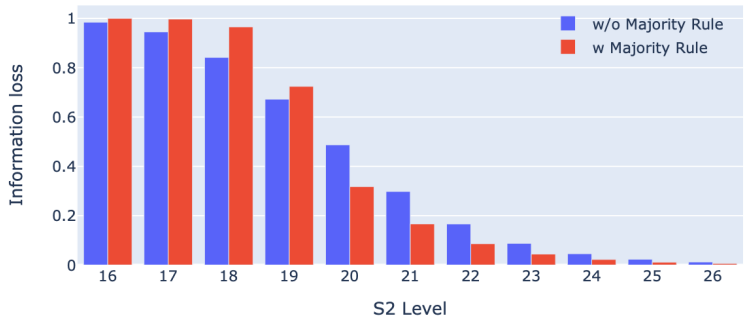


Figure 6: S2 error comparison at different hierarchy levels, with and without majority rule

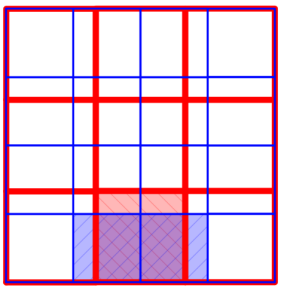


Figure 7: Upsampling and downsampling rasters of different resolutions

Using the S2 Geometry as the spatial layer is not without issues, especially when using a granularity for the S2 cells. The grid of the raster files used for this case study is approximately 35,000 × 45,000, resulting in 1.575 billion raster cells. For an S2 level of 24, each raster cell is linked to 625 S2 cells on average. Combining these two numbers yields a total of approximately 1 trillion triples. However, as the raster files for this case study lack measurements for raster cells located in the ocean, the actual amount of integrated raster cells diminishes by around 75%, yielding approximately 250 billion triples solely for the integration of raster cells.

Up- and Downsampling. To evaluate the gain of using the S2 Geometry as an integration layer, we compare it to using up- and downsampling for combining raster files of different resolutions. Upsampling refers to taking a set of cells at a lower granularity and then aggregating them into higher granularity and vice versa for downsampling. We exemplify the information loss from up- and downsampling via two hypothetical resolutions 10 × 10 and 7.5 × 7.5 meters, in red and blue, respectively, as illustrated in Figure 7.

In order to upsample and downsample, we define how the majority rule works in this setting. For the highlighted red cell, we see that it is covered by four different blue cells. Since only two of the blue cells have their centroids within the red cell, we downsample only into those two cells. Conversely, we upsample the highlighted two blue cells into the single red cell in which their centroids are located. We note that upsampling is more difficult than simply aggregating if we deal with categorical attributes.

An information loss of 0.5 is obtained for upsampling into the 10 × 10 grid and 0.298 for downsampling into the 7.5 × 7.5 grid. In comparison, integrating the two grids into the S2 Geometry at level 24 results in an information loss of 0.028 and 0.019, respectively.

6. Conclusion

In this paper, we have presented an approach for integrating multi-modal heterogeneous data sources through a spatial KG layer in the context of integrating geographically annotated microbial data and environmental features to enable joint analysis. While the emphasis has been on integrating the spatial data sources, we have also discussed the design of the complete multi-modal data integration of raster data, DNA reads, and ontologies. We propose an approach based on the S2 Geometry that can integrate raster files of different resolutions, translations, and rotations without performing significant aggregations of the raster cell measurements. In the future, we plan to work on improving scalability given the large number of RDF triples related to the use of the S2 Geometry as well as to capture provenance through the use of the PROV-O ontology[10].

Appendix E

Acknowledgments

This work is partially funded by the Villum Foundation (DarkScience project, 50093). We further thank the SustainScapes group from Aarhus University for sharing the environmental dataset.

[10] https://www.w3.org/TR/prov-o/

References

[1] M. Cheatham, C. Pesquita, Semantic Data Integration, in: Handbook of Big Data Technologies, Springer, 2017, pp. 263–305.

[2] R. Stahl, P. Staab, Why Is Data Integration So Hard?, in: Measuring the Data Universe, Springer, 2018, pp. 23–34.

[3] J. H. Phan, et al., Integration of multi-modal biomedical data to predict cancer grade and patient survival, in: BHI, IEEE, 2016, pp. 577–580.

[4] X. L. Dong, D. Srivastava, Big Data Integration, Proc. VLDB Endow. 6 (2013) 1188–1189.

[5] K. Timmis, et al., The contribution of microbial biotechnology to sustainable development goals, Microbial biotechnology 10 (2017) 984–987.

[6] V. Bellandi, et al., Toward a General Framework for Multimodal Big Data Analysis, Big Data 10 (2022) 408–424.

[7] X. Zhu, et al., Multi-Modal Knowledge Graph Construction and Application: A Survey, IEEE Trans. Knowl. Data Eng. 36 (2024) 715–735.

[8] M. Leida, A. Gusmini, J. Davies, Semantics-aware data integration for heterogeneous data sources, J. Ambient Intell. Humaniz. Comput. 4 (2013) 471–491.

[9] J. J. Assmann, et al., EcoDes-DK15: High-resolution ecological descriptors of vegetation and terrain derived from Denmark's national airborne laser scanning data set, ESSD 14 (2022) 823–844.

[10] L. C. o. Gomes, Soil assessment in Denmark: Towards soil functional mapping and beyond, Frontiers in Soil Science (2023).

[11] A. Ghosh, M. Simkus, D. Calvanese, Semantic Querying of Integrated Raster and Relational Data: A Virtual Knowledge Graph Approach, in: RuleML+RR (Companion), 2023.

[12] K. Kyzirakos, et al., GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings, J. Web Semant. 52-53 (2018) 16–32.

[13] N. Gür, et al., A foundation for spatial data warehouses on the Semantic Web, Semantic Web 9 (2018) 557–587.

[14] A. B. Andersen, et al., Publishing Danish Agricultural Government Data as Semantic Web Data, in: JIST, 2014, pp. 178–186.

[15] K. Patroumpas, et al., TripleGeo: an ETL Tool for Transforming Geospatial Data into RDF Triples, in: EDBT/ICDT Workshops, volume 1133 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014, pp. 275–278.

[16] B. Tran, et al., Semantic Integration of Raster Data for Earth Observation: An RDF Dataset of Territorial Unit Versions with their Land Cover, ISPRS Int. J. Geo Inf. 9 (2020) 503.

[17] R. Zhu, et al., Environmental Observations in Knowledge Graphs, in: DaMaLOS, 2021, pp. 1–11.

[18] K. Janowicz, et al., Know, Know Where, Knowwheregraph: A Densely Connected, Cross-Domain Knowledge Graph and Geo-Enrichment Service Stack for Applications in Environmental Intelligence, AI Mag. 43 (2022) 30–39.

[19] C. Shimizu, et al., A Pattern for Features on a Hierarchical Spatial Grid, in: IJCKG, ACM, 2021, pp. 108–114.

[20] P. L. Buttigieg, et al., The environment ontology: contextualising biological and biomedical entities, J. Biomed. Semant. 4 (2013) 43.

Introduction to the Appendix

The appendix elaborates, adds perspective, and considers future directions for selected parts of the main paper and, therefore, does not follow an explicit structure linking the individual sections. However, we attempt to preserve a coherent flow throughout the appendix.

In Appendix A, we present the relevant background information that supports the main paper, with a particular focus on the Resource Description Framework (RDF) as the foundation for representing the data in a Knowledge Graph (KG). In Appendix B, we give an in-depth introduction to each data source and how they are related. As the source data does not follow best relational database management practices, we discuss potential solutions to correcting the tables. Different options for representing spatial data are described in Appendix C, where we discuss the advantages and limitations of the individual approaches. The procedures of transforming the different data sources into a KG are described in Appendix D. In Appendix E, the main paper is set in the scope of the project, and we propose directions to expand upon the work as well as new avenues to explore.

An overview of the relation of the sections in the main paper to the sections in the appendix is given in Table 3.

Table 3: Relation between sections of the main paper and sections of the appendix.

Sections of the Main Paper	**Related Appendix**
1. Introduction	
2. Related Work	C. Spatial Representation
3. Data Sources and Use Case Description	B. Data
4. Spatial Integration Approach	D. Pseudocode
5. Evaluation	
6. Conclusion	E. Future Work

A Background

In this section, we give an overview of different relevant background information needed to understand the key concepts used throughout the main paper and the appendix.

A.1 Resource Description Framework

Resource Description Framework (RDF) [21] is a framework designed to represent web-based data. In the RDF framework, each entry (called an RDF statement) is a *triple*, which consists of a *subject*, a *predicate*, and an *object*. A set of RDF statements is called an RDF graph and can be visualized as a directed graph, with source nodes as subjects, edges as predicates, and destination nodes as objects. An RDF graph containing a single RDF statement is depicted in Figure 8.

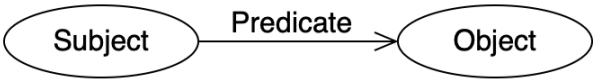


Figure 8: An RDF graph consisting of a single RDF statement [21].

In RDF, the nodes can be either Internationalized Resource Identifiers (IRIs), literals, or blank nodes. The edges in an RDF graph are also IRIs. An IRI is an identifier formed using Unicode characters that uniquely identifies, but does not necessarily provide access to, an online resource. A Uniform Resource Identifier (URI) is a special case of an IRI that only permits ASCII characters in the identifier, and a URL is a special case of a URI that also provides access to the online resource. A literal is a 3-tuple $l = (\nu, \tau, \lambda)$, where ν is a concrete value, such as a number, a string, or a date; τ is an IRI of a resource that identifies the data type of ν; and λ identifies the language of ν if τ is the IRI for the string data type, and is an empty entry otherwise. Blank nodes are also identifiers, but they only identify a resource locally. This means that two blank nodes in two different RDF graphs refer to different resources with near certainty.

A collection of resources with a common prefix, called a namespace IRI, is called an RDF vocabulary. For example, `https://schema.org` is an RDF vocabulary, and provides IRIs for resources, including:

- `https://schema.org/Person` and
- `https://schema.org/accountablePerson`.

The namespace IRI is `https://schema.org/`. Namespace IRIs are often associated with a shorthand version known as a namespace prefix. The namespace prefix associated with `https://schema.org/` is `schema:`, and the IRIs for the resources it describes can then be written as, e.g., `schema:Person`. Note that these IRIs are cases of URLs. For example, the `schema:Person` resource could be used to denote the type of other IRIs, such as illustrated in Figure 9, where it denotes the type of two instances of people, who are represented using their e-mail addresses.

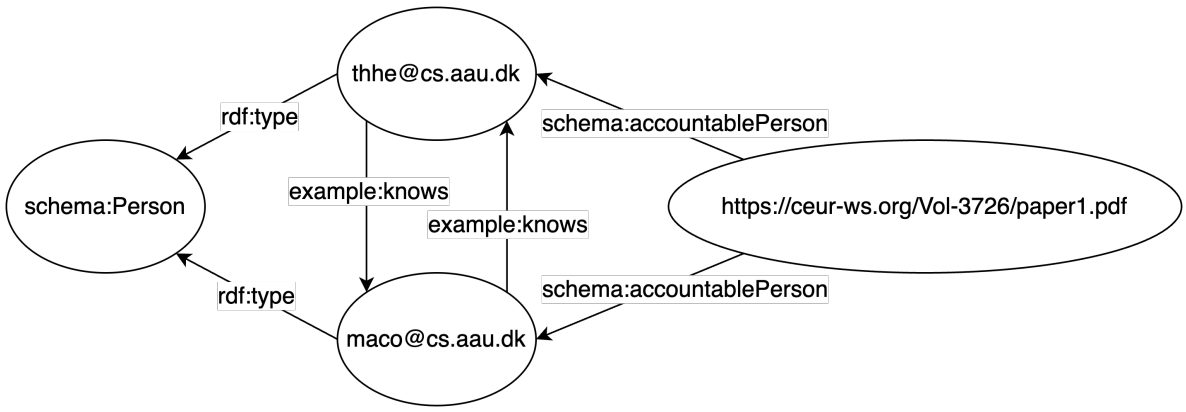


Figure 9: Example of namespace IRIs (`schema`, `example` and `rdf`) used in an RDF graph.

A.1.1 RDF Schema

RDF Schema (RDFS) [22] is an RDF vocabulary and a semantic extension of RDF that offers tools to describe collections of related resources and the relationships connecting them. It is expressed using RDF itself. These descriptions help define attributes of other resources, including the domains and ranges associated with properties. The domain of a property specifies the type of resources the property applies to, i.e., it specifies which class of things can have that property. The range of a property specifies the types of values that a property can take, i.e., it defines the class or datatype of the value the property points to.

In RDFS, classes are used to group resources that share common characteristics. The fundamental class construct is `rdfs:Class`, which is used to declare a resource as a class using the `rdf:type` property. Instances can be assigned to classes, also using the property `rdf:type`. As an example, consider Figure 10, where an IRI representing the specific taxon *Escherichia coli* is typed as an instance of the class *Taxon*, which is itself typed as an instance of *Class*.

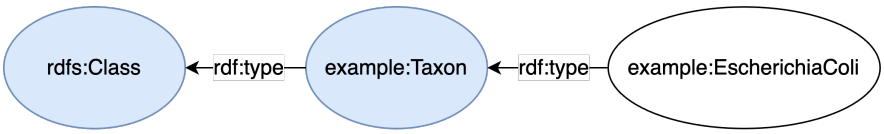


Figure 10: Example of the definition of a class using RDFS. Blue nodes represent classes, and white nodes represent instances.

Properties in RDFS are described using the class `rdf:Property`. These properties link resources to other resources or to literal values. RDFS allows the definition of property characteristics through domain and range. The domain of a property specifies the class of the subject that the property applies to; for example, if a property `example:observedLocation` has a domain `example:Taxon`, then this property can only be used on resources classified as `example:Taxon`. The range of a property defines the class or datatype of the object that the property points to; continuing the example, if `example:observedLocation` has a range `example:Site`, the value of this property must be a resource of type `example:Site`. This is illustrated in Figure 11, which expands upon the example from Figure 10.

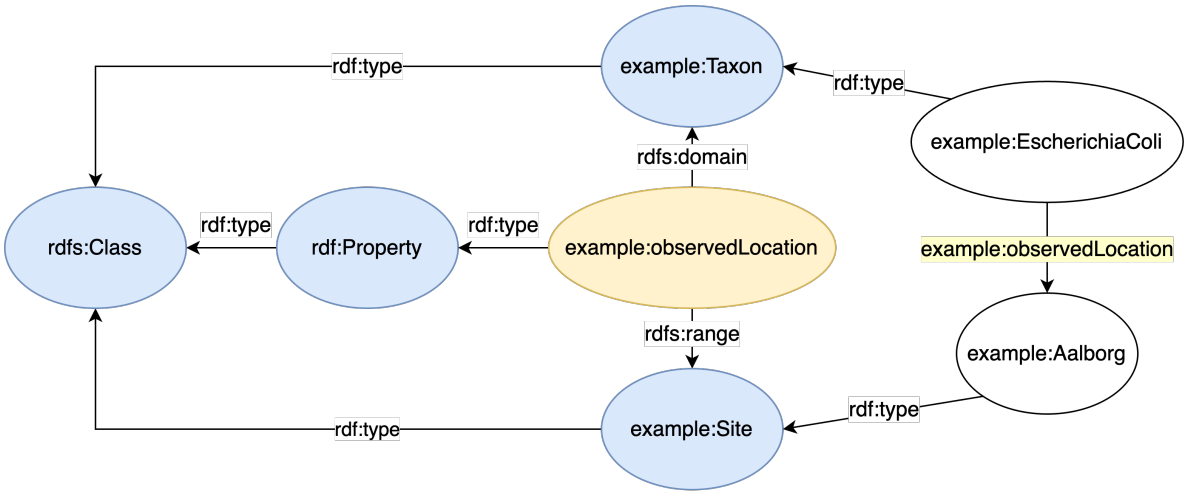


Figure 11: Example of the definition of range and domain for a property using RDFS. Blue nodes represent classes, yellow nodes represent properties, and white nodes represent instances.

Additional constructs in RDFS include `rdfs:subClassOf`, which allows the definition of class hierarchies by indicating that one class is a subclass of another and thus inherits its properties; and `rdfs:subPropertyOf`, which defines hierarchies among properties. Furthermore, `rdfs:label` and `rdfs:comment` are used to provide human-readable names and descriptions for resources. These constructs collectively form the foundation for creating rich ontologies and schemas that enhance the expressiveness and interoperability of RDF data.

A.2 Knowledge Graphs

A **knowledge graph** (KG) is commonly understood as a graph-structured knowledge base whose nodes denote entities (or values) and whose edge labels denote typed relations, and it integrates data and schema-level knowledge [23, 24]. Formally, we follow the RDF-style modelling described in Appendix A.1 and view a KG as a directed, edge-labelled graph $G = (\mathcal{N}, \mathcal{E})$. Here, $\mathcal{N} \subseteq \{\mathcal{I} \cup \mathcal{L} \cup \mathcal{B}\}$ denotes a node as either an IRI $\mathcal{I}$, a literal $\mathcal{L}$, or a blank node $\mathcal{B}$, and $\mathcal{E} \subseteq \mathcal{N}^- \times \mathcal{I} \times \mathcal{N}$ denotes the set of edges, where source nodes $\mathcal{N}^- = \mathcal{N} \setminus \mathcal{L}$ cannot be literals, since subjects in an RDF graph must be IRIs.

It is possible to think about a KG having two interconnected parts. First, there is the part of the KG that deals with the ontology and schema, i.e., classes, properties, and constrains, usually denoted the *terminology box* (TBox); and the part that deals with the instantiations of actual entities, usually denoted the *assertion box* (ABox) [25]. For example, the blue and yellow nodes of Figure 11, and the edges between them, would all be part of the TBox, as they are high-level definitions of classes and constraints, while all triples that has a white node as either the subject or object would be part of the ABox, as they contain information on particular entities, namely that the *Escherichia Coli* has been observed in *Aalborg*, and that *Aalborg* has the class *Site*.

A.3 SPARQL

SPARQL is the W3C standard query language for RDF graphs, designed around *graph pattern matching* [26]. The core building block is the *triple pattern*, which is an RDF triple following specific conditions, e.g., the subject has to be of type `ex:Site`. Sets of triple patterns form *basic graph patterns*, specifying all conditions to be met for the query to return a match. Queries combine basic graph patterns with operators such as `FILTER`, `OPTIONAL`, `UNION`, and `MINUS`, along with projection, ordering, limits, and aggregates.

Below is an example query that finds up to 10 sites and their observed taxa. For each taxon, its English name is included if available. First, the prefixes are defined for readability in the first three lines. Line 5 specifies the variables to be returned: `?site`, `?taxon`, and `?name`. In the `WHERE` clause, it is specified that `?site` must be an instance of the `ex:Site` class (line 7), and must have an edge with label `ex:observedTaxon` to `?taxon` (line 8); and it is also specified that `?taxon` must be an instance of the `ex:Taxon` class (line 9). Within the `OPTIONAL` block from line 10 to line 13, the query attempts to retrieve a human-readable label for the taxon; the `FILTER` retains only English labels (language tag matching "`en`"). If no English label exists, the solution is still returned without `?name`. The result set is limited to at most 10 results by `LIMIT 10` on line 15.

```
1  PREFIX  ex:       <https://example.org/ontology#>
2  PREFIX  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX  rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT ?site ?taxon ?name
6  WHERE {
7    ?site  rdf:type                ex:Site ;
8           ex:observedTaxon  ?taxon .
9    ?taxon rdf:type                ex:Taxon .
10   OPTIONAL {
11     ?taxon rdfs:label ?name .
12     FILTER(LANGMATCHES(LANG(?name), "en"))
13   }
14 }
15 LIMIT 10
```

The result from running the query could be the example shown in Table 4, where only seven results matched the queried basic graph pattern.

Table 4: Example of result from SPARQL query.

?site	?taxon	?name
MfD_1241	OTU_2	Cornyebacterium
MfD_10093	OTU_7259	
MfD_118	OTU_93371	MFD_s_328350
MfD_1241	OTU_4124	MFD_s_7312
MfD_4801	OTU_152092	Planococcus Kocurii
MfD_10142	OTU_14482	MFD_s_501
MfD_18	OTU_531	

A.4 Raster Maps

Raster files represent the world, or a subset thereof, as a regular set of cells in a grid. The cells of a raster are typically evenly spaced squares. A value is attributed to each raster cell and corresponds to a measurement at the particular spatial area. Raster files are either created from images, representing continuous information of an area, or spatial point measurements extrapolated by statistical or machine learning methods to provide a map of estimated measurements over an entire region. This section is based on [27].

The embedded geographic metadata of a raster file includes: (i) the **cell resolution**, which describes the size of each, typically square, raster cell; (ii) the **height and width** of the raster file, denoted as the number of raster cells in either direction; (iii) the **orientation**, which is usually parallel to the coordinate reference system; (iv) the **location of top left cell** to calculate the locations of the remaining cells with the resolution, height, width, and orientation; and (v) the **Coordinate Reference System (CRS)**, to map the raster cells to corresponding locations, e.g., to latitude and longitude coordinates.

The cell resolution can vary across datasets, e.g., some data might be collected at 10×10 meters, whereas others might be collected at 1×1 kilometers. Halving the cell resolution results in four times as many cells, so the cell resolution is often a trade-off between the spatial resolution and data volume, which in turn affects the processing time and storage requirements.

The coordinate reference system can, for example, be the European Terrestrial Reference System 1989 (ETRS89). Each location in the reference system is denoted by the geodesic distance in meters from the equator (northing) and the Greenwich meridian (easting). For example, the location for Aalborg, Denmark, could by ETRS89 be written as (N 6,322,850; E 555,860).

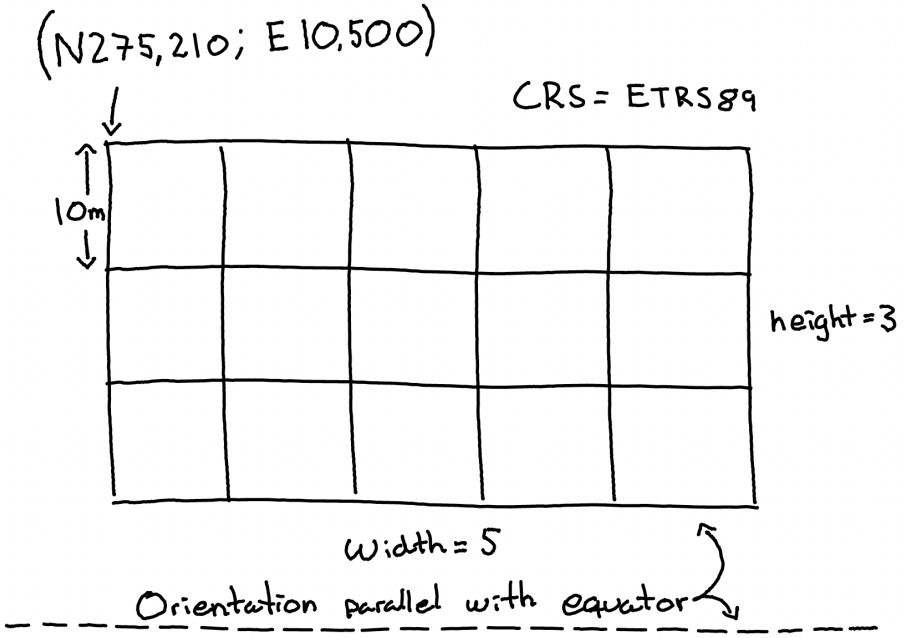


Figure 12: Example of metadata of a simple raster file.

An example of the metadata of a simple raster file is visualized in Figure 12. The example illustrates a cell resolution of 10×10 meters and a height and width of the raster to be 3 and 5, respectively. Also, there is no rotation, as the grid is parallel to the equator. Lastly, the CRS is ETRS89 and the location of the top left cell is (N 275,210; E 10,500).

To calculate the locations of the remaining cells, if no rotation, the equation below can be used:

$$\begin{aligned} x &= x_{\text{origin}} + \text{col} \cdot \text{cell\_resolution} \\ y &= y_{\text{origin}} - \text{row} \cdot \text{cell\_resolution} \end{aligned} \tag{1}$$

Here, x_{origin} corresponds to the location of the top left cell, and the row and col denote the index of the row and column of interest. Thereby, calculations for the location of the bottom right cell would be the following:

$$\begin{aligned} x_5 &= 10{,}500 + 5 \cdot 10 \\ &= 10{,}550 \\ y_3 &= 275{,}210 - 3 \cdot 10 \\ &= 275{,}240 \end{aligned}$$

B Data

In Section 3 of the main paper, we briefly gave an overview of the different data sources we wanted to integrate into a knowledge graph. In this part of the appendix, we reintroduce the Microflora Danica project and explain the data sources in greater detail as well as visualize them.

B.1 The Microflora Danica Dataset

Microflora Danica[1] (MfD) is an ongoing project to create a database of all microorganisms that occur naturally in Denmark. This work is important, as the role of microbial organisms for addressing both present and future challenges within a wide range of topics cannot be overstated. Solving environmental challenges, developing new antibiotics, and the transformation of waste into valuable resources are all possible challenges that microbial organisms can contribute to. The basis for this database is a large number of spatially distributed samples across Denmark from soil, sediment, and water. These samples were then analyzed in a laboratory to determine which microorganisms were found in the different samples. We denote this collection of data sources as the Microflora Danica Dataset. A map of the sample locations is shown in Figure 13.

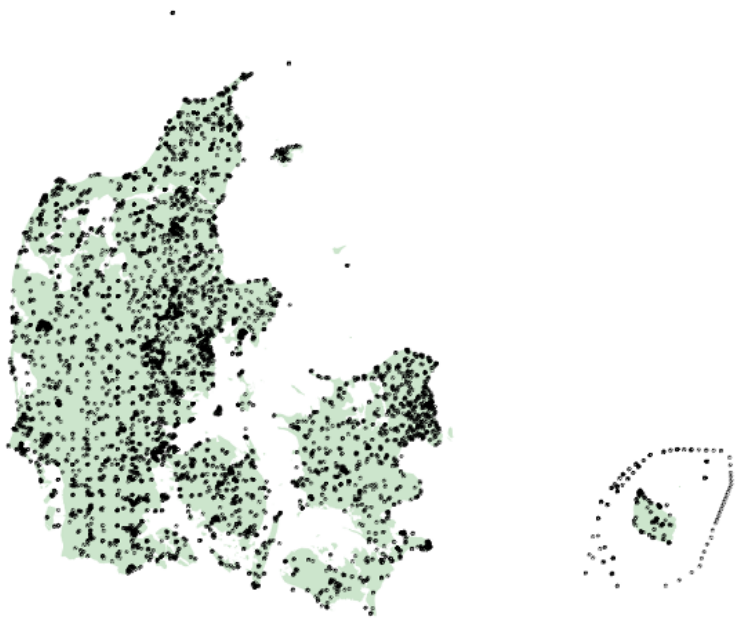


Figure 13: MfD sample locations.

The MfD Dataset consists of multiple, inter-referencing data tables as shown in Figure 14. The **Fieldsample Metadata** contains information on the samples from across Denmark, such as the location and the sample type; the **Habitat Metadata** is a taxonomy for the different habitats of Denmark; **Projects Metadata** contains information on the different projects that were part of collecting all the samples; **Sequence Metadata** is information that is relevant to how each sample was processed in the laboratory; and the **Read Data** contains the reads sequenced from each sample. Each data table, besides the **Read Data**, is stored in a separate, non-normalized .csv file. Each arrow represents a 1-to-n relation and also shows which columns are the foreign keys of the relation. The dotted arrow from **Environmental Raster Data** to **Fieldsample Metadata** represents the integration of external environmental data with the MfD Dataset. We explain each data table in detail in the following sections. The dotted arrow from **Read Data** to **Operational Taxonomic Unit Table** represents the fact the **Read Data** was used as a basis to construct the **Operational Taxonomic Unit Table**.

[1] https://www.en.bio.aau.dk/research/projects/microflora-danica

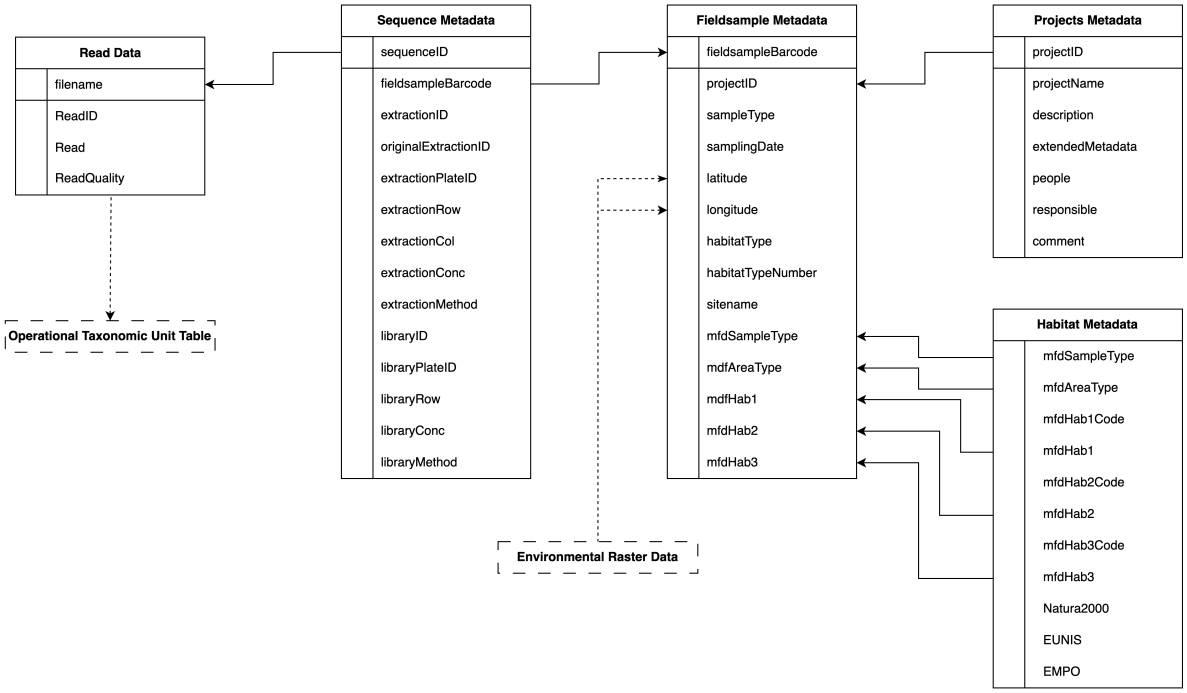


Figure 14: Overview of the different parts of the Microflora Danica dataset. Dashed parts represent non-source data, which are either obtained from external parties or generated through post-processing.

B.1.1 Fieldsample Metadata

The **Fieldsample Metadata** source table contains information on the sample site where a soil sample was taken, including the type of sample, the sampling date, the location, and information about the environmental conditions of the site.

Of particular interest in this data table are the *latitude* and *longitude* columns, as these are the foreign keys on which this table must be joined with the environmental raster files. A complete list and explanation of table columns is given in Table 5.

In Table 6, we provide an example row of the **Fieldsample Metadata** table.

Table 5: Explanation of each column in the **Fieldsample Metadata**. Foreign keys are labelled as FKey, and composite foreign keys are labelled as CFKey.

Column	Key	Description
fieldsampleBarcode	Primary Key	Unique identifer for the sampling site
projectID	FKey → **Projects Metadata**	Identifier for the project that contributed the sample
sampleType		Type of material sampled (soil, sediment, water, etc.)
samplingDate		Date of the sampling
latitude	CFKey → **EnvRasterData**	Latitude of the sampling site
longitude	CFKey → **EnvRasterData**	Longitude of the sampling site
habitatType		Composite of the mfdSampleType and mfdAreaType
habitatTypeNumber		A four-digit code that uniquely encodes for a specific combination of mfdSampleType, *mfdAreaType*, mfdHab1, mfdHab2, and mfdHab3
sitename		Common name for the sampling site
mfdSampleType	CFKey → **Habitat Metadata**	Level 1 (most coarse) of the MfD habitat hierarchy. Approximately corresponds to sampleType
mfdAreaType	CFKey → **Habitat Metadata**	Level 2 of the MfD habitat hierarchy
mfdHab1	CFKey → **Habitat Metadata**	Textual description of level 3 of the MfD habitat hierarchy
mfdHab2	CFKey → **Habitat Metadata**	Textual description of level 4 of the MfD habitat hierarchy
mfdHab3	CFKey → **Habitat Metadata**	Textual description of level 5 of the MfD habitat hierarchy

Table 6: Transposed example of **Fieldsample Metadata** row.

Column	Example Value
fieldsampleBarcode	MFD00001
projectID	P08_1
sampleType	soil
samplingDate	2019-08-29
latitude	55.6771
longitude	9.2778
habitatType	natural_soil
habitatTypeNumber	91E0
mfdAreaType	Natural
sitename	6r, Bindeballe By, Randbøl
mfdSampleType	Soil
mfdHab1	Forests
mfdHab2	Temperate forests
mfdHab3	Alluvial woodland

B.1.2 Habitat Metadata

The **Habitat Metadata** is a taxonomy of the habitats in Denmark. The habitats are divided into five separate levels, with the most coarse level being the *mfdSampleType* column, and the finest level being the *mfdHab3Code* column. The hierarchy can be seen in Figure 15.

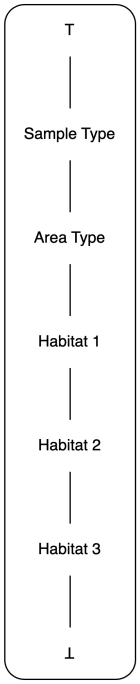


Figure 15: The hierarchy of the **Habitat Metadata** taxonomy

Furthermore, each *habXCode* column is also associated with a textual description of the habitat Moreover, each row of the Habitat Metadata table contains a reference to three external taxonomies, mapping the MfD habitats to *Natura2000*[2], *EUNIS*[3], and *EMPO*[4].

Natura2000 is a term for protected areas across the European Union. In Denmark, specifically, 9% of the landmass og 28% of the sea have been designated as part of Natura2000. The *Natura2000* ontology is a definition of the habitats of these protected areas. *EUNIS* (European Nature Information System) is a classification system for European habitats, maintained by the European Environment Agency. *EMPO* (Earth Microbiome Project Ontology) is an ontology of habitats for use worldwide. It consists of a four-level taxonomy of habitats:

- empo_0: Root level of the EMPO habitat taxonomy
- empo_1: Free-living, Host-associated, Control, or Unknown
- empo_2: Saline, Non-saline, Animal, Plant, or Fungus
- empo_3: The most specific habitat name

The columns of the **Habitat Metadata** are explained in detail in Table 7, and an example row of the **Habitat Metadata** is given in Table 8.

We note that the source data contains some unexpected values. We expected that each entry of the *mfdHabXCode* column would refer to precisely one entry of the *mfdHabX* column, but that was not the case. For example, the code 1110 refers to three distinct habitats - 'Sandbanks', 'Enclosed water', and 'Lillebælt'. It is possible that the value 1110 actually refers to the union of these three habitats, or perhaps there is an error in the data. The statistics for how prevalent this unexpected observation is for each level of the habitat hierarchy, as well as the number of missing values, are summarized in Table 9.

Moreover, the table also has some other issues. First, there are no clear candidate keys. The *mfdHab3Code* column would have been a candidate, if not for the issues discussed above, resulting in each row not being uniquely identifiable using this column. Instead, the entire combination of the habitat hierarchy is currently needed to uniquely identify each row.

[2] https://sgavmst.dk/natur-og-jagt/naturindsatser/natura-2000
[3] https://eunis.eea.europa.eu
[4] https://earthmicrobiome.org/protocols-and-standards/empo/

Table 7: Explanation of each column in the **Habitat Metadata**. Composite primary keys are labelled as CPrimary Key, foreign keys are labelled as FKey, and composite foreign keys are labelled as CFKey.

Column	Key	Description
mfdSampleType	CPrimary Key CFKey → **Fieldsample Metadata**	Level 1 of the MfD habitat hierarchy
mfdAreaType	CPrimary Key CFKey → **Fieldsample Metadata**	Level 2 of the MfD habitat hierarchy
mfdHab1Code	CPrimary Key	Level 3 of the MfD habitat hierarchy
mfdHab1	CFKey → **Fieldsample Metadata**	Textual description of level 3 of the MfD habitat hierarchy
mfdHab2Code	CPrimary Key	Level 4 of the MfD habitat hierarchy
mfdHab2	CFKey → **Fieldsample Metadata**	Textual description of level 4 of the MfD habitat hierarchy
mfdHab3Code	CPrimary Key	Level 5 of the MfD habitat hierarchy
mfdHab3	CFKey → **Fieldsample Metadata**	Textual description of level 5 of the MfD habitat hierarchy
Natura2000	FKey → Natura2000	Foreign key linking the Microflora Danica habitat to the external ontology Natura2000
EUNIS	FKey → EUNIS	Foreign key linking the Microflora Danica habitat to the external ontology EUNIS
EMPO	FKey → EMPO	Foreign key linking the Microflora Danica habitat to the external ontology EMPO

Table 8: Transposed example of the **Habitat Metadata** source table.

Column	Example Value
mfdSampleType	Soil
mfdAreaType	Natural
mfdHab1Code	2000
mfdHab1	Dunes
mfdHab2Code	2100
mfdHab2	Sea dunes
mfdHab3code	2130
mfdHab3	Fixed dunes (grey dunes)
Natura2000	2130
EUNIS	N15
EMPO	Free-living; Non-saline; Soil (non-saline)

Table 9: Observation statistics for **Habitat Metadata**

Metric	Value
Missing Values	
hab1	0 (0.00%)
hab2	33 (11.8%)
hab3	97 (34.8%)
Non-unique Codes	
hab1	6 (2.20%)
hab2	9 (3.23%)
hab3	14 (5.02%)

B.1.3 Sequence Metadata

The **Sequence Metadata** source table contains information on the sequencing process that each sample was exposed to in the laboratory, such as sequencing method, in-laboratory identifiers, and used solution concentrations.

Note that one fieldsample could have been sequenced multiple times, such that multiple primary keys (*sequenceID*) could refer to the same foreign key (*fieldsampleBarcode*), making the relation between this table and **Fieldsample Metadata** 1-to-n; however, the majority of entities have unique foreign key entries.

Table 10: Explanation of each column in the **Sequence Metadata** Foreign keys are labelled as FKey.

Column	Key	Description
sequenceID	Primary Key FKey → **Read Data**	Unique identifier for the sequencing
fieldsampleBarcode	FKey → **Fieldsample Metadata**	Identifier for the sampling site
extractionID		Identifier for the extract containing the fieldsample
originalExtractionID		Some fieldsamples were extracted multiple times, so this column links new extractions to old ones
extractionPlateID		Identifier for the extraction plate
extractionRow		Identifier for the extraction row. Included as the last letter in the extractionID
extractionCol		Identifier for the extraction column. Included as the last number in the extractionID
extractionConc		The concentration of the extraction
extractionMethod		Identifier for the extraction method
libraryID		Identifier for the library used for sequencing
libraryPlateID		Identifier for the libary plate
libraryRow		Identifier for the library row. Included as the last letter in the sequenceID
libraryConc		The concentration of the library
libraryMethod		The library method used for sequencing

An example row of the **Sequence Metadata** is given in Table 11. To reiterate, the

information in this table is solely for experts to see the provenance of the sequencing of the different soil samples.

Table 11: Transposed example of the **Sequence Metadata** source table.

Column	Example Value
fieldsample_barcode	MFD00009
extraction_id	EXT-MJ019-A1
original_extraction_id	EXT-MJ019-A1
extractionplate_id	EXT00001
extraction_row	A
extraction_col	1
extraction_conc	55.67
extraction_method	PowerSoil-Pro-HT
library_id	LIB-MJ050-A1
libraryplate_id	LIB00001
library_row	A
library_conc	5.87675
library_method	ILLUMINA-DNA-PREP.V1
seq_id	LIB-MJ050-A1_02

B.1.4 Read Data

The **Read Data** consists of read files of the .fastq format[5], which is the output format of the utilized sequencing method, Illumina. Each file stores multiple reads, R_i, which are sequences of the four DNA bases Adenine (A), Thymine (T), Cytosine (C), and Guanine (G), $R_i \in \{A, T, C, G, N\}^+$, where N is used if the base was not able to be determined in the sequencing. Aside from the reads themselves, a read file also contains a read ID and a base-specific quality for each read. The .fastq format is a plain-text file that is newline-delimited, such that each data entry takes up exactly four lines. The first line of a read file is the read ID, the second is the read itself, the third is a '+' sign used as a separator between the read itself and the base quality, and the fourth is the base quality. An example of a read file is in Figure 16.

```
1   @A00595:256:H5NHYDSX5:4:1101:4770:1016 1:N:0:TGCAAGATAA+NCGTCCTCAA
2   ANATGATCGCGCGCTGGCGCGAAGGCTTCGATATCGTCCACGCCAAGCGCTGCGAACGCGACGGCG...
3   +
4   F#FFFFFFFFFFFFFFFFFFF:FFF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...
5   @A00595:256:H5NHYDSX5:4:1101:6253:1016 1:N:0:TGCAAGATAA+NCGTCCTCAA
6   CNTCGGTCTCGGTGCGATGCAGATCCTGGCGCTGCTGCCGGGCATCAGCCGGGACGGCATCGTGAT...
7   +
8   F#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...
9   @A00595:256:H5NHYDSX5:4:1101:11134:1016 1:N:0:TGCAAGATAA+NCGTCCTCAA
10  ANGACGGCCAGGCCCGCGGGGTGGTGCTGGAGAACGGCGACGAGATCCGCGCCCAGGTGATCGTGT...
11  +
12  F#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...
```

Figure 16: A snippet of a read file showing three data entries. Note that the reads and base qualities have been truncated for readability.

[5] knowledge.illumina.com/software/general/software-general-reference_material-list

The quality of a base is an encoding of a quality score $s \in \mathbb{N}_{[0;93]}$ as an ASCII character between ASCII(33) and ASCII(126). Formally, the encoding is given as

$$Q(s) = \text{ASCII}(s + 33).$$

For example, for the quality score $s = 37$, the corresponding quality in the .fastq file is F, since $Q(37) = \text{ASCII}(37 + 33) = \text{ASCII}(70) = \text{F}$.

B.1.5 Projects Metadata

The soil samples were collected as part of a collaborative effort involving multiple projects across different periods. The **Projects Metadata** source table provides information on each of these contributing projects. A complete list and explanation of table columns is shown below:

Column	Description
projectID	Unique identifier for the project
projectName	Common name of the project
description	Description of the project
extendedMetadata	Freeform extra information when available
people	Name and e-mail of research partners
responsible	Name and e-mail of the person responsible for the project
comment	Freeform comment to the project

An example hereof is shown in Table 12. Notably, due to the *people* and *responsible* columns, the table does not follow the 1NF as these do not contain only atomic values. For example, the *people* column contains two people. To solve the improper database design, one would make another table for `people` with columns *projectID* and e-mail of *people*. These would form a composite primary key. A similar table for `responsible` would be made, following the same design. Separately, a third relational table should be created with the e-mail as a primary key and the name as a column, which is linked to both the `people` and the `responsible` tables through foreign keys.

Table 12: Transposed example of **Projects Metadata** source table.

Column	Example Value
projectID	P04_1
projectName	Agriculture - Potato
description	Samples from potato fields looking at potato diseases...
extendedMetadata	*None*
people	John Doe <john@bio.aau.dk>; Jane Doe <jane@bio.aau.dk>
responsible	Jane Doe <jane@bio.aau.dk>
comment	Double check methods for inclusion in MfD.

B.2 Operational Taxonomic Unit Table

The operational taxonomic unit (OTU) table is derived from the **Read Data** and thereby is not part of the source data for the MfD project. An OTU table records which species are present at the different sampling sites based on the reads returned from the sequencing. In the table, the structure of which is shown in Table 13, the first column is *OTU*. Each OTU is an ID of a DNA sequence that corresponds to a microorganism (a *taxon*). Each *MFD_x* column corresponds to one fieldsampleBarcode from the **Fieldsample Metadata** data table, and the entries in these columns are a count of how many occurrences of each OTU were observed in that sampling site. Note that these columns are quite sparse, with 99.51% empty records, as not many microbial taxa are present everywhere. The *Kingdom*, *Phylum*, *Class*, *Order*, *Family*, *Genus*, *Species* columns define the taxonomic hierarchy that the taxon (OTU) corresponds to. The hierarchy is shown in Figure 17. Following the hierarchy in Figure 17, humans have the following classification: *Animalia* (Kingdom), *Chordata* (Phylum), *Mammalia* (Class), *Primates* (Order), *Hominidae* (Family), *Homo* (Genus), *Homo sapiens* (Species).

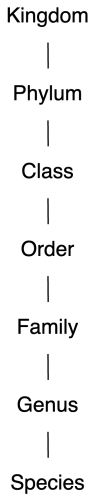


Figure 17: The taxonomy of organisms.

Table 13: The structure of the OTU table

OTU	**MFD_1**	**MFD_2**	⋯	**Kingdom**	⋯	**Species**
OTU_1	0	7	⋯	Archaea	⋯	MFD_s_17257
OTU_2	633	482	⋯	Bacteria	⋯	Cornyebacterium
⋮	⋮	⋮	⋱	⋮	⋱	⋮

The `OTU` table does not follow the normal forms of database design, as the sample sites are stored in a wide format with a column for each separate site. Instead, to adhere to the normal forms, the `OTU` table should first be split into two subtables, one for the *counts* and one for the taxonomy. The first subtable should contain one column for the *OTU* (as it does), one for the *fieldsampleBarcode*, and one for the *count*, as shown in Table 14. The primary key would be the combination of *OTU* and *fieldsampleBarcode*, and *fieldsampleBarcode* would be a foreign key to the **Fieldsample Metadata** and **Sequence Metadata** source tables. Furthermore, any *count* of 0 does not need to be included in the new table.

Intuitively, the taxonomic hierarchy does not follow the second normal form, as the children are dependent on their parents. To normalize it, we would create a table for each level of the hierarchy, each with three columns *(parent_id, child_id, child_label)*, resulting in six tables. The *child_id* column would act as the primary key in these tables, and the *parent_id* column would act as a foreign key to the next-courser level in the hierarchy. A seventh table, for the coarsest level of the hierarchy *Kingdom* would have two columns: *kingdom_id* (primary key) and *kingdom_label*.

To elaborate on Table 13, we report some statistics for the data in Figure 18 and Table 15. Figure 18a shows the number of taxa for each site, summarized in a violin plot.

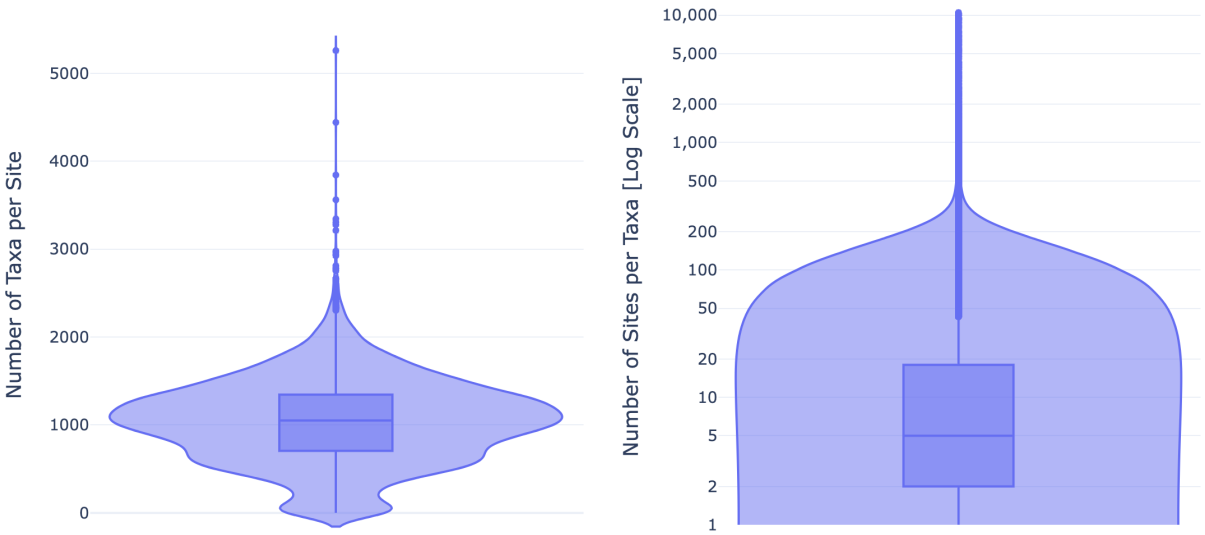
Table 14: Revised structure of *counts* part of the OTU table. No entry of the composite primary key *OTU* and *fieldsampleBarcode* means that the OTU was not observed at the site, resulting in a count of 0.

OTU	**fieldsampleBarcode**	**count**
OTU_1	MFD_2	7
⋮	⋮	⋮
OTU_2	MFD_1	633
OTU_2	MFD_2	482
⋮	⋮	⋮

Looking into the number of unique taxa per site, the 25th percentile is 705, the median is 1,051, the 75th percentile is 1,345, and the average is 1,038, which provides information on how rich the biodiversity is at each sampling site. In total, 10,609 sites are included in the OTU table with 211,517 different taxa. On average, 50% of the taxa are included in 10% or fewer sites. Only a few sites have a high prevalence of taxa, with the three highest having 5,259, 4,442, and 3,843 different taxa, which is all less than 2.5% of all observed taxa.

Figure 18b illustrates the number of sites per taxa on a logarithmic scale. The majority of taxa are present in only a small number of sites. This is further illustrated by the 25th percentile, median, and 75th percentile values, which are 2, 5, and 18 sites, respectively. The average is 52 sites per taxa; however, this is highly influenced by some taxa being present in more than half the sites. For example, only 268 (0.13%) of the taxa are present in more than half the sites, and only 3,221 (1.52%) taxa are present in more than 5% of sites. In contrast, 136,624 (64.49%) taxa occur in fewer than 10 sites with 41,530 (19.63% of total) taxa only occurring once.

Although the number of taxa per site is fairly evenly distributed, the sites per taxa is very skewed.



(a) Number of taxa per site.

(b) Number of sites per taxa in log scale.

Figure 18: Violin plots for the OTU table.

The taxonomic hierarchy is shown in the last seven columns of Table 13, where each taxon is mapped to the corresponding level in the hierarchy for which it is possible – some taxa cannot be mapped or assigned a new rank as they are not close enough in comparison to the reference database taxa or different enough to be a new taxon. Table 15 summarizes the unique taxa for each hierarchical rank and the number of unknown taxa. For example, within the *Kingdom* rank, we observe 3 different taxa and they are all mapped, whereas for the species, we observe 175,732 unique taxa with 35,786 unable to be mapped.

Table 15: Taxon statistics for OTU table.

Taxon Metric	Kingdom	Phylum	Class	Order	Family	Genus	Species
Unique taxa	3	94	368	1,332	5,253	39,101	175,732
No taxa	0	3	93	407	1,508	5,748	35,786

B.3 Environmental Raster Files

The 17 available raster files include information such as pH, temperature, elevation, and salinity, each at a 10×10 meter resolution. They cover the entirety of Denmark, resulting in a width of approximately 45,000 raster cells and a height of approximately 35,000 raster cells. In total, this results in roughly 1.575 billion raster cells. As the measurements are only made inland in Denmark, all other raster cells can be ignored, and effectively, only about 400 million of raster cells include information. An example of a raster file is shown in Figure 19, measuring the annual mean temperature of locations in Denmark.

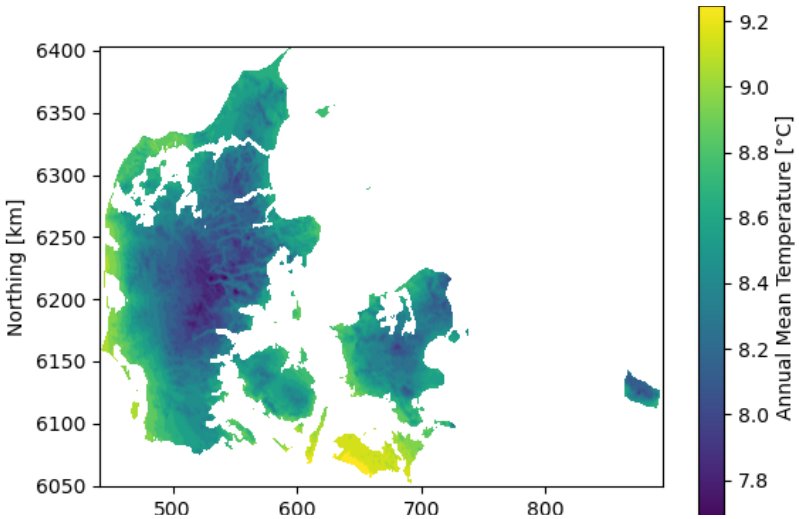


Figure 19: Example of a raster file measuring annual mean temperature.

Some of the raster files are created from LiDAR images, which reduces the uncertainty, as individual raster cell values are not estimated. However, other raster files are extrapolated based on spatial point measurements, adding uncertainty to the value of the raster cells that lie far from the point observations. An example of a LiDAR image is the elevation raster file, whereas the pH raster file is created by extrapolation methods. The raster files come from external parties and are therefore not part of the source MfD data.

The spatial integration deals with linking each raster cell with the spatial point locations of the MfD samples. We describe ways to represent spatial information in Appendix C.

C Spatial Representation

In this section, we explore various approaches to spatial data representation, outlining the advantages and disadvantages of each. Specifically, we introduce and examine the capabilities of various Hierarchical Discrete Global Grids (HDGGs) [28] and GeoSPARQL [29], assessing if they are applicable for our spatial knowledge representation.

C.1 Hierarchical Discrete Global Grids

In 2. Related Work of our main paper, we briefly described three hierarchical discrete global grids (HDGGs) for spatial representational mapping. In this section, we describe the three grids: H3[6], Bing Maps Tile System[7], and S2 Geometry[8], while elaborating their strengths and weaknesses as well as our reasoning for using the S2 Geometry.

C.1.1 H3

The hexagonal hierarchical spatial index, H3, by Uber, divides the Earth into hexagons covering the whole world. H3 has 16 different resolution levels, dividing the Earth from 122 cells down to approximately 570 trillion cells. At the lowest possible granularity, each cell is 0.9 m^2.

A key advantage of using hexagons is their ability to wrap around the Earth without distorting individual cells, besides the slight curvature to avoid sharp angles. This ensures that all cells at a given resolution remain the same size. Most of the distortion comes from the fact that the Earth is not a perfect sphere, but the projection loss is minimal. The H3 grid is illustrated in Figure 20.

However, one drawback of the H3 hexagonal grid system is the parent/child mismatch that arises during resolution changes. Because H3 uses a hierarchical structure, where each hexagon is subdivided into seven smaller hexagons, the resulting children do not perfectly nest within their parent. This leads to uneven spatial representation and potential inconsistencies when aggregating or downsampling data across resolutions, especially in edge regions.

Integrating microbial data, where small changes may have huge implications, such aggregation errors can potentially misplace the location of microbial species or the environmental features of the sampling site.

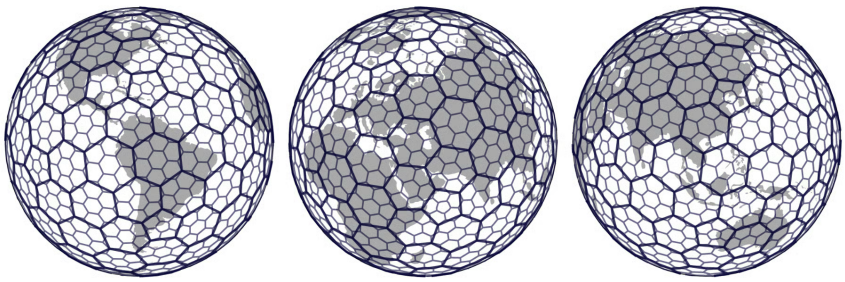


Figure 20: Visualization of H3[6].

[6] https://www.uber.com/en-SE/blog/h3/
[7] https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system
[8] https://s2geometry.io

C.1.2 Bing Maps Tile System

The Bing Maps Tile System[7] is based on the Mercator projection, dividing Earth into a 2-dimensional map with $256 \cdot 2^1 \times 256 \cdot 2^1$ cells at the top level and $256 \cdot 2^{23} \times 256 \cdot 2^{23}$ cells at the lowest granularity at level 23. With the cells being squares, the topology is simple, and each parent cell perfectly contains its child cells.

However, the Mercator projection comes with the cost of significant distortion, especially near the poles. The projection stretches the map vertically as latitude increases, causing cells near the poles to appear much larger than those near the equator, despite representing much smaller areas on the Earth's surface. This projection loss leads to poor spatial uniformity and can introduce bias in spatial analyses, particularly for global-scale applications.

C.1.3 S2 Geometry

The S2 Geometry by Google divides the Earth into a cube, which is then rounded to mimic a sphere. The six faces of the cube can each be perfectly divided into four child cells for a total of 31 levels of resolution. At the finest resolution, each cell corresponds to approximately 0.7 cm^2. Because the S2 Geometry projects Earth onto the faces of a cube, some distortion occurs: child cells near the center of each face are slightly larger to accommodate the Earth's curvature. Although this distortion is less severe than that of the Mercator projection, the resulting cells are not equal in size, which may introduce additional noise. An illustration of the S2 Geometry is shown in Figure 21, where, looking at the green grid cells, the distortion is visible by the fact that the grid cells are not of equal size.

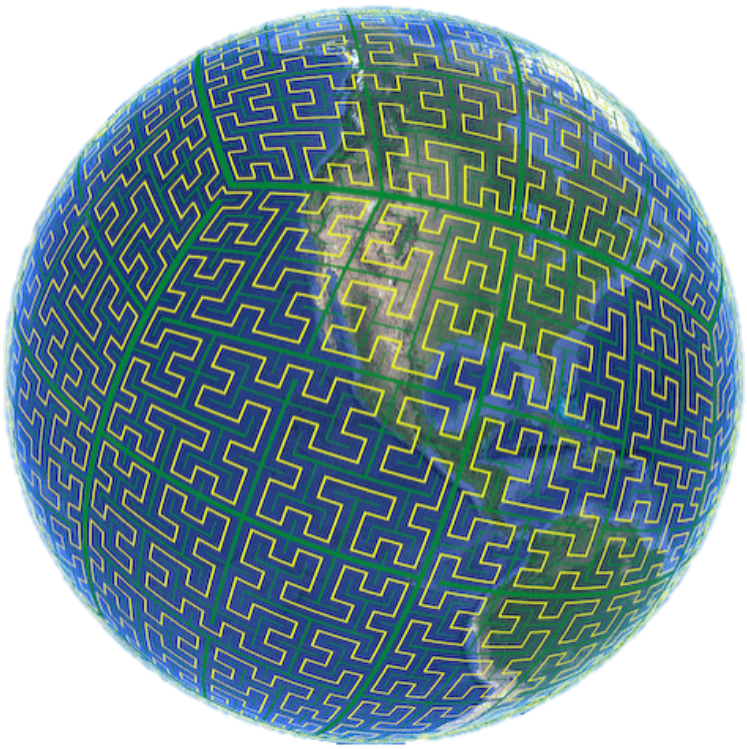


Figure 21: Visualization of the S2 Geometry[8].

The S2 Geometry uses a 64-bit cell ID for each cell, where the hierarchy is inherent. The cell ID is given by the face as a 3-bit number, followed by each child-level given by a 2-bit number, with the final bit after the last child always being 1. Assume that the granularity is k, then the S2 ID is given as $[face][child]^k[10^{60-2k}]$. For example, if we want to find the ID of the first child of the second face cell, then the bit number for the child is **01**, and the bit number for the face is 010, yielding an ID of 010 **01** 100...0. This is illustrated in Figure 22.

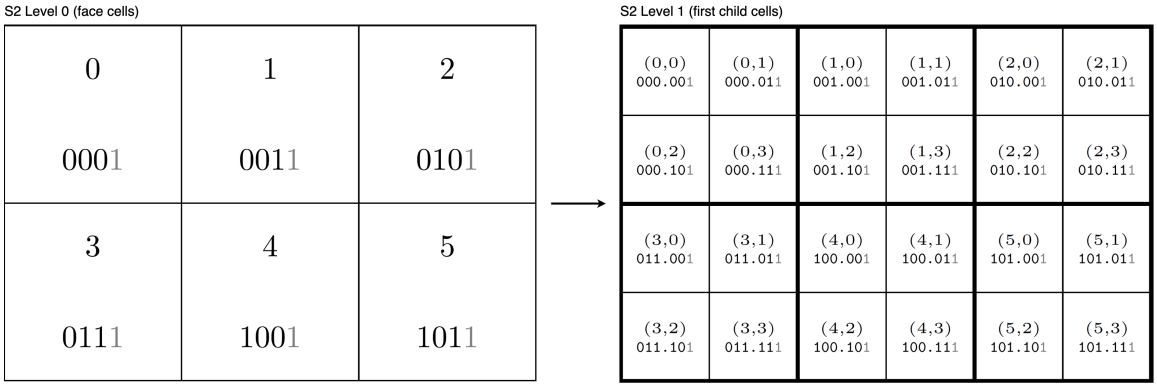


Figure 22: Visualization of the 64-bit ID of the S2 Geometry. The trailing 1 is colored gray to distinguish it from the rest of the ID. Trailing zeroes are pruned for readability.

C.1.4 Comparison

Each of the HDGGs comes with advantages and limitations. Table 16 shows an overview of how each grid corresponds to the following metrics: cell type, resolution levels, lowest granularity, distortion, and topological guarantees. Some are not directly comparable in the sense that one is better than another, whereas others can be ordered. For example, a hexagonal shape is not necessarily better or worse than a square shape, whereas low distortion is better than high, and no distortion is preferred.

	H3	**Bing**	**S2**
Cell Type	Hexagon	Square	Quadrilateral
Resolution Levels	16	23	31
Lowest Granularity	0.9 m^2	Differs	0.7 cm^2
Distortion	Negligible	High	Moderate
Topological Guarantees	Parent/Child Loss	Perfect	Perfect

Table 16: Comparison of spatial hierarchical indexing schemes.

The lowest granularity for Bing does not make sense to calculate or report, as, for the lowest resolution level, the area of cells near the poles has a vastly different area than those near the equator. Thereby, the distortion is high for Bing.

In summary, while both H3 and S2 have their merits, we ultimately preferred the perfect topological guarantees offered by S2 as we found the moderate distortion to be negligible. Moreover, the fine granularity made it possible to reduce the loss moving from raster files to the S2 Geometry even less. H3 did not offer topological guarantees as the parent cells did not fully cover the child cells; as such, S2 was the ideal approach for our use case. Bing was excluded from consideration due to its high distortion.

C.2 GeoSPARQL

When working with Knowledge Graphs (KGs) and spatial data, GeoSPARQL [29] is a geospatial extension to SPARQL, standardized by the Open Geospatial Consortium (OGC). GeoSPARQL also defines an ontology for describing spatial features, geometries, relationships, and coordinate reference systems. Additionally, GeoSPARQL introduces spatial functions to query geometries for operations such as distances, intersections, and containment.

Despite GeoSPARQL's strengths, such as enabling geospatial queries within RDF and supporting advanced spatial reasoning like identifying rivers that cross forests, it poses challenges for large-scale applications. The queries of GeoSPARQL are computationally complex, not scaling well with the increase of spatial components. Thereby, it effectively makes GeoSPARQL impractical for large-scale applications [30].

For our use case, we are looking at a grid of approximately 35,000 $\times$ 45,000 cells, corresponding to 1.575 billion cells, that each would need to be represented by a polygon within the GeoSPARQL framework. However, due to 75% of the raster cells covering the sea, where no measurements are available, the effective number of raster cells is closer to 400 million. This impracticality is further validated by [31], stating that storing raster files in RDF is reasonable, but may not be suitable for queries, as they may simply time out.

D Pseudocode

In this section, we describe the general implementation of converting our data, described in Appendix B, to RDF triples. We use the Python libraries RDFLib[9] and BabelGrid[10] for constructing our RDF triple files and mapping raster cells to the S2 geometry, respectively. For background on RDF, please read Appendix A.1.

D.1 Metadata to RDF

The process of converting the MfD dataset to RDF is very similar for the different metadata source tables described in Appendix B.1. Therefore, we only describe the process for converting the **Projects Metadata** source table.

First, we identify how the columns are related using the normalized projects metadata table, as outlined in Appendix B.1.5. We see that all columns except the *names* and *emails* are related to the *projectID* and therefore we create triples with the *projectID* as the subject. For the *names* columns, the email is the primary key, and therefore becomes the subject for the remaining triples. The resulting triples are shown in Table 17.

Table 17: Links for **Projects Metadata** source table.

Subject	**Predicate**	**Object**
projectID	rdfs:label	*projectName*
projectID	schema:description	*description*
projectID	mfd:extendedDescription	*extendedMetadata*
projectID	schema:identifier	*projectID*
projectID	prov:wasAttributedTo	*people.email*
people.email	schema:givenName	*people.firstname*
people.email	schema:familyName	*people.familyname*
projectID	schema:accountablePerson	*responsible.email*
responsible.email	schema:givenName	*responsible.firstname*
responsible.email	schema:familyName	*responsible.familyname*
projectID	schema:comment	*comment*

The pseudocode implementation for transforming the **Projects Metadata** source table, without prefixes and class definitions, into RDF triples, is shown in Algorithm 1. Most notably, we need to normalize the column of *people* and *responsible*. We do this in lines 9-11 and 13-15 for *people* and *responsible*, respectively. A person involved in the project can also be responsible for the project. In this case, two triples with different predicates from the projectID to the e-mail are created, one with *wasAttributedTo* and another with *responsible*.

[9] https://rdflib.readthedocs.io/en/stable/
[10] https://github.com/EL-BID/BabelGrid

Algorithm 1 Project Metadata to RDF

```
G ← Graph()
for each row in projects_table do
    projectID ← row["projectID"]
    G.write(projectID label row["projectName"])
    G.write(projectID description row["description"])
    G.write(projectID extendedDescription row["extendedMetadata"])
    G.write(projectID comment row["comment"])
    for each person in row["people"] do
        G.write(projectID wasAttributedTo person.email)
        G.write(person.email givenName person.firstname)
        G.write(person.email familyName person.familyname)
    for each responsible in row["responsible"] do
        G.write(projectID accountablePerson responsible.email)
        G.write(responsible.email givenName responsible.firstname)
        G.write(responsible.email familyName responsible.familyname)
```

D.2 Operational Taxonomic Units to RDF

The Operational Taxonomic Unit table illustrated in Table 13 does not adhere to best practices for relational database design, as discussed in Appendix B.2. To reiterate, the entries of *fieldsampleBarcode* appear as columns, while the last seven columns redundantly store taxonomic information.

When transforming each row to RDF, we consider each *fieldsampleBarcode* with the presence of a species, i.e., all non-zero entries, and make a triple for the corresponding OTU. Additionally, we create a link from each OTU to the corresponding species. For the remaining taxonomic tree, we create a link from the child to the parent until the root, i.e., *Kingdom*. Following this logic, we obtain the below triple file when transforming the first line of Table 13. Additionally, the corresponding RDF graph is shown in Figure 23.

```
mfd:MFD_2
    rdf:type mfd:Sample ;
    oboe:hasMeasurement _:measurementA .

_:measurementA
    rdf:type oboe:Measurement ;
    oboe:hasValue 7 ;
    oboe:containsMeasurementsOfType mfd:OTU_1 .

mfd:OTU_1
    rdf:type mfd:OTU ;
    owl:sameAs mfd:MFD_s_17257 .

mfd:MFD_s_17257
    rdf:type mfd:Species ;
    rdfs:subClassOf mfd:some_level .

mfd:some_level
```

```
19      ...
20      rdfs:subClassOf  mfd:Archaea  .
21
22  mfd:Archaea
23      rdf:type  mfd:Kingdom  .
```

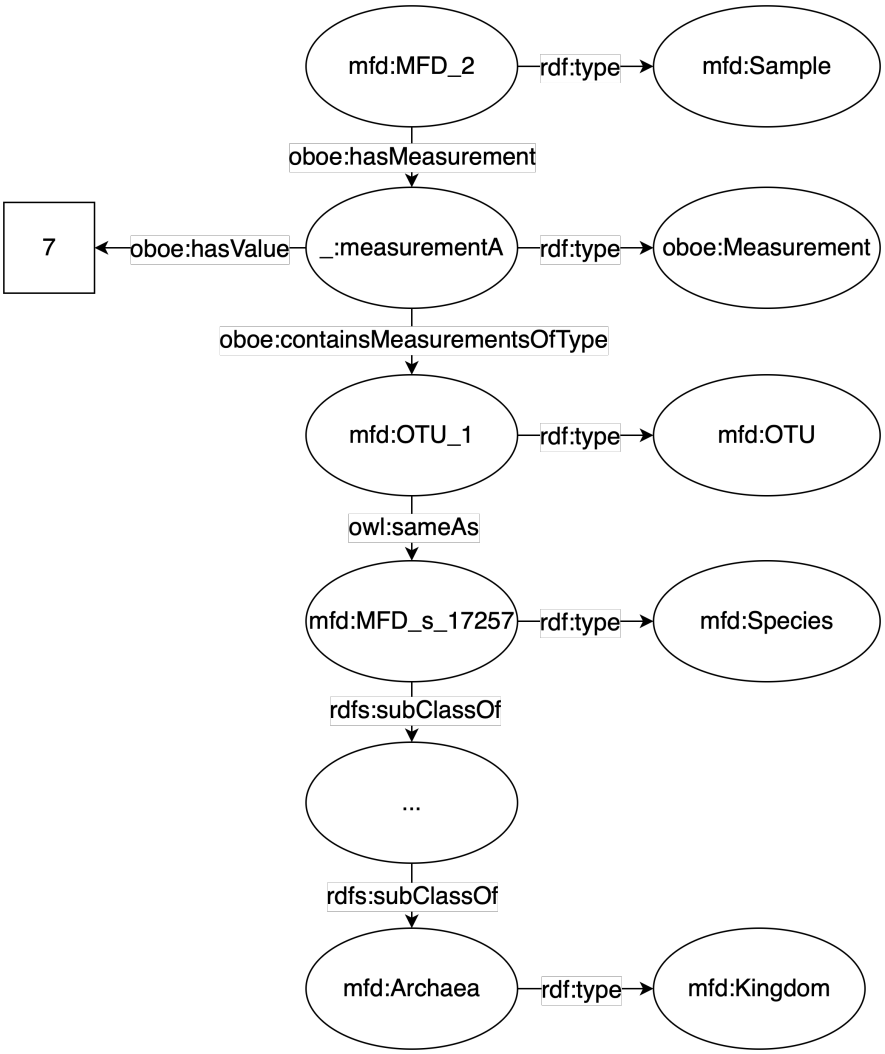


Figure 23: First line of Table 13 converted into RDF. The taxonomic hierarchy is shown through the some_level node.

We obtain the RDF files through the pseudo-implementation provided in Algorithm 2.

The prefixes are omitted for ease of interpretability, yet they are described in Main Paper Figure 4. For each row in the **OTU** table, corresponding to a single OTU, we create links between the fieldsample and measurementID in line 7, links between the measurementID and the measured value in line 8, and links between the measurementID and the OTU in line 9. Additionally, we create a link from the OTU to the corresponding species in line 10. Lastly, we write the hierarchy in the while loop in line 12.

Algorithm 2 Transform OTU Table Row to RDF

```
1: G ← Graph()
2: for each row in otu_table do
3:     for each sample in row do
4:         if sample is observed then
5:             measurementID ← blankNode()
6:             G.write(sample type Sample)
7:             G.write(sample hasMeasurement measurementID)
8:             G.write(measurementID hasValue row[sample])
9:             G.write(measurementID containsMeasurementsOfType row[OTU])
10:            G.write(row[OTU] sameAs row[species])
11:    while not at root do
12:        G.write(child subClassOf parent)
13:        G.write(child type get_taxon_level_of(child))
```

D.3 Raster Files to RDF

Converting the raster files to RDF is simple, yet computationally expensive. For each of the approximately 400 million raster cells, we need to retrieve the corresponding S2 cells at level 24, as decided in the evaluation from Figure 6. We parallelize the task of retrieving the S2 cells to speed up our implementation. The pseudocode implementation is provided in Algorithm 3.

Algorithm 3 Convert Raster Cells to RDF

```
1: G ← Graph()
2: for each rasterCell in parallel do
3:     measurementID ← blankNode()
4:     G.write(rasterCell.ID hasMeasurement measurementID)
5:     G.write(measurementID hasValue rasterCell.value)
6:     G.write(measurementID type Measurement)
7:     s2Cells <- calculateCoveredS2Cells(rasterCell)
8:     for each s2Cell in s2Cells do
9:         G.write(rasterCell.ID covers s2Cell)
10:        G.write(s2Cell type S2Cell)
```

For each unique raster cell, we generate a measurementID in line 3. Identical raster cells, but from different raster files, will thereby have different measurementIDs for each of the different measurements as written through lines 4-6. Finally, we calculate the covering S2 cells of the raster cell and create a link between them as seen through lines 7-10.

We omit the prefixes in the pseudocode implementation; however, the prefixes are shown in Figure 24 and described in Main Paper Figure 2.

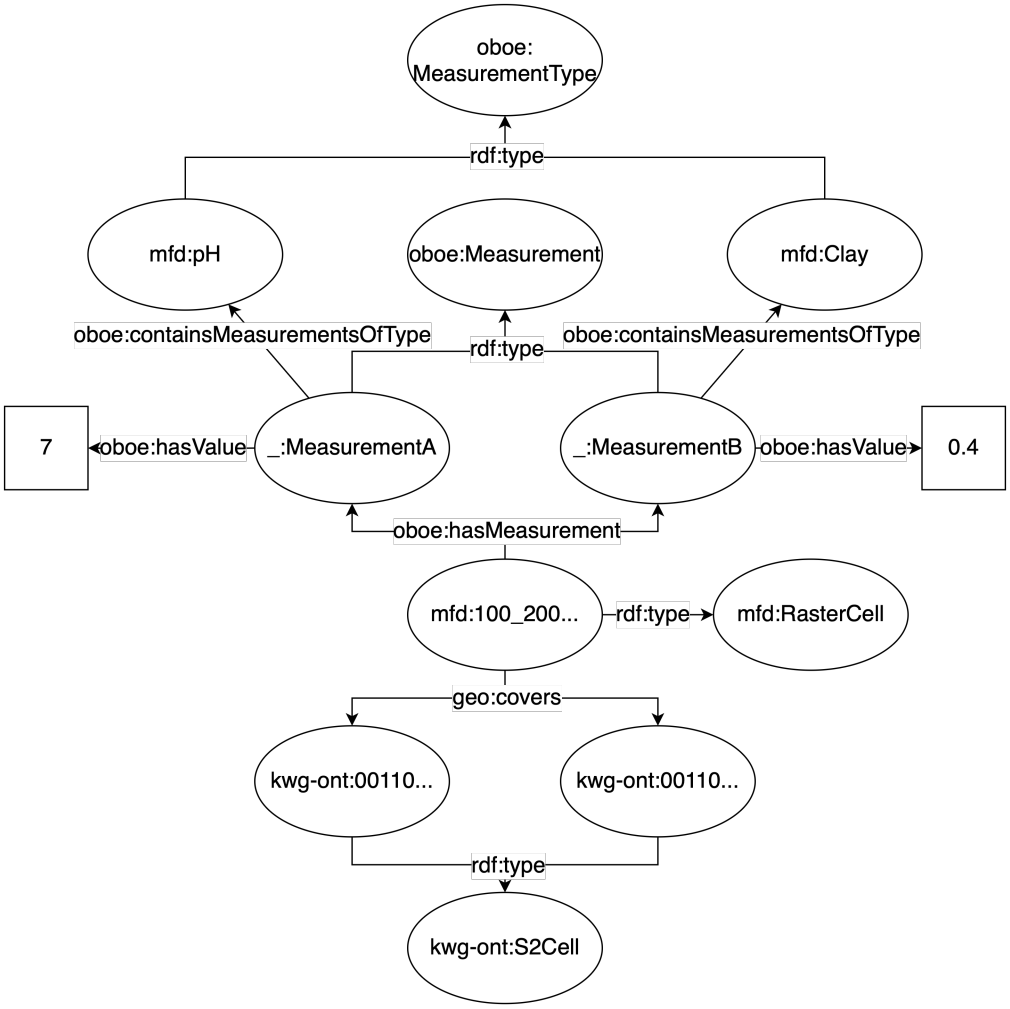


Figure 24: Example of RDF graph created by Algorithm 3.

At the center of Figure 24 we have a raster cell, `mfd:100_200...`, which has two measurements. These measures are the pH and clay, and have the values 7 and 0.4, respectively. Below the raster cell, it is linked to different S2 cells.

E Future Work

A key challenge that we mentioned in the conclusion of the main paper is the fact that we need to work on improving the scalability of the knowledge graph, given the large number of RDF triples related to the use of the S2 Geometry at the chosen level of granularity. In this section, we discuss possible future work that can be done to better handle the integration of large quantities of geospatial data into a knowledge graph framework.

E.1 Merge Identical Cells

An approach to reduce the number of triples required to model the geospatial data is to utilize the inherent hierarchical structure of the S2 cell IDs, as described in Appendix C.1.3, to merge S2 cells if all raster cells cover the entirety of their parent cell. For example, consider Figure 25 where three raster cells, measuring different attributes, all cover the same 15 S2 cells.

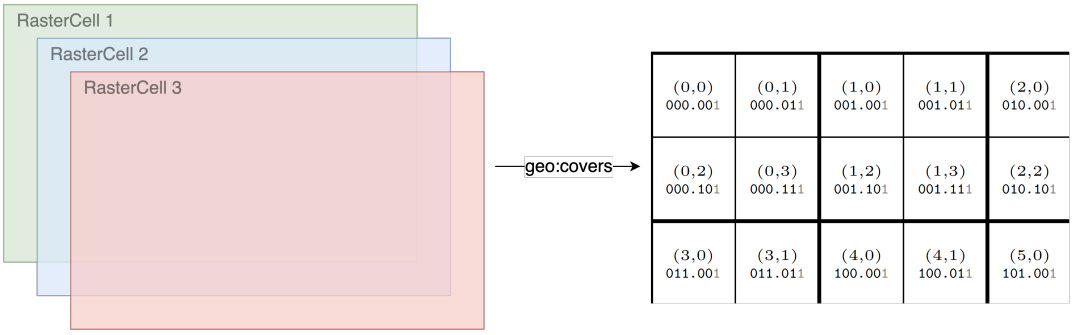


Figure 25: Three raster cells (left) cover the same S2 cells (right).

To model this geospatial data, 15 triples per raster cell are necessary when following the design from the main paper (see Figure 2 of the paper):

```
ex:rasterCell1
    geo:covers  S2:000.001  ;
    geo:covers  S2:000.011  ;
    geo:covers  S2:000.101  ;
    geo:covers  S2:000.111  ;
    geo:covers  S2:001.001  ;
    geo:covers  S2:001.011  ;
    geo:covers  S2:001.101  ;
    geo:covers  S2:001.111  ;
     ...
```

Instead, since all raster cells cover the child cells of face cell 0 and 1, we can replace those child cells with their parents, see Figure 26, without losing any information.

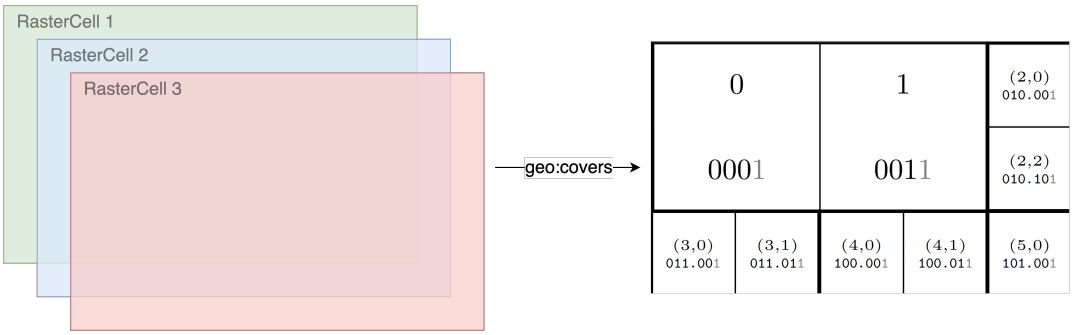


Figure 26: Merging S2 cells into parent S2 cells when possible.

Thus, the number of triples for each raster cell is reduced from 15 to 9, reducing the required storage needed to model the geospatial data.

```
ex:rasterCell1
    geo:covers  S2:0001     ;
    geo:covers  S2:0011     ;
    ...
```

Experimental results show that, at an S2 resolution of 24, we are able to reduce the approximate average number of links from 10×10 meter raster cells to S2 cells from 630 to 120 S2 cells, all without losing any information. Thereby, the number of S2 cells can be reduced by approximately 81%. We are able to aggregate some of the originally linked S2 cells to S2 cells at a resolution of 20. This approach, in combination with not linking raster cells covering the sea, will drastically reduce the number of triples.

E.2 Rasters in a Separate Graph

Another direction to optimize the storage and query speed of the knowledge graph would be to change the design to store the rasters as a separate graph and use that only as provenance to document the relation between raster cells and S2 cells. Figure 27 shows a possible design of this separate "provenance RDF graph". We define raster files as a subclass of a `sosa:ObservationCollection` and raster cells also as a subclass of a `sosa:ObservationCollection` under the Semantic Sensor Network Ontology[11] (SOSA). Then, we link one blank node of type `sosa:Observation` to each raster cell for each different observed property. This blank node carries the information on the type of property through the link `sosa:observedProperty`, and the actual measurement through the link `sosa:hasResult`.

[11] https://www.w3.org/TR/vocab-ssn/

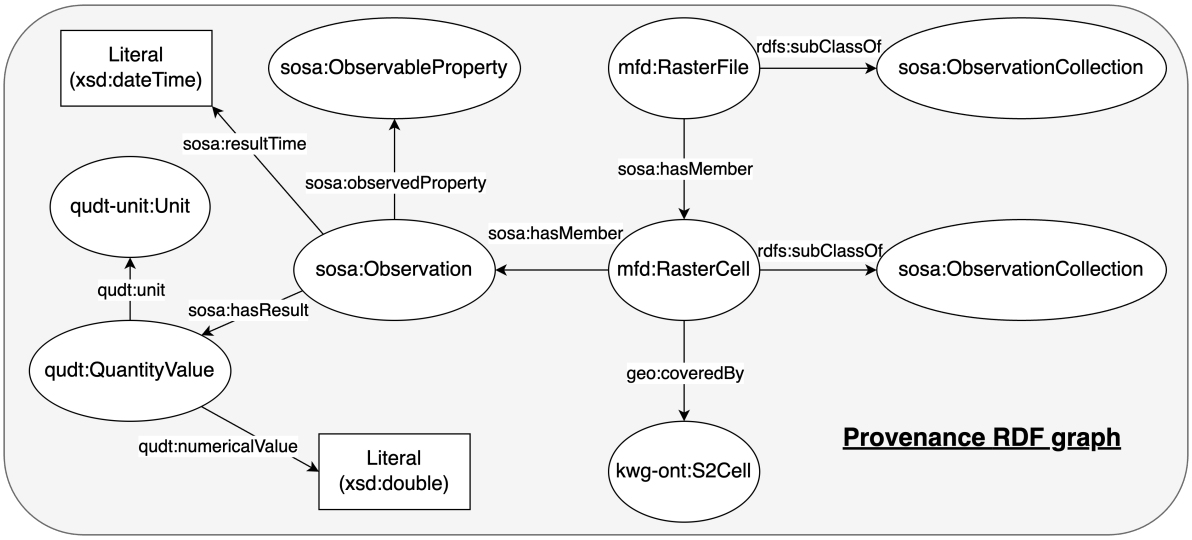


Figure 27: Design of the "provenance RDF graph" that documents the relation between raster cells and S2 cells.

As a result, the measurements in the "data RDF graph" need to be linked to the S2 cells instead of the raster cells, and also be aggregated. An updated design to allow for this is shown in Figure 28.

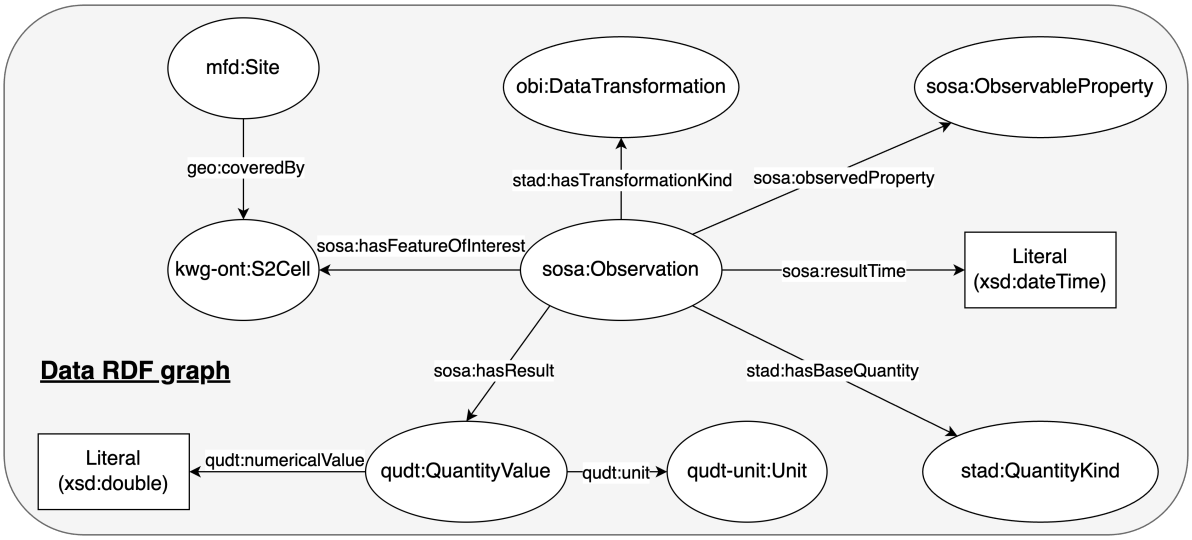


Figure 28: Design of the "data RDF graph" to capture information on aggregation.

The design is highly similar to the design in Figure 27, but includes the `DataTransformation`, from the Ontology for Biomedical Investigations (OBI)[12], in order to convey information on how the `Observation` was aggregated. The `Observation` is linked to the `DataTransformation` using the link `hasTransformationKind` from the Spatial and Temporal Aggregate Data ontology (STAD) described in [32]. Furthermore, the `Observation` is linked to a `QuantityKind` through the link `hasBaseQuantity` in order to show whether the `QuantityValue` is also an aggregate.

[12] https://obi-ontology.org

To exemplify how this design allows us to model aggregations of observations, assume that the `sosa:ObservableProperty` is *pH*, and that we have aggregated the observations into their mean. Then, the `stad:QuantityKind` could be *stad:SingleQuantityKind*, and the `obi:DataTransformation` could be *obi:ArithmeticMeanCalculation*.

E.3 Virtual Knowledge Graph

Another avenue for future work is in utilizing a Virtual Knowledge Graph (VKG) framework, illustrated in Figure 29, to reduce the number of realized triples in the knowledge graph. In the VKG framework, instead of storing all data in a triple store, a mapping between the KG ontology and the data sources is created such that the data is queried directly from the original sources. In practice, a VKG is a tuple $P = (\mathcal{O}, \mathcal{M}, \mathcal{S})$, where $\mathcal{O}$ is an ontology; $\mathcal{S}$ is a set of data source schemas; and $\mathcal{M} : \mathcal{S} \rightarrow \mathcal{O}$ is a mapping between them [33, 34, 35].

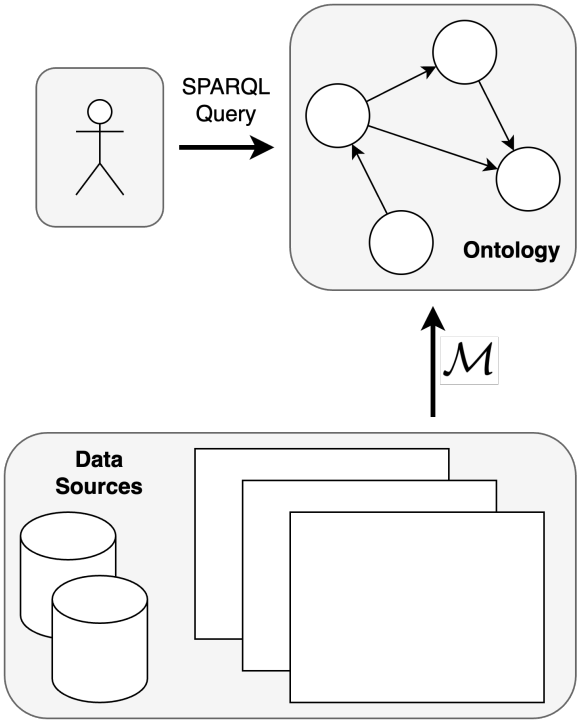


Figure 29: The Virtual Knowledge Graph framework.

Given that only the subset of the source data that is actually queried is ever materialized, using VKG methods can largely reduce the storage required, as opposed to materializing all source data in a knowledge graph. This possibly also allows for the queried data to be more fine-grained, i.e., having a finer-grained S2 level, which allows for more accurate results due to the aggregations explained in Appendix E.2.

The disadvantage of using VKG methods is that it requires a data integration system that supports it. For typical VKG frameworks, such as Ontop [35], the data source schemas $\mathcal{S}$ must be defined in a relational database management system, and the data sources themselves must be relational tables; furthermore, a framework for defining the mapping $\mathcal{M}$ is needed, such as R2RML [36]. Since this means that, specifically, source data defined as raster data is not supported, the design of a flexible VKG framework that supports source data in more varied formats is another avenue for future work.

E.4 Graph Representation Learning

The intention of the integrated KG is many-fold. For one, it facilitates easy access to, and integration of, information that may have been stored in different places and formats. Furthermore, the Additionally, it also allows the application of graph machine learning methods. While the current work has focused on building and validating the structure of the KG, several promising directions remain for future exploration using graph representation learning techniques.

One such direction is node classification, where the goal is to predict the class or category of entities within the KG, based on their attributes and relationships. This can be particularly useful for automatically labeling entities with missing or ambiguous type information. An example, for our KG, could be to classify species into functional groups.

Another avenue is link prediction, which involves inferring missing edges between entities. This can uncover hidden associations or potential missing connections not explicitly represented in the data. For example, a species might not be observed at a location, but in reality, it is present, which a link prediction method might discover.

Clustering and community detection also present interesting opportunities. By identifying densely connected subgraphs or communities of nodes, we can gain insights into the underlying structure of the KG, revealing groups of entities that share semantic or functional similarity. Different from node classification, community detection does not infer missing information. Instead, an example could be to cluster species that often appear together and thereby show some mutuality.

Appendix References

[21] W3C RDF Working Group, "Rdf 1.1 concepts and abstract syntax." https://www.w3.org/TR/rdf11-concepts/, 2014. W3C Recommendation, 25 February 2014.

[22] World Wide Web Consortium (W3C), "Rdf schema 1.1." https://www.w3.org/TR/rdf-schema/, 2014. Accessed: 2025-08-10.

[23] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 71:1–71:37, 2021.

[24] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2022.

[25] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev, "The dl-lite family and relations," *Journal of Artificial Intelligence Research*, vol. 36, pp. 1–69, 2009.

[26] "Sparql 1.1 query language." W3C Recommendation, Mar. 2013.

[27] P. Bolstad, *GIS fundamentals*, vol. 4. Eider Press White Bear Lake, MN, 2012.

[28] B. Bondaruk, S. A. Roberts, and C. Robertson, "Assessing the state of the art in discrete global grid systems: Ogc criteria and present functionality," *Geomatica*, vol. 74, no. 1, pp. 9–30, 2020.

[29] R. Battle and D. Kolas, "Geosparql: enabling a geospatial semantic web," *Semantic Web Journal*, vol. 3, no. 4, pp. 355–370, 2011.

[30] S. Stephen, M. Faulk, K. Janowicz, C. Fisher, T. Thelen, R. Zhu, P. Hitzler, C. Shimizu, K. Currier, M. Schildhauer, *et al.*, "The s2 hierarchical discrete global grid as a nexus for data representation, integration, and querying across geospatial knowledge graphs," *arXiv preprint arXiv:2410.14808*, 2024.

[31] B. Regalia, K. Janowicz, and G. McKenzie, "Computing and querying strict, approximate, and metrically refined topological relations in linked geographic data," *Transactions in GIS*, vol. 23, no. 3, pp. 601–619, 2019.

[32] K. Wiafe-Kwakye, T. Hahmann, and K. Beard, "An ontology design pattern for spatial and temporal aggregate data (stad)," in *Proceedings of the 13th Workshop on Ontology Design and Patterns (WOP 2022), co-located with the 21st International Semantic Web Conference (ISWC 2022)* (V. Svátek, V. A. Carriero, M. P. Villalón, C. Kindermann, and L. Zhou, eds.), vol. 3352 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[33] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Ontology-Based Data Access and Integration," in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), pp. 1–7, New York, NY: Springer New York, 2017.

[34] G. Xiao, L. Ding, B. Cogrel, and D. Calvanese, "Virtual Knowledge Graphs: An Overview of Systems and Use Cases," *Data Intelligence*, vol. 1, pp. 201–223, June 2019.

[35] "Ontop - A Virtual Knowledge Graph System." https://ontop-vkg.org. Accessed: 2025-06-30.

[36] "R2RML: RDB to RDF Mapping Language." https://www.w3.org/TR/r2rml/. Accessed: 2025-06-30.