KnowWhereGraph-Lite: A Perspective of the KnowWhereGraph

Cogan Shimizu[1], Shirly Stephen[2], Antrea Christou[1], Kitty Currier[2], Mohammad Saeid Mahdavinejad[3], Sanaz Saki Norouzi[3], Abhilekha Dalal[3], Adrita Barua[3], Colby K. Fisher[4], Anthony D'Onofrio[5], Thomas Thelen[6], Krzysztof Janowicz[2,7], Dean Rehberger[5], Mark Schildhauer[8], and Pascal Hitzler[3]

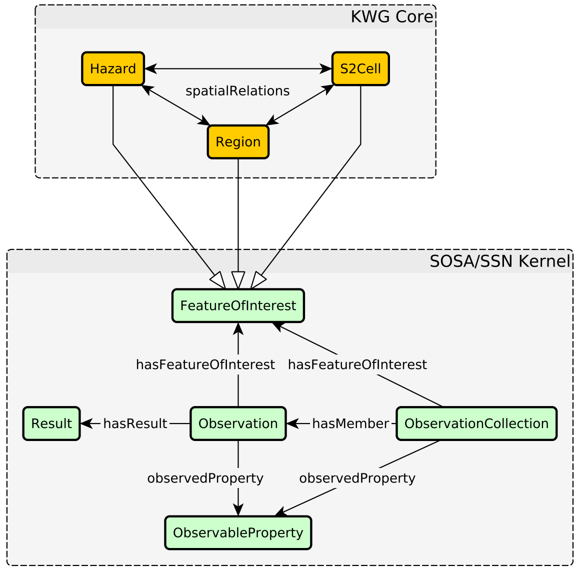
[1] Wright State University, USA
[2] University of California, Santa Barbara, USA
[3] Kansas State University, USA
[4] Hydronos Labs, USA
[5] Michigan State University
[6] Harvard University, USA
[7] University of Vienna
[8] National Center for Ecological Analysis & Synthesis, USA

Abstract. KnowWhereGraph (KWG) is a massive, geo-enabled knowledge graph with a rich and expressive schema. KWG comes with many benefits including helping to capture detailed context of the data. However, the full KWG can be commensurately difficult to navigate and visualize for certain use cases, and its size can impact query performance and complexity. In this paper, we introduce a simplified framework for discussing and constructing *perspectives* of knowledge graphs or ontologies to, in turn, construct simpler versions; describe our exemplar KnowWhereGraph-Lite (KWG-Lite), which is a perspective of the KnowWhereGraph; and introduce an interface for navigating and visualizing entities within KWG-Lite called KnowWherePanel.

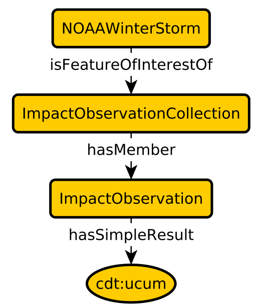
1 Introduction

KnowWhereGraph[9] (KWG) is one of the largest, publicly available geospatial knowledge graphs in the world [7,17]. KWG generally supports applications in the food, agriculture, humanitarian relief, and energy sectors and their attendant supply chains; and more specifically supports environmental policy issues relative to interactions among agricultural sustainability, soil conservation practices, and farm labor; and delivery of emergency humanitarian aid, within the US and internationally. To do so, KWG brings together over 30 datasets related to observations of natural hazards (e.g., hurricanes, wildfires, and smoke plumes), spatial characteristics related to climate (e.g., temperature, precipitation, and air quality), soil properties, crop and land-cover types, demographics, human health, and spatial representations of human-meaningful places, resulting in a

[9] `https://knowwheregraph.org/`



(a) This schema diagram depicts the two central modules of the KnowWhereGraph ontology.



(b) This schema diagram depicts an example chain between a type of hazard and the quantification of its impact.

Fig. 1: The KnowWhereGraph Ontology thoroughly encodes context, which, while effective, can result in a complex ontological structure.

knowledge graph with over 16 billion triples. To integrate these data, we have added an additional layer: KWG uses a schema that provides a thorough and rich[10] ontological representation formally describing the relationships between the types of data. The geospatial integration is performed by a consistent alignment to a Discrete Global Grid (DGG) [2], where we partition the surface of the Earth into small squares. These squares form an approximation of the spatial extent of physical and regional phenomena within the graph and act as a geospatial backbone.

However, due to the size and complexity of the graph and its schema, this can result in a steep learning curve and usability obstacles for those who are not well versed in ontologies, knowledge graphs, or SPARQL [4]. Likewise, for certain use cases a deep or rich ontological representation is not necessary; for some users visualizing or navigating graph data values can be unintuitive; and finally, the use of the DGG can be a barrier itself, as it can result in long and expensive queries. For example, a central piece of our schema is depicted in Figure 1a. Learning the specific value of an observation pertaining to a particular feature of interest is already a complex, multi-hop query. Figure 1b shows how a particular instance of a Hazard finally relates to an observation on its impact, which is encoded as a

[10] The OWL ontology has over 300 classes and about 3,000 axioms.

data type. This rather complex way of representing the data is necessary because some of the target applications of our graph are aimed at specialists who require a high level of detail. However, for less involved application use cases, we aim to simplify this process and reduce the conceptual barrier to entry, as well as improve performance for certain use cases. To do so, we have created a simplified version of KWG, which we call KnowWhereGraph Lite (KWG-Lite) and is a *perspective* of the base graph. The "lite" version of the graph is constructed from a series of SPARQL CONSTRUCT queries, which, in turn, are formulated from the sets of shortcuts and views that define the perspective.

To quickly access knowledge and data from KWG-Lite, we have developed a visualization interface, which we call KnowWherePanel (KWP), inspired by Google's knowledge panels[11] and Wikipedia's infoboxes.[12] Entities within KWG-Lite can be viewed through KWP, which shows the simplified views in an easily consumable tabular format. Furthermore, the tool provides a way to export embeddable, styled snapshots of each panel, similar to how one can embed Tweets from Twitter, with a styled, living view of the data.

In summary, our contributions are as follows.

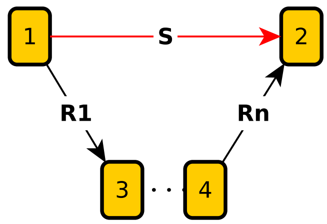
1. Perspective Development – A method for defining, creating, and utilizing perspectives over a graph;
2. Perspective Deployment – a method for creating an effective, efficient, and powerful UI for defined user groups from large knowledge graphs;
3. KnowWhereGraph-Lite – a subgraph of the KnowWhereGraph that is intended for simple queries, easy visualization, and quick consumption;
4. KnowWherePanel – a visual interface for generating "panels," which contain a view of an entity; these panels can then be embedded in other media.

The rest of this paper is organized as follows. Section 2 presents the underlying technique used to generate and describe this perspective of the KnowWhereGraph, which is, in turn, used to create KnowWhereGraph-Lite – which we introduce in Section 4. Section 3 briefly discusses related work. Entities in KWG-Lite can be viewed in a number of different ways. In Section 5, we present the KnowWherePanel interface for transferable and embeddable views of the entity. In Section 6 we discuss resource availability. Finally, in Section 7, we conclude with future work and next steps.

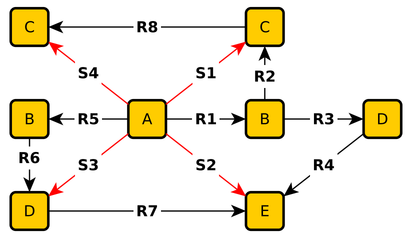
2 Technical Foundation

The technical basis for constructing the KWG-Lite graph is relatively straightforward. The process is to reduce an expressive structure (i.e., one that encodes context, provenance, and so on in a rich manner) into something more easily queryable, approachable by humans, and easily visualized. We often call such simplified structure a *star pattern*, due to its shape. That is, important entities

[11] `https://support.google.com/knowledgepanel/answer/9163198?hl=en`
[12] `https://en.wikipedia.org/wiki/Infobox`



(a) Golden boxes are arbitrary classes. The dotted line between boxes 3 and 4 indicates an arbitrary number of binary predicates. The *shortcut* is depicted as a red arrow.



(b) This figure displays a *view* for the class A. Essentially, it constructs a star pattern out of the more expressive structure.

Fig. 2: Depictions of a shortcut and a view.

are directly related to each other (or explicit data values) with minimal context (or the context is encoded into the name of the relationship, reducing machine interpretability).

Our approach to this process is to define a *perspective* of an expressive ontology. A *perspective* is defined as a set of *shortcuts* that link classes and datatypes by removing or reducing contextual information (i.e., the "skipped-over" nodes in the graph).

A *shortcut* is effectively a bidirectional role chain that is intended to reduce the complexity of an expressive ontological structure, either by facilitating querying or improving human understanding of the encoded data. However, due to the limitations on tractability, we cannot actually express within OWL-DL [5] that a shortcut also implies the expressive structure [10]. We are limited to expressing only that the shortcut is the super-property of the role chain. This role chain,

$$\mathsf{R}_1 \circ \cdots \circ \mathsf{R}_n \sqsubseteq \mathsf{S},$$

is depicted graphically in Figure 2a. As a rule, it can also be written as

$$\mathsf{R}_1(x_0, x_1) \wedge \cdots \wedge \mathsf{R}_n(x_{n-1}, x_n) \rightarrow \mathsf{S}(x_0, x_n).$$

The body (left-hand side) of the rule, as well as its head (right-hand side), can also take more complex forms, e.g., by adding type information to the variables, such as in

$$\begin{aligned} \mathsf{C}_0(x_0) \wedge \mathsf{R}_1(x_0, x_1) \wedge \mathsf{C}_1(x_1) \wedge \cdots \wedge \mathsf{C}_{n-1}(x_{n-1}) \wedge \mathsf{R}_n(x_{n-1}, x_n) \wedge \mathsf{C}_n(x_n) \\ \rightarrow \mathsf{D}(x_0) \wedge \mathsf{S}(x_0, x_n) \wedge \mathsf{E}(x_n), \end{aligned} \tag{1}$$

or even more complex rule bodies (or heads). In the case of KWG-Lite that we present in this paper, we cast such rules into SPARQL CONSTRUCT queries (discussed further below), but we would like to also remark that many (but not all) of these rules can be expressed in OWL-DL using a technique called *rolification*, which was introduced in [14] (or see [9] for a tutorial). Rules that cannot be converted that way to OWL can be approximated using so-called *nominal schemas* [11].

A *view* is a set of such shortcuts for a particular class in the knowledge graph. This is shown in Figure 2b. A *perspective* of an ontology (or knowledge graph) is defined as a set of views.[13] Essentially, for some (sub)set of the classes in the ontology, we construct simplified views of the data or knowledge. Intuitively, one now has *a* perspective of the graph. Indeed, we provide an example of one such perspective in Section 4 as a core contribution of this paper: KnowWhereGraph-Lite.

In some cases, it is undesirable to materialize directly, or even include, these shortcuts in the base ontology or schema. Nor is it always desirable to even attach formal semantics (in the form of ontology axioms) to the shortcut. Frequently, a shortcut will encode context in the name of the connecting predicate. For example, in our KWG-Lite, we have averageHeatingDegreeDaysPerMonthAug2021. As such, we can also leave annotations within the base ontology, indicating where convenient (and human-meaningful) shortcuts exist, and thus also leave tooling to enable the retrieval and rendering of a view.

In doing so, these annotations can be consumed and mapped to SPARQL CONSTRUCT queries to construct a materialized perspective. We provide examples of such in Section 4.

3 Related Work

The concept of leveraging short paths through a graph for aiding understanding, navigation, visualization, or publishing is not altogether new.

The concept of shortcuts, as it applies to linked data, was first explored in [13]. There, shortcuts are intrinsically tied to the notion of pattern-flattening and pattern-expansion, which are methods for publishing and ingesting data at different levels of conceptual granularity. A similar but less principled approach was taken in creating the GeoLink Base Ontology from the GeoLink Modular Ontology [19]. In our case, we have built on these concepts to go beyond a simple pattern-based method to produce a navigable "lite" version of KnowWhereGraph. In [10] the formal logical underpinnings of shortcuts, in the context of OWL, were discussed.

From a tooling perspective, WebVOWL [12] offers a customizable view of an instance graph by automatically flattening or condensing paths of certain lengths. This can be exceptionally useful when examining the graph in its force-directed

[13] The general identification of *which* classes should feature prominently in the perspective is a human-centric process, which is outside of the scope of this paper.

layout. However, in our case, not all shortcuts in the perspective are of the same length. Equally as important, it would be effectively impossible to use a WebVOWL view of an instance graph to render KnowWhereGraph meaningfully.

WiSP, or "Weighted Shortest Paths" [18], is a strategy for finding "interesting" connections across the graph. In their discussion, these tend to be meaningful to humans and sometimes even insightful. However, due to how they are constructed, they are not guaranteed to return the same sort of information for entities of the same type. WiSP would be an interesting component to add from a visualization and discovery perspective, but for the formulation of a consistent "lite" graph, it will not work.

4 KnowWhereGraph-Lite

For KnowWhereGraph-Lite (KWG-Lite), two classes were selected to serve as the central concepts of the perspective: HazardEvent and Place, which correspond to the classes Hazard and Region from KnowWhereGraph. We opted to modify the names of the classes to make it human-meaningful and to reduce confusion between to the two graphs.

The schema diagram for KWG-Lite is shown in Figure 3. Note the distinctive (mostly) star shape of the diagram. As we have stated above, we have removed much of the contextual information, for example, by collapsing the SOSA/SSN kernel into a single relation (with a complex name), such as averageHeatingDegreeDaysPerMonthAug2021. How this name is constructed is shown in Figure 4. The ObservableProperty (i.e., averageHeatingDays) is combined with the result time (i.e., a month and year), which points directly to the value (i.e., the simple result) of the observation pertaining to that observed property. This results in a large number of unique predicates. However, since these predicates are manually curated, and from a panel perspective *all* relations are meaningful, querying to populate a panel for a particular entity is quite simple.

To construct KWG-Lite (i.e., to materialize the graph), we developed a set of SPARQL CONSTRUCT queries based on the schema diagram in Figure 3. One example of such a query is shown in Figure 5. This query is meant to be executed against the base graph (i.e., KnowWhereGraph) and will produce a set of triples that shall constitute a portion of the lite graph. The queries pull double duty in also rebinding the entity from KWG to the KWG-Lite namespace.

In total, we utilized over 20 queries to construct the full KWG-Lite, which are documented in our repository.[14] In all, this results in a graph that contains approximately 200,000 triples (which is about four orders of magnitude smaller than the base graph). Figure 6 shows the respective triple structures for KWG and KWG-Lite.

[14] `https://github.com/KnowWhereGraph/knowwheregraph-lite/blob/main/construct-queries.md`

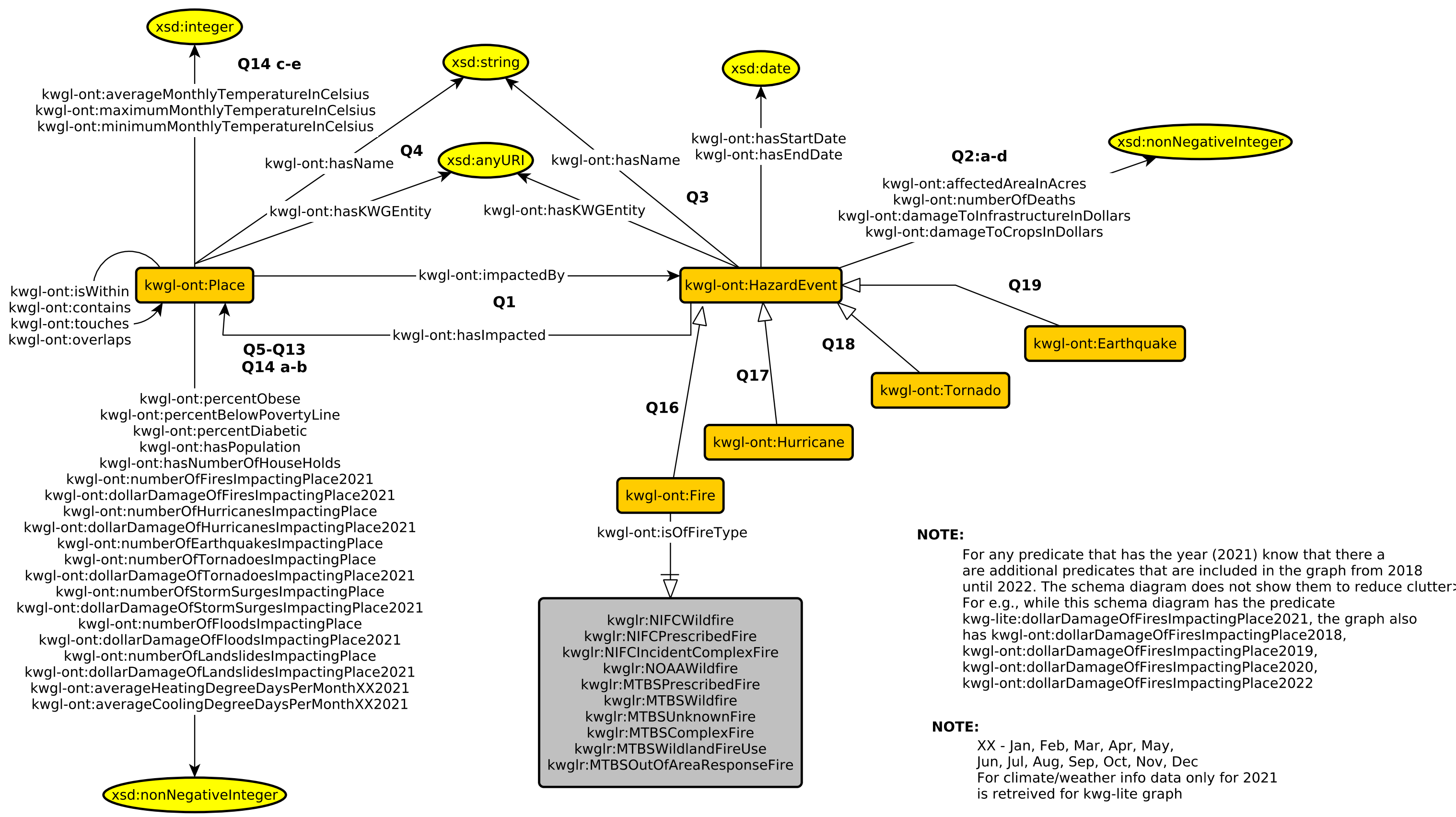


Fig. 3: This schema diagram depicts the entities and relations within KnowWhereGraph-Lite.

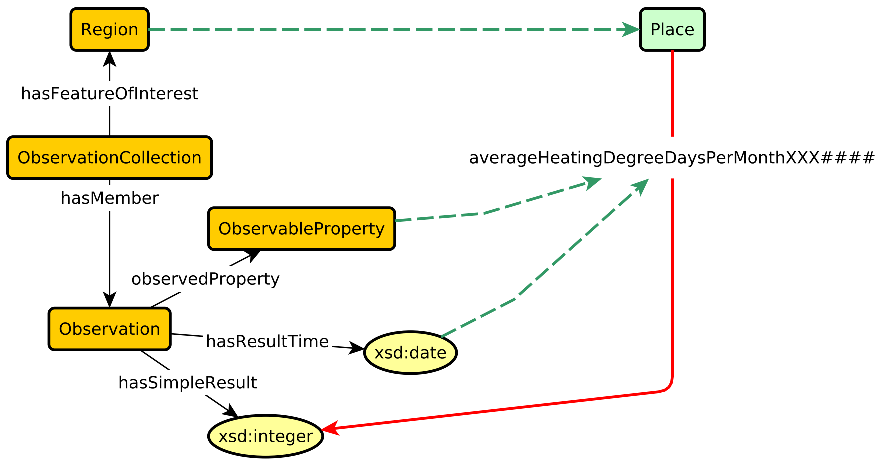


Fig. 4: Several components of the complex ontological structure contribute to the KnowWhereGraph-Lite perspective. Instead of having a reified construction, context is encoded in the relation name. The green box distinguishes Place as part of KWG-Lite.

In addition to providing a SPARQL endpoint specifically including KWG-Lite, we provide the KnowWherePanel interface, which we describe in the next section.

5 KnowWherePanel

KnowWherePanel (KWP) is a custom-built interface for viewing entities in KWG-Lite. The name and shape are inspired by Google's knowledge panels and Wikipedia's infoboxes. Indeed, it is actually quite similar to the interaction between Wikidata and Wikipedia. KWG-Lite serves as an underlying knowledge base, and KWP provides a formatted and shareable panel that summarizes the pertinent information for that entity.

Panels can adapt to the type of entity that they are visualizing. Currently, we support two different KWP visualizations: KWP HazardEvents and KWP Places.

Both panel views display the information that is pertinent to the type of entity being described (as depicted in Figure 3). However, not all HazardEvents have the same characteristics. For example, a tornado and earthquake are characterized by significantly different data provided by different agencies. Instead, what changes is the format of the visualization.

KWP HazardEvent panels, as shown in Figure 7a, display the relevant information in a (mostly) tabular format, including the type of hazard (currently either tornado, hurricane, fire, or earthquake), its temporal scope, and a list of its

```
CONSTRUCT {
    ?p kwgl-ont:impactedBy ?h.
    ?h kwgl-ont:hasImpacted ?p.
}
WHERE {
    ?place a kwg-ont:Region.
    ?place kwg-ont:spatialRelation ?hazard.
    ?hazard a kwg-ont:Hazard.
    BIND(
      STRAFTER(STR(?place),
               "http://stko-kwg.geog.ucsb.edu/lod/resource/")
               as ?placeName)
    BIND(
      CONCAT("http://stko-kwg.geog.ucsb.edu/lod/lite-resource/",
             ?placeName)
             as ?litePlace)
    BIND(IRI(?litePlace) AS ?p ).

    BIND(
      STRAFTER(STR(?hazard),
               "http://stko-kwg.geog.ucsb.edu/lod/resource/")
               as ?hazardName)
    BIND(
      CONCAT("http://stko-kwg.geog.ucsb.edu/lod/lite-resource/",
             ?hazardName)
             as ?liteHazard)

    BIND( IRI(?liteHazard) AS ?h ).
}
```

Fig. 5: This is an example SPARQL CONSTRUCT query that identifies Places that have been impacted by a HazardEvent.

impacts. HazardEvents (and also Places) are back-linked to the full representation in KWG, as well.

KWP Place panels, as shown in in Figure 7b, display basic statistics in tabular format, followed by a categorical list of the types of hazards that have impacted that place. The resultant statistics are summed per entity type and displayed as time-series data, as in Figure 7c. In most cases, KWG (and thus KWG-Lite) have data up to 2022. However, the query is, itself, dynamic and will update its temporal scope as new data become available.

KWP is, itself, implemented as a static website that creates browsable panels for Places and Hazards in KWG-Lite. The site operates fully in JavaScript: using jQuery/AJAX to query the public graph, and Fuse.js, which provides a fuzzy search functionality.

KWG triple structure:

```
kwgr:noaaClimateDiv.403 sosa:isFeatureOfInterestOf
    kwgr:noaaClimateDivObservationCollection.403 .
kwgr:noaaClimateDivObservationCollection.403 sosa:hasMember
    kwgr:noaaClimateDivPropObservationCollection.403.hdd .
kwgr:noaaClimateDivPropObservationCollection.403.hdd sosa:hasMember
    kwgr:noaaClimateDivObservation.403.HDD.202108 .
kwgr:noaaClimateDivObservation.403.HDD.202108 sosa:hasResult
    kwgr:noaaClimateDivObservationResult.403.hdd.202108 .
kwgr:noaaClimateDivObservationResult.403.hdd.202108 kwg-ont:value
    kwgr:noaaClimateDivObservationQuantityValue.403.hdd.202108 .
noaaClimateDivObservationQuantityValue.403.hdd.202108 qudt:numericValue
    "59.0"^^xsd:float .
```

KWG-Lite triple structure:

```
kwglr:noaaClimateDiv.403
  kwgl-ont:averageHeatingDegreeDaysPerMonthAug2021 "59"^^xsd:float .
```

Fig. 6: This block shows the respective triple structure for KWG and KWG-Lite, as indicated by Figures 3 and 4.

6 Resource Availability

We have attempted to follow as many applicable best practices as possible for the deployment and provisioning of KWG-Lite and KWP. It should be noted that KWG and, thus, KWG-Lite are both RDF graphs, leveraging W3C standards (e.g., OWL:Time [1], SOSA/SSN [6], and PROV-O [15]), and designed using practices already anchored in the literature [17,16,13].

From an availability standpoint, we have deployed KWG-Lite in a GraphDB [3] repository, which can be directly queried from a SPARQL endpoint.[15] Documentation for KWG-Lite (e.g., its schema diagram), as well as our set of SPARQL queries used to construct KWG-Lite, are housed in our public repository.[16] The KWP interface has open source code[17] and is publicly available for use.[18] KWG-Lite and KWP are released under the CC-BY-4.0 license.[19]

15 `https://stko-kwg.geog.ucsb.edu/workbench/` and choosing KWG-Lite as the repository (top-right).
16 `https://github.com/KnowWhereGraph/knowwheregraph-lite`
17 `https://github.com/KnowWhereGraph/kw-panels`
18 `https://knowwheregraph.org/kw-panels/`
19 `https://creativecommons.org/licenses/by/4.0/`

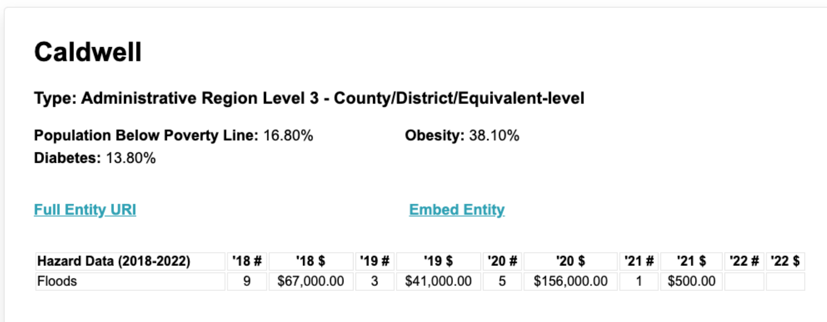
NOAATornado Occurred in EARLY from 2005-01-13-1758 to 2005-01-13-1802, EST

Tornado

Start Date: Thu Jan 13 2005 07:58:00 GMT-0500
End Date: Thu Jan 13 2005 08:02:00 GMT-0500
Death toll: 2
Loss of Infrastructure: $250,000.00

Full Entity URI Embed Entity

(a) This figure depicts a KnowWherePanel for a HazardEvent from KWG-Lite.



Caldwell

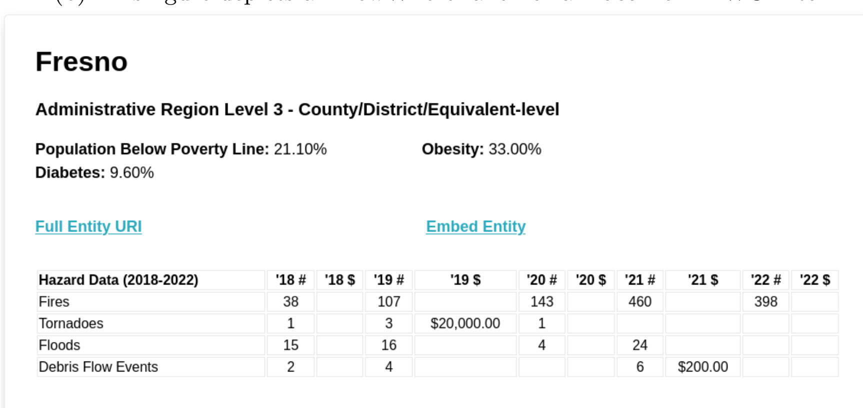
Type: Administrative Region Level 3 - County/District/Equivalent-level

Population Below Poverty Line: 16.80%
Obesity: 38.10%
Diabetes: 13.80%

Full Entity URI
Embed Entity

Hazard Data (2018-2022)	'18 #	'18 $	'19 #	'19 $	'20 #	'20 $	'21 #	'21 $	'22 #	'22 $
Floods	9	$67,000.00	3	$41,000.00	5	$156,000.00	1	$500.00		

(b) This figure depicts a KnowWherePanel for a Place from KWG-Lite.



Fresno

Administrative Region Level 3 - County/District/Equivalent-level

Population Below Poverty Line: 21.10%
Obesity: 33.00%
Diabetes: 9.60%

Full Entity URI
Embed Entity

Hazard Data (2018-2022)	'18 #	'18 $	'19 #	'19 $	'20 #	'20 $	'21 #	'21 $	'22 #	'22 $
Fires	38		107		143		460		398	
Tornadoes	1		3	$20,000.00	1					
Floods	15		16		4		24			
Debris Flow Events	2		4				6	$200.00		

(c) This figure depicts a KnowWherePanel for a Place from KWG-Lite that has had multiple types of HazardEvents impact it. Currently, missing data are left blank, as it is not currently known if data are missing or if there were simply no impacts, at all.

Fig. 7: Examples of different KnowWherePanels and the information they are capable of displaying.

KWG-Lite and KWP are managed under the auspices of the KnowWhereGraph project. As such, both resources described herein will be maintained under the same sustainability plan.

- Resource repositories and documentation will remain available in perpetuity.
- Services (e.g., the endpoints and websites) are hosted on institutional, archival resources. The URIs are expected to be indefinitely available.
- Maintenance and updates to the resources are guaranteed through 2025 or longer, pending Foundation establishment.

Finally, we intend for this paper to serve as the canonical citation.

7 Conclusion

KnowWhereGraph is a massive knowledge graph [7] with a rich and expressive schema [8,17]. This comes with many benefits, insofar as it helps to capture provenance, lineage, and spatiotemporal context of the data, and other aspects relevant for expert-level applications. However, it can be commensurately difficult to navigate and visualize for certain (generally simpler) use cases, and its size can impact query performance and complexity.

In this paper, we have introduced a simplified framework for discussing and constructing *perspectives* of knowledge graphs or ontologies which allow us to construct simpler versions of the graph or ontology; introduced our exemplar KnowWhereGraph-Lite, which is a perspective of the KnowWhereGraph; and introduced, as well, our interface for navigating and visualizing entities within KWG-Lite: KnowWherePanel.

Future Work

We have identified the following items for potential next steps regarding KWG-Lite and KnowWherePanel:

1. include additional places and types of hazards (e.g., drought zones) as part of KWG-Lite,
2. include additional top-level entities for KWG-Lite (e.g., create a panel view for the units – Cells or squares – composing the DGG), and
3. develop additional alternative adaptive views for KnowWherePanel, for either the new top-level entities or alternative visualizations for existing entity types.

Finally, we note that KnowWhereGraph-Lite and KnowWherePanel are both living entities; the schema and materialization of KWG and thus the queries are expected to evolve. We have tried to keep a snapshot of KWG, KWG-Lite, and KWP versioned, but these *are* expected to change in the future.

Acknowledgement. This work was funded by the National Science Foundation under Grant 2033521 A1: KnowWhereGraph: Enriching and Linking Cross-Domain

Knowledge Graphs using Spatially-Explicit AI Technologies. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Cox, S., Little, C.: Time ontology in OWL. W3C recommendation, W3C (Oct 2017), https://www.w3.org/TR/2017/REC-owl-time-20171019/
2. Goodchild, M.F.: Discrete global grids for digital earth. In: International Conference on Discrete Global Grids. pp. 26–28. Citeseer (2000)
3. GraphDB. `http://graphdb.ontotext.com/`
4. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C recommendation, W3C (Mar 2013), https://www.w3.org/TR/2013/REC-sparql11-query-20130321/
5. Hitzler, P., Parsia, B., Rudolph, S., Patel-Schneider, P., Krötzsch, M.: OWL 2 web ontology language primer (second edition). W3C recommendation, W3C (Dec 2012), https://www.w3.org/TR/2012/REC-owl2-primer-20121211/
6. Janowicz, K., Haller, A., Cox, S., Lefrançois, M., Phuoc, D.L., Taylor, K.: Semantic sensor network ontology. W3C recommendation, W3C (Oct 2017), https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/
7. Janowicz, K., Hitzler, P., Li, W., Rehberger, D., Schildhauer, M., Zhu, R., Shimizu, C., Fisher, C.K., Cai, L., Mai, G., Zalewski, J., Zhou, L., Stephen, S., Estrecha, S.G., Mecum, B.D., Lopez-Carr, A., Schroeder, A., Smith, D., Wright, D.J., Wang, S., Tian, Y., Liu, Z., Shi, M., D'Onofrio, A., Gu, Z., Currier, K.: Know, know where, Knowwheregraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. AI Mag. **43**(1), 30–39 (2022). https://doi.org/10.1609/aimag.v43i1.19120
8. Janowicz, K., Shimizu, C., Hitzler, P., Mai, G., Stephen, S., Zhu, R., Cai, L., Zhou, L., Schildhauer, M., Liu, Z., Wang, Z., Shi, M.: Diverse data! Diverse schemata? Semantic Web **13**(1), 1–3 (2022). https://doi.org/10.3233/SW-210453
9. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and rules. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P.F. (eds.) Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer (2011). https://doi.org/10.1007/978-3-642-23032-5_7
10. Krisnadhi, A.A., Hitzler, P., Janowicz, K.: On the capabilities and limitations of OWL regarding typecasting and ontology design pattern views. In: Tamma, V.A.M., Dragoni, M., Gonçalves, R.S., Lawrynowicz, A. (eds.) Ontology Engineering – 12th International Experiences and Directions Workshop on OWL, OWLED 2015, co-located with ISWC 2015, Bethlehem, PA, USA, October 9-10, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9557, pp. 105–116. Springer (2015). https://doi.org/10.1007/978-3-319-33245-1_11
11. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 – April 1, 2011. pp. 645–654. ACM (2011). https://doi.org/10.1145/1963405.1963496

12. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016). https://doi.org/10.3233/SW-150200
13. Rodríguez-Doncel, V., Krisnadhi, A.A., Hitzler, P., Cheatham, M., Karima, N., Amini, R.: Pattern-based linked data publication: The linked chess dataset case. In: Hartig, O., Sequeda, J.F., Hogan, A. (eds.) Proceedings of the 6th International Workshop on Consuming Linked Data co-located with 14th International Semantic Web Conference (ISWC 2105), Bethlehem, Pennsylvania, USA, October 12th, 2015. CEUR Workshop Proceedings, vol. 1426. CEUR-WS.org (2015), `https://ceur-ws.org/Vol-1426/paper-05.pdf`
14. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: Baader, F., Lutz, C., Motik, B. (eds.) Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008. CEUR Workshop Proceedings, vol. 353. CEUR-WS.org (2008), `https://ceur-ws.org/Vol-353/RudolphKraetzschHitzler.pdf`
15. Sahoo, S., McGuinness, D., Lebo, T.: PROV-O: The PROV ontology. W3C recommendation, W3C (Apr 2013), http://www.w3.org/TR/2013/REC-prov-o-20130430/
16. Shimizu, C., Hammar, K., Hitzler, P.: Modular ontology modeling. Semantic Web **14**(3), 459–489 (2023). https://doi.org/10.3233/SW-222886
17. Shimizu, C., Stephen, S., Currier, K., Hitzler, P., Zhu, R., Janowicz, K., Schildhauer, M., Mahdavinejad, M.S., Dalal, A., Barua, A., Cai, L., Mai, G., Wang, Z., Tian, Y., Norouzi, S.S., Liu, Z., Shi, M., Fisher, C.K.: The knowwheregraph ontology. Pre-print (May 2023), https://daselab.cs.ksu.edu/publications/knowwheregraph-ontology
18. Tartari, G., Hogan, A.: Wisp: Weighted shortest paths for RDF graphs. In: Ivanova, V., Lambrix, P., Lohmann, S., Pesquita, C. (eds.) Proceedings of the Fourth International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 17th International Semantic Web Conference, VOILA@ISWC 2018, Monterey, CA, USA, October 8, 2018. CEUR Workshop Proceedings, vol. 2187, pp. 37–52. CEUR-WS.org (2018), `https://ceur-ws.org/Vol-2187/paper4.pdf`
19. Zhou, L., Cheatham, M., Krisnadhi, A., Hitzler, P.: GeoLink data set: A complex alignment benchmark from real-world ontology. Data Intell. **2**(3), 353–378 (2020). https://doi.org/10.1162/dint_a_00054