Department of Computing Science
Umeå University, SE-901 87 Umeå, Sweden
www.umu.se/english

UMEÅ UNIVERSITY

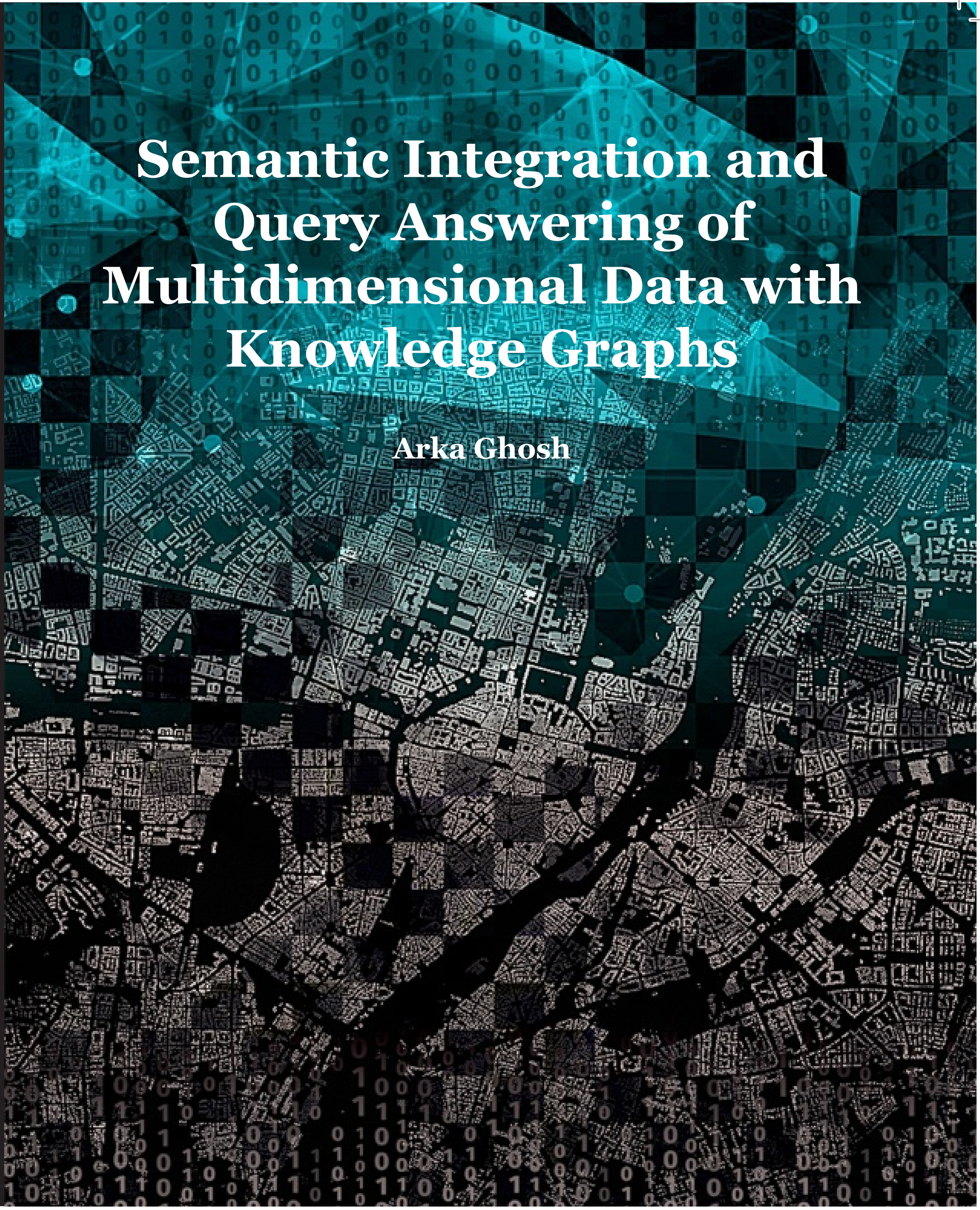
ISBN 978-91-8070-971-2 (print)
ISBN 978-91-8070-972-9 (pdf)
ISSN 0348-0542

Arka Ghosh

Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Umeå University

2026



Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Arka Ghosh

UMEÅ UNIVERSITY

UMEÅ UNIVERSITY

Report / UMINF

Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Arka Ghosh

Academic dissertation

Which, with the due permission of the Vice-Chancellor of Umeå University for the examination for the Degree of Doctor of Philosophy in Science, is presented for public defence in MIT.A.121. on Wednesday, 8 April, 2026 at 13:00.

The thesis will be defended in English.

Faculty opponent:
Professor dr. Ahmet Soylu, Kristiania University of Applied Sciences, Oslo, Norway

Department of Computing Science

Organisation
Umeå University
Department of Computing Science

Document type
Doctoral thesis

Date of publication
18 March 2026

Author
Arka Ghosh

Title
Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Abstract

Modern enterprises face an overwhelming volume of big data—continuously generated at high velocity, in diverse formats, from multiple sources, and stored at significant expense. To derive value for AI training and insightful analytics, this heterogeneous data must be properly linked and processed in machine-readable form. Traditional data ingestion pipelines, such as Extract-Transform-Load (ETL/ELT), materialise structured and unstructured data for query processing, but they become inefficient or overloaded when updates occur frequently. This thesis investigates Virtual Knowledge Graphs (VKGs), also known as Ontology-based Data Access (OBDA), as a virtualisation approach for flexible, efficient management of large-scale structured and unstructured data without materialisation. Our core focus is on integrating multidimensional raster data (e.g., images or continuous spatial-temporal phenomena represented as arrays) with relational/vector data (points, lines, polygons) and linked data. Conventional VKG systems mediate OWL ontologies over relational databases via declarative R2RML mappings, translating SPARQL queries over the ontology into optimised SQL executions. However, they lack native support for raster data's multidimensional nature and seamless integration with vector geometries, which restricts their use in geospatial settings such as Earth Observation, Geographical Information Systems, and Urban planning. To overcome these limitations, we propose OntoRaster, a novel extended VKG framework that enables virtual integration and querying of raster, relational, and vector data. OntoRaster leverages domain-specific ontologies and supports on-the-fly query processing, thus minimising expensive data transfers by delegating computations to specialised back-ends. For rigorous evaluation, we introduce comprehensive benchmarks for assessing such VKG systems, i.e., OntoRaster on integrated raster-vector data processing workloads, using domain ontologies and synthetically generated, similar, and scalable datasets that vary in properties (resolution, coverage, pixel density, polygons, etc). Finally, we explore semantic enrichment for location-based business intelligence by exposing a SQL-accessible knowledge graph to BI tools. Additionally, we explore an LLM-based system for generating raster-based SPARQL queries based on ontologies from natural language queries. This empowers business experts to obtain actionable insights for location-based decision-making and risk assessment without requiring advanced geospatial expertise. Overall, this dissertation bridges the Big Data, Semantic Web, AI, and BI communities, advancing scalable, ontology-mediated access to heterogeneous, multidimensional, spatial-temporal geo-data.

Keywords: Artificial Intelligence (AI), Semantic Web (SW), Business Intelligence (BI), Virtual Knowledge Graph (VKG), Multidimensional Raster Data, Relational Data, Vector Data, Linked Data, Large Language Models (LLMs)

Language
English

ISBN
978-91-8070-971-2 (print)
978-91-8070-972-9 (pdf)

ISSN
0348-0542

Number of pages
171 pages

Department of Computing Science, 2026

Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Arka Ghosh

UMEÅ UNIVERSITY

PhD Defence Committee Members

ISBN: 978-91-8070-971-2 (print)
ISBN: 978-91-8070-972-9 (pdf)
ISSN: 0348-0542
Report / UMINF
Cover reproduced with permission
Electronic version available at: http://umu.diva-portal.org/
Printed by: Scandinavian Print Group, Skarpnäck, 2026

UMEÅ UNIVERSITY

Semantic Integration and Query Answering of Multidimensional Data with Knowledge Graphs

Arka Ghosh

Doctoral Thesis, April 2026
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

arkag@cs.umu.se

ISBN (printed) 978-91-8070-971-2
ISBN (digital) 978-91-8070-972-9
ISSN 0348-0542
UMINF 26.04

The cover image was designed by the author with permission
Printed by Scandinavian Print Group, Skarpnäck, 2026

Abstract

Modern enterprises face an overwhelming volume of big data—continuously generated at high velocity, in diverse formats, from multiple sources, and stored at significant expense. To derive value for AI training and insightful analytics, this heterogeneous data must be properly linked and processed in machine-readable form. Traditional data ingestion pipelines, such as Extract-Transform-Load (ETL/ELT), materialise structured and unstructured data for query processing, but they become inefficient or overloaded when updates occur frequently. This thesis investigates Virtual Knowledge Graphs (VKGs), also known as Ontology-based Data Access (OBDA), as a virtualisation approach for flexible, efficient management of large-scale structured and unstructured data without materialisation. Our core focus is on integrating multidimensional raster data (e.g., images or continuous spatial-temporal phenomena represented as arrays) with relational/vector data (points, lines, polygons) and linked data. Conventional VKG systems mediate OWL ontologies over relational databases via declarative R2RML mappings, translating SPARQL queries over the ontology into optimised SQL executions. However, they lack native support for the multidimensional nature of raster data and seamless integration with vector geometries, which limits their use in geospatial settings such as Earth Observation, Geographical Information Systems, and Urban planning. To overcome these limitations, we propose OntoRaster, a novel extended VKG framework that enables virtual integration and querying of raster, relational, and vector data. OntoRaster leverages domain-specific ontologies and supports on-the-fly query processing, thus minimising expensive data transfers, by delegating computations to specialised back-ends. For rigorous evaluation, we introduce comprehensive benchmarks to assess such VKG systems, i.e., OntoRaster on integrated raster-vector data processing workloads, using domain ontologies and synthetically generated, similar, scalable datasets that vary in properties (resolution, coverage, pixel density, polygons, etc.). Finally, we explore semantic enrichment for location-based business intelligence by exposing a SQL-accessible knowledge graph to BI tools. Additionally, we explore an LLM-based system for generating raster-based SPARQL queries based on ontologies from natural language queries. This empowers business experts to obtain actionable insights for location-based decision-making and risk assessment without requiring advanced geospatial expertise. Overall, this dissertation bridges Big Data, Semantic Web, AI and BI communities, advancing scalable, ontology-mediated access to heterogeneous multidimensional spatial-temporal geo data.

Sammanfattning

Moderna organisationer står inför en överväldigande mängd Big Data , som genereras kontinuerligt i hög hastighet, varierande format, från flera källor och lagras till en betydande kostnad. För att utvinna värde för AI-träning och avancerad analys måste denna heterogena data kopplas samman och bearbetas så att den kan läsas av maskiner. Traditionella pipelines för datahantering, såsom Extract-Transform-Load (ETL/ELT), materialiserar strukturerad och ostrukturerad data för frågebearbetning. Dessa metoder blir dock ineffektiva eller överbelastade när uppdateringar sker frekvent. Denna avhandling undersöker Virtual Knowledge Graphs (VKGs), även kända som Ontology-based Data Access (OBDA), som en virtualiseringsmetod för flexibel och effektiv hantering av storskalig strukturerad och ostrukturerad data utan materialisering. Vårt huvudsakliga fokus är integrationen av multidimensionell rasterdata (t.ex. bilder eller kontinuerliga rumsligt-temporala fenomen representerade som matriser) med relations- och vektordata (punkter, linjer, polygoner) samt länkad data. Konventionella VKG-system använder OWL-ontologier och relationsdatabaser sammankopplade med hjälp av deklarativa R2RML-mappingar, där SPARQL-frågor för ontologin översätts till optimerade SQL-exekveringar. Dessa system saknar dock inbyggt stöd för multidimensionell raster data och sömlös integration med vektorgeometrier. Detta begränsar deras användning i geospatiala tillämpningsområden såsom Earth Observation (EO), Geographical Information Systems (GIS) och stadsplanering. För att överkomma dessa begränsningar föreslår vi ett nytt utökat VKG-ramverk, OntoRaster,som möjliggör virtuell integration och frågebearbetning av raster-, relations- och vektordata. OntoRaster utnyttjar domänspecifika ontologier och stödjer bearbetning av frågor on-the-fly, vilket minimerar kostsamma dataöverföringar genom att delegera beräkningar till specialiserade backend-system. För en rigorös utvärdering av VKG-system såsom OntoRaster introducerar vi omfattande benchmarks med arbetslaster för integrerad bearbetning av raster-vektor-data. Dessa benchmarks använder domänontologier samt syntetiskt genererade, skalbara dataset med varierande egenskaper (upplösning, täckning, pixeltäthet, polygoner etc.). Slutligen undersöker vi metoder för semantisk berikning för platsbaserad business intelligence (BI) genom att exponera en SQL-åtkomlig knowledge graph för BI-verktyg. Dessutom utforskar vi ett LLM-baserat system för att generera rasterbaserade SPARQL-frågor från naturligt språk baserat på ontologier. Detta gör det möjligt för experter inom verksamheter att ta reda på handlingsbara insikter för platsbaserat beslutsfattande och riskbedömning utan krav på expertis med avancerade geospatiala verktyg Sammantaget bygger denna avhandling en bro mellan Big Data-, Semantic Web, AI och BI forskningsområdena genom att främja skalbar, ontologibaserad åtkomst till heterogen multidimensionell rumsligt-temporal geodata.

Sommario

Le imprese moderne affrontano un volume travolgente di big data — generati continuamente ad alta velocità, in formati eterogenei, provenienti da molteplici fonti e archiviati a costi significativi. Per estrarre valore ai fini dell'addestramento dell'AI e di analisi avanzate, tali dati eterogenei devono essere opportunamente collegati ed elaborati in forma leggibile dalle macchine. Le tradizionali pipeline di data ingestion, come Extract-Transform-Load (ETL/ELT), materializzano dati strutturati e non strutturati per l'elaborazione delle query; tuttavia, esse diventano inefficienti o sovraccariche quando gli aggiornamenti avvengono con elevata frequenza. Questa tesi studia i Virtual Knowledge Graphs (VKGs), noti anche come Ontology-based Data Access (OBDA), come approccio di virtualizzazione per la gestione flessibile ed efficiente di dati strutturati e non strutturati su larga scala senza materializzazione. L'obiettivo principale è l'integrazione di dati raster multidimensionali (ad esempio immagini o fenomeni spazio-temporali continui rappresentati come array) con dati relazionali/vettoriali (punti, linee e poligoni) e linked data. I sistemi VKG convenzionali mediano ontologie OWL su database relazionali tramite mapping dichiarativi R2RML, traducendo query SPARQL definite sull'ontologia in esecuzioni SQL ottimizzate. Tuttavia, tali sistemi non forniscono supporto nativo alla natura multidimensionale dei dati raster né un'integrazione fluida con le geometrie vettoriali, limitandone l'utilizzo in contesti geospaziali quali Earth Observation, Geographical Information Systems e pianificazione urbana. Per superare tali limitazioni proponiamo OntoRaster, un nuovo framework VKG esteso che consente l'integrazione virtuale e l'interrogazione congiunta di dati raster, relazionali e vettoriali. OntoRaster sfrutta ontologie specifiche di dominio e supporta l'elaborazione delle query on-the-fly, riducendo al minimo i costosi trasferimenti di dati mediante la delega dei calcoli a back-end specializzati. Per una valutazione rigorosa introduciamo benchmark completi per l'analisi di tali sistemi VKG, ovvero OntoRaster, su carichi di lavoro integrati di elaborazione raster-vector, utilizzando ontologie di dominio e dataset sintetici, comparabili e scalabili che variano in diverse proprietà (risoluzione, copertura, densità di pixel, poligoni, etc.). Infine, esploriamo l'arricchimento semantico per la business intelligence basata sulla localizzazione mediante l'esposizione di un knowledge graph accessibile tramite SQL agli strumenti BI. Inoltre, analizziamo un sistema basato su

LLM per la generazione di query SPARQL basate su raster a partire da interrogazioni in linguaggio naturale, sfruttando ontologie di dominio. Ciò consente agli esperti aziendali di ottenere insight operativi per il processo decisionale basato sulla localizzazione e la valutazione del rischio senza richiedere competenze geospaziali avanzate. Nel complesso, questa dissertazione crea un ponte tra le comunità Big Data, Semantic Web, AI e BI, promuovendo un accesso scalabile e mediato da ontologie a dati geospaziali eterogenei multidimensionali spazio-temporali.

সংক্ষিপ্তসার

আধুনিক প্রতিষ্ঠানসমূহ বিপুল পরিমাণ বিগ ডেটার সম্মুখীন, যা ধারাবাহিকভাবে উচ্চ গতিতে বিভিন্ন ফরম্যাটে, একাধিক উৎস থেকে অবিরত উৎপন্ন হচ্ছে এবং উল্লেখযোগ্য ব্যয়ে সংরক্ষিত হচ্ছে। AI প্রশিক্ষণ এবং অন্তর্দৃষ্টিপূর্ণ বিশ্লেষণের জন্য মূল্য আহরণ করতে হলে এই বৈচিত্র্যময় ডেটাকে সঠিকভাবে সংযুক্ত করে মেশিন-পাঠযোগ্য আকারে প্রক্রিয়াকরণ করা আবশ্যক। প্রচলিত ডেটা ইনজেশন পাইপলাইন গুলো (ETL/ELT) সংগঠিত এবং অসংগঠিত ডেটাকে ম্যাটেরিয়ালাইজ করে বিস্তৃত প্রক্রিয়াকরণের জন্য প্রস্তুত করে, কিন্তু এগুলো অদক্ষ বা অতিরিক্ত চাপগ্রস্ত হয়ে পড়ে যদি ডেটা প্রায়ই পরিবর্তিত বা আপডেট হতে থাকে। এই গবেষণা প্রবন্ধে ভার্চুয়াল নলেজ গ্রাফ (VKG), যা অন্টোলজি-ভিত্তিক ডেটা অ্যাক্সেস (OBDA) নামেও পরিচিত, যা একটি ভার্চুয়ালাইজেশন পদ্ধতি, যা ম্যাটেরিয়ালাইজেশন ব্যতিরেকে বৃহৎ পরিসরের সংগঠিত এবং অসংগঠিত ডেটা দক্ষভাবে পরিচালনা করতে সক্ষম। এর মূল লক্ষ্য হলো বহুমাত্রিক রাস্টার ডেটা (যেমন চিত্র বা অ্যারে হিসেবে উপস্থাপিত ধারাবাহিক স্থানীয়-সাময়িক ঘটনা) কে রিলেশনাল/ভেক্টর ডেটা (বিন্দু, রেখা, বহুভুজ) এবং লিঙ্কড ডেটার সাথে সংহতিসাধন করা । প্রচলিত VKG সিস্টেমগুলো R2RML ম্যাপিংয়ের মাধ্যমে রিলেশনাল ডাটাবেসের ওপর OWL অন্টোলজির মাধ্যমে, SPARQL প্রশ্নগুলোকে অপ্টিমাইজড SQL এক্সিকিউশনে অনুবাদ করে। তবে, এগুলোতে রাস্টার ডেটার বহুমাত্রিক প্রকৃতি এবং ভেক্টর জিওমেট্রিগুলোর নির্বিঘ্ন একত্রীকরণের জন্য নেটিভ সমর্থন নেই, যা পৃথিবী পর্যবেক্ষণ, ভৌগোলিক তথ্য ব্যবস্থা, এবং নগর পরিকল্পনা-এর মতো ভূ-স্থানীয় পরিবেশে VKG এর ব্যবহার সীমিত করে। এই সীমাবদ্ধতাগুলো কাটিয়ে ওঠার জন্য, আমরা প্রস্তাব করছি , একটি নতুন বিস্তৃত VKG ফ্রেমওয়ার্ক যা রাস্টার, রিলেশনাল এবং ভেক্টর ডেটার ভার্চুয়াল একত্রীকরণ এবং অনুসন্ধান সক্ষম করে। ডোমেইন-নির্দিষ্ট অন্টোলজি ব্যবহার করে এবং অন-দ্য-ফ্লাই প্রশ্ন প্রক্রিয়াকরণ সমর্থন করে, ফলস্বরূপ ব্যয়বহুল ডেটা স্থানান্তর হ্রাস ঘটে এবং অন্যদিকে গুরুত্বপূর্ণ গণনাগুলি বিশেষায়িত ব্যাকএন্ডে অর্পিত হয়। এর মতো VKG সিস্টেমগুলোর রাস্টার-ভেক্টর ডেটা সংযুক্ত প্রক্রিয়াকরণ ক্ষমতার উপযুক্ত এবং পরিপূর্ণ মূল্যায়নের উদ্দেশ্যে আমরা বিস্তৃত বেঞ্চমার্ক উপস্থাপন করছি। এতে ডোমেইন অন্টোলজি অনুযায়ী ভিন্ন ভিন্ন সিন্থেটিকভাবে তৈরি অনুরূপ স্কেলযোগ্য ডেটাসেট ব্যবহৃত হয়েছে, যেগুলো বিভিন্ন বৈশিষ্ট্যে পরিবর্তিত হয় (রেজোলিউশন, কভারেজ, পিক্সেল ঘনত্ব, মিশ্র বহুভুজ ইত্যাদি)। পরিশেষে, আমরা অবস্থান-ভিত্তিক ব্যবসায়িক বুদ্ধিমত্তা (BI)-এর জন্য অর্থগত সমৃদ্ধি অনুসন্ধান করেছি, যেখানে BI টুলগুলির জন্য একটি SQL অ্যাক্সেসযোগ্য নলেজ গ্রাফ উন্মুক্ত করা হয়েছে। একটি অতিরিক্ত বৈশিষ্ট্য হিসেবে, আমরা প্রাকৃতিক ভাষার প্রশ্ন থেকে অন্টোলজি ব্যবহার করে রাস্টার-ভিত্তিক SPARQL প্রশ্ন সৃষ্টি করার জন্য একটি LLM ভিত্তিক সিস্টেম তৈরি করেছি, যার মাধ্যমে ব্যবসায়িক বিশেষজ্ঞরা উন্নত ভূ-স্থানীয় দক্ষতার প্রয়োজন ছাড়াই অবস্থান-ভিত্তিক সিদ্ধান্ত গ্রহণ এবং ঝুঁকি মূল্যায়নের জন্য ব্যবহারযোগ্য অন্তর্দৃষ্টি অর্জন করতে সক্ষম হবে। সামগ্রিকভাবে, এই থিসিসটি বিগ ডেটা, সেম্যান্টিক ওয়েব, AI এবং BI বিষয়গুলো কে সংযুক্ত করে, এবং ভিন্নধর্মী ভূ-স্থানীয় বহুমাত্রিক ডেটা, অন্টোলজি-মধ্যস্থিত অ্যাক্সেস এর মাধ্যমে সম্পৃক্ত করে।

Acknowledgements

These past few years of PhD have been a true adventure of the heart and mind. I've grown in ways I never expected, faced challenges that taught me and discovered resolutions that shaped me, and learned so much along the way. Coming abroad for my PhD was not only a scientific venture, but also a rich cultural experience, thanks to the incredible people I've met from every corner of the world, as well as Sweden, a beautiful country with its own rich, inclusive culture. Looking back, my heart is full of gratitude for every single one of you who laughed with me, supported me, challenged me, and made this journey an amazing experience of my life. First, I extend my heartfelt thanks to my supervisor, Prof. **Diego Calvanese**, for his invaluable guidance and unwavering support throughout the PhD journey. Every discussion with you has been enlightening, continually pushing me to become a better researcher. Your clarity of thought and exceptional work ethic have always inspired me. Thank you sincerely for taking that leap of faith and believing in me, and for always being willing to work with me—even when your busy schedule often does not allow it. It meant a lot to me. Hopefully, my persistent nagging about the thesis checking wasn't too much trouble. I also want to thank the Artificial Intelligence for Data Management (AiDM) research group and its members, **Romuald, Divya, Cem, Younes**, and **Nazeer**, for their constant support, research feedback, and witty humour regarding PhD research and life.

I would also like to thank my co-supervisor, Prof. **Kai-Florian Richter**, for standing by me during challenging times and for his valuable insights regarding research and administrative work. I am also grateful to Prof. **Mantas Šimkus**, my former co-supervisor, for his support during the limited period we worked together. I am also grateful to Prof. **Frank Drewes**, my reference person, who provided me with a different lens through which to view my PhD study and research. Also, thanks to Prof. **Erik Elmroth** for his valuable advice on the research and working environment in Sweden. Beyond my research group, I would like to acknowledge the warm support and friendship of the members of the Department of Computer Science at the Free University of Bozen-Bolzano, Italy, Smart Brain Factory and Ontopic s.r.l., where I spent a significant amount of time, especially **Albulen, Marco, Benjamin, Mattia, Davide, Francesco, Alessandro**, and **Oliver**. Finally, I would like to thank my collaborators, **Guohui Xiao** at the University of Bergen, Norway, Prof. **Linfang Ding**, at the Norwegian University of Science and Technology (NTNU), Norway and the D2G2 project collaborators, such as Prof. **Liqiu Meng**, Dr. **Yu Feng**, etc., at the Cartography and Visual Analytics, TU Munich, Germany, for their valuable insights into my PhD work.

I would like to thank my wonderful colleagues from the Department of Computing Science, MIT-huset at Umeå University, especially **Luis, Julian, Maarten, Arvid, Adam, Timotheus, Kaan, Igor, Eren, Adwitiya, Anindya, Sudipta, Ayush, Saloni, Sargam**, and **Minal** for their wonderful company, engaging discussions, and delightful banter we had in the corridor and outside of work. My gratitude extends to the awesome and energetic people of the Computing Science Department, such as **Annakarin, Ennie, Hedwig, Heini, Anne-Lee, Mikael**, and the witty, humorous, and professional IT staff, **Tomas, Andreas, Mattias** and **Bertil**, who have always been very kind and helpful. A special thanks (though not enough) to **Monika** and **Ahmed** for their wonderful companionship, kindness, and deep conversations during the initial years of PhD, which made the journey so much more enjoyable and memorable.

I also express my gratitude to my former supervisors, Prof. **Anirban Mukhopadhyay** at Kazimierz Wielki University, Poland, Prof. **Tuhin Ghosh** at Jadavpur University, India, and Prof. **Sugata Hazra** at Jadavpur University, India, as well as Scientist/Engineer **Manu Mehta** at Indian Space Research Organisation (ISRO), Dehradun, India, and Prof. **S. Siva Sathya** at Pondicherry University, India, for their constant encouragement and background support over the past years.

My heartfelt gratitude goes to my **parents**, whose unwavering support, inspiration, sacrifice, and motivation have been instrumental to the success of my PhD story. **Mampa**, my constant motivator, partner-in-crime, and so much more, has been immeasurable in her enormous support during these tough years. Thank you for your unconditional love and unwavering belief in me. Also, to my extended family and friends, no words can fully express my gratitude for your endless love and support. Despite the miles between us (approx. 5000 miles), every call, message, and shared moment of connection with you gave me the strength to carry on.

Although there were many happy and positive moments during this PhD, it was not without difficulties. Nevertheless, despite what happened, I am glad that I always trusted my intuition and my passion for science and life. I have always wanted to share and discuss my scientific ideas with people around the world, and this became possible during my PhD journey. Ultimately, for all the wonderful moments such as badminton, beach volleyball, swimming at IKSU, barbecues, bowling, laser-tag, pool, fika breaks, and countless other events, big or small, which enriched my life and made me a better person, I thank you, **Umeå**.

Publications

Paper I	Arka Ghosh, Mantas Simkus, and Diego Calvanese. "**Semantic Querying of Integrated Raster and Relational Data: A Virtual Knowledge Graph Approach**". In: *Proceedings of the 17th International Rule Challenge and 7th Doctoral Consortium* @ RuleML+RR'23, Oslo, Norway, 18 - 20 September. 2023. [.pdf]
Paper II	Arka Ghosh, Albulen Pano, Guihui Xiao and Diego Calvanese. "**OntoRaster: Extending VKGs with Raster Data**". In: *Proceedings of the 8th International Joint Conference on Rules and Reasoning* @ RuleML+RR'24, Bucharest, Romania, 16 - 18 September. 2024. [.pdf]
Paper III	Arka Ghosh, Albulen Pano, Benjamin Cogrel, and Diego Calvanese. "**Semantic Enrichment of Location-based Business Intelligence using Virtual Knowledge Graphs**". In: International Semantic Intelligence Conference (ISIC 2025). Lubeck, Germany: Springer, 2025 [Accepted]
Paper IV	Arka Ghosh, and Diego Calvanese. "**The OntoRaster Benchmark: Evaluating VKGs Over Raster Data**". [Manuscript]

List of Figures

Nomenclature

$\mathcal{A}_{MDA}$	Multi-Dimensional Array	22
$\boldsymbol{a} = \{a_1, a_2, ..., a_M\}$	Set of Attributes	22
$\boldsymbol{d} = \{d_1, d_2, ..., d_N\}$	Set of Dimensions	22
$\mathcal{D}^{rel}$	Relational Database Instance over Schema	46
$\mathcal{J} = (\mathcal{P}, \mathcal{D}^{rel})$	VKG instance	46
$\mathcal{K} = (\mathcal{O}, \mathcal{M}(\mathcal{D}^{rel}))$	Knowledge Base	44
$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$	Knowledge Base in terms of *TBox* & *ABox*	46
$\mathcal{M}$	Mapping	45
$\mathcal{M}(\mathcal{D}^{rel})$	Knowledge Graph	46
$\mathbb{R}$	Real Numbers	22
$\mathbb{Z}$	Integers	22
$\mathcal{O}$	Ontology	45
$\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$	VKG specification	45
$\mathcal{R}$	Generic Raster Data	22
$\mathcal{S}$	Data Source Schema	45
lb_i	Lower Bound	22
N	Dimension of Raster Data	22
$Q_{sql}(\vec{x})$	Source SQL query over $\mathcal{S}$ with answer variables $\vec{x}$	45
ub_i	Upper Bound	22

Acronyms

AI Artificial Intelligence. 3, 8, 12, 39, 40, 124

AOI Area of Interest. 5, 64, 70, 109, 111

ARD Analysis Ready Data. 125

ArrayDBMS Array Database Management System. 24

ATKIS Authoritative Topographic-Cartographic Information System. 52

BI Business Intelligence. 8, 123

BIM Building Information Modelling. 13, 124

BPMN Business Process Model and Notation. 7

BSBM Berlin SPARQL Benchmark. 108

C2RML Coverage to RDF Mapping Language. 51

CIS Coverage Implementation Schema. 23, 67

CityGML City Geography Markup Language. 19, 67

CityJSON City-JavaScript Object Notation. 67

CoverageJSON Coverage-JavaScript Object Notation. 67

CQ Conjunctive Query. 14, 45

CRS Coordinate Reference System. 16, 61, 67, 112

DBMS Database Management System. 14, 15

DIKW Data-Information-Knowledge-Wisdom. 11, 12

DLM Digital Landscape Model. 52

DLP Description Logic Program. 36

DLs Description Logics. 43

DSR Design Science Research. 7

ECOSTRESS ECOsystem Spaceborne Thermal Radiometer Experiment on Space Station. 73

ELT Extract-Load-Transform. 124

EO Earth Observation. 13, 23, 24, 38, 50, 51, 123, 125

EPSG European Petroleum Survey Group. 16, 112

ESA European Space Agency. 52, 73

ETL Extract-Transform-Load. 50, 124

FDW Foreign Data Wrapper. 51

FOL First-Order Logic. 14, 44

FUA Functional Urban Area. 5

GADM Global Administrative Areas. 17, 38

GDAL Geospatial Data Abstraction Library. 54

GeoBI Geospatial Business Intelligence. 10

GeoKG Geospatial Knowledge Graph. 38

GEOS Geometry Engine Open Source. 61

GeoSPARQL Geographic Query Language for RDF Data. 16, 67, 127

GeoTIFF Geographic Tagged Image File Format. 23, 67

GIS Geographic Information System. 13, 16, 29, 38, 124

GML Geography Markup Language. 36

GPS Global Positioning System. 16

HDF5 Hierarchical Data Format 5.0. 23

IRI Internationalized Resource Identifier. 32, 33, 46

ISO International Organization for Standardization. 67

KB Knowledge Base. 44

KG Knowledge Graph. 3, 37

KR&R Knowledge Representation & Reasoning. 3, 9, 39

LGD LinkedGeoData. 19, 26, 38, 103

LLMs Large Language Models. 128, 134

LOD Linked Open Data. 25, 26

LoD Levels of Detail. 19

LoRA Low-Rank Adaptation. 130

LUBM Lehigh University Benchmark. 107

MBR Minimum Bounding Rectangle. 111

MDA Multi-Dimensional Arrays. 22, 55

MDD Multidimensional Discrete Data. 21

MKG Materialized Knowledge Graph. 122

MODIS Moderate Resolution Imaging Spectroradiometer. 73–75

NASA National Aeronautics and Space Administration. 13, 73

NDSI Normalized Difference Snow Index. 73

NDVI Normalized Difference Vegetation Index. 73, 98, 99

NetCDF Network Common Data Form. 23, 67

OBDA Ontology Based Data Access. 45

ODC Open Data Cube. 125

OECD Organisation for Economic Cooperation and Development. 5

OGC Open Geospatial Consortium. 16, 36, 38, 67, 107, 127

OGE Open Geospatial Engin. 24

OLAP Online Analytical Processing. 125

OSM OpenStreetMap. 17, 26, 38, 72, 74

OWL Web Ontology Language. 25, 35, 107

OWL2 OWL 2 Web Ontology Language. 35, 67

PEFT	Parameter-Efficient Fine-Tuning. 130
PL/pgSQL	Procedural Language/PostgreSQL. 53
PL/Python	Python Procedural Language. 53
POI	Point of Interest. 102, 103
QUDT	Quantities, Units, Dimensions, and Types. 59
R2RML	RDB to RDF Mapping Language. 45, 46, 67
RAG	Retrieval-Augmented Generation. 134
RasDaMan	Raster Data Manager. 25
RDBMS	Relational Database Management System. 15, 16, 46, 48, 50, 53, 102
RDF	Resource Description Framework. 32, 67
RDFS	Resource Description Framework Schema. 34, 58, 67
SFA	Simple Feature Access. 67
SPARQL	SPARQL Protocol And RDF Query Language. 36, 67
SRS	Spatial Reference System. 54, 71
SRTM	Shuttle Radar Topography Mission. 73–75
UCQ	Union of Conjunctive Queries. 14, 47
UCUM	Unified Code for Units of Measure. 59
URI	Uniform Resource Identifier. 32, 59
URL	Uniform Resource Locator. 32
URN	Uniform Resource Name. 32
UTM	Universal Transverse Mercator. 113
VGI	Volunteered Geographic Information. 17
VIG	Virtual Instance Generator. 108
VKG	Virtual Knowledge Graph. 6, 45, 47, 49
W3C	World Wide Web Consortium. 31, 35, 38, 67

WCPS Web Coverage Processing Service. 55

WCS Web Coverage Service. 29, 55

WFS Web Feature Service. 29, 55

WGS84 World Geodetic System 1984. 16, 26, 28, 61

WKB Well-Known Binary. 54, 62, 71

WKT Well-Known Text. 15, 16, 19, 36, 54, 61–63, 67, 78, 112

WMS Web Map Service. 55

YASGUI Yet Another SPARQL Graphical User Interface. 126

Contents

Chapter 1

Introduction

We are overwhelmed with information in the current era of prolific advancements of Artificial Intelligence (AI) and big data. Data is generated and collected at an unprecedented scale, in myriad formats from heterogeneous sources, often at considerable expense for storage and maintenance. Yet, despite its inherent value, this deluge rarely yields actionable insights without adequate mechanisms for comprehension and interpretation. This thesis explores the paradigm of *Virtual Knowledge Graphs* (VKG), also known as *Ontology-based Data Access* (OBDA), with a particular emphasis on enhancing their ability to query vast, heterogeneous big datasets—such as multidimensional raster data alongside relational data, including vector data. Building on foundational advances in knowledge representation and semantic technologies, this work confronts the shortcomings of conventional relational databases in managing semantically rich geospatial information. Within the broader landscape of AI, the structuring of information in a machine-readable form is pivotal to the efficacy of intelligent systems and their emergent behaviours. Knowledge Representation & Reasoning (KR&R) is a foundational domain in AI research that focuses on encoding real-world information in a symbolic, machine-interpretable format. A robust logical formalism supports this format, facilitating human-like reasoning and intelligent behaviour in addressing complex challenges across various industrial domains, such as medicine, engineering, and business. However, the colossal growth in data generated from multiple sources, in diverse formats, along with their metadata, presents significant challenges for data administration and processing. Virtual Knowledge Graphs (VKGs), part of Semantic Web Technology (or Web 3.0), address these challenges by harmonising heterogeneous data with embedded knowledge to derive novel inferences. At their core lies the *ontology*, which endows the underlying data sources with semantic meaning. Instances of ontologies manifest as *knowledge graphs (KGs)*, wherein data is organised as directed graphs: *nodes* denote domain entities or literal values, while *edges* capture their attributes or inter-relationships. This graph-based structure not only facilitates intuitive traversal and querying but

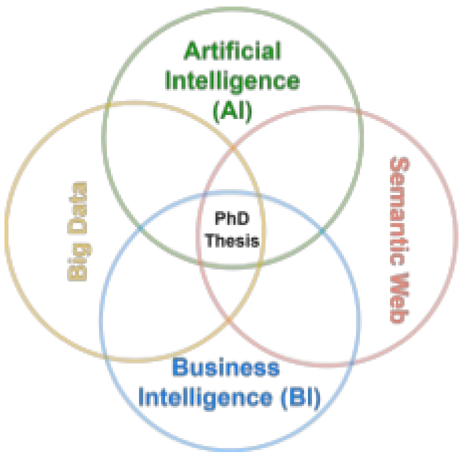


Figure 1.1: Big Picture of the thesis

also supports automated reasoning to uncover implicit knowledge, bridging the gap between raw data and domain-specific intelligence. The big picture of the thesis is displayed in Figure 1.1.

1.1 Motivation

Scientific and business applications produce data in different representations at a massive scale, leading to bottlenecks in data management and processing [6]. This gives rise to the notorious issue of data heterogeneity [203, 210], which includes *structural* (e.g., schemas), *syntactic* (e.g., formats), and *semantic* (e.g., vocabularies).

- ***Structural Heterogeneity***: Refers to differences in the organisation, schema, or data models of information sources, such as variations in table structures, attribute groupings, relationships, or hierarchies, even when the underlying data describe similar concepts [71].

- ***Syntactic Heterogeneity***: Concerns differences in the data formats, encodings, or representations used to store and exchange information [187].

- ***Semantic Heterogeneity***: involves differences in meaning, interpretation, or conceptualisation of data across variable sources, including the use of different vocabularies, terminologies, or ontologies to describe the same or related real-world entities [203].

The motivation for this research is to mitigate data heterogeneity by efficiently integrating and flexibly querying large-scale structured, semi-structured, and unstructured data across domains, especially geospatial data

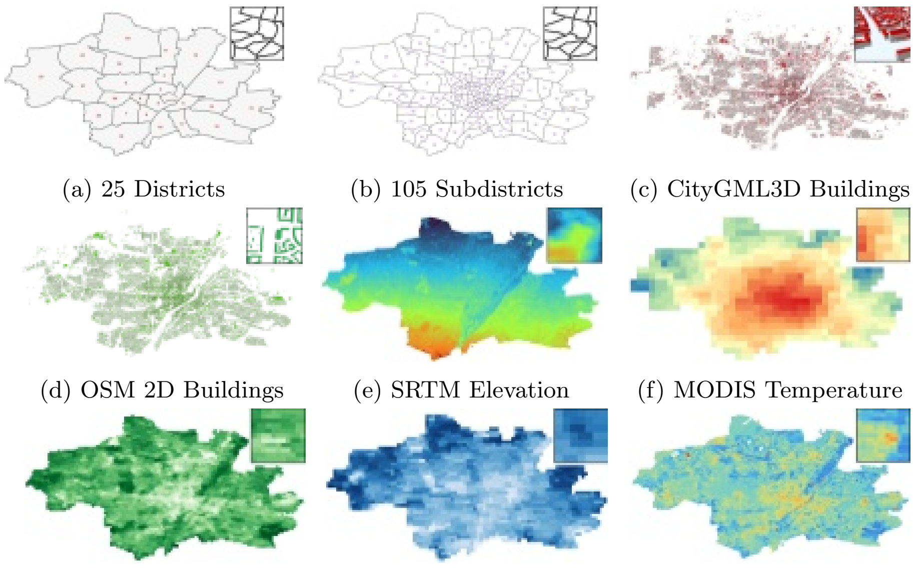

(a) 25 Districts (b) 105 Subdistricts (c) CityGML3D Buildings
(d) OSM 2D Buildings (e) SRTM Elevation (f) MODIS Temperature
(g) MODIS NDVI (h) NDSI Snow Cover (i) ECO Soil Moisture

Figure 1.2: Heterogeneous geospatial data over Munich

enriched with semantic components under the VKG paradigm. VKGs offer a robust framework where an ontology, typically specified in the standard language OWL 2 QL, serves as a conceptual layer exposed to users for querying. The actual data resides in relational databases and is connected to the ontology via mapping assertions that define how classes and properties are populated from these sources. Traditional VKG research has emphasised data integration and query answering over plain relational data. However, many real-world large datasets possess inherent semantics that require specialised handling. This applies specifically to raster data, which we concentrate on in this thesis.

Geospatial data exemplify this; indeed, their semantic heterogeneity is a significant impediment to integration and sharing. The effective integration and extensive dissemination of geospatial data are fundamental prerequisites for advancing research and applications in geographic information science. To exemplify the practical significance and challenges, we choose a specific area of interest (AOI), namely Munich, the capital of Bavaria, Germany, along with its adjacent functional urban area (FUA) as delineated by EU-OECD [66], which denotes densely populated urban centres interconnected with the surrounding labour market ('commuting zone').

Imagine a real-world scenario in which a public representative of Munich is responsible for efficient urban planning, including the management of residential, commercial, educational, medical, and other essential public amenities to enhance the quality of life for

Munich's taxpaying citizens. The public representative now has various types of data on Munich, as shown in Figure 1.2. The goal is to gather additional information by querying the data to inform decision-making for Munich's final urban plan for its citizens. For a policymaker skilled in politics but not an expert data scientist, formulating queries can be daunting without a deep knowledge of the data schemas, making it challenging to craft them. We consider relational data and its spatial geometric version, known as *vector data*, which encompasses administrative district and sub-district boundaries, 3D building data, and OSM 2D building footprints, as shown in Figure 1.2(a-d). Moreover, we have large, multidimensional *raster data* depicting space-time-varying continuous phenomena, e.g., temperature, vegetation, snow cover and soil moisture, as shown in Figure 1.2(e-i). In general, there may be 100 or more distinct vector and raster datasets covering any area of interest. The public representative of Munich may not even be aware of these vast datasets, making it difficult to query them.

Enter the *Virtual Knowledge Graph* (VKG) paradigm, which addresses data heterogeneity by relying on an *ontology* to expose domain knowledge (e.g., medical, robotics, or geospatial), in a conceptually coherent manner, while abstracting the technical schema-level intricacies of the underlying relational *data sources*. The ontology is 'connected' to the data via *mappings*, which a VKG engine Ontop [20] uses to automatically translate queries posed over the ontology into queries against the underlying database management system. The primary issue is that traditional VKGs support only relational databases and do not natively support raster data and related functionality. Furthermore, raster data pose significant challenges for conventional data frames and relational databases in terms of efficient management and querying. Their multidimensional, heterogeneous, and intricate nature complicates access for users in business sectors, including analysts, policymakers, and managers, who lack the necessary domain expertise. In the relevant literature, the issue has been addressed only minimally so far.

1.2 Research Questions

This proposed doctoral research combines foundational and systems-oriented contributions to the scalable, semantics-aware management of large-scale heterogeneous data through the Virtual Knowledge Graphs (VKG) paradigm. This thesis aims to advance the integration of geospatial data, particularly multidimensional raster data, into VKGs (henceforth, GeoVKGs) and to develop sophisticated techniques for ontology-mediated visual query formulation and the visual explanation of query results over such enriched graphs. Whereas the traditional VKG framework primarily addresses query answering over relational sources via ontologies and declarative mappings, geospatial data intro-

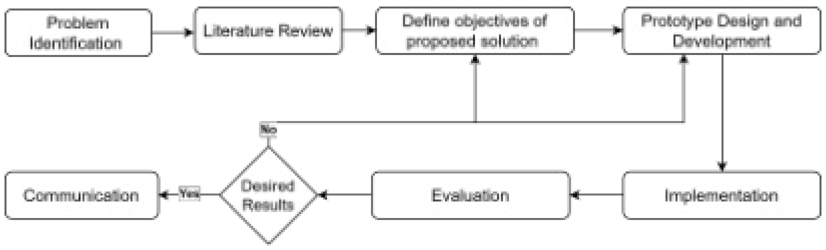


Figure 1.3: Design Science Research Methodology

duces specialised semantics, data models (vector and raster), and operations that demand explicit system-level support. Contemporary VKG engines have taken initial steps in this direction by supporting GeoSPARQL and translating to native extensions, such as PostGIS. This thesis seeks to substantially extend these capabilities by systematically addressing the following research questions:

RQ1: *What is an effective architecture for the integration of different data formats relevant to geospatial data, especially, raster data, general relational data, geometric vector data and linked data using the VKG paradigm?*

RQ2: *How can we devise effective query plans that guide the structuring and efficient processing of spatial queries over integrated data?*

RQ3: *How can we support the formulation of rich geospatial queries over the integrated datasets through intelligent, context-aware, reasoning-supported visual interfaces?*

RQ4: *How can the developed techniques and prototype system be rigorously evaluated using diverse, real-world datasets that vary in structural complexity, spatial distribution, and semantic density?*

RQ5: *How can the developed techniques and prototype system be used in spatial business intelligence?*

1.3 Methodology

This thesis adheres to the Design Science Research (DSR) [47] methodology, which has been extensively utilised in information systems, computer science, and engineering since its inception in 1969 [206], as illustrated in Figure 1.3 using a Business Process Model and Notation (BPMN) [228] diagram. DSR provides a structured framework for addressing complex research challenges by developing and evaluating innovative artefacts. These artefacts, which take the form of models, frameworks, and algorithms, are designed to address research challenges while advancing theoretical understanding and practical application. Here, we outline the application of the DSR methodology in our context.

We begin by identifying a significant problem in our research domain, which lies at the intersection of Big Data, Artificial Intelligence (AI), the Semantic Web (SW), and Business Intelligence (BI), as depicted in Figure 1.1. Our research objectives are defined as research questions (cf. section 1.2) which are focusing on achieving seamless integration and query processing of heterogeneous multidimensional data while maintaining virtualisation within the VKG paradigm. Next, we enter the *Design and Development* phase, where we devise novel solutions to address the identified problem. These solutions may include new software, algorithms, or innovative processes. Through an iterative design process, we create an initial version of the artefact, test it, evaluate its strengths and weaknesses, and refine it accordingly. Understanding both successes and failures is crucial in research, as failure models are as valuable as successful outcomes. In the *Evaluation* phase, we assess our artefact by testing its performance on publicly available datasets, measuring metrics such as speed and accuracy, and evaluating its effectiveness relative to existing solutions. Additionally, we develop theories to explain the artefact's functionality and its significance. Finally, the resulting work and its findings are encapsulated in scientific articles and submitted to well-known journals and conferences in the relevant community. Ultimately, our goal is to ensure that our work is precise, reproducible, and valuable to others.

1.4 Thesis Contributions

This thesis aims to deliver several key contributions to the field of knowledge graphs and geospatial data management:

- **Novel VKG Framework for Heterogeneous Raster Data.** The thesis proposes OntoRaster, an extended VKG architecture that integrates multidimensional raster data with relational (including vector data) and linked data within a single semantic framework. By connecting ontology-based data access technology with an array DBMS, raster data can be queried directly in its native multidimensional representation, avoiding costly transformations and enabling domain-agnostic data integration.

- **Federated Semantic Query Processing with Raster Support.** A novel query processing approach is introduced through *RasSPARQL*, an extension of SPARQL supporting raster operations. A federated execution pipeline coordinates Ontop, PostgreSQL, and RasDaMan to decompose, route, and combine queries across heterogeneous storage systems, enabling efficient spatial and spatiotemporal query answering over integrated datasets.

- **Semantic Querying and Visualisation of Integrated Geo-enriched-Knowledge Graphs.** The thesis demonstrates expressive querying and visual analysis over combined raster and vector datasets, including OpenStreetMap, CityGML, and GeoNames. The proposed VKG framework supports reasoning-driven geospatial exploration through interpretable tabular results and geo map-based visualisations.

- **Benchmarking Framework for raster-based VKG Systems.** A comprehensive evaluation methodology is developed for VKG systems handling raster–vector integration. The work introduces benchmark criteria, synthetic OGC-compliant datasets, and controlled performance experiments, providing reusable resources and empirical insights into scalability and query efficiency.

- **Semantic Enrichment in Geospatial Business Intelligence.** The research bridges semantic technologies and business analytics by integrating OntoRaster, with Business Intelligence tools such as MS Power BI via semantic SQL interfaces. Furthermore, an LLM-based retrieval-augmented generation (RAG) approach enables natural language queries to be translated into executable RasSPARQL, improving accessibility to semantic geospatial analytics for business executives.

These contributions advance the current state of the art in the Virtual Knowledge Graph (VKG) paradigm for query answering over multidimensional raster data, and provide practical methods for real-world applications. The updated OntoRaster framework is available on Github[1].

1.5 Thesis Outline

The rest of the thesis is organised as follows,

- **Chapter 2** provides a detailed introduction to the heterogeneous data used in this thesis, including relational data (including vector data), linked data, and raster data.

- **Chapter 3** explores the semantic web (SW) and big data technologies related to this thesis. First, the traditional Semantic Web (SW) stack is presented, along with its main components. Then, advanced SW technologies are described, including knowledge graphs, ontologies, KR&R, and the VKG paradigm for semantic data integration.

- **Chapter 4** presents the main contribution of the thesis, OntoRaster, a novel framework built on top of traditional VKG to support querying and managing raster data ingestion alongside relational data. It addresses two primary research questions **RQ1** and **RQ2**.

[1] https://github.com/aghoshpro/OntoRaster.git

- **Chapter 5** outlines practical application of OntoRaster. This chapter presents various spatial-temporal RasSPARQL queries posed over heterogeneous datasets across multiple areas of interest, with varying levels of complexity and query results, utilising the OntoRaster framework. This addresses **RQ3**.
- **Chapter 6** provides a benchmark comprising 10 parameters to test OntoRaster across different raster datasets and synthetically generated vector geometries. This addresses **RQ4**.
- **Chapter 7** outlines approaches to utilise semantic query results from OntoRaster within BI tools to produce semantically enriched business information through map-based visualisations, contributing to *Geospatial Business Intelligence* (GeoBI) and **RQ5**. Furthermore, we present Ontopic Suite's Semantic SQL interface to enable direct querying of KGs from BI tools, and we propose enhancing the SQL interface with raster functions to support querying integrated raster and relational data within OntoRaster.
- **Chapter 8** concludes the thesis by summarising the key findings and discussing potential directions for future research.

Chapter 2

Heterogeneous Data Management

In computer science, how to process and manage data is studied in various disciplines, ranging from Databases, to Abstract Data Types, and Data Science. Data are converted into information when they are processed, e.g., to transfer them from one entity to another, and the receiving entity will conceptualise that information as new knowledge. Overall, this enables significant advancements across diverse domains, including decision-support applications. The data-to-knowledge pipeline is hindered by the prevalence of heterogeneous data, which refers to diverse types of raw data generated from multiple sources at volumes and velocities beyond human comprehension. This heterogeneity complicates effective data management and underscores the need for appropriate data integration techniques to support the transformation of data into higher levels of information, knowledge, and ultimately wisdom.

2.1 Data – Information – Knowledge – Wisdom

The Data-Information-Knowledge-Wisdom (DIKW) hierarchy is a conceptual model introduced by Ackoff et al. [2] which delineates the progressive transformation of raw data into meaningful information, actionable knowledge, and ethical wisdom. Originating from early philosophical and literary sources, with an initial reference in T.S. Eliot's poem "The Rock" (1934), which laments the loss of wisdom amid abundant knowledge and information [238]. The DIKW framework, often visualised as a pyramid as shown in Figure 2.1, which illustrates how data evolves through contextualisation, application, and evaluation to culminate in wisdom [81]. According to Ackoff [2], *data* consist of symbols that represent the properties of objects, events, and their environments; they are raw and lack inherent meaning or utility until they are processed. *Informa-*

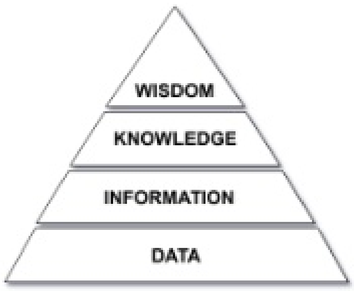


Figure 2.1: DIKW Pyramid

tion emerges when data are processed to become useful, providing answers to *who, what, where*, and *when* questions through description and contextualisation. *Knowledge* is the application of data and information to achieve efficiency, answering "how-to" questions by transforming sources or processes towards desired outcomes. Finally, *wisdom* involves evaluated understanding, focusing on increasing effectiveness by addressing "why" questions; it is an extrapolative, non-deterministic process that integrates prior levels to apply insights in novel contexts with ethical and long-term considerations.

The DIKW hierarchy models the progression of cognitive value in systems thinking, with direct implications for Computer Science, Information Science, and Artificial Intelligence (AI) [172]. In those domains, data are raw symbols representing object properties (e.g., sensor readings stored in databases). Information emerges from processed data, providing descriptive utility (e.g., aggregated stats in queries). Knowledge involves applying instructions to achieve efficiency (e.g., algorithms that transform information into decisions). Wisdom adds evaluative judgment regarding effectiveness, incorporating values and ethics (e.g., AI systems that assess long-term impacts).

In our research, this framework guides the handling of diverse data sources, promoting scalable systems that convert raw data into actionable knowledge for academic and enterprise contexts.

2.2 Challenges of Big Data

Big Data refer to extensive, intricate, and heterogeneous datasets that continuously expand in size and diversity. These datasets originate from sources such as sensor data, IoT devices, social media, transactional records, geographical location information, and public databases, posing challenges for traditional processing and analytical techniques [125, 201]. Various studies have conventionally characterised Big Data by Gartner's 3 V's [144]— *volume, velocity*, and *variety*— expanded to 5 V's [221] with *veracity* and *value*, and further extended to 7 V's [89] with *visualisation* and *variability*, with claims of up to

51 V's [131].

We focus particularly on the efficient management of geospatial data in Geographic Information System (GIS) [88], which requires understanding Earth as an integrated system and necessitates examining a wide range of phenomena. The massive utilisation of geo-referenced datasets across various scientific and industrial domains, such as Earth Observation [217], environmental sciences [134], urban planning [10], building information modelling (BIM) [114], real-time processing [84], and geospatial data analytics [231], renders geospatial data management an increasingly pivotal component in modern big data processing [118]. Recent methodologies employ open data platforms and containers to efficiently manage geospatial raster and vector data. However, geospatial data management entails the laborious task of integrating data from diverse sources [44].

Geospatial data are typically categorised into two fundamental structures: *vector* data and *raster* data. Vector data employs *points, lines, polygons*, and their combinations, originating from Euclidean geometry, to represent geographical features, facilitating the precise delineation of location, length, area, and other attributes of geographical entities. In contrast, raster data is organised into uniform grids, with each grid cell assigned at least one attribute value to represent continuous geographical phenomena within a spatial extent [120]. The challenges of geospatial big data are amplified in case of integration of raster and vector data, as they both present unique hurdles due to their scale, heterogeneity and complexity. This fits perfectly with the five Vs of big data, i.e., *volume, velocity, veracity, variety, and value* and our research's primary goal. For instance, the exponential growth of geospatial raster data, driven by advancements in data acquisition technologies and high-precision simulations, has led to datasets reaching petabyte scale [186]. The Landsat archive alone holds over five million raster images comprising 5 PB, and NASA is projected to host approximately 350 PB of climate data by 2030 [118]. This immense *volume* poses grand challenges for efficient data management solutions that can handle massive, multi-dimensional raster arrays without significant performance degradation.

In terms of *velocity*, geospatial data generation is rapid and continuous. Satellites produce several terabytes of raster data daily, and the archived volume is expected to reach the zettabyte scale soon [205]. Vector data, such as trajectories from GPS-equipped devices like smartphones, also contribute to this data explosion, necessitating real-time processing to manage streaming sensor and IoT data. *Variety* in geospatial big data arises from diverse formats, coordinate systems, resolutions, and sources. Integrating these heterogeneous sources is cumbersome, as coordinate transformations and alignment are required for accurate spatial queries [118]. For example, combining raster elevation data with vector building models in BIM/GIS integration requires handling diverse data models, which poses challenges for data fusion and analysis [44]. *Veracity* concerns the quality and reliability of geospatial data. Issues such as sensor inaccuracies, incomplete datasets, or errors in data conversion between raster and vector formats can introduce uncertainties [118]. Traditional relational databases are ill-suited for managing multi-dimensional geospatial data, prompting the use of array databases [118].

2.3 Types of Big Data

Different types of big data sources fall under three major categories: *structured*, *semi-structured* and *unstructured* data. Representation of all these types requires different data structures, data models, databases and ultimately different database management systems (DBMS) to manage and query this data adequately. In this dissertation, we are concerned with three types of data: structured data (i.e., relational tabular data), semi-structured data (i.e., RDF triples, JSON, and XML), and unstructured data (e.g., multi-dimensional arrays and raster data). Let's begin with the classic relational model, which has been a dominant data structure in data-driven enterprises for many years.

2.3.1 Relational Data

Relational data are represented as a collection of two-dimensional tables, or relations, each consisting of rows (or records or tuples) and columns (or attributes), organised under a *schema* that serves as a blueprint for the database. A relational database *schema* is a finite set $\mathcal{S} = \{r_1/n_1, \ldots, r_k/n_k\}$ of relation schemas, where each r_i is a predicate name of arity n_i. A database instance $\mathcal{D}$ over $\mathcal{S}$ maps each predicate r/n in $\mathcal{S}$ to an n-ary relation, denoted r^D. An atom for r/n has the form $r(t_1, \ldots, t_n)$, or simply $r(t)$, where each t_j is a term, which can be a constant from $\mathbf{N}_I$ or a variable. If all t_j's are constants, the atom is called ground, or simply a tuple. A first-order logic (FOL) formula over a relational schema $\mathcal{S}$ is constructed by using the relation names in $\mathcal{S}$, the equality predicate $=$, and the constants in $\mathbf{N}_I$. A formula ϕ with free variables x is denoted $\phi(x)$, and is also called a (relational) *query* with *answer* variables x. The formula is closed when x is empty. The closed formula obtained from $\phi(\boldsymbol{x})$ by replacing every variable in $\boldsymbol{x}$ by the corresponding constant $\boldsymbol{c}$ is denoted $\phi(\boldsymbol{c})$. For an instance D over $\mathcal{S}$, we use $D \models \phi(\boldsymbol{c})$ to denote that $\phi(\boldsymbol{c})$ holds in D. For a closed formula ϕ and a set Σ of closed formulae over $\mathcal{S}$, we use $\Sigma \models \phi$ to indicate that ϕ is implied by Σ, i.e., ϕ holds in all instances in which all formulae of Σ hold.

A *conjunctive query* (CQ) $Q(\boldsymbol{x})$ over the schema $\mathcal{S}$ is a query defined by a formula of the form $\exists \boldsymbol{y}.\phi(\boldsymbol{x}, \boldsymbol{y})$, where $\phi = r_1(\boldsymbol{t_1}) \wedge \cdots \wedge r_n(\boldsymbol{t_n})$ is a conjunction of atomic formulae whose variables belong to $\boldsymbol{x} \cup \boldsymbol{y}$. The variables y are called existential variables. We also express such a CQ as a logical rule of the form $Q(\boldsymbol{x}) \leftarrow r_1(\boldsymbol{t_1}), \ldots, r_n(\boldsymbol{t_n})$, where $Q(\boldsymbol{x})$ is called the head and $r_1(\boldsymbol{t_1}), \ldots, r_n(\boldsymbol{t_n})$ the body of the rule. A *union of conjunctive queries* (UCQ) is a finite disjunction of CQs with the same answer variables, also represented as a finite set of rules with the same head.

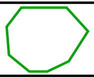
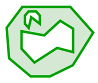
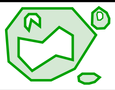
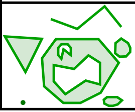
OGC Geometry Type	Region Geometry Illustration	Exterior & interior boundaries	OGC WKT Literal Representation (i.e. regionWkt)	Description
Point		**NA**	**POINT** (x, y)	A WKT point (e.g. pin location)
LineString		**NA**	**LINESTRING(**POINT$_{01}$,POINT$_{02}$,..... ..,POINT$_{m}$ **)**	A LineString with at least two or more points (e.g., road network)
LinearRing		**exterior = 1** **interior = 0**	**LINEARRING (** POINT$_{01}$,POINT$_{02}$,.... .....,POINT$_{n}$, POINT$_{01}$**)**	An enclosed LineString where start point = end point with zero measurable area
Polygon		**exterior = 1** **interior = 2**	**POLYGON** ((LINEARRING$_{01}$), [(LINEARRING)]*****)	A LinearRing with a valid measurable area with zero or more holes or interiors (e.g., countries, lake within forest)
Multi-Polygon		**exterior = 3** **interior = 3**	**MULTIPOLYGON** (((POLYGON$_{01}$), [(POLYGON)]*****))	Collection of two or more Polygons with zero or more interiors or holes (e.g., scattered islands, enclaves)
Geometry-Collection		**exterior = 4** **interior = 2**	**GEOMETRYCOLLECTION**([POINT*], [LINEARSTRING*], [POLYGON*], [MULTIPOLYGON*])	Heterogeneous collection of every OGC standard geometries

Figure 2.2: Elements of vector data in their respective OGC WKT forms

Relational Database Management System (RDBMS) have been the dominant DBMS for the storage, retrieval, and management of relational data since their inception by E.F. Codd (1970) [59].

2.3.2 Vector Data

Vector data or *geometrical* relational data represent the spatial characteristics of discrete real-world phenomena, with each phenomenon modelled as a feature in accordance with the *ISO 19123 Geographic Inf. Part 1: Fundamentals* standard[1]. Typical examples of such discrete features are rivers, lakes, and administrative regions. These are represented by a set of one or more geometric primitives, such as *points*, *curves*, and *surfaces* with their positional attributes either in Cartesian coordinates (X and Y) in a topological planar surface or geographic coordinates (*longitude* a.k.a. *geoX* and *latitude* a.k.a. *geoY*) with additional feature attributes.

We consider vector geometries that conform to the *OGC ISO 19125 OpenGIS* Standard[2], which foresees *Well-Known Text* (WKT) literals[3] as the most common representation of geometries. According to this standard, a *Point* is a pair of longitude and latitude (which in the WKT representation are separated by blanks). In contrast, a *LineString* is a sequence of points separated by '**,**'. A valid *Polygon* is a topologically closed planar surface defined by one

[1] https://www.iso.org/obp/ui/en/#iso:std:iso:19123:-1:ed-1:v1:en
[2] https://www.ogc.org/standard/sfa/
[3] https://docs.ogc.org/is/18-010r7/18-010r7.html

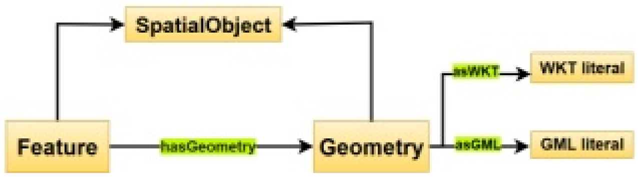


Figure 2.3: GeoSPARQL ontology (simplified UML representation)

exterior boundary and zero or more interior boundaries (where each interior boundary defines a hole in the polygon). Each boundary is a *LinearRing*, which is a LineString in which the last point must coincide with the first point. A *MultiPolygon* is a collection of one or more valid polygons, separated by ','. Finally, a *GeometryCollection* is a sequence of one or more of the previously introduced elements. Figure 2.2 illustrates some of the elemental geometries of an arbitrary region of vector data with their representation as OGC WKT literals (i.e., `regionWkt`) with examples. All geometries in OGC ISO 19125 are a combination of these elemental geometries. *Part 2: SQL option* of the OGC ISO 19125[4] defines an implementation in the relational query language SQL of geometric data types and a set of geometric predicates as SQL functions. Vector data are represented in popular file formats[5] such as `.csv`, ESRI shapefiles (`.shp`) [76], `.geojson` [50], `.geopackage` or can be connected to RDBMS such as PostgreSQL, OracleDB, etc. through spatial extensions such as PostGIS, Oracle Spatial, etc.

Note that, in the GIS community, a prevalent ambiguity exists over the sequence of *latitude* and *longitude*, as it is often unclear which coordinate is designated as the first and which as the second [38]. This confusion is further emphasised by the two most popular, widely used coordinate reference systems (CRSs) that refer to the World Geodetic System 1984 (WGS84), such as CRS:84[6] and EPSG:4326[7]. CRS:84 uses the coordinate order (*longitude, latitude*), the default CRS for Open Geospatial Consortium (OGC) standards such as WKT literals, GeoJSON, PostGIS, etc., whereas EPSG:4326 uses the coordinate order (*latitude, longitude*) based on the Earth's centre of mass, used by the Global Positioning System (GPS), Google Maps, or Leaflet Maps [38]. We consider both variants in our study, although the first is more prevalent.

The OGC approved GeoSPARQL standard introduces spatial data extensions to the Semantic Web. GeoSPARQL's vocabularies and definitions are contained in an ontology as shown in Figure 2.3, which describes vector data and their properties. In the rest of this thesis, the GeoSPARQL ontology[8] entities will be referred to using the prefix `geo:`. Figure 2.3 depicts a main

[4]https://www.ogc.org/standard/sfs/
[5]https://gdal.org/en/stable/drivers/vector/index.html
[6]https://www.opengis.net/def/crs/OGC/1.3/CRS84
[7]https://epsg.io/4326
[8]https://opengeospatial.github.io/ogc-geosparql/geosparql11/geo.ttl

class `geo:SpatialObject`, which has two disjoint sub-classes `geo:Feature` and `geo:Geometry` and is linked by the object property `geo:hasGeometry`. The first represents a real-world entity, whereas the latter represents all geometric forms defined in a spatial coordinate reference system [176]. `geo:Geometry` instances can have various data properties. The most important one concerns the data themselves, provided in serialised form. All serialisation data properties are sub-properties of `geo:hasSerialization`, and the predefined ones are `geo:asWKT` and `geo:asGML`. Other datatype properties are `dimension` (topological dimension), `geo:spatialDimension` (dimension of the spatial portion of the direct positions), `geo:coordinateDimension` (dimension of direct positions), `geo:isEmpty` (has no points), and `geo:isSimple` (contains no self-intersections except its boundary). GeoSPARQL defines object properties for topological relations whose domain and range are both `geo:SpatialObject`.

In the next subsection, we will describe the publicly available vector data sources that are considered in this thesis. In general, our proposed method will work with any OGC-approved vector data sources.

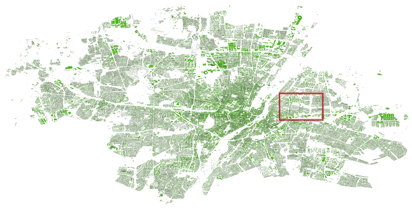
2.3.2.1 Global Administrative Areas (GADM)

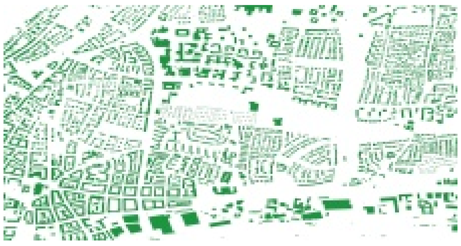
The Global Administrative Areas (GADM) v4.1[9] dataset contains map data on the administrative areas of all countries, at all levels of subdivision, at high spatial resolution, and includes an extensive set of attributes. Another open-source public dataset, geoBoundaries [195], provides a comprehensive collection of detailed geographic boundaries of subnational units—such as states and counties—encoded using ISO 3166 -1 alpha-3[10] standards and unique identifiers for every country worldwide. It supports up to five levels of administrative hierarchy and includes 351,819 individual shapes.

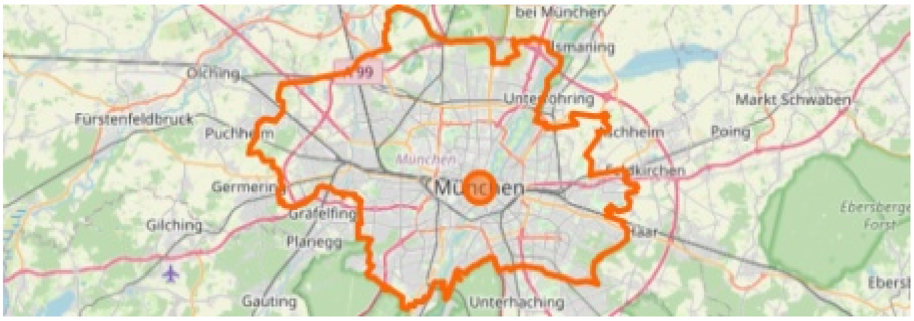
2.3.2.2 OpenStreetMap (OSM)

OpenStreetMap[11] (OSM) [98] represents a massive public repository of volunteered geographic information (VGI) [90] gathered through collaborative efforts worldwide. The OSM dataset provides geographic map objects and physical ground features (e.g., roads, buildings, tourist attractions) using *tags*[12], which are *key, value* pairs. For instance, "Hospital is a building" is implied as `tag: key="building", value="hospital"` in OSM terminology. However, this may not reflect a building's current usage; for example, an abandoned hospital or one repurposed for another use is still `building=hospital`, whereas active hospitals are marked with `amenity=hospital`. Tags are also attached with additional entities, which are encoded as *nodes* (spatially represented by points of interest or centroids), *ways* (ordered lists of points connected by lines) and *relations* (consisting, e.g. of a collection of ways, forming a polygon).

[9] https://gadm.org/data.html
[10] https://www.iso.org/iso-3166-country-codes.html
[11] https://wiki.openstreetmap.org/wiki/Main_Page
[12] https://wiki.openstreetmap.org/wiki/Tags


(a) OSM Buildings Munich (310 km^2)


(b) Density 5000+ buildings in 10.8 km^2



(c) OSM ID 62428 of Munich as visualised in OpenStreetMap web interface

Figure 2.4: OpenStreetMap Data

This thesis is concerned with OSM building 2D footprint data[13] (polygon) as displayed in Figure 2.4b for the running example AOI Munich (OSM ID 62428[14] as shown in Figure 2.4c). It can be extended to any arbitrary AOI, depending on the use case. Apart from buildings, other OSM features such as *amenity, tourism, sport* as described in the map features[15] can also be used.

OSM data for any geographic area worldwide is available at http://planet.openstreetmap.org/ in several formats, e.g., a compressed XML file, a custom PBF file and a shapefile. A utility *osm2pgsql*[16] imports OSM data into a PostGIS database. The OpenStreetMap community developed two notable command-line tools, i.e., *Osmosis*[17] and *Osmium*[18] to download and manipulate OSM data. Another notable open-source project is OSMnx [43], which provides a Python package for extracting, modelling, analysing, and visualising geospatial features from OSM data across various disciplines, including urban network planning, transport engineering, and computer science. As of March 4, 2026, the global dataset[19] contains approximately 10.3 billion nodes, 1.1 billion ways, and 14 million relations.

[13] https://wiki.openstreetmap.org/wiki/Key:building
[14] https://www.openstreetmap.org/relation/62428#map=11/48.1397/11.5380
[15] https://wiki.openstreetmap.org/wiki/Map_features
[16] https://osm2pgsql.org/
[17] https://wiki.openstreetmap.org/wiki/Osmosis
[18] https://osmcode.org/
[19] https://planet.openstreetmap.org/statistics/data_stats.html

The LinkedGeoData (LGD) ontology[20] [211] conceptualises the semantics of a vast collection of OSM data . At its core, LinkedGeoData converts OSM's primitive objects *nodes*, *ways*, and *relations*—together with their open tag-value annotations into RDF following a lightweight but expressive RDFS/OWL ontology automatically derived from the OSM tag ecosystem. The prefix `lgdo:` prefix will be used to refer to the LinkedGeoData ontology entities in this thesis. LinkedGeoData Ontology dynamically generates classes and properties from the most frequent OSM key-value pairs (e.g., `lgdo:Bakery, lgdo:Motorway, lgdo:School`), balancing semantic precision with the organic, community-driven nature of OSM tagging practices. The ontology has evolved considerably since 2009: URIs are now human-readable (camel-case), literals are correctly typed and language-tagged, geometries are serialised as WKT literals, and comprehensive class hierarchies are maintained through statistical analysis of tag co-occurrence.

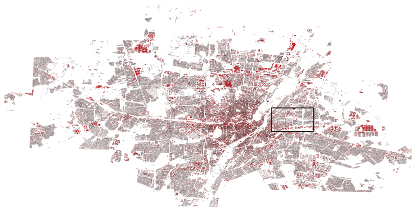
2.3.2.3 CityGML

The OGC CityGML standard provides a comprehensive information model and encoding for the storage, exchange, and representation of 3D urban objects, integrating both geometric and semantic properties of features such as buildings, transportation infrastructure, vegetation, water bodies, and terrain [93, 140]. By combining thematic attributes (e.g., building function, roof type, construction material) with topology and geometry, CityGML supports advanced spatial and semantic queries for applications including urban planning, disaster management, energy simulation, and population analysis [138]. However, its full analytical potential remains underexploited because the primary GML/XML encoding is optimised for data exchange rather than for efficient querying or seamless integration with Semantic Web technologies [93].

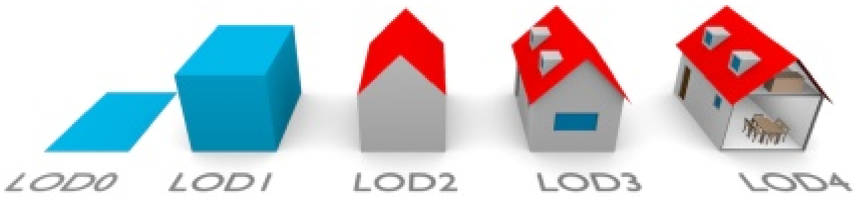
A central feature of CityGML is its multiple Levels of Detail (LoDs)[21], which allow the same objects to be represented with increasing geometric accuracy and semantic richness [93]. CityGML 2.0 defines five LoDs (LoD0–LoD4), ranging from regional footprints (LoD0) to architecturally detailed models with interiors (LoD4), as shown in Figure 2.5c. Recently, Zhu et al. [243] introduced GlobalBuildingAtlas, which provides approximately 2.75 billion 3D buildings for the whole world. Although highly useful, the original LoD specification has been criticised for geometric and semantic ambiguities that cause implementation inconsistencies and interoperability problems [37]. Biljecki et al. [37] addressed this by proposing a refined framework with 16 discrete LoDs[22] focused on exterior building shells, which provides stricter criteria and significantly reduces modelling ambiguity.

CityGML datasets are typically managed in relational databases via 3DCityDB, which maps the hierarchical CityGML structure to relational tables

[20] https://linkedgeodata.org/ontology/
[21] https://docs.ogc.org/guides/20-066.html#overview-section-levelsofdetail
[22] https://filipbiljecki.com/phd


(a) CityGML 3D Buildings of Munich

(b) 5000+ buildings within 10.8 km^2 area



(c) Five Levels of Detail (LoDs) of CityGML [37]

Figure 2.5: CityGML Data

and enables SQL-based access and spatial indexing [236]. Despite its effectiveness for large-scale storage and visualisation, the relational schema flattens CityGML's rich semantic hierarchy (e.g., part–whole relationships between buildings and their components), making ad-hoc analytical and thematic queries difficult to formulate [69, 173].

Integration with the semantic web is further hindered by CityGML's deeply nested GML/XML schema, which complicates transformation into lightweight, graph-oriented RDF/OWL representations required for reasoning and linked data [173, 138, 115]. CityJSON [150], adopted as an OGC standard in 2021, offers a compact JSON encoding that reduces file size by factors of 5–6 while preserving core CityGML 2.0 semantics, greatly simplifying parsing and web-based processing. Ongoing developments align CityJSON with CityGML 3.0's enhanced conceptual model, including dynamic features and improved coverage of non-building themes [140].

Ontology-based approaches [119] overcome the semantic gap between the conceptual model and storage by transforming CityGML data into RDF triples via formal OWL ontologies. Especially some recent studies [104, 67, 69] which rely on VKGs to enable expressive SPARQL querying and reasoning without full data materialisation, yet CityGML–raster integration and query answering remain under-explored. The CityGML ontology is described in Figure 2.6 and the prefix `bldg:` will be used to refer to the CityGML ontology entities in this thesis.

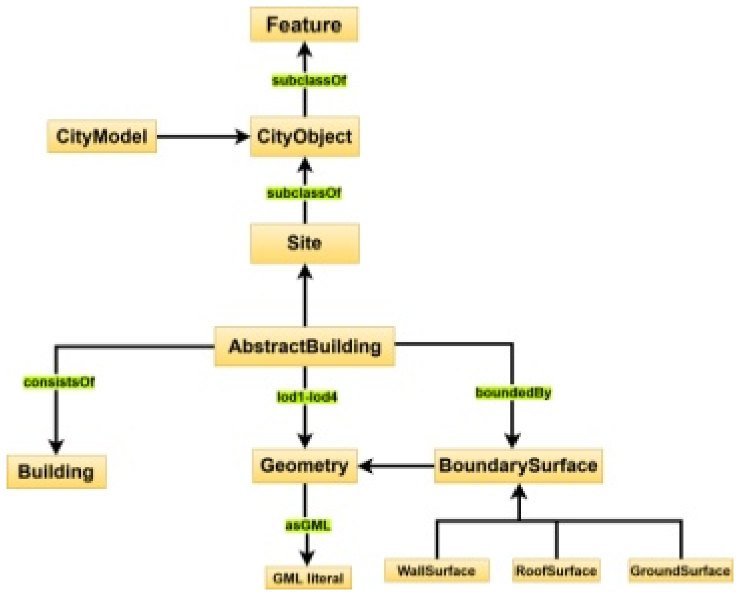


Figure 2.6: CityGML Building ontology (simplified UML representation)

2.3.2.4 Other Data

Our proposed methodology can also be extended to work with other popular vector datasets such as Natural Earth [130], Global Building Atlas (GBA) [242], World Database on Protected Areas (WDPA) [39], WorldPop 2020 [230] and the Global Human Settlement Layer (GHSL) [80].

2.3.3 Raster Data

Raster data, also referred to as *images* [197], *gridded data* [28], *multidimensional discrete data* (MDD)[22], or datacubes [137], represent real-world phenomena that vary continuously over space, time, and maybe even more dimensions, depending on the particular *domain of interest*, such as computer graphics, medical, astronomy, geospatial, etc.

Here, we present a few examples of raster data to illustrate its versatility in capturing dense, multidimensional phenomena that are challenging to model in other formats. In medical imaging, 3D `x/y/t` time series raster images from CT scans and fMRI are represented in grids of *voxel* [196]. Astronomy utilises raster data for telescope observations, such as radio signals, which form 4D arrays `x/y/z/t` that model celestial phenomena [196]. In engineering and business analytics, raster-like structures appear in simulation outputs, such as fluid dynamics models or statistical aggregates in data cubes [212] for sales forecasting [116]. In the geospatial domain, satellite imageries are stored as a 2D or 3D grid to capture land cover, vegetation indices, or atmospheric conditions [217]. Additionally, hyperspectral imaging in botany and agriculture employs raster data to analyse plant health by examining spectral signatures across hundreds of wavelength bands, for the detection of stress or nutrient deficiencies [166].

2.3.3.1 Data Structure - Arrays

Raster data is natively represented by arrays, which are essential data structures in computer science that consist of collections of uniformly typed variables [159]. Elements of an array can be accessed by indices (integers) that are directly translated to memory addresses. However, many programming languages support associative arrays with arbitrary index data types. *Multidimensional Arrays* (MDAs) extend the array concept to higher dimensions, forming a grid-like structure that captures relationships across multiple axes [196].

A *multidimensional array* $\mathcal{A}_{MDA}$ with N dimensions and M attributes a.k.a. a N-dimensional array is defined by a set $\boldsymbol{d} = \{d_1, d_2, \ldots, d_N\}$ of *dimensions* and a set $\boldsymbol{a} = \{a_1, a_2, \ldots, a_M\}$ of *attributes*. Every dimension d_i, $i \in \{1, 2, ..., N\}$ is a finite ordered set over a discrete domain $[lb_i, ub_i]$ that contains the integers between *lower bound* lb_i and *upper bound* ub_i . Every combination of dimension indices $[i_1, i_2, \ldots, i_N]$, defines a *cell*, also known as *pixel* [197]. Cells have the same scalar type (e.g., integers or floats), given by the set $\boldsymbol{a}$ of attributes. Dimensions and attributes define the *schema* of the array. Based on these concepts, an array can be thought of as a function defined over dimensions and returning attribute tuples as values:

$$Array : [d_1, d_2, \ldots, d_N] \mapsto \langle a_1, a_2, \ldots, a_M \rangle$$

This formalisation aligns with the notion of N-ordered (or dimensional) tensors [136], which differ from multidimensional vectors, which do not distinguish between dimensions and attributes. The functional formalisation of arrays also makes explicit the functional dependency between dimensions and attributes specific to the relational data model [59], in which the N dimensions form a key of the corresponding relation.

2.3.3.2 Mathematical Formulation of Raster Data

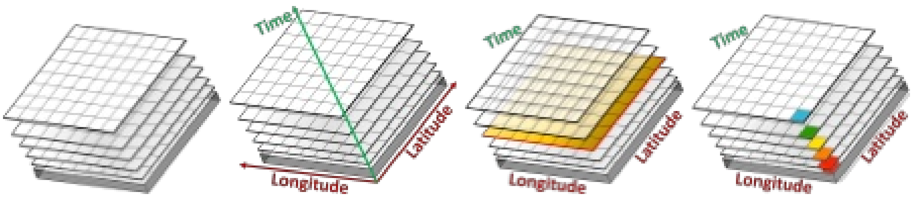
Consider a generic raster $\mathcal{R}$ as a function mapping a discrete domain $\mathcal{D}$ in $\mathbb{Z}^N$ (for an N-dimensional space) to a range of cell values from type T (e.g., real numbers $\mathbb{R}$ or integers $\mathbb{Z}$):

$$\mathcal{R} : \mathcal{D} \to T$$

Here, $\mathcal{D}$ denotes the grid domain, defined as the Cartesian product of extents along each dimension d_i (for $i = 1, \ldots, N$).

In the geospatial domain, most raster data representations use two-dimensional (2D) arrays, where the domain is given by $\mathcal{D} = [lb_x : ub_x] \times [lb_y : ub_y]$, which map either to geographic coordinates (ϕ, λ) with a known datum or to projected coordinates (x, y) in a known coordinate reference system [159]. Higher-dimensional geospatial rasters naturally extend this: a 3D raster might include time (t), yielding $\mathcal{D} = [lb_x : ub_x] \times [lb_y : ub_y] \times [lb_t : ub_t]$ for time-series imagery, while 4D or ND forms incorporate additional axes like *depth* (z) or spectral wavelength (λ) [196]. Irregular grids or non-uniform resolutions require auxiliary mappings, and nodata values can denote invalid cells. This formulation underscores raster data's versatility as MDAs, bridging dense grids to analytical data cubes, though practical implementations must address discretisation artefacts and semantic ambiguities to ensure robust scientific application [159, 196].

2.3.3.3 Geo Raster Data



(a) EO Raster (ND) (b) Coordinates (c) A Raster Layer (d) Cell per Time

Figure 2.7: Four Different AOIs

This thesis focuses on geospatial raster data, or Earth Observation (EO) datacubes (Figure 2.7), a quintessential example of big raster data, representing Earth-surface phenomena through gridded cells georeferenced to coordinate systems [152]. Geospatial raster data is also referred to as gridded *coverage* according to the OGC *Coverage Implementation Schema* (CIS) standard [27], as adopted by ISO 19123 *Geographic information - Schema for coverage geometry and functions*. According to Part 2 of this ISO standard[23], the concept of *coverage* represents multi-dimensional grids of both regular and irregular type as the natural representation of various space-time-varying phenomena predominantly in the geospatial domain. Geospatial raster data can be encoded in different raster file formats[24], such as NetCDF[25], GeoTIFF[26], HDF5[27], GeoZarr[28] etc., which can be accessed ad hoc via Python, R, or MATLAB scripts. Geospatial raster data contribute considerably to the complexities of big data, addressing the "four Vs": volume (petabytes from continuous satellite streams), velocity (real-time ingestion from sensors), variety (heterogeneous formats), and veracity (noise from sensor interference) [28]. Their integration with metadata

[23] https://www.iso.org/obp/ui/en/#iso:std:70948:en
[24] https://gdal.org/en/stable/drivers/raster/index.html
[25] https://www.unidata.ucar.edu/software/netcdf/
[26] https://www.ogc.org/publications/standard/geotiff/
[27] https://gdal.org/en/latest/drivers/raster/hdf5.html
[28] https://zarr.dev/geozarr-spec/documents/standard/template/geozarr-spec.html

(e.g., projections, provenance) adds semantic layers, amplifying challenges in big data management ecosystems [7].

In raster data, the terms *layers* and *bands* describe two distinct ways of organising multiple raster datasets within a single gridded dataset. While often used interchangeably in casual contexts, they carry specific semantic and structural meanings in scientific data management.

Layers refer to a stack of multiple related rasters sharing similar spatial and temporal dimensions, such as time-series stacks or the `x/y/z` axes in climate models, with each layer capturing a specific temporal snapshot [116]. *Bands*, in contrast, denote multiple colour channels within an individual raster, typically corresponding to various spectral ranges or attributes; examples include the three bands (red, green, blue) in RGB images or hundreds of bands in hyperspectral data for precise material identification [166].

These features support efficient array operations, such as slicing (extracting raster subsets or portion of raster data, along dimensions) and aggregation (e.g., averaging across bands). Yet, they pose storage and computational difficulties due to the substantial volume and density of the data [217]. Consequently, raster data offers strong scalability for analytics but requires tailored approaches to address memory and processing demands. Some of the popular EO raster datacubes are the following: OpenEO datacube[29] [182], Australian datacube [153], and Swiss datacube [57].

2.3.3.4 Array DBMS and Query Processing

Querying raster data encompasses operations such as induction (e.g., cell-wise arithmetic), slicing, sub-setting, and aggregation (e.g., summing across dimensions) [22]. Standards such as SQL/MDA (Multidimensional Arrays) [167, 168] extend SQL with array constructors and operators, and OGC WCPS provides domain-specific languages for coverage queries, supporting filtering and re-projection of geospatial rasters [28]. Array Database Management Systems (ArrayDBMS) such as RasDaMan [23], SciDB [48], etc., extend beyond traditional DBMS to natively support multidimensional arrays, enabling efficient storage, indexing, and query answering. In array DBMSs, queries are optimised through algebraic rewriting and partitioning, ensuring efficient execution on large arrays [196].

One of the earliest works, Tan et al. [217] proposed a multidimensional array model to ingest EO data in common data formats, e.g., NetCDF, HDF, GeoTIFF, to the array DBMS, i.e., SciDB [48] and demonstrated a forest fire simulation. Gao et al. [83] introduced in 2022 GeoCube, a physical spatial-temporal data cube infrastructure designed for large-scale management of multi-source EO raster and vector data under the Open Geospatial Engine (OGE) framework [237, 239]. GeoCube ingest heterogeneous raster data (MODIS, Sentinel-2, CMIP6 climate models, etc.) and vector data into an array DBMS, i.e., RasDaMan, supplemented by Zarr [169] for cloud-native

[29] `https://openeo.org/documentation/1.0/datacubes.html#what-are-datacubes`

compatibility and xarray [116] for in-memory manipulation. However, the materialised nature of GeoCube introduces data duplication and update overhead.

However, challenges emerge when data volumes become very large (e.g., terabytes), making it difficult to query the required information [217]. Moreover, when dealing with gridded coverages or raster data, one must also include related metadata such as domain, range, and provenance information. In contrast to arrays, the structure of this metadata is substantially less regular and may be incomplete or vary across arrays. In our research, we rely on Raster Data Manager (RasDaMan), a domain-agnostic array DBMS that implements OGC standards to manage extensive multidimensional raster data or gridded coverages through rich array algebra (cf. Section 4.3.2).

2.3.4 Linked Data

Linked Data [41, 40, 107] refers to a set of best practices for technically publishing and connecting structured data on the web based on the four basic principles[30] [108] of the Semantic Web as mentioned below:

1. URIs should be used to denote things.
2. HTTP URIs should be used so that these things can be referred to and dereferenced (looked up) by human users and software agents.
3. W3C standards such as RDF and Web Ontology Language (OWL) should be used, so that valuable information can be provided when looking up the URIs.
4. Data should be interlinked using URIs to create a densely interconnected graph of knowledge (e.g., Linked Open Data cloud[31]) to infer further knowledge.

These data are categorised as Linked Open Data (LOD) [91] when they are publicly available, accessible online, and subject to an open license. To publish linked data, it is essential to establish connections from the instances of a source to external data utilising background knowledge (e.g., linking DBpedia to Wikipedia), standardised identifiers (e.g., ISBN numbers), or pattern recognition (e.g., names, latitude, longitude, and other data used to connect GeoNames to DBpedia) [181]. The majority of Linked Data is produced automatically by transforming existing structured data sources, often relational databases, into RDF, utilising an ontology that closely aligns with the original data source. The links in the Web of Linked Data make the Semantic Web navigable and, in addition, augment knowledge by integrating data from one source with information from other sources. A prevalent method for linking

[30]https://www.w3.org/DesignIssues/LinkedData.html
[31]https://lod-cloud.net

data on the Web is to use *owl:sameAs*, which represents *identity links* [100]. Instead of reusing existing URIs, new URIs are often automatically generated when publishing linked data. The *owl:sameAs* link is then employed to assert that two distinct URI references denote the same real-world entity.

2.3.4.1 DBpedia

DBpedia[32] [13] is a popular linked data source covering an extensive range of topics which are extracted from Wikipedia. It is also the centre point of the Linked Open Data Cloud [149]. The current version describes approximately 4.5 million things, including 1.4 million persons and 730,000 places, which are mutually interlinked. It is connected to other linked data sources, including Freebase and GeoNames, via approximately 50 million links.

2.3.4.2 LinkedGeoData

LinkedGeoData (LGD[33]) [14, 211] is one of the most ambitious and enduring contributions to represent geospatial data sources (typically vector geometries) to the Semantic Web. It transforms the collaboratively curated OSM dataset (cf. 2.3.2.2) into a queryable RDF knowledge base (classes and properties), adhering to the four basic principles [108] of linked data. LinkedGeoData has effectively added a spatial dimension to the LOD cloud.

2.3.4.3 GeoNames

People typically use names rather than coordinates to designate geographic features [223]. Consequently, to implement query processing utilising geographic names rather than coordinates, it is essential to identify the precise geographic feature represented by a name through an ontology and its corresponding database. The GeoNames [229] geographical database contains over 12 million unique geographical point features, along with their 25 million names, including alternate and translated names (16 million), population data, time zones, and associated geo-coordinates (latitude/longitude in WGS84 [34]), collected from 400+ different data sources[35]. A typical entry in the database is an instance of type *Feature* and has a *Feature Class* (administrative divisions, populated places, etc.), a *Feature Code* (subcategories of *Feature Class*). All features are categorised into one of 9 feature categories, i.e., `A, H, L, P, R, S, T, U, V` and further subcategorised into one of 680 feature codes[36] as shown in Table 2.1. Listing 2.1 depicts the RDF data of Munich[37] in the GeoNames database, where it is depicted by a feature code `P.PPLA` which signifies class

[32] http://wiki.dbpedia.org/
[33] http://linkedgeodata.org/
[34] https://www.w3.org/2003/01/geo/
[35] https://www.geonames.org/datasources/
[36] https://www.geonames.org/export/codes.html
[37] https://www.geonames.org/2867714/munich.html

Table 2.1: GeoNames Feature Classes hierarchy and Feature Codes. [215]

#	Feature Class	Feature Codes (fClass.fCode)
1	Administrative Region (**A**)	**A**.ADM1 (admin region 1), **A**.TERR (territory), **A**.ZN (Zone)...
2	Hydrographic (**H**)	**H**.AIRS, **H**.ANCH, **H**.BAY, **H**.BGHT, **H**.BNK, ...
3	Locality (**L**)	**L**.AGRC, **L**.AMUS, **L**.AREA, **L**.BSND, L.BSNP, **L**.BTL, **L**.CLG ...
4	Populated Place (**P**)	**P**.PPL, **P**.PPLA, **P**.PPLAZ, **P**.PPLA3, **P**.PPLA4, **P**.PPLC,...
5	Road (**R**)	**R**.PTGE, **R**.RD, **R**.RDA, **R**.RDB, **R**.RDCUT, **R**.RDJCT...
6	Spot (**S**)	**S**.ADMF, **S**.AGRF, **S**.AIRB, **S**.AIRF, **S**.AIRP, **S**.AIRQ ...
7	Hypsographic (**T**)	**T**.ASPH, **T**.ATOL, **T**.BAR, **T**.BCH, **T**.BCHS, **T**.BDLD,...
8	Undersea (**U**)	**U**.APNU, **U**.ARCU, **U**.ARRU, **U**.BDLU, **U**.BKSU, **U**.BNKU...
9	Vegetation (**V**)	**V**.BUSH, **V**.CULT, **V**.FRST, **V**.FRSTF, **V**.GRSLD, **V**.GRVC...

`P` for populated place and `PPLA` signifies ”seat of a first-order administrative division”.

Listing 2.1: A snippet of the GeoNames description of Munich in RDF/XML.

```xml
<rdf:RDF>
       <gn:Feature rdf:about="https://sws.geonames.org/2867714/">
           <rdfs:isDefinedBy
            rdf:resource="https://sws.geonames.org/2867714/about.rdf"/>
           <gn:name>Munich</gn:name>
           <gn:officialName xml:lang="de">München</gn:officialName>
           <gn:alternateName>München</gn:alternateName>
           <gn:alternateName xml:lang="no">München</gn:alternateName>
           <gn:alternateName xml:lang="sv">München</gn:alternateName>
           <gn:alternateName xml:lang="en">Munich</gn:alternateName>
           <gn:alternateName xml:lang="fr">Munich</gn:alternateName>
           <gn:alternateName xml:lang="it">Monaco di Baviera</gn:alternateName>

           <gn:featureClass rdf:resource="https://www.geonames.org/ontology#P"/>
           <gn:featureCode rdf:resource="https://www.geonames.org/ontology#P.PPLA"/>
           <gn:countryCode>DE</gn:countryCode>
           <gn:population>1260391</gn:population>
           <gn:postalCode>80331</gn:postalCode>
           <wgs84_pos:lat>48.13743</wgs84_pos:lat>
           <wgs84_pos:long>11.57549</wgs84_pos:long>
           <gn:parentFeature rdf:resource="https://sws.geonames.org/6559171/"/>
           <gn:parentCountry rdf:resource="https://sws.geonames.org/2921044/"/>
           <gn:parentADM1 rdf:resource="https://sws.geonames.org/2951839/"/>
           <gn:parentADM2 rdf:resource="https://sws.geonames.org/2861322/"/>
           <gn:parentADM3 rdf:resource="https://sws.geonames.org/3220837/"/>
           <gn:parentADM4 rdf:resource="https://sws.geonames.org/6559171/"/>
           <gn:nearbyFeatures rdf:resource="https://sws.geonames.org/2867714/nearby.rdf"/>
           <gn:locationMap rdf:resource="https://www.geonames.org/2867714/munich.html"/>
           <gn:wikipediaArticle rdf:resource="https://en.wikipedia.org/wiki/Munich"/>
           <rdfs:seeAlso rdf:resource="https://dbpedia.org/resource/Munich"/>
           <gn:wikipediaArticle rdf:resource="https://ru.wikipedia.org/wiki/Munich"/>
       </gn:Feature>

       <foaf:Document rdf:about="https://sws.geonames.org/2867714/about.rdf">
           <foaf:primaryTopic rdf:resource="https://sws.geonames.org/2867714/"/>
           <cc:license rdf:resource="https://creativecommons.org/licenses/by/4.0/"/>
           <cc:attributionURL rdf:resource="https://www.geonames.org"/>
           <cc:attributionName>GeoNames</cc:attributionName>
           <dcterms:created>2006-01-15</dcterms:created>
           <dcterms:modified>2023-10-12</dcterms:modified>
       </foaf:Document>
</rdf:RDF>
```

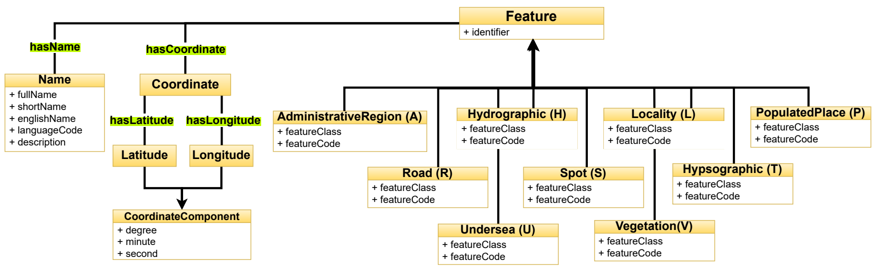


Figure 2.8: GeoNames ontology (simplified UML representation)

The semantics of all these geographic point features, along with classes and codes, are described by a GeoNames ontology[38] as shown in the Figure 2.8. The main class of the ontology is *Feature* (or *GeographicFeature*) that has several subclasses to differentiate between the different forms of geographic features, such as *AdministrativeRegions*, which is a superclass of all countries and political states, class *PopulatedPlace* holds all country capitals, cities and villages, the class *Hydrographic* represents all water-related geographic features, and the rest of the subclasses are listed in Table 2.1. Every point feature is associated with a geographic coordinate, i.e., a pair of latitude and longitude (in WGS84) based on W3C Basic Geo Vocabulary[39]] which is widely used by GPS. The ontology also contains several properties, e.g., *neighbours, nearby, name, postalCode*, and *population* [223], which are not visible in the Figure 2.8. GeoNames ontology is also linked to other data sources, e.g., DBpedia and LinkedGeoData by utilising FOAF[40] and SKOS[41] vocabularies. In the rest of In the thesis, the `gn:` prefix refers to GeoNames ontology entities.

2.4 Data Integration

Data integration aims to unify disparate sources, enabling seamless access and processing without altering underlying structures [203]. The proliferation of diverse, voluminous, and heterogeneous data types in the above sections necessitates querying them in an integrated manner to yield actionable insights, often requiring deep domain knowledge of each data format and its metadata [151]. This becomes more challenging in a geospatial context because many real-world applications must process ever-increasing large geospatial datasets (petabytes) with associated metadata from diverse sources and in various formats and semantics to perform complex location-based business analyses and informed decision-making [95, 205].

[38]https://www.geonames.org/ontology/documentation.html
[39]https://www.w3.org/2003/01/geo/
[40]https://lov.linkeddata.es/dataset/lov/vocabs/foaf
[41]https://www.w3.org/TR/swbp-skos-core-spec/

Geospatial data integration is particularly complex due to the dichotomy between vector and raster models. Vector data represent discrete features, such as points, lines, and polygons, whereas raster data use grid cells to represent continuous phenomena, such as satellite imagery [60]. Most research addresses these independently, with limited focus on joint handling [205]. International standards such as the OGC Web Feature Service (WFS) [178] for vectors, and the OGC Web Coverage Service (WCS 2.0) [177] for rasters maintain this separation and lack unified query languages, data structures, or algorithms for simultaneous processing. Popular GIS tools like ArcGIS [75] and GRASS [171] convert vectors to rasters for operations like zonal statistics in Map Algebra. This computes summary statistics (e.g., mean, sum) over raster zones defined by vector polygons but incurs overhead from format conversion. Brown et al. [48] propose a multidimensional array model that abstracts over both data types and storage and operators, yet provides no implementation details. Brisaboa et al. [46] introduce a framework that utilises *R-tree* data structure for storing vector data and a compact data structure k^2-`acc` [63] for storing compressed raster data. It supports overlap queries with range constraints but fails to pinpoint exact raster cells; hence, it becomes inefficient for rasters with many distinct values, as `k`^{2}`-acc` requires a separate tree per value, leading to space bloat and poor scalability [143]. To overcome this limitation, the `k`^{2}`-raster` data structure has been introduced, which compresses and indexes rasters simultaneously for faster queries on clustered data [205]. Predominant vector-raster data integration methods convert one format to the other: rasterising vectors for pixel-based joins or vectorising rasters into points for vector joins. These suffice for low- to medium-resolution data but fail for high-resolution satellite data, where the size scales quadratically [207]. Singla et al. [208] propose the *Raptor Join* operator with a *Flash Index*, enabling in-situ joins in native formats and outputting aligned pixel ranges for ad-hoc queries. All these works align with our research questions but lack reproducible code and clarity regarding interoperability with geospatial database functions for downstream processing, particularly for raster data within the VKG paradigm.

Chapter 3

Semantic Web Technology

Semantic Web technologies are widely acknowledged as a crucial element of Web 3.0, the next-gen evolution of internet technology, which relies heavily on AI and semantic technologies to manage vast amounts of data from diverse sources. Envisioned by Tim Berners-Lee [33] in 2001, the Semantic Web (a.k.a Web of Data) has become a World Wide Web Consortium (W3C) standard for the representation and sharing of data available in the World Wide Web (WWW) to enable machines to comprehend the semantics of the data [32]. These semantics allow intelligent machines to process, analyse and manipulate web resources to accomplish complex tasks on behalf of humans.

Key challenges include handling distributed, inconsistent data under open-world semantics, ensuring scalability for vast web knowledge, and integrating heterogeneous formalisms [213]. Standardisation efforts mitigate fragmentation, but evolving content demands resilient systems. Applications span data integration, where ontologies align disparate sources; semantic search, enhancing precision via Linked Data [41, 40]; and knowledge management, unifying enterprise data. Overall, the Semantic Web facilitates intelligent applications, from e-commerce to scientific collaboration, by leveraging formal knowledge representation.

This thesis focuses on the semantic management of geospatial data. In this context, interestingly, the initial example presented in Tim Berners Lee's groundbreaking article [33] pertains to geospatial data, which conceptualised a semantic web agent capable of identifying local health-care providers within a 20-mile radius of a residence by comprehending a user's health coverage plan [122].

3.1 Components

The Semantic Web architecture is built with several technological components stacked together as proposed by Tim Berners-Lee [32, 115] and illustrated in

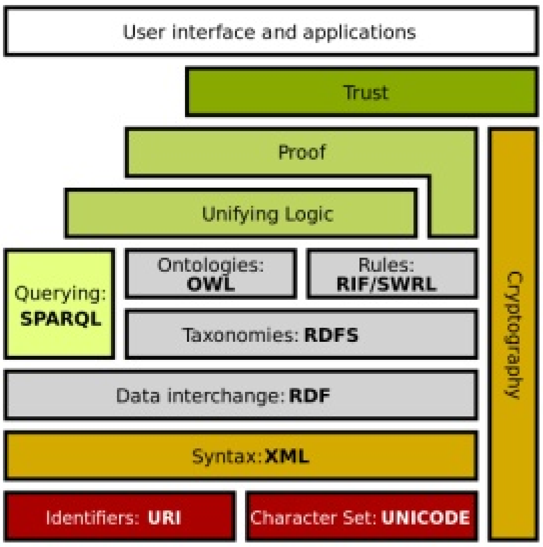


Figure 3.1: Semantic Web Architecture

Figure 3.1. At the base of the stack, URIs and XML provide identification and syntax. RDF forms the data model, representing statements as triples (*subject-predicate-object*), with RDFS adding schema vocabulary for classes and properties [45]. OWL builds on this for expressive ontologies, supporting DL-based reasoning, and SPARQL enables querying RDF data, while rule languages like RIF/SWRL integrate logic programming [115]. Higher layers include proof and trust mechanisms, though implementation focuses on lower levels [32]. The following subsections describe the most relevant components of the semantic web stack in a bottom-up approach.

3.1.1 URI/IRI

A Uniform Resource Identifier (URI) is a sequence of characters that universally and uniquely identifies a resource in web. It is a superset of Uniform Resource Locator (URL) and Uniform Resource Name (URN). Whereas an Internationalized Resource Identifier (IRI) is a generalised form of URI. The difference between the two types of identifiers lies in the supported character sets: URI encoding is limited to US-ASCII characters, while IRIs support the Unicode character set (a superset of US-ASCII).

3.1.2 RDF

The *Resource Description Framework* (RDF) [163] is a W3C standard language for processing metadata, providing a common semantic layer between heterogeneous web applications that need to exchange machine-understandable information and/or automatically process one or more semantic web resources.

The RDF syntax is based on the following ideas, *(1)* everything can be referenced using an IRI, *(2)* everything can be reused for describing the characteristics of something else, and the fact that, *(3)* the language should have a simple structure (low expressive power) suitable for describing any resource in any possible situation.

The RDF data model consists of resources, values, and properties to connect resources and values. The fundamental element for representing semantic information in RDF is a *triple*, which describes a *resource* and consists of a *subject* (representing the resource), a *predicate* (expressing the property), and an *object* (indicating the value of the property). Listing 3.1 demonstrates a simple RDF data model, which represents *Alfred* as a resource, asserts his role as a *student* at TU Munich, specifies his *residence* in Munich, and captures his place of birth from Stockholm, Sweden, using standard RDF triples.

Listing 3.1: RDF triples

```
ex:people/Alfred rdf:type rdfs:Resource.
dbp:ontology/Student rdf:type rdfs:Class.
ex:people/Alfred rdf:type dbp:ontology/Student.

dbp:ontology/university rdf:type rdf:Property.
ex:people/Alfred dbp:ontology/university dbp:resource/TU_Munich.

dbp:ontology/residence rdf:type rdf:Property.
ex:people/Alfred dbp:ontology/residence dbp:resource/Munich.

dbp:ontology/birthPlace rdf:type rdf:Property.
ex:people/Alfred dbp:ontology/birthPlace dbp:resource/Stockholm.

dbp:ontology/country rdf:type rdf:Property.
dbp:resource/Stockholm dbp:ontology/country dbp:resource/Sweden.
```

The key feature of Linked Data (cf. 2.3.4) is that an author of one data set can link its objects to objects in another data set. In the above example 3.1, the data publisher is using the domain `example.org` (`ex`) to describe Alfred, also stating the corresponding birthplace, e.g., the city of Stockholm as an object of another data publisher DBpedia (`dbp`).

A collection of one or more RDF triples is called an RDF graph, which can be represented graphically as a set of *nodes* (resources or primitive types) connected by *edges* (properties), and can be explored using semantic query, i.e., SPARQL. Figure 3.2 shows the RDF graph for the aforementioned example.

Data in RDF graphs are physically represented using the following serialisations:

- **N-Triples**[1]: a simple triple-per-line format, suitable for large data files for parallel processing,

[1]https://www.w3.org/TR/n-triples/

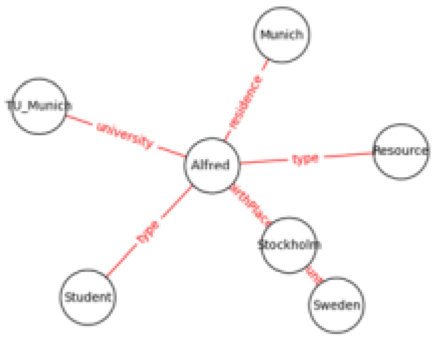


Figure 3.2: RDF Graph

- **Turtle**[2]: a superset of N-Triples, which is more compact and more human-readable.
- **Notation3 (N3**[3]**)**: a superset of Turtle, going even beyond the RDF model.
- **JSON-LD**[4]: a JSON-based format for Linked Data, capable of serialising any RDF graph.
- **RDF/XML**[5]: a widely used XML syntax; see Listing 2.1 for an example.

3.1.3 RDFS

RDF Schema (RDFS) [45] is a W3C-approved extension of the RDF language, containing a new set of classes and properties designed for implementing RDF vocabularies since 2004. RDFS is a language used for defining lightweight ontologies. In fact, the RDFS vocabulary is not domain-specific and allows the specification of the semantics of arbitrary RDF vocabularies.

The main RDFS classes are `rdfs:Resource`, the superclass to which each RDF resource belongs, and `rdfs:Class`, the class of resources that are RDF classes. Other important classes are `rdfs:Literal`, `rdfs:Datatype` and `rdf:Property`, all of which are subclasses of `rdfs:Class`.

Properties describe the relation between two resources, called the subject and object. The most essential ones are `rdfs:domain` , `rdfs:range` , `rdf:type`, `rdf:label`, `rdfs:subClassOf` and `rdfs:subPropertyOf`.

Listing 3.2: RDFS triples

```
        dbp:ontology/Student rdfs:subClassOf dbp:ontology/Person.
```

[2]https://www.w3.org/TR/turtle/
[3]https://www.w3.org/TeamSubmission/n3/
[4]https://www.w3.org/TR/json-ld/
[5]https://www.w3.org/TR/rdf-syntax-grammar/

```
    dbp:ontology/residence rdfs:subPropertyOf dul:hasLocation.
    dbp:ontology/residence rdfs:range dbp:ontology/Place.
```

3.1.4 OWL

The Web Ontology Language (OWL) [165] is a W3C standard markup language designed for applications that need to process information content, improving the machine interpretability of Web content (w.r.t. to XML, RDF, and RDFS) by providing an additional vocabulary along with a formal semantics. OWL2 [170] is updated version of OWL

The OWL2 is based on the ontology, which is a formal representation of concepts and domain-specific knowledge. The OWL2 syntax is based on RDFS, which provides constructs for building ontologies and describing classes, properties, individuals, and data values. Several syntax serialisation options exist, such as RDF/XML, Turtle, OWL2 XML, and the OWL2 Manchester Syntax, which are helpful when a "readable" syntax is needed [170].

Being both OWL and OWL2 based on the Description Logic (DL) theory, it is possible to apply reasoning techniques to infer implicit knowledge from knowledge bases' explicit axioms. Reasoning techniques can infer this knowledge by applying a set of rules (e.g., entailment rules) to the knowledge base, with computation times directly proportional to the ontology's expressiveness level. To provide different levels of response performances, the W3C has defined three OWL2 sub-languages, called *profiles*, which offer essential advantages in particular application scenarios [170]:

- **OWL2 EL**: this profile is designed for applications employing ontologies that define a huge number of classes and/or properties, as it captures the expressive power of the most commonly used ontologies and, at the same time, provides reasonable response times [170]. In particular, the OWL2 EL profile is suitable for scenarios that require polynomial response times and involve reasoning tasks for schema and data properties. The expressiveness power is limited to the EL family of description logic (existential quantification and conjunction).

- **OWL2 QL**: this profile is designed to enable sound and complete query answering capabilities, while ensuring logspace response times with respect to the size of the data [170]. The OWL2 QL syntax is sufficient for representing conceptual models of UML class diagrams and ER diagrams. The expressiveness power is limited to the DL-Lite family of description logic (subclass and superclass axioms, conjunctions (RHS only), property axioms). This profile is commonly used for designing query rewriting engines with reasoning capabilities. It implements a set of techniques for extrapolating RDF triples from relational databases, while delegating data storage and query evaluation to standard relational DBMSs.

- **OWL2 RL**: this profile is designed for those applications that require scalable reasoning capabilities without sacrificing too much expressive power [170]. The design of OWL2 RL is close to Description Logic Program (DLP) and can be implemented by rule-based reasoning engines. Indeed, the expressive power of OWL2 RL resembles a (near maximal) fragment of OWL2 as it supports all the axioms apart from disjoint unions of classes and reflexive object property axioms [170].

3.1.5 SPARQL

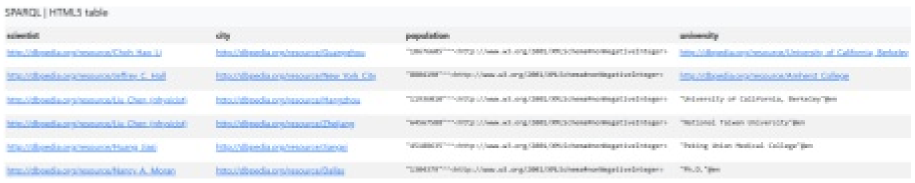
SPARQL Protocol And RDF Query Language (SPARQL) [106] is the W3C standard for querying RDF data representing information in the Semantic Web. Moreover, the query language's syntax is well-suited for data integration, as the queried data can originate from various sources. A simple example of a SPARQL query is displayed in Listing 3.3, with the corresponding response in Figure 3.3, after executed at DBpedia SPARQL endpoint[6]. This SPARQL query retrieves scientists whose birthplace cities have populations exceeding one million, along with their associated city, population, and educational institution. Query results are grouped by scientist to consolidate multiple associations, and the LIMIT 100 restricts output size. The query leverages RDF properties (`dbo:birthPlace, dbp:education`) to semantically model person–city–university relationships. Notice that the "`a`" in line 5 in Listing 3.3 is an alias for `rdf:type` referring class `Scientist`. A response to a SPARQL query can be formatted in XML, JSON, CSV, and TSV.

Listing 3.3: An example SPARQL Query

```
1   PREFIX dbo: <http://dbpedia.org/ontology/>
2   PREFIX dbp: <http://dbpedia.org/property/>
3
4   SELECT ?scientist ?city ?population ?university {
5     ?scientist a dbo:Scientist .
6     ?scientist dbo:birthPlace ?city ; dbp:education ?university .
7     ?city dbo:populationTotal ?population .
8     FILTER (?population > 1000000)
9   }
10  GROUP BY ?scientist
11  LIMIT 100
```

Open Geospatial Consortium (OGC) standard GeoSPARQL[7] extends SPARQL for querying of geospatial knowledge [21], which is represented in RDFS/OWL that introduces classes, features using Geography Markup Language (GML) and geometry literals WKT format. GeoSPARQL also provides a set of topological relationship vocabularies and ontologies for qualitative and quantitative reasoning.

[6]https://dbpedia.org/sparql
[7]https://www.ogc.org/standard/geosparql/



SPARQL | HTML5 table

scientist	city	population	university
http://dbpedia.org/resource/Chih_Han_Li	http://dbpedia.org/resource/Guangzhou	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	http://dbpedia.org/resource/University_of_California,_Berkeley
http://dbpedia.org/resource/Jeffrey_C._Hall	http://dbpedia.org/resource/New_York_City	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	http://dbpedia.org/resource/Amherst_College
http://dbpedia.org/resource/Liu_Chen_(physicist)	http://dbpedia.org/resource/Hangzhou	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	"University of California, Berkeley"@en
http://dbpedia.org/resource/Liu_Chen_(physicist)	http://dbpedia.org/resource/Zhejiang	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	"National Taiwan University"@en
http://dbpedia.org/resource/Huang_Jiali	http://dbpedia.org/resource/Tianjin	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	"Peking Union Medical College"@en
http://dbpedia.org/resource/Harry_A._Moran	http://dbpedia.org/resource/Dallas	"[illegible]"^^<http://www.w3.org/2001/XMLSchema#nonNegativeInteger>	"Ph.D."@en

Figure 3.3: SPARQL Query response at DBpedia Endpoint

3.2 Semantic Data Integration

In the Semantic Web, the Linked Data model has become a paradigm that facilitates the transition from a document-oriented web to a web of interlinked data [15]. The two most used linked data models are *property graphs* (PGs) [192] and *knowledge graphs* (KGs) [74]. The property graph focuses on analytics and querying, whereas the knowledge graph prioritises seamless incremental data integration via an ontology and applies a reasoner to derive new knowledge, which is the primary focus of the thesis.

3.2.1 Knowledge Graphs (KGs)

Knowledge Graphs (KGs) have been utilised in academia and business, often tied to semantic web technologies, linked data, large-scale data analytics, and cloud computing [74]. The term *knowledge graph* was coined by Austrian linguist Edgar W. Schneider in 1972 in a discussion of how to build modular instructional systems for courses [199]. KGs gained prominence in AI research and industry following the introduction of Google's Knowledge Graph[8] in 2012, which utilised KG resources such as DBpedia and Freebase to facilitate "searching for things, not strings". Since then, more organisations, such as eBay, Facebook, IBM, and Microsoft, have invested in knowledge graphs [174]. At present, KGs power a wide range of applications, including personal assistants such as Alexa, Siri, and Google Assistant, popular AI-enabled search engines, e-commerce platforms like eBay and Amazon, and other applications [123].

In the Semantic Web, knowledge graphs are used to represent, retrieve, and integrate data from highly heterogeneous sources [110]. A knowledge graph (KG) is a directed, labelled multi-relational graph where *nodes* typically represent either domain entities or their attributes, and (labelled) *edges* represent either relationships between entity-entity pairs or properties of entities [129]. KG is serialised as a set of triples, where each triple is a formal description of a fact and of the form *subject* (s), *predicate* (p), and *object* (o). e.g. For example, the statement "Munich is the Capital of Bavaria" can be expressed with the triple $\langle s, p, o \rangle \simeq \langle Munich,\ isCapitalof,\ Bavaria \rangle$. KG-based solutions often rely on an ontology to give meaning to the data, and the instances of an ontology are represented as KGs [111].

[8]https://blog.google/products/search/introducing-knowledge-graph-things-not/

3.2.2 Geospatial Knowledge Graphs (GeoKGs)

Geospatial Knowledge Graphs (GeoKGs) emerged as a prominent example of KGs for representing and reasoning over geospatial information concerning Geographic Information System (GIS) [240]. Geospatial entities are modelled using geo-coordinates (as pairs of longitude and latitude) that capture, on the one hand, geometric regions delimited by polygons (i.e., vector data) and, on the other hand, values from a continuous domain (e.g., temperature, precipitation) associated to all points in a specified region (i.e., raster data) [70]. GeoKGs are often converted from geospatial data sources [69], which are stored in spatial databases (PostGIS/PostgreSQL) or other popular formats like shapefile, CSV, and GeoJSON.

OGC standard GeoSPARQL ontology [21] utilises a representation of vector geometry literals compliant with *Geography Markup Language* (GML) and *Well-Known Text* (WKT), and relies on a vocabulary for topological relationships and ontologies for qualitative reasoning [240]. The GeoSPARQL language extends SPARQL with geometric functionality to represent and query geo-enriched KGs. However, it still suffers from several limitations that restrict its effective use in practical settings, specifically when raster data plays a prominent role. The GeoSPARQL+ framework [112] provides an enhanced version of the GeoSPARQL vocabulary, query language, and ontology, extending RDF to support geospatial raster data. However, it has limited support for complex multidimensional raster and multi-polygonal vector data. Moreover, it relies on materialising geospatial data as RDF triples, resulting in very large KGs that may pose efficiency issues. One of the most famous Geospatial KG projects is LinkedGeoData (LGD) [211], which primarily relies on the VKG approach to expose OpenStreetMap OSM[9] data as RDF knowledge graphs.

Another popular GeoKG is YAGO2geo[10] [127], which contains detailed vector geometries collected from the (GADM) and (OSM). It is claimed to be the largest GeoKG, which currently contains 703 thousand polygons and 3.8 million lines [127]. YAGO2geo represents precise geographic knowledge by utilising the YAGO2 and GeoSPARQL ontologies, as well as specialised ontologies developed by the YAGO2geo team. Nevertheless, a query-answering system that supports the contained KG and temporal geographic knowledge is still not available. W3C standard *RDF Data Cube Vocabulary* [218] represents statistical data cubes but provides minimal support for representing the multidimensional array data model, e.g., raster data. These limitations are addressed by QB4OLAP [222], a vocabulary for BI over Linked Data that facilitates the representation of OLAP cubes in RDF and provides standard OLAP operations (including roll up, slice, dice, and drill-across) through SPARQL queries directly over RDF. Still, it does not support raster-based querying of complex EO data cubes.

GeoKGs align with geography's relational nature, where "everything is re-

[9]https://www.openstreetmap.org/
[10]https://yago2geo.di.uoa.gr/

lated to everything else," and address GIS limitations in handling quantitative and qualitative data [240]. Key research areas include knowledge representation (e.g., ontologies like GeoSPARQL and OWL-Time for spatial and temporal modelling) and reasoning (e.g., qualitative spatial reasoning via RCC8 and DE-9IM, integrated into GeoSPARQL) [240]. GeoKGs bridge Symbolic AI (logic-based) and Connectionist AI (machine learning), as demonstrated in studies employing KG embeddings to learn spatial/temporal rules or neural networks for advanced spatial relations [240, 160]. This facilitates the development of a neurosymbolic GeoAI, enabling the integration of theory and data to ethically guide AI systems in processing geospatial data [160].

Some other well-known geospatial knowledge graphs are Wikidata [224], WorldKG [72], KnowWhereGraph [123, 241], FineGeoKG [227].

3.3 Knowledge Representation & Reasoning

Knowledge Representation & Reasoning (KR&R), a foundational branch of artificial intelligence, e.g., symbolic AI, focuses on the formal encoding of information to enable computational systems to process and reason about it in a manner akin to human cognition [209]. At its core, this discipline involves creating machine-interpretable models of real-world domains, where symbols act as proxies for entities such as objects, events, or relationships [213]. These models facilitate automated reasoning, allowing systems to derive new insights from explicitly stated facts. A knowledge-based system typically maintains a repository of domain-specific assertions and employs inference mechanisms to respond to queries or support decision-making processes. The principles underlying knowledge representation emphasise formality, explicitness, and machine-processability. Formality ensures that representations are grounded in well-defined semantics, enabling unambiguous interpretation by algorithms. Explicitness requires that all relevant domain knowledge be articulated clearly, avoiding implicit assumptions that could lead to misinterpretation. Machine-level interpretation requires that the representation be compatible with computational tools for storage, retrieval, and manipulation. Methods for developing such representations include conceptual modelling, in which domain experts identify key entities and relations, and axiomatisation, which formalises these elements using logical constructs [220]. In practice, these methods are applied in scenarios such as business trip planning, where systems must integrate heterogeneous data sources to automate processes, including booking and matching offers to requests [213].

Knowledge representation manifests in several forms, each suited to different aspects of domain modelling and reasoning. Semantic networks, originating in early graphical logics such as Peirce's existential graphs (1896), represent knowledge as directed graphs with nodes denoting concepts and arcs indicating relationships [209]. This structure excels at capturing taxonomic hierarchies and relational links, such as "is-a" (subsumption) or "part-of" relations.

3.4 Ontology

In 1993, Gruber et al., [94] defined the term *ontology* as a subdivision of metaphysics that pertains to the philosophical examination of existence in its original context. It addresses the essential enquiries of ”what constitutes existence?” and ”what categories of entities exist?” [61]. In AI and computer science, ontologies are conceptual models that encode knowledge about any *domain of interest* and are rendered into a machine-processable format (e.g, RDF graphs) through knowledge representation [213]. More specifically, ontologies describe the domain of interest in terms of *concepts* (a.k.a classes), *roles* (a.k.a object properties), and *attributes* (a.k.a data properties) [220, 200].

3.4.1 Components

The ontology provides the schema and rules for interpreting the entities and facts comprising the domain knowledge. An ontology has the following main components [183] :

- ***Classes (or concepts)***, are the primary formalised components of the domain of interest.

    Let’s assume an ontology describing a *PhD degree* in Computer Science, representative classes include *Person*, *PhDStudent*, *Supervisor*, *ResearchArea*, *Course*, and *Thesis*. These classes may be hierarchically organised; for example, *PhDStudent* and *Supervisor* are subclasses of *Person*, while *Artificial Intelligence* is a subclass of *ResearchArea*, alongside others such as *BigData* or *Natural Language Processing*. Each class is associated with specific properties and constraints that characterise its members.

- ***Instances (or objects)*** represent concrete individuals of the classes inside a domain as defined by the ontology structure.

    For example, an individual named *Alice* may be an instance of the class *PhDStudent*, while *Prof. John Doe* may be an instance of the class *Supervisor*. Similarly, *Deep Learning* and *Cloud Computing* can be instances of the class *Course*. These instances populate the ontology with concrete domain data.

- ***Relations (or properties or predicates)*** define the connections between classes and instances, capturing both taxonomic and non-taxonomic relationships within the ontology. A relation often belongs to a specific type or class that delineates the manner in which one object is associated with another within the ontology.

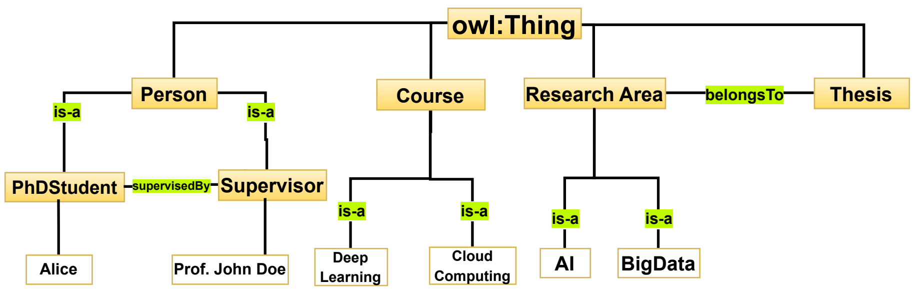


Figure 3.4: Ontology Example: PhD degree in Computer Science

In the PhD ontology, examples of relations include *supervisedBy* (linking a *PhDStudent* to a *Supervisor*), *belongsToResearchArea* (linking a *PhDStudent* or *Supervisor* to a *ResearchArea*), and *enrolledIn* (linking a *PhDStudent* to a *Course*). Relations may also be hierarchically organised; for instance, a general relation *participatesInAcademicActivity* may subsume more specific relations such as *teachesCourse* and *attendsCourse*.

- ***Axioms*** are the constraints, rules, and logical correspondences that must be adhered to in the relationships of ontology elements. Axioms can also be seen as the smallest unit of knowledge within an ontology and they ensure the semantic consistency and correctness of the ontology [216].

  For example, an axiom may state that every *PhDStudent* must be associated with exactly one *Supervisor*, or that a *Thesis* must be supervised by a *Supervisor* whose *ResearchArea* includes *Artificial Intelligence*. Another axiom may state that an individual cannot be an instance of both *PhDStudent* and *Supervisor* simultaneously. Such axioms encode domain knowledge at a formal level and enable reasoning over the ontology.

Figure. 3.4 presents the class taxonomy of the example PhD ontology. All domain concepts are modelled as subclasses of `owl:Thing`, with *Person, ResearchArea, Course*, and *Thesis* as top-level classes. Specialised concepts such as *PhDStudent, Supervisor* and *Artificial Intelligence* are defined through subclass relationships.

3.4.2 Types of Ontologies

Ontologies in computer science are classified primarily based on their subject of conceptualisation, which determines their level of generality and potential for reuse in knowledge-based system engineering [96]. This classification forms an inclusion hierarchy in which more general ontologies offer broader interoperability and reuse, while specific ones narrow the scope to particular contexts [96]. Figure 3.5 illustrates an inclusion hierarchy, wherein lower ontologies inherit and specialise concepts and relations from those positioned above them. The lower ontologies exhibit greater specificity and consequently have a more limited application scope, while the upper ontologies offer greater potential for reuse.

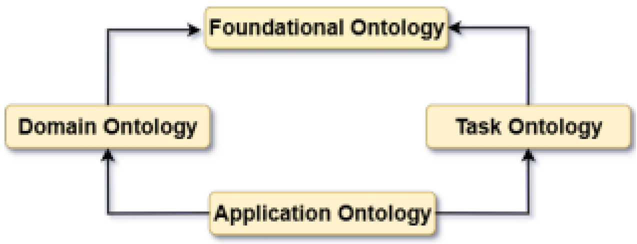


Figure 3.5: Types of ontologies

3.4.2.1 Foundational Ontologies

Foundational ontologies, also known as upper or top-level ontologies, have the broadest scope and aim to describe abstract, general concepts that are independent of specific domains or applications. They draw from philosophical notions to model universal phenomena such as time, space, processes, physical objects, and abstract entities. These ontologies are extensively axiomatised for precision and serve as alignment points for more specialised ontologies, rather than being used directly in applications. Their broad generality enables reuse across diverse fields.

Example: The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [82] models core entities, such as *endurant* (persistent entities, like physical objects) and *perdurant* (events unfolding over time), which can serve as anchors for ontologies in various fields, including engineering and linguistics. For instance, the city of Munich can be classified as an *endurant* (a physical object persisting through time with spatial location), while events such as the annual Oktoberfest held in Munich are modelled as *perdurants.*

3.4.2.2 Domain Ontologies

Domain ontologies narrow the scope to a specific domain of discourse, capturing knowledge about entities, relations, and constraints within that area while remaining independent of particular tasks. They provide a detailed conceptual model for a field, facilitating shared understanding and data integration. Compared to top-level ontologies, their scope is more restricted but still reusable across tasks within the domain.

Example: A domain ontology in a geographic context might include concepts such as *City*, *River*, and *Mountain*, along with relations like *flowsThrough* (e.g., the Isar flows through the City of Munich) and *locatedIn* (e.g., Munich lis ocated in the state of Bavaria), which are used in applications such as mapping or environmental analysis.

3.4.2.3 Task Ontologies

Task ontologies focus on a specific activity or process and describe the knowledge required for that task in a domain-neutral way. Their scope is limited to the generic structure of the task, including roles, steps, and constraints, which allows for reuse across different domains where the same task applies. This type complements domain ontologies by providing a procedural layer.

Example: A scheduling task ontology could define concepts such as *Task*, *Resource*, and *Constraint*, with relations like *precedes* (e.g., one task must precede another), applicable in intelligent computer-based tutoring, execution of clinical guidelines, etc.

3.4.2.4 Application Ontologies

Application ontologies have the narrowest scope, tailoring concepts to a specific task enactment within a particular application context. They integrate and specialise elements from domain and task ontologies, often assigning roles to domain entities in the context of the task. This makes them highly specific, with limited reuse beyond the targeted system, but essential for practical implementations.

Example: In a business trip booking application, an ontology might combine travel domain concepts (e.g., *Flight* and *City* such as Munich as a common departure hub) with booking task concepts (e.g., *Reservation* process), defining roles like a *Company* headquartered in Munich as a booker and an *Employee* as a participant in a flight from Munich to Stockholm.

3.5 Description Logic (DL)

Description Logics (DLs) [17, 18] serve as the foundation for contemporary knowledge representation systems that depict the universe of discourse in a logically consistent and computationally tractable manner. As the building blocks of this domain, DLs employ two fundamental entities: *concepts* and *roles*. Concepts denote unary relations, while roles denote binary relations.

Both are constructed by applying specific constructs starting from atomic concepts and roles.

DLs allow for modelling a domain of interest by employing only unary and binary predicate symbols known as *concept names* and *role names* within a *knowledge base* (KB) [200]. In DL, a knowledge base $\mathcal{K}$ is usually expressed as a pair of components, the ***TBox*** $\mathcal{T}$ and the ***ABox*** $\mathcal{A}$. A ***TBox*** is the intentional level of the KB, capturing the terminological knowledge (or schema) of the domain. An ***ABox*** comprises information at the instance level, providing concrete instances of the concepts and roles defined in the ***TBox***.

Our research concerns with a specific family of DLs known as *DL-Lite* [53, 185, 11], which aligns with the tractable OWL 2 QL[11] [170, 139] profile of the Web Ontology Language OWL 2, and, specifically, we consider *DL-Lite*$_R$, which is one of the most expressive variants in the *DL-Lite* family for which a query-rewriting based approach to query answering can be adopted.

In *DL-Lite*$_R$, a ***TBox*** is defined as a finite set of intensional assertions of the form, $C_1 \sqsubseteq C_2$ (*concept inclusion*), $C_1 \sqsubseteq \neg C_2$ (*concept disjointedness*), $R_1 \sqsubseteq R_2$ (*role inclusion*), $R_1 \sqsubseteq \neg R_2$ (*role disjointedness*), and $(\mathsf{funct}\ R)$ (*functionality*). Here, R (possibly subscripted) denotes an atomic role P or its inverse P^-. Instead, C (possibly subscripted) denotes a *basic concept*, which is either an atomic concept A, or a concept defined as $\exists R$, representing the set of objects that appear as the first argument of R. Finally, $(\mathsf{funct}\ R)$ asserts the functionality of R, i.e., that its first argument operates as a key of the relation denoted by R. An ***ABox*** in *DL-Lite*$_R$ is represented as a finite set of assertions of the form $A(a)$ or $P(a, b)$, with a and b belonging to N_I.

The semantics of a DL KB is given in terms of First-Order interpretations, referring to the interpretation structures of First-Order Logic (FOL) [18]. An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of an interpretation *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$, which maps each atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role P to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The role P^- is interpreted as the inverse of the relation $P^{\mathcal{I}}$, while $(\exists R)^{\mathcal{I}} = \{o \mid \text{there exists } o' \text{ s.t. } (o, o') \in R^{\mathcal{I}}\}$. Finally, $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$, and $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is said to be a *model* of (or *satisfies*) an ***ABox*** assertion $A(a)$ (resp., $P(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$), and it is a model of a ***TBox*** assertion $E_1 \sqsubseteq E_2$ (where E_1 and E_2 are either concepts or roles) if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of all assertions in $\mathcal{T}$ and $\mathcal{A}$. We denote with $Mod(\mathcal{K})$ the set of all models of $\mathcal{K}$. A KB $\mathcal{K}$ is *consistent* if it has at least one model, i.e., $Mod(\mathcal{K}) \neq \emptyset$. We say that $\mathcal{A}$ is *consistent with* $\mathcal{T}$, or $\mathcal{T}$*-consistent*, if $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, and $\mathcal{T}$*-inconsistent* otherwise. Given an ***ABox*** $\mathcal{A}$ and a ***TBox*** $\mathcal{T}$, we denote by $cl_{\mathcal{T}}(\mathcal{A})$ the *closure* of $\mathcal{A}$ w.r.t. $\mathcal{T}$ that is, the set of ***ABox*** assertions over individuals in $\mathcal{A}$ that are logically implied by $\langle \mathcal{T}, \mathcal{A} \rangle$. Similarly, the deductive closure of a ***TBox*** $\mathcal{T}$, denoted $cl(\mathcal{T})$ is the set of inclusions and functionality assertions that follow from $\mathcal{T}$. Finally, given two ***ABox*** es $\mathcal{A}$ and

[11] https://www.w3.org/TR/owl2-profiles/#OWL_2_QL

$\mathcal{A}'$, we say that $\mathcal{A}$ is logically equivalent to $\mathcal{A}'$ w.r.t. $\mathcal{T}$, denoted $\mathcal{A} \equiv_{\mathcal{T}} \mathcal{A}'$ if $cl_{\mathcal{T}}(\mathcal{A}) = cl_{\mathcal{T}}(\mathcal{A}')$.

3.6 Virtual Knowledge Graph (VKG)

Also known as *Ontology-Based Data Access* (OBDA) [185, 232, 233] that provides a data management framework that simplifies access to heterogeneous data sources (e.g., relational) for end-users by letting them formulate high-level queries over a conceptual representation of the *domain of interest*, provided in terms of a domain ontology. The underlying data sources are accessed via a domain ontology that links to the sources via semantic mappings and exposes the data as a "virtual" KG represented in RDF. This enables connecting data silos through conceptual graph representations that provide an integrated view of the data. The three main ideas of VKG are:

- ***Virtualisation (V)***: VKGs provide a conceptual view that shields end users from the actual data sources. This conceptual view is typically not materialised to enable querying the data without paying a price in terms of storage and time to make it accessible.
- ***Domain Knowledge (K)***: Graphs can be enriched and contextualised with domain knowledge that makes it possible to derive new implicit knowledge from the asserted facts at the time a query is executed.
- ***Graph Representation (G)***: The data are represented as a graph where object and data values are represented as nodes, and properties of those objects are represented as edges. Compared with traditional relational integration tables, graph representations offer greater flexibility and, through mapping and merging, facilitate data linking and integration.

3.6.1 VKG specification

A *VKG specification* $\mathcal{P} = (\mathcal{O}, \mathcal{M}, \mathcal{S})$ consists of *(i)* an *ontology* $\mathcal{O}$ expressed as a TBox $\mathcal{T}$ in the lightweight ontology language OWL 2 QL, *(ii)* a relational *data source schema* $\mathcal{S}$, and *(iii)* a declarative mapping $\mathcal{M}$ that associates to each element (i.e., class or property) in $\mathcal{O}$ a (SQL) query over $\mathcal{S}$, specifying how to (virtually) populate that element through the data retrieved from the source. . In traditional VKGs, the mapping $\mathcal{M}$ consists of a set of R2RML [62] *mapping assertions* of the form

$$Q_{sql}(\vec{x}) \rightsquigarrow E(\vec{f}(\vec{x})),$$

where $Q_{sql}(\vec{x})$ is a SQL query (also called *source query* or CQ) over $\mathcal{S}$ of arity $n > 0$ with answer variables $\vec{x}$, and the *target* $E(\vec{f}(\vec{x}))$ consists of a class or property E of $\mathcal{O}$, and a set $\vec{f}(\vec{x})$ of so-called *IRI-templates* applied to the variables in $\vec{x}$. Each IRI-template is a function that constructs an ontology literal or an IRI identifying an ontology object, from the values in each answer to $Q_{sql}(\vec{x})$ instantiating $\vec{x}$. Such IRI-templates[12] are used to generate strings representing object IRIs or (RDF) literals, starting from DB values retrieved by the source query in the mapping.

A *VKG instance* is a pair $(\mathcal{P}, \mathcal{D}^{rel})$, where $\mathcal{D}^{rel}$ is a relational database instance conforming to $\mathcal{S}$. By "applying" the mapping assertions in $\mathcal{M}$ to $\mathcal{D}^{rel}$, i.e., by evaluating each source query $Q_{sql}(\vec{x})$ over $\mathcal{D}^{rel}$ and using the returned answers to instantiate the target $E(\vec{f}(\vec{x}))$, one obtains a KG $\mathcal{M}(\mathcal{D}^{rel})$, which, however, is kept 'virtual'. Semantic queries formulated in SPARQL are posed over $\mathcal{O}$ and are answered by accessing the relational data $\mathcal{D}^{rel}$ through the mapping $\mathcal{M}$. Specifically, given a SPARQL query q over a VKG instance $\mathcal{J} = (\mathcal{P}, \mathcal{D}^{rel})$, we are interested in the *certain answers* to q over $\mathcal{J}$, denoted $\mathsf{cert}(q, \mathcal{J})$, which are the answers obtained by evaluating q over the knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{M}(\mathcal{D}^{rel}))$ under the OWL 2 QL entailment regime.

Actual VKG systems, such as ONTOP [52, 234] avoid costly materialisation of the KG $\mathcal{M}(\mathcal{D}^{rel})$ and its storage in a triple store, and instead translate the SPARQL query into a relational query (e.g., in SQL) that is directly evaluated by the underlying RDBMS (e.g., PostgreSQL), thus ensuring also the freshness of query answers concerning source updates.

3.6.2 Addressing Data Heterogeneity

As discussed in motivation (cf. Section 1.1), data heterogeneity, which includes *syntactic* (e.g., formats), *structural* (e.g., schemas), and *semantic* (e.g., vocabularies) heterogeneity, is a core challenge in the semantic web and AI, where integrating data silos leads to inconsistencies that hinder unified query processing [233]. VKG addresses this by virtualising data under a unified ontology, using mappings to reconcile differences in representation, without physical integration [232].

- ***Structural Heterogeneity***: VKG mappings handle schema mismatches by connecting ontology axioms to underlying data structures, enabling federated queries over disparate schemas. Kharlamov et al. [132] demonstrated this resolution by providing ontology-based access to more than 1500 relational tables within Statoil's enterprise databases.

- ***Syntactic Heterogeneity***: VKG systems abstract over heterogeneous relational data formats through virtualisation and data-access adapters; for instance, ONTOP-spatial [31] spatially enables VKG systems to import diverse relational-formatted geospatial data into compatible relational backends via PostGIS, a spatial extension of PostgreSQL.

- ***Semantic Heterogeneity***: Ontologies provide a shared conceptual vocabulary that resolves differences in meaning across data sources. Logical

[12]IRI-templates correspond to the R2RML string templates

axioms and class hierarchies enable semantic alignment by relating synonymous or context-dependent terms under common concepts. In the biomedical field, ICD9CM[13] ontology with the MIMIC-III database, harmonises clinical terminology and supports consistent interpretation across datasets [51].

In the context of geospatial data integration, VKG resolves vector data (river networks) from heterogeneous GIS repositories using ontologies for semantic enrichment [49]. Xiao et al. [232] explore VKG's role in Statoil, where VKG reduced query formulation time from days to minutes by addressing schema obscurity. Rodriguez-Muro et al. [193, 54] demonstrate VKG in the Optique project, integrating oil/gas data.

3.6.3 Query Answering

Query answering constitutes the core functionality of VKGs, enabling users to pose high-level SPARQL queries over an ontology while retrieving complete and sound results from underlying heterogeneous data sources without materialization [232, 233]. This process leverages the declarative nature of VKGs to abstract from low-level data details and incorporates ontological reasoning to infer implicit knowledge. Unlike traditional database querying, VKG query answering reduces to first-order logic reformulation, ensuring tractability under data complexity when the ontology and mappings are fixed and only the data varies [185].

The query answering process involves several interconnected steps (parsing, rewriting, unfolding, optimisation, and execution) [20] as described below:

- ***Rewriting***: Expand SPARQL into a union of conjunctive queries using ontology axioms to capture implicit knowledge [185].
- ***Unfolding***: Using declarative mappings to produce SQL, unfolding ontology terms into underlying database-level queries [20].
- ***Optimisation***: is crucial for scalability, addressing redundancies from large query rewritings. Techniques include semantic query optimisation (e.g., eliminating non-satisfiable subqueries via ontology constraints), structural simplifications (e.g., join minimisation), and database dependencies (e.g., pushing filters or aggregations) [147, 132]. Ontop employs non-recursive Datalog intermediates for efficient UCQ handling, reducing unfolding size [234]. Evaluations demonstrate superior performance compared to RDF stores, e.g., in geospatial benchmarks, where Ontop-spatial outperforms Strabon by leveraging PostGIS optimizations [31].

An example of query answering is presented in Calvanese et al. [55], where an SPARQL query over a biomedical ontology is rewritten in SQL, and patient eligibility for trials is inferred using temporal rules.

[13]https://bioportal.bioontology.org/ontologies/ICD9CM

3.6.4 Conclusion

Although VKG effectively handles relational data, including vector formats (points, lines, polygons), the management of raster data (multidimensional arrays) is constrained by scalability and expressivity [31, 233]. Current VKGs focus on vector operations via GeoSPARQL, but raster queries (e.g., array aggregations) require extensions to mappings and ontologies, often resulting in inefficient SQL translations for large arrays. Buccella et al. [49] argue that ontologies struggle with array semantics, and no paper provides a comprehensive solution to multidimensional raster integration. In one of our papers [87], the support for raster data has also been attempted in classical RDBMS through spatial extensions. To achieve this, extensive raster data must undergo a conversion (e.g., using *raster2pgsql*[14] for PostgreSQL extended with PostGIS) from their native format of multidimensional arrays to a suitable relational form (e.g., *postgis raster*) that can be queried using SQL-like query language. This conversion typically increases the data volume substantially, which can be challenging to manage in a relational table.

Hence, the limitations of VKGs managing raster data in its current state are: (i) Lack of native array support in OWL/SPARQL; (ii) Performance bottlenecks in relational backends; (iii) Reasoning over arrays is undecidable in standard DLs. To address these limitations, this thesis proposed an alternative approach that does not rely on the RDBMS's internal raster data materialisation, as discussed in the next section.

[14] https://postgis.net/docs/using_raster_dataman.html#RT_Loading_Rasters

Chapter 4

OntoRaster - A Novel VKG Framework

This chapter is based on the works of our papers [87, 86] that describes OntoRaster, an extended VKG framework for semantic integration and query answering of heterogeneous relational, vector, and raster data by devising custom raster functions for SPARQL, leveraging PostgreSQL as a federator, and delegating raster operations to RasDaMan for efficient, conversion-free processing—minimising data transfers while supporting hybrid queries. Key features include raster functions that extend SPARQL for efficient raster operations and are delegated directly to the array database management system (i.e., RasDaMan). We define *Raster Ontology* and link it to raster data sources via mappings, enabling semantic query answering without materialising rasters. OntoRaster supports complex vector geometries (e.g., multi-polygons, polygons with holes) and incorporates public vector datasets (e.g., GADM, OpenStreetMap, and CityGML) along with corresponding tailored ontologies.

4.1 State of the Art

Here, we survey prominent works on integrating heterogeneous data sources, particularly geospatial raster data and relational data (including vector geometries, e.g., polygons), using VKG, describe each approach, and compare it with our proposed OntoRaster framework.

4.1.1 Materialisation Approach

Sun et al. [215]) propose a theoretical ontology framework, GeoDataOnt, to address general semantic heterogeneity in geospatial data integration and sharing. It analyses semantic issues across geospatial data lifecycles, categorising characteristics into three compound modules: the essential ontology (e.g., themes, spatial/temporal coverage), the morphology ontology (e.g., data types, formats, structures), and the provenance ontology (e.g., sources, processes). GeoDataOnt lacks practical querying mechanisms and native support for multidimensional array operations under the VKG paradigm, unlike OntoRaster. Mai et al. [161] develop a linked data connector for Esri's ArcGIS, enabling ontology-based retrieval and analysis of RDF data (e.g., from DBPedia) within GIS. It supports reasoning, vector spatial analysis, and visualisations, but it requires KG materialisation and lacks raster support. OntoRaster virtualises raster and vector data for query processing without any data conversions and GIS tools.

A notable work GeoSPARQL+ [112] extends GeoSPARQL with raster filters (e.g., aggregation) for hybrid graph-raster-vector querying. It provides well-formalised semantics for raster data, and a prototype validates its feasibility on datasets such as aerial imagery for risk assessment. GeoSPARQL+ materialises raster data alongside vector data for further query processing, without VKG virtualisation. OntoRaster relies on the virtualisation of VKGs and RasDaMan to enable native, conversion-free raster querying, thereby integrating relational data in a more scalable manner. Tran et al. [219] addresses semantic integration of EO raster data by proposing (i) a modular ontology for describing spatial-temporal features of predefined geographic units (e.g., land cover over time) utilizing standards like GeoSPARQL for vector geometries and OWL-Time[1] for temporality; (ii) a semantic ETL process to extract pixel values from rasters, compute statistics (e.g., land cover ratios), and link them to RDF representations of territorial units; and (iii) an RDF triples store dataset exposed via a SPARQL endpoint and semantic interface. Evaluations emphasise semantic querying but note scalability limits for large rasters. This study emphasises materialised RDF integration via ETL, requiring raster pixel extraction and statistical computation upfront, which can entail data conversion and storage overhead for large EO datasets—lacking virtualisation for on-the-fly access, such as OntoRaster.

Liu et al. [157] propose a KG-based retrieval method for multi-source heterogeneous geospatial vector data, including OSM data. It constructs a domain-specific KG by extracting implicit semantics and relationships, then applies query-expansion rules (conceptual similarity and entity associations) and KG-to-DB mappings to automatically generate SQL statements. But it lacks support for raster data cubes or multidimensional operations, relying on SQL generation over spatial databases without native array handling.

4.1.2 Virtualisation Approach

Bereta et al. [31] present Ontop-spatial, an extended VKG engine Ontop that virtualises geospatial vector data stored in RDBMS (e.g., PostGIS) as RDF graphs. It adapts GeoSPARQL as an SPARQL entailment regime and augments SPARQL-to-SQL translations for geospatial operations, such as spatial

[1]`https://www.w3.org/TR/owl-time/`

joins and topological relations, but does not support multidimensional raster arrays. In contrast, OntoRaster integrates raster cubes with vector and relational data, extends GeoSPARQL to RasSPARQL to support raster-specific functions (e.g., multidimensional aggregations), and delegates these operations to RasDaMan for native array querying without conversions or SQL approximations, thereby enabling more scalable hybrid spatiotemporal queries. Ontop-temporal [126] is another extension of Ontop with OWL 2 QL ontologies and metric temporal logic rules for querying relational timestamped logs (1D time series data) in the biological domain. But there is no mention of spatial data (vector or raster). OntoRaster, however, supports spatiotemporal querying over raster data with vector geometries, utilising RasDaMan's temporal array operations, e.g., temporal aggregation and slicing without conversion.

Almobydeen et al. [5] propose GeoLD query engine, which extends SPARQL with WCPS operators for scientific raster array data using the *Coverage to RDF Mapping Language (C2RML)*, an extension of the R2RML under VKG paradigm. It maps coverages to RDF and optimises by delegating to WCPS, outperforming relational GeoSPARQL on meteorological data. GeoLD delegates raster queries via WCPS but focuses on raster-only access without vector-relational semantics. OntoRaster relies on RasSPARQL with raster functions and GeoSPARQL capabilities for hybrid queries, using RasDaMan directly without WCPS, enhancing expressivity (e.g., multi-polygon clipping).

Hamdani et al. [101] proposed an architecture to query raster data stored in an RDBMS using the VKG paradigm and extending the GeoSPARQL ontology[2] with a semantic representation model for raster data cubes. Using R2RML and SPARQL, it supports spatiotemporal queries; evaluations compare scalability to other systems. But this work involves converting and materialising raster data for query processing, whereas OntoRaster relies on RasDaMan for direct array management without conversions, enabling more precise, efficient hybrid operations via RasSPARQL. A recent work [38] enhances Apache Jena for GeoSPARQL 1.1 with vector data, adding transformations, aggregations, and H3 grid [3] support, with grid approximations for rasters. OntoRaster handles full multidimensional rasters natively via RasDaMan without grid approximations.

Plato [34, 35, 164], a notable semantic data cube system, has been proposed within the EU H2020 project DeepCube[4]. The system relies on a geospatial extension of the VKG system, i.e., Ontop-spatial [31] and uses PostgreSQL's foreign-data wrappers (FDW)[5] to virtualise arrays (e.g., from DeepCube) as RDF graphs, linking numerical values to concepts (e.g., heatwaves) via SPARQL. It has been deployed in a case study on fire risk-management in Greece [36]. Plato queries EO raster data through relational wrappers for array access, potentially involving conversions and lacking seamless vector topology

[2]https://www.ogc.org/standard/geosparql/
[3]https://h3geo.org
[4]https://deepcube-h2020.eu/
[5]https://www.postgresql.org/docs/current/postgres-fdw.html

integration. OntoRaster advances this by using array DBMS RasDaMan natively for array operations without conversions, extending GeoSPARQL to RasSPARQL for unified hybrid queries (e.g., raster aggregations within vector regions), and reducing overhead in large-scale raster data scenarios.

Xiao et al. [235] introduce a KG-based framework for unifying Germany's ATKIS-DLM[6] a vector dataset, which is distributed as fragmented shapefiles across themes (e.g., settlements, hydrography) and geometries. It develops an OWL/GeoSPARQL ontology from DLM documentation, uses R2RML mappings to link vector data in a PostGIS database, and supports virtualisation via Ontop (SPARQL-to-SQL rewriting) or materialisation in triple stores like GraphDB. It only mentioned raster data inclusion as future work. Ranatunga et al. [188] introduce a VKG framework for integrating heterogeneous geospatial vector data (e.g., lakes and roads). This framework employs an extended GeoSPARQL ontology, utilises PostGIS for data storage, Ontop as a reasoning engine, and a web-based SPARQL Query Interface (SQI) with Leaflet for tabular and map visualisations. It accommodates both spatial (e.g., intersections) and non-spatial queries; however, it lacks raster support. OntoRaster's delegated raster processing capability provides greater efficiency for EO-intensive use cases, thereby mitigating the vector focus and temporal gaps. One of our collaborators, Pano et al. [180], introduced ontopEO, a tool that leverages the VKGs paradigm to query EO raster data via the ontopEO [182] API calls and does not rely on array DBMS such as RasDaMan. It is mainly tailored to query EO data from the European Space Agency (ESA) Copernicus platform, leveraging a higher-level conceptualisation enabled by a domain ontology.

Most of these works emphasise the integration of heterogeneous vector data within a specific region and lack the semantic integration and virtualisation of raster data under the VKG paradigm. Whereas OntoRaster extends VKG paradigm by integrating raster cubes through RasDaMan, using RasSPARQL for raster-specific functions (e.g., aggregation over raster arrays) and PostgreSQL federation to minimise data transfers—enabling hybrid queries like raster values within OGC complement arbitrary vector geometries over any arbitrary region.

4.2 Three Layer Architecture

In this section, we describe the architecture of OntoRaster, which supports seamless integration and on-the-fly query answering across relational, vector, linked, and raster data, while minimising costly data transfer operations under the VKG paradigm. The architecture is organised in three layers: the *Data Source Preparation Layer*, which interfaces with the underlying heterogeneous data sources, the *VKG Layer*, which provides integrated SPARQL access to these sources using the VKG paradigm via the Ontop system and *Query Translation Layer*, which bridges the previous two layers by using *stored procedures*

[6] https://www.ioer-monitor.de/en/methodology/glossary/a/atkis-basis-dlm/

in PL/Python and PL/pgSQL to issue queries to an array DBMS Rasdaman and combine the results with relational data, including vector data.

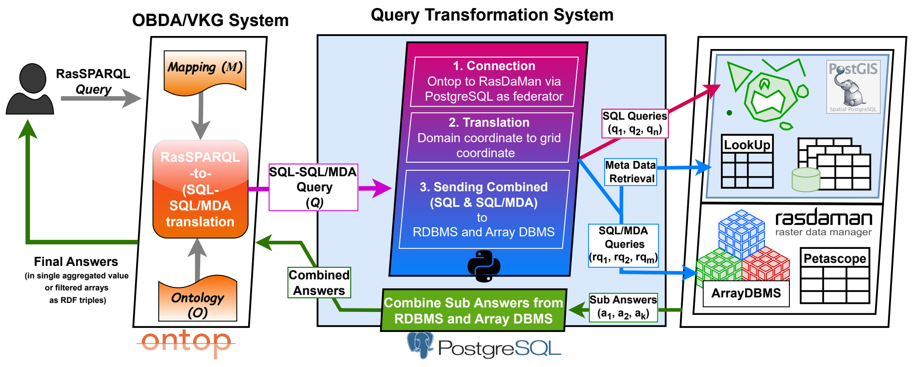

OBDA/VKG System
Query Transformation System
RasSPARQL Query
Mapping $(\mathcal{M})$
RasSPARQL -to- (SQL- SQL/MDA translation
Ontology $(\mathcal{O})$
ontop
Final Answers
(in single aggregated value or filtered arrays as RDF triples)
SQL-SQL/MDA Query $(\mathcal{Q})$
Combined Answers
1. Connection
Ontop to RasDaMan via PostgreSQL as federator
2. Translation
Domain coordinate to grid coordinate
3. Sending Combined (SQL & SQL/MDA) to RDBMS and Array DBMS
Combine Sub Answers from RDBMS and Array DBMS
PostgreSQL
SQL Queries (q_1, q_2, q_n)
Meta Data Retrieval
SQL/MDA Queries (rq_1, rq_2, rq_m)
Sub Answers (a_1, a_2, a_k)
PostGIS
LookUp
rasdaman
raster data manager
Petascope
ArrayDBMS

Figure 4.1: OntoRaster – An Extended VKG framework

Figure 4.1 illustrates the high-level architecture of OntoRaster, depicting the flow from user queries to integrated data processing across heterogeneous sources. The diagram is divided into three main sections: *(i)* the OBDA/VKG System on the left, *(ii)* the Query Transformation System in the centre, *(iii)* the Heterogeneous Data Sources on the right.

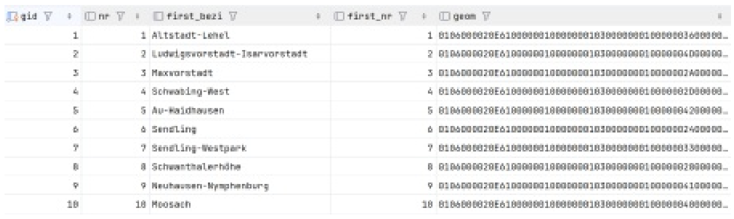
4.3 Data Source Preparation Layer

The OntoRaster framework supports geospatial raster and vector data, as well as generic tabular datasets, including publicly available linked data collections. In this dissertation, four areas of interest (AOIs) of progressively increasing spatial extent, geometric complexity, data volume, and organisational diversity have been selected (Figure 5.1): the city of Munich, the federal state of Bavaria, the autonomous province of South Tyrol, and the entire country of Sweden. The OntoRaster architecture is expressly designed to accommodate any geographic AOI or custom region, provided that datasets of comparable type and structure are available. The four selected AOIs are therefore illustrative rather than restrictive.

4.3.1 PostgreSQL and PostGIS

At the core of the OntoRaster architecture is PostgreSQL, a robust relational database management system (RDBMS), extended with the PostGIS spatial module. PostGIS is a widely adopted[7] open-source geospatial extension for relational databases, such as PostgreSQL, that enables efficient storage, indexing, and processing of both vector and raster geospatial data within the SQL

[7]https://db-engines.com/en/ranking/spatial+dbms



gid	nr	first_bezi	first_nr	geom
1	1	Altstadt-Lehel	1	0106000020E6100000010000000103000000010000003600000…
2	2	Ludwigsvorstadt-Isarvorstadt	2	0106000020E6100000010000000103000000010000004D00000…
3	3	Maxvorstadt	3	0106000020E6100000010000000103000000010000002A00000…
4	4	Schwabing-West	4	0106000020E6100000010000000103000000010000002D00000…
5	5	Au-Haidhausen	5	0106000020E6100000010000000103000000010000004200000…
6	6	Sendling	6	0106000020E6100000010000000103000000010000002400000…
7	7	Sendling-Westpark	7	0106000020E6100000010000000103000000010000003300000…
8	8	Schwanthalerhöhe	8	0106000020E6100000010000000103000000010000002800000…
9	9	Neuhausen-Nymphenburg	9	0106000020E6100000010000000103000000010000004100000…
10	10	Moosach	10	0106000020E6100000010000000103000000010000004000000…

Figure 4.2: 25 Districts of Munich stored in VectorTablesDB

environment. Vector datasets are imported from standard interchange formats (shapefile, GeoJSON, GeoPackage, etc.) using either the dedicated `shp2pgsql`[8] utility or the versatile `ogr2ogr` tool from the Geospatial Data Abstraction Library (GDAL/OGR[9]) [194]. These capabilities make PostgreSQL+PostGIS the natural choice for consolidating heterogeneous geospatial vector and tabular data within the OntoRaster federation layer.

In practice, each AOI consists of multiple regions; for example, our running example AOI Munich has two distinct sets of regions: Districts and Subdistricts and their respective vector data together with at least three attributes, e.g., `regionId`, `regionName` and `regionGeometry`, are stored in dedicated relational tables within a PostgreSQL database named *VectorTablesDB*. Figure 4.2 shows the *munich_dist25* table in VectorTablesDB, which depicts 10 of Munich's 25 districts. It has five columns such as `gid` (or `regionId`), `nr`, `first_bezi` (or `regionName`), first_nr and *geom* (or `regionGeometry`). Every table contains exactly one geometry column registered with a specific *spatial reference system* (SRS), which defines semantic meaning to the stored coordinate values. Source vector datasets, obtained as shapefiles from authoritative providers (e.g., GADM, geoBoundaries, OpenStreetMap), are imported via `shp2pgsql` in hex-encoded Well-Known Binary (WKB) format.

Notice that all table names and column names mentioned above are illustrative for conceptual understanding and employed throughout this work. Any consistent naming convention may be adopted without loss of functionality, provided the same names are used in the VKG mapping assertions that connect the ontologies to the underlying heterogeneous data sources.

In our architecture, we utilise the procedural language PL/Python within PostgreSQL to facilitate integration with the array DBMS RasDaMan. This approach permits unrestricted use of the entire Python ecosystem directly inside the PostgreSQL back-end. The current implementation uses Python 3.13, PostgreSQL 16, and Shapely 2.0.2 to convert between PostGIS well-known text (WKT) representations and Rasdaman-compatible geometric specifications.

[8]https://postgis.net/docs/using_postgis_dbmanagement.html#shp2pgsql_usage
[9]https://pcjericks.github.io/py-gdalogr-cookbook/

4.3.2 RasDaMan ("Raster Data Manager")

Rasdaman is a domain-agnostic array DBMS that implements the OGC standards for gridded coverages (i.e., multi-dimensional raster data) and manages large datasets through its rich array algebra. Rasdaman offers a SQL-like query language known as *RaSQL*[10] (*Rasdaman Query Language*) to query any kind of raster data of arbitrary dimensions [26] following *ISO 9075 SQL Part 15: Multi-Dimensional Arrays* (SQL/MDA)[11]. It features a geo service front-end component called *petascope*[12], which adds geo semantics on top of arrays, thereby enabling regular and irregular grids based on the OGC CIS v1.1. It also supports the OGC standards *Web Feature Service (WFS)* [178], *Web Coverage Service (WCS) 2.0* [177], *Web Map Service (WMS)* [179] and *Web Coverage Processing Service (WCPS)* [25]. Petascope utilises a relational database, such as PostgreSQL, to store related metadata for each raster dataset, thereby providing the respective geo-semantics. This metadata is distributed across 62 relational tables in a separate PostgreSQL database called *petascopedb* [4]. Rasdaman also includes *rasdapy*[13], a client API that enables building and executing RaSQL queries in Python.

4.3.3 Raster LookUp Table Creation

The prerequisite metadata included in petascopedb must accompany the correct raster. Whenever a user queries specific raster data, the pertinent metadata must be retrieved from 62 distinct tables in petascopedb and automatically associated with the underlying raster data for further processing. When the user selects an alternative raster dataset for querying, the automatic search and search-and-combine procedures are repeated, incurring additional overhead. To resolve this issue, we developed *Raster Lookup*, a comprehensive table that retains all essential metadata for each raster dataset stored in rasdaman. We have built this table inside the VectorTablesDB database from the petascopedb database using *dblink*[14] where petascopedb serves as a remote database. Upon uploading new raster data to rasdaman, a PostgreSQL trigger automatically refreshes the lookup table with the metadata for the newly uploaded raster.

4.4 VKG Layer

Traditional VKGs have no means of connecting to an array DBMS, nor do they support arrays natively. So, we introduce new methods to extend VKG's capabilities to support array data and array functions for querying raster data within the VKG layer.

[10] https://doc.rasdaman.org/04_ql-guide.html
[11] https://www.iso.org/obp/ui#iso:std:iso-iec:9075:-15:ed-2:v1:en
[12] https://doc.rasdaman.org/02_inst-guide.html?highlight=petascope#petascope
[13] https://pypi.org/project/rasdapy3/
[14] https://www.postgresql.org/docs/current/contrib-dblink-function.html

Table 4.1: RasSPARQL raster functions and respective PL/Python stored procedures

RasSPARQL function	**Input arguments**	**Output type**	**PL/Python stored proc.**
`rasDimension()`	`rasterName`	`xsd:string`	`query2string()`
`rasCellOp()`	`timeStamp, operator, operand, rasterName`	`xsd:string`	`query2array()`
`rasSpatialAverage()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasSpatialMinimum()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasSpatialMaximum()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalAverage()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalMinimum()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalMaximum()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasClipRaster()`	`timeStamp, regionGeometry, rasterName`	`xsd:string`	`query2array()`
`rasClipRasterAnyGeom()`	`timeStamp, regionGeometry, rasterName`	`xsd:string`	`query2array()`
`rasGeoTIFF()`	`timeStamp, regionGeometry, rasterName, fillNAN`	`xsd:string`	`query2array()`

4.4.1 RasSPARQL: SPARQL with Raster Functions Enabled

To facilitate querying over raster data within the VKG layer, OntoRaster introduces RasSPARQL, an extension of SPARQL that incorporates OGC GeoSPARQL functions for managing vector data, along with custom raster functions for managing multidimensional raster data. RasSPARQL allows users to express complex queries involving multidimensional arrays, such as computing aggregations or transformations, in a declarative manner. We first devised and added custom raster-based functions to SPARQL and corresponding SQL DB functions to be passed to RasDaMan for execution. The currently supported RasSPARQL raster-based functions with their input arguments and output type are listed in Table 4.1. These functions are integrated into RasSPARQL queries and translated into PostgreSQL PL/Python stored procedures, which delegate execution to rasdaman.

For example, a RasSPARQL query might compute the spatial average value over an arbitrary region, whose vector geometry intersects a temperature raster dataset at a particular timestamp using the introduced raster function `rasSpatialAverage()`. In this case, the function will take the following input, i.e., a particular datetime (e.g. `timeStamp`), arbitrary vector data (e.g. `regionGeometry`) and the name of chosen raster data `rasterName` and output a spatially aggregated value (in this case average temperature of that vector region). Table 4.1 also lists temporal aggregation raster functions such as `rasTemporalAverage()`, `rasTemporalMaximum()`, and `rasTemporalMinimum()`, which take start and end timestamps along with arbitrary vector data and raster data.

Some RasSPARQL functions, such as `rasClipRaster()` and `rasClipRasterAnyGeom()`, return a portion of raster data as an array of values (e.g., pixels) by relying on rasdaman's embedded 'clip' function[15], which extracts a raster array based on the geometry (or shape) specified by the vector data. All pixels external to the selected vector geometry are assigned a value of null or 'NaN'; however, pixels located on or within the geometry are retained. The returning arrays are expressed as strings because RDF presently lacks support for arrays [87]. The last function in Table 4.1 is `rasGeoTIFF()`, which returns a GeoTIFF version of clipped raster data which can be used directly in a GIS application.

4.4.2 RasSPARQL to SQL-SQL/MDA Translation

To properly deal with RasSPARQL functions within Ontop, we have implemented an extension of the system that, as a part of query reformulation, translates each such function into a corresponding SQL function and embeds it into the generated SQL/MDA query as indicated in Fig. 4.1. Notice that the RasSPARQL query is, in general, a SPARQL query that might contain both GeoSPARQL functions and raster functions. This query is translated into a plain SQL query that uses corresponding PostGIS functions and PL/Python stored procedures to connect to RasDaMan, as specified in the last column of Table 4.1. As an example, when Ontop parses a RasSPARQL query that embeds the raster function `rasSpatialAverage()`, it translates it to a call to `query2numeric()`, which in turn executes the SQL/MDA standard RaSQL query over RasDaMan.

4.4.3 Raster Ontology

In the context of geospatial raster (gridded) data management, we have defined the *Raster Ontology* shown in Fig. 4.3, which provides a structured framework for representing $\mathcal{N}$-dimensional multidimensional arrays (e.g., remote-sensing imagery, climate simulations, and time-series data) within the Semantic Web paradigm. Inspired by Andrejev et al. [7], this ontology integrates generic raster data with metadata using the RDF model, enabling machine-readable semantics for spatial-temporal grids. It builds on the OGC standard *coverage* model (ISO 19123:2005) [27] and supports efficient querying via SPARQL extensions. The ontology addresses the impedance mismatch between array-based data and irregular metadata, enabling hybrid storage in which arrays are embedded as nodes in RDF graphs.

The core objective of the Raster Ontology is to capture essential metadata to support understanding and querying of grid coverages while remaining extensible to application-specific details. It models coverages as functions mapping from a domain set (spatial-temporal extents) to a range set (attribute values), with support for regular, rectified, and referenceable grids. This facilitates integration with semantic web vocabularies and enables queries that combine data selection, processing, and metadata retrieval in a single declarative statement.

[15]https://doc.rasdaman.org/04_ql-guide.html#clipping-operations

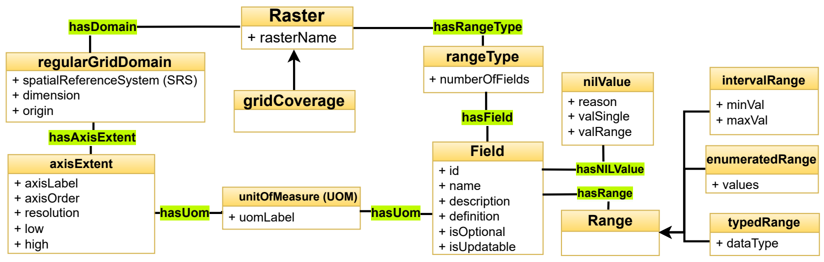


Figure 4.3: Ontology for generic raster data (including grid coverage)

4.4.3.1 Key Classes and Properties

The ontology is formalised in RDF Schema (RDFS) and comprises several interconnected classes, properties, and relationships. Below, we outline the primary components, drawing on the conceptual model presented in [7].

1. **GridCoverage** This central class represents the grid coverage as a whole, modelling a spatial-temporal phenomenon discretised into a grid.

   Properties:

   - `rasterName`: A unique name for the gridded raster or grid coverage (e.g., "myCoverage").
   - `hasDomain`: Links to a `GridDomain` (or subclass) describing the spatial-temporal extent.
   - `hasRangeType`: Links to a `RangeType` defining the structure of cell values.

   Semantics: Instances hold array-valued properties corresponding to fields (e.g., NIR or red channels in satellite imagery). Arrays are stored efficiently (e.g., via proxies to array databases such as rasdaman) and linked to metadata for lazy evaluation during queries.

2. **GridDomain** (and Subclasses) Describes the domain set, defining the grid's spatial-temporal bounds and topology. Supports regular, rectified, and referenceable grids.

   Properties:

   - `dimensionality`: Integer specifying the number of axes (e.g., 3 for x/y/t).
   - `axisNames`: Vector of string labels for axes (e.g., (x", y", "t")).
   - `low` and `high`: Bounding vectors for the grid extents.
   - `CRS`: URI referencing a Coordinate Reference System (e.g., to a GML document).

- `origin`: Vector of coordinates for the grid's starting point.

Subclasses:

- `RectifiedGridDomain`: Adds `offset` (vector for spacing in each direction).
- `ReferenceableGridDomain`: Adds `coordinateGrid` (array of explicit coordinates for each grid point, preserving topology).

Semantics: The domain ensures cells are locatable in real-world coordinates, supporting skewed or warped grids.

3. **RangeType** Describes the range set's structure, modelled as a record from Sensor Web Enablement (SWE) standards [191]. **Properties:**

   - `hasField`: Links to one or more `Field` instances.

   Semantics: Defines the attribute types for cell values, supporting multi-band rasters (e.g., 7-band Landsat imagery).

4. **Field** represents individual components of the range (e.g., a spectral band).

   Properties:

   - `name`: Human-readable label (e.g., nir").
   - `description`: Textual explanation (e.g., Near infra-red part of the spectrum").
   - `definition`: URI to an ontology term (e.g., OGC radiance property).
   - `hasRange`: Links to a `Range` subclass constraining values.
   - `hasNILValue`: Links to `NILValue` for handling missing data.
   - `unitOfMeasure`: Specified as a string according to the Unified Code for Units of Measure (UCUM[16]) or by the *Quantities, Units, Dimensions, and Types* (QUDT) ontology [189] or an external UoM definition via URI.
   - `isOptional` and `isUpdatable`: Booleans for flexibility and mutability.

   Semantics: Fields allow for heterogeneous ranges, with array storage for actual data.

5. **Range** (and Subclasses) Constrains allowable values for a field. **Subclasses:**

   - `IntervalRange`: With `low` and `high` (e.g., 0–255 for grayscale).

[16]https://ucum.org/

- `EnumeratedRange`: With `values` (explicit list).
- `TypedRange`: With `dataType` (e.g., `xsd:integer` or custom).

Semantics: Ensures data integrity and type safety in queries.

6. **NILValue** Handles null or invalid values. **Properties:**

   - `reason`: URI to a human-readable explanation (e.g., "BelowDetectionRange").
   - `value`: Single value or range denoting nil.

   Semantics: Supports standards-compliant missing data representation.

4.5 Query Translation Layer

The core of the architecture (as illustrated in Figure 4.1) is the *Query Transformation Layer*, which processes the ONTOP-generated SQL–SQL/MDA queries and delegates their execution to the respective RDBMS and array DBMS. The PostGIS functions embedded in the SQL part can be executed directly by PostgreSQL via its PostGIS extension. We rely on PL/Python and PL/pgSQL stored procedures to establish a connection between ONTOP and RasDaMan, federated via PostgreSQL via rasdapy, to enable smooth retrieval of metadata for the relevant raster data and to enable the execution of suitable RaSQL queries directly by RasDaMan.

Table 4.2: PL/Python stored procedures

Stored procedure	Input arguments	Output
`geo2grid_coords()`	`GEOMETRY regionWkt, DOUBLE minLon, DOUBLE minLat, DOUBLE resLon, DOUBLE resLat`	`GEOMETRY regionGrid`
`query2numeric()`	`STRING rasqlQuery`	`DOUBLE value`
`query2array()`	`STRING rasqlQuery`	`DOUBLE[] array`

4.5.1 PL/Python Stored Procedures

The stored procedures are specified in the PL/Python procedural language[17], which also supports all PostgreSQL and PostGIS functions, and *rasdapy* to connect to RasDaMan. These procedures are stored in the `rasdaman_op` schema of the VectorTablesDB database in PostgreSQL. We elaborate now on the stored procedures shown in Table 4.2.

- `geo2grid_coords()`: Being a domain-agnostic array DBMS, RasDaMan (RaSQL precisely) only supports array indices (`i` and `j`) or grid coordinates (`gridX` and `gridY`) and does not consider any domain-specific coordinates, such as geo-coordinates (i.e., longitude and latitude) natively. Therefore, we devised this mapping function, inspired by a generic affine transformation

[17] https://www.postgresql.org/docs/current/plpython.html

model [226], that translates geo-coordinates to corresponding grid coordinates, accounting for the respective geographic coordinate reference system (CRS). OntoRaster assumes that both vector and raster data use the OGC default CRS[18] i.e., WGS84 longitude-latitude[19]). In practice, this function takes five input arguments: `regionWkt` and `minLon`, `minLat`, `resLon`, and `resLat` of the selected raster data. It returns a translated grid geometry of the chosen region (`regionGrid`). This enables the user to send the geometry (polygon or multi-polygon) of any region within the AOI to RasDaMan as a part of the RaSQL query. The embedded geometry will be translated into grid coordinates for use as input to RaSQL's array operations to extract raster data. For example, RaSQL's 'clip' operation can crop raster data based on the translated geometry of a user-defined region. `geo2grid_coords()` is also extended to support complex geometries, such as polygons with holes (e.g., a lake within a land) and multipolygons (e.g., islands) [86]. When dealing with polygons containing holes, both the exterior ring and all interior rings (or holes) are processed separately, and then merged in grid space. For multi-polygons, this function employs a sophisticated approach that processes each constituent polygon individually and transforms its coordinates into grid coordinates. The function then attempts to merge these transformed polygons using Shapely's `unary_union` operation. Shapely[20] is a Python library for manipulating and analysing planar geometric objects using the widely distributed open-source geometry library GEOS [85] (the engine of PostGIS), which conforms to OGC's simple feature access specification [109]. Suppose the merging operation fails for any reason, such as tiny constituent polygons (e.g., a tiny island). In that case, the function falls back to creating a new multi-polygon by combining the remaining transformed polygons. Error handling and edge cases are also implemented in `geo2grid_coords()`, including checks for invalid geometries (e.g., those with zero area) or invalid WKT representations, and specific handling for unsupported geometry types. It also mitigates potential failures in the transformation process and polygon-merging operations, ensuring robust operation even with complex input geometries.

- `query2numeric()`: Takes a RaSQL query as a string input and executes it over raster data using RasDaMan's supported array condenser operations[21], such as `avg_cells`, `max_cells`, etc. and retrieves aggregated numeric results back to PostgreSQL.

- `query2array()`: Evaluates RaSQL queries over raster arrays using RasDaMan-supported array operations[22] such as clip, concatenation, scaling, and retrieve filtered arrays back to PostgreSQL.

- `query2string()`: Evaluate simple rasql queries over raster arrays using RasDaMan-supported simple array operations such as `sdom()` to retrieve the

[18] https://docs.ogc.org/is/18-058r1/18-058r1.html#crs-discovery
[19] https://epsg.io/4326
[20] https://shapely.readthedocs.io/en/stable/index.html
[21] https://doc.rasdaman.org/04_ql-guide.html#condensers
[22] https://doc.rasdaman.org/04_ql-guide.html#array-operations

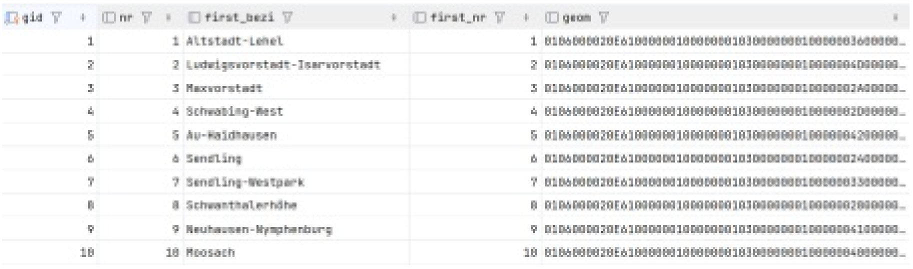
spatial dimension of raster data as a string.

4.5.2 PL/pgSQL Stored Procedures

• `timestamp2grid()`: RasDaMan treats timestamps as integers rather than the true "Date-Time" format. So, we introduced this PL/pgSQL[23] function that converts a timestamp from the "DateTime" format to an integer format (such as gridTime) that RasDaMan can understand.

4.6 SQL Translation

In VKGs, query answering is typically achieved via query reformulation. This means translating SPARQL queries over the VKG into equivalent SQL queries over the underlying relational data sources. In our case, however, reformulation must go beyond standard SQL translation: it must also include generating calls combined with raster functions to the RasDaMan. Ontop operates by taking an ontology that represents the domain knowledge, along with mappings that link the underlying database tables to the ontology. Designing mappings is a crucial user-driven process for generating VKGs. Individual RDB2RDF mappings exploit attributes from PostgreSQL to populate the RDF knowledge graph. Therefore, it is necessary to formulate SQL queries that map individual data attributes to their corresponding ontological concepts.



gid	nr	first_bezi	first_nr	geom
1	1	Altstadt-Lehel	1	0106000020E61000000100000001030000000100000036000000…
2	2	Ludwigsvorstadt-Isarvorstadt	2	0106000020E6100000010000000103000000010000004D000000…
3	3	Maxvorstadt	3	0106000020E6100000010000000103000000010000002A000000…
4	4	Schwabing-West	4	0106000020E6100000010000000103000000010000002D000000…
5	5	Au-Haidhausen	5	0106000020E61000000100000001030000000100000042000000…
6	6	Sendling	6	0106000020E61000000100000001030000000100000024000000…
7	7	Sendling-Westpark	7	0106000020E61000000100000001030000000100000033000000…
8	8	Schwanthalerhöhe	8	0106000020E61000000100000001030000000100000028000000…
9	9	Neuhausen-Nymphenburg	9	0106000020E61000000100000001030000000100000041000000…
10	10	Moosach	10	0106000020E61000000100000001030000000100000040000000…

Figure 4.4: Relational table `public.munich_dist25` contains 25 Districts of Munich stored in VectorTablesDB

As a running example with Munich, Figure 4.4 shows an excerpt from a table `public.munich_dist25`, containing vector data for all districts of Munich, including their attributes such as `gid`, which serves as a unique identifier for each district; `first_bezi`, which contains the district's name; and `geom`, which stores the geometry in hex-encoded WKB format. The PostGIS function `ST_AsText` converts geometries from WKB to the widely adopted WKT format, which is better suited for use in RDF representations.

[23] https://www.postgresql.org/docs/current/plpgsql-overview.html

Several mappings have been established to connect the data attributes of the `:Region` and `:Raster` classes. Note that the class `:Region` can designate one or more arbitrary places within an area of interest, depending on the real-world scenario or use case. Our running example, Munich, has two types of regions: districts and sub-districts. To semantically designate districts and sub-districts separately, we have introduced two subclasses of class `:Region`, i.e., `:Region_District_Munich` and `:Region_SubDistrict_Munich` in the ontology level, along with the data properties `geo:asWKT` and `rdfs:label`. More mappings for the rest of the AOIs are mentioned in section 5.2.3.

The listing below shows three example mappings for the districts of Munich. In the first mapping, class `:Region_District_Munich` is mapped to gid, the primary key of the district table. The second and third mappings extend the remaining data properties, such as name (i.e., first_bezi) and geom in WKT serialisation (i.e., `regionWkt`), to all districts of Munich. Based on these mappings, the vector geometries which are required for query answering are retrieved at query execution time using the class `:Region_District_Munich`. The source query (SQL) of the mapping contains a `CASE` statement that determines whether the input region's geometry is a polygon or a multipolygon, as these are handled differently by the `geo2grid_coords` function. This is a necessary step to transform any single polygon represented as a multi-polygon (which is technically possible but an inadequate representation) into its actual polygonal form, thereby ensuring OGC compliance.

- Mappings for 25 Districts of Munich

```
mappingId Munich_Districts_id
target :vector/bavaria/munich/districts/{regionId} a :Region_District_Munich .
source SELECT gid AS regionId FROM public.munich_dist25

mappingId Munich_Districts_name
target :vector/bavaria/munich/districts/{regionName} rdfs:label
      {regionName}^^xsd:string .
source SELECT gid AS regionId, first_bezi AS regionName FROM public.munich_dist25

mappingId Munich_Districts_regionWkt
target :vector/bavaria/munich/districts/{regionWkt} geo:asWKT
      {regionWkt}^^geo:wktLiteral .
source SELECT gid AS regionId,
                         CASE
                             WHEN ST_NumGeometries(geom) = 1 THEN
                                    ST_AsText(ST_GeometryN(geom, 1))
                             ELSE ST_AsText(geom)
                         END AS regionWkt
                 FROM public.munich_dist25
```

To explain our solution, we rely on the following example RasSPARQL query $\boldsymbol{Q}_{ex1}$, enquiring for those districts of Munich whose spatial average digital elevation is above 515 meters at a specific timestamp (11th February, 2000):

- $\boldsymbol{Q}_{ex1}$: Find all districts of Munich where the spatial average elevation is more than 515 meters

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?distName ?elevation {
?region a :Region_District_Munich .
?region rdfs:label ?distName ; geo:asWKT ?distWkt .
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (CONTAINS(?rasterName, 'Elevation'))
BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
     elevation)
FILTER(?elevation > 515)
}
```

OntoRaster takes the aforementioned RasSPARQL as input and produces the following SQL translation, which is executed separately on the underlying relational database PostgreSQL and the array database RasDaMan, and outputs the respective districts and their spatially averaged elevation. The generated SQL translation may appear poor and contain only 67 lines, given that the example RasSPARQL query is simple. In practice, for a given complex RasSPARQL query, we observed that the auto-generated SQL can be very complex, with more than 200 lines, making it challenging for humans to formulate or comprehend without VKG. In a standard GIS workflow, the user must manually specify the study regions of an AOI as polygons and load the corresponding raster dataset. In contrast, our integrated approach, OntoRaster leverages the capabilities of underlying relational and array databases connected to respective ontologies via mappings to automatically retrieve relevant vector geometries and raster data, enabling users to express queries in terms of semantic entities (e.g.g, `:Region_District_Munich` for districts), rather than low-level spatial coordinates. This simple RasSPARQL query illustrates the benefit of using an ontology that operates at a higher level of abstraction, freeing the end user from knowing the underlying data structures (e.g., columns and rows) of the database, i.e., relational or array.

- Translated SQL of corresponding RasSPARQL query $\boldsymbol{Q}_{ex1}$

```
SELECT
 V1."first_bezi" AS "first_bezi1m2",
 RASDAMAN_OP.QUERY2NUMERIC (
  CONCAT(
   'select avg_cells(clip(c[',
   RASDAMAN_OP.TIMESTAMP2GRID ('2000-02-11T00:00:00+00:00',
        V2."raster_name"),
   ', 0:* , 0:*]*',
   RASDAMAN_OP.GET_SCALEFACTOR (V2."raster_name"),
   ',',
   RASDAMAN_OP.GEO2GRID_COORDS (
    CASE
```

```
      WHEN (ST_NUMGEOMETRIES (V1."geom") = 1) THEN ST_ASTEXT (ST_GEOMETRYN
           (V1."geom", 1))
      ELSE ST_ASTEXT (V1."geom")
     END,
     RASDAMAN_OP.GET_MIN_LONGITUDE (V2."raster_name"),
     RASDAMAN_OP.GET_MAX_LATITUDE (V2."raster_name"),
     RASDAMAN_OP.GET_RES_LON (V2."raster_name"),
     RASDAMAN_OP.GET_RES_LAT (V2."raster_name")
    ),
    ')) from ',
    V2."raster_name",
    ' as c'
   )
  ) AS "v0",
  CASE
   WHEN (ST_NUMGEOMETRIES (V1."geom") = 1) THEN ST_GEOMETRYN (V1."geom",
        1) IS NOT NULL
   ELSE V1."geom" IS NOT NULL
  END AS "v1"
 FROM
  "munich_dist25" V1,
  "raster_lookup" V2
 WHERE
  (
   CASE
    WHEN (ST_NUMGEOMETRIES (V1."geom") = 1) THEN ST_GEOMETRYN (V1."geom",
         1) IS NOT NULL
    ELSE V1."geom" IS NOT NULL
   END
   AND (
    CAST(
     RASDAMAN_OP.QUERY2NUMERIC (
      CONCAT(
       'select avg_cells(clip(c[',
       RASDAMAN_OP.TIMESTAMP2GRID ('2000-02-11T00:00:00+00:00',
            V2."raster_name"),
       ', 0:* , 0:*]*',
       RASDAMAN_OP.GET_SCALEFACTOR (V2."raster_name"),
       ',',
       RASDAMAN_OP.GEO2GRID_COORDS (
        CASE
         WHEN (ST_NUMGEOMETRIES (V1."geom") = 1) THEN ST_ASTEXT
              (ST_GEOMETRYN (V1."geom", 1))
         ELSE ST_ASTEXT (V1."geom")
        END,
        RASDAMAN_OP.GET_MIN_LONGITUDE (V2."raster_name"),
        RASDAMAN_OP.GET_MAX_LATITUDE (V2."raster_name"),
        RASDAMAN_OP.GET_RES_LON (V2."raster_name"),
        RASDAMAN_OP.GET_RES_LAT (V2."raster_name")
       ),
       ')) from ',
       V2."raster_name",
       ' as c'
      )
     ) AS DOUBLE PRECISION
    ) > 515::DOUBLE PRECISION
 )
 AND (POSITION('Elevation' IN V2."raster_name") > 0)
 AND V1."first_bezi" IS NOT NULL
 AND V2."raster_name" IS NOT NULL
)
```

Now, if the user wishes to query for the sub-districts of Munich with similar raster data, then they just need to change the class name, e.g., `:Region_SubDistrict_Munich` in place of `:Region_District_Munich` as shown in the second example RasSPARQL query $\boldsymbol{Q}_{ex2}$ below. The query retrieves all sub-districts of Munich from the underlying database according to the mapping below, and then links the corresponding ontology to the underlying data sources.

- Mappings for 105 Sub-Districts of Munich

```
mappingId Munich_Sub_Districts_id
target  :vector/bavaria/munich/subdistricts/{regionId} a :Region_SubDistrict_Munich .
source  SELECT gid AS regionId FROM public.munich_subdist105

mappingId Munich_Sub_Districts_name
target  :vector/bavaria/munich/subdistricts/{regionId} rdfs:label
     {regionName}^^xsd:string .
source  SELECT gid AS regionId, name AS regionName FROM public.munich_subdist105

mappingId Munich_Sub_Districts_regionWkt
target  :vector/bavaria/munich/subdistricts/{regionId} geo:asWKT
     {regionWkt}^^geo:wktLiteral .
source  SELECT gid AS regionId,
                 CASE
                     WHEN ST_NumGeometries(geom) = 1 THEN
                          ST_AsText(ST_GeometryN(geom, 1))
                     ELSE ST_AsText(geom)
                 END AS regionWkt
          FROM public.munich_subdist105
```

- $\boldsymbol{Q}_{ex2}$: Find all sub districts of Munich where the spatial average elevation is more than 515 meters

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?distName ?elevation {
?region a :Region_SubDistrict_Munich . # inplace of Region_District_Munich
?region rdfs:label ?distName ; geo:asWKT ?distWkt .
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (CONTAINS(?rasterName, 'Elevation'))
BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
     elevation)
FILTER(?elevation > 515)
}
```

4.7 Conformance to International Standards

In our methodology and its implementation, i.e., OntoRaster, we adhere to many international standards from the World Wide Web Consortium (W3C), Open Geospatial Consortium (OGC), and ISO, as listed in Table 4.3.

Sl.	Body	Name	Latest By	Url	Remark
1	W3C	OWL2	2004	link	Ontology language for knowledge graphs
2	W3C	SPARQL 1.1	2013	link	RDF query language
3	W3C	RDF 1.1	2014	link	Graph-based data model
4	W3C	RDFS 1.1	2014	link	RDF vocabulary schema
5	W3C	R2RML	2012	link	RDB to RDF mapping
6	OGC	WKT 2.1	2023	link	Vector geometry encoding format
7	OGC	CityGML 3.0	2021	link	3D city semantic model
8	OGC	CityJSON 2.0	2023	link	JSON encoding of CityGML
9	OGC	GeoSPARQL 1.1	2024	link	Spatial RDF query standard
10	OGC	CoverageJSON 1.0	2023	link	JSON encoding for coverages
11	OGC	GeoTIFF 1.1	2019	link	Georeferenced raster format
12	OGC	NetCDF 1.0	2011	link	Multidimensional array data format
13	ISO 6709	Geographic location	2022	link	Standard representation of geo-coordinates
14	ISO 8601	Date-Time	2019	link	International Date-Time format
15	ISO 19111	CRS	2019	link	Coordinate Reference System
16	ISO 19125-1	SFA Part 1	2011	link	Common Architecture
17	ISO 19125-2	SFA Part 2	2010	link	SQL Option
18	ISO 19123-1	Part 1 - Fundamentals	2023	link	Schema for coverage geometry and functions
19	ISO 19123-2	CIS Part 2	2018	link	Coverage Implementation Schema
20	ISO 9075-15:2023	Part 15 - SQL/MDA	2018	link	SQL for Multidimensional Arrays

Table 4.3: Global standards considered in OntoRaster

4.8 Conclusion

We have presented OntoRaster, a novel extended VKG framework that, for the first time, enables on-the-fly querying of multidimensional raster data combined with relational data, including vector data, by connecting the array DBMS RasDaMan to the VKG system Ontop. To achieve this, we defined the RasSPARQL language by extending SPARQL with custom raster functions to query over VKGs. We then developed a query transformation system that includes several stored procedures defined in PL/Python and PL/pgsql, enabling PostgreSQL to serve as a federator between the VKG engine Ontop and diverse data sources (e.g., raster and relational). It routes the translated SQL-SQL/MDA query from Ontop to the corresponding data sources in the RDBMS & array DBMS, as specified in the RasSPARQL query, and executes them separately. After execution, the retrieved sub-answers (which contain relational and raster arrays) are joined and returned to Ontop to generate a virtual knowledge graph that answers the user's semantic query. The most notable contribution of this chapter is its utilisation of the array-handling capabilities of an array DBMS, i.e., RasDaMan, to execute queries on large raster datasets in their original multidimensional array format, without conversion. This eliminates the need for costly raster data transfer into relational databases. OntoRaster supports a wide range of queries, limited only by the data-processing and functional capabilities of PostgreSQL (RDBMS) and RasDaMan (array DBMS).

Chapter 5

OntoRaster in Action

OntoRaster ensures efficiency, scalability, and minimal overhead during query answering over raster data, combined with relational data and linked data. The latest version of the OntoRaster framework is available as open-source on Github[1]. The chapter provides an overview of the OntoRaster workflow and several practical queries with results to demonstrate the application of OntoRaster. This chapter is based on the papers [87, 86].

5.1 Overall Work Flow

Essential components of traditional VKG systems [233] : *(i)* the conceptual layer formed by the ontology; *(ii)* the data layer supplied by the data sources; and *(iii)* the mapping layer encompassing the declarative specification that associates each (class and property) symbol in the ontology with a (SQL) view over (potentially federated) data. It is this mapping layer that separates the queried virtual instance from the physical data contained in the data sources.

Systems that rely on the VKG framework, i.e., OntoRaster, typically adapt the following workflow for query answering:

1. ***Initialisation***: The system initially loads the ontology, mappings, and data-source configurations, performing auxiliary preprocessing tasks such as schema alignment and optional reasoning or inference materialisation to prepare the semantic layer via mapping integration.

2. ***Query Input***: A user submits a query over the ontology vocabulary (typically SPARQL or SQL via a BI interface), expressed independently of the underlying data schemas.

3. ***Query rewriting phase***: The input query is transformed into a more complex query that incorporates the inferences derived from the intensional level of the ontology.

[1]https://github.com/aghoshpro/OntoRaster.git

4. ***Query Translation***: The rewritten query is translated into source-specific queries based on user-defined mapping assertions and executed on the corresponding systems (e.g., relational/vector queries in an RDBMS and raster queries delegated to specialised engines such as RasDaMan).

5. ***Query Execution***: The data query is executed on the original data source, generating answers based on the data source schema. These answers are then translated into the ontology vocabulary and RDF data types, yielding a response to the original query.

6. ***Assembly of Sub-Query Results***: Partial results returned from heterogeneous sources via sub queries. These results are integrated at the VKG layer and translated into ontology-compliant RDF triples.

7. ***Query Output***: Final results are returned to the user and may optionally be visualised through geospatial interfaces for integrated exploration of vector and raster data.

5.2 VKG specification

5.2.1 Ontology ($\mathcal{O}$)

OntoRaster relies on the GeoSPARQL 1.1 ontology describing the semantics of vector geometries and on the *Raster Ontology* (prefix `rasdb:`), which represents N-dimensional generic raster data or coverage based on the OGC CIS v1.1 Standard. OntoRaster adopted the GeoNames v3.3 ontology (prefix `gn:`), which exposes the semantics of 12 million unique geographical point features with names including alternate and translated names, population, time-zone, geo coordinates, etc. OntoRaster relies on the LinkedGeoData ontology (prefix `lgdo:`), integrates OSM building data with different types of raster data. For CityGML 3D building data, OntoRaster adopted the most well-known CityGML ontology[2] (prefix `bldg:`), which is developed by the Knowledge Engineering @ CUI group at the University of Geneva. All the used prefixes are listed in Table 5.1, which refer to the base namespaces of the auxiliary vocabularies used in RasSPARQL queries within the OntoRaster framework.

5.2.2 Data Sources ($\mathcal{S}$)

Data sources depend on the study area of interest. We have selected four different areas of interest (AOIs) across the EU, as shown in Figure 5.1, with increasing area, geometric complexity, research group diversity, and data availability. These four AOIs are Munich (city), Bavaria (state), South Tyrol (province), and Sweden (country). OntoRaster is compatible with any AOIs that encompass datasets of similar types, but is not restricted to them.

[2] https://cui.unige.ch/isi/ke/ontologies

Table 5.1: List of prefixes used for RasSPARQL queries

Prefix	IRI Namespace
`:`	`https://github.com/aghoshpro/OntoRaster/`
`gn`	`https://www.geonames.org/ontology#`
`geo`	`http://www.opengis.net/ont/geosparql#`
`geof`	`http://www.opengis.net/def/function/geosparql/`
`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`
`lgdo`	`http://www.linkedgeodata.org/ontology/`
`bldg`	`http://www.opengis.net/citygml/building/2.0/`
`rasdb`	`https://github.com/aghoshpro/RasterDataCube/`

(a) Munich (311.20 km^2) (b) S.Tyrol (7,400 km^2) (c) Bavaria (70,550 km^2) (d) Sweden (450,295 km^2)

Figure 5.1: Four Different AOIs

In our work, we represent vector data using one or more relational tables for every AOI, stored in *VectorTablesDB* database within PostgreSQL. Every AOI table has exactly one geometry column, which is associated with a *spatial reference system* (SRS) that identifies the coordinate system for all geometric objects stored in the column and provides meaning for the numerical coordinate values of those objects. All of the relevant vector data for each AOI are obtained as shapefiles from respective sources and imported into separate tables within VectorTablesDB as illustrated in Fig. 5.2 as a suitable default SQL binary object format Well-Known Binary (WKB) (in hex format) by utilising the `shp2pgsql`[3] data loader.

Notice that, for better illustration, we have used convenient names for the tables and their columns (e.g., `regionId`, `regionName`, `regionWkb`, etc.) in VectorTablesDB. However, any other user-preferred name would have been acceptable, provided it was used consistently across the VKG mappings that link the respective ontologies to heterogeneous data sources (both vector and raster).

Raster data are stored within RasDaMan using ingredients and recipes. Table 5.2 showing *Raster Lookup*, a singular table that maintains the necessary

[3]https://postgis.net/docs/using_postgis_dbmanagement.html#shp2pgsql_usage

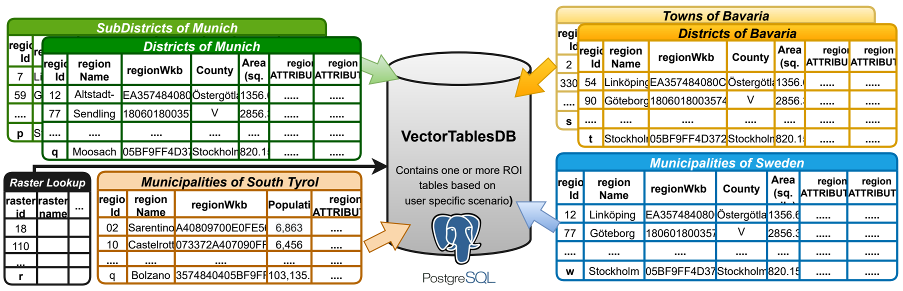


Figure 5.2: VectorTablesDB containing tables for AOIs and Raster Lookup

metadata of the raster datasets and is stored within VectorTablesDB. The entire lookup table is available at Github[4].

Table 5.2: Raster Lookup

raster_id	raster_name	field_id	field_name	fill_nan	scale_factor	min_lon	max_lon	...
18	Sweden_Surface_Temperature	36	LST_Night_1km	0	0.02	10.958333332	24.17499999	...
41	Bavaria_Temperature_MODIS_1km	59	LST_Night_1km	0	0.02	8.974999999	13.841666665	...
64	South_Tyrol_Temperature_MODIS_1km	82	LST_Night_1km	0	0.02	10.38333333	12.48333333	...
87	Munich_SRTM_Elevation_30m	105	SRTMGL1_DEM	-32768	1	11.360694444	11.7231944444	
110	Munich_MODIS_Temperature_1km	128	LST_Night_1km	0	0.02	11.35833333	11.72499999	...
133	Munich_MODIS_NDVI_250m	151	NDVI	-3000	0.0001	11.3604166656	11.724999998	...
156	Munich_MODIS_SnowCover_500m	174	SnowCover	200	1	11.358333332	11.724999998	...
179	Munich_ECOSTRESS_SoilMoisture_70m	197	SoilMoisture	nan	1	11.36055525728	11.72328651	...

We now discuss each dataset for each AOI considered in this thesis. Starting with Munich, as it has the widest variety of available datasets, as shown in Figure 5.3.

5.2.2.1 Munich

Munich, the capital of Bavaria, Germany, and its surrounding functional urban area (approx. 311.20 km^2), i.e., densely populated city centres integrated with the surrounding labour market (commuting zone through high travel-to-work flow, as defined by EU-OECD [66]). Figure 5.3 displays different types of geospatial data about Munich (each small box in the upper-right corner provides a zoomed-in view of the data), and their information is shown in Table 5.3. We are using 25 districts (Figure 5.3a) and 105 sub-districts (Figure 5.3b) of Munich as our primary vector data. Additionally, *OpenStreetMap* (OSM) data from Geofabrik Server[5] for Oberbayern and clipped based on the spatial extent of Munich. CityGML LoD2 building data (in `.gml`) format is obtained from the Bavarian Surveying Administration Open Data Portal[6] as shown in Figure 5.3c. Finally, five distinct types of raster data with varying fields and resolutions

[4] https://github.com/aghoshpro/OntoRaster/tree/main?tab=readme-ov-file#52-raster-data-darr

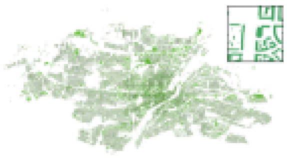
[5] https://download.geofabrik.de/europe/germany/bayern/oberbayern.html

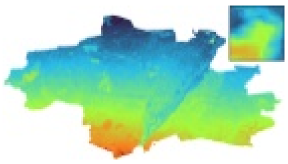
[6] https://geodaten.bayern.de/opengeodata/

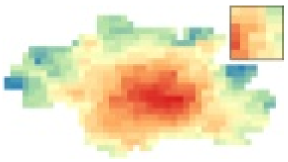
(a) 25 Districts

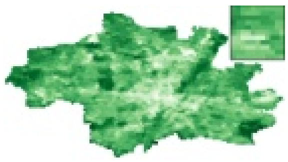
(b) 105 Subdistricts

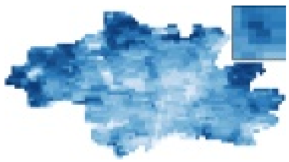
(c) CityGML 3D!Buildings


(d) OSM 2D Buildings


(e) SRTM Elevation


(f) MODIS Temperature


(g) MODIS NDVI


(h) NDSI Snow Cover

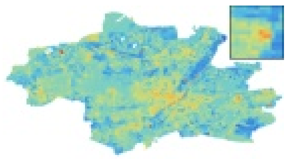

(i) ECO Soil Moisture

Figure 5.3: Heterogeneous geospatial data over Munich

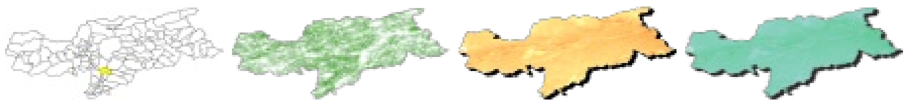
(both spatial and temporal) from NASA and ESA, are considered. These include the SRTM digital elevation data [78] (Figure. 5.3e), MODIS Temperature data [225] (Figure 5.3f), MODIS Vegetation (NDVI) [65] (Figure 5.3g), MODIS Snow Cover (NDSI) data [99] (Figure 5.3h) and ECOSTRESS Soil Moisture data [113] (Figure 5.3i).

Table 5.3: Munich's Data Information

	Data Type	**Data Field**	**Source and Sensors**	**Feature Unit**	**Feature Count**	**Spatial Resolution**	**Temporal Resolution**
(a)	Relational	25 Districts	-	point	4820	-	-
(b)	Relational	105 Sub-districts	-	point	6692	-	-
(c)	Relational	184K+ 2D Bldg Footprints	OSM	point	3285853	-	-
(d)	Relational	184K+ 3D Buildings	CityGML	point	5513088	-	-
(e)	Raster	Elevation	SRTM	pixel	873010	30m x 30m	-
(f)	Raster	Land Temperature	MODIS	pixel	1012	1km x 1km	daily
(g)	Raster	NDVI Vegetation	MODIS	pixel	15925	250m x 250m	16 days
(h)	Raster	Snow Cover	NDSI	pixel	4048	500m x 500m	daily
(i)	Raster	Soil Moisture	ECOSTRESS	pixel	3513088	70m x 70m	seconds

5.2.2.2 South Tyrol (province)

Figure 5.4 displays all the geospatial data which are considered for South Tyrol, Italy. We consider 116 municipalities (Figure 5.4a) in South Tyrol as primary vector data, with the capital, Bolzano, shown in yellow. For more complex vector data, we consider the 2D footprint of OSM buildings data. GeoNames point feature data is also considered. MODIS Temperature data [225](Figure 5.4c) and SRTM digital elevation data [78](Figure 5.4d) are selected as raster data.



(a) 116 Municipals (b) OSM Buildings (c) MODIS Temp (d) SRTM Elevation

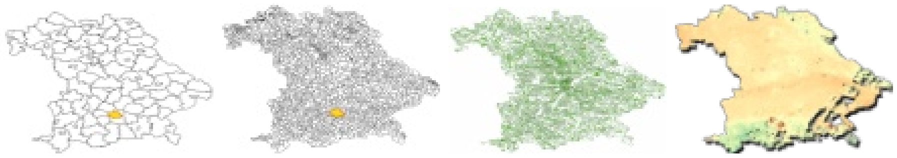
Figure 5.4: Heterogeneous geospatial data over South Tyrol

Table 5.4: Tyrol's Data Information

	Data Type	**Data Field**	**Source** and **Sensors**	**Feature Unit**	**Feature Count**	**Spatial Resolution**	**Temporal Resolution**
(a)	Relational	116 Municipalities	GADM	point	116	-	-
(b)	Relational	2D Bldg Footprints	OSM	point	3285853	-	-
(c)	Raster	Elevation	SRTM	pixel	873010	30m x 30m	-
(d)	Raster	Land Temperature	MODIS	pixel	1012	1km x 1km	daily

5.2.2.3 Bavaria (state)

Figure 5.5 displays all the geospatial data of Bavaria State, Germany, which are considered here. As primary vector data, we selected 96 districts (Figure 5.5a) and 2238 towns (Figure 5.5b) in Bavaria, with the capital, Munich, shown in yellow. For more complex vector data, we consider the 2D footprint of OSM buildings data. GeoNames point feature data are also considered. MODIS Temperature data [225] is chosen for raster data (Figure 5.5d) .



(a) 96 Districts (b) 2238 Towns (c) OSM Buildings (d) MODIS Temp

Figure 5.5: Heterogeneous geospatial data over Bavaria

5.2.2.4 Sweden (country)

Figure 5.6 displays all the geospatial data of Sweden. As the primary vector dataset, we selected 290 municipalities (*kommuns*) (Figure 5.6a) and 21 counties (Figure 5.6b) in Sweden, with the capital, Stockholm, shown in yellow. GeoNames point feature data is also considered. For raster data we

Table 5.5: Bavaria's Data Information

	Data Type	**Data Field**	**Source and Sensors**	**Feature Unit**	**Feature Count**	**Spatial Resolution**	**Temporal Resolution**
(a)	Relational	96 Districts	GADM	point	96	-	-
(b)	Relational	2238 Towns	GADM	point	2238	-	-
(c)	Relational	2D Bldg Footprints	OSM	point	3285853	-	-
(d)	Raster	Land Temperature	MODIS	pixel	1012	1km x 1km	daily

chose MODIS Temperature data [225](Figure 5.6c) and SRTM digital elevation data [78](Figure 5.6d).

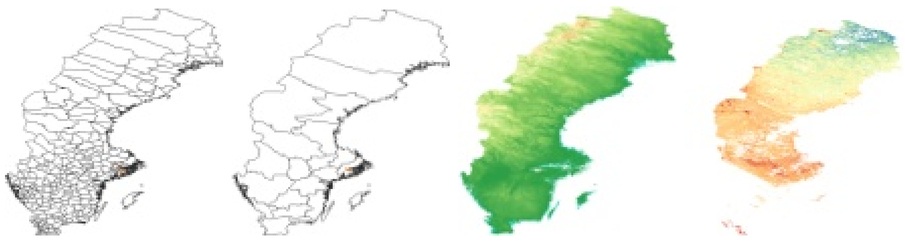

(a) 290 Municipals (b) 21 Counties (c) SRTM Elevation (d) MODIS Temp

Figure 5.6: Heterogeneous geospatial data over Sweden

Table 5.6: Sweden's Data Information

	Data Type	**Data Field**	**Source and Sensors**	**Feature Unit**	**Feature Count**	**Spatial Resolution**	**Temporal Resolution**
(a)	Relational	290 Municipalities	GADM	point	4820	-	-
(b)	Relational	21 Counties	GADM	point	6692	-	-
(c)	Relational	2D Bldg Footprints	OSM	point	3285853	-	-
(e)	Raster	Elevation	SRTM	pixel	873010	30m x 30m	-
(f)	Raster	Land Temperature	MODIS	pixel	1012	1km x 1km	daily
(h)	Raster	Snow Cover	NDSI	pixel	4048	500m x 500m	daily

5.2.3 Mappings ($\mathcal{M}$)

Designing mappings is a crucial user-driven process in the construction of Virtual Knowledge Graphs (VKGs). As previously mentioned in the section 4.6, VKG mappings establish declarative correspondences between heterogeneous data sources and domain ontologies, enabling the generation of a virtual RDF knowledge graph at query time. In practice, mappings exploit relational database attributes (e.g., from PostgreSQL) by defining SQL queries that map source data attributes to their corresponding ontological classes and properties, thereby enabling SPARQL queries to be translated into executable database queries. Thus, for different data sources, suitable mappings must be designed to connect them to the corresponding ontologies. In this dissertation, we have several AOIs, each with different sets of regions: Bavaria has towns and districts, South Tyrol has only provinces, and Sweden has municipalities and states. In general, for an arbitrary AOI and its regions, mappings must be designed. For each of the aforementioned AOIs considered in this work, we have outlined mappings to the corresponding subclasses, `:Region_Bavaria`, `:Region_Tyrol`, and `:Region_Sweden`, which also exist in the corresponding ontologies. The reader is referred to the GitHub repository[7] for more details.

- Mappings for 116 Municipalities of South Tyrol, Italy

```
mappingId Get_region_class_table_south_tyrol
target  :vector/tyrol/{regionId} a :Region_Tyrol .
source  SELECT gid AS regionId FROM public.region_south_tyrol

mappingId mapping_regionName_south_tyrol
target  :vector/tyrol/{regionId} rdfs:label {regionName}^^xsd:string .
source  SELECT gid AS regionId, name_3 AS regionName FROM public.region_south_tyrol

mappingId mapping_region_regionWkt_south_tyrol
target  :vector/tyrol/{regionId} geo:asWKT {regionWkt}^^geo:wktLiteral .
source  SELECT gid AS regionId,
                 CASE
                     WHEN ST_NumGeometries(geom) = 1 THEN
                          ST_AsText(ST_GeometryN(geom, 1))
                     ELSE ST_AsText(geom)
                 END AS regionWkt
           FROM public.region_south_tyrol
```

- Mappings for 96 Districts of Bavaria, Germany

```
mappingId Get_region_class_table_bavaria
target  :vector/bavaria/{regionId} a :Region_Bavaria .
source  SELECT gid AS regionId FROM public.region_bavaria

mappingId mapping_regionName_bavaria
target  :vector/bavaria/{regionId} rdfs:label {regionName}^^xsd:string .
source  SELECT gid AS regionId, name_2 AS regionName FROM public.region_bavaria

mappingId mapping_region_regionWkt_bavaria
target  :vector/bavaria/{regionId} geo:asWKT {regionWkt}^^geo:wktLiteral .
source  SELECT gid AS regionId,
                 CASE
                     WHEN ST_NumGeometries(geom) = 1 THEN
                          ST_AsText(ST_GeometryN(geom, 1))
                     ELSE ST_AsText(geom)
                 END AS regionWkt
           FROM public.region_bavaria
```

- Mappings for 290 Municipalities (kommuns) of Sweden

```
mappingId Get_region_class_table_sweden
target  :vector/sweden/{regionId} a :Region_Sweden .
source  SELECT gid AS regionId FROM public.region_sweden

mappingId mapping_regionName_sweden
target  :vector/sweden/{regionId} rdfs:label {regionName}^^xsd:string .
source  SELECT gid AS regionId, name_2 AS regionName FROM public.region_sweden

mappingId mapping_region_regionWkt_sweden
target  :vector/sweden/{regionId} geo:asWKT {regionWkt}^^geo:wktLiteral .
```

[7] https://github.com/aghoshpro/OntoRaster.git

```
source  SELECT gid AS regionId,
               CASE
                   WHEN ST_NumGeometries(geom) = 1 THEN ST_AsText(ST_GeometryN(geom,
                        1))
                   ELSE ST_AsText(geom)
               END AS regionWkt
            FROM public.region_sweden
```

5.3 Semantic Query Answering and Results

Here we provide three types of RasSPARQL queries: simple, spatial (aggregation and filtering), and spatial-temporal (aggregation and filtering) to demonstrate query answering over virtually integrated vector and raster data, and the corresponding results obtained from OntoRaster. A combination of them, along with many query types, is feasible and is limited only by RasDaMan's ability to handle multidimensional arrays. We will list different types of queries, ranging from simple to complex, encompassing various raster and vector data. All other possible queries can be found in Appendix A.

$\boldsymbol{Q}_{Simple}$ is a simple raster query that retrieves dimension information of the input raster data. Query $\boldsymbol{Q}_{CellOp}$ conducts element-wise operations on raster array cells.

- $\boldsymbol{Q}_{Simple}$: Find all current raster datasets and their dimension

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?rasterName ?dimension {
    ?gridCoverage a :Raster .
    ?gridCoverage rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, ''))
    BIND (rasdb:rasDimension(?rasterName) AS ?dimension)
}
```

- $\boldsymbol{Q}_{CellOp}$: Perform element-wise operation over a raster array cells with user-specific operator and operand on a specific timestamp and return the array

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?array {
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Sweden_Surface_Temperature'))
    BIND ('*' AS ?operator)
    BIND (0.02 AS ?operand)
    BIND ('2022-04-18T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasCellOp(?timeStamp, ?operator, ?operand, ?rasterName) AS
          ?array)
}
```

Spatial (Q_S). RasSPARQL query Q_S targets districts of Munich, Germany, to identify those exceeding a spatial average elevation of 515 meters—a threshold potentially indicative of topographic influences on microclimates, urban planning, or ecological zoning.

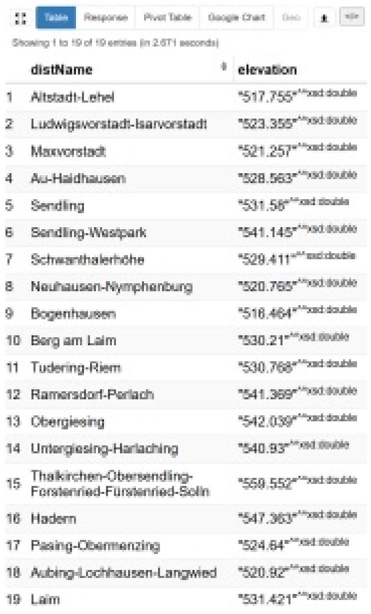
- Q_S: Find all districts of Munich where the spatial average elevation is more than 515 meters.

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?distName ?elevation {
   ?region a :Region_District_Munich .
   ?region rdfs:label ?distName .
   ?region geo:asWKT ?distWkt .
   ?gridCoverage a :Raster .
   ?gridCoverage rasdb:rasterName ?rasterName .
   FILTER (CONTAINS(?rasterName, 'Elevation'))
   BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
   BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
       elevation)
   FILTER(?elevation > 515)
}
```

Leveraging ontologies such as GeoNames (`gn`), GeoSPARQL (`geo`), and RDF Schema (`rdfs`), alongside custom namespaces for RasterDataCube (`rasdb`), the query constructs a multifaceted retrieval. It begins by defining regional entities (`?region`) as labelled districts (`?distName`) with WKT geometries (`?distWkt`). These are intersected with raster coverages (`?gridCoverage`) linked with `?rasterName` to filter out ”Elevation” raster among other rasters and are bound by a fixed timestamp (`2000-02-11T00:00:00+00:00`). The core computation employs the *rasSpatialAverage* function from the Raster ontology to compute mean elevations (`?elevation`) across district geometries, applying a strict filter (>515) to retain only qualifying entries, i.e., the nineteen districts. Figure 5.7 reveals the nineteen resultant districts with their respective spatial average elevation based on the query constraint.

- Q_{S-viz}: Find and visualise all districts of Munich where the spatial average elevation is more than 515 meters.

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?distName ?elevation ?distWkt {
   ?region a :Region_District_Munich .
   ?region rdfs:label ?distName .
   ?region geo:asWKT ?distWkt .
```



Showing 1 to 19 of 19 entries (in 2.071 seconds)

	distName	elevation
1	Altstadt-Lehel	"517.755"^^xsd:double
2	Ludwigsvorstadt-Isarvorstadt	"523.355"^^xsd:double
3	Maxvorstadt	"521.257"^^xsd:double
4	Au-Haidhausen	"528.563"^^xsd:double
5	Sendling	"531.58"^^xsd:double
6	Sendling-Westpark	"541.145"^^xsd:double
7	Schwanthalerhöhe	"529.411"^^xsd:double
8	Neuhausen-Nymphenburg	"520.765"^^xsd:double
9	Bogenhausen	"516.464"^^xsd:double
10	Berg am Laim	"530.21"^^xsd:double
11	Tudering-Riem	"530.768"^^xsd:double
12	Ramersdorf-Perlach	"541.369"^^xsd:double
13	Obergiesing	"542.039"^^xsd:double
14	Untergiesing-Harlaching	"540.93"^^xsd:double
15	Thalkirchen-Obersendling-Forstenried-Fürstenried-Solln	"559.552"^^xsd:double
16	Hadern	"547.363"^^xsd:double
17	Pasing-Obermenzing	"524.64"^^xsd:double
18	Aubing-Lochhausen-Langwied	"520.92"^^xsd:double
19	Laim	"531.421"^^xsd:double

Figure 5.7: Resultant 19 districts of Munich whose elevation > 515 meter

```
    ?gridCoverage a :Raster .
    ?gridCoverage rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Elevation'))
    BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
        elevation)
    FILTER(?elevation > 515)
}
```

This RasSPARQL $\boldsymbol{Q}_{S-viz}$ extends the prior query $\boldsymbol{Q}_S$ by incorporating district geometries (`?distWkt`) into the `SELECT` clause, enabling cartographic rendering of WKT polygons of resultant 19 districts of Munich over OpenStreetMap as depicted in Figure 5.8. This result vividly illustrates urban elevational disparities of Munich's topography. In a broader sense, this integration of query results with visualisation empowers dynamic geospatial analysis, aiding urban ecology and planning research by revealing patterns in terrain-driven vulnerabilities. Query formulation also leverages ontologies such as GeoNames (`gn`), GeoSPARQL (`geo`), and RDF Schema (`rdfs`), as well as (`rasdb`), a custom namespace for raster data.

Spatial-Temporal ($\boldsymbol{Q}_{ST}$). The presented RasSPARQL query $\boldsymbol{Q}_{ST}$ targets districts of Munich, Germany, to identify those with an average elevation

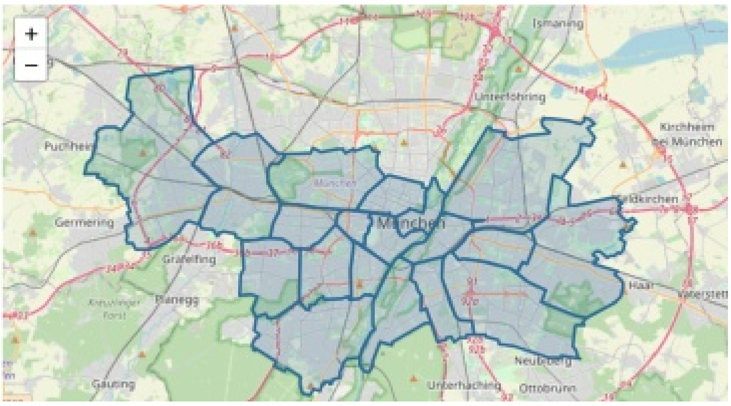


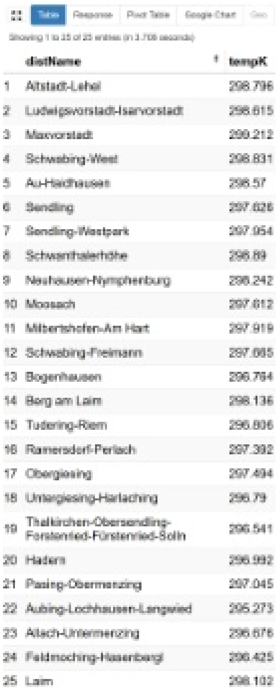
Figure 5.8: Visualizing resultant 19 districts of Munich whose elevation > 515m

exceeding 515 meters—a threshold potentially indicative of topographic influences on microclimates, urban planning, or ecological zoning.

- $\boldsymbol{Q_{ST}}$: Find maximum temperature for all districts of Munich over the time frame between 01st January, 2022 and 01st January, 2024.

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?distName ?tempK {
    ?region a :Region_District_Munich ; rdfs:label ?distName ;
                                        geo:asWKT ?distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
    BIND ('2024-01-01T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
    BIND (rasdb:rasTemporalMaximum(?startTimeStamp, ?endTimeStamp, ?
         distWkt, ?rasterName) AS ?tempK)
}
```

This query formulation begins by defining regional entities (`?region`) as labelled districts (`?distName`) with WKT geometries (*?distWkt*). These are intersected with raster coverages (`?gridCoverage`) named for temperature data (`?rasterName`), filtered to contain "Temperature" and bound to a fixed timeframe defined by `?startTimeStamp` (`2022-01-01T00:00:00+00:00`) and `?endTimeStamp` (`2024-01-01T00:00:00+00:00`). The core computation employs the `rasTemporalMaximum` function derive the maximum temperature (`?elevation`) across district geometries. This query yields only qualifying entries, i.e., the resultant twenty-five districts as depicted in the Figure 5.9 based on the query constraint.



Showing 1 to 25 of 25 entries (in 3.708 seconds)

	distName	tempK
1	Altstadt-Lehel	298.796
2	Ludwigsvorstadt-Isarvorstadt	298.615
3	Maxvorstadt	299.212
4	Schwabing-West	298.831
5	Au-Haidhausen	298.57
6	Sendling	297.626
7	Sendling-Westpark	297.954
8	Schwanthalerhöhe	298.69
9	Neuhausen-Nymphenburg	298.242
10	Moosach	297.612
11	Milbertshofen-Am Hart	297.919
12	Schwabing-Freimann	297.665
13	Bogenhausen	296.764
14	Berg am Laim	298.136
15	Trudering-Riem	296.806
16	Ramersdorf-Perlach	297.392
17	Obergiesing	297.494
18	Untergiesing-Harlaching	296.79
19	Thalkirchen-Obersendling-Forstenried-Fürstenried-Solln	296.541
20	Hadern	296.992
21	Pasing-Obermenzing	297.045
22	Aubing-Lochhausen-Langwied	295.273
23	Allach-Untermenzing	296.676
24	Feldmoching-Hasenbergl	296.425
25	Laim	298.102

Figure 5.9: Resultant 25 districts of Munich

- $\boldsymbol{Q}_{ST-viz}$: Find and visualise maximum temperature for all districts of Munich over the time frame between 01st January, 2022 and 01st January, 2024.

```
SELECT ?distName ?tempK ?distWkt {
    ?region a :Region_District_Munich ; rdfs:label ?distName ; geo:asWKT ?
        distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
    BIND ('2024-01-01T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
    BIND (rasdb:rasTemporalMaximum(?startTimeStamp, ?endTimeStamp, ?
        distWkt, ?rasterName) AS ?tempK)
}
```

This RasSPARQL $\boldsymbol{Q}_{ST-viz}$ extends the prior query $\boldsymbol{Q}_{ST}$ by incorporating district geometries (`?distWkt`) into the `SELECT` clause, enabling cartographic rendering of WKT polygons of resultant 25 districts of Munich over OpenStreetMap as depicted in Figure 5.10.

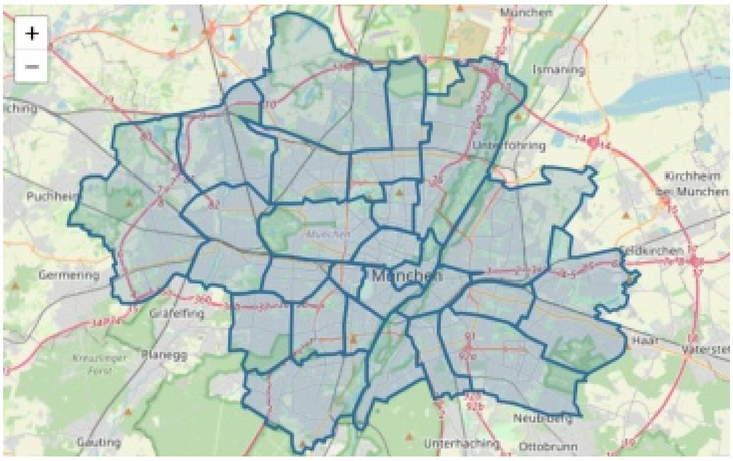


Figure 5.10: Visualising the resultant 25 districts of Munich

5.3.1 Region Geometries + Raster Data

In this section, we will list additional RasSPARQL queries (without prefixes) that integrate raster data with simple vector data.

- $\boldsymbol{Q}_{S1}$: Find all sub-districts of Munich whose spatial average elevation is above 515 meters

```
SELECT ?distName ?elevation ?distWkt ?distWktColor{
    ?region a :Region_SubDistrict_Munich .
    ?region rdfs:label ?distName .
    ?region geo:asWKT ?distWkt .
    ?gridCoverage a :Raster .
    ?gridCoverage rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Elevation'))
    BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp) .
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
        elevation) .
    FILTER(?elevation > 515) .
      BIND(
            IF(?elevation < 525, "blue" ,
            IF(?elevation < 535, "#008AFF",
            IF(?elevation < 545, "magenta",
            IF(?elevation < 555, "red",
            IF(?elevation < 565, "#C60000",
            "black"))))) AS ?distWktColor
          ).
}
GROUP BY ?distName ?elevation ?distWkt ?distWktColor
```

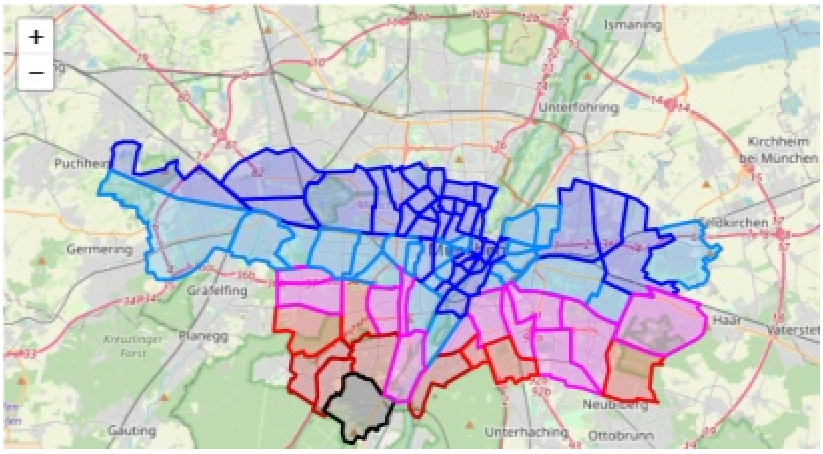


Figure 5.11: Visualising resultant 81 sub-districts of Munich

RasSPARQL query $\boldsymbol{Q}_{S1}$ retrieves all sub-districts of Munich whose spatially averaged elevation exceeds 515 m. Each sub-district is represented as an instance of class `:Region_SubDistrict_Munich` with an associated label (`rdfs:label`) and geometry (`geo:asWKT`). Elevation values are extracted from a raster dataset identified by a raster name containing the keyword "Elevation". Using the function `rasdb:rasSpatialAverage`, the query computes the mean elevation within the polygonal boundary of each sub-district for a fixed timestamp (`2000-02-11T00:00:00+00:00`). Figure 5.11 displays the 81 sub-districts of Munich that satisfy the elevation criterion, and they are spatially positioned according to their geographic boundaries on the OSM map. For visual interpretation, each qualifying sub-district is assigned a colour class based on its elevation range using a nested `IF` statement. The colour scheme progresses from blue (lower elevations) through magenta and red to black (highest elevations). The final output includes the sub-district name, mean elevation, geometry, and assigned visualisation colour, grouped by these attributes to ensure unique results.

- $\boldsymbol{Q}_{S2/Clip}$: Clip and return a temperature raster array over the district of Laim of Munich on 24 September 2023 at time 00:00:00, and return filtered arrays.

```
SELECT ?regionName ?clippedArray {
  ?region a :Region_District_Munich ; rdfs:label ?regionName ; geo:asWKT ?
      regionWkt .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName = 'Laim')
  FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
  BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasClipRaster(?timeStamp, ?regionWkt, ?rasterName)
      AS ?clippedArray)
}
```

This SPARQL query $\boldsymbol{Q}_{S2/Clip}$ extracts a spatially clipped temperature raster subset for the district of `Laim` in Munich at a specified time. The district is represented as an instance of `:Region_District_Munich` and is associated with both a semantic label (`rdfs:label`) and a polygon geometry in WKT format (`geo:asWKT`). The query targets raster datasets whose names contain the string "`Munich_MODIS_Temperature_1km`", indicating that the data originate from MODIS satellite-derived land surface temperature products with a spatial resolution of approximately 1 km. A fixed temporal reference is used to select the raster slice corresponding to the specified date and time, i.e. `2022-01-01T00:00:00+00:00`. The function `rasdb:rasClipRaster` performs a spatial clipping operation, extracting only those raster cells that intersect with the polygon geometry of the `Laim` district. The result is a clipped raster array representing the temperature distribution within the administrative boundary of `Laim` for the selected timestamp. The final output, as illustrated in Figure 5.12, consists of the district name and the corresponding clipped temperature raster, enabling localised spatial-temporal analysis of thermal conditions at the district scale.

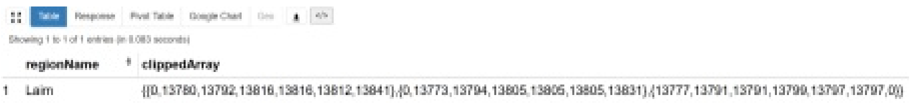


Table Response Pivot Table Google Chart Geo

Showing 1 to 1 of 1 entries (in 0.083 seconds)

	regionName	clippedArray
1	Laim	{{0,13780,13792,13816,13816,13812,13841},{0,13773,13794,13805,13805,13805,13831},{13777,13791,13791,13799,13797,13797,0}}

Figure 5.12: Visualising resultant array for district Laim, Munich

- $\boldsymbol{Q}_{S3/Clip}$: Clip a portion of an available temperature raster data over Munich based on a user-defined custom region on 24 July 2024 at time 00:00:00, and return filtered arrays.

```
SELECT ?clippedArray ?customRegionWkt {
   ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
   FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
   BIND ('POLYGON ((11.538391 48.155093,
   11.460114 48.180281, 11.528778 48.136308,
   11.45256 48.129893, 11.541138 48.117976,
   11.502686 48.082208, 11.56311 48.115225,
   11.602936 48.068444, 11.586456 48.127143,
   11.66748 48.131726, 11.580276 48.148679,
   11.624908 48.178907, 11.570663 48.167917,
   11.545944 48.206829, 11.538391 48.155093))' AS ?customRegionWkt)
   BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
   BIND (rasdb:rasClipRasterAnyGeom(?timeStamp, ?customRegionWkt, ?
       rasterName) AS ?clippedArray)
}
```

Query $\boldsymbol{Q}_{S3/Clip}$ extracts a spatially clipped subset of temperature raster data over Munich based on a user-defined custom polygon rather than an administrative boundary. A custom region of interest is explicitly defined using a Well-Known Text (WKT) polygon that represents an arbitrary spatial extent specified by the user. The function `rasdb:rasClipRasterAnyGeom` performs a spatial clipping operation between the raster and the custom polygon geometry, returning only the raster cells that intersect with the geometry of the specified region. The result preserves the original raster structure and contains pixel-level temperature values for the user-defined area in `?clippedArray`.

Figure 5.13 displays a tabular version of the query output, both the clipped raster array and the geometry of the custom region, enabling flexible, on-demand spatial subletting of raster data for non-standard regions of interest. Figure 5.14 visualises the resultant geometry on the OSM map.

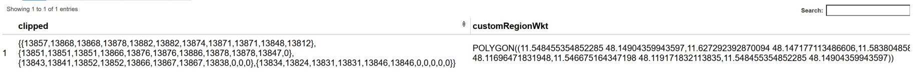


Table Response Pivot Table Google Chart Geo

Showing 1 to 1 of 1 entries

Search:

	clipped	customRegionWkt
1	{{13857,13868,13868,13878,13882,13882,13874,13871,13871,13848,13812}, {13851,13851,13851,13866,13876,13876,13886,13878,13878,13847,0}, {13843,13841,13852,13852,13866,13867,13867,13838,0,0,0},{13834,13824,13831,13831,13846,13846,0,0,0,0,0}}	POLYGON((11.548455354852285 48.14904359943597,11.627292392870094 48.147177113486606,11.5838048 48.11696471831948,11.546675164347198 48.119171832113835,11.548455354852285 48.14904359943597))

Figure 5.13: Clipped array of user-defined, arbitrary-shaped vector region

In contrast to $\boldsymbol{Q}_{S2/Clip}$, which relies on predefined administrative territories (e.g., the district of Laim), $\boldsymbol{Q}_{S3/Clip}$ enables ad hoc spatial analysis over arbi-

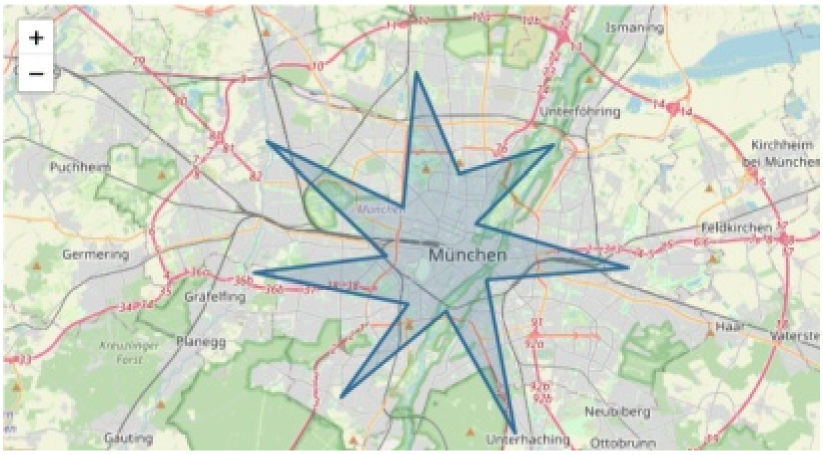


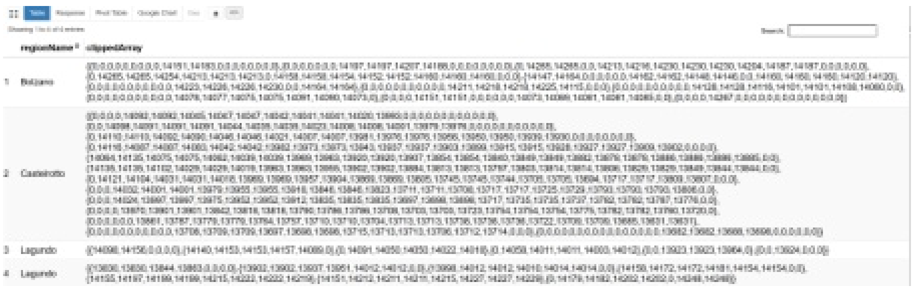
Figure 5.14: Visualising user-defined arbitrary-shaped vector region over Munich

trary geometries. This extends user-driven exploration of urban thermal patterns to use-case-specific neighbourhoods, corridors, or functional urban zones that do not align with official boundaries. Similarly, $\boldsymbol{Q}_{S4/Clip}$ performs a simultaneous spatial clipping of a raster dataset over multiple predefined regions in Tyrol, namely Bolzano, Castelrotto, and Lagundo. Each region is modelled as an instance of `Region_Tyrol`. For each selected region, the function `rasdb:rasClipRaster` clips the raster to the corresponding polygon geometry, returning a region-specific raster subset (`?clippedArray`). The query thus produces one clipped raster array per region, preserving the original spatial resolution and pixel-level values within each administrative boundary.

- $\boldsymbol{Q}_{S4/Clip}$: Clip an available temperature raster data over South Tyrol with regions 'Bolzano', 'Castelrotto', 'Lagundo' on 24 September 2023 at time 00:00:00, and return filtered arrays.

```
SELECT ?regionName ?arrayClipped {
  ?region a :Region_Tyrol ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
      .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName in ('Bolzano','Castelrotto','Lagundo'))
  FILTER (CONTAINS(?rasterName, 'Tyrol'))
  BIND ('2023-09-24T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasClipRaster(?timeStamp, ?regionWkt, ?rasterName) AS ?
      arrayClipped)
}
```

The final output consists of the region name and its associated clipped raster array as shown in Figure 5.15 with map visualisation in Figure 5.16. This type of query enables parallel, comparative analysis of raster data (e.g., temperature, precipitation, or vegetation indices) across multiple vector regions in a single query.



	regionName	clippedArray
1	Bolzano	[illegible]
2	Castelrotto	[illegible]
3	Lagundo	[illegible]
4	Lagundo	[illegible]

Figure 5.15: Visualising 'Bolzano', 'Castelrotto', 'Lagundo' and respective clipped arrays

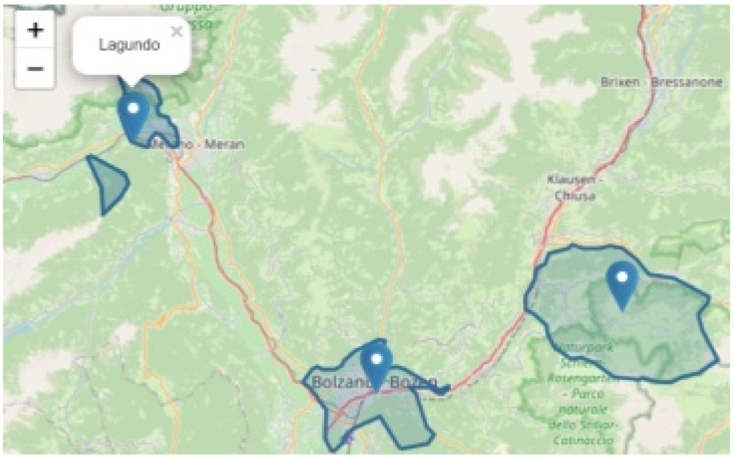

Lagundo
Merano - Meran
Brixen - Bressanone
Klausen - Chiusa
Bolzano - Bozen
Naturpark Schlern-Rosengarten - Parco naturale dello Sciliar-Catinaccio

Figure 5.16: Visualising 'Bolzano', 'Castelrotto', 'Lagundo' districts of S.Tyrol

- ***Q***$_{S5}$: List all vineyards in South Tyrol and their spatial average temperature on 03 March and September 2023.

```
SELECT ?regionName ?tempK ?regionWkt {
 ?region a :Region_Tyrol.
 ?region rdfs:label ?regionName .
 ?region geo:asWKT ?regionWkt .
 ?gridCoverage a :Raster .
 ?gridCoverage rasdb:rasterName ?rasterName .
 FILTER (CONTAINS(?regionName, 'Vino'))
 FILTER (CONTAINS(?rasterName, 'Tyrol'))
 BIND ('2023-10-33T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialMinimum(?timeStamp, ?regionWkt, ?rasterName) AS ?
     tempK)
}
```

- $\boldsymbol{Q}_{ST6}$: Determine the spatial temporal average temperature over the districts 'Würzburg' and 'Bayreuth' in the state of Bavaria over the time period between 15th June and 21st July 2023.

```
SELECT ?regionName ?tempK ?regionWkt {
?region a :Region_Bavaria ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (?regionName in ('Würzburg', 'Bayreuth')) # Polygons with holes
FILTER (CONTAINS(?rasterName, 'Bavaria'))
BIND ('2023-06-15T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
BIND ('2023-07-21T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
BIND (rasdb:rasTemporalAverage(?startTimeStamp, ?endTimeStamp , ?regionWkt
     , ?rasterName) AS ?tempK)
}
```

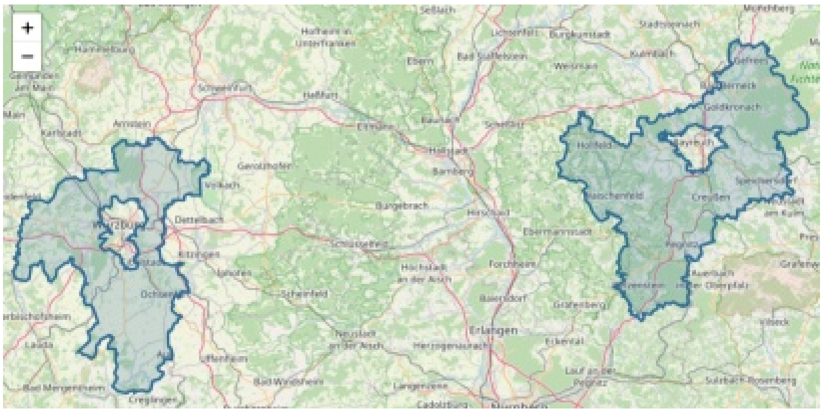


Figure 5.17: Visualising 'Würzburg' and 'Bayreuth' (Polygon with holes)

SPARQL query $\boldsymbol{Q}_{ST6}$ computes the spatial-temporal average temperature for the Bavarian districts of Würzburg and Bayreuth over the period from 15 June to 21 July 2023. Each district is represented as an instance of `:Region_Bavaria` and the region's name is associated with a semantic label (`rdfs:label`) and a polygon geometry in WKT format (`geo:asWKT`). Two temporal boundaries are defined using ISO 8601 timestamps, specifying the start (2023-06-15) and end (2023-07-21) of the analysis period. The function `rasdb:rasTemporalAverage` then aggregates raster temperature values both spatially, across each district's polygon, and temporally, across all raster snapshots (or array slices) within the specified time interval. The resulting value (`?tempK`) represents the spatial-temporal mean temperature in Kelvin for each district over the full six-week period. The query returns the district names, the calculated spatial-temporal average temperature, and the corresponding district geometries, enabling comparative assessment of thermal conditions between Würzburg and Bayreuth during early summer 2023.

- ***Q***$_{ST7}$: Find the temporal average temperature of the Söderköping municipality of Sweden over the time period between 05th April and 19th April 2022.

```
SELECT ?regionName ?tempK ?regionWkt {
?region a :Region_Sweden ; rdfs:label ?regionName ; geo:asWKT ?regionWkt .
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (?regionName = 'Söderköping')
FILTER (CONTAINS(?rasterName, 'Sweden_Surface_Temperature'))
BIND ('2022-04-05T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
BIND ('2022-06-19T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
BIND (rasdb:rasTemporalMaximum(?startTimeStamp, ?endTimeStamp , ?regionWkt
     , ?rasterName) AS ?tempK)
```

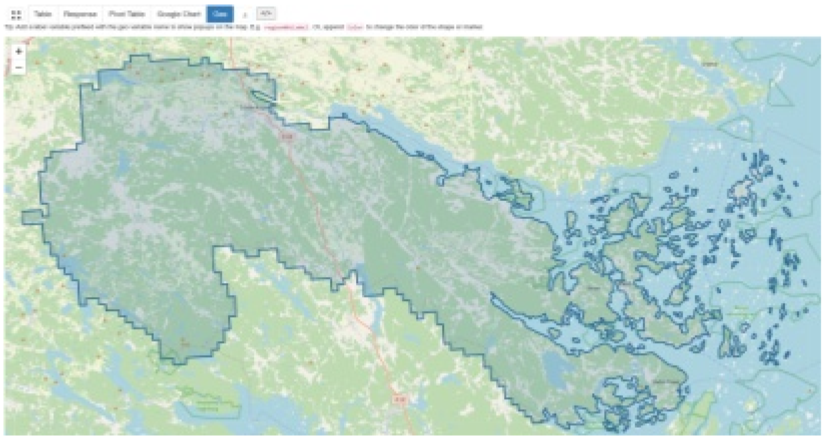


Figure 5.18: Söderköping with islands (MultiPolygon)

This SPARQL query computes a spatial-temporal maximum temperature for the municipality of Söderköping, Sweden, over the period from 5 April to 19 April 2022. Söderköping municipality, which includes multiple island features, is represented as an instance of `:Region_Sweden`. The raster function **`rasdb:rasTemporalMaximum`** aggregates raster values across all raster snapshots within this period and computes the temporal maximum temperature for each raster cell intersecting the geometry of Söderköping. The resulting output (`?tempK`) represents the peak surface temperature pattern over the specified time interval for the entire municipality, including its island components. The query returns the municipality name, the derived maximum temperature from raster data snapshots, and the corresponding multipolygon geometry as shown in Figure 5.18.

This demonstrates the OntoRaster framework's capability to perform spatial-temporal raster analytics across complex multipolygonal regions, enabling robust environmental assessments for coastal or archipelago administrative territories.

5.3.2 GeoNames + Raster Data

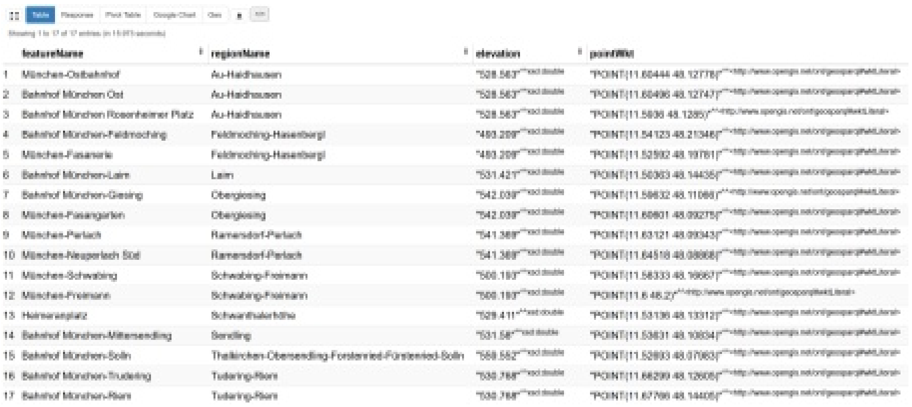
GeoNames has point features and is semantically conceptualised by GeoNames ontology with prefix (`gn:`). The following SPARQL query $Q_{S8/GN}$ identifies all rail stations within Munich's administrative districts whose district-level mean elevation falls within a specified mid-range threshold: below 505m or above 525m on a particular datetime. Munich districts are modelled as polygonal regions of class `:Region_District_Munich`, each associated with a WKT geometry. Rail stations are represented by class `gn:RSTN` and their spatial point locations are also encoded using WKT geometries (`pointWkt`). A GeoSPARQL spatial predicate `geof:sfWithin` is applied to retain only those rail stations whose point geometries (`pointWkt`) fall inside the corresponding district polygons (`regionWkt`).

Aggregated raster information is derived from available raster datasets using raster names described in a class (`:Raster`), filtered to include only the elevation raster dataset. The function `rasdb:rasSpatialAverage` computes the mean elevation over each district polygon, rather than at individual station locations over a fixed timestamp (11 February 2000). This district-wise elevation value is then associated with all rail stations contained within that district. A logical filter selects only stations located within districts with a spatial average elevation below 505m or above 525m, highlighting both low-lying and topographically elevated urban areas.

- $Q_{S8/GN}$: Find all rail stations within those districts in Munich whose spatial average land elevation is less than 505 or more than 525 meters.

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX gn: <https://www.geonames.org/ontology#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?featureName ?regionName ?elevation ?pointWkt {
 ?region a :Region_District_Munich ; rdfs:label ?regionName ; geo:asWKT ?
      regionWkt .
 ?gname a gn:RSTN; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Elevation'))
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS ?
      elevation)
 FILTER(?elevation < 505 || ?elevation > 525)
}
```

The query returns the station name (as `featureName`), the district name (as `regionName`), the average elevation, and the station geometry (`pointWkt`) as shown in Figure 5.19, and it enables combined semantic, spatial, and environmental analysis. The query integrates vector-based spatial reasoning, raster-based elevation analysis, and point-feature data from GeoNames.

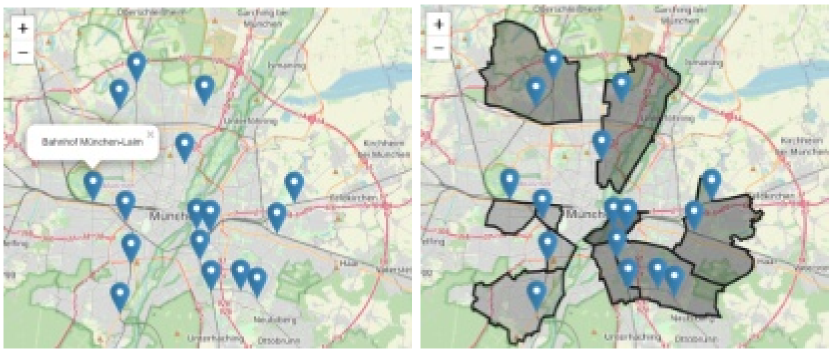


Showing 1 to 17 of 17 entries (in 15.973 seconds)

	featureName	regionName	elevation	pointWkt
1	München-Ostbahnhof	Au-Haidhausen	"528.563"^^xsd:double	"POINT(11.60444 48.12778)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
2	Bahnhof München Ost	Au-Haidhausen	"528.563"^^xsd:double	"POINT(11.60496 48.12747)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
3	Bahnhof München Rosenheimer Platz	Au-Haidhausen	"528.563"^^xsd:double	"POINT(11.5936 48.1285)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
4	Bahnhof München-Feldmoching	Feldmoching-Hasenbergl	"493.209"^^xsd:double	"POINT(11.54123 48.21346)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
5	München-Fasanerie	Feldmoching-Hasenbergl	"493.209"^^xsd:double	"POINT(11.52592 48.19761)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
6	Bahnhof München-Laim	Laim	"531.421"^^xsd:double	"POINT(11.50363 48.14435)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
7	Bahnhof München-Giesing	Obergiesing	"542.039"^^xsd:double	"POINT(11.59632 48.11068)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
8	München-Fasangarten	Obergiesing	"542.039"^^xsd:double	"POINT(11.60801 48.09275)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
9	München-Perlach	Ramersdorf-Perlach	"541.389"^^xsd:double	"POINT(11.63121 48.09343)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
10	München-Neuperlach Süd	Ramersdorf-Perlach	"541.389"^^xsd:double	"POINT(11.64518 48.08868)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
11	München-Schwabing	Schwabing-Freimann	"500.193"^^xsd:double	"POINT(11.58333 48.16667)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
12	München-Freimann	Schwabing-Freimann	"500.193"^^xsd:double	"POINT(11.6 48.2)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
13	Heimeranplatz	Schwanthalerhöhe	"529.411"^^xsd:double	"POINT(11.53136 48.13312)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
14	Bahnhof München-Mittersendling	Sendling	"531.58"^^xsd:double	"POINT(11.53631 48.10834)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
15	Bahnhof München-Solln	Thalkirchen-Obersendling-Forstenried-Fürstenried-Solln	"558.552"^^xsd:double	"POINT(11.52893 48.07063)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
16	Bahnhof München-Trudering	Tudering-Riem	"530.768"^^xsd:double	"POINT(11.66299 48.12606)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
17	Bahnhof München-Riem	Tudering-Riem	"530.768"^^xsd:double	"POINT(11.67766 48.14405)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>

Figure 5.19: Resultant 17 districts with their elevation and rail stations (markers) of Munich

Figure 5.20 visualises the GeoNames point features on the map by adding `?pointWkt, ?pointWktLabel ?pointWktColor ?regionWkt ?regionWktColor` in the `SELECT` clause of $\boldsymbol{Q}_{S7/GN}$ and adding two BINDs in the same query.



(a) GeoNames pointWkt (b) pointWkt + regionWkt

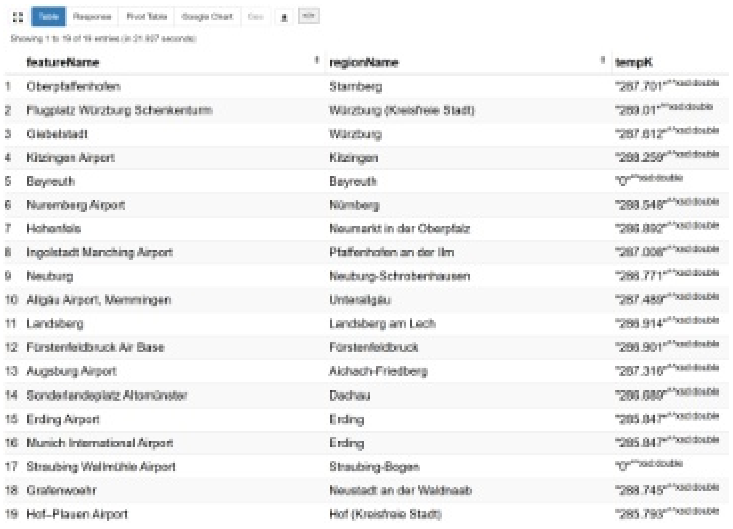
Figure 5.20: Visualising resultant railstations (`gn:RSTN`) and 17 districts of Bavaria

Similarly, SPARQL query $\boldsymbol{Q}_{S9/GN}$ retrieves all airports in Bavaria and their corresponding administrative districts, together with the spatially averaged land surface temperature (LST) for 4 September 2023. The query provides a unified semantic–geospatial framework that integrates GeoNames point features, i.e., airports (`gn:S.AIRP`), administrative regions, i.e., states, and aggregated temperature data from the corresponding raster data for Bavaria.

- $\boldsymbol{Q}_{S9/GN}$: Find all airports and receptive districts in Bavaria and their spatial average land surface temperature on 4th September 2023.

```sparql
SELECT ?featureName ?regionName ?tempK ?pointWkt ?regionWkt ?
    regionWktColor {
 ?region a :Region_Bavaria ; rdfs:label ?regionName ; geo:asWKT ?
     regionWkt.
 ?gname a gn:S.AIRP; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
    FILTER (CONTAINS(?rasterName, 'Bavaria'))
    BIND('#181818' AS ?regionWktColor)
    BIND ('2023-09-04T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS
        ?tempK)
    BIND (rasdb:rasTemporalAverage(?startTimeStamp, ?endTimeStamp , ?
        regionWkt, ?rasterName) AS ?tempK)
}
```

As shown in Figure 5.21, the query returns the airport name, district name, spatially averaged temperature (in Kelvin), and both point and polygon geometries, enabling integrated semantic, spatial, and environmental analysis of airport–district relationships under regional thermal conditions.

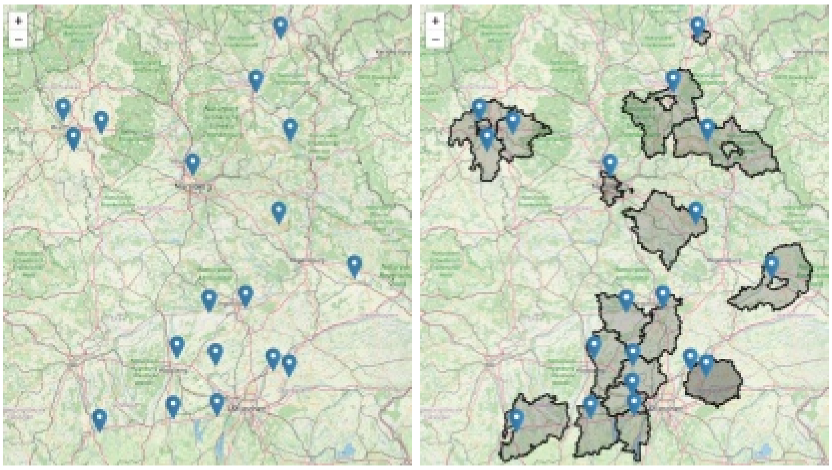


Showing 1 to 19 of 19 entries (in 21.937 seconds)

	featureName	regionName	tempK
1	Oberpfaffenhofen	Starnberg	"287.701"^^xsd:double
2	Flugplatz Würzburg Schenkenturm	Würzburg (Kreisfreie Stadt)	"289.01"^^xsd:double
3	Giebelstadt	Würzburg	"287.812"^^xsd:double
4	Kitzingen Airport	Kitzingen	"288.259"^^xsd:double
5	Bayreuth	Bayreuth	"0"^^xsd:double
6	Nuremberg Airport	Nürnberg	"288.548"^^xsd:double
7	Hohenfels	Neumarkt in der Oberpfalz	"286.892"^^xsd:double
8	Ingolstadt Manching Airport	Pfaffenhofen an der Ilm	"287.008"^^xsd:double
9	Neuburg	Neuburg-Schrobenhausen	"288.771"^^xsd:double
10	Allgäu Airport, Memmingen	Unterallgäu	"287.489"^^xsd:double
11	Landsberg	Landsberg am Lech	"286.914"^^xsd:double
12	Fürstenfeldbruck Air Base	Fürstenfeldbruck	"286.901"^^xsd:double
13	Augsburg Airport	Aichach-Friedberg	"287.316"^^xsd:double
14	Sonderlandeplatz Altomünster	Dachau	"286.089"^^xsd:double
15	Erding Airport	Erding	"285.847"^^xsd:double
16	Munich International Airport	Erding	"285.847"^^xsd:double
17	Straubing Wallmühle Airport	Straubing-Bogen	"0"^^xsd:double
18	Grafenwoehr	Neustadt an der Waldnaab	"288.745"^^xsd:double
19	Hof–Plauen Airport	Hof (Kreisfreie Stadt)	"285.793"^^xsd:double

Figure 5.21: Resultant airports and respective districts of Bavaria

The SPARQL query $\boldsymbol{Q}_{S10/GN}$ extends query $\boldsymbol{Q}_{S9/GN}$ by adding temporal dimensions. It retrieves all airports in Bavaria and their corresponding administrative districts, along with the spatial-temporal average temperature over the period from 15th June, 2023 to 21 July, 2023.



(a) Resultant airports (markers) (b) Airports with respective districts

Figure 5.22: Visualising resultant airports (`gn:S.AIRP`) with respective 19 districts of Bavaria

- ***Q***$_{S10/GN}$: Find all airports and receptive districts in Bavaria and their temporal average land surface temperature over the time period between 15th June and 21st July 2023.

```
SELECT ?featureName ?regionName ?tempK ?pointWkt ?regionWkt
      ?regionWktColor {
 ?region a :Region_Bavaria ; rdfs:label ?regionName ; geo:asWKT ?
     regionWkt.
 ?gname a gn:S.AIRP; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
    FILTER (CONTAINS(?rasterName, 'Bavaria'))
    BIND('#181818' AS ?regionWktColor)
    BIND ('2023-06-15T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
    BIND ('2023-07-21T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
    BIND (rasdb:rasTemporalAverage(?startTimeStamp, ?endTimeStamp , ?
         regionWkt, ?rasterName) AS ?tempK)
}
```

Now we change the AOI from Bavaria to South Tyrol in the following SPARQL query ***Q***$_{S11/GN}$, which identifies mountain peaks located within municipalities of South Tyrol whose spatially aggregated elevation exceeds 2000 m, integrating semantic point features, administrative boundaries, and raster-based elevation data. Municipalities in South Tyrol are modelled as polygonal regions of type `:Region_Tyrol`. Mountain peaks are retrieved as GeoNames point features of class `gn:T.PK` along with WKT point geometries.

The query then filters for municipalities in which the spatially aggregated elevation exceeds 2000 m, thereby selecting only high-altitude alpine areas. The final result returns the peak name, municipality name, and the corresponding raster-derived elevation value, enabling integrated semantic, spatial, and environmental analysis of mountainous terrain in South Tyrol.

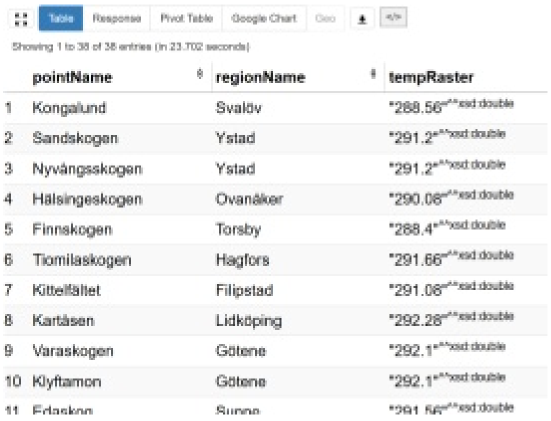
- $\boldsymbol{Q}_{S11/GN}$: Find all mountain peaks within those municipalities in South Tyrol whose spatial average elevation is more than 2000 meters.

```
SELECT ?pointName ?regionName ?elevation {
 ?region a :Region_Tyrol ; rdfs:label ?regionName ; geo:asWKT ?regionWkt .
 ?gname a gn:T.PK; rdfs:label ?pointName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Tyrol'))
 BIND ('2023-03-03T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialMaximum(?timeStamp, ?regionWkt, ?rasterName) AS
         ?elevation)
    FILTER(?elevation> 2000) .
}
```

The SPARQL query $\boldsymbol{Q}_{S12/GN}$ retrieves forest locations within municipalities of Sweden whose spatially aggregated maximum land surface temperature exceeds 281 K, integrating semantic point features, administrative boundaries, and raster-based thermal data. Municipalities whose spatially aggregated maximum temperature exceeds 281 K are then retained. The query returns the forest name, municipality name, and the corresponding raster-derived temperature value as shown in Figure 5.23 and visualised in Figure 5.24, enabling integrated semantic, spatial, and environmental analysis of forested areas under elevated thermal conditions in Sweden.

- $\boldsymbol{Q}_{S12/GN}$: Find all forests within those municipalities in Sweden whose spatial maximum temperature is above 281 Kelvin.

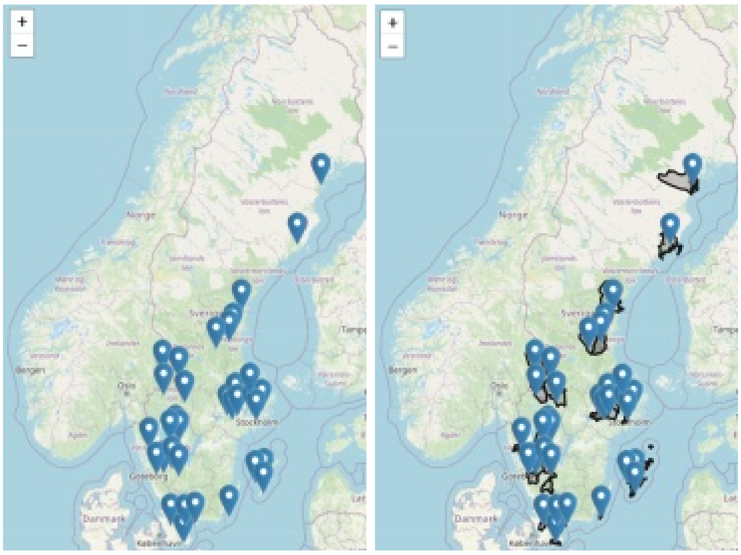
```
SELECT ?pointName ?regionName ?tempK {
 ?region a :Region_Sweden ; rdfs:label ?regionName; geo:asWKT ?regionWkt .
 ?gname a gn:V.FRST; rdfs:label ?pointName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Sweden'))
 BIND ('2022-08-24T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialMaximum(?timeStamp, ?regionWkt, ?rasterName) AS
         ?tempK)
 FILTER(?tempK> 281) .
}
```



Showing 1 to 38 of 38 entries (in 23.702 seconds)

	pointName	regionName	tempRaster
1	Kongalund	Svalöv	"288.56"^^xsd:double
2	Sandskogen	Ystad	"291.2"^^xsd:double
3	Nyvångsskogen	Ystad	"291.2"^^xsd:double
4	Hälsingeskogen	Ovanåker	"290.08"^^xsd:double
5	Finnskogen	Torsby	"288.4"^^xsd:double
6	Tiomilaskogen	Hagfors	"291.66"^^xsd:double
7	Kittelfältet	Filipstad	"291.08"^^xsd:double
8	Kartåsen	Lidköping	"292.28"^^xsd:double
9	Varaskogen	Götene	"292.1"^^xsd:double
10	Klyftamon	Götene	"292.1"^^xsd:double
11	Edaskog	Sunne	"291.56"^^xsd:double

Figure 5.23: Visualising the 38 municipalities of Sweden



(a) Resultant forest(s) (markers) (b) Respective municipalities

Figure 5.24: Visualising resultant forest(s) (`gn:V.FRST`) with respective 38 municipalities of Sweden

5.3.3 OpenStreetMap + Data

As previously mentioned, OpenStreetMap provides 2D footprints (polygons) of Buildings such as schools, churches, Parking Lots, hospitals, hotels, and more than 100 categories. The RasSPARQL query $\boldsymbol{Q}_{S13/OSM}$ retrieves school buildings (classified under `lgdo:School` from the LinkedGeoData ontology) located within the administrative districts of Munich, but only those districts where the spatially averaged elevation exceeds 515 meters as computed from an elevation raster data at a specified timestamp on 11th February. It integrates vector geometries of OSM building footprints (`?bldgWkt`) with district geometries (`?distWkt`) via a GeoSPARQL spatial containment function (`geof:sfWithin`). The raster spatial average function (`rasdb:rasSpatialAverage`) operates on elevation-specific coverage data, ensuring that only districts above the threshold contribute schools to the result set, thereby highlighting elevation-driven geographic patterns in Munich's educational infrastructure.

For qualifying districts, the query extracts the school name, district name, computed mean elevation, district and building geometries, and assigns fixed styling colours (dark gray for districts by `?distWktColor = '#181818'`, red for school buildings by `?bldgWktColor = 'red'`) to facilitate subsequent geo map visualization as shown in Figure 5.25.

- $\boldsymbol{Q}_{S13/OSM}$: Find all schools and respective districts of Munich whose spatial average elevation is above 515 meters

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?bldgName ?distName ?elevation ?distWkt ?distWktColor ?bldgWkt ?
     bldgWktColor {
 ?building a lgdo:School . # Church, Parking, Hospital, Hotel
    ?building rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
 ?region a :Region_District_Munich ; rdfs:label ?distName ; geo:asWKT ?
      distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND('#181818' AS ?distWktColor)
 BIND('red' AS ?bldgWktColor)
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
      elevation)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
 FILTER (CONTAINS(?rasterName, 'Elevation'))
 FILTER(?elevation > 515)
}
```

This query directly generates the visualisation, in which the resulting school buildings are plotted exclusively within the selected districts of Munich whose average elevation exceeds 515 m, as computed from the raster. The map reveals a pronounced concentration of qualifying schools in the southern and southwestern peripheral districts of Munich (e.g., areas toward Planegg, Gräfelfing, and Neubiberg), which exhibit higher terrain as the city transitions toward the Alpine foreland, while central and northern districts—generally closer to the city's 520 m reference elevation but locally averaging below the strict 515 m threshold in parts—contain few or no represented schools. Insets magnify clusters of 7 OSM 2D buildings in one southwestern area and 3 in another, underscoring localised hotspots of elevated terrain hosting school facilities.

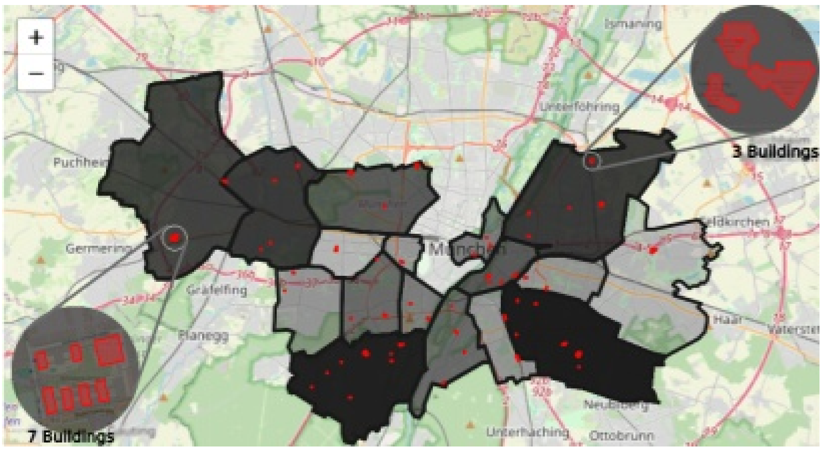


Figure 5.25: Resultant school buildings and respective districts of Munich

The RasSPARQL query $\boldsymbol{Q}_{S14/OSM}$ mimic the operations of $\boldsymbol{Q}_{S13/OSM}$ with one change, that is, vegetation raster data instead of temperature raster data. The SPARQL query identifies OSM school buildings (via `lgdo:School`) that fall within Munich's districts where the district-averaged vegetation exceeds 0.4, derived from NDVI raster dataset on 1st January, 2022. NDVI serves as a standardised remote-sensing metric of vegetation density and vigour (typically ranging from -1 to +1), with values ¿0.4 generally indicating moderate to substantial green biomass—such as well-vegetated suburban or semi-urban areas with trees, parks, or managed greenery—rather than sparse or absent live vegetation characteristic of highly impervious urban cores [65]. By applying this vegetation threshold, i.e., 0.4, the query $\boldsymbol{Q}_{S14/OSM}$ selectively highlights schools in districts with comparatively higher greenery, revealing potential correlations between educational sites and enhanced urban/semi-urban vegetation quality or quantity in Munich as depicted in Figure 5.26. It contrasts with elevation-based filtering (as in the prior figure) by emphasising vegetation-based drivers of school distribution.

- $\boldsymbol{Q}_{S14/OSM}$: Find all schools and respective districts of Munich whose spatial average vegetation (NDVI) is high

```
SELECT ?bldgName ?distName ?ndvi ?distWkt ?distWktColor ?bldgWkt ?
     bldgWktColor {
   ?building a lgdo:School ; rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
   ?region a :Region_District_Munich ; rdfs:label ?distName ; geo:asWKT ?
        distWkt .
   ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
   BIND('#181818' AS ?distWktColor)
   BIND('red' AS ?bldgWktColor)
   BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
   BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
        ndvi)
   FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
   FILTER (CONTAINS(?rasterName, 'NDVI'))
   FILTER(?ndvi > 0.4) # NDVI Scale [0.1 (Desert) to >= 1.0 (Forest)]
}
```

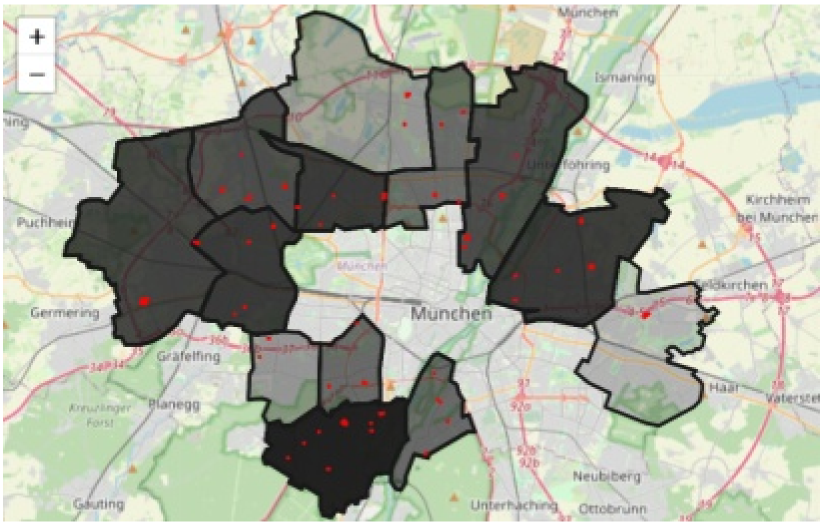


Figure 5.26: Resultant school buildings and respective districts of Munich

The RasSPARQL query $\boldsymbol{Q}_{S15/OSM}$ extends the operations of $\boldsymbol{Q}_{S14/OSM}$ at finer-grained administrative units of Munich, such as sub-districts instead of districts. The SPARQL query retrieves school buildings within those sub-districts (via class `:Region_SubDistrict_Munich`) where the spatial average vegetation exceeds 0.4, derived from NDVI raster dataset on 1st January, 2022.

Compared to the prior district-level query $\boldsymbol{Q}_{S14/OSM}$, this query enables identification of greenery at finer spatial resolution (sub-districts), revealing intra-district heterogeneity in vegetation-school associations as depicted in Figure 5.27.

- $\boldsymbol{Q}_{S15/OSM}$: Find all schools and their respective sub-districts of Munich whose spatial average vegetation (NDVI) is moderately high

```
SELECT ?bldgName ?distName ?ndvi ?distWkt ?distWktColor ?bldgWkt ?
     bldgWktColor {
    ?building a lgdo:School ; rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
    ?region a :Region_SubDistrict_Munich ; rdfs:label ?distName ; geo:
         asWKT ?distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    BIND('#181818' AS ?distWktColor)
    BIND('red' AS ?bldgWktColor)
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
         ndvi)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
    FILTER (CONTAINS(?rasterName, 'NDVI'))
    FILTER(?ndvi > 0.4)
}
```

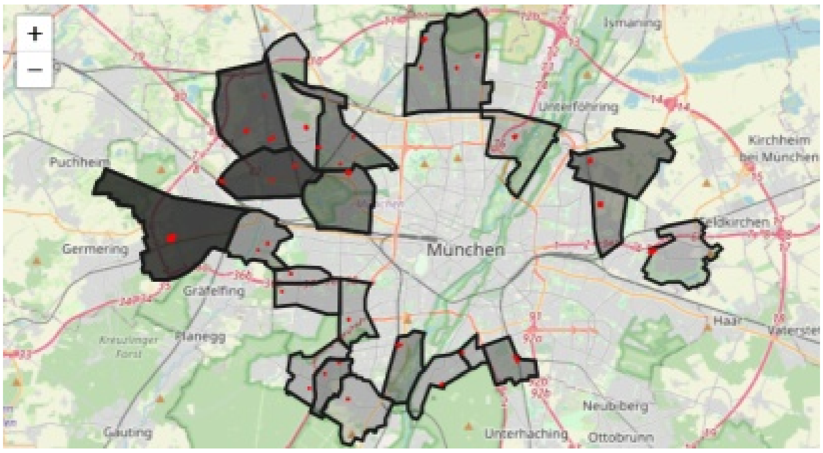


Figure 5.27: Resultant school buildings and respective sub-districts of Munich

The RaSPARQL query $\boldsymbol{Q}_{S16/OSM}$ identifies OSM 2D buildings of type hospital (`lgdo:Hospital`) within sub-districts of Munich, where the spatial average vegetation (NDVI) exceeds 0.4 on 1st January, 2022, indicating moderate-to-high vegetation cover (e.g., parks, trees, or suburban greenery). Compared to the prior school-focused sub-district query, this shifts the facility type to hospitals while retaining the same vegetation threshold at the sub-district level, enabling analysis of healthcare infrastructure in greener sub-areas versus denser urban cores. Figure 5.28 displays a visualisation similar to the school sub-district map.

- $\boldsymbol{Q}_{S16/OSM}$: Find all hospitals and their respective sub-districts of Munich whose spatial average vegetation (NDVI) is high

```
SELECT ?bldgName ?distName ?ndvi ?distWkt ?distWktColor ?bldgWkt ?
     bldgWktColor {
    ?building a lgdo:Hospital; rdfs:label ?bldgName; geo:asWKT ?bldgWkt .
    ?region a :Region_SubDistrict_Munich ; rdfs:label ?distName ; geo:
         asWKT ?distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    BIND('#181818' AS ?distWktColor)
    BIND('red' AS ?bldgWktColor)
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
         ndvi)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
    FILTER (CONTAINS(?rasterName, 'NDVI'))
    FILTER(?ndvi > 0.4)
}
```

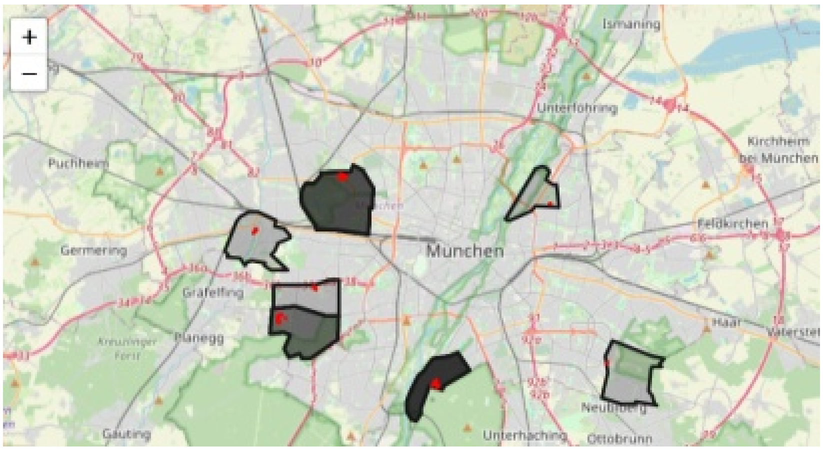


Figure 5.28: Resultant hospitals and corresponding sub-districts of Munich

5.3.4 CityGML + Raster Data

Huang et al. [119] employ a materialised knowledge graph workflow: OWL ontologies + semantic constraints (for matching multiple representations) + SWRL rules (for inferring both data integration mappings and visualisation parameters). Ahmadian et al. [3] study the integration of OpenStreetMap and CityGML using formal concept analysis. Hansson et al. [104] outlined a semantic framework to integrate CityGML data using the VKG paradigm. One of our collaborators, Ding et al. [69], adopts another VKG approach using the VKG system Ontop, in which CityGML data are exposed as a SPARQL-endpoint-compatible KG without materialisation, using the CityGML ontology and declarative mappings. This enables complex GeoSPARQL queries that integrate OpenStreetMap and CityGML data while preserving the efficiency of the underlying relational database, but querying raster data is not yet supported. We are extending the work [69] to integrate CityGML 3D building data with raster data.

5.3.4.1 3DCityDB: Storage for CityGML

The 3D City Database (3DCityDB)[8] is a free and open-source package for managing, analysing, and utilising virtual 3D city models on top of an RDBMS, specifically PostgreSQL and PostGIS. We have utilised the predefined SQL schema[9] and the importer-exporter[10] tool provided by 3DCityDB to import CityGML files into the *citydb* table in PostgreSQL. Here, we observed that several columns in the CityGML building table are empty, and there is no column describing building locations, which makes it challenging to use this data to compose queries that incorporate raster data and additional spatial analysis. We created mappings to connect the data to the aforementioned LinkedGeoData and CityGML ontology.

5.3.4.2 Linking OSM and CityGML

OSM data primarily comprises building footprints (2D polygons) and *points of interest* (POIs), and the CityGML dataset provides 3D building models ranging from coarse (LoD0) to very detailed (LoD4) with associated building information, e.g., building address, height, roof surfaces, etc. To resolve this heterogeneity, a link between CityGML and OSM data at both the data and ontology levels is required [69].

Database Level. We observe that ground footprints (polygons) are common features shared by OSM and CityGML data (specifically, L0D0) and can be geometrically matched to link the two datasets. Ding et al. [69] provided three stages for OSM-CityGML linking.

[8]https://github.com/3dcitydb
[9]https://docs.3dcitydb.org/1.2/3dcitydb/relational-schema/
[10]https://github.com/3dcitydb/importer-exporter

Initially, we used a geometric matching technique [77, 158] to spatially align CityGML 3D building ground footprints (LoD0) with OSM 2D building footprints. This provides geolocation information to CityGML data, which initially contained only building information. Given any CityGML building (`city3D`) and OSM building (`osm2D`), their direct spatial correspondences are identified based on Equation (1) [77, 158], derived from individual areas of any CityGML (Area($\texttt{city3D}_j$)) and OSM polygons (Area($\texttt{osm2D}_k$))

$$\frac{\text{Area}(osm2D_k \cap city3D_j)}{\min(\text{Area}(osm2D_k), \text{Area}(city3D_j))} \geq t$$

whereArea($osm2D_k \cap city3D_j$) represents the overlapping area between the k-th OSM polygon and the j-th CityGML polygon; t is an empirical hyperparameter, which can be adjusted based on the spatial consistency between the two datasets. Fan et al. [77] demonstrated four possible matching ratios: 1:1, 1:n, m:1, and m:n. Ding et al. [69] acknowledged that 1:1 correspondences are not always feasible due to dataset differences. Step 2 enriches spatial matching by including adjacent surfaces, reducing 0:1 relations by associating ancillary features (e.g., stairs) with main buildings. Step 3 matches OSM POIs to CityGML buildings using OSM building footprints as mediators to enrich the semantic information in CityGML data.

In OntoRaster framework, OSM building footprint data are already integrated with raster data for spatial query processing, we can argue that CityGML building data associated with OSM data can also be queried with raster data, provided that relevant ontologies are linked.

Ontological Level. Integration at the database level should be extended upstream to the ontology and knowledge graph levels by creating additional appropriate mappings and ontological axioms to capture the correspondences between OSM buildings and CityGML buildings. The base CityGML ontology (adapted from the University of Geneva) is extended with the LinkedGeoData (LGD) ontology for OSM data by adding respective classes and properties. Suitable VKG mappings are defined to align the integrated database with the aforementioned ontologies. To model linkages between OSM and CityGML data within the OntoRaster framework, two approaches can be adopted. A simple approach is to use the `owl:sameAs` assertion to map OSM IDs to CityGML IDs. Ding et al. [69] introduce a rectified "`Association_CityGML_OSM`" class with properties such as "match" and "adjacent", which enables GeoSPARQL queries over the unified VKG to capture nuanced relations. OntoRaster is extended with both approaches.

5.3.4.3 Linking CityGML and Raster

Here, we present several experimental RasSPARQL queries that integrate raster data with CityGML 3D building data.

- $Q_{S17/CityGML}$: Find all hotels with a height above 30 meters and their respective districts of Munich, and corresponding elevation

```
PREFIX : <https://github.com/aghoshpro/OntoRaster/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX lgdo: <http://linkedgeodata.org/ontology/>
PREFIX bldg: <http://www.opengis.net/citygml/building/2.0/>
PREFIX rasdb: <https://github.com/aghoshpro/RasterDataCube/>

SELECT ?bldgName ?bldgHeight ?distName ?elevation {
 ?building a lgdo:Hotel ; rdfs:label ?bldgName ; geo:asWKT ?bldgWkt ;
              owl:sameAs ?citygml3D .
 ?citygml3D bldg:measuredHeight ?bldgHeight .
 ?region a :Region_District_Munich ; rdfs:label ?distName ; geo:asWKT ?
      distWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName)
              AS ?elevation)
  FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
  FILTER (CONTAINS(?rasterName, 'Elevation'))
  FILTER(?bldgHeight > 30) .
}
```

- $Q_{S18/CityGML}$: Find the addresses of all school buildings in Munich that are taller than 30 meters, along with their respective sub-districts and their average elevation

```
SELECT ?bldgName ?bldgHeight ?distName ?elevation {
 ?building a lgdo:School ; rdfs:label ?bldgName ; geo:asWKT ?bldgWkt ;
              owl:sameAs ?citygml3D .
 ?citygml3D bldg:measuredHeight ?bldgHeight .
 ?region a :Region_SubDistrict_Munich ; rdfs:label ?distName ; geo:asWKT ?
      distWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName)
              AS ?elevation)
  FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
  FILTER (CONTAINS(?rasterName, 'Elevation'))
  FILTER(?bldgHeight > 30) .
}
```

5.4 Conclusion

This work validated the design and applicability of OntoRaster as a semantic framework for the unification of raster–vector geospatial data. Through a series of spatial and spatiotemporal RasSPARQL queries, we demonstrated how multidimensional raster data can be semantically integrated with heterogeneous vector datasets, including regional geometries, OpenStreetMap 2D building footprints, CityGML 3D building models, and GeoNames point features. These queries demonstrate that complex analytical tasks can be expressed declaratively within a single-query environment, thereby bridging the long-standing gap between raster processing and semantic geospatial reasoning under the VKG framework. By presenting both tabular query outputs and corresponding visualisations, this study established a complete analytical pipeline that supports interpretable and reproducible spatial analysis. The results confirm that RasSPARQL enables interoperable querying across multiple spatial-temporal representations and abstraction levels, thereby reducing workflow fragmentation and improving analytical transparency.

The `rasGeoTIFF()` function (Table 4.1) remains under development. Future research will focus on extending OntoRaster to support multiple raster datasets within a single RasSPARQL query. These developments will enable more advanced spatial-temporal data integration and large-scale semantic geospatial processing under the VKG paradigm.

Chapter 6

Performance Evaluation

To establish the VKG paradigm as a practical and industrially relevant framework, VKG systems must deliver query responses within a reasonable time, even when processing large-scale, heterogeneous, and multidimensional raster datasets across domains, particularly in the geospatial domain. Existing benchmarking efforts in the VKG literature, as surveyed in Section 6.1, have primarily focused on relational data sources. These efforts have produced well-established benchmark configurations and standardised datasets for assessing query performance, scalability, and optimisation behaviour on structured tabular data. However, to the best of our knowledge, no comprehensive, systematic performance evaluation has yet been conducted for VKG systems that support query processing over integrated raster, relational, and vector data. Meaningful assessment of the performance of such VKG systems, i.e., OntoRaster, in such settings requires the development of realistic benchmarks that faithfully capture the defining characteristics of real-world industrial workloads and encompass the volume, variety, and velocity of large datasets, as well as the intricacies of ontologies and the sophistication of queries. In addition to designing a benchmark, we introduced a novel synthetic data generator for vector and raster data with varying characteristics, aligned with OWL ontologies and OGC standards. These benchmarks are tested using the open-source VKG framework OntoRaster

6.1 Related Works

The Lehigh University Benchmark (LUBM) [97] has been a prominent ontology benchmark, serving as a test suite for ontology-based applications. LUBM comprises an ontology that delineates the university domain, a data generator that produces instance data for the university ontology, and test queries for the data; however, the ontology is small, and the benchmark is not tailored to VKG, since no mappings to a (relational) data source are provided. Another popular benchmark is the Berlin SPARQL Benchmark (BSBM) [42], which compares the performance of native RDF stores and virtualisation-based VKG systems on artificially generated data and simple queries in the e-commerce domain. In contrast to current benchmarks that primarily prioritise scalability and performance, OntoBench [156] emphasises evaluating OWL coverage and specific concept pairings in ontology tools. It enables users to dynamically generate consistent benchmark ontologies by selecting diverse OWL 2 language components via a graphical user interface. A recent work [124] proposed a compliance benchmark to objectively assess the extent to which various RDF triple stores and databases comply with the GeoSPARQL [176] standard for geospatial vector data. Kefalidis et al. [128] proposed a GeoQuestions1089 dataset containing questions and their corresponding GeoSPARQL queries over vector data to benchmark their geospatial question answering engines. A notable gap in these works is the lack of evaluation of query performance on raster data under the VKG paradigm. In the context of VKG, Lanti et al. [145] proposed a novel benchmark for VKG systems based on a real-world dataset from the Norwegian Petroleum Directorate (NPD) within the EU project Optique [54]. They adopted the NPD ontology, which has been mapped to the NPD fact pages dataset stored in a relational database, and queries are formed. Lanti et al. [146] extended this work by introducing an automated testing platform, new experiments, and a larger, more challenging query set that includes aggregate queries, thereby highlighting the importance of semantic query optimisation in the SPARQL-to-SQL translation phase. This paper [146] also introduces *Virtual Instance Generator* (VIG) as a data generator and scaler, which analyses the original relational database instances and their mappings to acquire statistics that guide the generation of new data instances. VIG ensures the newly generated data comply with constraints deriving from the respective ontology and the mappings of the corresponding application domain [148]. The LUBM benchmark is extended by LUBM4OBDA [9] for VKG systems, providing mappings to a relational representation of the original LUBM RDF data. This enables comparison with triple stores and the evaluation of meta-knowledge. This study [155] provide a comprehensive review of the efficiency of various semantic data repositories for managing and querying large spatial datasets within knowledge graph settings. Li et al. [155] showed systematic experiments on diverse geospatial vector data using Virtuoso and the Ontop system to assess processing times for both complex and simple spatial-semantic queries. However, their study did not mention raster data. Existing work on the VKG benchmarks and their supporting infrastructure is fundamentally built on the relational data model. Transitioning to multidimensional raster data would significantly impact the comparability and applicability of these benchmarks, necessitating a substantial redesign across multiple layers. In this chapter, we aim to address this research gap by introducing a novel benchmark for VKGs that support query processing over raster data, combined with relational data, including vector data.

Table 6.1: Benchmark Requirements

Vector								Raster	
BM1	**BM2**	**BM3**	**BM4**	**BM5**	**BM6**	**BM7**	**BM8**	**BM9**	**BM10**
#Points	**Area**	**Uniformity**	**Holes**	**GeoNames**	**OSM**	**CityGML**	**AOI**	**Spatial**	**Temporal**

6.2 Benchmark Requirements

We outline the requirements for the proposed benchmark to evaluate the VKG system's access to raster data. As discussed at the beginning of the chapter 5, we focus here on the last three phases of benchmarking requirements for VKG systems, since these are the ones that facilitate the integrated querying of heterogeneous raster and vector data, which differ from conventional VKG systems [87]. We identify benchmark requirements (BM1–BM10) for assessing specialised KG systems that support querying raster data integrated with vector data, as illustrated in Table 6.1. The proposed benchmark requirements provide a systematic approach to evaluating VKG systems by isolating individual parameters across both the vector and raster domains. This methodological framework follows the principle of controlled experimentation, in which each benchmark varies precisely one parameter while holding all others constant, thereby providing a comprehensive testing environment for VKG systems operating on heterogeneous geospatial raster and vector datasets across any area of interest (AOI).

6.2.1 For vector data (BM1 – BM7)

BM1 (# of points) Benchmark BM1 focuses on variation in point count while holding other parameters constant, including the area, uniformity, and hole configuration of vector geometries, as well as the spatial and temporal resolution of raster data. Therefore, this study provides insights into the computational overhead of the VKG framework's query processing, which varies with geometric complexity in one dimension.

BM2 (area variation) BM2's area variation with fixed geometric parameters enables evaluation of how spatial extent affects query performance, particularly for operations involving spatial joins, buffer calculations, sub-setting, and containment queries, which are fundamental to spatial reasoning within the VKG framework.

BM3 (uniformity) introduces the uniformity parameter to regulate the shape of vector geometries. Regularly shaped geometries (polygons and multipolygons) typically exhibit more predictable spatial relationships, enabling optimisation opportunities in spatial query processing. In contrast, irregular shapes may pose challenges for edge cases in geometric algorithms and spatial relationship computations.

BM4 (number of holes) BM4's variation in the number of holes or inner rings addresses the complexity of polygons and multi-polygons, as holes fundamentally alter spatial containment relationships and require sophisticated geometric processing that can handle multi-ring polygon structures commonly found in administrative boundaries and land use classifications.

To make benchmark requirements for vector data more adaptable to real-world spatial data, we expand the requirements to include GeoNames data (**BM5**), OpenStreetMap 2D building data (**BM6**), and CityGML 3D building data (**BM7**).

6.2.2 Different Areas of Interest (BM8)

In practical applications, both vector and raster datasets are inherently associated with specific areas of interest (AOIs), such as cities, lakes, countries, continents, or user-defined regions. Benchmark BM8 evaluates the impact of AOI variation on query processing performance within the VKG setting. By integrating vector datasets of varying structures with raster datasets of varying resolutions across multiple AOIs, this benchmark enables a systematic assessment of VKG robustness across heterogeneous spatial contexts.

6.2.3 Raster Data Benchmarks (BM9–BM10)

AOI variation is particularly relevant for raster-oriented benchmarks (BM9–BM10), as raster processing cost generally scales with the number of cells involved. Raster benchmarks (BM9–BM10) extend the evaluation framework to multidimensional, grid-based raster data within VKG systems.

BM9 (Spatial Resolution Variation) This benchmark analyses the effect of changing spatial resolution on system performance. Spatial resolution directly influences memory requirements, processing time, and the granularity of spatial analysis. While higher resolutions enable more detailed representations and finer analytical precision, they substantially increase computational cost due to the growth in raster cell count and the complexity of raster algebra operations.

BM10 (Temporal Resolution Variation) BM10 investigates the influence of temporal granularity on query execution. Many geospatial phenomena exhibit dynamic behaviour, requiring the management of time-series raster data. This benchmark evaluates the VKG system's ability to handle temporal indexing, execute spatial-temporal queries, and perform reasoning, thereby assessing performance as temporal resolution and data volume increase.

The combined variation of AOI extent (BM8) and spatial-temporal resolution (BM9-BM10) generates a spectrum of scenarios, ranging from high-resolution analysis over small regions to coarse-resolution processing at the continental scale, which reflects realistic raster-based VKG workloads. Furthermore, evaluating spatial-temporal raster queries across different AOIs assesses the system's capacity to execute spatial-temporal operations consistently under varying spatial extents.

6.3 Synthetic Data Generation

We engineered novel data generators to produce OGC-compliant vector and raster data, each exhibiting distinct features that reflect real-world conditions, enabling evaluation of VKG systems against the specified benchmarks. We also ensure that these newly generated data comply with the semantics of the respective ontologies and the mappings in the geospatial domain as supported by OntoRaster. A random seed has been initialised across all relevant frameworks and libraries used in the benchmarking experiment to ensure reproducibility.

6.3.1 Vector Data

The workflow for generating OGC-compliant vector data (all possible types of polygons and multi-polygons) begins by defining an area of interest (AOI). The spatial extent of the AOI is determined by a *minimum bounding rectangle* (MBR) also known as a *bounding box* (bbox) specified by four corner points designated by a combination of geographic coordinates, i.e., min longitude, min latitude, max longitude, max latitude. In this research, all vector geometries are generated with reference to the AOI's bounding box, ensuring consistency and comparability across geometries with different configurations. This makes our data-generation method widely applicable to any AOI.

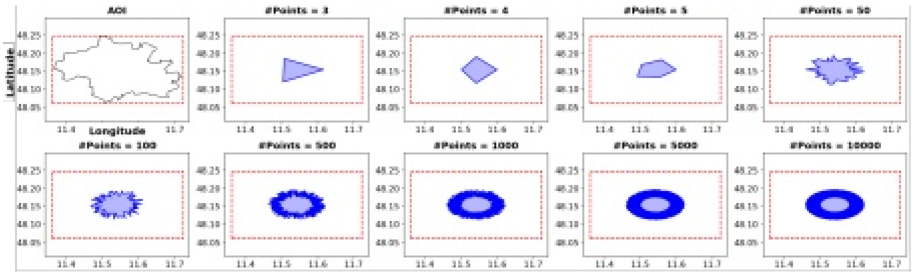


Figure 6.1: Generated polygons with increasing number of points (BM1)

Figure 6.1 depicts the synthetically generated polygons about Munich (AOI), arranged in a grid of subplots, each representing a different number of points, while other parameters, i.e., area, uniformity, are fixed under BM1. The AOI is shown in the initial subplot as a reference for spatial context.

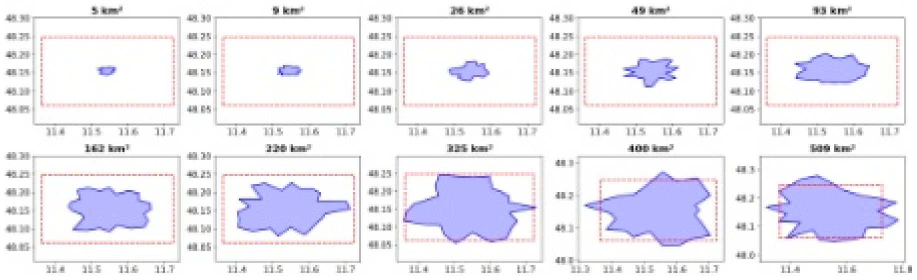


Figure 6.2: Polygons with increasing area (in km^2) - BM2

Subsequent subplots illustrate the evolution from simple polygons, such as triangles (3 points) and quadrilaterals (4 points), to increasingly intricate shapes that appear circular as the number of points increases. As the point density increases to 500-1,000 points, the polygon edges exhibit distinctive spiky, star-like characteristics, with high-frequency oscillations surrounding the centre. At elevated point densities (5,000-10,000 points), the polygon boundaries coalesce into a smooth, circular shape with minimal differences between spikes, which likely signifies the optimal approximation of the AOI. The bounding box (red dashed line) is uniformly illustrated in each subplot to enable spatial comparison. The generation process starts by calculating the centre point, or *centroid*, of the AOI's bounding box, which serves as the reference point for all polygons and their point placement. For each specified number of points, ranging from as few as three to as many as ten thousand, points are distributed uniformly around the centroid. This is achieved by computing equal angular measures and placing each point at a corresponding angle, thereby forming a regular polygon. To introduce a degree of natural variability and avoid perfectly regular shapes, small random perturbations are applied to the radius of each point. However, these perturbations are carefully controlled to ensure that the overall area remains consistent and that all points remain within the bounding box. Once the points are determined, the polygon is constructed and represented in WKT format, a standard for encoding geometric objects. Each polygon undergoes a validation check to ensure compliance with OGC standards, which require that it be simple, valid, and correctly oriented. If any issues are detected, such as incorrect ring orientation or self-intersections, automated correction procedures are applied to restore compliance. Figure 6.2 illustrates the synthetically generated polygons with varying area (in km^2) while the number of points and uniformity are fixed (BM2). We have implemented a function in the data generator that generates polygons with areas measured in km^2. We note this because all the considered geospatial data (both vector and raster) use EPSG:4326 as the coordinate reference system (CRS), which supports area measurements in degrees. This is not significant for standard area calculation, given that the Earth is a sphere. Projecting geographic coordinates to a planar coordinate system is necessary for accurate area calculation because angular

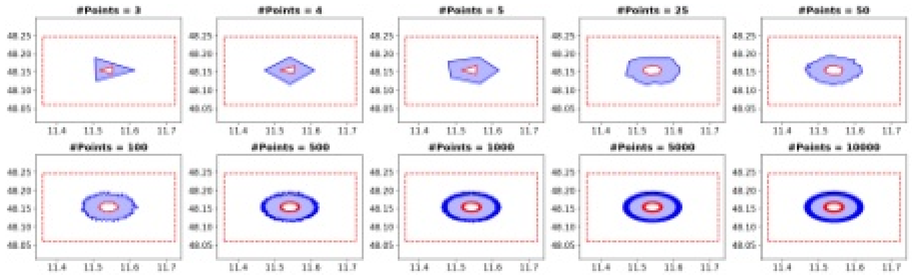


Figure 6.3: Polygons with increasing number of points and single hole - BM4

measurements on a spheroidal Earth are distorted. Here we rely on another coordinate reference system named Universal Transverse Mercator (UTM), which splits the globe into sixty north-south zones, each being six degrees (6°) wide in longitude [175]. In each zone, coordinates are measured in meters. UTM zones are numbered one to sixty consecutively, starting with Zone 1, which covers longitude 180° to 174°W; increasing eastward to Zone 60, which covers longitude 174°E to 180°. Our function automatically determines the UTM zone based on the AOI's centroid (zone 33 for Munich) and generates the polygon in UTM projection. Ultimately, the function converts the generated polygon from UTM to EPSG:4326 geographic coordinates and outputs the WKT version. Figure 6.3 illustrates the synthetically generated polygons with a single hole and varying numbers of points, while area and uniformity are fixed (BM4).

6.3.2 Raster Data

Figure 6.4 illustrates the synthetically generated raster data with reference to the bounding box of Munich (AOI) with varying spatial resolution (2000 meters to 50 meters per pixel). We calculated the spatial distance for each raster pixel in UTM coordinates. Note that the generated raster data do not reflect any particular continuous phenomenon (e.g, temperature, rainfall) over the earth. So, we relied on four distinct geospatial raster datasets, as displayed in 5.3, including their data type, satellite sensors, feature unit, feature count, and spatial and temporal resolutions, to benchmark OntoRaster, as shown in Table 6.2.

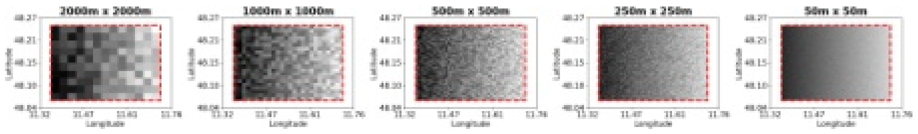
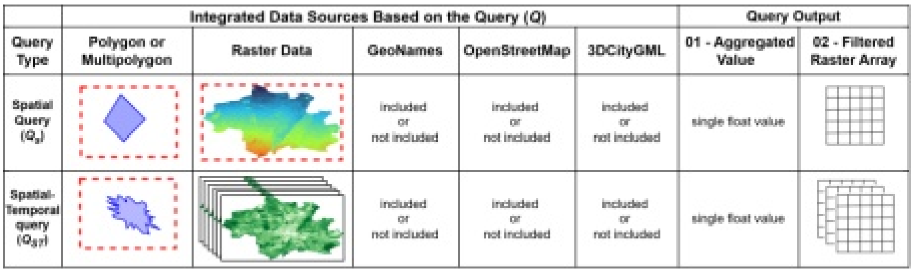


Figure 6.4: Generated raster data with varying spatial resolution

Table 6.2: Munich Raster Data Used in Benchmarks

Data Type	Data Field	Source and Sensors	Feature Unit	Feature Count	Spatial Resolution	Temporal Resolution
Raster (R1)	Elevation	SRTM	PIXEL	873010	30m x 30m	-
Raster (R2)	NDVI Vegetation	MODIS	PIXEL	15925	250m x 250m	16 days
Raster (R3)	Snow Cover	NDSI	PIXEL	4048	500m x 500m	daily
Raster (R4)	Land Temperature	MODIS	PIXEL	1012	1000m x 1000m	daily



	Integrated Data Sources Based on the Query (Q)					Query Output	
Query Type	Polygon or Multipolygon	Raster Data	GeoNames	OpenStreetMap	3DCityGML	01 - Aggregated Value	02 - Filtered Raster Array
Spatial Query (Q_S)			included or not included	included or not included	included or not included	single float value	
Spatial-Temporal query (Q_{ST})			included or not included	included or not included	included or not included	single float value	

Figure 6.5: Spatial and Spatial Temporal Queries

6.4 Experiments

Here we describe the experimental benchmark to test the extended VKG framework, OntoRaster in the context of query answering over raster and vector data over Munich (running example AOI). Data generators are used to generate different vector geometries, e.g., polygons and multipolygons, with an increasing number of points while holding the area (km^2), uniformity, and the number of inner rings (or holes) fixed. To test the scalability of VKG systems with respect to growing vector datasets (polygons and multipolygons), the benchmark scripts run each benchmark by varying a single parameter in the RasSPARQL queries while keeping the others fixed.

6.4.1 Benchmark Queries

Spatial queries require at least one raster dataset and a geometry (polygon or multipolygon) in OGC WKT format, subsequently cropping the input raster dataset to conform to the spatial shape of the specified geometry. The resulting raster data can be spatially aggregated to a single value (*output1*) or preserved as a filtered array (*output2*), as illustrated in Figure 6.5. On the other hand, spatial-temporal queries accept several time slices of raster data over a specified time frame. These queries generate either temporally aggregated values or collections of filtered raster arrays conforming to the input geometry's spatial bounds, as demonstrated in Figure 6.5. The first query ($\boldsymbol{Q}_{BM1}$) integrate synthetically generated geometry i.e., `syntheticWkt` and temperature raster data (R4) over Munich. This query can also be extended with different properties of vector data as mentioned in the section 6.2, which results in $\boldsymbol{Q}_{BM2}$, $\boldsymbol{Q}_{BM3}$,

$\boldsymbol{Q}_{BM4}$ and $\boldsymbol{Q}_{BM8}$

$\boldsymbol{Q}_{BM1}$: Find spatial average temperature for synthetically generated geometry over Munich.

```
SELECT ?answer {
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
 BIND ('{syntheticWkt}'^^geo:wktLiteral AS ?regionWkt)
 BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS
       ?answer)
}
```

To enable more comprehensive benchmarking, such as BM5, BM6, and BM7, we enriched spatial and spatial-temporal queries with GeoNames data, OSM 2D Building data, and CityGML 3D data, which results in $\boldsymbol{Q}_{BM5}$, $\boldsymbol{Q}_{BM6}$, and $\boldsymbol{Q}_{BM7}$. The spatial query $\boldsymbol{Q}_{BM5}$ utilises `S.RSTN` ("railroad station"), one of 680+ GeoNames features[1] (e.g., `S.UNIV, H.LKS, S.MTRO etc.`) as an additional input to integrate GeoNames with raster data w.r.t. respective ontologies.

$\boldsymbol{Q}_{BM5}$: Find spatial average temperature for synthetically generated geometry in Munich (AOI) and all railroad stations lying within it.

```
SELECT ?featureName ?answer {
 ?gname a gn:S.RSTN ; # or S.UNIV, H.LKS etc., 680+ GeoNames classes
 ?gname rdfs:label ?featureName ; geo:asWKT ?featureWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND ('{syntheticWkt}'^^geo:wktLiteral AS ?regionWkt)
 FILTER (geof:sfWithin(?featureWkt,?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
 BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS
       ?answer)
}
```

Similarly, spatial query $\boldsymbol{Q}_{BM6}$ takes `residential` as an additional input, which comprises 100+ OSM building types (e.g. `Hospital, Schools, Industrial, Townhall`) to join OpenStreetMap building data with raster data conforming to respective ontologies.

$\boldsymbol{Q}_{BM6}$: Find spatial average temperature for synthetically generated geometry in Munich (AOI) and all residential buildings lying within it.

```
SELECT ?bldgName ?answer {
 ?building a lgdo:Residential .#or Hospital, School.., 100+ OSM
 ?building rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND ('{syntheticWkt}'^^geo:wktLiteral AS ?regionWkt)
 FILTER (geof:sfWithin(?bldgWkt,?regionWkt))
```

[1] https://www.geonames.org/export/codes.html

```
    FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS
          ?answer)
   }
```

Q$_{BM10}$: Find monthly average temperature for synthetically generated vector geometry over Munich.

```
   SELECT ?answer {
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Munich_MODIS_Temperature_1km'))
    BIND ('{syntheticWkt}'^^geo:wktLiteral AS ?regionWkt)
    BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
    BIND ('2022-02-01T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
    BIND (rasdb:rasTemporalAverage(?startTimeStamp, ?endTimeStamp,
          ?regionWkt, ?rasterName) AS ?answer)
    }
```

6.5 Performance

We developed Python scripts for each defined benchmark to be executed on the Docker implementation of OntoRaster[2], which handles raster, relational, and vector data and relies on PostgreSQL and RasDaMan as the underlying RDBMS and array DBMS. The Docker implementation includes all the essential raster functionality and supports incremental integration of multiple data sources while maintaining portability. The operating system is based on Ubuntu 20.04 LTS and leverages the full hardware capacity, comprising an x64 architecture with an 11th Gen Intel(R) Core(TM) i7-1185G7 CPU (4 cores @3.47GHz), 32 GB of RAM, and a 1 TB NVMe WDC SSD.

6.5.1 Result

The benchmarks were executed using spatial aggregated queries that produce aggregated outputs (e.g., statistical averages, minimum, and maximum). This establishes a controlled performance baseline, as aggregation returns a single result size, whereas queries that return filtered raster subsets (e.g., clipped arrays) inherently require additional processing, memory allocation, and data serialisation, leading to longer execution times. Moreover, the performance characteristics of such non-aggregated queries are strongly dependent on specific application requirements, including output resolution, spatial extent, and downstream processing needs. By focusing on aggregated queries, the evaluation ensures comparability across experiments and captures the system's core computational behaviour.

[2]https://github.com/aghoshpro/OntoRaster.git

Figures 6.6 – 6.10 show bar diagrams to portray the performance of the extended VKG framework OntoRaster's query answering over raster and vector data query answering over varying benchmark conditions. Each figure shows the number of polygon points on the x-axis, with query processing time, measured in seconds, plotted on the y-axis using a logarithmic scale. This enables a comprehensive understanding of the computational demands of raster data processing under VKG.

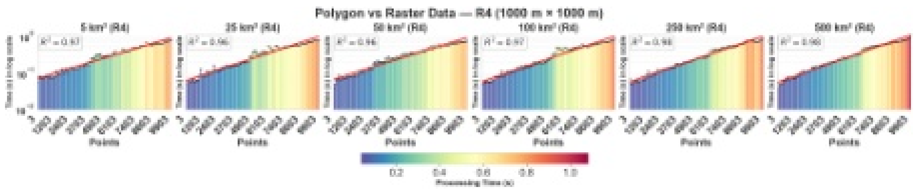


Figure 6.6: Polygon with varying number of points vs raster

Figure 6.6 demonstrates the direct relationship between polygonal complexity and query processing time for spatial operations executed over the temperature raster dataset (R4). As the number of polygon vertices increases from 3 (simple shape polygon) to approximately 10,000 (highly detailed polygon), a clear and consistent increase in query processing time is observed across all evaluated polygon areas, ranging from 5 km^2 to 500 km^2. This means that spatial queries take longer to execute when polygons become more geometrically detailed, even if their overall area remains the same. In practical terms, polygons with more vertices require more computational effort to evaluate their boundaries against the raster grid. Although larger polygons (e.g., 250–500 km^2) generally show slightly higher processing times than smaller ones (e.g., 5–25 km^2), the overall trend is similar for all area categories. This indicates that polygon complexity (number of points) has a stronger impact on query performance than polygon size (area) for raster R4. In other words, increasing the number of vertices consistently slows down processing, regardless of how large the polygon is. The fitted regression lines closely follow the measured values, and the high R^2 values (0.96–0.98) indicate that the number of polygon points accounts for most of the variation in processing time. The relatively small error bars further indicate that the measurements are stable and reproducible. Overall, spatial query performance is primarily driven by polygon geometry complexity rather than polygon area for coarse resolution raster data such as R4. This highlights the importance of simplifying polygon boundaries when performing spatial queries over raster datasets, especially in workflows involving large or highly detailed vector geometries.

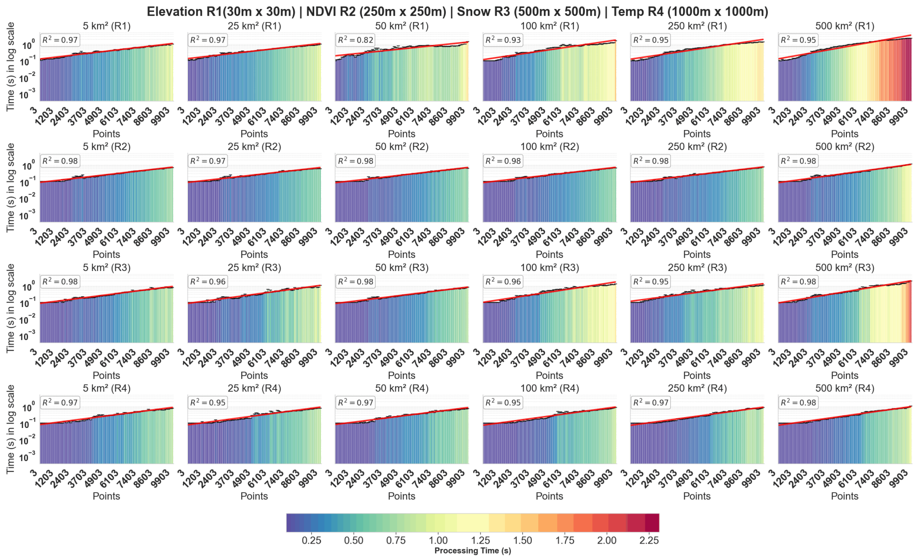


Figure 6.7: Polygon vs different rasters

Figure 6.7 compares the effect of polygon complexity on spatial query processing time across four raster datasets as listed in Table 6.2, with different spatial resolutions: Elevation (R1, 30 m), NDVI (R2, 250 m), Snow (R3, 500 m), and Temperature (R4, 1000 m). For each raster, polygons with areas ranging from 5 km^2 to 500 km^2 are evaluated while increasing the number of polygon vertices from 3 to approximately 10,000. Polygon complexity, expressed as the number of vertices, is the primary driver of processing time across all experiments: for every raster and polygon area, execution time increases steadily as the number of points grows from simple to highly detailed geometries. In contrast, raster resolution primarily affects the overall processing time rather than the scaling behaviour. Finer-resolution rasters (e.g., R1 at 30 m) consistently exhibit higher processing times than coarser rasters (e.g., R3 at 500 m and R4 at 1000 m) because more raster cells must be evaluated per query. However, the similar slopes and high R^2 values across all rasters indicate that resolution does not alter the fundamental relationship between polygon complexity and query cost. Thus, while raster resolution determines the absolute cost of a query, polygon complexity determines how rapidly that cost increases as geometries become more detailed. Overall, polygon complexity and area are the dominant factors affecting spatial query performance, as they determine the amount of raster data cropped out, whereas spatial resolution primarily scales execution time. To rigorously evaluate OntoRaster under realistic, data-intensive conditions, we incorporated high-density feature classes from real-world vector datasets across the selected AOIs, including GeoNames point features, OSM 2D building footprints, and CityGML 3D building data. For GeoNames, railroad stations (i.e., `S.RSTN`) were chosen because they exhibit

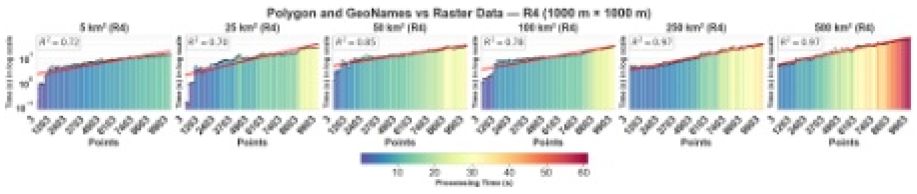


Figure 6.8: Polygon encompassing GeoNames data vs raster data

the highest frequency of occurrence in Munich compared to other feature types such as `universities`, `lakes`, or `mountains`.

Figure 6.8 illustrates the relationship between polygon complexity and query processing time when spatial queries (i.e., $\boldsymbol{Q}_{BM5}$) integrate GeoNames point data with raster dataset R4. Across all polygon areas, increasing geometric complexity leads to a gradual increase in execution time, confirming a direct dependence of processing cost on vertex count. As the polygon area expands from 5 km^2 to 500 km^2, the number of enclosed railroad stations grows proportionally, introducing an additional workload component. This results in a steady upward shift in absolute processing times. While smaller polygons (area of 5–25 km^2) show greater dispersion and comparatively lower R^2 values—likely due to localised variations in point distribution—larger extents (250–500 km^2) exhibit stronger linear trends and improved goodness-of-fit. Overall, the results demonstrate that polygon complexity remains the dominant driver of performance, whereas the increasing number of intersecting GeoNames features primarily affects the magnitude of execution time. For the OSM dataset,

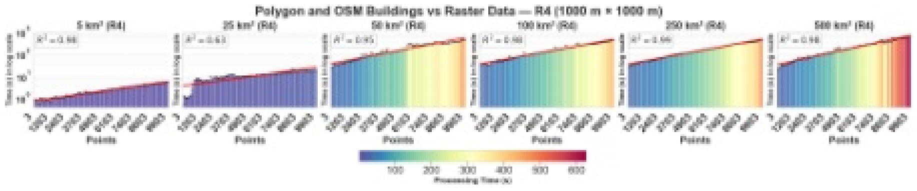


Figure 6.9: Polygon encompassing the OSM buildings vs raster data

`residential` buildings constitute the numerically dominant class, substantially exceeding other categories such as `schools`, `hospitals`, and `industrial`, and are better suited for street testing OntoRaster. Figure 6.9 depicts the relationship between polygon complexity and spatial query processing time when polygons containing OSM building data are evaluated against the temperature raster dataset R4. Although the first subplot (5 km^2) appears visually sparse, this effect is caused by the uniform logarithmic y-axis scale applied across all panels; the underlying measurements are present and follow the same general trend. As polygon area increases from 5 km^2 to 500 km^2, the number of enclosed `residential` buildings grows proportionally, leading to a systematic rise in absolute processing time. Across nearly all polygon areas (with the exception of the 25 km^2 case, which exhibits greater dispersion and a lower R^2 value), the regression results indicate a strong linear relationship between vertex count and query time on a logarithmic scale. This demonstrates that geometric complexity remains the primary determinant of performance, while the increasing volume of embedded building features contributes an additive cost component. Overall, the results confirm that when raster queries are combined with dense vector features such as OSM buildings, execution time scales predictably with polygon complexity and spatial extent. The dominant influence of vertex count persists even in the presence of large numbers of intersecting `residential` objects, indicating robust and consistent system behaviour for integrated raster–vector workloads.

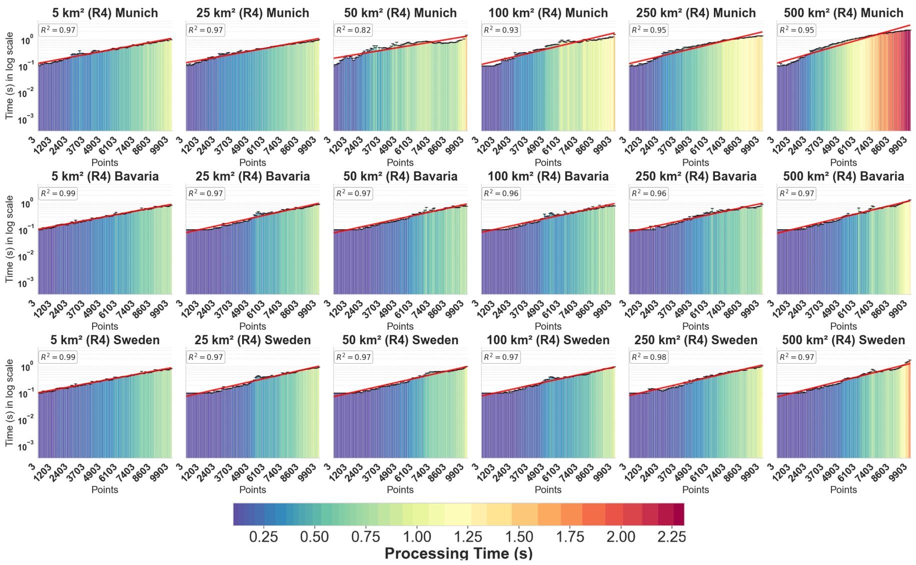


Figure 6.10: Polygon Vs temperature raster data (R4) over Munich (urban), Bavaria (state) and Sweden (country)

Figure 6.10 presents spatial query processing times for increasing polygon complexity across three areas of interest (AOIs): Munich, Bavaria, and Sweden over the temperature raster dataset (R4). Polygon complexity is defined by the number of vertices and evaluated over the polygon with an area of 5–500 km^2. Processing time is shown on a logarithmic scale. Across all AOIs, scaling behaviour is highly consistent despite differences in spatial extent and geographic structure. Munich, representing a compact urban setting, shows slightly greater variability for some intermediate polygon area (e.g., 50 km^2), reflected in marginally lower R^2 values. Bavaria and Sweden exhibit more uniform trends across all polygonal areas, indicating stable performance in larger, more heterogeneous regions. These variations primarily influence measurement dispersion, not the fundamental relationship between complexity and execution time. Within each AOI, increasing polygon area leads to a moderate rise in absolute processing time; however, its effect is secondary to geometric complexity. The near-parallel regression lines across AOIs and polygon areas demonstrate that performance degradation scales consistently with vertex count, independent of whether queries operate at local, regional, or national extents. Overall, the results confirm that polygon complexity is the principal driver of raster-based spatial query performance.

6.6 Conclusions

This study presents a comprehensive performance evaluation of specialised VKG systems designed to execute queries over heterogeneous raster data integrated with diverse relational and vector datasets. To enable a systematic and reproducible assessment, we define ten benchmarking criteria (BM1–BM10) that encompass key characteristics of both raster and vector data. A parameter-isolation methodology is employed, where individual parameters are varied independently while all other parameters are held constant. Using this approach, we evaluate the performance of OntoRaster across varying raster properties, including spatial resolution, temporal resolution, pixel density, and spatial coverage. Correspondingly, vector data properties such as point density, spatial extent, and structural complexity are analysed to assess their impact on system performance. Furthermore, the evaluation incorporates complex real-world vector datasets, notably GeoNames and OpenStreetMap, to reflect practical usage scenarios. Benchmarking CityGML data combined with raster data is considered in this chapter, but the results are not included in the thesis due to incomplete experimentation. Synthetic data generators are developed to produce OGC-compliant polygon and multipolygon geometries that are consistent with the underlying ontologies. The logarithmic scale is employed for query processing time (y-axis) to compress wide-ranging values and highlight proportional increases and exponential trends in computational overhead, enabling clear discernment of performance patterns across magnitudes that a linear scale would obscure.

Figure 6.6 shows, for the temperature raster (R4), query processing time is increasing with vertex count, from 10^{-2} seconds at 3 points to 10^{0} seconds at 10,000 points across 5–500 km^{2} areas, with strong linearity (R^2 = 0.96–0.98). Figure 6.7 depicts query processing time over four different raster data sets, indicating amplification with finer spatial resolution, with elevation (R1, 30 m) raster data reaching 10^{0} seconds compared with coarser snow (R3, 500 m) and temperature data for complex polygons. Figure 6.8 depicts a result bar chart for RasSPARQL query processing time with GeoNames point data, attaining 10^{0} seconds for 5 km^{2} polygons but 10^{1} seconds for 500 km^{2} polygons. Figure 6.9 shows OSM buildings yielding processing time 10^{0} seconds for 5 km^{2} polygons but 10^{3} seconds for 500 km^{2} polygons as the number of OSM features increased along with polygon area. Figure 6.10 confirms consistent trends across different AOIs, Munich, Bavaria, and Sweden, for similar complexities. These findings underscore the need for vector-geometry optimisation to enhance VKG scalability in industrial geospatial contexts and to accommodate heterogeneous data.

Due to the absence of standardised benchmarks specifically designed for combined raster–vector scenarios, we designed a custom evaluation framework grounded in realistic use cases and informed by requirements from geospatial domain experts. In contrast to traditional synthetic benchmarks, our evaluation emphasises practical applicability by using real-world datasets. However, a limitation of this approach is that it does not explicitly evaluate scalability as data volumes increase, a dimension typically addressed by synthetic benchmarks through controlled, repeatable experiments. Future work will apply the proposed benchmarks to other VKG systems supporting raster data, such as Plato [164], as well as to materialised knowledge graph (MKG) systems, including GeoSPARQL+ and GeoLD, enabling broader and more robust comparative evaluations against the results obtained with OntoRaster.

Chapter 7

Semantic Enrichment of Location-Based Business Intelligence

This chapter is based on our paper [86]. Data-driven businesses seek assurance that their information is streamlined, production-ready, and trustworthy before using it to understand key business factors and make informed decisions at any given time [79]. *Business Intelligence* (BI), primarily facilitates storing and disseminating an organisation's data by relying on a suite of decision support tools designed to empower people such as executives, managers, analysts, etc., to enhance the quality and speed of their integrated decision-making across all aspects [58]. BI has traditionally focused on analysing tabular data with attributed numerical values and is now rapidly expanding to include other data types, such as graph and array data [64]. Nowadays, data are generated relentlessly from various sources in diverse representations at massive scale, leading to bottlenecks in data management and processing and giving rise to the notorious issue of data heterogeneity [6].

Effective BI over heterogeneous data sources (both structured and unstructured) requires interconnecting the data in a semantically coherent manner so that the data can be queried and analysed in a uniform way to extract business insights that aid informed decision-making [64]. However, current data management mechanisms do not natively support all data formats; for instance, geospatial raster data represented as multidimensional arrays poses a significant challenge in the BI domain.

Uniformly querying and integrating these diverse types of data requires practical geospatial data management expertise and extensive domain knowledge, which may not be readily available to business managers, analysts, or policymakers [86]. This hinders the resolution of contemporary issues in large-scale geo data applications, including Earth Observation (EO), Geographic

Information Systems/Building Information Modelling (GIS/BIM) integration, and 3D/4D urban planning [44]. Moreover, it involves visualising query results as vectors and rasters to facilitate the perpetual accessibility and reproducibility of geo-visual analytics, thereby enabling the generation of location-based intelligence.

7.1 Related Works

Laborie et al. [142] described two possible approaches to combine the semantic web and BI: *(i)* the analysis-oriented and *(ii)* the modelling-oriented approach. In the former case, conventional BI relies on popular materialisation approaches, such as Extract-Transform-Load (ETL) and Extract-Load-Transform (ELT), to prepare heterogeneous datasets for analysis, with the transformed data typically stored in a relational data warehouse [12]. Consequently, the entire dataset is re-materialised to incorporate recent external changes, incurring additional overhead. It becomes more challenging in the context of heterogeneous geospatial data, including large raster data (e.g., satellite images) and vector data (e.g., geometrical). In a recent work, Kyriakos et al. [141] discuss business insight generation and policymaking by utilising multidimensional satellite imagery and machine learning to detect unplanned urbanisation, which contributes to the financial crisis and impedes economic growth. A prevalent method for accessing and analysing these diverse data sources is to develop ad hoc scripts in Python, R, or MATLAB; however, this requires extensive domain knowledge and proficiency across many tools at various levels. Furthermore, whenever the data sources change, the entire script may need only minimal adjustment or, in the worst case, be completely rewritten.

The latter entails conducting the analysis directly on the Linked Data without prior ETL. This strategy appears more effective, but it requires an advanced conceptual representation of the data (using a domain ontology). In the literature, this issue has been addressed only to a limited extent in the VKG settings and their application to BI, which could benefit both the GIS and Semantic Web communities. GeoKGs bridge Symbolic AI (logic-based) and Connectionist AI (machine learning), as demonstrated in studies employing KG embeddings to learn spatial/temporal rules or neural networks for advanced spatial relations [240, 160]. This facilitates the development of a neurosymbolic Geospatial AI (GeoAI), enabling the integration of theory and data to ethically guide AI systems in processing geospatial data [160]. GeoAI combines symbolic approaches—such as geospatial knowledge graphs for structured reasoning and data integration [121]—with sub-symbolic methods like deep learning for large-scale image analysis and pattern recognition [184, 154]. This synergy addresses limitations in traditional GIS, enabling spatially explicit models that handle complex data types, including remote sensing imagery and point clouds, while incorporating semantic interoperability [198]. By leveraging RDF-based ontologies to address vagueness and uncertainty in spatial relations, GeoAI enhances knowledge discovery, geographic question answering, and data fusion. Future research should focus on hybrid symbolic-sub-symbolic frameworks to improve interpretability and scalability in semantic web applications [204]. Most of these works mentioned in the literature are related to GIS/BIM utilising basic vector data, and none employ raster data in VKG settings.

7.2 Geospatial Business Intelligence (GeoBI)

GeoBI[1] integrates with established BI technologies [8], and is becoming a prominent driving force that empowers geographical location-based decision-making and product design in industrial organisations [103]. GeoBI integrates spatial analysis and map visualisation with established BI technologies to enhance corporate data analysis and facilitate more informed decision-making for enterprises [19]. The majority of GeoBI applications are based on relational data, including vector data (e.g., points, lines, and polygons)[56].

However, the inclusion of raster data is often overlooked due to its complex nature and the requirement of specialised domain expertise. Traditional GIS-based tools, such as ArcGIS[2] and QGIS[3], provide robust tools for working with geospatial data in geographic applications. Still, geospatial data and processing are no longer limited to those products. Thanks to OGC standards, geospatial data has become a key enabler for many mass-market applications, including web-based mapping and location-based services.

7.2.1 Raster Data in GeoBI

As mentioned earlier, raster data is also referred to as a *data cube* or a *hyper-cube*, a term that originates from the BI domain in the 1990s, where it is referred to as Online Analytical Processing (OLAP) cube [212, 105]. Generally, it refers to an array of multiple dimensions utilised to aggregate statistical measurements, including the mean, variance, and median, across combinations of values across several dimensions, which may be hierarchical in nature [92]. Later, Spatial-OLAP (SOLAP) was introduced by adding spatial features and dimensions to support vector data within the OLAP cube [102]. GIS experts in the Earth Observation (EO) domain use *datacube* to refer to large analysis-ready raster data sets retrieved from satellite sensors such as Landsat and Sentinel [212]. State-of-the-art EO data cubes have long been used to facilitate the representation of geospatial data, making it easier to access, analyse, visualise, and distribute analysis-ready data (ARD) [83]. The *Open Data Cube*[4] (ODC) [133] is a free and open-source initiative that specifies best practices for

[1]https://www.ogc.org/about-ogc/domains/geobi/
[2]https://www.esri.com/en-us/arcgis/geospatial-platform/overview
[3]https://qgis.org/
[4]https://www.opendatacube.org/

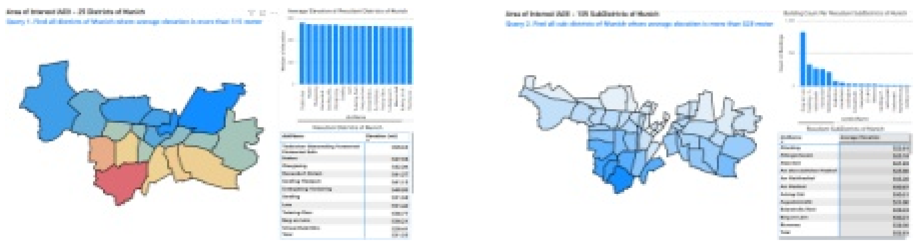
designing operational EO data cubes, hence improving the global utilisation of EO satellite data for the community to advance technology and its applications for societal benefit. [16].

The concept behind EO data cubes is similar to that of BI data cubes, though the two are not identical. EO data cubes are always densely populated, while BI data cubes are typically sparse, which limits the use of the OLAP approach in EO data cubes or geospatial raster data [24, 214, 29]. The complexity of raster data can be daunting for business analysts without adequate GIS expertise, hindering their ability to interpret it and derive location-based business intelligence. Therefore, we aim to establish a comprehensive data management pipeline to streamline the generation of location-based business insights from multidimensional raster data using knowledge graphs (virtual), requiring minimal user input. Note that Abad et al., [1] investigated the usage of *vector data cubes* for the structuring and analysis of features that evolve in space and time, with a special focus on geomorphological features due to their significant spatiotemporal variability.

7.3 Extending VKGs to GeoBI

Formulating SPARQL queries over a KG can be challenging, especially for business professionals who are not familiar with its syntax and semantics, and semantic web notions in general [135]. We have therefore experimented with different methods to support the use of BI tools in the VKG setting. Extending VKGs for GeoBI requires addressing integration barriers, such as the complexity of SPARQL for non-experts and handling diverse data types, including multi-polygons and high-resolution raster data [234, 44]. The enhanced OntoRaster framework supports complex vector geometries (e.g., multi-polygons with holes) via functions such as 'geo2grid_coords()', leveraging affine transformations and libraries such as Shapely, while incorporating public datasets such as GeoNames, OSM, and CityGML [229, 70, 138]. Multi-resolution raster data is managed natively, aligned with array database standards such as WCPS for server-side processing [25]. Geographic map visualisations of query results, including choropleths and heat maps, provide at-a-glance insights, as in GOdIVA's interactive interfaces for spatial-temporal analysis [68]. Integration with BI tools occurs via: (1) exporting RasSPARQL results as CSV/JSON for Power BI dashboards; (2) SQL-Interface (via JDBC), translating SQL to internal queries, reducing joins and extending for raster [142]. Recent advancements in geospatial data management, including GIS/BIM at the data-processing level and topology for spatial relations, inform these expansions, with future directions in GeoAI, streaming, and graph databases for real-time BI [44, 203]. This transforms VKGs into GeoBI facilitators, supporting applications such as urban planning, environmental resource management, and earth observation [141, 64].

We used YASGUI, an integrated SPARQL editor and result set visual-



(a) $Q1$ (b) $Q2$

Figure 7.1: Dashboard reports for $Q1$–$Q2$ in Ms Power BI

izer [190], developed by Triply[5] to accommodate OGC GeoSPARQL standard capabilities, which facilitate the querying of geospatial relationships, presenting them in a standard result set in tabular format with automatically rendered geographical map visualisation [30].

7.3.1 Method 1: OntoRaster + Power BI Tools

The OntoRaster framework yields two outputs: spatial-temporal aggregated values and filtered raster arrays. The first method is simple and essentially restricted to aggregated values, since handling filtered raster arrays (which are retrieved as RDF strings) requires further processing, depending on the use case, since RDF does not yet support the array data type. Hence, processing large arrays simply as strings would negatively impact performance. OntoRaster provides users with the option to export query results in two file formats: `.csv` and `.json`, which are widely used in the business domain. The exported `.csv` file containing the query results can be loaded into Power BI using the native built-in CSV connector. Similarly, if the output is in `.json` format, then it can be loaded into Power BI using a JSON connector. Fig. 7.1 illustrates the interactive dashboards based on the results of queries $Q1$–$Q2$, created with Power BI's supported map visualisations: *shape map*, *filled map*, and *world map*. Business professionals can interact with these dashboards to evaluate RasSPARQL query results based on VKGs generated from combined relational and raster data via mappings.

7.3.2 Method 2: Ontopic Suite's Semantic SQL Interface and BI Tools.

Ontopic Suite[6] is an intuitive, no-code environment that connects to cloud or on-premises data and provides a user-friendly interface for designing declarative mappings of a VKG, thereby exposing the underlying legacy data. In general,

[5]https://yasgui.triply.cc/
[6]https://docs.ontopic.ai/suite/

VKGs cannot be queried directly from BI tools, since such tools expect a traditional relational interface and issue SQL queries directly (e.g., via JDBC). To overcome this restriction while still leveraging the KG's semantic abstraction layer, Ontopic Suite can expose the KG via its *Semantic SQL Interface* component as a set of relational tables that can be queried using traditional SQL code in BI tools. Such tables are organised to support efficient query evaluation with fewer relational joins. e.g., when data properties are relevant for the instances of a class C, they are grouped as attributes in a single table, whose primary key is given by an attribute containing the IRIs of C. In this way, one can avoid numerous costly joins whenever the query needs to return instances of the class, along with (some or all) of their data properties, which is a common request in a BI scenario. A SQL query issued via Ontopic Suite's Semantic SQL Interface is translated into Ontop's internal *intermediate query* (IQ) representation, a uniform format for processing queries over the KG. From that moment onwards, queries are processed by Ontop in the same manner as SPARQL queries issued directly over the VKG. We are currently extending the SQL Interface component to support raster query functionality, enabling corresponding requests to be issued directly from BI tools. Such extended function calls are recognised internally by OntoRaster and processed in the same way as those appearing in RasSPARQL queries. We observe that the Semantic SQL Interface is a proprietary software component commercialised by Ontopic and, therefore, its extension to raster functionality cannot be released as open source.

7.3.3 Method 3: Natural Language-to-SPARQL query by LLMs.

Large Language Models (LLMs) have shown significant potential for both spatial analysis and KG-based solutions separately. However, their synergistic integration within the Virtual Knowledge Graph (VKG) paradigm remains largely unexplored, particularly for querying geospatial raster data, where complex raster functionality must be seamlessly integrated with semantic queries generated from natural language. To address this challenge, we proposed an experimental method, a Retrieval-Augmented Generation (RAG)-based approach, that leverages LLMs to generate RasSPARQL queries with raster functionality from natural-language queries in VKG settings. Figure 7.2 displays the proposed RAG implementation, which employs a systematic data preprocessing pipeline in which ontologies and RasSPARQL query syntaxes undergo format standardisation through a document abstraction layer of LangChain[7]. This transformation encapsulates comprehensive textual information while preserving both primary content and associated metadata. Automated segmentation is applied to each data object using configurable chunking algorithms that partition large-scale textual data into semantically coherent fragments.

[7]https://python.langchain.com/docs/introduction/

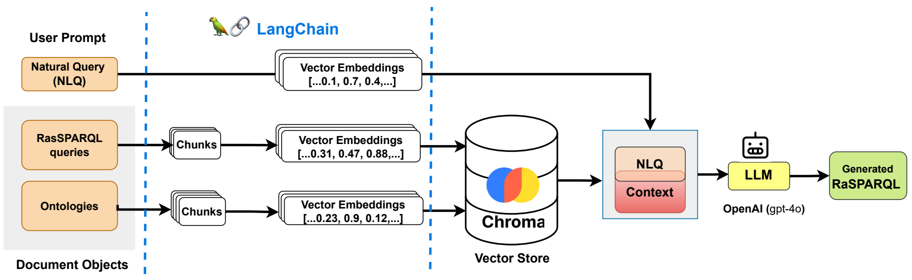


Figure 7.2: RAG Pipeline - Natural Language Query to RasSPARQL

This chunking strategy addresses the contextual window limitations inherent in transformer-based LLMs while optimising information density per retrieval unit, thereby minimising noise injection during the generation phase. Each document is transformed into a high-dimensional vector representation using pre-trained embedding models, thereby capturing semantic relationships within the latent space. These embeddings are subsequently indexed in ChromaDB, a specialised vector storage optimised for similarity search, laying the foundation for efficient similarity-based retrieval. Semantic similarity is computed using cosine similarity between query embeddings and stored document chunk embeddings. Relevant chunks are retrieved based on similarity scores using LangChain's *RetrievalQA* framework, providing contextually relevant information to augment the language model's response generation process (RasSPARQL).

7.4 Conclusion

We have outlined approaches to utilise semantic query results from OntoRaster within standard BI tools (e.g., PowerBI[8]) to produce semantically enriched business information through map-based visualisations, contributing to Geospatial Business Intelligence (GeoBI). Furthermore, we have presented Ontopic Suite's Semantic SQL interface to enable direct querying of KGs from BI tools, and we have proposed enhancing the SQL interface with raster functions to query integrated raster and relational data under OntoRaster. As a valuable addition given recent advancements in AI, we have developed an LLM-based RAG approach that translates natural-language queries into RasSPARQL queries, which can be executed on OntoRaster. RAG is effective at producing basic RasSPARQL queries from natural language; however, its performance tends to decline, leading to hallucination artefacts when generating structurally complex queries. This limitation stems from the inherent constraints of retrieval-based augmentation in capturing intricate ontological relationships and query composition patterns required for sophisticated seman-

[8]https://www.microsoft.com/en-us/power-platform/products/power-bi

tic query construction. In future, we plan to follow two LLM-based approaches, such as GraphRAG [73, 202] and Low-Rank Adaptation (LoRA) [117], a well-known Parameter-Efficient Fine-Tuning (PEFT) [162] technique to adapt pre-trained LLMs for the comprehension of ontology axioms and the synthesis of complex RasSPARQL queries suited for VKG specifications of OntoRaster. This approach enables the language model to internalise domain-specific semantic patterns and query-generation heuristics through supervised learning on curated training exemplars, with minimal changes to the model's parameters. Additionally, we intend to provide a more intuitive business dashboard for OntoRaster to facilitate direct and robust query formulation and to better visualise the resulting raster data in OGC formats such as GeoTIFF[9] and CovJSON[10] over OpenStreetMap.

[9]https://www.ogc.org/publications/standard/geotiff/
[10]https://www.ogc.org/publications/standard/coveragejson/

Chapter 8

Discussion

The work in this thesis broadens our understanding of multidimensional raster data management under the Virtual Knowledge Graph (VKG) paradigm. It introduces a novel VKG framework, OntoRaster, to support the management and querying of multidimensional raster data integrated with relational data (including vector data) and linked data, by leveraging VKG virtualisation across several ontologies and mappings. We conclude this thesis by summarising our key contributions by addressing the research questions posed in section 1.2. We also suggest potential directions for future research.

8.1 Key Contributions

This section discusses the primary contributions of the thesis. To ensure clarity and systemic organisation, we first list each research question and subsequently outline the specific contribution for each chapter.

RQ1: *What is an effective architecture for the integration of different data formats relevant to geospatial data, especially, raster data, general relational data, geometric vector data and linked data using the VKG paradigm?*

RQ2: *How can we devise effective query plans that guide the structuring and efficient processing of spatial queries over integrated data?*

- **Chapter 4** contributes to the resolution of two research questions **RQ1** and **RQ2** by presenting OntoRaster, a novel extended VKG framework to query for the first time multidimensional raster data combined with relational data, including vector data, on the fly by connecting the array DBMS RasDaMan to the VKG system Ontop. To achieve this, we defined the RasSPARQL language by extending SPARQL with custom raster functions to query over VKGs. We then developed a query transformation system that includes several stored procedures written in

PL/Python and PL/pgsql, thereby enabling PostgreSQL to act as a federator between the VKG engine Ontop and array DBMS RasDaMan to query heterogeneous raster data. It routes the translated SQL-SQL/MDA query from Ontop to the corresponding relational (RDBMS) and array data sources (RasDaMan), as specified in the RasSPARQL query, and executes them separately. After execution, the retrieved sub-answers (containing relational and raster arrays) are joined and then returned to Ontop to generate a virtual knowledge graph that answers the user's semantic query. The most notable contribution of this article lies in its utilisation of the array-handling capabilities of an array DBMS to perform queries on large raster datasets in their original data structure, i.e., multidimensional arrays, without conversion. This eliminates the need for costly data transfer and for raster data to be transformed into relational databases. OntoRaster supports a wide range of queries, limited only by the data-processing and functional capabilities of PostgreSQL (RDBMS) and RasDaMan (array DBMS) for managing relational and raster data, respectively.

RQ3: *How can we support the formulation of rich geospatial queries over the integrated datasets through intelligent, context-aware, reasoning-supported visual interfaces?*

- **Chapter 5** presented a set of spatial and spatiotemporal RasSPARQL queries executed on OntoRaster, which integrates multidimensional raster data with heterogeneous vector datasets, including OpenStreetMap (OSM) 2D building data, CityGML 3D building models, and GeoNames point feature data. The corresponding query results (tabular) and visualisations of the geo map were also demonstrated, with a detailed explanation. This answered research question **RQ3**.

RQ4: *How can the developed techniques and prototype system be rigorously evaluated using diverse, real-world datasets that vary in structural complexity, spatial distribution, and semantic density?*

- **Chapter 6** explored a solution to research question **RQ4** by providing several benchmarks for the performance evaluation of specialised VKG systems, i.e., OntoRaster that support query answering over multidimensional raster data combined with various vector data. To enable a thorough assessment of such systems, we define 10 benchmarking criteria (BM1-BM10). Utilizing a parameter isolation approach—where one parameter is varied while others are held constant—we systematically analyse performance with respect to different raster data attributes (including spatial and temporal resolution, pixel density, and coverage area), the geometric complexity of vector data (such as point density, coverage area, and structural uniformity), and the integration of public datasets such as GeoNames, OpenStreetMap. Synthetic data generators were developed to produce OGC-compliant geometries (polygons and multipolygons) aligned with relevant ontologies. Our research provided interesting insights, such as query processing times ranging from 0.1 to 600 seconds across all benchmarks, which are reasonable for handling both raster and vector data.

RQ5: *How can the developed techniques and prototype system be used in spatial business intelligence?*

- **Chapter 7** provide solution of research question **RQ5** by demonstrating BI application of OntoRaster. It outlined approaches to utilise semantic query results from OntoRaster within standard BI tools (e.g., Power BI) to produce semantically enriched business information through map-based visualisations, thereby contributing to Geospatial Business Intelligence (GeoBI). Furthermore, we have presented Ontopic Suite's Semantic SQL interface to enable direct querying of KGs from BI tools, and we have proposed enhancing the SQL interface with raster functions to query integrated raster and relational data under OntoRaster. As a valuable addition, we have developed an LLM-based RAG approach that translates natural language queries into RasSPARQL queries, ready for execution on OntoRaster.

8.2 Future Directions

Here, we outline several potential research directions that may provide useful avenues for extending our current work.

- **Support for Arbitrary Raster Data of arbitrary domains**

  A key direction is to extend the query processing capabilities of the OntoRaster framework to accommodate arbitrary raster datasets that originate from diverse application domains. These include, but are not limited to, medical imaging (e.g., fMRI, CT, and histopathological scans), bioinformatics data (e.g., microscopy images, spatial transcriptomics, etc.), industrial imaging (e.g., non-destructive testing, radiographs, thermal imaging, and hyperspectral quality control data, etc.), material science, as well as other scientific and industrial fields. By achieving this generalisation, our proposed VKG framework OntoRaster will attain greater robustness and completeness in handling general-purpose raster data processing. This directly aligns with the original motivation of the research: to establish a unified, domain-agnostic VKG framework that seamlessly integrates and queries across a wide range of real-world raster datasets, as well as relational (including vector) and linked data sources, irrespective of their thematic domain or underlying physical semantics.

- **Visual Explanations for Query Validation and Refinement**

A promising direction for future research is to investigate how obtained query results can be explained through visual geospatial information and how such explanations can, in turn, be used to validate and refine user queries. This line of work can build on the declarative nature of the VKG approach, which enables the extraction of explicit symbolic representations of how specific query results are produced via provenance information. By visualising relevant geospatial datasets, mapping assertions, and ontological axioms involved in the reasoning process, users can gain clearer insights into why particular results are returned, identify potential mismatches or errors, and iteratively adjust their queries to improve accuracy and expressiveness.

- **Systematic Robust Benchmarks for Performance Evaluation Raster-based VKGs**

  Beyond the current case-study–driven evaluation, future benchmarks should incorporate controlled scalability experiments in which all possible combinations of raster and vector features are varied independently to enable formal complexity analysis and bottleneck identification. In addition to real-world datasets such as CityGML data, parametrically generated synthetic raster data should be used for reproducible stress testing across different benchmarks. A standardised query suite covering spatial joins, zonal statistics, and semantic reasoning should be defined, with formal characterisation of spatial-temporal complexity to ensure fair comparison. Evaluation metrics should extend beyond query processing time to include memory usage, indexing overhead, update latency, concurrency, and distributed performance. Finally, an open, containerised benchmark suite would enhance reproducibility and foster community-wide comparability of such specialised VKG systems.

- **More robust query formulation support using LLMs**

  Although RasSPARQL offers expressive capabilities, creating correct and efficient queries remains challenging, especially for users lacking expertise in semantic technologies, ontologies, and geospatial raster data. The difficulty arises from the need to understand ontology structures, spatial functions, raster semantics, and query syntax simultaneously. Future work will explore integrating Large Language Models (LLMs) to facilitate more user-friendly and robust query creation. A key focus will be on developing natural language interfaces that convert user queries into syntactically valid and semantically accurate RasSPARQL queries, utilising domain-aware prompting and ontology-based reasoning to align with data schemas and spatial constraints. Instead of relying solely on general-purpose language models, more advanced Retrieval-Augmented Generation (RAG) techniques will be investigated that incorporate ontology definitions, schema metadata, and example queries as contextual knowledge. Additionally, ensuring query correctness and refinement will be prioritised, with LLMs used to identify logical errors, missing spatial relations, or incompatible raster–vector operations before execution. Query explainability features will also be integrated, helping users understand how natural language inputs are transformed into executable RasSPARQL queries, thereby increasing transparency and trust. Future plans include fine-tuning LLMs with domain-specific datasets—containing RasSPARQL queries, ontology structures, and geospatial reasoning examples—to improve spatial understanding, especially for spatiotemporal queries involving multidimensional raster data. Overall, leveraging LLMs for query generation seeks to lower barriers in semantic geospatial analysis while maintaining formal query accuracy.

8.3 Closing Remarks

This thesis has demonstrated that multidimensional raster data can be seamlessly integrated into the Virtual Knowledge Graph paradigm, transforming how heterogeneous geospatial and semantic data are accessed, queried, and analysed. By introducing RasSPARQL and developing a federated query transformation architecture under a novel VKG framework OntoRaster, we enabled efficient, on-the-fly querying across raster, relational, vector, and linked data without sacrificing native data structures or performance. This thesis also provides practical query demonstrations, performance benchmarking, and GeoBI applications, collectively validating OntoRaster's feasibility and real-world value, while the integration of LLM-based query generation represents a step toward more accessible semantic analytics. Beyond the current contributions, this work lays the foundation for semantic integration and query processing of raster data across arbitrary domains and different relational and linked data under the VKG paradigm. Future efforts may expand toward broader real-world, industrial datasets, explainable visual query validation, and more intelligent LLM-based query-formulation support. Together, these directions point toward a new generation of semantic data infrastructures where complex spatial-temporal knowledge becomes intuitively discoverable, interoperable, and actionable.

Bibliography

[1] L. Abad, M. Sudmanns, and D. Hölbling. "Vector Data Cubes For Features Evolving In Space And Time". In: *Proceedings of the 27th AGILE Conference on Geographic Information Science* 5 (2024), p. 16. DOI: 10.5194/agile-giss-5-16-2024.

[2] Russell L Ackoff. "From Data to Wisdom". In: *Journal of Applied Systems Analysis* 16.1 (1989), pp. 3–9. URL: https://cir.nii.ac.jp/crid/1573105976486097280.

[3] Somayeh Ahmadian and Parham Pahlavani. "Semantic Integration of OpenStreetMap and CityGML with Formal Concept Analysis". In: *Transactions in GIS* 26.8 (2022), pp. 3349–3373. DOI: 10.1111/tgis.13006.

[4] Andrei Aiordăchioaie and Peter Baumann. "PetaScope: An Open-Source Implementation of the OGC WCS Geo Service Standards Suite". In: *Scientific and Statistical Database Management*. Springer, 2010. ISBN: 978-3-642-13818-8. DOI: 10.1007/978-3-642-13818-8_13.

[5] Shahed Bassam Almobydeen, José RR Viqueira, and Manuel Lama. "GeoSPARQL Query Support for Scientific Raster Array Data". In: *Computers & Geosciences* (2022). DOI: 10.1016/j.cageo.2021.105023.

[6] Daichi Amagata, Takahiro Hara, and Shojiro Nishio. "Distributed Top-k query Processing on Multi-Dimensional Data with Keywords". In: *Proceedings of the 27th International Conference on Scientific and Statistical Database Management (ISSDM)*. 2015. DOI: 10.1145/2791347.2791355.

[7] Andrej Andrejev, Dimitar Misev, Peter Baumann, and Tore Risch. "Spatio-Temporal Gridded Data Processing on the Semantic Web". In: *IEEE Int. Conf. on Data Science and Data Intensive Systems*. 2015. DOI: 10.1109/DSDIS.2015.109.

[8] Michele Angelaccio, Berta Buttarazzi, Alessandra Basili, and Walter Liguori. "Using Geo-Business Intelligence to Improve Quality of Life". In: *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*. 2012. DOI: 10.1109/ESTEL.2012.6400196.

[9] Julián Arenas-Guerrero, María S. Pérez, and Óscar Corcho. "LUBM4OBDA: Benchmarking OBDA Systems with Inference and Meta Knowledge". In: *J. Web Eng.* 22.8 (2023), pp. 1163–1186. DOI: 10.13052/JWE1540-9589.2284.

[10] Ken Arroyo Ohori, Abdoulaye Diakité, Thomas Krijnen, Hugo Ledoux, and Jantien Stoter. "Processing BIM and GIS Models in Practice: Experiences and Recommendations from a GeoBIM Project in The Netherlands". In: *ISPRS International Journal of Geo-Information* 7.8 (2018). ISSN: 2220-9964. DOI: 10.3390/ijgi7080311.

[11] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. "The DL-Lite Family and Relations". In: *Journal of Artificial Intelligence Research* 36 (Oct. 2009), pp. 1–69. DOI: 10.1613/jair.2820.

[12] Martin Aruldoss, Miranda Lakshmi Travis, and V Prasanna Venkatesan. "A Survey On Recent Research In Business Intelligence". In: *Journal of Enterprise Information Management* 27.6 (2014), pp. 831–866. DOI: 10.1108/JEIM-06-2013-0029.

[13] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. "DBpedia: A Nucleus for a Web of Open Data". In: *The Semantic Web*. Springer, 2007, pp. 722–735. ISBN: 978-3-540-76298-0. DOI: 10.1007/978-3-540-76298-0_52.

[14] Sören Auer, Jens Lehmann, and Sebastian Hellmann. "LinkedGeoData: Adding a Spatial Dimension to the Web of Data". In: *The Semantic Web - ISWC 2009*. Springer, 2009, pp. 731–746. ISBN: 978-3-642-04930-9. DOI: 10.1007/978-3-642-04930-9_46.

[15] Sören Auer, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. "Introduction to Linked Data and Its Lifecycle on the Web". In: *Proceedings of the 7th International Conference on Reasoning Web: Semantic Technologies for the Web of Data*. RW'11. Galway, Ireland: Springer-Verlag, 2011, pp. 1–75. ISBN: 9783642230318. DOI: 10.5555/2033313.2033314.

[16] Hannah Augustin, Martin Sudmanns, Dirk Tiede, Stefan Lang, and Andrea Baraldi. "Semantic Earth Observation Data Cubes". In: *Data* 4.3 (2019). ISSN: 2306-5729. DOI: 10.3390/data4030102.

[17] Franz Baader, Diego Calvanese, Deborah McGuinness, and Daniele Nardi. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, UK., 2002. DOI: 10.1017/CBO9780511711787.

[18] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *Introduction to Description Logic*. Cambridge University Press, 2017. DOI: 10.1017/9781139025355.

[19] Thierry Badard and Etienne Dubé. "Enabling Geospatial Business Intelligence". In: *Open Source Business Resource* (Sept. 2009). ISSN: 1913-6102. URL: http://timreview.ca/article/289.

[20] Timea Bagosi, Diego Calvanese, Josef Hardi, Sarah Komla-Ebri, Davide Lanti, Martin Rezk, Mariano Rodríguez-Muro, Mindaugas Slusnys, and Guohui Xiao. "The Ontop Framework for Ontology Based Data Access". In: *The Semantic Web and Web Science*. Springer, 2014, pp. 67–77. ISBN: 978-3-662-45495-4. DOI: 10.1007/978-3-662-45495-4_6.

[21] Robert Battle and Dave Kolas. "Enabling The Geospatial Semantic Web With Parliament and GeoSAPRQL". In: *Semantic Web* (2012). DOI: 10.3233/SW-2012-0065.

[22] Peter Baumann. "Management of Multidimensional Discrete Data". In: *The VLDB Journal* 3.4 (1994), pp. 401–444. DOI: 10.1007/BF01231603.

[23] Peter Baumann. *Rasdaman - Raster Data Manager*. Version 9.5.0. Jan. 2018. DOI: 10.5281/zenodo.1163021.

[24] Peter Baumann. *The Datacube Manifesto*. Tech. rep. 2017. URL: https://earthserver.eu/tech/datacube-manifesto/The-Datacube-Manifesto.pdf.

[25] Peter Baumann. "The OGC Web Coverage Processing Service (WCPS) Standard". In: *GeoInformatica* 14 (Oct. 2010), pp. 447–479. DOI: 10.1007/s10707-009-0087-2.

[26] Peter Baumann, Paula Furtado, Roland Ritsch, and Norbert Widmann. "The RasDaMan Approach to Multidimensional Database Management". In: *Proceedings of the 1997 ACM Symposium on Applied Computing*. SAC '97. San Jose, California, USA: ACM, 1997, pp. 166–173. ISBN: 0897918509. DOI: 10.1145/331697.331732.

[27] Peter Baumann, Peter Eric Hirschorn, and Joan Masó. *OGC Coverage Implementation Schema (CIS), v1.1*. Tech. rep. Open Geospatial Consortium, 2017. URL: http://docs.opengeospatial.org/is/09-146r8/09-146r8.html.

[28] Peter Baumann, Dimitar Misev, Vlad Merticariu, and Bang Pham Huu. "Array databases: concepts, standards, implementations". In: *Journal of Big Data* 8.1 (Dec. 2021), p. 28. ISSN: 2196-1115. DOI: 10.1186/s40537-020-00399-2.

[29] Peter Baumann, Dimitar Misev, Vlad Merticariu, and Bang Pham Huu. "Datacubes: Towards Space/Time Analysis-Ready Data". In: *Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*. Springer, 2019, pp. 269–299. ISBN: 978-3-319-72434-8. DOI: 10.1007/978-3-319-72434-8_14.

[30] W. Beek, E. Folmer, L. Rietveld, and J. Walker. “GeoYasgui: The Geosparql Query Editor And Result Set Visualizer”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4/W2 (2017), pp. 39–42. DOI: 10.5194/isprs-archives-XLII-4-W2-39-2017.

[31] Konstantina Bereta, Guohui Xiao, and Manolis Koubarakis. “Ontop-spatial: Ontop of Geospatial Databases”. In: *J. of Web Semantics* 58 (2019). DOI: 10.1016/j.websem.2019.100514.

[32] Tim Berners-Lee, Wendy Hall, James A. Hendler, Kieron O’Hara, Nigel Shadbolt, and Daniel J. Weitzner. “A Framework for Web Science”. In: *Foundations and Trends® in Web Science* 1.1 (Sept. 2006), pp. 1–130. ISSN: 1555-077X. DOI: 10.1561/1800000001.

[33] Tim Berners-Lee, James A. Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (2001), pp. 34–43. URL: https://courses.cs.umbc.edu/graduate/691/spring13/01/papers/semanticWebSciAm.pdf.

[34] Dimitris Bilidas, Anastasios Mantas, Filippos Yfantis, George Stamoulis, and Manolis Koubarakis. “Plato: A Semantic Data Cube Implementation Using Ontology-Based Data Access Technologies”. In: 2023. URL: https://cgi.di.uoa.gr/~koubarak/publications/2023/Plato-BiDS2023.pdf.

[35] Dimitris Bilidas, Anastasios Mantas, Filippos Yfantis, George Stamoulis, and Manolis Koubarakis. “The Semantic Data Cube System Plato and Its Applications”. In: Athens, Greece, 2024. URL: https://cgi.di.uoa.gr/~koubarak/publications/2024/plato_igrass.pdf.

[36] Dimitris Bilidas, Anastasios Mantas, Filippos Yfantis, George Stamoulis, Manolis Koubarakis, Spyros Kondylatos, Ioannis Prapas, and Ioannis Papoutsis. “Fire Risk Management using Data Cubes, Machine Learning and OBDA systems”. In: *Proc. of the 31st ACM Int. Conf. on Advances in Geographic Information Systems (SIGSPATIAL)*. ACM, 2023. DOI: 10.1145/3589132.3625615.

[37] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. “An Improved LOD Specification For 3D Building Models”. In: *Computers, Environment and Urban Systems* 59 (2016), pp. 25–37. ISSN: 0198-9715. DOI: 10.1016/j.compenvurbsys.2016.04.005.

[38] Simon Bin, Claus Stadler, Lorenz Bühmann, and Michael Martin. “Getting Practical with GeoSPARQL and Apache Jena”. In: *Proceedings of the 6th International Workshop on Geospatial Linked Data, GeoLD2024, co-located with 21st Extended Semantic Web Conference (ESWC’24), Hersonissos, Greece, May 26th, 2024*. CEUR Workshop Proceedings. 2024. URL: https://ceur-ws.org/Vol-3743/paper2.pdf.

[39] Heather C Bingham, Diego Juffe Bignoli, Edward Lewis, Brian MacSharry, Neil D Burgess, Piero Visconti, Marine Deguignet, Murielle Misrachi, Matt Walpole, Jessica L Stewart, et al. “Sixty Years of Tracking Conservation Progress Using The World Database on Protected Areas”. In: *Nature ecology & evolution* 3.5 (2019), pp. 737–743. DOI: 10.1038/s41559-019-0869-3. URL: https://www.protectedplanet.net/en.

[40] Christian Bizer, Tom Heath, and Tim Berners-Lee. “Linked Data - The Story So Far”. In: *Linking the World’s Information: Essays on Tim Berners-Lee’s Invention of the World Wide Web*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023, pp. 115–143. ISBN: 9798400707940. DOI: 10.1145/3591366.3591378.

[41] Christian Bizer, Tom Heath, and Tim Berners-Lee. “Linked Data - The Story So Far”. In: 5 (), pp. 1–22. DOI: 10.4018/jswis.2009081901.

[42] Christian Bizer and Andreas Schultz. “The Berlin SPARQL Benchmark”. In: *Int. J. Semantic Web Inf. Syst.* 5.2 (2009), pp. 1–24. DOI: 10.4018/JSWIS.2009040101.

[43] Geoff Boeing. “Modeling and Analyzing Urban Networks and Amenities With OSMnx”. In: *Geographical Analysis* (May 2025), p. 11. DOI: 10.1111/gean.70009.

[44] Martin Breunig, Patrick Erik Bradley, Markus Jahn, Paul Kuper, Nima Mazroob, Norbert Rösch, Mulhim Al-Doori, Emmanuel Stefanakis, and Mojgan Jadidi. “Geospatial Data Management Research: Progress and Future Directions”. In: *ISPRS International Journal of Geo-Information* 9.2 (2020). ISSN: 2220-9964. DOI: 10.3390/ijgi9020095.

[45] Dan Brickley, R Guha, et al. *RDF Schema 1.1*. W3CRecommendation. Available at https://www.w3.org/TR/rdf-schema/. 2004.

[46] Nieves R. Brisaboa, Guillermo de Bernardo, Gilberto Gutiérrez, Miguel R. Luaces, and José R. Paramá. “Efficiently Querying Vector and Raster Data”. In: *The Computer Journal* (2017). DOI: 10.1093/comjnl/bxx011.

[47] Jan vom Brocke, Alan Hevner, and Alexander Maedche. “Introduction to Design Science Research”. In: *Design Science Research Cases*. Springer, 2020, pp. 1–13. DOI: 10.1007/978-3-030-46781-4_1.

[48] Paul G. Brown. “Overview of SciDB: Large Scale Array Storage, Processing and Analysis”. In: *Proceedings of the International Conference on Management of Data*. SIGMOD ’10. Indiana, USA: ACM, 2010. ISBN: 9781450300322. DOI: 10.1145/1807167.1807271.

[49] Agustina Buccella, Alejandra Cechich, and Pablo Fillottrani. “Ontology-Driven Geographic Information Integration: A Survey of Current Approaches”. In: *Computers & Geosciences* 35.4 (2009). Geoscience Knowledge Representation in Cyberinfrastructure, pp. 710–723. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2008.02.033.

[50] H. Butler, M. Daly, A. Doyle, Sean Gillies, T. Schaub, and Stefan Hagen. *The GeoJSON Format*. RFC 7946, Internet Engineering Task Force (IETF). Aug. 2016. DOI: 10.17487/RFC7946.

[51] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. "Ontop: Answering SPARQL Queries over Relational Databases". In: *Semantic Web J.* 8.3 (2017). DOI: 10.3233/SW-160217.

[52] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. "Ontop: Answering SPARQL queries over relational databases". In: *Semantic Web* 8.3 (2017). DOI: 10.3233/SW-160217.

[53] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family". In: *J. of Automated Reasoning* 39.3 (2007). DOI: 10.1007/s10817-007-9078-x.

[54] Diego Calvanese, Martin Giese, Peter Haase, Ian Horrocks, Thomas Hubauer, Yannis E. Ioannidis, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Herald Kllapi, Manolis Koubarakis, Steffen Lamparter, Ralf Möller, Christian Neuenstadt, Özgür L. Özçep, Mariano Rodriguez-Muro, Mikhail Roshchin, Marco Ruzzi, Domenico Fabio Savo, Michael Schmidt, Ahmet Soylu, Arild Waaler, and Dmitriy Zheleznyakov. "The Optique Project: Towards OBDA Systems for Industry (Short Paper)". In: *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED'13) co-located with 10th Extended Semantic Web Conference (ESWC'13), Montpellier, France, May 26-27, 2013*. Vol. 1080. CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL: https://ceur-ws.org/Vol-1080/owled2013%5C_20.pdf.

[55] Diego Calvanese, Davide Lanti, Tarcisio Mendes De Farias, Alessandro Mosca, and Guohui Xiao. "Accessing Scientific Data Through Knowledge Graphs with Ontop". In: *Patterns* 2.10 (2021). DOI: 10.1016/j.patter.2021.100346.

[56] Araya Chaiprasert, Naphat Taweekarn, and Jongsawas Chongwatpol. "Applications Of A Geographic Information System–Based Business Intelligence System For Decision Making At Coffee Refresh". In: *Journal of Information Technology Teaching Cases* 13.1 (2023), pp. 16–29. DOI: 10.1177/20438869211040514.

[57] Bruno Chatenoux, Jean-Philippe Richard, David Small, Claudia Roeoesli, Vladimir R. Wingate, Charlotte Poussin, Denisa Rodila, Pascal Peduzzi, Charlotte Steinmeier, Christian Ginzler, Achileas Psomas, Michael E. Schaepman, and Gregory Giuliani. "The Swiss data cube, Analysis Ready Data Archive Using Earth Observations of Switzerland". In: *Scientific Data* 8 (2021). DOI: 10.1038/s41597-021-01076-6.

[58] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. "An Overview of Business Intelligence Technology". In: *ACM* 54.8 (Aug. 2011), pp. 88–98. ISSN: 0001-0782. DOI: 10.1145/1978542.1978562.

[59] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". In: *Commun. ACM* 13.6 (June 1970), pp. 377–387. ISSN: 0001-0782. DOI: 10.1145/362384.362685.

[60] Helen Couclelis. "People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS". In: *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. Ed. by A. U. Frank, I. Campari, and U. Formentini. Springer, 1992.

[61] Edward Craig. "Ontology". In: *Routledge Encyclopedia of Philosophy* (1998). DOI: 10.4324/9780415249126-N039-1.

[62] Souripriya Das, Seema Sundara, and Richard Cyganiak. *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. Available at http://www.w3.org/TR/r2rml/. World Wide Web Consortium, Sept. 2012.

[63] Guillermo De Bernardo, Sandra Álvarez-García, Nieves R. Brisaboa, Gonzalo Navarro, and Oscar Pedreira. "Compact Querieable Representations of Raster Data". In: *Proc. of the 20th Int. Symp. on String Processing and Information Retrieval (SPIRE)*. Springer, 2013. DOI: 10.1007/978-3-319-02432-5_14.

[64] Lipika Dey. "Knowledge Graph-Driven Data Processing For Business Intelligence". In: *WIREs Data Mining and Knowledge Discovery* (2024). DOI: 10.1002/widm.1529.

[65] Kamel Didan. *MODIS/Terra Vegetation Indices 16-Day L3 Global 250m SIN Grid V061*. 2021. DOI: 10.5067/MODIS/MOD13Q1.061.

[66] Lewis Dijkstra, Hugo Poelman, and Paolo Veneri. "The EU-OECD Definition Of A Functional Urban Area (FUA)". In: (2019). DOI: 10.1787/d58cb34d-en. URL: https://ec.europa.eu/regional_policy/sources/focus/2012_01_city.pdf.

[67] L. Ding, G. Xiao, A. Pano, H. Fan, D. Calvanese, and L. Meng. "Querying CityGML Data through Virtual Knowledge Graphs". In: *Abstracts of the International Cartographic Association, 31st International Cartographic Conference (ICC'23* 6 (Aug. 2023), p. 53. DOI: 10.5194/ica-abs-6-53-2023.

[68] Linfang Ding, Guohui Xiao, Diego Calvanese, and Liqiu Meng. "A Framework Uniting Ontology-Based Geodata Integration and Geovisual Analytics". In: *ISPRS International Journal of Geo-Information* 9.8 (2020). ISSN: 2220-9964. DOI: 10.3390/ijgi9080474.

[69] Linfang Ding, Guohui Xiao, Albulen Pano, Mattia Fumagalli, Dongsheng Chen, Yu Feng, Diego Calvanese, Hongchao Fan, and Liqiu Meng and. “Integrating 3D City Data through Knowledge Graphs”. In: *Geospatial Information Science* 28.2 (2025), pp. 780–799. DOI: 10.1080/10095020.2024.2337360.

[70] Linfang Ding, Guohui Xiao, Albulen Pano, Claus Stadler, and Diego Calvanese. “Towards the Next Generation of the LinkedGeodata Project Using Virtual Knowledge Graphs”. In: *Journal of Web Semantics* 71 (2021). ISSN: 1570-8268. DOI: 10.1016/j.websem.2021.100662.

[71] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration.* 2012. ISBN: 978-0-12-416044-6. DOI: 10.1016/C2011-0-06130-6. URL: https://doi.org/10.1016/C2011-0-06130-6.

[72] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. “WorldKG: A World-Scale Geographic Knowledge Graph”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management.* CIKM ’21. Queensland, Australia: ACM, 2021, pp. 4475–4484. ISBN: 9781450384469. DOI: 10.1145/3459637.3482023.

[73] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. *From Local to Global: A Graph RAG Approach to Query-Focused Summarisation.* 2025. URL: https://arxiv.org/abs/2404.16130.

[74] Lisa Ehrlinger and Wolfram Wöß. “Towards A Definition of Knowledge Graphs.” In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016), p. 2. URL: https://ceur-ws.org/Vol-1695/paper4.pdf.

[75] Environmental Systems Research Institute (ESRI). *How Zonal Statistics Works, ArcGis 10.8.* https://desktop.arcgis.com/en/arcmap/latest/tools/spatial-analyst-toolbox/how-zonal-statistics-works.htm. (Accessed on 13 July 2023). 2021.

[76] Environmental Systems Research Institute (ESRI). *Shapefile Technical Description White Paper.* Tech. rep. Redlands, CA, USA, 1998. URL: https://downloads.esri.com/support/whitepapers/mo_/shapefile.pdf.

[77] Hongchao Fan, Alexander Zipf, Qing Fu, and Pascal Neis and. “Quality Assessment for Building Footprints Data on OpenStreetMap”. In: *International Journal of Geographical Information Science* 28.4 (2014), pp. 700–719. DOI: 10.1080/13658816.2013.867495.

[78] Tom G. Farr, Paul A. Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, David Seal, Scott Shaffer, Joanne Shimada, Jeffrey Umland, Marian Werner, Michael Oskin, Douglas Burbank, and Douglas Alsdorf. "The Shuttle Radar Topography Mission". In: *Reviews of Geophysics* 45.2 (2007), p. 33. DOI: 10.1029/2005RG000183.

[79] Erwin Folmer, W. Beek, Luuk Rietveld, S. Ronzhin, R. Geerling, and D. den Haan. "Enhancing the Usefulness of Open Governmental Data with Linked Data Viewing Techniques". In: *Proceedings of the 52nd Annual Hawaii International Conference on System Sciences, (HICSS 2019).* Hawaii, USA, Jan. 2019, pp. 2912–2921. ISBN: ISBN: 978-0-9981331-2-6. URL: https://hdl.handle.net/10125/59728.

[80] Sergio Freire and Martino Pesaresi. "GHS Population Grid, Derived from GPW4, Multitemporal (1975, 1990, 2000, 2015)". In: *European Commission, Joint Research Centre (JRC)* (2015). URL: https://ghsl.jrc.ec.europa.eu/.

[81] Martin Frické. "The Knowledge Pyramid: the DIKW Hierarchy". In: *Knowledge Organization (KO)* 46.1 (2019), pp. 33–46. DOI: 10.5771/0943-7444-2019-1-33.

[82] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. "Sweetening Ontologies with DOLCE". In: EKAW '02 (2002), pp. 166–181. DOI: 10.5555/645362.650863.

[83] Fan Gao, Peng Yue, Zhipeng Cao, Shuaifeng Zhao, Boyi Shangguan, Liangcun Jiang, Lei Hu, Zhe Fang, and Zheheng Liang. "A Multi-source Spatio-Temporal Data Cube For Large-Scale Geospatial Analysis". In: *International Journal of Geographical Information Science* 36.9 (2022). DOI: 10.1080/13658816.2022.2087222.

[84] Mouzhi Ge and Vlastislav Dohnal. "Quality Management in Big Data". In: *Informatics* 5.2 (2018). DOI: 10.3390/informatics5020019.

[85] GEOS contributors. *Geometry Engine Open Source (GEOS) Computational Geometry Library.* Open Source Geospatial Foundation, 2025. DOI: 10.5281/zenodo.11396894. URL: https://libgeos.org/.

[86] Arka Ghosh, Albulen Pano, Benjamin Cogrel, and Diego Calvanese. "Semantic Enrichment of Location-based Business Intelligence using Virtual Knowledge Graphs". In: *International Semantic Intelligence Conference (ISIC 2025).* Lübeck, Germany: Springer, 2026, pp. 3–10.

[87] Arka Ghosh, Albulen Pano, Guohui Xiao, and Diego Calvanese. "OntoRaster: Extending VKGs with Raster Data". In: *Proceedings of the 08th International Joint Conference on Rules and Reasoning (RuleML+RR'24), a part of Declarative AI, Bucharest, Romania.* Springer, 2024, pp. 108–123. ISBN: 978-3-031-72407-7. DOI: 10.1007/978-3-031-72407-7_9.

[88] Mario A Gomarasca. “Basics Of Geomatics”. In: *Applied Geomatics* 2 (2010), pp. 137–146. DOI: 10.1007/s12518-010-0029-6.

[89] Cristian González García and Eva Álvarez-Fernández. “What Is (Not) Big Data Based on Its 7Vs Challenges: A Survey”. In: *Big Data and Cognitive Computing* 6.4 (2022). ISSN: 2504-2289. DOI: 10.3390/bdcc6040158.

[90] Michael F. Goodchild. “Citizens As Sensors: The World of Volunteered Geography”. In: *GeoJournal* 69.4 (Nov. 2007), pp. 211–221. DOI: 10.1007/s10708-007-9111-y.

[91] Thomas Gottron and Steffen Staab. “Linked Open Data”. In: *Encyclopedia of Social Network Analysis and Mining.* New York, USA: Springer, 2014, pp. 811–813. DOI: 10.1007/978-1-4614-6170-8_111.

[92] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. “Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab, and Sub-totals”. In: *Data mining and knowledge discovery* 1 (1997). DOI: 10.1023/A:1009726021843.

[93] Gerhard Gröger, Thomas H Kolbe, Claus Nagel, and Karl-Heinz Häfele. “OGC City Geography Markup Language (CityGML) Encoding Standard”. In: (2012), p. 344. URL: https://www.ogc.org/standards/citygml/.

[94] Thomas R. Gruber. “A Translation Approach To Portable Ontology Specifications”. In: *Knowledge Acquisition* 5.2 (1993), pp. 199–220. ISSN: 1042-8143. DOI: 10.1006/knac.1993.1008. URL: https://tomgruber.org/writing/ontolingua-kaj-1993.pdf.

[95] Stéphane Grumbach, Philippe Rigaux, and Luc Segoufin. “Manipulating Interpolated Data is Easier than You Thought”. In: *VLDB*. 2000. URL: http://www.vldb.org/conf/2000/P156.pdf.

[96] Nicola Guarino. “Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration”. In: *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology.* Springer, 1997, pp. 139–170. ISBN: 978-3-540-69548-6. DOI: 10.1007/3-540-63438-X_8.

[97] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. “LUBM: A Benchmark for OWL Knowledge Base Systems”. In: *Journal of Web Semantics* 3.2 (2005), pp. 158–182. DOI: 10.1016/j.websem.2005.06.005.

[98] Mordechai Haklay and Patrick Weber. “OpenStreetMap: User-Generated Street Maps”. In: *IEEE Pervasive Computing* 7.4 (2008), pp. 12–18. DOI: 10.1109/MPRV.2008.80.

[99] Dorothy K. Hall and George A. Riggs. *MODIS/Terra Snow Cover Daily L3 Global 500m SIN Grid, Version 61.* 2021. DOI: 10.5067/MODIS/MOD10A1.061.

[100] Harry Halpin, Patrick J. Hayes, Jamie P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. "When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data". In: *The Semantic Web – ISWC 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 305–320. ISBN: 978-3-642-17746-0. DOI: 10.1007/978-3-642-17746-0_20.

[101] Younes Hamdani, Guohui Xiao, Linfang Ding, and Diego Calvanese. "An Ontology-Based Framework for Geospatial Integration and Querying of Raster Data Cube Using Virtual Knowledge Graphs". In: *ISPRS International Journal of Geo-Information* 12.9 (2023). ISSN: 2220-9964. DOI: 10.3390/ijgi12090375.

[102] Jiawei Han, Nebojsa Stefanovic, and Krzysztof Koperski. "Selective Materialisation: An Efficient Method for Spatial Data Cube Construction". In: *Research and Development in Knowledge Discovery and Data Mining*. Springer, 1998. DOI: 10.1007/3-540-64383-4_13.

[103] Mohamed Hanine, Omar Boutkhoum, Tarik Agouti, and Abdessadek Tikniouine. "A New Integrated Methodology Using Modified Delphi-Fuzzy AHP-PROMETHEE for Geospatial Business Intelligence Selection". In: *Information Systems and e-Business Management* 15 (2017), pp. 897–925. DOI: 10.1007/s10257-016-0334-7.

[104] Hansson, Felix. "Linked Geodata: CityGML Represented As A Virtual Knowledge Graph". PhD thesis. Lund, Sweden: Department of Physical Geography and Ecosystem Science, Lund University, 2024. URL: http://lup.lub.lu.se/student-papers/record/9169387.

[105] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. "Implementing Data Cubes Efficiently". In: *SIGMOD Rec.* 25.2 (June 1996), pp. 205–216. ISSN: 0163-5808. DOI: 10.1145/235968.233333.

[106] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. W3C Recommendation. Available at http://www.w3.org/TR/sparql11-query. World Wide Web Consortium, 2013.

[107] Tom Heath and Christian Bizer. "Linked Data: Evolving the Web into a Global Data Space". In: vol. 11. Morgan & Claypool Publishers, Feb. 2011. DOI: 10.2200/S00334ED1V01Y201102WBE001.

[108] Tom Heath and Christian Bizer. "Principles of Linked Data". In: *Linked Data: Evolving the Web into a Global Data Space*. Springer, 2011, pp. 7–27. ISBN: 978-3-031-79432-2. DOI: 10.1007/978-3-031-79432-2_2.

[109] John Herring et al. *OpenGIS® Implementation Standard for Geographic information- Simple Feature Access - Part 1: Common architecture [Corrigendum]*. Tech. rep. OGC, 2011. DOI: 10.25607/OBP-630.

[110] Aidan Hogan. "Knowledge graphs: Research directions". In: *Reasoning Web. Declarative Artificial Intelligence: 16th International Summer School 2020, Oslo, Norway, June 24–26, 2020, Tutorial Lectures 16* (2020). DOI: 10.1007/978-3-030-60067-9_8.

[111] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. "Knowledge Graphs". In: *ACM Computing Surveys* (2021). DOI: 10.1145/3447772.

[112] Timo Homburg, Steffen Staab, and Daniel Janke. "GeoSPARQL+: Syntax, Semantics and System for Integrated Querying of Graph, Raster and Vector Data". In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Proceedings, Part I*. LNCS. Athens, Greece: Springer, Nov. 2020. DOI: 10.1007/978-3-030-62419-4_15.

[113] Simon Hook and Gregory Halverson. *ECOSTRESS Tiled Downscaled Soil Moisture Instantaneous L3 Global 70 m v002*. 2024. DOI: 10.5067/ECOSTRESS/ECO_L3T_SM.002.

[114] AH Hor, A Jadidi, and G Sohn. "BIM-GIS Integrated Geospatial Information Model Using Semantic Web and RDF Graphs". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3 (2016), pp. 73–79. DOI: 10.5194/isprsannals-III-4-73-201673.

[115] Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler. "Semantic Web Architecture: Stack or Two Towers?" In: *Principles and Practice of Semantic Web Reasoning (PPSWR'05)*. Springer, 2005, pp. 37–41. ISBN: 978-3-540-32028-9. DOI: 10.1007/11552222_4.

[116] S. Hoyer and J. Hamman. "Xarray: N-D Labeled Arrays and Datasets in Python". In: *Journal of Open Research Software* 5.1 (2017). DOI: 10.5334/jors.148.

[117] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. "LORA: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3. URL: https://arxiv.org/pdf/2106.09685v1/1000.

[118] Fei Hu, Mengchao Xu, Jingchao Yang, Yanshou Liang, Kejin Cui, Michael M. Little, Christopher S. Lynnes, Daniel Q. Duffy, and Chaowei Yang. "Evaluating the Open Source Data Containers for Handling Big Geospatial Raster Data". In: *ISPRS International Journal of Geo-Information* 7.4 (2018). ISSN: 2220-9964. DOI: 10.3390/ijgi7040144.

[119] Weiming Huang, Khashayar Kazemzadeh, Ali Mansourian, and Lars Harrie. "Towards Knowledge-Based Geospatial Data Integration and Visualization: A Case of Visualizing Urban Bicycling Suitability". In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.2992023.

[120] Yaqi Huang, Minrui Zheng, Tianle Li, Fei Xiao, and Xinqi Zheng. "An Integrated Framework for Landscape Indices' Calculation with Raster–Vector Integration and Its Application Based on QGIS". In: *ISPRS International Journal of Geo-Information* 13.7 (2024). ISSN: 2220-9964. DOI: 10.3390/ijgi13070242.

[121] Krzysztof Janowicz, Song Gao, Grant McKenzie, Yingjie Hu, and Budhendra Bhaduri. "GeoAI: spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond". In: *International Journal of Geographical Information Science* 34.4 (2020), pp. 625–636. DOI: 10.1080/13658816.2019.1684500.

[122] Krzysztof Janowicz and Pascal Hitzler. "Geospatial Semantic Web". In: *International Encyclopedia of Geography*. John Wiley & Sons, Ltd, 2017, pp. 1–6. DOI: 10.1002/9781118786352.wbieg1158.

[123] Krzysztof Janowicz, Pascal Hitzler, Wenwen Li, Dean Rehberger, Mark Schildhauer, Rui Zhu, Cogan Shimizu, Colby K. Fisher, Ling Cai, Gengchen Mai, Joseph Zalewski, Lu Zhou, Shirly Stephen, Seila Gonzalez, Bryce Mecum, Anna Lopez-Carr, Andrew Schroeder, David Smith, Dawn Wright, Sizhe Wang, Yuanyuan Tian, Zilong Liu, Meilin Shi, Anthony D'Onofrio, Zhining Gu, and Kitty Currier. "Know, Know Where, KnowWhereGraph: A Densely Connected, Cross-domain Knowledge Graph and Geo-Enrichment Service Stack for Applications in Environmental Intelligence". In: *AI Magazine* 43.1 (2022), pp. 30–39. DOI: 10.1002/aaai.12043.

[124] Milos Jovanovik, Timo Homburg, and Mirko Spasić. "A GeoSPARQL Compliance Benchmark". In: *ISPRS International Journal of Geo-Information* 10.7 (2021). ISSN: 2220-9964. DOI: 10.3390/ijgi10070487.

[125] Stephen Kaisler, J Alberto Espinosa, William Money, and Frank Armour. "Big Data and Analytics: Issues and Challenges For The Past and Next Ten Years". In: *Proceedings of the 56th Hawaii International Conference on System Sciences* (2023). DOI: 10.24251/HICSS.2023.101.

[126] Elem G zel Kalayci, Guohui Xiao, Vladislav Ryzhikov, Tahir Emre Kalayci, and Diego Calvanese. "Ontop-temporal: A tool for ontology-based query answering over temporal data". In: *International Conference on Information and Knowledge Management, Proceedings* (2018), pp. 1927–1930. DOI: 10.1145/3269206.3269230.

[127] Nikolaos Karalis, Georgios Mandilaras, and Manolis Koubarakis. "Extending the YAGO2 Knowledge Graph with Precise Geospatial Knowledge". In: *The Semantic Web – ISWC 2019*. Springer, 2019, pp. 181–197. DOI: 10.1007/978-3-030-30796-7_12.

[128] Sergios-Anestis Kefalidis, Dharmen Punjani, Eleni Tsalapati, Konstantinos Plas, Mariangela Pollali, Michail Mitsios, Myrto Tsokanaridou, Manolis Koubarakis, and Pierre Maret. "Benchmarking Geospatial Question Answering Engines Using the Dataset GeoQuestions1089". In: *The Semantic Web – ISWC 2023*. Springer Nature Switzerland, 2023, pp. 266–284. DOI: 10.1007/978-3-031-47243-5_15.

[129] Mayank Kejriwal. "What Is a Knowledge Graph?" In: *Domain-Specific Knowledge Graph Construction*. Springer, 2019, pp. 1–7. DOI: 10.1007/978-3-030-12375-8_1.

[130] Nathaniel Vaughn Kelso and Tom Patterson. *Natural Earth Data*. Tech. rep. 2010. URL: https://www.naturalearthdata.com/features/.

[131] Nawsher Khan, Arshi Naim, Mohammad Rashid Hussain, Quadri Noorulhasan Naveed, Naim Ahmad, and Shamimul Qamar. "The 51 V's Of Big Data: Survey, Technologies, Characteristics, Opportunities, Issues and Challenges". In: COINS '19. Crete, Greece: ACM, 2019, pp. 19–24. ISBN: 9781450366403. DOI: 10.1145/3312614.3312623.

[132] Evgeny Kharlamov, Dag Hovland, Martin Georg Skjæveland, Dimitris Bilidas, Ernesto Jiménez-Ruiz, Guohui Xiao, Ahmet Soylu, Davide Lanti, Martín Rezk, Dmitriy Zheleznyakov, Martin Giese, Hallstein Lie, Yannis E. Ioannidis, Yannis Kotidis, Manolis Koubarakis, and Arild Waaler. "Ontology Based Data Access in Statoil". In: *J. Web Semant.* 44 (2017), pp. 3–36. DOI: 10.2139/ssrn.3199136.

[133] Brian Killough. "Overview of the Open Data Cube Initiative". In: *IGARSS 2018 - IEEE International Geoscience and Remote Sensing Symposium*. 2018, pp. 8629–8632. DOI: 10.1109/IGARSS.2018.8517694. URL: https://www.opendatacube.org/.

[134] Han-Saem Kim, Chang-Guk Sun, and Hyung-Ik Cho. "Geospatial Big Data-Based Geostatistical Zonation of Seismic Site Effects in Seoul Metropolitan Area". In: *ISPRS International Journal of Geo-Information* 6.6 (2017). ISSN: 2220-9964. DOI: 10.3390/ijgi6060174.

[135] Jakub Klímek, Petr Škoda, and Martin Nečaský. "Survey of Tools For Linked Data Consumption". In: *Semantic Web* 10.4 (2019), pp. 665–720. DOI: 10.3233/SW-180316.

[136] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Sept. 2009), pp. 455–500. DOI: 10.1137/07070111X.

[137] Steve Kopp, Peter Becker, Abhijit Doshi, Dawn J. Wright, Kaixi Zhang, and Hong Xu. "Achieving the Full Vision of Earth Observation Data Cubes". In: *Data* 3 (2019). ISSN: 2306-5729. DOI: 10.3390/data4030094.

[138] Thomas Krijnen, Francesca Noardo, Ken Arroyo Ohori, and Jantien Stoter. "Multi-disciplinary Use of Three-Dimensional Geospatial Information". In: *Industry 4.0 for the Built Environment: Methodologies, Technologies and Skills*. Springer, 2022. ISBN: 978-3-030-82430-3. DOI: 10.1007/978-3-030-82430-3_12.

[139] Markus Krötzsch. "OWL 2 profiles: An Introduction to Lightweight Ontology Languages". In: *Reasoning Web International Summer School*. Springer, 2012. DOI: 10.1007/978-3-642-33158-9_4.

[140] Tatjana Kutzner, Kanishk Chaturvedi, and Thomas H Kolbe. "CityGML 3.0: New Functions Open Up New Applications". In: *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88.1 (2020), pp. 43–61. DOI: 10.1007/s41064-020-00095-z.

[141] Christos Kyriakos and Manolis Vavalis. "Business Intelligence Through Machine Learning from Satellite Remote Sensing Data". In: *Future Internet* 15.11 (2023). ISSN: 1999-5903. DOI: 10.3390/fi15110355.

[142] Sébastien Laborie, Franck Ravat, Jiefu Song, and Olivier Teste. "Combining Business Intelligence with Semantic Web: Overview and Challenges". In: 33rd Congrès sur l' INformatique des ORganisations et Systèmes d'Information et de Décision (INFORSID'15), 2015, pp. 99–114. URL: https://publications.ut-capitole.fr/id/eprint/29563.

[143] Susana Ladra, José R. Paramá, and Fernando Silva-Coira. "Scalable and Queryable Compressed Storage Structure for Raster Data". In: *IS* 72 (2017). DOI: 10.1016/j.is.2017.10.007.

[144] Douglas Laney. *3D Data Management: Controlling Data Volume, Velocity, and Variety*. Tech. rep. Feb. 2001.

[145] Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, and Diego Calvanese. "The NPD Benchmark for OBDA Systems". In: *Proceedings of the 10th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy*. Vol. 1261. CEUR-WS.org, Oct. 2014, pp. 3–18. URL: https://ceur-ws.org/Vol-1261/SSWS2014%5C_paper1.pdf.

[146] Davide Lanti, Martín Rezk, Guohui Xiao, and Diego Calvanese. "The NPD Benchmark: Reality Check for OBDA Systems". In: *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*. OpenProceedings.org, 2015, pp. 617–628. DOI: 10.5441/002/EDBT.2015.62.

[147] Davide Lanti, Guohui Xiao, and Diego Calvanese. “Fast and Simple Data Scaling for OBDA Benchmarks”. In: *Proceedings of the Workshop on Benchmarking Linked Data (BLINK'16) co-located with the 15th International Semantic Web Conference (ISWC), Kobe, Japan.* Vol. 1700. CEUR-WS.org, Oct. 2016. URL: https://ceur-ws.org/Vol-1700/paper-06.pdf.

[148] Davide Lanti, Guohui Xiao, and Diego Calvanese. “VIG: Data scaling for OBDA benchmarks”. In: *Semantic Web* 10.2 (2019), pp. 413–433. DOI: 10.3233/SW-180336.

[149] Susan Leach-Murray. “The Linked Open Data Cloud”. In: *Technical Services Quarterly* 38.2 (2021), pp. 193–194. DOI: 10.1080/07317131.2021.1892352.

[150] Hugo Ledoux, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski, and Stelios Vitalis. “CityJSON: A Compact and Easy-to-use Encoding of the CityGML Data Model”. In: *Open Geospatial Data, Software and Standards* 4.1 (2019), pp. 1–12. DOI: 10.1186/s40965-019-0064-0.

[151] Maurizio Lenzerini. “Data Integration: A Theoretical Perspective”. In: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* 2002, pp. 233–246. DOI: 10.1145/543613.543644.

[152] Adam Lewis, Leo Lymburner, Matthew B. J. Purss, Brendan Brooke, Ben Evans, Alex Ip, Arnold G. Dekker, James R. Irons, Stuart Minchin, Norman Mueller, Simon Oliver, Dale Roberts, Barbara Ryan, Medhavy Thankappan, Rob Woodcock, and Lesley Wyborn. “Rapid, High-resolution Detection of Environmental Change Over Continental Scales From Satellite Data – the Earth Observation Data Cube”. In: *International Journal of Digital Earth* 9.1 (2016). DOI: 10.1080/17538947.2015.1111952.

[153] Adam Lewis, Simon Oliver, Leo Lymburner, Ben Evans, Lesley Wyborn, Norman Mueller, Gregory Raevksi, Jeremy Hooke, Rob Woodcock, Joshua Sixsmith, Wenjun Wu, Peter Tan, Fuqin Li, Brian Killough, Stuart Minchin, Dale Roberts, Damien Ayers, Biswajit Bala, John Dwyer, Arnold Dekker, Trevor Dhu, Andrew Hicks, Alex Ip, Matt Purss, Clare Richards, Stephen Sagar, Claire Trenham, Peter Wang, and Lan-Wei Wang. “The Australian Geoscience Data Cube — Foundations and lessons learned”. In: *Remote Sensing of Environment* 202 (2017). ISSN: 0034-4257. DOI: 10.1016/j.rse.2017.03.015.

[154] Wenwen Li and Chia-Yu Hsu. “GeoAI for Large-Scale Image Analysis and Machine Vision: Recent Progress of Artificial Intelligence in Geography”. In: *ISPRS International Journal of Geo-Information* 11.7 (2022). ISSN: 2220-9964. DOI: 10.3390/ijgi11070385.

[155] Wenwen Li, Sizhe Wang, Sheng Wu, Zhining Gu, and Yuanyuan Tian. "Performance Benchmark on Semantic Web Repositories For Spatially Explicit Knowledge Graph Applications". In: *Computers, Environment and Urban Systems* 98 (2022). ISSN: 0198-9715. DOI: 10.1016/j.compenvurbsys.2022.101884.

[156] Vincent Link, Steffen Lohmann, and Florian Haag. "OntoBench: Generating Custom OWL 2 Benchmark Ontologies". In: *The Semantic Web – ISWC 2016*. Springer, 2016, pp. 122–130. ISBN: 978-3-319-46547-0. DOI: 10.1007/978-3-319-46547-0_13.

[157] Junnan Liu, Haiyan Liu, Xiaohui Chen, Xuan Guo, Qingbo Zhao, Jia Li, Lei Kang, and Jianxiang Liu. "A Heterogeneous Geospatial Data Retrieval Method Using Knowledge Graph". In: *Sustainability* 13.4 (2021). ISSN: 2071-1050. DOI: 10.3390/su13042005.

[158] Lingjia Liu, Zhiyi Fu, Yu Xia, Hui Lin, Xiaohui Ding, and Kaitao Liao. "A Building Polygonal Object Matching Method Based On Minimum Bounding Rectangle Combinatorial Optimisation And Relaxation Labelling". In: *Transactions in GIS* 27.2 (2023), pp. 541–563. DOI: 10.1111/tgis.13039.

[159] Meng Lu, Marius Appel, and Edzer Pebesma. "Multidimensional Arrays for Analysing Geoscientific Data". In: *ISPRS International Journal of Geo-Information* 7.8 (2018). ISSN: 2220-9964. DOI: 10.3390/ijgi7080313.

[160] Gengchen Mai, Yingjie Hu, Song Gao, Ling Cai, Bruno Martins, Johannes Scholz, Jing Gao, and Krzysztof Janowicz. "Symbolic and Subsymbolic GeoAI: Geospatial Knowledge Graphs and Spatially Explicit Machine Learning". In: *Transactions in GIS* 26.8 (2022), pp. 3118–3124. DOI: 10.1111/tgis.13012.

[161] Gengchen Mai, Krzysztof Janowicz, Bo Yan, and Simon Scheider. "Deeply Integrating Linked Data with Geographic Information Systems". In: *Transactions in GIS* 23.3 (2019), pp. 579–600. DOI: 10.1111/tgis.12538.

[162] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. https://github.com/huggingface/peft. 2022.

[163] Frank Manola, Eric Miller, Brian McBride, et al. *RDF 1.1 primer*. W3CRecommendation. Available at https://www.w3.org/TR/rdf11-primer/. 2014.

[164] Anastasios Mantas, Filippos Yfantis, Dimitris Bilidas, George Stamoulis, Spyros Kondylatos, Ioannis Prapas, Ioannis Papoutsis, José María Tárraga Habas, Eva Sevillano Marco, Fabien Castel, Camille Laine, and Manolis Koubarakis. “Plato: A Semantic Data Cube System using Ontology-Based Data Access Technologies”. In: *IEEE Access* (Sept. 2024), pp. 1–1. ISSN: 21693536. DOI: 10.1109/access.2024.3453494.

[165] Deborah L McGuinness, Frank Van Harmelen, et al. *OWL web ontology language overview*. Tech. rep. 10. 2004. URL: https://static.twoday.net/71desa1bif/files/W3C-OWL-Overview.pdf.

[166] Narek Meloyan, Aleksandr Hayrapetyan, and Narine Sarvazyan. *Deep Learning Approaches to Multidimensional Hyperspectral Unsupervised Classification*. Tech. rep. Yerevan, Armenia, 2025. URL: https://cse.aua.am/files/2025/06/Narek-Meloyan-Paper.pdf.

[167] Dimitar Misev and Peter Baumann. *SQL Support for Multidimensional Arrays*. Tech. rep. Information Resource Center, Jacobs University, July 2017, pp. 1–74. URL: http://nbn-resolving.org/urn:nbn:de:gbv:579-opus-1007237.

[168] Dimitar Mišev and Peter Baumann. “Remote Sensing Analytics in Databases with ISO SQL/MDA”. In: *IGARSS 2019 - IEEE International Geoscience and Remote Sensing Symposium*. 2019, pp. 4515–4518. DOI: 10.1109/IGARSS.2019.8898179.

[169] Josh Moore and Susanne Kunis. “Zarr: A Cloud-Optimized Storage for Interactive Access of Large Arrays”. In: 1 (Sept. 2023). DOI: 10.52825/cordi.v1i.285.

[170] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C Recommendation. Available at http://www.w3.org/TR/owl2-profiles/. Dec. 2012.

[171] Markus Neteler. *GRASS GIS Manual*. https://grass.osgeo.org/grass82/manuals/v.rast.stats.html. (Accessed on 13 July 2023).

[172] Mykola Nikitchenko. “Revisiting Data-Information-Knowledge-Wisdom Hierarchy from a Logic Perspective”. In: *23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2021, pp. 2159–2165. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00322.

[173] Francesca Noardo, Claire Ellul, Lars Harrie, Ivar Overland, Masoome Shariat, Ken Arroyo Ohori, and Jantien Stoter. “Opportunities and Challenges for GeoBIM in Europe: Developing a Building Permits Use-case to Raise Awareness and Examine Technical Interoperability Challenges”. In: *Journal of Spatial Science* 65.2 (2020), pp. 209–233. DOI: 10.1080/14498596.2019.1627253.

[174] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. *Industry-scale Knowledge Graphs: Lessons and Challenges: Five Diverse Technology Companies Show How It’s Done.* Tech. rep. 2. 2019, pp. 48–75. DOI: 10.1145/3329781.3332266.

[175] John A. O’Keefe. “The Universal Transverse Mercator Grid and Projection”. In: *The Professional Geographer* 4.5 (1952), pp. 19–24. DOI: 10.1111/j.0033-0124.1952.45_19.x.

[176] Open Geospatial Consortium. *GeoSPARQL - A Geographic Query Language for RDF Data.* https://www.ogc.org/standard/geosparql/.

[177] Open Geospatial Consortium. *Web Coverage Service 2.0 Interface Standard - Earth Observation Application Profile.* https://docs.ogc.org/is/10-140r2/10-140r2.html. 2018.

[178] Open Geospatial Consortium. *Web Feature Service 2.0 Interface Standard – With Corrigendum.* https://docs.ogc.org/is/09-025r2/09-025r2.html. 2014.

[179] Open Geospatial Consortium. *Web Map Service Interface Standard – With Corrigendum.* https://www.ogc.org/standard/wms/. 2006.

[180] Albulen Pano, Davide Lanti, and Diego Calvanese. “Virtual Knowledge Graphs over Earth Observation Data”. In: *The Semantic Web – ISWC 2025.* Springer Nature, 2026, pp. 149–166. ISBN: 978-3-032-09530-5. DOI: 10.1007/978-3-032-09530-5_9.

[181] Rahul Parundekar, Craig A. Knoblock, and José Luis Ambite. “Linking and Building Ontologies of Linked Data”. In: *The Semantic Web – ISWC 2010.* Springer, 2010, pp. 598–614. ISBN: 978-3-642-17746-0. DOI: 10.1007/978-3-642-17746-0_38.

[182] Edzer Pebesma, Wolfgang Wagner, Matthias Schramm, Alexandra Von Beringe, Christoph Paulik, Markus Neteler, Johannes Reiche, Jan Verbesselt, Jeroen Dries, Erwin Goor, Thomas Mistelbauer, Christian Briese, Claudia Notarnicola, Roberto Monsorno, Carlo Marin, Alexander Jacob, Pieter Kempeneers, and Pierre Soille. *OpenEO - a Common, Open Source Interface Between Earth Observation Data Infrastructures and Front- End Applications.* Version 1.0. 2017. DOI: 10.5281/zenodo.1065474.

[183] Cecilia Reyes Peña and Mireya Tovar Vidal. “Ontology: Components and Evaluation, A Review”. In: *Research in Computing Science* 148.3 (2019), pp. 257–265. ISSN: 1870-4069. DOI: 10.13053/rcs-148-3-21.

[184] R. Pierdicca and M. Paolanti. “GeoAI: A Review of Artificial Intelligence Approaches for The Interpretation of Complex Geomatics Data”. In: *Geoscientific Instrumentation, Methods and Data Systems* 1 (2022), pp. 195–218. DOI: 10.5194/gi-11-195-2022.

[185] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. “Linking Data to Ontologies”. In: *J. on Data Semantics* 10 (2008). DOI: 10.1007/978-3-540-77688-8_5.

[186] Marco Quartulli and Igor G. Olaizola. “A Review of EO Image Information Mining”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 75 (2013). ISSN: 0924-2716. DOI: /10.1016/j.isprsjprs.2012.09.010.

[187] Erhard Rahm and Hong Hai Do. “Data Cleaning: Problems and Current Approaches”. In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13. URL: http://sites.computer.org/debull/A00dec/A00DEC-CD.pdf.

[188] Sajith Ranatunga, Rune Strand Ødegård, Knut Jetlund, and Erling Onstein. “Use of Semantic Web Technologies to Enhance the Integration and Interoperability of Environmental Geospatial Data: A Framework Based on Ontology-Based Data Access”. In: *ISPRS International Journal of Geo-Information* 14.2 (2025). ISSN: 2220-9964. DOI: 10.3390/ijgi14020052.

[189] Steve Ray. *Quantities, Units, Dimensions and Types (QUDT)*. Tech. rep. Fairsharing.org, 2011. DOI: 10.25504/FAIRsharing.d3pqw7.

[190] Laurens Rietveld and Rinke Hoekstra. “YASGUI: Not Just Another SPARQL Client”. In: *The Semantic Web: ESWC 2013 Satellite Events*. Berlin, Heidelberg: Springer, 2013, pp. 78–86. ISBN: 978-3-642-41242-4. DOI: 10.1007/978-3-642-41242-4_7. URL: https://yasgui.triply.cc/.

[191] Alexandre Robin. *OGC® SWE Common Data Model Encoding Standard. Version 2.0. 0.* Tech. rep. 2011. DOI: https://dx.doi.org/10.25607/OBP-629.

[192] Marko A. Rodriguez and Peter Neubauer. “Constructions from dots and lines”. In: *Bulletin of the American Society for Information Science and Technology* 36.6 (2010), pp. 35–41. DOI: 10.1002/bult.2010.1720360610.

[193] Mariano Rodriguez-Muro, Josef Hardi, and Diego Calvanese. “Quest: Efficient SPARQL-to-SQL for RDF and OWL”. In: *Proceedings of the ISWC 2012 Posters & Demonstrations Track: ISWC 2012 Posters & Demos* 914 (Nov. 2012), pp. 53–56. URL: https://ceur-ws.org/Vol-914/paper_59.pdf.

[194] Even Rouault, Frank Warmerdam, Kurt Schwehr, Andrey Kiselev, Howard Butler, Mateusz Łoskot, Tamas Szekeres, Etienne Tourigny, Martin Landa, Idan Miara, Ben Elliston, Kumar Chaitanya, Lucian Plesea, Daniel Morissette, Ari Jolma, Nyall Dawson, Daniel Baston, Craig de Stigter, and Hiroshi Miura. *GDAL*. Version v3.12.0. Open Source Geospatial Foundation. Zenodo, Nov. 2025. DOI: 10.5281/zenodo.5884351. URL: https://gdal.org.

[195] Daniel Runfola, Austin Anderson, Heather Baier, Matt Crittenden, Elizabeth Dowker, Sydney Fuhrig, Seth Goodman, Grace Grimsley, Rachel Layko, Graham Melville, Maddy Mulder, Rachel Oberman, Joshua Panganiban, Andrew Peck, Leigh Seitz, Sylvia Shea, Hannah Slevin, Rebecca Youngerman, and Lauren Hobbs. "geoBoundaries: A Global Database of Political Administrative Boundaries". In: *PLOS ONE* 15.4 (2020). DOI: 10.1371/journal.pone.0231866.

[196] Florin Rusu. "Multidimensional Array Data Management". In: *Foundations and Trends in Databases* (2023). ISSN: 1931-7883. DOI: 10.1561/1900000069. URL: https://faculty.ucmerced.edu/frusu/Papers/Report/2022-09-fntdb-arrays.pdf.

[197] David Salomon. "Raster Graphics". In: *The Computer Graphics Manual*. London: Springer London, 2011, pp. 29–134. ISBN: 978-0-85729-886-7. DOI: 10.1007/978-0-85729-886-7_2.

[198] Simon Scheider and Kai-Florian Richter. "GeoAI". In: *KI-Künstliche Intelligenz* 37.1 (2023), pp. 5–9. DOI: 10.1007/s13218-022-00797-z.

[199] Edward W Schneider. "Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis." In: (1973). URL: https://eric.ed.gov/?id=ED088424.

[200] Thomas Schneider and Mantas Šimkus. "Ontologies and Data Management: A Brief Survey". In: *KI-Künstliche Intelligenz* 34.3 (2020), pp. 329–353. DOI: 10.1007/s13218-020-00686-3.

[201] Mohammad Shahnawaz and Manish Kumar. "A Comprehensive Survey on Big Data Analytics: Characteristics, Tools and Techniques". In: *ACM Comput. Surv.* 57.8 (Mar. 2025). DOI: 10.1145/3718364.

[202] Kartik Sharma, Peeyush Kumar, and Yunqing Li. "OG-RAG: Ontology-grounded retrieval-augmented generation for large language models". In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Nov. 2025. ISBN: 979-8-89176-332-6. DOI: 10.18653/v1/2025.emnlp-main.1674.

[203] Amit P Sheth and James A Larson. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases". In: *ACM Computing Surveys (CSUR)* (1990). DOI: 10.1145/96602.96604.

[204] Renee Sieber. “GeoAI and Its Implications”. In: *International Encyclopedia of Geography*. John Wiley & Sons, Ltd, Mar. 2022, pp. 1–8. ISBN: 9781118786352. DOI: 10.1002/9781118786352.wbieg2043.

[205] Fernando Silva-Coira, José R. Paramá, Susana Ladra, Juan R. López, and Gilberto Gutiérrez. “Efficient Processing of Raster and Vector Data”. In: *PLOS ONE* 15.1 (Jan. 2020), pp. 1–35. DOI: 10.1371/journal.pone.0226943.

[206] Herbert A. Simon. *The Sciences of the Artificial*. First. MIT Press, 1969. URL: https://monoskop.org/images/9/9c/Simon_Herbert_A_The_Sciences_of_the_Artificial_3rd_ed.pdf.

[207] Samriddhi Singla and Ahmed Eldawy. “Raptor Zonal Statistics: Fully Distributed Zonal Statistics of Big Raster + Vector Data”. In: *2020 IEEE International Conference on Big Data*. 2020. DOI: 10.1109/BigData50022.2020.9377907.

[208] Samriddhi Singla, Ahmed Eldawy, Tina Diao, Ayan Mukhopadhyay, and Elia Scudiero. “The Raptor Join Operator for Processing Big Raster + Vector Data”. In: *Proc. of the 29th Int. Conf. on Advances in Geographic Information Systems*. ACM, 2021. DOI: 10.1145/3474717.3483971.

[209] John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. CA, USA: Brooks/Cole Publishing Co., 1999. ISBN: 0-534-94965-7. DOI: 10.5555/318183. URL: https://www.jfsowa.com/krbook/.

[210] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. “Bringing Relational Databases into the Semantic Web: A Survey”. In: *Semant. Web* 3.2 (Apr. 2012). ISSN: 1570-0844. DOI: 10.5555/2590194.2590199.

[211] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. “LinkedGeoData: A Core For A Web of Spatial Open Data”. In: *Semantic Web* 3.4 (2012), pp. 333–354. DOI: 10.3233/SW-2011-0052.

[212] Paolo Mazzetti Stefano Nativi and Max Craglia. “A View-based Model of Data-Cube to Support Big Earth Data Systems Interoperability”. In: *Big Earth Data* 1.1-2 (2017). DOI: 10.1080/20964471.2017.1404232.

[213] Grimm Stephan, Hitzler Pascal, and Abecker Andreas. “Knowledge Representation and Ontologies”. In: *Semantic Web Services: Concepts, Technologies, and Applications*. Springer, 2007, pp. 51–105. ISBN: 978-3-540-70894-0. DOI: 10.1007/3-540-70894-4_3.

[214] Peter Strobl, Peter Baumann, Adam Lewis, Zoltan Szantoi, Brian Killough, Matthew Purss, Max Craglia, Stefano Nativi, Alex Held, and Trevor Dhu. “The Six Faces of the Data Cube”. In: *Proceedings of the 2017 conference on big data from space, Toulouse, France*. 2017. DOI: 10.2760/383579.

[215] Kai Sun, Yunqiang Zhu, Peng Pan, Zhiwei Hou, Dongxu Wang, Weirong Li, and Jia Song. “Geospatial Data Ontology: The Semantic Foundation of Geospatial Data Integration and Sharing”. In: *Big Earth Data* 3.3 (2019), pp. 269–296. DOI: 10.1080/20964471.2019.1661662.

[216] Chien D. C. Ta and Tuoi Phan Thi. “Automatic Evaluation of the Computing Domain Ontology”. In: *Future Data and Security Engineering.* Springer, 2015, pp. 285–295. ISBN: 978-3-319-26135-5. DOI: 10.1007/978-3-319-26135-5_21.

[217] Zhenyu Tan, Peng Yue, and Jianya Gong. “An Array Database Approach for Earth Observation Data Management and Processing”. In: *ISPRS International Journal of Geo-Information* 6.7 (2017). ISSN: 2220-9964. DOI: 10.3390/ijgi6070220.

[218] Jeni Tennison. *The RDF Data Cube Vocabulary.* W3C Recommendation. url https://www.w3.org/TR/vocab-data-cube/. World Wide Web Consortium, 2014.

[219] Ba-Huy Tran, Nathalie Aussenac-Gilles, Catherine Comparot, and Cassia Trojahn. “Semantic Integration of Raster Data for Earth Observation: An RDF Dataset of Territorial Unit Versions with their Land Cover”. In: *ISPRS International Journal of Geo-Information* 9 (2020). ISSN: 2220-9964. DOI: 10.3390/ijgi9090503.

[220] Mike Uschold and Michael Gruninger. “Ontologies: Principles, methods and applications”. In: *The Knowledge Engineering Review* 11.2 (1996), pp. 93–136. DOI: 10.1017/S0269888900007797.

[221] Tiffany C. Vance, Thomas Huang, and Kevin A. Butler. “Big Data in Earth Science: Emerging Practice and Promise”. In: *Science* 383.6688 (2024). DOI: 10.1126/science.adh9607.

[222] Jovan Varga, Lorena Etcheverry, Alejandro A. Vaisman, Oscar Romero, Torben Bach Pedersen, and Christian Thomsen. “QB2OLAP: Enabling OLAP on Statistical Linked Open Data”. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE).* 2016, pp. 1346–1349. DOI: 10.1109/ICDE.2016.7498341.

[223] Raphael Volz, Joachim Kleb, and Wolfgang Mueller. “Towards Ontology-based Disambiguation of Geographical Identifiers.” In: *I3.* 2007. URL: https://ra.ethz.ch/CDstore/www2007/www2007.org/workshops/paper_132.pdf.

[224] Denny Vrandečić and Markus Krötzsch. “Wikidata: a Free Collaborative Knowledgebase”. In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489.

[225] Zhengming Wan, Simon Hook, and Glynn Hulley. *MODIS/Terra Land Surface Temperature/Emissivity Daily L3 Global 1km SIN Grid V061.* 2021. DOI: 10.5067/MODIS/MOD11A1.061.

[226] Frank Warmerdam. “The Geospatial Data Abstraction Library (GDAL)”. In: *Open Source Approaches in Spatial Data Handling*. Springer, 2008. DOI: 10.1007/978-3-540-74831-1_5.

[227] Bo Wei, Xi Guo, Xiaodi Li, Ziyan Wu, Jing Zhao, and Qiping Zou. “Construct and Query a Fine-Grained Geospatial Knowledge Graph”. In: *Data Science and Engineering* 9.2 (2024), pp. 152–176. DOI: 10.1007/s41019-023-00237-4.

[228] Stephen A White. *Introduction to BPMN*. Tech. rep. 2004. URL: http://yoann.nogues.free.fr/IMG/pdf/07-04_WP_Intro_to_BPMN_-_White-2.pdf.

[229] Mark Wick and Bernard Vatant. *The GeoNames Geographical Database*. Available at https://www.geonames.org/. 2012.

[230] WorldPop and Maksym Bondarenko. “Global 1km Population Total Adjusted To Match The Corresponding UNPD Estimate”. In: (June 2020). DOI: 10.5258/SOTON/WP00671. URL: https://eprints.soton.ac.uk/441895/.

[231] Chen Wu, Qing Zhu, Yeting Zhang, Zhiqiang Du, Xinyue Ye, Han Qin, and Yan Zhou. “A NoSQL–SQL Hybrid Organization and Management Approach for Real-Time Geospatial Data: A Case Study of Public Security Video Surveillance”. In: *ISPRS International Journal of Geo-Information* 6.1 (2017). ISSN: 2220-9964. DOI: 10.3390/ijgi6010021.

[232] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. “Ontology-Based Data Access: A Survey”. In: *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI Org., 2018. DOI: 10.24963/ijcai.2018/777.

[233] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. “Virtual Knowledge Graphs: An Overview of Systems and Use Cases”. In: *Data Intelligence* 1.3 (June 2019), pp. 201–223. ISSN: 2641-435X. DOI: 10.1162/dint{_}a{_}00011.

[234] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalaycı, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. “The Virtual Knowledge Graph System Ontop”. In: *The Semantic Web – ISWC 2020*. Springer, 2020. DOI: 10.1007/978-3-030-62466-8_17.

[235] Guohui Xiao, Diluxan Sathalingam, Albulen Pano, Yu Feng, and Linfang Ding. “ATKIS-DLM-KG: A Unified Knowledge Graph Framework For Integrating and Querying Fragmented Topographic-Cartographic Data In the German Digital Landscape Model”. In: *International Journal of Digital Earth* 18.1 (2025). DOI: 10.1080/17538947.2025.2528703.

[236] Zhihang Yao, Claus Nagel, Felix Kunde, György Hudra, Philipp Willkomm, Andreas Donaubauer, Thomas Adolphi, and Thomas H Kolbe. "3DCityDB, A 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML". In: *Open Geospatial Data, Software and Standards* 3.1 (2018), pp. 1–26. DOI: 10.1186/s40965-018-0046-7.

[237] P. Yue, K. Wang, H. Xu, J. Gong, and L. Xiang. "From Geospatial Data Cube to AI Cube: the Open Geospatial Engine (OGE) Approach". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-4-2024 (2024), pp. 441–446. DOI: 10.5194/isprs-annals-X-4-2024-441-2024.

[238] Milan Zeleny. "Management Support Systems: Towards Integrated Knowledge Management". In: *Human Systems Management* 7.1 (1987), pp. 59–70. DOI: 10.3233/HSM-1987-7108.

[239] Xianyuan Zhang, Longgang Xiang, Peng Yue, Jianya Gong, and Huayi Wu. "Open Geospatial Engine: A Cloud-based Spatiotemporal Computing Platform". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 10 (2024), pp. 453–459. DOI: 10.5194/isprs-annals-X-4-2024-453-2024.

[240] Rui Zhu. "Geospatial Knowledge Graphs". In: *International Encyclopedia of Geography*. John Wiley & Sons, Ltd, 2024, pp. 1–7. DOI: 10.1002/9781118786352.wbieg2207.

[241] Rui Zhu, Cogan Shimizu, Shirly Stephen, Colby K. Fisher, Thomas Thelen, Kitty Currier, Krzysztof Janowicz, Pascal Hitzler, Mark Schildhauer, Wenwen Li, Dean Rehberger, Adrita Barua, Antrea Christou, Ling Cai, Abhilekha Dalal, Anthony D'Onofrio, Andrew Eells, Mitchell Faulk, Zilong Liu, Gengchen Mai, Mohammad Saeid Mahdavinejad, Bryce Mecum, Sanaz Saki Norouzi, Meilin Shi, Yuanyuan Tian, Sizhe Wang, Zhangyu Wang, and Joseph Zalewski. *The KnowWhereGraph: A Large-Scale Geo-Knowledge Graph for Interdisciplinary Knowledge Discovery and Geo-Enrichment*. Feb. 2025. DOI: 10.48550/arXiv.2502.13874.

[242] Xiao Xiang Zhu, Sining Chen, Fahong Zhang, Yilei Shi, and Yuanyuan Wang. *GlobalBuildingAtlas: An Open Global and Complete Dataset of Building Polygons, Heights and LoD1 3D Models*. Dataset. 2025. DOI: 10.14459/2025mp1782307.

[243] Xiao Xiang Zhu, Sining Chen, Fahong Zhang, Yilei Shi, and Yuanyuan. Wang. "GlobalBuildingAtlas: An Open Global and Complete Dataset of Building Polygons, Heights and LoD1 3D Models". In: *Earth System Science Data* 17.12 (2025), pp. 6647–6668. DOI: 10.5194/essd-17-6647-2025. URL: https://mediatum.ub.tum.de/1782307.

Appendix A

All Queries

Legend

● Keyword | ● Identifier | ● Variable | ● Class | ● Prefix | ● User Input

Prefixes

Remember to add the following prefixes in the SPARQL queries.

Table A.1: List of prefixes used for RasSPARQL queries

Prefix	IRI Namespace
:	https://github.com/aghoshpro/OntoRaster/
gn	http://www.geonames.org/ontology#
geo	http://www.opengis.net/ont/geosparql#
geof	http://www.opengis.net/def/function/geosparql/
rdfs	http://www.w3.org/2000/01/rdf-schema#
bldg	http://www.opengis.net/citygml/building/2.0/
lgdo	http://www.linkedgeodata.org/ontology/
rasdb	https://github.com/aghoshpro/RasterDataCube/

All Queries

Q1. List all raster datasets of Sweden, along with their dimensions?

```
SELECT ?answer {
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (CONTAINS(?rasterName, 'Sweden'))
BIND (rasdb:rasDimension(?rasterName) AS ?answer)
}
```

Q2. Perform cell operation on an array of an input raster dataset at a particular timestamp with the user-specific operator & operand.

```
SELECT ?answer {
?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
FILTER (CONTAINS(?rasterName, 'Bavaria'))
BIND ('*' AS ?operator)
BIND (0.02 AS ?operand)
BIND ('2022-04-18T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
BIND (rasdb:rasCellOp(?timeStamp, ?operator, ?operand, ?rasterName) AS ?
    answer)
}
```

Q3. Find the spatial average temperature over districts of München on 24 July 2023, using the available temperature raster data over Bavaria.

```
SELECT ?tempK {
  ?region a :Region_District_Munich ; rdfs:label ?regionName ; geo:asWKT ?
      regionWkt .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (CONTAINS(?rasterName, 'Bavaria')
  BIND ('2023-07-24T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName)
      AS ?tempK)
}
```

Q4. Find all districts of Munich and the respective spatial average vegetation on 16 August 2023.

```
SELECT ?ndvi {
  ?region a :Region_District_Munich ; rdfs:label ?regionName ; geo:asWKT ?
      regionWkt .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (CONTAINS(?rasterName, 'NDVI')
  BIND ('2023-08-16T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialMaximum(?timeStamp, ?regionWkt, ?rasterName)
      AS ?ndvi)
}
```

Q5. Find all districts of Munich and the respective spatial average soil moisture on 20 January 2023

```
SELECT ?soilmoisture {
  ?region a :Region_District_Munich ; rdfs:label ?regionName ; geo:asWKT ?
      regionWkt .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (CONTAINS(?rasterName, 'Munich_ECOSTRESS_SoilMoisture_70m')
  BIND ('2023-01-20T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialMinimum(?timeStamp, ?regionWkt, ?rasterName)
      AS ?soilmoisture)
}
```

Q6. Find spatial-temporal average temperature over Göteborg, Sweden, between 5 April 2022 and 19 June 2022.

```
SELECT ?tempK {
  ?region a :Region_Sweden ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
      .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName = 'Göteborg')
  FILTER (CONTAINS(?rasterName, 'Sweden'))
  BIND ('2022-04-05T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
  BIND ('2022-06-19T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
  BIND (rasdb:rasTemporalAverage(?startTimeStamp, ?endTimeStamp,
        ?regionWkt, ?rasterName) AS ?tempK)
}
```

Q7. Find the maximum temperature over Göteborg, Sweden, between 5 April 2022 and 19 June 2022.

```
SELECT ?tempK {
  ?region a :Region_Sweden ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
       .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName = 'Göteborg')
  FILTER (CONTAINS(?rasterName, 'Sweden'))
  BIND ('2022-04-05T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
  BIND ('2022-06-19T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
  BIND (rasdb:rasTemporalMaximum(?startTimeStamp, ?endTimeStamp,
          ?regionWkt, ?rasterName) AS ?tempK)
}
```

Q8. Find the minimum temperature over Göteborg, Sweden, between 5 April 2022 and 19 June 2022.

```
SELECT ?tempK {
  ?region a :Region_Sweden ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
       .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName = 'Göteborg')
  FILTER (CONTAINS(?rasterName, 'Sweden'))
  BIND ('2022-04-05T00:00:00+00:00'^^xsd:dateTime AS ?startTimeStamp)
  BIND ('2022-06-19T00:00:00+00:00'^^xsd:dateTime AS ?endTimeStamp)
  BIND (rasdb:rasTemporalMinimum(?startTimeStamp, ?endTimeStamp,
          ?regionWkt, ?rasterName) AS ?tempK)
}
```

Q9. Clip an available temperature raster data for Bolzano, South Tyrol on 24 September 2023, and return filtered arrays.

```
SELECT ?clippedArray {
  ?region a :Region_Tyrol ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
      .
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (?regionName = 'Bolzano')
  FILTER (CONTAINS(?rasterName, 'Tyrol'))
  BIND ('2023-09-24T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasClipRaster(?timeStamp, ?regionWkt, ?rasterName)
      AS ?clippedArray)
}
```

Q10. Clip a portion of user-specific raster data based on a custom vector region on 24 July 2024, and return filtered arrays.

```
SELECT ?clippedArray {
  ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  FILTER (CONTAINS(?rasterName, 'Bavaria'))
  BIND ('POLYGON((11.324157714843748 48.29050321714061,
          11.911926269531248 48.279537342260085,
          11.88995361328125 48.01932418480118,
          11.340637207031248 48.01564978668938,
        11.324157714843748 48.29050321714061))' AS ?customRegionWkt)
  BIND ('2023-07-24T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasClipRasterAnyGeom(?timeStamp, ?customRegionWkt,
        ?rasterName) AS ?clippedArray)
}
```

Q11. Find all districts of Munich where the spatial average elevation is more than 515 meters.

```
SELECT ?distName ?elevation {
    ?region a :Region_District_Munich .
    ?region rdfs:label ?distName .
    ?region geo:asWKT ?distWkt .
    ?gridCoverage a :Raster .
    ?gridCoverage rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Elevation'))
    BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
        elevation)
    FILTER(?elevation > 515)
}
```

Q12. Find all sub-districts of Munich where the spatial average elevation is more than 515 meters.

```
SELECT ?distName ?elevation {
    ?region a :Region_SubDistrict_Munich .
    ?region rdfs:label ?distName .
    ?region geo:asWKT ?distWkt .
    ?gridCoverage a :Raster .
    ?gridCoverage rasdb:rasterName ?rasterName .
    FILTER (CONTAINS(?rasterName, 'Elevation'))
    BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
    BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
        elevation)
    FILTER(?elevation > 515)
}
```

Q13. Find all metro stations within those districts in Bavaria whose spatial average land elevation is more than 520 meters.

```
SELECT ?featureName ?regionName ?elevation ?pointWkt {
 ?region a :Region_Bavaria ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
    .
 ?gname a gn:MTRO; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Elevation'))
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS ?
     elevation)
 FILTER (?elevation > 520)
}
```

Q14. Find all schools within those districts in Bavaria whose average greenery (NDVI) is low.

```
SELECT ?featureName ?regionName ?ndvi ?pointWkt {
 ?region a :Region_Bavaria ; rdfs:label ?regionName ; geo:asWKT ?regionWkt
     .
 ?gname a gn:SCH; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'NDVI'))
 BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS ?
      ndvi)
 FILTER(?ndvi < 0.2)
}
```

Q15. Find all radio observatories within those municipalities in South Tyrol whose average elevation is more than 1000 meters.

```
SELECT ?featureName ?regionName ?elevation ?pointWkt {
 ?region a :Region_Sweden; rdfs:label ?regionName ; geo:asWKT ?regionWkt .
 ?gname a gn:OBSR; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'NDVI'))
 BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS ?
      elevation)
 FILTER(?elevation > 1000)
}
```

Q16. Find all farmlands within those municipalities in Sweden whose average soil moisture is high.

```
SELECT ?featureName ?regionName ?soilmoisture ?pointWkt {
 ?region a :Region_Sweden; rdfs:label ?regionName ; geo:asWKT ?regionWkt .
 ?gname a gn:FRM; rdfs:label ?featureName ; geo:asWKT ?pointWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
    FILTER (geof:sfWithin(?pointWkt, ?regionWkt))
 FILTER (CONTAINS(?rasterName, 'Munich_ECOSTRESS_SoilMoisture_70m'))
 BIND ('2022-01-01T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?regionWkt, ?rasterName) AS ?
      soilmoisture)
 FILTER(?soilmoisture > 0.5)
}
```

Q17. Find all hotels and respective districts of Bavaria whose spatial average elevation is above 500 meters

```
SELECT ?bldgName ?distName ?elevation ?distWkt ?distWktColor ?bldgWkt
                ?bldgWktColor {
 ?building a lgdo:Hotel . # Church, Parking, Hospital, Hotel
    ?building rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
 ?region a :Region_Bavaria ; rdfs:label ?distName ; geo:asWKT ?distWkt .
```

```
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND('#181818' AS ?distWktColor)
 BIND('red' AS ?bldgWktColor)
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
     elevation)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
    FILTER (CONTAINS(?rasterName, 'Elevation'))
 FILTER(?elevation > 500)
}
```

Q18. Find all parking spots and corresponding districts of South Tyrol whose spatial average elevation is above 520 meters

```
SELECT ?bldgName ?distName ?elevation ?distWkt ?distWktColor ?bldgWkt
                ?bldgWktColor {
 ?building a lgdo:Parking .
    ?building rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
 ?region a :Region_Tyrol ; rdfs:label ?distName ; geo:asWKT ?distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND('#181818' AS ?distWktColor)
 BIND('red' AS ?bldgWktColor)
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
     elevation)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
 FILTER (CONTAINS(?rasterName, 'Elevation'))
 FILTER(?elevation > 520)
}
```

Q19. Find all bridges and corresponding towns of Sweden whose spatial average elevation is above 400 meters

```
SELECT ?bldgName ?distName ?elevation ?distWkt ?distWktColor ?bldgWkt
                ?bldgWktColor {
 ?building a lgdo:Bridge .
    ?building rdfs:label ?bldgName ; geo:asWKT ?bldgWkt .
 ?region a :Region_Sweden ; rdfs:label ?distName ; geo:asWKT ?distWkt .
    ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
 BIND('#181818' AS ?distWktColor)
 BIND('red' AS ?bldgWktColor)
 BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
 BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
     elevation)
    FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
 FILTER (CONTAINS(?rasterName, 'Elevation'))
 FILTER(?elevation > 400)
}
```

Q20. Find the addresses of all residential buildings in Munich that are taller than 100 meters, along with their respective sub-districts and their average elevation

```
SELECT ?bldgName ?bldgHeight ?distName ?elevation {
 ?building a lgdo:School ; rdfs:label ?bldgName ; geo:asWKT ?bldgWkt ; owl
      :sameAs ?citygml3D .
 ?citygml3D bldg:measuredHeight ?bldgHeight .
 ?region a :Region_SubDistrict_Munich ; rdfs:label ?distName ; geo:asWKT ?
      distWkt .
 ?gridCoverage a :Raster ; rasdb:rasterName ?rasterName .
  BIND ('2000-02-11T00:00:00+00:00'^^xsd:dateTime AS ?timeStamp)
  BIND (rasdb:rasSpatialAverage(?timeStamp, ?distWkt, ?rasterName) AS ?
       elevation)
  FILTER (geof:sfWithin(?bldgWkt, ?distWkt))
  FILTER (CONTAINS(?rasterName, 'Elevation'))
  FILTER(?bldgHeight > 100) .
}
```