INTERNATIONAL DOCTORAL
SCHOOL OF THE USC

Shahed
Bassam Al-Mobydeen

PhD Thesis

EFFICIENT ACCESS TO HETEROGENEOUS ENVIRONMENTAL DATA REPOSITORIES THROUGH LINKED DATA STANDARDS

Santiago de Compostela, 2022

Doctoral Programme in Information Technology Research

TESE DE DOUTORAMENTO

EFFICIENT ACCESS TO HETEROGENEOUS ENVIRONMENTAL DATA REPOSITORIES THROUGH LINKED DATA STANDARDS

Shahed Bassam Al-Mobydeen

ESCOLA DE DOUTORAMENTO INTERNACIONAL DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA
2022

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DECLARACIÓN DO AUTOR/A DA TESE

D./Dna. **Shahed Bassam Al-Mobydeen**

Título da tese: **Efficient access to heterogeneous environmental data repositories through linked data standards**

Presento a miña tese, seguindo o procedemento axeitado ao Regulamento, e declaro que:

1) A tese abarca os resultados da elaboración do meu traballo.
2) De ser o caso, na tese faise referencia ás colaboracións que tivo este traballo.
3) Confirmo que a tese non incorre en ningún tipo de plaxio doutros autores nin de traballos presentados por min para a obtención doutros títulos.
4) A tese é a versión definitiva presentada para a súa defensa e coincide a versión impresa coa presentada en formato electrónico

E comprométome a presentar o Compromiso Documental de Supervisión no caso de que o orixinal non estea na Escola.

En **Santiago De Compostela**, **20 de Xullo de 2022**.

Sinatura electrónica

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE

D./Dna. **José Ramón Ríos Viqueira**

En condición de: **Titor/a e director/a**

Título da tese: **Efficient access to heterogeneous environmental data repositories through linked data standards**

INFORMA:

Que a presente tese, correspóndese co traballo realizado por D/Dna Shahed Bassam Al-Mobydeen, baixo a miña dirección/titorización, e autorizo a súa presentación, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como director/titor desta non incorre nas causas de abstención establecidas na Lei 40/2015.

En **Santaigo De Compostela**, **20 de Xullo de 2022**

Sinatura electrónica

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE

D./Dna. **Manuel Lama Penín**

En condición de: **Director/a**

Título da tese: **Efficient access to heterogeneous environmental data repositories through linked data standards**

INFORMA:

Que a presente tese, correspóndese co traballo realizado por D/Dna Shahed Bassam Al-Mobydeen, baixo a miña dirección/titorización, e autorizo a súa presentación, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como director/titor desta non incorre nas causas de abstención establecidas na Lei 40/2015.

En **Santaigo De Compostela**, **20 de Xullo de 2022**

Sinatura electrónica

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Dedication

I am dedicating this Thesis to four beloved people who have meant and continue to mean so much to me. To my father, mother, wife, and daughter

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acknowledgments

I would like to express my special thanks and appreciation to my supervisor Prof. José Ramón Ríos Viqueira for the patient guidance, encouragement and advice he has provided. I have benefited greatly from his wealth of knowledge and meticulous editing. I am extremely grateful to have a supervisor who cared so much about my work, and who responded to my questions so promptly.

I would like also to express my special thanks to my supervisor prof. Manuel Lama Penín for his support and guidance and advice he has provided.

Thanks and great appreciation to my father and mother, your prayers for me were what helped me thus far.

I also would like to thank my daughter for giving me unlimited happiness and pleasure.

Finally, of course, is my inspiration, my world: without my wife, I may never have completed this thesis. To have a woman of such infinite talent and dynamism put her own professional career on hold to support the dreams of the man she's married is a powerful thing.

The work was partially funded by European Union under the Erasmus Mundus Peace II mobility program.

20 de xullo de 2022

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumo

O coñecemento preciso do noso medio é tanto entrada como saída para moitos traballos en diferentes disciplinas da ciencia e da enxeñaría, con resultados que teñen un gran impacto nas nosas condicións de vida. Algúns exemplos importantes de problemas que consumen e/ou xeran datos ambientais son a investigación do cambio climático, a vixilancia e control da calidade do aire, a predición meteorolóxica, o control de enfermidades, etc. Por iso, as administracións públicas e os organismos relacionados coa investigación invisten grandes cantidades de recursos na construción e mantemento de grandes infraestruturas de observación e modelización do medio.

Os datos xerados por estas infraestruturas son moi heteroxéneos e, en función da natureza da súa dimensión espacial, podemos clasificalos en dous grandes grupos: Entidades e Coberturas Xeoespaciais. Rexístranse as propiedades convencionais e xeométricas das entidades ou obxectos. As propiedades xeométricas definen a localización e a forma da entidade no espazo xeográfico. Exemplos de entidades xeoespaciais son concellos, estradas, cidades, ríos, etc. As coberturas rexistran a variación, continua ou discreta, no espazo e/ou no tempo, de propiedades de tipo convencional. Un exemplo de cobertura xeoespacial rexistraría a variación das propiedades meteorolóxicas, como a temperatura do aire ou a precipitación acumulada, nunha rexión do espazo e nun período de tempo.

As infraestruturas de datos espaciais e ambientais dispoñibles na actualidade proporcionan sistemas de almacenamento dos datos anteriormente descritos e tamén servizos que permiten o descubrimento de conxuntos de datos e o seu acceso a través da web. Estes servizos baséanse en estándares específicos destas comunidades e o seu uso está restrinxido xeralmente a comunidades científicas ou expertos no campo dos Sistemas de Información Xeográfica (GIS). Por outra banda, as infraestruturas de datos abertos de propósito xeral están evolucionando cara á construción da web de datos enlazados, combinando o uso de identificadores de recursos

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

universais (URI), co protocolo HTTP para acceder a eles e co uso de estándares da web semántica para representalos e vinculalos. Estes estándares inclúen o RDF (Resource Description Framework) para a representación de datos e o protocolo e linguaxe SPARQL para o acceso declarativo e a consulta.

Moitos dominios de aplicación beneficiaríanse de incorporar as grandes cantidades de datos xerados polas fontes de datos ambientais en infraestruturas web de datos enlazados. En xeral, isto permitiría aos expertos en TIC achegar os grandes esforzos realizados pola comunidade científica no ámbito da observación e modelización do medio a moitos máis problemas da cidadanía, mellorando o aproveitamento que se fai destes recursos e infraestruturas tan difíciles de crear e manter. Un exemplo sería o uso que se pode facer no ámbito do turismo e das actividades ao aire libre da información xerada polos modelos meteorolóxicos. Para conseguilo, hai que adaptar as tecnoloxías da web semántica para que poidan xestionar de forma eficiente os datos de carácter xeoespacial. O estándar GeoSPARQL xa permite a consulta de datos de entidades xeoespaciais cunha eficiencia razoable. Non obstante, non existen solucións no ámbito da web semántica co nivel de efectividade e eficiencia requiridos que permitan a consulta eficiente dos datos de coberturas xeoespaciais, e moito menos a consulta integrada de datos de entidades e coberturas. Por todo o anterior, o obxectivo desta Tese de doutoramento é proporcionar avances na construción dun motor de consulta GeoSPARQL que permita o acceso e consulta eficientes a fontes de datos ambientais de carácter heteroxéneo, ás que se poderá acceder a través dos servizos estandarizados dunha infraestrutura de datos xeoespaciais.

Estado da arte

As fontes de datos de observación ambiental pódense clasificar segundo as características do proceso que xera os datos. Hai procesos simples e procesos compostos, que se constrúen encadeando resultados doutros procesos simples ou compostos. Se un proceso ten algún compoñente físico (normalmente un sensor), entón as súas observacións poden ser observacións in situ ou observacións remotas. Os procesos de observación in situ xeran datos sobre entidades situadas moi preto do proceso. Por outra banda, os procesos de observación remota, xeran observacións sobre entidades que están a certa distancia do proceso de observación. Son exemplos de observacións in situ as xeradas polas estacións de observación ambiental, con sensores que miden variables nos puntos nos que se atopan os propios sensores. Exemplos de observacións remotas son as xeradas por dispositivos de radar terrestres ou dispositivos

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

remotos montados a bordo de satélites ou aeronaves. Os procesos de xeración de observacións ambientais nos que participan expertos están agora apoiados no uso de tecnoloxías baseadas en dispositivos móbiles. Un paso máis é o uso dos sentidos dos cidadáns e dos dispositivos móbiles para xerar grandes cantidades de datos, en xeral menos fiables, pero que proporcionan unha cobertura espacial e temporal moito maior. A obtención de datos observados polos cidadáns, directa ou indirectamente, denomínase crowdsensing, e xa se está a utilizar para observar variables de distinto tipo, como a calidade do aire ou o ruído ambiental. Unha parte importante das fontes de datos de tipo crowdsensing son aquelas que supoñen a análise do contido das redes sociais, dado o gran volume de información que estas redes xeran de forma continuada.

Os datos xerados polos procesos de observación compleméntanse cos resultados dos modelos ambientais. Estes modelos poden ter a forma de modelos físicos a escala do medio real, como un túnel de vento, ou ser modelos matemáticos que tentan aproximar o comportamento real do sistema con conxuntos de expresións matemáticas. Estes modelos matemáticos poden ser de natureza empírica, onde o modelo se xera mediante un proceso de xeneralización dun conxunto de observacións sobre as que se adestra o modelo. Os modelos matemáticos tamén se poden basear na simulación matemática do sistema físico. Estes últimos úsanse xeralmente para a predición do tempo, por exemplo. Os modelos matemáticos pódense utilizar para predicir condicións futuras, pero tamén para estimar as condicións actuais ou pasadas en lugares que non foron observados.

Os modelos de datos conceptuais utilizados tradicionalmente como base para a construción de sistemas de almacenamento e xestión baséanse na definición de entidades ou obxectos, que se agrupan en tipos ou clases e teñen propiedades sobre as que se rexistran os datos. Seguindo esta idea, o Open Geospatial Consortium (OGC) propón un modelo de referencia no que as entidades xeoespaciais (chamadas Features) poden ter propiedades de tipos de datos convencionais (alfanuméricos) e propiedades de tipos de datos xeométricos, que definen a súa localización e forma sobre o espazo xeográfico. Non obstante, dado que nalgúns casos as propiedades convencionais dunha Feature poden variar ao longo da súa extensión espacial e/ou temporal, a OGC define un tipo especial de Feature, denominada Coverage, que permite representar a variación, continua ou discreta, de unha ou varias propiedades de tipos convencionais ao longo do espazo e/ou do tempo. O modelo de datos que hai debaixo do concepto de Coverage é un modelo multidimensional, similar aos utilizados no ámbito dos almacéns de datos. As representacións dixitais do espazo utilizadas para codificar os valores

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

das propiedades xeométricas das Features baséanse no uso de vectores de coordenadas dos bordes dos obxectos xeométricos (puntos, liñas e polígonos). É por iso que chamaremos a estas Features representadas con xeometrías vectoriais Vector Features. Por outra banda, as representacións dixitais das Coverages baséanse na partición do espazo en celas, para as que se define algún tipo de método de estimación da variación das propiedades a representar. A representación máis utilizada baséase na división do espazo nunha cuadrícula de celas, e no uso dun valor constante da propiedade en cada cela. Esta representación coñécese como representación ráster e é por iso que chamaremos a estas estruturas de datos Raster Coverages.

O OGC define formatos estándar e codificacións baseándose nos modelos conceptuais mencionados anteriormente. En concreto, os formatos de almacenamento de datos para Vector Features baséanse nunha simplificación do modelo proposto polp OGC para a representación de xeometrías vectoriais, denominado Simple Feature Access (SFA). Este modelo define xeometrías simples (Point, Linestring e Polygons) e coleccións de xeometrías simples de tipo heteroxéneo (GeometryCollection) ou de tipos homoxéneos (MultiPoint, MultiLineString, MultiPolygon). Este modelo, que tamén define un amplo conxunto de funcións para a xestión de xeometrías, é tamén a base para a implementación de extensións espaciais dos Sistemas de Xestión de Bases de Datos (SXBD). Por outra banda, unha das codificacións máis utilizadas para as coberturas ráster é o formato GeoTIFF, que utiliza as cabeceiras do formato de imaxe TIFFF para representar os datos necesarios para xeorreferenciar a imaxe (punto da esquina inferior esquerda, resolucións en ambos eixes, número de celas en ambos eixes, etc.). Nas aplicacións ambientais, as coberturas ráster adoitan representarse con formatos de almacenamento de arrays xeocientíficos. O exemplo máis coñecido é o formato NetCDF. Tamén hai exemplos de sistemas de xestión de datos de array ou ráster, porén, a súa madurez e implementación non son para nada comparables á das extensións espaciais dos SXBDs.

As infraestruturas de datos xeoespaciais e ambientais proporcionan sistemas de almacenamento baseados nos modelos e codificacións descritos anteriormente e implementan servizos que permiten o descubrimento e o acceso a datos a través da web. Se nos centramos no acceso aos datos, diferentes estándares de servizo encárganse do acceso aos datos segundo estean en forma de coleccións de Vector Features ou Raster Coverages. En concreto, o servizo máis utilizado para proporcionar acceso web a Vector Features é o Web Feature Service (SFS), definido pola OGC. Este é o servizo tamén recomendado pola directiva INSPIRE, que regula a creación da Infraestrutura Europea de Datos Espaciais. Este servizo permite recuperar entidades xeoespaciais dispoñibles nun almacén, admitindo a especificación de filtros

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

para consulta, que inclúen filtros sobre propiedades convencionais, pero tamén filtros sobre propiedades xeométricas que fan uso de predicados topolóxicos. Así, por exemplo, a través dun WFS podemos consultar os concellos almacenados que conteñen un determinado punto polo que pretendemos filtrar. En canto ao acceso ás Coberturas Raster, no ámbito dos datos ambientais, o máis habitual é utilizar un servidor TDS (THREDDS Data Server), que permite o acceso e filtrado de datos representados en formatos de arrays científicos, como o NetCDF. Non obstante, no ámbito máis xeral dos datos xeoespaciais, o servizo máis utilizado é o Web Coverage Service (WCS) definido polo OGC. Este é o servizo recomendado por INSPIRE. Este servizo da acceso aos datos dunha cobertura xeoespacial, permitindo o filtrado polas súas dimensións xeoespacial e temporal. Un servizo que vai un pouco máis aló do WCS no acceso aos datos de coberturas é o WCPS (Web Coverage Processing Service), tamén definido polo OGC e que proporciona capacidades declarativas de procesamento de datos.

Por outra banda, as infraestruturas de datos abertos de propósito xeral basean as súas implementacións en estándares de modelos, formatos e puntos de acceso definidos no ámbito da web semántica. As representacións de datos baséanse no modelo de datos proposto dentro do estándar RDF (Resource Description Framework) do consorcio W3C. En RDF, os datos represéntanse por triplas (suxeito predicado obxecto). O suxeito representa un recurso (cousa, obxecto, etc.) identificado universalmente por un IRI (Internationalized Resource Identifier). O obxecto pode ser outro recurso ou un literal (valor) dun tipo de datos (enteiro, real, cadea, temporal, etc.). O predicado, que etiqueta a natureza da relación entre o suxeito e o obxecto, tamén se identifica cun IRI. Ademais deste modelo básico, pódense construír esquemas complexos que definen clases e instancias de obxectos, e relacións entre clases e instancias de varios tipos, incluíndo xerarquías de herdanza/especialización entre clases. Estes esquemas pódense definir dun xeito sinxelo utilizando o estándar RDF Schema ou dun xeito máis complexo coa creación de ontoloxías representadas en OWL (Web Ontology Language). O W3C tamén define ontoloxías de tipo xeral que se poden utilizar para a representación uniforme de datos en campos específicos, e que poden facilitar a integración semántica de diferentes conxuntos de datos mediante o uso de ditos vocabularios comúns. O protocolo de acceso a datos web e a linguaxe de consulta de datos declarativos RDF están definidos no estándar W3C SPARQL. O W3C tamén define solucións para mapear, ben directamente ou con mapeos definidos polo usuario, o modelo de datos relacionais a RDF, co obxectivo de facilitar a integración das fontes de datos representadas co modelo relacional, que é con diferenza o modelo máis utilizado na práctica, dentro da web dos datos enlazados.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

A gran maioría dos esforzos para a integración de datos espaciais e ambientais dentro das tecnoloxías da web semántica tiveron que ver coa definición de ontoloxías de alto nivel. Así, dentro do estándar GeoSPARQL, o OGC define unha ontoloxía para a representación xeral dos datos de Vector Features, seguindo o modelo de xeometría simplificado SFA tamén definido polo OGC. Non existen solucións concretas para a representación de coberturas ráster en RDF, non obstante, con certas limitacións, poderíase utilizar o estándar RDF Cube, deseñado para datos multidimensionais. A consulta de datos de Vector Features pódese facer utilizando implementacións GeoSPARQL existentes cun rendemento razoable. Porén, ainda son necesarios esforzos para incorporar nos motores de consulta todos os avances en indexación, procesamento de consultas e optimización desenvolvidos no dominio da base de datos espaciais. Non obstante, as solucións dispoñibles para consultar datos de coberturas ráster aínda son inmaduras e sofren problemas de rendemento e usabilidade. En concreto, a principal característica destas solucións é que incorporan un tipo específico de datos para representar coberturas, que serve para almacenar as teselas de unha cobertura (zonas). Para xestionar estes mosaicos, o programador ten que facer uso dunha longa lista de funcións definidas para este tipo de datos. Este enfoque, que xa proporcionan algúns SXBDs espaciais, demostrou ser ineficiente en comparación con sistemas e formatos específicos para xestionar arrays. Como resumo desta exposición do estado da arte, podemos dicir que GeoSPARQL proporciona o estándar axeitado para a consulta de datos xeoespaciais na web dos datos enlazados, non obstante, as implementacións actuais do estándar non resolven de forma eficaz e eficiente a consulta de datos de Raster Coverages, e polo tanto, tampouco a consulta integrada de todo tipo de datos xeoespaciais e ambientais dispoñibles nas fontes de datos de observación e modelado ambiental.

Obxectivo e achegas da tese

O obxectivo principal desta Tese é contribuír ao desenvolvemento de tecnoloxías da web semántica que permitan un acceso eficiente a repositorios heteroxéneos de datos ambientais mediante estándares da web dos datos enlazados. En concreto, na Tese desenvólvense algúns compoñentes específicos dun motor de consulta GeoSPARQL que melloran o rendemento das avaliacións de consultas cando se accede ás Vector Features e ás Raster Coverages mediante servizos web xeoespaciais estandarizados. As principais características da solución proposta pódense resumir do seguinte xeito.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- O sistema permite a consulta de Vector Features accesibles a través dun servidor de datos xeoespaciais que implementa o estándar OGC WFS. A optimización de consultas implementada polo motor de consultas é capaz de delegar o filtrado convencional e xeoespacial dos datos no servizo WFS. O mapeado das Vector Features a RDF baséase na solución de mapeo directo do modelo relacional a RDF proposta polo W3C.

- O sistema permite a consulta de Raster Coverages accesibles a través dun servidor xeoespacial que implementa o estándar OGC WCPS. A optimización de consultas consegue delegar o filtrado das coberturas polas súas dimensións espaciais e/ou temporais no servizo WCPS. O mapeo entre as Raster Coverages e o RDF realízase mediante unha nova proposta de solucións de mapeado definida na Tese.

- O sistema implementa operacións de unión e agrupación que poden ser utilizadas polo motor de consulta para xerar plans de consulta eficientes que combinen Vector Features e Raster Coverages.

O sistema resultante, chamado GeoLD, implementouse como unha extensión do procesador SPARQL de Apache Jena. O servidor xeoespacial utilizado para o desenvolvemento e avaliación da solución baséase na combinación de Geoserver, para a implementación do servizo WFS, e Petascope para a implementación do servizo WCPS.

As principais achegas da Tese, máis aló do estado da arte das tecnoloxías da web semántica para datos xeoespaciais e ambientais, son as seguintes.

- Un novo enfoque para mapear Raster Coverages a RDF, denominado C2RML (Coverage to RDF Mapping Language), que se basea na linguaxe R2RML (Relational to RDF Mapping Language) definida polo W3C. Ao contrario do enfoque seguido polas extensións actuais de SPARQL para datos ráster, C2RML non require a definición dun tipo de datos ráster, xa que o RDF xérase a nivel de cela do propio ráster.

- A definición de novos operadores SPARQL que permiten un acceso eficiente ás Vector Features e ás Raster Coverages dispoñibles a través de servizos que implementan os estándares OGC WFS e WCPS.

- A definición dunha estratexia de optimización de consultas que é capaz de identificar as subárbores SPARQL máximas que poden ser substituídas por instancias dos operadores recentemente definidos, que permiten un acceso eficiente aos datos. O optimizador

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

debe procurar delegar o maior procesamento posible nos servizos web, seguindo así a heurística de realizar o maior procesamento posible o máis preto posible dos datos. Os resultados das avaliacións realizadas amosan que GeoLD mellora o rendemento das soluciones existentes no estado do arte.

- A definición e implantación de novos operadores de Join e Agrupación que permiten a agregación eficiente de datos de Raster Coverages nas zonas definidas por Vector Features. En concreto, os plans de consulta que utilizan o novo operador Join execútanse unha orde de magnitude máis rápido que os plans de consulta obtidos cos operadores Join convencionais de SPARQL. Ademais, o uso do novo operador Agrupación consegue plans de consulta que funcionan entre unha e dúas ordes de magnitude máis rápido que os plans xerados co novo operador de Join.

Publicacións

Algunhas das achegas da Tese xa foron publicadas nos seguintes artigos e congresos

Revista internacional: Shahed Bassam Almobydeen, José R.R.Viqueira, Manuel Lama. GeoSPARQL query support for scientific raster array data. Computers & Geosciences 159:105023. Elsevier. 2022. doi: https://doi.org/10.1016/j.cageo.2021.105023.

Congreso internacional: Shahed Bassam Almobydeen, Khalid Alkharabsheh, José RR Viqueira, Manuel Lama Penín. Environmental Linked Data: Challenges. 1st International Conference in computer sciences and information systems researchers. January 12-13, 2016, Amman, Jordan.

Congreso nacional: Shahed Almobydeen, José R.R. Viqueira, Manuel Lama Penín. A Federated Approach for Array and Entity Environmental Linked Data. XXI Jornadas de Ingeniería del Software y Bases de Datos, V Congreso Español de Informática. 14-16 de Septiembre de 2016. Salamanca.

O seguinte artigo que publicou o autor da Tese durante o período de realizado da mesma corresonde a traballo fora do ámbito da Tese

Revista internacional: Khalid Alkharabsheh, Shahed Almobydeen, Jose A Taboada, Yania Crespo. Influence of nominal project knowledge in the detection of design smells:

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

An exploratory study with god class. International Journal of Advanced Studies in Computers, Science and Engineering 5(11):120-127. 2016.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumen

El conocimiento preciso de nuestro medioambiente es tanto entrada como salida para muchos trabajos en distintas disciplinas de las ciencias e ingenierías, cuyos resultados tienen un gran impacto en nuestras condiciones de vida. Algunos ejemplos importantes de problemas que consumen y/o generan datos medioambientales son la investigación en cambio climático, la monitorización y control de la calidad del aire, la predicción meteorológica, el control de enfermedades, etc. Por lo tanto, las administraciones púbicas y las organizaciones relacionadas con la investigación invierten grandes cantidades de recursos en la construcción y mantenimiento de grandes infraestructuras de observación y modelado del medioambiente.

Los datos generados por dichas infraestructuras son altamente heterogéneos, y en base a la naturaleza de su dimensión espacial podemos clasificarlos en dos grandes grupos: Entidades y Coberturas geoespaciales. De las entidades u objetos se registran propiedades de tipos convencionales y geométricas (definen la localización y forma de la entidad en el espacio geográfico). Ejemplos de entidades geoespaciales son municipios, carreteras, ciudades, ríos, etc. Las coberturas registran la variación, continua o discreta, en el espacio y/o en el tiempo, de propiedades de tipo convencional. Un ejemplo de una cobertura geoespacial registraría la variación de propiedades meteorológicas, como la temperatura del aire o la precipitación acumulada, a lo largo de una región del espacio y a lo largo de un período de tiempo.

Las infraestructuras de datos espaciales y medioambientales disponibles en la actualidad proporcionan sistemas de almacenamiento para los datos arriba descritos y también servicios que permiten el descubrimiento de conjuntos de datos y su acceso a través de la web. Estos servicios están basados en estándares propios de estas comunidades y su uso están en general restringido a comunidades científicas o expertos del ámbito de los Sistemas de Información Geográfica (SIG). Por otro lado, las infraestructuras de datos abiertos de propósito general están evolucionando hacia la construcción de la web de los datos enlazados, combinando el

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

uso de identificadores universales de los recursos (URI), con el protocolo HTTP para acceder a los mismos y con el uso de estándares de la web semántica para representarlos y enlazarlos. Estos estándares incluyen el RDF (Resource Description Framework) para la representación de los datos y el protocolo y lenguaje SPARQL para su acceso y consulta declarativa.

Muchos dominios de aplicación se beneficiarían de la incorporación de las enormes cantidades de datos generadas por las fuentes de datos medioambientales en las infraestructuras de la web de los datos enlazados. En general, esto permitiría a los expertos en TIC acercar los grandes esfuerzos realizados por la comunidad científica en el ámbito de la observación y modelado del medioambiente a muchos más problemas de la ciudadanía, mejorando el uso que se hace de estos recursos e infraestructuras tan difíciles de crear y mantener. Un ejemplo sería el uso que se puede hacer en el ámbito del turismo y de las actividades al aire libre de la información generada por los modelos meteorológicos. Para conseguir esto, las tecnologías del ámbito de la web semántica deben de ser adaptadas para que puedan gestionar de forma eficiente datos de naturaleza geoespacial. El estándar GeoSPARQL ya permite la consulta de datos de entidades geoespaciales con una eficiencia razonable. Sin embargo, no existen soluciones en el ámbito de la web semántica con el nivel requerido de eficacia y eficiencia que permitan la consulta eficiente de datos de coberturas geoespaciales, y mucho menos la consulta integrada de datos de entidades y coberturas.

Por todo lo anterior, el objetivo de esta Tesis de doctorado es el de proporcionar avances hacia la construcción de un motor de consultas GeoSPARQL que permita el acceso y consulta eficiente de fuentes de datos medioambientales de naturaleza heterogénea, que estarán accesibles a través de los servicios estandarizados de una infraestructura de datos geoespaciales.

Estado del arte

Las fuentes de datos de observación medioambiental pueden clasificarse según las características del proceso que genera los datos. Existen procesos simples y procesos compuestos, que se construyen encadenando resultados de otros procesos simples o compuestos. Si un proceso tiene algún componente físico (típicamente un sensor), entonces sus observaciones pueden ser observaciones in-situ u observaciones remotas. Los procesos de observación in situ generan datos sobre entidades localizadas muy cerca del proceso. Por otro lado, los procesos de observación remota generan observaciones sobre entidades que se encuentran a cierta distancia del proceso de observación. Ejemplos de observaciones in-situ son las generadas por las

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

estaciones de observación medioambiental, cuyos sensores miden variables en los puntos en los que se encuentran los propios sensores. Ejemplos de observaciones remotas son aquellas generadas por dispositivos de radar instalados en tierra o dispositivos remotos montados a bordo de satélites o aeronaves. Los procesos de generación de observaciones medioambientales en los que participan expertos son ahora apoyados por el uso de tecnologías basadas en dispositivos móviles. Un paso más allá es el uso de los sentidos y de los dispositivos móviles de los ciudadanos para generar grandes cantidades de datos, en general menos fiables, pero que proporcionan un cobertura espacial y temporal mucho mayor. La obtención de datos observados por la ciudadanía, directa o indirectamente, se denomina crowdsensing, y ya se está utilizando para observar variables de distinto tipo, como por ejemplo la calidad del aire o el ruido ambiental. Una parte importante de fuentes de datos de tipo crowdsensing son aquellas que involucran el análisis del contenido de redes sociales, dado el gran volumen de información que generan estas redes de forma continua.

Los datos generados por procesos de observación son complementados por resultados de modelos medioambientales. Estos modelos pueden tener la forma de modelos físicos a escala del entorno real, como puede ser un túnel de viento, o de modelos matemáticos que intenta aproximar el comportamiento real del sistema con conjuntos de expresiones matemáticas. Estos modelos matemáticos pueden tener naturaleza empírica, en donde el modelo se genera mediante un proceso de generalización de un conjunto de observaciones sobre las que se entrena el modelo. Los modelos matemáticos también pueden estar basados en la simulación matemática del sistema físico. Estos últimos son los generalmente utilizados para la predicción meteorológica, por ejemplo. Los modelos matemáticos se pueden utilizar para predecir condiciones futuras, pero también para estimar condiciones actuales o pasadas en localizaciones que no han podido ser observadas.

Los modelos conceptuales de datos utilizados tradicionalmente como base para la construcción de sistemas de almacenamiento y gestión se basan en la definición de entidades u objetos, que se agrupan en tipos o clases y que tienen propiedades sobre las que se registran datos. Siguiendo esta idea, el Open Geospatial Consortium (OGC) propone un modelo de referencia en el que las entidades geoespaciales (llamadas Features) pueden tener propiedades de tipos de dato convencionales (alfanuméricos) y propiedades de tipos de dato geométricos, que definen su localización y forma sobre el espacio geográfico. Sin embargo, dado que en algunos casos las propiedades convencionales de una Feature pueden variar a lo largo de su extensión espacial y/o temporal, el OGC define un tipo especial de Feature, llamada Coverage,

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

que permite representar la variación, continua o discreta, de una o varias propiedades de tipos convencionales a lo largo del espacio y/o el tiempo. El modelo de datos que subyace detrás del concepto de Coverage, es un modelo multidimensional, similar a los utilizados en el campo de los almacenes de datos. Las representaciones digitales del espacio utilizadas para codificar los valores de las propiedades geométricas de las Features se basan en el uso de vectores de coordenadas de los bordes de los objetos geométricos (puntos, líneas y polígonos). Es por eso por lo que llamaremos a estas Features representadas con geometrías vectoriales Vector Features. Por otro lado, las representaciones digitales de las Coverages se basan en particionamientos del espacio en celdas, para las que se define algún tipo de método de estimación de la variación de las propiedades a representar. La representación más comúnmente utilizada se basa en la división del espacio en un grid de celdas cuadradas, y en el uso de un valor constante de la propiedad en cada celda. Esta representación se conoce como representación raster y es por ello que llamaremos a estas estructuras de datos Raster Coverages.

El OGC define formatos y codificaciones estándar basadas en los modelos conceptuales mencionados arriba. En concreto, los formatos de almacenamiento de datos de Vector Features están basados en una simplificación del modelo propuesto por el OGC para la representación de geometrías vectoriales, llamado Simple Feature Access (SFA). Este modelo define geometrías simples (Point, Linestring y Polygons) y colecciones de geometrías simples de tipo heterogéneo (GeometryCollection) o de tipos homogéneos (MultiPoint, MultiLineString, MultiPolygon). Este modelo, que define también un conjunto amplio de funciones para la gestión de geometrías, es también la base para la implementación de extensiones espaciales de Sistemas Gestores de Bases de Datos (SGBDs). Por otro lado, una de las codificaciones más utilizadas para Raster Coverages es el formato GeoTIFF, que usa las cabeceras del formato de imagen TIFFF para representar los datos necesarios para georreferenciar la imagen (punto de la esquina inferior izquierda, resoluciones en ambos ejes, número de celdas en ambos ejes, etc.). En aplicaciones medioambientales, las Raster Coverages se suelen representar con formatos de almacenamiento de arrays geocientíficos. El ejemplo más conocido es el formato NetCDF. Existen también ejemplos de sistemas de gestión de datos de arrays o raster, sin embargo, su madurez e implantación no es para nada comparable al de las extensiones espaciales de los SGBDs.

Las infraestructuras de datos geoespaciales y medioambientales proporcionan sistemas de almacenamiento basados en los modelos y codificaciones descritas arriba e implementan servicios que permite el descubrimiento y acceso a los datos a través de la web. Si nos

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

centramos en el acceso a los datos, distintos estándares de servicio se encargan del acceso a los mismos dependiendo de si estos tienen la forma de colecciones de Vector Features o de Raster Coverages. En concreto, el servicio más utilizado para proporcionar acceso web a Vector Features es el Web Feature Service (SFS), definido por el OGC. Este es el servicio recomendado también por la directiva INSPIRE, que regular la creación de la Infraestructura de datos Espaciales de Europa. Este servicio, permite recuperar entidades geoespaciales disponibles en un almacén, soportando para su consulta la especificación de filtros, que incluyen filtros sobre propiedades convencionales, pero también filtros sobre propiedades geométricas que hacen uso de predicados topológicos. Así, por ejemplo, a través de un WFS podremos consultar los municipios almacenados que contienen a un determinado punto por el que pretendemos filtrar. Respecto al acceso a Raster Coverages, en el ámbito de los datos medioambientales lo más común es utilizar un servidor TDS (THREDDS Data Server), que permite el acceso y filtrado de datos representados en formatos de array científicos, como el NetCDF. Sin embargo, en el ámbito más general de los datos geoespaciales, el servicio más utilizado es el Web Coverage Service (WCS) definido por el OGC. Este es el servicio recomendado por INSPIRE. Este servicio permite recuperar los datos de una cobertura geoespacial, habilitando el filtrado por sus dimensiones geoespacial y temporal. Un servicio que va un poco más allá del WCS en el acceso a datos de coberturas es el WCPS (Web Coverage Processing Service), también definido por el OGC y que proporciona capacidades de procesamiento declarativo de los datos.

Por otro lado, las infraestructuras de datos abiertos de propósito general basan sus implementaciones en estándares de modelos, formatos y puntos de acceso definidos en el ámbito de la web semántica. Las representaciones de datos están basadas en el modelo de datos propuesto dentro del estándar RDF (Resource Description Framework) del consocio W3C. En RDF los datos se representan con tripletas (sujeto predicado objeto). El sujeto representa a un recurso (cosa, objeto, etc.) identificado universalmente mediante una IRI (Internationalized Resource Identifier). El objeto puede ser otro recurso o un literal (valor) de un tipo de dato (entero, real, string, temporal, etc.). El predicado, que etiqueta la naturaleza de la relación entre el sujeto y el objeto también se identifica con una IRI. Sobre este modelo básico, se pueden construir esquemas complejos que definen clases de objetos e instancias, y relaciones entre clases e instancias de distinta naturaleza, incluyendo jerarquías de herencia/especialización entre clases. Estos esquemas pueden definirse de forma simple usando el estándar RDF Schema o de forma más compleja con la creación de ontologías representadas en OWL (Web Ontology Language). El W3C define también ontologías de tipo general que puedan servir para la representación

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

uniforme de datos en ámbitos concretos, y que puedan facilitar la integración semántica de conjuntos de datos diferentes mediane el uso de dichos vocabularios comunes. El protocolo de acceso a los datos a través de la web y el lenguaje declarativo de consulta de datos RDF se definen en el estándar SPARQL del W3C. El W3C define también soluciones para mapear de forma directa o definida por el usuario el modelo de datos relacional a RDF, con el objetivo de facilitar la integración de fuentes de datos representadas con modelo relacional, que es el modelo más utilizado con diferencia en la práctica, dentro de la web de los datos enlazados.

La gran mayoría de los esfuerzos realizados para la integración de los datos espaciales y medioambientales dentro de las tecnologías de la web semántica han tenido que ver con la definición de ontologías de alto nivel. Así, dentro del estándar GeoSPARQL, el OGC define una ontología para la representación general de datos de Vector Features, siguiendo el modelo de geometrías simplificado SFA también definido por el OGC. No existen soluciones concretas para la representación de Raster Coverages en RDF, sin embargo, con ciertas limitaciones, el estándar RDF Cube, diseñado para datos multidimensionales, podría ser utilizado. La consulta de datos de Vector Features puede realizarse usando implementaciones existentes de GeoSPARQL con un rendimiento razonable, aunque todavía se necesiten esfuerzos para incorporar en los motores de consulta todos los avances en indexación, procesamiento de consultas y optimización desarrollados en el ámbito de las bases de datos espaciales. Sin embargo, las soluciones disponibles para la consulta de datos de Raster Coverages son todavía inmaduras y sufren de problemas de rendimiento y de facilidad de uso. En concreto, la principal característica de estas soluciones es que incorporan un tipo de datos específico para representar coberturas, que se usa para almacenar tiles (zonas) de las coberturas. Para gestionar estos tiles, el programador tiene que hacer uso de una lista larga de funciones del tipo de dato. Esta aproximación, que ya es proporcionada por algunos SGBDs espaciales, ha demostrado ser poco eficiente, comparada con sistemas y formatos específicos para la gestión de arrays.

Como resumen de esta exposición del estado del arte podemos decir que GeoSPARQL proporciona el estándar apropiado para la consulta de datos geoespaciales en la web de los datos enlazados, sin embargo, las implementaciones actuales del estándar no resuelven de forma eficaz y eficiente la consulta de datos de Raster Coverages y por lo tanto tampoco la consulta integrada de todos los tipos de datos geoespaciales y medioambientales disponibles en las fuentes de datos de observación y modelado del medioambiente.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Objetivo y Contribuciones de la Tesis

El principal objetivo de esta Tesis es el contribuir al desarrollo de tecnologías de la web semántica que permitan el acceso eficiente a repositorios heterogéneos de datos medioambientales a través de los estándares de la web de los datos enlazados. En concreto, en la Tesis se desarrollan algunos componentes específicos de un motor de consulta GeoSPARQL que mejora el rendimiento de las evaluaciones de las consultas cuando se accede a través de servicios web geoespaciales estandarizados a Vector Features y Raster Coverages. Las características principales de la solución propuesta pueden resumirse como sigue.

- El sistema permite la consulta de Vector Features accesibles a través de un servidor de datos geoespacial que implementa el estándar del OGC WFS. La optimización de consultas implementada por el motor de consultas es capaz de delegar el filtrado convencional y geospatial de los datos en el servicio WFS. El mapeado de las Vector Features a RDF se basa en la solución de mapeo directo del modelo relacoinal a RDF propuesta por el W3C.

- El sistema permita la consulta de Raster Coverages accesibles a través de un servidor geoespacial que implementa el estándar WCPS del OGC. La optimización de consultas consigue delegar el filtrado de las coberturas por sus dimensiones espacial y/o temporal en el servicio WCPS. El mapeado entre el modelo de Raster Coverages y el RDF se realiza utilizando una nueva propuesta de soluciones de mapeado definidas en la Tesis.

- El sistema implementa operaciones de Join y Agrupación que pueden usarse por el motor de consultas para generar planes de consulta eficientes que combinan Vector Features y Raster Coverages.

El sistema resultante, llamado GeoLD, ha sido implementado como una extensión del procesador SPARQL de Apache Jena. El servidor geoespacial utilizado para el desarrollo y la evaluación de la solución está basado en la combinación de Geoserver, para la implementación del servicio WFS, y Petascope para la implementación del servicio WCPS.

Las principales contribuciones de la Tesis, más allá del estado del arte de las tecnologías de la web semántica para datos geoespaciales y medioambientales, son las siguientes.

- Una aproximación para mapear Raster Coverages a RDF, llamada C2RML (Coverage to RDF Mapping Language), que está basada en el lenguaje R2RML (Relational to RDF

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Mapping Language) definido por el W3C. Al contrario de la aproximación seguida por las actuales extensiones SPARQL para datos raster, C2RML no necesita de la definición de un tipo de datos raster, ya que el RDF se genera al nivel de celda del propio raster.

- La definición de nuevos operadores SPARQL que permiten el acceso eficiente a Vector Features y Raster Coverages disponibles a través de servicios que implementan los estándres WFS y WCPS del OGC.

- La definición de una estrategia de optimización de consultas que identifica subárboles SPARQL máximos que puedan reemplazarse por instancias de los nuevos operadores definidos, que permiten el acceso eficiente a los datos. El optimizador debe de buscar la delegación de la mayor parte de procesamiento posible in los servicios web, siguiendo así la heurística de realizar cuanto más procesamiento mejor los más cerca de los datos como sea posible. Los resultados de las evaluaciones realizadas muestran que GeoLD mejora el rendimiento de las soluciones del estado del arte.

- La definición e implementación de nuevos operadores de Join y Agrupación que permiten la agregación eficiente de datos de Raster Coverages sobre las zonas definidas por Vector Features. En concreto, los planes de consulta que usan el nuevo operator de Join funcionan un orden de magnitud más rápido que los planes de consulta obtenidos con los operadores convencionales de Join de SPARQL. Más allá, el uso del nuevo operador de Agrupación consigue planes de consulta que funcionan entre uno y dos órdenes de magnitud más rápido que los planes generados con el nuevo operador de Join.

Publicaciones

Algunas de las contribuciones de la Tesis ya han sido publicadas en los siguientes artículos y congresos.

Revista internacional: Shahed Bassam Almobydeen, José R.R.Viqueira, Manuel Lama. GeoSPARQL query support for scientific raster array data. Computers & Geosciences 159:105023. Elsevier. 2022. doi: https://doi.org/10.1016/j.cageo.2021.105023.

Congreso internacional: Shahed Bassam Almobydeen, Khalid Alkharabsheh, José RR Viqueira, Manuel Lama Penín. Environmental Linked Data: Challenges. 1st International Con-

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ference in computer sciences and information systems researchers. January 12-13, 2016, Amman, Jordan.

Congreso nacional: Shahed Almobydeen, José R.R. Viqueira, Manuel Lama Penín. A Federated Approach for Array and Entity Environmental Linked Data. XXI Jornadas de Ingeniería del Software y Bases de Datos, V Congreso Español de Informática. 14-16 de Septiembre de 2016. Salamanca.

El siguiente artículo que ha publicado el autor de la Tesis durante el periodo de realización de la misma corresponde a trabajo fuera del ámbito de esta Tesis.

Revista internacional: Khalid Alkharabsheh, Shahed Almobydeen, Jose A Taboada, Yania Crespo. Influence of nominal project knowledge in the detection of design smells: An exploratory study with god class. International Journal of Advanced Studies in Computers, Science and Engineering 5(11):120-127. 2016.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Abstract

Environmental data is of great importance in many fields of science and engineering. Due to that, an important amount of resources is spent every year in the construction and maintenance of complex environmental observation and modeling infrastructures, which are now also complemented with citizen data provided through crowdsensing approaches. Based on the geospatial dimension of the data, and broadly speaking, two major types of environmental datasets may be identified: i) Vector Features corresponding to entities or objects that have properties of conventional data types and possibly properties of geometric data types, which are represented with vectors of spatial coordinates. ii) Raster Coverages that represent the variation in space and/or time of conventional properties.

Geospatial and environmental data infrastructures enable the storage of the above types of data, and also the discovery of the datasets and their access through standard web service interfaces. However, the use of those infrastructures is generally limited to experts with very specific skills. Contrary to the above, general purpose open data infrastructures are evolving towards the construction of a linked web of data, using technologies from the semantic web area. Those technologies include the RDF data model and the SPARQL query language.

Many application domains would benefit from the incorporation of geospatial and environmental data sources to the linked data web. In general, such geospatial enablement of the semantic web would allow ICT practitioners to bring environmental data sources closer to a wider variety of citizens problems. To achieve this however, underlying technologies have to be extended to support geospatial data. Some efforts have already been done, and thus, the GeoSPARQL standard supports already the querying of Vector Features with a reasonable performance. However, the querying of Raster Coverages and also the integrated querying of both types of data is not adequately supported by current semantic web query engines.

Based on the above, this Thesis proposes advances in various components of a new

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

GeoSPARQL query engine, called GeoLD, that enables the efficient integrated querying of heterogeneous environmental datasets, including Vector Features and Raster Coverages, which are available through standard web services of geospatial data infrastructures. The proposed solution enables the access to Raster Coverages by providing a new Coverage to RDF Mapping Language (C2RML), which enables the programmer to incorporate specific vocabularies during the definition of the mapping between the coverage data schema and RDF. New SPARQL operators and a new query optimization strategy provide the algorithms required to reach query response times orders of magnitude faster than those of state of the art technologies. Additionally, contrary to existing approaches, raster data querying is achieved without the need to use large lists of specific raster manipulation functions, which simplifies the programming task.

Future research lines are mainly related to the improvement of the data representation, query processing and optimization artifacts of the system, and also to the increasing of the Technology Readiness Level (TRL) of the implementation, to bring it closer to real technology transfer objectives.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Objectives and Contributions of the Thesis 4

1.3 Publications 6

1.4 Outline 7

2 Background and Related Work **9**

2.1 Geospatial and Environmental Data Management 9

2.2 Open Linked Data Management 26

2.3 Geospatial Linked Data Approaches 33

2.4 Concluding Remarks 36

3 Problem Description **39**

3.1 Query Example 39

3.2 Query Processing and Optimization 42

4 Architecture of a Federated Vector-Raster GeoSPARQL Query Engine **49**

4.1 Web Services for Geospatial Features and Coverages 49

4.2 Extension of the SPARQL Evaluation Engine 51

4.3 Extension of the SPARQL Optimizer 53

5 Mapping Geospatial Datasets To RDF **55**

5.1 Mapping Geospatial Vector Features to RDF 55

5.2 Mapping Geospatial Raster Coverages to RDF 60

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

6 Efficient Vector and Raster Geospatial Data Access from GeoSPARQL **65**
6.1 Geospatial Data Access Operators 66
6.2 Query Optimization 72
6.3 Performance Evaluation 81

7 Federated Aggregation of Geospatial Raster Coverages over Vector Features **89**
7.1 Problem Description and GeoLD Solution 89
7.2 Vector-Raster Indexed Spatial Join 92
7.3 Vector-Raster Spatial Group and Aggregation 94
7.4 Performance Evaluation 106

8 Conclusions and Further Work **113**
8.1 Conclusions 113
8.2 Further Work 114

Bibliography **117**

List of Figures **129**

List of Tables **131**

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The precise knowledge of our environment is both input and output of many research efforts in different disciplines of both science and engineering, whose results have a big impact in our life conditions. Some examples of important problems that require and/or produce such kind of knowledge are climate change research, environmental pollution monitoring and control, weather forecast, disease control, etc. Thus, public administrations and scientific organizations devote a lot of resources to construct and maintain very big infrastructures for environmental monitoring and modeling. Currently, the above conventional environmental data generation infrastructures are combined with very large amounts of less reliable data collected with mobile crowdsensing [43].

The above sources of environmental data have a highly heterogeneous nature. Furthermore, required data analytics need to combine them with other sources of geospatial data. If we focus on data modelling considerations, two major types of geospatial datasets may be identified: i) Collections of geospatial *Features* [65], each of them describing an entity or object with properties of conventional and geometric data types. Geometries are represented by vectors of geospatial coordinates [52], thus we will call them *Vector Features*. ii) Geospatial *Coverages* [84] that describe the variation of some conventional properties over space and/or time. The most commonly used discrete representation of geographic space for the modelling of *Coverages* is based on square grids (*Raster Coverages*).

Currently available data infrastructures in the geospatial [82, 40] and environmental [79, 80] domain provide services that enable the discovery and access of datasets of both of the two

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

above forms (*Vector Features* and *Raster Coverages*). The use of these infrastructures is currently mainly restricted to experts in the Geographic Information Systems (GIS) domain and to scientists of different disciplines. On the other hand, conventional open data infrastructures are evolving towards the construction of a web of linked data [51], whose best practices [18] consist of: i) the use of Universal Resource Identifiers (URIs) to identify resources, ii) the use of HTTP to look up those resources, iii) the use of the Resource Description Framework (RDF) [76] graph data model for the representation of data of the resources and iv) the use of links to interconnect resources. The SPARQL [47] protocol and query language is used to implement access points with declarative query support. Conventional data sources may be incorporated by the use of mappings from the relational model to RDF [7, 32].

To enable the access to the huge amount of geospatial and environmental data, which is being generated in a daily basis, through open linked data infrastructures, their base technologies must be extended to support effective and efficient geospatial data representation and querying. Many applications could benefit from such new geospatial and environmental linked data technologies. One example would be tourism, where existing linked data repositories could be combined with specific collections of geospatial *Vector Features* that include locations to visit, transportation facilities, accommodation infrastructure, leisure infrastructures and sightseeing. Continuing with this example, Figure 1.1 shows a mockup of a portal that aggregates data of municipalities of the Spanish region of Galicia. The portal would use a GeoSPARQL end point to obtain RDF that combines descriptions from DbPedia[1] and geospatial data obtained from a geospatial server that implements standard geospatial web service interfaces. The *Vector Features* accessible through the geospatial server include municipalities, roads, hotels and beaches. The *Raster Coverage* records the prediction of meteorological variables such as air temperature, with a temporal resolution of one hour and for the next week. To show the information of one specific municipality (the municipality of "Muros" in the example), the GeoSPARQL engine must retrieve from the meteorological coverage the raster cells (rectangles) that lie inside the geometry of the municipality, to compute the average, minimum and maximum daily air temperature for the next days.

The Open Geospatial Consortium (OGC)[2] has already proposed a geospatial extension of SPARQL, called GeoSPARQL [86]. Some efficient implementations of GeoSPARQL are already available [67]. GeoSPARQL enables the representation and declarative querying of

[1] https://www.dbpedia.org/
[2] https://www.ogc.org/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

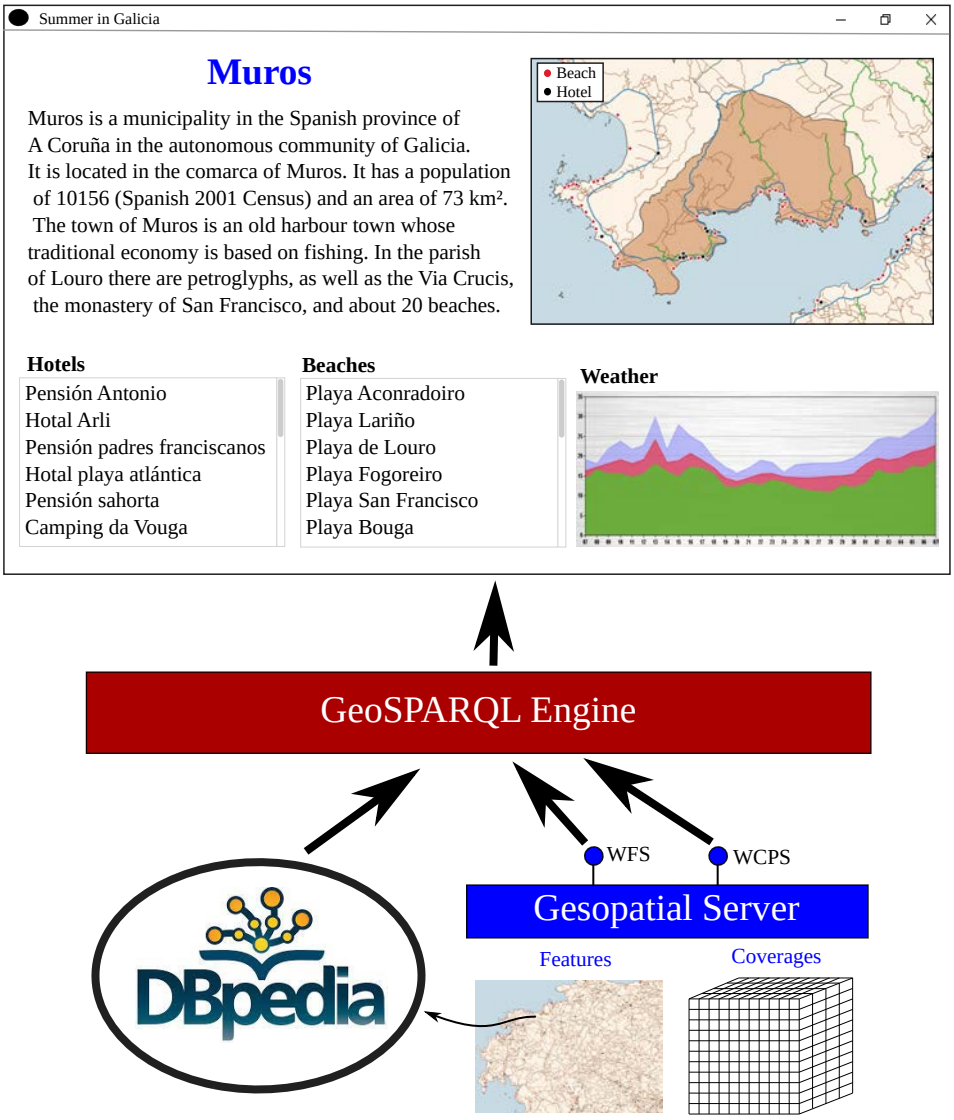


Figure 1.1: Illustration of an application example.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Vector Features with a reasonable performance, although some improvements are still required in existing implementations [85]. On the other hand, the storage and querying of *Raster Coverages* with semantic web technologies is not yet supported with a reasonable performance, to enable applications like the one illustrated above. In particular, the RDF Data Cube Vocabulary [31] may be used to represent multidimensional datasets, such as those managed by data warehouse technologies. However, the regular sampling nature that the geospatial and temporal dimensions of geosaptial coverages might have are not adequately supported by RDF Cube. On the other hand, a couple of SPARQL extensions with similar characteristics, called SciSPARQL [5, 6] and GeoSPARQL+ [55], have been implemented. SciSPARQL incorporates arrays as first class citizens in the SPARQL query language. GeoSPARQL+ follows a similar approach by defining a raster data type to be incorporated in GeoSPARQL. In both cases, the overall performance of the system is still not good. Besides, the user must deal with a large set of operations, on either arrays or rasters, to be able to perform effective queries. The maturity of these two implementations is still not at the level of that offered by other GeoSPARQL implementations [67].

Based on the above discussion, it is clear that much research work has still to be done to achieve efficient GeoSPARQL implementations, to bring this technology as a real solution for the development of real applications. This is particularly true when the system to be developed must perform analysis using data of *Raster Coverages*. It is finally noticed that most of the biggest datasets in the geospatial and environmental domain have the form of very large *Raster Coverages*, thus, most of the available data will not be incorporated in the linked data web until efficient support for *Raster Coverages* is not reached.

1.2 Objectives and Contributions of the Thesis

The main objective of the present Thesis is to contribute to the development of semantic web technologies that enable efficient access to heterogeneous environmental data repositories thought linked data standards. In particular, the Thesis develops some specific components of a GeoSPARQL query engine that improve the performance of the query evaluations when *Vector Features* and *Raster Coverages* are accessed through standard geospatial web services. The main characteristics of the proposed solution may be resumed as follows.

- The system enables the querying of *Vector Features* accessible through a geospatial server that implements the Web Feature Service (WFS) standard interface defined by

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

the OGC [100]. The query optimization implemented in the query engine is able to delegate the conventional and geosaptial filtering functionality on the WFS service. The mapping of *Vector Features* to RDF is based on the direct mapping from relational data to RDF [7].

- The system enables the querying of *Raster Coverages* accesible through a geospatial server that implements the Web Coverage Processing Service (WCPS) standard interface defined by the OGC [12, 13]. The query optimization implemented in the query engine is able to delegate the filtering of the input coverages by their dimensions on the WCPS service. The mapping of *Raster Coverages* to RDF is performed with a new mapping approach defined in the Thesis, which is based on the Relational To RDF Mapping Language (R2RML) [32] defined by the W3C.

- The system implements spatial join and group operations that may be used by the query engine to generate efficient query plans that combine *Vector Features* and *Raster Coverages*.

The resulting query engine, called GeoLD, has been implemented by extending the SPARQL processor of Apache Jena[3]. The geospatial server used for the development and evaluation of the system is based on the combination of Geoserver[4], for the WFS implementation, and Petascope [2], for the WCPS implementation.

The main contributions of the Thesis beyond the state of the art of geospatial and environmental semantic web technologies are the following.

- An approach to map *Raster Coverages* to RDF, called C2RML (Coverage to RDF Mapping Language), which is based on the R2RML [32] language of the W3C. Contrary to the approach followed by SciSPARQL [5, 6] and GeoSPARQL+ [55], C2RML does not require a raster data type, as RDF is generated at the level of raster cell.

- The definition of new SPARQL operators that enable the efficient access of *Vector Features* and *Raster Coverages* from standard WFS and WCPS web services.

- The definition of a query optimization strategy that identifies maximum SPARQL query subtrees that might be replaced by the new SPARQL operators, which enable the

[3] https://jena.apache.org/documentation/query/index.html
[4] https://geoserver.org/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

efficient access to the geospatial data sources. The optimization strategy must chase the delegation of as much functionality as possible to the underlying geospatial web services, following the heuristic of doing as much processing as possible as close to the data source as possible. The evaluation shows that GeoLD outperforms both GeoSPARQL implementations [67] and also specific array (SciSPARQL [5, 6]) and raster (GeoSPARQL+ [55]) extensions for almost all the types of queries over *Raster Coverage* datasets.

- The definition and implementation of new join and group operators that enable the efficient aggregation of *Raster Coverage* data over *Vector Feature* zones. In particular, the query plans that use the new join operator work an order of magnitude faster than equivalent plans using conventional SPARQL operators. Beyond that, the use of the new group operator enables achieving query plans between one and two orders of magnitude faster than equivalent ones using the new join operator.

1.3 Publications

Some of the contributions of this Thesis have already been published in the following papers.

International Journal: Shahed Bassam Almobydeen, José R.R.Viqueira, Manuel Lama. GeoSPARQL query support for scientific raster array data. *Computers & Geosciences* 159:105023. Elsevier. 2022. doi: `https://doi.org/10.1016/j.cageo.2021.105023`.

International Conference: Shahed Bassam Almobydeen, Khalid Alkharabsheh, José RR Viqueira, Manuel Lama Penín. Environmental Linked Data: Challenges. 1st International Conference in computer sciences and information systems researchers. January 12-13, 2016, Amman, Jordan.

National Conference: Shahed Almobydeen, José R.R. Viqueira, Manuel Lama Penín. A Federated Approach for Array and Entity Environmental Linked Data. XXI Jornadas de Ingeniería del Software y Bases de Datos, V Congreso Español de Informática. 14-16 de Septiembre de 2016. Salamanca.

The following paper corresponds to work out of the scope of the Thesis.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

International Journal: Khalid Alkharabsheh, Shahed Almobydeen, Jose A Taboada, Yania Crespo. Influence of nominal project knowledge in the detection of design smells: An exploratory study with god class. International Journal of Advanced Studies in Computers, Science and Engineering 5(11):120-127. 2016.

1.4 Outline

The remainder chapters of the Thesis are organized as follows.

Chapter 2 provides a detailed description of the state of the art in the topic of the Thesis. In particular, first, geospatial and environmental data management technologies are reviewed, including a description of main environmental data sources, a review of geospatial data modelling approaches and encodings and a description of main technologies involved in the construction of spatial and environmental data infrastructures. Next, the main techniques used in the scope of the management of open linked data are also described, including data representation and querying solutions. The chapter ends with a review of the most important works found in the literature related to the management of geospatial and environmental data with linked data technologies, including the definition of new ontologies and the definition of new solutions for data querying, both for *Vector Features* and *Raster Coverages*. The chapter concludes with some remarks that provide the motivation to start the work of the Thesis.

Chapter 3 provides a more detailed description of the problem to be solved in the Thesis. More precisely, it describes some input geosptial datasets and introduces a motivating query example. Based on that, it describes the main subproblems that have to be solved to enable the efficient evaluation of the query.

Chapter 4 defines the architecture of the GeoLD query engine. The main components include the geospatial data sources, the new operators that have to be included in the SPARQL evaluation engine and the new query optimization strategies that have to be included in the SPARQL optimizer.

Chapter 5 describes the solutions adopted for the mapping from geospatial data models to RDF. In particular, it describes how *Vector Features* are directly mapped to RDF with a straightforward adoption of a direct mapping approach. Regarding *Raster Coverages*,

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

it defines both a direct mapping and a mapping language that is used to customize the generation of RDF from *Raster Coverage* cells.

Chapter 6 describes the query processing and optimization solutions implemented in GeoLD to achieve the efficient query of both *Vector Features* and *Raster Coverages*, available through standard OGC web services. The chapter describes the new data access operators for both types of geospatial data. It describes also the algorithms used by the query optimizer to achieve efficient query plans. Finally, it shows the results achieved in terms of query performance.

Chapter 7 describes the new operations defined and implemented that enable the efficient aggregation of *Raster Coverage* cells over the zones defined by a *Vector Feature* dataset. The algorithms of the new join and group operators are described with some detail and the results achieved in terms of performance improvement are also discussed.

Chapter 8 concludes the Thesis with a discussion of the main advantages and disadvantages of the undertaken approach. A discussion on future research work lines is also included.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

BACKGROUND AND RELATED WORK

Background knowledge and approaches related to the one developed in the present Thesis are reviewed in the following sections. In particular, the state of the art on both geospatial and environmental data management and on open linked data management is first described. Next, different approaches in the scope of geospatial linked data management are presented. The chapter finishes with some concluding remarks.

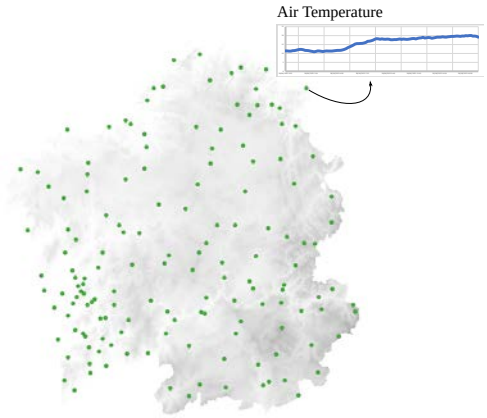
2.1 Geospatial and Environmental Data Management

The following subsections provide the background on geospatial and environmental data management that is required to understand the problems and solutions proposed in the Thesis. Such background includes an brief review of available environmental data sources, a description of data models and data representations used for geospatial and environmental data and metadata, and a description of the technologies and standards involved in the development of geosaptial and environmental data infrastructures.

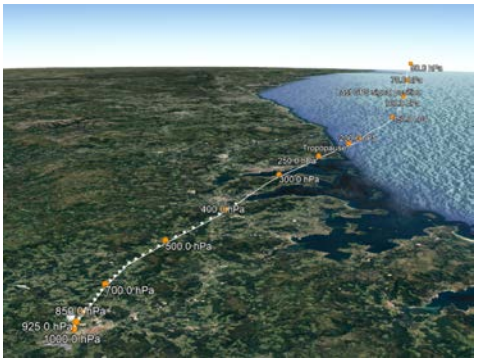
2.1.1 Environmental Data Sources

Geospatial data refers to data elements that reference in some manner a position or shape in geographic space, i.e., the space defined by the Earth surface. Many different application domains may benefit from the existence of geosaptial datasets. One such application domain of special relevance, due to the very large amount of geospatial data that it generates, is the observation and modelling of our environment. Environmental data sources may be

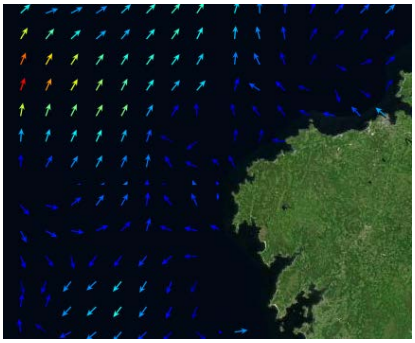
USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA



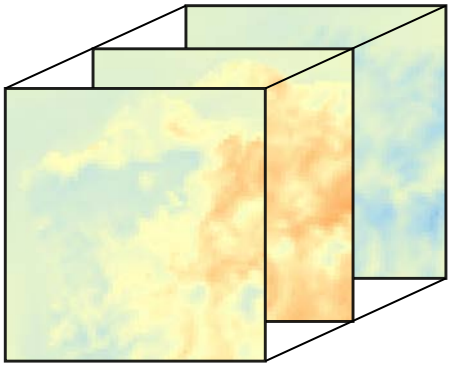
(a) Environmental Stations



(b) Radiosounding Observation



(c) HF Radar Observation



(d) Weather Research and Forecasting Model

Figure 2.1: Examples of Environmental Data Sources.

broadly classified into two main groups: i) those generated by environmental observation infrastructures and ii) those generated by environmental modeling infrastructures.

Many different types of observation processes are generating huge amounts of data of our environment in a daily basis. A classification of the data generation processes used in environmental applications is given by the OGC Sensor Model Language (SensorML) standard specification [19]. SensorML provides data modelling mechanisms to represent simple processes that directly generate data using some process method. Aggregate processes interconnect other simple and/or aggregate processes to generate more elaborated observations.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Specific classes are defined in SensorML to model observation processes that consist of simple physical devices (sensors) and also to model complex systems that combine physical components with non physical elements (software components for example).

Physical devices and systems may be classified according to where they are located and installed. If we consider the location of the device with respect to the location of the entity that it is observing, sensors may be classified as either in-situ or remote. The former generate observations of entities that are located in the immediate surroundings of the sensor location, whereas the latter observe remote entities, i.e., entities located at some distance of the sensor. On the other hand, if we consider the platform were the sensor is installed, then we may classify them as either static or mobile. An example of an observation system that is static and that measures in-situ is an environmental station. Figure 2.1(a) shows a network of meteorological stations managed by the Galician meteorological agency MeteoGalicia[1]. Each sensor installed in the station generates a time series of observations of one or various observed properties. The figure illustrates the values generated for the air temperature during some hours of one day. The sensors in the station may be installed at different heights, thus, the location of the observed entity might combine the 2D geospatial location of the station and the elevation above sea level of the sensor. Many different types of instruments that measure in-situ may be installed in platforms that move. On example in the scope of meteorological observation is shown in Figure 2.1(b), which depicts the 3D trajectory followed by a sounding balloon dropped by MeteoGalicia in the city of Santiago de Compostela. The instruments of the device measure air temperature, air pressure and relative humidity at different points along the trajectory of the balloon. The generated data has the form of a time series where both values and observation location evolve in time. Remote sensors (either static or mobile) are usually very complex and expensive infrastructures that generate huge amounts of data. One example of static remote sensing technology is the High Frequency Radar (HF Radar). Figure 2.1 shows an observation of sea surface current speed and direction, generated by the HF Radar infrastructure associated to the RAIA Observatory [2], at the north-west of the Iberian Peninsula. Mobile remote sensors are commonly installed on board of aircrafts and satellites.

Apart from the above automated observation systems, traditional environmental observation rely also in the execution of field campaigns were experts perform some manual sampling of the observed entities. In many cases, a specimen is extracted that might be analyzed in-situ

[1] https://www.meteogalicia.gal/
[2] http://marnaraia.org/en/home-english/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

or taken to be analyzed in some laboratory. The broad use of mobile devices like smartphones and tablets opened the opportunity to support those filed campaigns with mobile applications [95]. However, going one step beyond in the use of mobile devices is still possible. Thus, the people-centring sensing approach [44], involves non-expert citizens in the observation of our environment, either by using the sensors already available in mobile devices (microphone, accelerometer, etc.), or by asking the citizen to input data derived by her senses. The use of mobile crowdsensing [43] has already been exploited to measure environmental properties such as air quality [38, 37] and noise [75].

An important subset of mobile crowdsensing applications are those based on the analysis of contents of social media, due to the huge amount of potential data that may be extracted in real time from those platforms [36, 94, 92, 104, 10]. Main challenges to be faced with these new types of environmental observation mechanisms are related to the data heterogeneity, data quality and ethics. The lack of certainty about the future availability of the datasets is another issue that has to be taken into account during the design of their incorporation into existing environmental observation infrastructures.

The use of environmental models is a common practice in many application domains to obtain a better understanding of the environmental system dynamics and to perform predictions of interest. Models are classified in [102] as either physical or mathematical. The former are physical twins of the real systems at a feasible scale. Due to their generally very high cost, they are used only when mathematical models do not give good results. On the other hand, mathematical models use mathematical expressions to define relationships between the values of the properties being analyzed. Mathematical models may be empirical or based on the simulation of physical knowledge of the real systems. Examples of empirical models are those generated by statistical or machine learning techniques [9]. Examples of a mathematical physical-based models are those used in weather forecast. Part of the output of one such model, namely the Weather Research and Forecasting (WRF) model, is illustrated in Figure 2.1(d). In particular, the figure illustrates the evolution of the air temperature property at each location of Galicia and for each hour, generated by the WRF model implementation done by MeteoGalicia. Mathematical models may be used to estimate past, present and future conditions. More precisely, the objective of forecast models is to guess future system conditions. However, due to the lack of complete geospatial coverage of the environmental observation infrastructures at each time instant, it is also needed to use models that estimate the properties of interest during the past (reanalysis models) and present (nowcast models).

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

2.1.2 Geospatial and Environmental Data Models

The definition of geospatial data modeling frameworks was a topic of active research a couple of decades ago [48, 98, 93]. Broadly speaking, two main approaches for data modeling were proposed.

Entity-Based Model This model considers geographic space as populated by entities or object of different types. Those objects have conventional properties of interest that are represented with values of conventional data types, which include numeric types, character strings, Boolean and temporal types. Besides, the location or shape of the entities in geographic space is represented by properties of geometric types, which include points, lines, surfaces and combinations thereof.

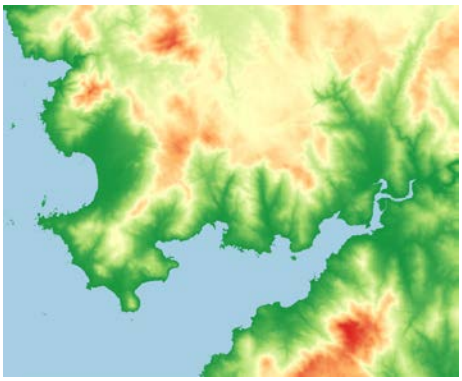
Field-Based Model This model considers the points of geographic space as one of the dimensions that may be used to define the domain of functions called fields. Geospatial fields map each point of geographic space to some value of a range set defined from a conventional data type. Spatio-temporal fields add a temporal dimension to the geospatial fields to map each combination of geographic point and time instant to a value of the range set. Other dimensions beyond space and time may also be consider to define the domain of fields. A typical example is the use of height above sea level as a separate dimension of the field domain.

Nowadays, the Open Geospatial Consortium (OGC) has proposed a reference model [65], which is also an ISO standard (ISO 19101), that enables the conceptual representation of geospatial entities, their types and their properties and relationships. A geospatial entity is called a *Feature* in this model and its type is called a *Feature Type*. Figure 2.2(a) shows the geometric representation of *Features* of three different *Feature types*. Purple circles are used to represent the point geometries of *Features* of type "Beach". Red triangles represent the point geometries of "Hotels". Blue, green and brown lines are used to represent the trajectory of "Roads" of various ranks. Finally, the black lines represent the border of the polygon geometries of "Municipalities".

A special type of *Feature*, called *Coverage* [84], is used in the data modelling framework of OGC to represent entities whose properties vary over their geospatial and/or temporal extension. More precisely, a *Coverage* is defined as a collection of mappings (called bands or properties) from a temporal, geospatial or spatio-temporal domain to range sets of conventional

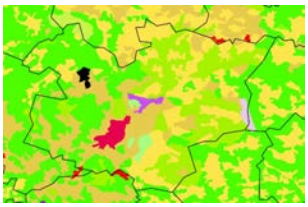
USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

(a) Geosptial Features

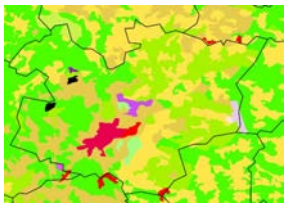


(b) Continuous Geospatial Coverage

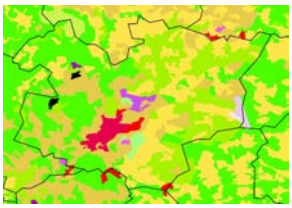
[1990, 2000)



[2000, 2006)



[2006, 2012)



(c) Discrete Spatio-temporal Coverage

Figure 2.2: Examples of Features and Coverages.

types. *Coverages* provide therefore the field-based view of geographic space described above. Two main types of geospatial coverages are considered in [84], depending on the way properties vary over geographic space. In a *Continuous Coverage* the properties vary smoothly over space. Five different types of such *Coverages* are defined by the OGC specification, depending the discrete approximation used to store them in digital form.

- Thiessen Polygon Coverages.
- Triangulated Irregular Network (TIN)
- Square Grid Coverages.
- Hexagonal Grid Coverages
- Curve Segment Coverages (the geospatial domain is a line).

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

The *Square Grid Coverages* is the type of *Coverage* most commonly used in most of the application domains. These types of coverages are also known as *Raster Coverages*. In these *Coverages* the geospatial domain is partitioned into a regular grid of square cells, called raster cells or pixels. The width and height of each square cell is called the resolution of the *Coverage*. Figure 2.2(b) shows a *Raster Coverage* with a single band (property) that records the elevation above sea level.

The geospatial domain of a *Discrete Coverage* consists on a set of zones where properties have constant values. Changes in the property values therefore occur only in the borders of the domain zones. Figure 2.2(c) shows the graphical representation of a spatio-temporal *Discrete Coverage*, whose zones record land use. Three versions of the *Coverage* are shown, corresponding to three periods of time. Red zones are urban areas and purple zones are industrial areas.

After the above description of the two main data modeling approaches used for geospatial data, it has to be pointed out that the same geospatial phenomena might be modeled either as a *Feature Type* or as a *Coverage*. To illustrate this, consider for example the *Discrete Spatio-temporal Coverage* of land use depicted in Figure 2.2(c). The same dataset could be perfectly modeled as a *Feature Type* with a conventional property that records the land use and a geometric property of type polygon that records the shape of each zone. Moreover, if we consider the municipalities represented in Figure 2.2(a), depending on the properties that might be of interest, either a *Feature Type* or a *Coverage* would fit better. Thus, if we are interested in the total population of the municipality at a specific moment, then the municipality should be represented as a Feature. However, if we are interested in the geospatial variation of the population density inside the municipality, them a *Coverage* must be used instead.

The representations defined by the OGC model (also ISO 19107) [52] for the geospatial properties of the *Features* include both geometric and topological objects. Primitive geometric objects include points, curves, surfaces and solids, and their representations are based on vectors of coordinates. Such representations are called *Vector Representations* and thus, in the following, we will call *Vector Features Types* the types whose geometric properties are represented with vector representations. In addition to the primitive geometries, the model supports also *Complex* and *Aggregate* geometries. Both of them are composed of primitive geometries. The main difference is that the former represent always a connected geometry isomorphic to a primitive geometry, i.e., it might also be represented with a primitive geometry, whereas the latter might be disconnected. In addition to the above structures, the

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

model provides also a set of methods that support geometric processing, including constructor methods to create geometries from coordinates, methods to extract properties from geometries such as distance, area or length, methods to compute new geometries from existing ones such as intersection, union, buffer, etc., methods to encode geometries in text (Well Knonw Text - WKT) or binary representations (Well Known Binary - WKB) and methods to evaluate topological relationships like intersects, contains, etc., which are based on a well known formalism [25].

Topological objects represent nodes, arcs, faces or solids of a geospatial network. Theses representations are good to model and store data of networks such as transportation, electricity, water, gas, etc. Besides, they are also effective to represent the zones of *Discrete Coverages* using faces. In these geospatial representations, the above topological primitives are combined with geometric primitives. Thus, for example, topological Nodes include geometric points to model their location in geographic space. Similarly, topological arcs, that link two nodes, include geometric lines to model their geospatial trajectory.

The above data models are very important to provide a standard data storage architecture that support interoperable tools. They restrict however to data modelling and they do not include standard sets of metadata. Such metadata standards have also been proposed by standardization organizations like OGC and ISO, and they include dataset-level metadata and also record level metadata, which provide semantics of specific application domains.

The ISO19115 standard [59], adopted by both OGC and ISO, defines metadata elements at the level of dataset. The elements include both mandatory and optional elements that enable the specification of a minimum set of metadata useful for most applications, but also an extensibility mechanism that might be used to include other elements of interest in specific applications. The metadata elements of ISO19115 include the following.

- Identification of the dataset. They include identifier, title, abstract, date, organization, responsible person, geographic and temporal extents, languages used, information about scale and resolution, URL for data access, keywords, access restrictions, etc.
- Spatial and temporal extension of the dataset.
- Data quality and provenance (lineage) metadata for the whole dataset.
- Other spatial and temporal characteristics of the dataset.
- Descriptions of the contents.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- Other facets that include visualization, distribution, etc.

More specific models for data and metadata have been proposed in the environmental data management domain. As an example, the Observations Data Model (ODM) [57] has been defined in the scope of hydrological information systems to model hydrological observations. Those observations are accompanied by metadata of the sites where they were obtained, metadata of the methods used to generated them and the variables being measured. The VODM data model [77] is an extension of ODM with data streams and dataset level metadata. Another extension of ODM is proposed in [56] to work with discrete Earth observations. A recent model in the scope of water data management is proposed in [1]. The OGC adopts also a model for sensor observation data called Observations and Measurements (O&M) [30], which incorporates most of the concepts defined in the above models. This model is also adopted by ISO as ISO19156. The key entity of the O&M data model is the *Observation*. Each observation is generated by a *Process*, and it yields a data value for a specific *Observed Property* (or set of properties) of an observed feature called *Feature of Interest* (FOI). The model defines a broad variety of different types of observations depending on their result values, including simple measurements (real values with a unit of measure) and complex records and coverages (resulting generally from remote observations). Besides, the model incorporates entities that enable the modelling of the sampling strategies used to observe FOIs with wide geospatial extension. Those sampling strategies include the use of sampling points such as environmental stations, the use of moving artefacts like those used in radiosounding, the use of remote sensing devices like radars and the capture of specimens to be analyzed in the lab.

2.1.3 Geospatial and Environmental Data Encoding

Based on the general geospatial and environmental data modeling frameworks and solutions described in the previous subsection, more specific and simplified profiles have also been defined that enable the specification of encodings that are used in current system implementations. Some of the more important encodings are described below.

A simplified profile of the Feature Geometry OGC specification [52] is defined in the OGC Simple Feature Access (SFA) specification citeogcsfa. The main classes defined by this standard are depicted in Figure 2.3. Abstract class *Geometry* is a general artifact that enables the representation of any vector geometry. Primitive vector representations for points,

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

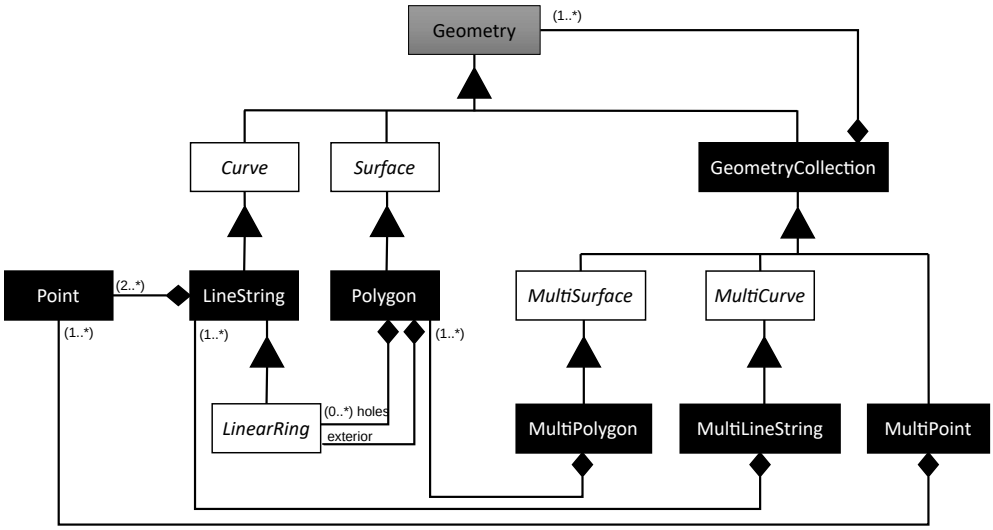


Figure 2.3: Classes of geometric objects defined in the Simple Feature Access (SFA) OGC Standard.

lines and polygons are provided by classes *Point*, *LineString* and *Polygon*, respectively. A *Point* is represented with coordinates of the underlying geographic space. A *LineString* is represented with a sequence of points, that define its trajectory. A *Polygon* is represented by an exterior *LinearRing*, whose start and end points are identical, and a possibly empty set of holes, each of them represented also by a *LinearRing*. Heterogeneous collections of geometries are represented with elements of class *GeometryCollection*. On the other hand, homogeneous collections of *Points*, *LineStrings* and *Polygons* are represented respectively by the subclasses *MultiPoint*, *MultiLineString* and *MultiPolygon*.

Various standard encodings are currently available for vector geometries. The Geography Markup Language (GML) [87] is the most complete one, as it enables the encoding of all the broad variety of geometric primitives, complex and aggregates defined by the OGC Feature Geometry Standard [52]. Despite of not being as complete as GML, the encodings derived or based on the SFA standard are more widely used, because they cover the needs of most applications and they are much simpler. Many tools, that range from geospatial libraries such as the Java Topology Suite (JTS) to geosaptial extensions of well known DBMSs, like PostGIS [83], support the classes defined by the SFA. The Well-Known Text (WKT) and Well-Known

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Binary (WKB) encodings are part of the SFA standard, thus they are supported by all these tools. Some examples of geometries encoded as WKT are given below.

Point A *Point* with coordinates (x, y) is encoded with the character string "POINT (x y)".

LineString A *LineString* defined by a sequence of points with coordinates $(x1, y1)$, $(x2, y2)$ ... (xn, yn) is encoded with the character string "LINESTRING (x1 y1, x2 y2, ..., xn yn)".

Polygon If "(L)" denote the encoding of the coordinates $(x1, y1)$, $(x2, y2)$ ... (xn, yn), $(x1, y1)$ of a *LinearRing* L, then a *Polygon* with exterior E and holes $H1$, $H2$, ..., Hn is represented with the character string "POLYGON((e), (H1), (H2), ..., (Hn))".

Another broadly used geospatial encoding based on the OGC SFA specification is the GeoJSON encoding [24].

Regarding geospatial coverages, a simplification of the OGC abstract specification [84] is defined in the Coverage Implementation Schema [16]. The encodings proposed for geospatial coverages in this simplified standard include the GeoTIFF and NetCDF formats. The GeoTIFF format [34] is by far the most widely used file format for coverages in Geographic Information Systems (GIS). The format makes use of the tags provided by the TIFF image file format to represent the information required to georreference the coverage, obtaining a compact format to store and transfer *Square Grid Coverages*. If we move more towards the scope of environmental application, the most common format to store and transfer geospatial coverages is the NetCDF format [35]. NetCDF provides a general format to encode multidimensional scientific arrays. The format includes *Dimensions*, *Variables* and *Attributes* and allows six primitive data types for the stored primitive values. Each *Variable* is an n-dimensional array of primitive values of some data type. The *Dimensions*, that may be shared by various variables, are defined by a name and a length. Special types of 1-d variables, called coordinate variables, are used to record the values of the dimensions. The metadata recorded in the file at the level of dataset, but also at the level of *Variable*, are recorded in *Attributes*.

As it was stated above, the adoption of the SFA standard by current DBMSs is wide, and thus many of them include geospatial extensions that enable the management of *Vector Features*. An example, which is open source and widely used in practice, is the PostGIS extension [83] of the PostgreSQL DBMS. The support offered by current DBMSs for *Raster Coverages* is not as mature as that offered for *Vector Features*, due to the intrinsic entity-based nature of the latter and to the array-based nature of the former. As a consequence, the

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

raster support in current DBMSs is still not as efficient as it should be, and therefore, *Raster Coverages* are usually stored in files and managed by specialized software frameworks. Some efforts were already undertaken to incorporate arrays as first class citizens in data management technologies. Examples of those efforts are the SciDB array engine [21] and the Rasdaman raster data management system [11, 15]. Both systems provide internal data structures and access methods optimized to store and access array data, and both of them provide a declarative language for array data querying.

Data encodings are also defined for metadata standards. Thus, for example, XML encodings have been proposed for both the ISO19115 dataset level metadata standard [61] and the O&M sensor observation model [29].

2.1.4 Spatial and Environmental Data Infrastructures

Spatial Data Infrastructures (SDI) provide technologies, policies and industrial arrangements devised to enable the access to geospatial datasets and related services [82, 40]. Main facilities provided by SDIs include geospatial data discovery, visualization and access. However, other services may be incorporated, for example those related to geoprocessing and geospatial analytics. The construction of the European Spatial Data Infrastructure is regulated by the INSPIRE directive[3]. The European infrastructure is based on the SDI implemented by the member states of the European Union, and recursively by their national, regional and local administrations.

The geospatial data modelling and encoding technologies described in the previous subsections provide the tools needed to achieve the efficient storage of both geospatial *Vector Features* and geospatial *Raster Coverages*. Research efforts undertaken in the area of spatial databases gave rise also to the emergence of efficient spatial data access methods and spatial query processing techniques [93]. Based on those efforts, spatial extensions of relational and object-relational DBMSs, based on the OGC SFA specification [53] and on the ISO SQL Multimedia standard [60], provide nowadays a mature and reliable technological infrastructure on top of which effective and efficient geospatial applications may be constructed with lower effort and investment. However, despite of the fact that some DBMS geospatial extensions include now raster data management capabilities (this is the case for example of the PostGIS extension [83] of PostgreSQL), the clear impedance between the relational structures

[3] https://inspire.ec.europa.eu/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

supported by DBMSs and the array nature of raster data causes problems in the use of those extensions. More precisely, two main problems arise.

Low Efficiency The use of relational structures to store and query raster data, which has a clear array nature, leads to solutions of lower performance, compared to that of specialized array-based data management systems. This is mainly due to the fact that relational structures, based on set theory, do not assume any ordering, whereas array structures are intrinsically ordered and designed to be accessed through array indices.

Complex Query Interfaces To alleviate the above efficiency problem, the raster data types supported by spatial DBMSs are not used to model complete coverage, but on the other hand, they are used to represent coverage tiles. The idea of a user managing coverage tiles is similar to the idea of a user managing disk blocks, which is clearly an issue of the physical level, that the user should not deal with. Thus, to achieve raster data management, the user must combine the SQL language with a very large set of raster operators and functions, which is not easy to remember.

Despite of some recent advances towards the integrated management of vector features and raster coverages [97], which still did not reach reliable implementations, current geospatial systems manage vector features and raster coverages with different technologies. In particular, specific vector file formats and geospatial DBMSs are used to manage vector features whereas raster file formats (like GeoTIFF and NetCDF) are usually in the basis of raster subsystems. It is expected that the use of array-based data management systems like SciDB [21] and Rasdaman [11, 15] will be replacing the direct use of raster files, however, as of today the use of these technologies is still not widespread.

As is was already stated above, the main services provided by a SDI enable data discovering, visualization and access of geospatial datasets, as it is shown in the service architecture depicted in Figure 2.4. These services are implemented on top of the geospatial data storage and management technologies described above. Catalog services are used by applications to discover both datasets and services based on their metadata. Such metadata is extracted from the data access infrastructure through harvesting transactions. Once a specific dataset or service is identified as relevant by the application, then data visualization and/or data access services may be used, depending on the type geospatial processing and analysis supported by the application.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

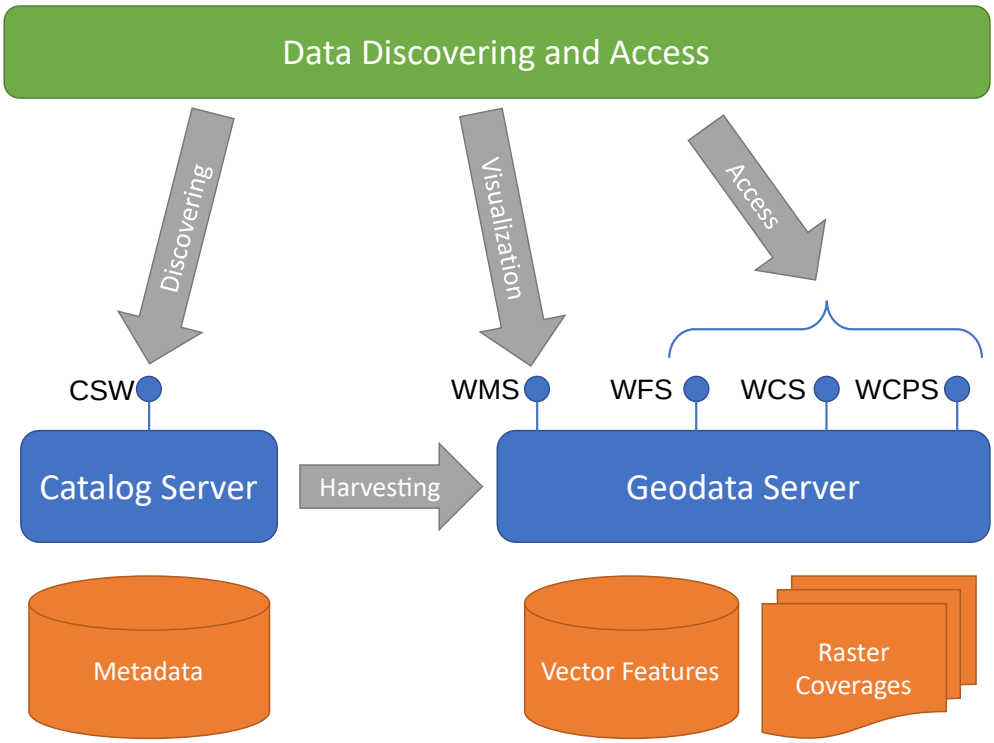


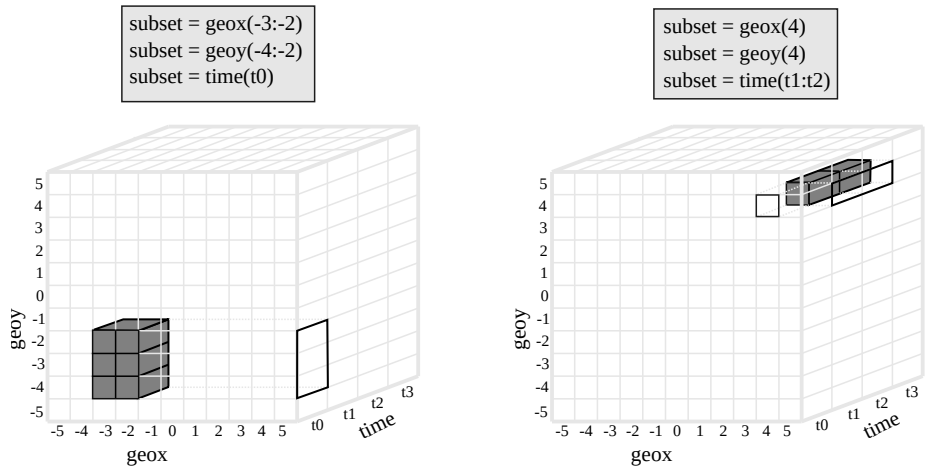
Figure 2.4: Architecture of main services provided by Spatial Data Infrastructures (SDI).

The OGC has defined standard interfaces for the main services that a SDI should provide. The standard OGC interface for a catalog service is the Catalog Service for the Web (CSW) [81]. The interface includes, amongst other, operations to describe the schema of the metadata records and operations to query and update the catalog. There is not any predefined schema for the metadata as part of the standard and thus schemes like the Dublin Core [4] or the ISO19115 [59] may be adopted.

For the visualization of geospatial data in maps, the OGC defines the well-known Web Map Service (WMS) interface [33]. This interface enables the users to generate maps using map layers constructed combining geospatial datasets with visualization styles. The result maps are obtained in well-known image file formats, including vector graphics formats such

[4] https://www.dublincore.org/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



(a) Trim on Space and Slice on Time

(b) Slice on Space and Trim on Time

Figure 2.5: Illustration of slicing and trimming operations on a *Raster Coverage*.

as svg and photographic image file formats like png and jpeg.

Web service interfaces are also provided to query *Vector Feature Types* and *Raster Coverages*. Thus, the Web Feature Service (WFS) interface [100] enables the querying of the *Vector Features* of the *Feature Types* available in a geospatial data store. The query language supported by the latest version of the WFS interface is quite complex. It references in general a collection of *Feature Types*, and it may include the following query elements.

- The specification of the set of result properties to be retrieved
- The specification of a query filter, expressed in the OGC Filter Encoding specification [101]. The functionality supports filtering both on conventional and geospatial properties, using topological predicates [25].
- The definition of the ordering of the results.

The implementation of the capabilities that enable the specification and evaluation of joins between various *Feature Types* is not mandatory in the standard. Thus, the characteristics of the WFS exploited by the developments of the present Thesis restrict to the filtering of *Vector Features* of a single *Feature Type*.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

The querying of *Raster Coverages* is supported by the Web Coverage Service (WCS) [14] specification. The query capabilities of this service restrict basically to slicing and trimming operations on the dimensions of the domain of the stored coverages. A slicing operation on one of the coverage dimensions filters the coverage to yield the cells that have a specific values in such dimension. A trimming operation behaves similar, but it uses a range of coordinates instead of a single coordinate to filter the coverage. Figure 2.5(a) illustrates a WCS request that performs a trimming operation on spatial coordinates and a slicing operation on the temporal coordinate of an input spatio-temporal coverage. The result coverage restricts to cells with $geox$ coordinates between -3 and -2, with $geoy$ coordinates between -4 and -2 and to time coordinate $t0$. A second WCS example is illustrated in Figure 2.5(b), now performing slicing operations on both spatial coordinates and a trimming operation on the temporal coordinate.

The OGC defines another standard web service interface that extends the functionality of WCS to achieve some kind of declarative processing of *Raster Coverages*. The service is called Web Coverage Processing Service (WCPS) [12, 13] and its main operation evaluates expressions with the following structure.

```
FOR c1 IN (CoverageName11, CoverageName12, ...)
    c2 IN (CoverageName21, CoverageName22, ...)
    ...
LET v1 := coverageExpression1,
    v2 := coverageExpression2,
    ...
WHERE booleanExpression
RETURN coverageProcessingExpression
```

Each $CoverageName$ references a coverage available in the service. Each variable ci iterates through the set of referenced coverages. The FOR clause performs a nested loop with all the iterator variables ci. For each combination of coverages in each iteration, clause LET defines new coverage obtained by a coverage expression. The RETURN clause is evaluated only if the WHERE clause evaluates to true. Coverage expressions and coverage processing expressions include slicing and trimming operations, scalar operations evaluated at each coverage cell and also aggregate functions. As an example, the following expression obtains the average of the cell values for band "band1" inside the cells defined by the trimming and slicing operations illustrated in Figure 2.5(a). The result is obtained as a GeoTIFF file.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

```
FOR c1 IN (coverage1)
RETURN encode (avg(c1.band1[geox(-3:-2), geoy(-4:-2), time(t0)]),
               "image/tiff")
```

Data infrastructures have also been constructed in the environmental observation and modelling domain. Thus, the Common Infrastructure of GEOSS (Global Earth Observation System of Systems) [79, 80] includes data discovering, data search and data access services over Earth observation data sources. The main types of services subject to be integrated in the GEOSS Component and Service Registry, according to [70], are data access services, catalog services, portrayal and display services and data transformation services. To achieve the integration of various data sources, the Discovery and Access Broker (DAB) framework of the above Common Infrastructure enables the brokering of data sources incorporating some kind of semantic interoperability.

More specific environmental data access infrastructures have been implemented in the scope of hydrologic systems [58, 77]. Their data storage layers are based on observation data models [57, 77] designed with in-situ observation processes in mind. On the other hand, data access infrastructures in the meteorological and oceanographic domain are usually based on the THREDDS Data Server (TDS)[5] implemented by UNIDATA. This server provides a catalog and also data access interfaces designed for geoscientific array data files that implement the Common Data Model (CDM). Currently the most common file format is the NetCDF file [35]. Data access services include OGC WMS and WCS interfaces. A specific interface that enables subsetting over the dimensions of NetCDF files, called NetCDF Subset Service (NCSS), is also provided by TDS.

The web service interface defined by the OGC to access sensor observation data, following the O&M data model, is the Sensor Observation Service (SOS) [22]. This interface enables the filtering of the stored observations by metadata (process, Feature of Interest, etc.) and also by spatial and temporal restrictions. As it was already shown, the O&M data model supports a wide variety of observation types, however, current implementations of this service restrict generally to in-situ observations recorded in a specific schema of a relational DBMS.

[5] https://www.unidata.ucar.edu/software/tds/current/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

2.2 Open Linked Data Management

The following subsections provide a brief description of the main background technologies defined by the World Wide Web Consortium (W3C) to support semantic data management. In particular, the provided overview includes data models and encodings, some important general ontologies, the main query language and solutions to generate semantic data from relational data.

2.2.1 Data Representation in the Semantic Web

Data in the semantic web is modeled using the Resource Description Framework (RDF) [76]. Broadly speaking, in RDF, things are identified with IRIs (Internationalized Resource Identifiers), and they are described by specifying the values of their properties. An RDF dataset is modeled with a graph, where nodes are either things or literals and arcs are properties also identified by IRIs. More formally, an RDF dataset is represented with a set of triples of the form

$$(subject \quad predicate \quad object)$$

where $subject$ and $predicate$ are identified with IRIs and the $object$ is either an IRI or a literal. Each such triple represents an arc of the RDF graph from the $subject$ to the $object$, tagged with the $predicate$. An example of an RDF graph is shown in Figure 2.6. The example represents an employee, identified by the IRI

$$<http://citius.usc.es/example1/employee/e321>$$

with four properties. The name of the employee is a property with IRI

$$<http://citius.usc.es/example1/hasName>$$

and value "Teresa", of data type $xsd{:}string$. The graph represents also the salary and area of the employee. The employee works in the project identified by IRI

$$<http://citius.usc.es/example1/project/p287>.$$

The fact that employee $e231$ works in project $p287$ is represented in RDF with the triple

```
<http://citius.usc.es/example1/employee/e231>
<http://citius.usc.es/example1/worksIn>
<http://citius.usc.es/example1/project/p287>.
```

Complex properties of things are represented using blank nodes. Blank nodes have an internal local identifier (they are not identified with an IRI). Thus, in the example, the funding of

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

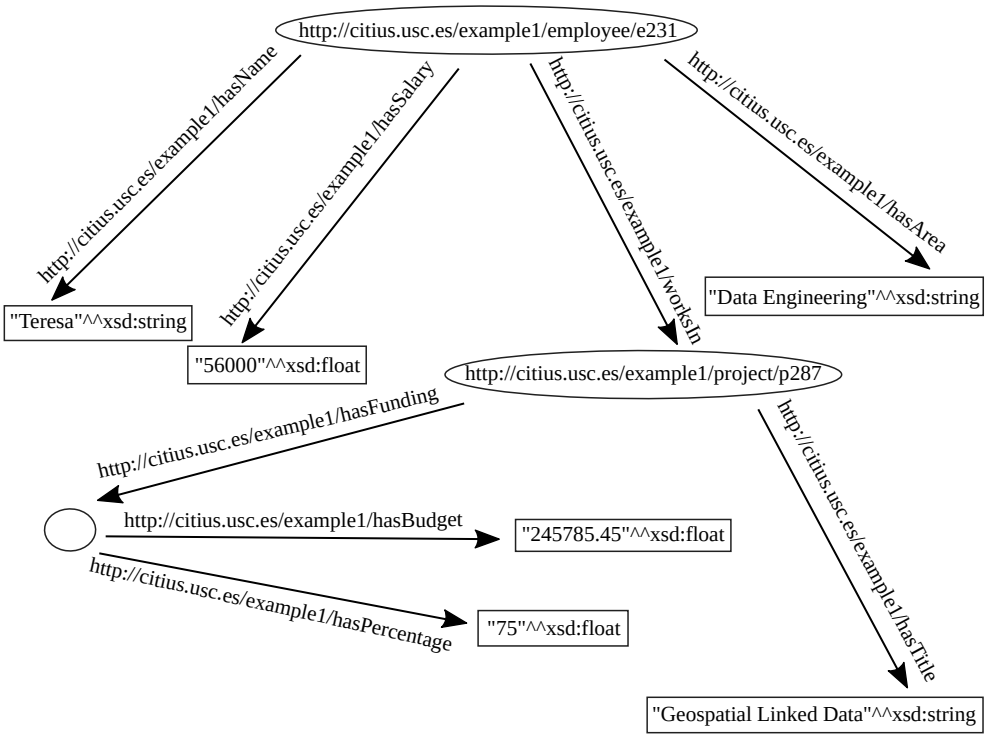


Figure 2.6: Example of RDF Graph.

project $p287$ is a complex property whose value is represented with a blank node. The *budget* and the *percentage* components of the funding are represented with respective arcs coming out of the blank node.

Various encodings have been proposed by the W3C to represent RDF graphs, including an XML syntax for RDF called RDF/XML [42] and a Terse RDF Triple Language, called Turtle [88]. Turtle enables the compact representation of RDF graphs in text format. The RDF triples may be directly written, but they may be compacted in various ways. For example, if various triples have a common subject, which is usually the case, such subject may be specified only once, to enable the representation of all the properties of a thing in a more compact manner. Similarly, if both subject and predicate are common to various triples, then they may also be represented only once. Solutions to reuse text during the specification of IRIs are also provided. The Turtle syntax is compatible with the triple pattern syntax of the SPARQL query

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

language. As an example of the use of this encoding, the triples of of RDF graph shown in Figure 2.6 are represented in Turtle as follows.

```
@base <http://citius.usc.es/example1> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<employee/e231> <hasName> "Teresa"^^xsd:string;
                <hasSalary> 56000;
                <hasArea> "Data Engineering"^^xsd:string;
                <worksIn> <project/p287> .
<project/287> <hasFunding> _:f1;
              <hasTitle> "Geospatial Linked Data"^^xsd:string .
_:f1 <hasBudget> 245785.45;
     <hasPercentage> 75 .
```

It is noticed how element @*base* is used to specify the base IRI, i.e., the IRI that is used as prefix by default for any universal identifier in the document. Similarly, the @*prefix* element is used to define a short alias for an IRI that is reused several times in the document. It may also be observed that the subjects corresponding to the two things represented in the document (an employee and a project) are specified only once now. Finally, the prefix _: is used to declare that the identifier is a local one, corresponding to a blank node, and therefore the base IRI should not be used in this case.

In order to describe data models of specific applications or application domains, the W3C defines the RDF Schema data modelling vocabulary [20]. RDF resources may be organized in classes, whose members are called instances. Both classes and instances are treated as RDF resources. The property *rdf*:*type* is used to declare that a given instance is member of a specific class. All the classes are instances of a general class called $rdfs:Class$. Property *rdfs*:*subClassOf* is used to define inheritance relationships between classes. All the RDF resources are instances of class *rdfs*:*Resource*. All the literals of RDF are instances of class *rdfs*:*Literal*. The RDF properties are instances of class *rdf*:*Property*. Subproperties of a given property may be defined using property $rdfs:subPropertyOf$. Property $rdfs:range$ is used to specify that the range of a given property consists of instances of one or various classes. Similarly, $rdfs:domain$ is used to define the domain of a given property. Human readable versions of a resource may be specified by using the property $rdfs:label$.

One step forward in the definition of schemas for RDF is brought by the Web Ontology Language (OWL) [54], defined also by the W3C. Ontologies provide classes, properties, individuals and data values, all of them represented with standards of the semantic web.

2.2.2 Top Level Ontologies

Top level ontologies provide generic vocabularies that may be reused in many application domains. Reusing vocabularies between applications has a clear and positive impact on their semantic interoperability, simplifying data integration tasks. Some important top level ontologies that should be taken into account when developing data infrastructures are briefly described below.

Data Catalog Vocabulary (DCAT)

The Data Catalog Vocabulary (DCAT) [3] is an W3C recommendation for the representation of metadata of datasets in the scope of open data. The concepts defined by DCAT are based on a profile of the Dublin Core metadata. Using this vocabulary, data publishers increase data discoverability and ease the integrated query of multiple catalogs by users. This standard is used by most of the available general purpose open data portal implementations.

PROV Ontology (PROV-O)

The PROV-O Ontology (PROV-O) [69], defines the concepts needed to represent generic provenance metadata. Provenance metadata refers to the data about organizations, entities, activities or people involved in the generation of a dataset. These metadata have a very important role to determine the quality, reliability and trustworthiness of the dataset. Provenance metadata may be specified in PROV-O with three different levels of detail. At the highest level, three main concepts are defined, $prov:Entity$, $prov:Activity$ and $prov:Agent$. A $prov:Activity$ acts upon or with an $prov:Entity$, to consume, process, transform, modify, relocate, use or generate it. Elements of class $prov:Agent$ have some responsibility related to other elements, that might be activities, entities or other agents. The other levels of detail provide more specialized means to describe provenance information, by specializing the above classes and adding new ones.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Data Quality Vocabulary (DQV)

The Data Quality Vocabulary (DQV) [4], defines concepts suitable for expressing data quality. Various different properties may be used to determine the quality of a dataset or distribution. Data quality measurements are defined based on dimensions and metrics. Dimensions are characteristics of the dataset related to data quality. An example is the availability of the dataset. Metrics define procedures to measure the quality in a given dimension. As an example, availability may be measured by the accessibility of a web service.

RDF Data Cube Vocabulary

The RDF Data Cube Vocabulary [31] defines concepts that enable the standard representation of multidimensional datasets in RDF. The vocabulary supports the representation of observations, dimensions and relevant metadata. The underlying model is the one of OLAP data cubes supported by data warehouses. Components of the structure of a data cube include dimensions, measures and attributes. The cube may be organized in slices. In one slice all the dimensions except one have a constant value.

2.2.3 SPARQL: A Protocol and Query Language for the Semantic Web

The SPARQL Protocol and RDF Query Languge [47] is a set of W3C standard specifications that provide protocols and languages that enable the querying and manipulation of RDF datasets through the web. The specifications include a query language, result formats, means to specify federated queries, a protocol for SPARQL services, etc. The most relevant specification for the present Thesis is the one that defines the query language [50].

In a SPARQL query, data is first extracted from the RDF graph using patterns and next processed and retrieved using declarative statements. The patterns used by SPARQL are called Basic Graph Patterns (BGP) and they are composed of sequences of triple patterns. A triple pattern is an RDF triple whose elements (subject, predicate and object) might have been replaced by a variable name. Triple patterns are specified in SPARQL with a syntax compatible with that of the Turtle encoding. During the evaluation, triple patterns are matched with the RDF graph to obtain sets of variable bindings. A variable binding assigns a value (an RDF term, i.e., an IRI, a blank node identifier or a literal) to each variable, to match one or various RDF triples of the RDF graph. After BGPs are matched and variable bindings are generated, other parts of the statement might be used to filter variable bindings, group results

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

and project and calculate new variables using primitive operators, functions, predicates and aggregate functions provided by the language. To illustrate the use of the language let us consider an RDF graph that represents employees and projects, with an structure similar to that of Figure 2.6. The following SPARQL query retrieves the total amount of funding of projects (percentage of the total budget) associated to each employee with a salary of at most 40000.

```
PREFIX ex: <http://citius.usc.es/example1>

SELECT ?ename (sum((?budget * ?percentage)/100) as ?funding)
WHERE
   { ?employee ex:worksIn ?project;
               ex:hasName ?ename;
               ex:hasArea ?salary.
     ?project ex:hasFunding ?f.
     ?f ex:hasBudget ?budget;
        ex:hasPercentage ?percentage.
     FILTER (?salary < 40000)
    }
GROUP BY ?ename
```

First, the BGP of the WHERE clause is matched with the RDF graph to produce a set of bindings for variables $\{?employee, ?project, ?ename, ?salary, ?f, ?budget, ?percentage\}$. Thus, for the RDF graph of Figure 2.6, the following variable binding is obtained:

$?employee = ex:employee/e231$

$?project = ex:project/p287$

$?ename = "Teresa"$

$?salary = 56000$

$?f =: f1$

$?budget = 245785.45$

$?percentage = 75$

Next, the FILTER clause is used to discard all the variable bindings where variable $?salary$ have a value lower than 40000. Next, the GROUP BY clause makes one group for each distinct value of variable $?ename$. Finally, for each group generated by clause GROUP BY, the SELECT clause generates a variable binding of two variables: the variable *ename* that defines

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

the group and a new variable $?funding$ whose value is computed by aggregating all the values of $?budget$ and $?percentage$ of bindings of the same group, using the specified expression. The result therefore is a set of bindings of two variables $\{?ename, ?funding\}$.

SPARQL queries are submitted to a SPARQL end point, using the protocol defined in the SPARQL standard and the results are obtained in some of the formats also defined by the standard. To evaluate the query, the SPARQL engine implements an algebra based on the SPARQL algebra described in the SPARQL standard. Providing a full description of the SPARQL language is far from the objective of the present subsection, which aims at just illustrating the main features required to understand the remainder chapters of the Thesis.

2.2.4 Mappings from the relational model to RDF

The relational model is the data modeling framework used by most of the datasets to be integrated in the linked data web. Therefore, mapping relational data to RDF is a piece of functionality that most linked data infrastructures have to deal with.

Currently, the W3C defines two complementary approaches to support the mapping from relational data to RDF. The Direct Mapping of Relational Data to RDF [7] uses directly the metadata extracted from the relational schema (table names, attribute names, etc.) to automatically define mappings from each relational tuple to RDF triples. Broadly speaking, each tuple generates a set of triples with a common subject, whose IRI is generated combining a base IRI with the table name and values of the primary key attributes. The predicate of each triple is generated from the name of an attribute of the table. Objects of triples are either literals recorded in the tuple or IRIs referencing other tuple (when a foreign key has been defined).

If the user needs to incorporate in the mapping a specific vocabulary, then the Relational Database to RDF Mapping Language (R2RML) [32] must be used instead of the direct mapping. With R2RML the user can specify highly customized mappings to define RDF views over relational data. To achieve this, R2RML enables the specification of rules that define the mapping of each relational element. Rules, called *TriplesMap*, are encoded in RDF using the Turtle encoding. Each rule has two parts:

- One *SubjectMap* that defines how the subject of each triple is generated from a IRI template that is filled with tuple attribute values.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- Various *PredicateObjectMap*, each of them defining how the predicate and object of a new triple is generated using the value of some tuple attribute.

Using R2RML, the access to relational data may be done either by materializing the triples generated from the relational tuples in a semantic data store or by providing direct access to a virtual graph whose triples are generated on demand when they are accessed. R2RML mappings may be defined on tables, views or queries. An example of the use of this mapping language will be shown in Chapter 5.

2.3 Geospatial Linked Data Approaches

The importance of the incorporation of semantic web technologies in the construction of future generation Spatial Data Infrastructures has already been noticed years ago [62]. During all these years, most of the research works aimed at the creation of new ontologies of interest for different applications in the geospatial and environmental domain. It is expected however that new efforts will now focus on problems related to effective and efficient data discovery and data access [78]. In the following subsections, some of these ontologies will be described and state of the art solutions for both vector feature and raster coverage querying will be analyzed. The section finishes with some concluding remarks that show the advantages of the application of semantic technologies in environmental applications and describe the main limitation of current technologies and the main challenges to be faced.

2.3.1 Geospatial and Environmental Ontologies

The representation of geospatial data may be performed today with state of the art semantic web technologies. In particular, the GeoSPARQL [86] standard of the OGC defines an ontology with core classes suitable for the representation of geospatial *Vector Features*. In particular, class $geo{:}Feature$ enables the representation of any geospatial feature, as defined by the OGC and ISO standard [65]. Geometries of *Vector Features* are represented with instances of class $geo{:}Geometry$. A property called $geo{:}hasGeometry$ may be used to link a geosaptial feature ($geo{:}Feature$) to its geometric representation ($geo{:}Geometry$). A new data type $geo{:}wktLiteral$, enables the textual encoding of the geometries using the WKT OGC encoding defined in the SFA standard [53].

Raster Coverages may be represented in RDF using classes of the RDF Data Cube Vocabulary [31], already discussed in the previous section of this chapter. It is noticed that RDF

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Cube enables the representation of any kind if multidimensional data cube, but it does not incorporate specific data representation mechanisms to enable the efficient representation of the geospatial and temporal dimensions based on regular samplings used by *Raster Coverages*.

Regarding geospatial metadata, the DCAT [3] ontology, already discussed in the previous section, has been extended to include the geospatial nature of the dataset, first in the DCAT-AP [27] ontology used by open data portals in the European Union, and next by its more complete GeoDCAT-AP [26] geospatial extension.

Regarding provenance metadata, the PROV-O ontology has been used in some specific domains like climate change reports [74]. An approach of more general purpose is adopted in [64], where PROV-O is extended with concepts extracted from the ISO19115 [59] geospatial metadata standard, to reach a generic provenance ontology to be used in the geospatial domain. If we restrict the scope to environmental observation data, the Semantic Sensor Network (SSN) ontology [49, 28] defines concepts to represent the record level metadata properties also defined by the O&M data model of the OGC. These concepts include classes to represent sensor data generation processes, features of interest, sampling features, properties and observations. In the environmental domain it is also worth mentioning the effort undertaken by NASA to create a general purpose vocabulary, called Semantic Web for Earth and Environment Technology Ontology (SWEET) [89].

Ontologies that attempt to incorporate top level semantics have also been proposed in some more specific application domains such as hydrology [39], geology [72] and mining [73].

2.3.2 Querying Semantic Vector Features

The querying of *Vector Features* with semantic technologies requires both the definition of relevant semantic representations and also an extension of the SPARQL query language. The representation of geospatial and environmental data with semantic technologies has already been discussed in the previous subsection, thus the discussion here focuses on the extension of the SPARQL query language. Some research initiatives have been dealing with geospatial and spatio-temporal SPARQL extensions during the last years [67]. As a consequence, an effort has been done in the OGC to standardize such an extension. The result of this effort is the GeoSPARQL standard [86].

GeoSPARQL defines an ontology to support the representation of *Vector Features* in the RDF data model. As it was already shown in the previous section, the ontology includes both classes to represent *Features* and their geometries, but also a data type that is used to

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

encode vector geometries as text. All the GeoSPARQL extensions are based on the Simple Feature Access (SFA) standard of the OGC [53], therefore, the text encoding is based on the Well-Known Text (WKT) encoding defined by this standard. The SFA defines also a collection of functions that enable the effective management of geometries. More precisely, with these methods, the user can perform declarative spatial data processing, including:

- The evaluation of geospatial topological predicates based on well-kwnown theoretical frameworks [25].
- The calculation of distances between geometries.
- The calculation of unions, intersections and differences of geometries and of collections of geometries.
- The retrieval of geometric properties such as the minimum bounding rectangle, the area of polygons and the length of lines.

Many current SPARQL engines provide support for GeoSPARQL. Additionally, GeoSPARQL implementations have also been undertaken on top of existing geospatial data management technologies, taking advantage of the good performance of these mature technologies. One example of such an approach is the STRABON GeoSPARQL implementation [67], which relies on the PostGIS [83] geospatial extension of PostgreSQL to store and query the geospatial data.

The names of the required functions will be introduced and explained through the Thesis in GeoSPARQL query examples as soon as they are required.

2.3.3 Querying Semantic Raster Coverages

GeoSPARQL [86] includes the artifacts of the OGC SFA standard [53], which is designed to support the management of *Vector Features*, therefore it is not a solution suitable for the management of *Raster Coverages*, and only their metadata could be efficiently managed [66]. To overcome these limitations of GeoSPARQL with raster data, some specific extensions have been proposed in the literature. Scientific SPARQL (SciSPARQL) [5, 6] extends SPARQL with array data management support. The extension provides support for arrays as first class citizens in the model and defines a collection of array data management functions that may be used within SPARQL. A similar approach is followed by GeoSPARQL+ [55], which defines

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

a raster data type and raster data management functions. Both solutions follow an approach similar to that of some geospatial extensions of relational DBMSs, like PostGIS [83]. As it was already discussed in Subsection 2.1.4, those approaches for raster data management do not achieve the performance of specialized raster or array data management systems. Besides, the user has to deal with complex query interfaces consisting of large lists of functions to perform specific tasks.

It is finally noticed also that neither SciSPARQL nor GeoSPARQL+ reach a level of maturity in their implementations comparable to that of most existing GeoSPARQL implementations.

2.4 Concluding Remarks

The application of the technologies of the semantic web in the geospatial domain bring new opportunities and challenges [85]. Thus, the use of semantic technologies to implement data discovering functionalities is something that has already been tried by many researchers [106, 105, 96, 8, 41]. The general approach is to use ontologies to encode conventional and geospatial metadata, and use semantic web technologies to solve the data discovery problems and retrieve results that are richer in terms of semantics. Regarding data access, a main advantage of the use of semantic technologies is related to the new chances that are opened to achieve better semantic interoperability during data integration tasks [103, 23, 71, 45, 91, 90].

However, to meet the above opportunities for the development of better data discovery and access systems over linked geospatial data sources, some important challenges have to be faced [99].

Data Representation and Querying: Geospatial *Vector Features* may be represented and queried using GeoSPARQL. However, the performance is still not comparable to that of conventional geospatial data management tools, thus some work has still to be done in the developing of geospatial data access methods and geospatial query processing and optimization [85]. Regarding *Raster Coverages*, only few approaches [5, 6, 55] have attempted their management with SPARQL extensions, which are not yet mature enough to be used in application developments. Additionally, these solutions suffer from having complex user interaction and poor performance. As a consequence, the representation and querying of *Raster Coverages* would have to be done with conven-

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

tional solutions, such as RDF Cube [31], or with solutions designed for *Vector Features*, such as GeoSPARQL, resulting in implementations with very poor performance.

Data Integration and Federated Querying: Relational data may be integrated into the semantic web using existing relational to RDF mapping approaches [7, 32]. Direct geospatial extensions of those mappings may be used to incorporate geospatial datasets with the form of *Vector Feature Types*. The incorporation may be done either virtually, by directly querying the geospatial data management system with GeoSPARQL through the mapping [17], or physically by running an Extract, Transform and Load (ETL) process that loads the resulting RDF in a GeoSPARQL system [68]. In any case, any of the above solutions and some other approaches for federated semantic querying [46] restrict their support to *Vector Features*.

As a synthesis of this chapter, it is clear that GeoSPARQL implementations are the way to support declarative querying of linked geospatial datasets in the web of data, although research work is still required to reach solutions with good performance for large datasets. However, despite of the very large amount of data available nowadays with the form of *Raster Coverages*, with great potential to be very relevant in existing an upcoming applications, their support by semantic web technologies have been neglected in general, as it was also for many years in the area of spatial databases.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 3

PROBLEM DESCRIPTION

This chapter provides a more detailed description of the main problems for which the Thesis provides effective and efficient solutions. More precisely, the problem of accessing and combining vector geospatial features and raster geospatial coverages in a linked data infrastructure will be presented. Besides, this chapter shows how current linked data technologies could solve the problem and also the main limitations of such a solution that have to be overcame. To achieve this, first, Section 3.1 provides an example of an GeoSPARQL query that combines *Vector Features* with a *Raster Coverage* over the network and it outlines the main functionalities that a relevant GeoSPARQL engine should provide to evaluate such queries. Next, in Section 3.2, a query plan that a current GeoSPARQL engine would produce is used to explain the main problems that motivate the objectives of the present Thesis.

3.1 Query Example

A typical example of a query of interest for the implementation of on-line analytical processing would aggregate the values of a *Raster Coverage* that lie inside each of the *Vector Features* of a given *Feature Type*.

To illustrate one such query let us consider a geospatial *Vector Feature* dataset consisting of the municipalities of a given region (let us say the region of Galicia in the north-west of Spain). For each municipality, the dataset records its identifier, its name, its geometry and the link to the corresponding dbpedia resource, that provides additional descriptive data. Let us assume also that there is a geospatial *Raster Coverage* that records for each location of Galicia and for each hour, estimated measures of meteorological variables such as air

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

```
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX feat:<http://www.mydomain.com/features/>
PREFIX cov:<http://www.mydomain.com/coverages/>
PREFIX geo:<http://www.opengis.net/ont/geosparql#>

SELECT ?mname ?abs ?time
       (avg(?temp) as ?avgtemp) (min(?temp) as ?mintemp) (max(?temp) as ?maxtemp)
WHERE
 {?mun a feat:municipality;
       feat:hasName ?mname; geo:hasGeometry ?mgeo; feat:hasDBPedia ?mdbpedia.
  ?mgeo geo:asWKT ?mgeowkt.
  SERVICE <http://dbpedia.org/sparql> {?mdbpedia dbo:abstract ?abs.}
  ?s a cov:Meteo;
     geo:hasLocation ?sloc; cov:Temp ?temp; geo:hasTime ?time.
  ?sloc geo:asWKT ?slocwkt.
FILTER (geof:sfContains(?mgeowkt, ?slocwkt) && ?time >= t1 && ?time <= t2)
 }
GROUP BY ?mname ?abs ?time
```

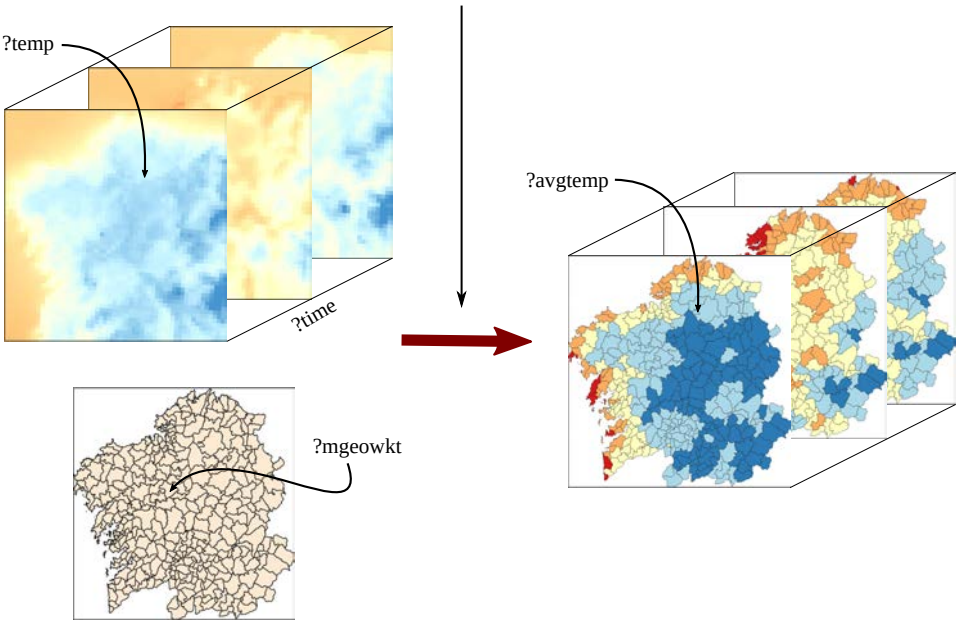


Figure 3.1: Example of a GeoSPARQL query that combines vector features with a raster coverage.

temperature, wind direction and speed, and humidity. A GeoSPARQL query that computes the minimum, maximum and average of the air temperature for each municipality and for each hour is shown in Figure 3.1. The query combines data obtained from the three data sources, namely, the vector feature dataset, the raster coverage and the dbpedia service, filtering the data for a specific time period of interest (between $t1$ and $t2$) and restricting the combinations between municipalities and raster cells to those for which the cell is contained inside the municipality geometry. It is noticed that both *Vector Features* and raster cells are treated by the query as geospatial *Vector Features* represented in GeoSPARQL, whose geometric properties are encoded with Well-Known Text (WKT) and accessed through GeoSPARQL $geo{:}asWKT$ properties. The GeoSPARQL $geof{:}sfContains$ function is used to test if the point that represents the location of each raster cell is contained inside the polygon that represents the bounds of each municipality. After the combination of the three datasets, the resulting data is grouped by municipality and time to enable the aggregation of raster temperatures inside each group (municipality, time pair) and generate this way the desired result. The temperature property (?temp) of the input raster coverage and the geometry of each municipality (?mgeowkt) are graphically represented in Figure 3.1, as it is also the output average of temperature (?avgtemp) of each municipality and hour.

To access the input data and achieve an efficient evaluation of the above query, a GeoSPARQL query engine should provide:

- Strategies and languages to provide mappings between geospatial *Vector Features* and raster cells to RDF, in a way similar to the already existing mappings between relational data and RDF [7, 32].

- Wrappers that enable the retrieval of vector features and raster coverage cells from standard OGC web services over the network and perform the required transforamtion to RDF according to the above mappings.

- Query processing algorithms that provide efficient implementation for geospatial join and aggregation operations that involve vector features and raster coverages in a federated architecture.

- Query optimization strategies that obtain query plans that leverage the existence of the above algorithms.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

3.2 Query Processing and Optimization

The query of the previous section has to be parsed by the query engine and the optimizer must generate an efficient query plan for its evaluation. The query plan has the form of a tree whose nodes represent algorithms that implement operations of the SPARQL algebra [50]. This section discusses the query plan that a conventional SPARQL query engine would generate for the above query, showing a basic optimization strategy, and explaining the main problems that would cause low performance with such a straightforward solution. A full description of the SPARQL algebra and of all the related optimization and query processing issues are out of the scope of this section.

An initial query plan composed of SPARQL operations resulting from the parsing of the query of Figure 3.1 is shown in Figure 3.2. The general functionality of each of the operations of the query plan is briefly described below.

Basic Graph Pattern (BGP)

This operator matches a Basic Graph Pattern (BGP) with the RDF graph of the data source to produce a multiset of variable bindings, where each variable binding links RDF terms to result variables. A BGP is composed of a set of triple patterns. A *triple pattern* is an expression of the form $(s\ p\ o)$, where:

- s is either a variable or an RDF term. An *RDF Term* is either an IRI (Internationalized Resource Identifier) or an RDF Literal or an RDF blank node.
- p is either the IRI of an RDF predicate or a variable
- o is either a variable or an RDF term

Each triple pattern of the BGP is matched with RDF triples of the data source to generate a multiset of variable bindings. A *Variable Binding* is a mapping from a set of variables to a set of RDF terms, i.e., it has the form $(V_i \rightarrow a_i)$, where each V_i is a variable and each a_i is an RDF term. As an example, in the municipalities data source, the match of the triple pattern $(feat : municipality/code = 15059\ feat : hasName\ ?name)$ generates the variable binding $(?name \rightarrow \text{“}Ordes\text{”})$, since "Ordes" is the name of the municipality with code 15059.

Each variable binding generated from the match of each triple pattern has to be merged with all the compatible variable bindings generated by all the other triple patterns of the BGP.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

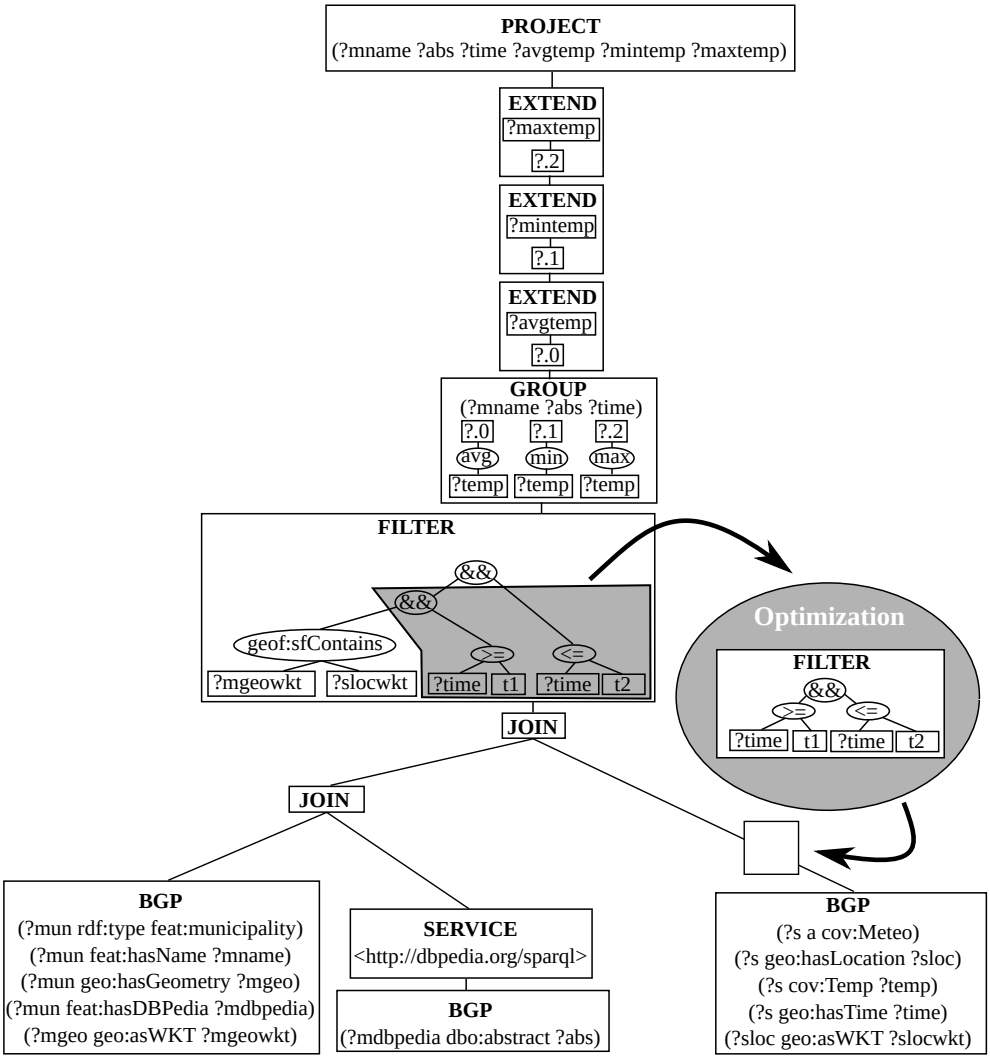


Figure 3.2: Query Plan and Optimization.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Two variable bindings are compatible if either they do not have variables in common, or if their common variables are assigned to identical values. Thus, for example, if a triple pattern generates the variable bindings

$\{(?mun \rightarrow \text{“}feat : municipality/code = 15059\text{”}),$
$(?mun \rightarrow \text{“}feat : municipality/code = 15086\text{”})\}$

and another triple pattern generates the bindings

$\{(?mun \rightarrow \text{“}feat : municipality/code = 15059\text{”}, ?mname \rightarrow \text{“}Ordes\text{”}),$
$(?mun \rightarrow \text{“}feat : municipality/code = 15086\text{”}, ?mname \rightarrow \text{“}Trazo\text{”}),$
$(?mun \rightarrow \text{“}feat : municipality/code = 9999\text{”}, ?mname \rightarrow \text{“}Unnamed\text{”})\}$

then the BGP composed of both triple patterns would obtain the following multiset of variable bindings

$\{(?mun \rightarrow \text{“}feat : municipality/code = 15059\text{”}, ?mname \rightarrow \text{“}Ordes\text{”}),$
$(?mun \rightarrow \text{“}feat : municipality/code = 15086\text{”}, ?mname \rightarrow \text{“}Trazo\text{”})\}.$

In the query plan of Figure 3.2, the BGP operators access the three input datasets (municipalities, dbpedia and coverage) to generate variable bindings with their data.

SERVICE

This operator submits a query plan to a service that implements a SPARQL end-point in a federated architecture. In our example query plan, the SERVICE operator evaluates a BGP operator in the SPARQL end-point of DBpedia, to obtain the abstract of each municipality.

JOIN

This operator enables the combination of two multisets of variable bindings, generated by two other operators. Each variable binding of the first input is combined with all the compatible variable bindings of the second input. Thus, informally, if the two input multisets have variables in common, then the operator performs what is known in relational theory as *NATURAL JOIN*. On the other hand, if the input multisets do not have variables in common, then the operator performs a *CARTESIAN PRODUCT* of the inputs. In our example query plan, the first JOIN operator combines the data obtained for each municipality with its abstract obtained from DBpedia (It performs a NATURAL JOIN in this case). Next, the second JOIN combines each municipality (already with its abstract) with each of the cells of the input raster coverage (it performs a CARTESIAN PRODUCT in this case).

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

FILTER

This operator retrieves the variable bindings of the input multiset for which a given Boolean expression evaluates to true. In our example, the FILTER operator is used to obtain the desired combinations of input municipalities with the coverage cells, i.e., those in which the cell is contained into the municipality geometry and the cell time is between instants $t1$ and $t2$.

GROUP

This operator enables the application of aggregate functions to sets of values generated by grouping together the input variable bindings that have identical values in some of their variables. In our example query plan, the GROUP operator is used to first generate groups of input variable bindings that have identical values in variables $?mname$, $?abs$ and $?time$, i.e., raster coverage cells of the same time instant and contained in the same municipality are grouped together. Next, aggregate functions avg, min and max are used to compute the average, minimum and maximum temperature for each group, generating new variables $?.0$, $?.1$ and $?.2$. At this stage, we have for each municipality and time instant, the name of the municipality, its abstract, the time instant and the average, minimum and maximum temperature.

EXTEND

This operator enables the evaluation of an expression for each of the input variable bindings, to generate values of a new variable. In our example query plan, this operator is used to generate new variables $?avgtemp$, $?mintemp$ and $?maxtemp$ to record the values generated by the aggregate functions generated by the GROUP operator.

PROJECT

This operator restricts the variables of the input multiset. In our example query plan, it obtains only the variables $?mname$, $?abs$, $?avgtemp$, $?mintemp$ and $?maxtemp$, discarding the variables $?.0$, $?.1$ and $?.2$ generated by the GROUP operator.

From the description given above, it follows that our example query plan generates all the combinations between input municipalities and raster coverage cells, before restricting, in the FILTER operator, to those required to generate the desired output. An optimization strategy

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

supported by most query engines would apply the heuristic of moving FILTER operations as much as possible down in the query plan. In our example, the expression that checks if the coverage cell lies between instants $t1$ and $t2$ might be moved just on top of the BGP operator that obtains the data from the coverage data source. This way, many combinations between municipalities and coverage cells are avoided in the JOIN operation, which leads to a more efficient query plan. This well-known optimization strategy is illustrated in Figure 3.2 and it would be supported by any SPARQL query engine.

However, there are other more specific optimizations that cannot be applied by a conventional SPARQL query engine, as they rely on the query capabilities of the geoespatial data sources. In particular, the following optimizations would be desirable and cannot be supported by state of the art conventional SPARQL query engines.

- Geospatial feature data sources generally support the specification of filters, which include geospatial filters. This is the case for example of services that implement the WFS standards. Therefore, a BGP operator involving triple patterns of a single geospatial feature type, possibly with a FILTER operator on top of it might be translated to a WFS request, avoiding the transference of much data over the network.

- In a similar manner, geospatial coverage data sources, like those implementing WCS and WCPS standard interfaces, support both filtering over the spatial and temporal dimensions and (in the case of WCPS) geoprocessing. Therefore, sequences of BGP and FILTER operators could also be translated to a WCS or WCPS request to avoid much data transfer over the network.

- The general purpose SPARQL JOIN operator, when the input multisets of variable bindings do not have variables in common, has to generate a Cartesian Product by applying a general purpose, but highly inefficient, nested loop strategy. This is the case of the top most JOIN operator of our example query plan. A more efficient SPATIAL JOIN operator could leverage the spatial filtering capabilities of the underlying WCPS service to implement a smarter join strategy, that avoids the generation of all the combinations between each input municipality and each raster cell.

- The geoprocessing capabilities of the WCPS interface include the computation of aggregate functions over the values of the raster cells. Such capabilities are not leveraged even if an efficient SPATIAL JOIN operator is available. To take advantage of such

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

geoprocessing power of the underlying data source, more advanced algorithms need to be implemented in a specific Vector-Raster SPATIAL GROUP operator.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 4

ARCHITECTURE OF A FEDERATED VECTOR-RASTER GEOSPARQL QUERY ENGINE

In this chapter, a description of the components architecture of the GeoLD query engine is described. Figure 4.1 provides a graphical representation of the main software layers of the architecture, detailing for each layer the components that must be added to extend an already existing SPARQL query engine. Those extensions affect both SPARQL optimization and query processing. As it is shown in the figure, first a *SPARQL Query Parser* generates a first query tree from the SPARQL query expression. Next, the *SPARQL Optimizer* chooses a query plan that enables achieving an efficient evaluation of the query. Such an evaluation is performed by the *SPARQL Evaluation Engine* by obtaining the data from the data sources and applying the algorithms of the operations of the query plan. A more detailed description of both the data sources and the main extensions of the query engine are given in the following sections.

4.1 Web Services for Geospatial Features and Coverages

As it was already discussed in Subsection 2.1.4, the general purpose OGC service interfaces that enable declarative open data access to geospatial features and coverages are the *Web Feature Servcie* (WFS) and the *Web Coverage Service* (WCS). Additionally, the *Web Coverage*

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

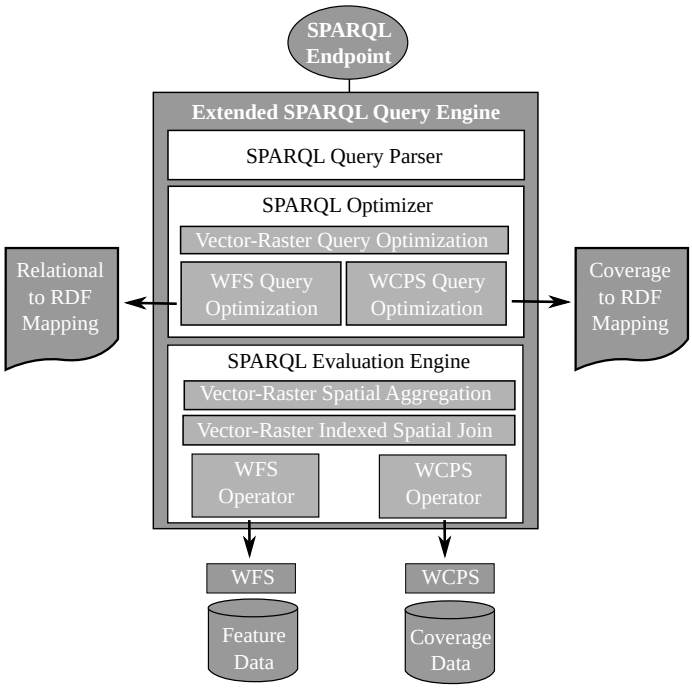


Figure 4.1: Extended SPARQL Query Engine Architecture

Processing Service (WCPS) provides all the data accesss funcionality of the WCS and additional declarative geoprocessing functionality over geospatial coverages. Thus, the current GeoLD design and implementation uses WFS services to obtain vector geosaptial features and WCPS to access raster geospatial coverages.

A complete implementation of the current 2.0 version of the WFS standard [100] is a complex interface that supports most of the typical data management functionality that any application might need over a geospatial feature data store. In particular, a WFS 2.0 enables: i) the description of the data contents, including the list of feature types and the metadata of their properties, ii) the querying of the data with both inline and stored query expressions and iii) the management of the stored data through data insert, update and delete actions. Besides, locking operations are also supported to enable the implementation of required concurrent data access. Despite of the above WFS complete functionality, since the supporting of join operations in the WFS interface is optional, the current GeoLD implementation leverages only basic scans

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

of feature types with optional filters, that might be either conventional or geospatial.

The WCPS language [12, 13] enables the declarative specification of statements to retrieve and process geospatial raster coverages. The supported functionality includes trimming and slicing operations over the spatial and temporal dimensions of the coverages and the evaluation of algebraic expressions over the values of the coverage bands. Algebraic expressions may combine operations and functions defined on the alphanumeric data types of the coverage bands with aggregate functions supported by the system. The current implementation of GeoLD makes use such WCPS functionality to delegate both filtering and aggregation operation on the WCPS engine.

4.2 Extension of the SPARQL Evaluation Engine

The *SPARQL Evaluation Engine* must provide efficient algorithms that implement the operations of the SPARQL algebra [50] used to construct the query plans. To enable the access to datasets of geospatial features and coverages, the engine must implement new operators whose functionality is delegated to WFS and WCPS services. Additionally, the efficient combination of vector features and raster coverages in queries can only be achieved by new algorithms that perform smarter join and aggregation operations. The functionality of the new operators implemented in the *SPARQL Evaluation Engine* is briefly sketched below.

WFS Operator

It leverages the basic scan and filtering functionality provided by the WFS service to provide over the network data access to vector feature datasets. Although the current capabilities of a complete WFS 2.0 service enable the specification of queries that involve various vector feature datasets, the present design and implementation of GeoLD exploits only basic data scans and filters over single feature collections of a given feature type. Thus, SPARQL BGP and Filter operators that refer to a single feature type may be replaced by a single WFS Operator that delegates the scan and filtering to the WFS interface. Performing the filtering in the service decreases the amount of data to be transmitted over the network and therefore improves the performance.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

WCPS Operator

It leverages the filtering and processing capabilities of WCPS services to enable raster coverage data access over the network and also to delegate part of the heavyweight raster processing to the WCPS service, moving this way processing closer to the data and consequently improving the overall system performance. In the current GeoLD implementation, the WCPS operation may be used in three different ways: i) to achieve scan and filtering over raster coverages, including filtering over both spatial and temporal dimensions, but also speeding up filtering over the coverage band values. ii) to provide spatial indexed access to the raster coverages as part of the implementaion of vector-raster indexed spatial joins. iii) to provide the spatial indexed access and aggregation functionality required by an advanced vector-raster spatial group operator.

Vector-Raster Indexed Spatial Join

Basic join algorithms that are independent of the join condition and of the characteristics of the input data streams are based on nested loops. However, nested loop joins are highly inefficient compared to other join strategies that take advantage of the knowledge related to join predicate and also of the knowledge of existing data structures. The current GeoLD implementation incorporates an indexed join strategy to implement spatial joins between vector feature collections and raster coverages. Performing indexed access in the raster coverage using the minimum bounding rectangle of each vector feature discards the testing of many spatial intersection conditions between raster pixels and vector geometries, improving the join performance.

Vector-Raster Spatial Group

A commonly required query in many applications performs the aggregation of raster coverage values over the extension of vector zones. An example of this is the retrieval of the average temperature in each municipality. To achieve this, first, the vector feature collection and the raster coverage has to be joined, possibly by using the vector-raster indexed spatial join operator described above. Next, the raster pixels have to be grouped by each municipality and finally the average aggregate function has to be computed inside each group. Such an algorithm does not leverage at all the geoprocessing capabilities supported by the underlying WCPS service, since the average of the pixel values has to be completely computed by the

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

SPARQL engine. Because of this, a new operator is added to the engine that implements a more advanced spatial aggregation algorithm between vector zones and raster pixels, allowing the partial delegation of the aggregate computation to the WCPS service, and therefore improving the overall system performance.

4.3 Extension of the SPARQL Optimizer

The *SPARQL Query Parser* already provided by the SPARQL query engine generates a query tree of SPARQL algebra operations, which does not include the new operations that have been added to the extended *SPARQL Evaluation Engine*. Therefore, new optimization stages must be added that incorporate the new operators in the query plans and take advantage of their new functionality. Such new functionality includes efficient access to vector features and raster coverages over OGC web services, and efficient algorithms for vector-raster spatial join and group. More precisely, the following optimization must be added.

WFS and WCPS Query Optimization

The maximum subtrees generated by the parser that could be translated to a WFS request have to be replaced by a new *WFS Operator*. To achieve this, the query tree has to be processed to isolate in a single BGP all the triple patterns referring to a single feature type. Additionally, if some Filter involves variables associated only to one such BGP, then Filter must be placed just on top of the BGP. Finally, isolated BGPs and couples of connected BGP and Filter operators have to be replaced by relevant *WFS Operators* that generate the same output. To be able to identify the triple patterns that refer to each feature type of a WFS service, a mapping between feature types and RDF must be provided. Such mappings are already supported by straightforward applications of direct mapping [7] and relational to RDF [32] mapping approaches. The current implementation of GeoLD supports only a direct mapping approach. A similar optimization has been implemented to replace BGP and Filter operators that refer to a single raster coverage by a single *WCPS Operator*. However, in this case, it has not been proposed yet any mapping approach between raster coverages and RDF. As part of this optimization stage, both a direct mapping approach and a Coverage to RDF Mapping Language (C2RML) have been defined and implemented.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Vector-Raster Query Optimization

Beyond the above optimizations that leverage the use of available OGC web services through new WFS and WCPS operators, specific situations should be detected to incorporate also the new join and group operators described above. Those optimizations should find query trees that combine JOIN, Filter, Group, WFS and WCPS operators, and that could be replaced by equivalent trees using the new advanced operators. The current implementation of GeoLD does not include such optimization and it is part of upcoming work beyond the present Thesis.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

CHAPTER 5

MAPPING GEOSPATIAL DATASETS TO RDF

The incorporation of geospatial data sources into GeoSPARQL query engines requires the definition of mappings from geospatial data models to RDF. Since two major types of geospatial datasets have been identified, namely, *Vector Feature Types* and *Raster Coverages*, two different mapping approaches have to be considered.

5.1 Mapping Geospatial Vector Features to RDF

The W3C proposes both a direct mapping [7] from relational datasets to RDF and also a Relational Database to RDF Mapping Language (R2RML) [32]. The former provides the default transformation from relational data to RDF, that uses the names of relations and attributes as part of the vocabulary of the generated RDF triples. On the other hand, R2RML enables the user to define customized mappings with a specific target vocabulary. The assumption of GeoSPARQL [86] as the target ontology enables the quite straightforward use of the above two approaches to define mappings from geospatial vector feature datasets to RDF. Given that definition of mapping approaches for geospatial features has not been identified as a relevant contribution of the present Thesis, the current version of GeoLD supports only direct mappings for geospatial vector features.

Direct Mapping

In a direct mapping [7] from the contents of a relation to RDF each tuple of the relation generates a set of RDF triples, all of them with a common subject. The subject of all those

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

triples identifies the resource represented by the tuple, therefore, it is represented with an IRI constructed from a base IRI, the table name and a representation of the primary key for the tuple. Thus, for example, for a relation "*Employee*" with a primary key defined by attribute "*ecode*", the subject would be an IRI of the form "*baseIRI*:*Employee*/*ecode*=*e*4", where *e*4 is the value of the attribute *ecode* for the current tuple. One of the triples generated for each tuple defines the type of the generated resource. In our example, such a triple would be "<*Employee*/*ecod*=*e*4> *rdf*:*type* <*Employee*>". The remainder triples generated for each tuple are used to represent the values of each of the attributes. Each such triple has the subject corresponding to the tuple, a predicate generated from the table and attribute name and an object consisting of a literal that represents the attribute value. Thus, for example, if "*John*" is the value of attribute "*name*" for Employee *e*4, then the following triple is generated: "<*Employee*/*ecode*=*e*4> <*Employee*#*name*> "*John*"". If the table does not have a primary key, then an RDF blank node is used as the subject for all the generated triples of each tuple. Primary keys with more than one attribute generate subjects with more than one pair attribute=value. Attributes with null values do not generate relevant triples in the output. Finally, an additional triple is generated for each foreign key defined for the relation, that enables the linking between the resource of the current tuple and the resource of the referenced tuple. Thus, for example, if attribute "*dept*" is a foreign key that references the primary key "*dcode*" of the "*Department*" relation, then the relevant generated RDF triple is as follows: "<*Employee*/*ecode*=*e*4> <*Employee*#*ref*−*dept*> <*Department*/*dcode*=*d*8>".

If we assume GeoSPARQL [86] as the default top ontology and vocabulary for the representation of geospatial features, the application of a direct mapping from feature datasets to RDF should use as much as possible GeoSPARQL elemnets. Therefore, the class associated to any geospatial feature should be a subclass of GeoSPARQL "*geo*:*Feature*". Besides, geometric properties of features should be represented with instances of some subclass of GeoSPARQL "*geo*:*Geometry*", such as "*sf*:*Point*" and "*sf*:*Polygon*", which in turn use literals of type "*geo*:*wktLiteral*" to encode the geometries as text. GeoSPARQL properties "*geo*:*hasGeometry*" and "*geo*:*asWKT*" are used to link features with their geometries and the geometries with their text encoding, respectively. To illustrate the direct mapping from a geospatial feature dataset to RDF let us consider the feature type *Mun* depicted in Figure 5.1, which records municipalities of the Spanish region of Galicia. Only three tuples are represented in the left side of the figure, and its direct mapping to RDF is shown in the right side of the figure. Class *Mun*, obtained from the name of the feature type, is defined as a

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Mun

mcode	name	geom
...	...	...
15059	Ordes	
...	...	...
15078	Santiago de Compostela	
...	...	...
15086	Trazo	
...	...	...

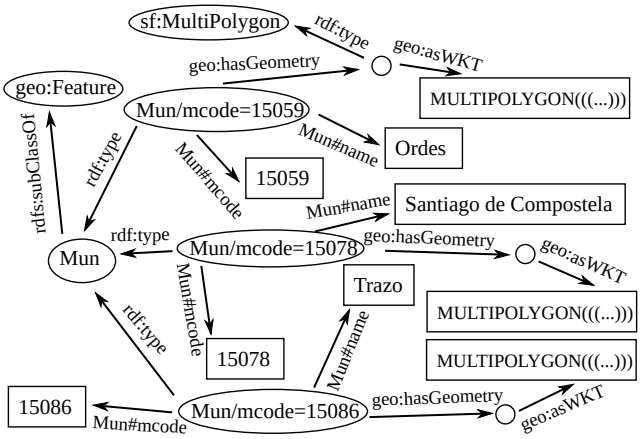


Figure 5.1: Direct mapping from geospatial features to RDF.

subclass of GeoSPARQL "$geo{:}Feature$". A set of RDF triples is generated for each tuple, all of them with the same subject "$Mun/mcode = c$", where c is the value of the $mcode$ attribute, primary key of the feature type. The values of conventional attributes $mcode$ and $name$ are represented with RDF literals and linked to the subject through properties "$Mun\#mcode$" and "$Mun\#name$", respectively. The values of the $geom$ attribute (of a $MultiPolygon$ geometric data type) are represented with instances of GeoSPARQL class "$sf{:}MultiPolygon$". Those instances are also linked to the OGC standard Well-Known Text (WKT) representation through GeoSPARQL property "$geo{:}asWKT$".

Relational Database to RDF Mapping Language (R2RML)

Using R2RML [32], highly customized RDF views of relational data structures may be generated. An R2RML document is written in RDF, using Turtle syntax [88], and its content are composed of rules called $TriplesMap$, which define the mapping from the contents of each table to RDF. Each $TriplesMap$ has three parts: i) a $LogicalTable$ that defines the contents to be mapped, i.e., either a table or a view or the results of an SQL query. ii) A $SubjectMap$ that specifies how the subject of the triples generated for each tuple is constructed from the tuple attribute values. iii) Multiple $PredicateObjectMap$ that define how the values of attributes and foreign keys are represented with RDF triples. A full description of this language is out of the scope of this document, however, for the purposes of the Thesis, an illustrative example

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

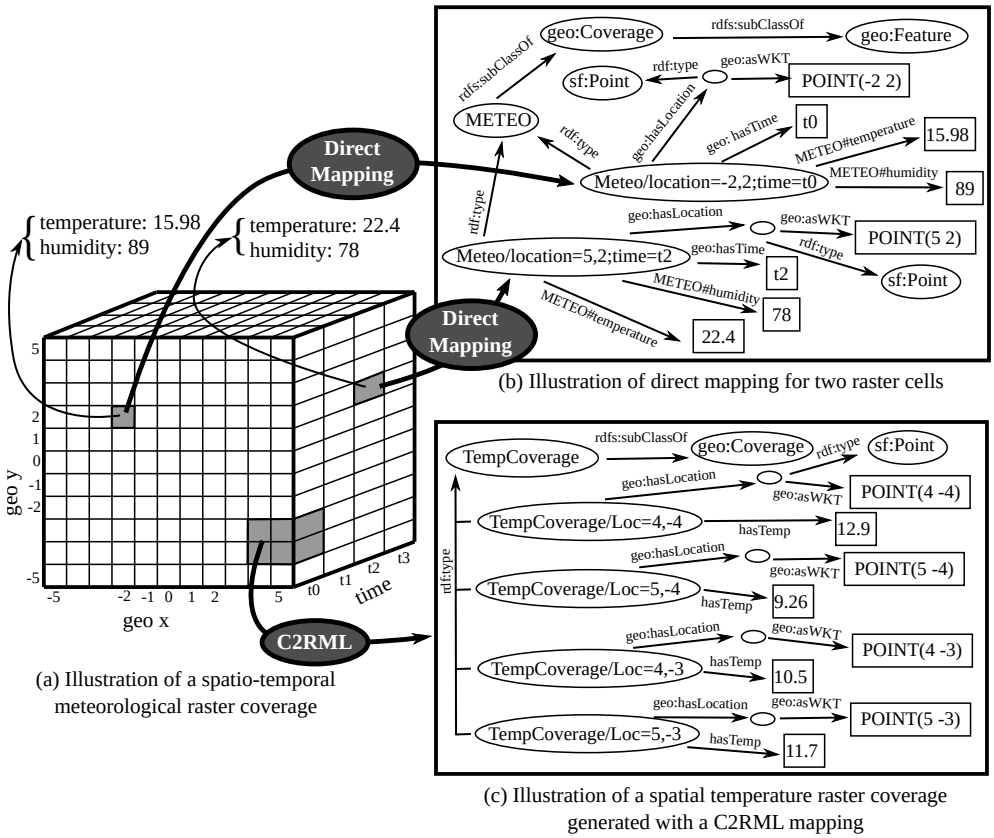
```
@PREFIX rr: <http://www.w3.org/ns/r2rml#>.
@PREFIX ex: <http://citius.usc.es/example1>.

<TriplesMap1>
  rr:logicalTable [rr:tableName "Mun"];
  rr:subjectMap [
     rr:template "http://citius.usc.es/example1/Municipality/{mcode}";
  rr:class ex:Municipality;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:hasName;
    rr:objectMap [rr:collumn "name"];
  ];
  rr:predicateObjectMap[
    rr:predicate ex:hasCapital;
    rr:objectMap [
      rr:template "http://citius.usc.es/example1/Town/mun={mcode}&town={capital}";
    ];
  ].

<TriplesMap2>
  rr:logicalTable [rr:tableName "Town"];
  rr:subjectMap[
    rr:template "http://citius.usc.es/example1/Town/mun={mun}&town={tcode}";
    rr:class ex:Town;
  ];
  rr:predicateObjectMap[
    rr:predicate ex:hasMunicipality;
    rr:objectMap [
      rr:parentTriplesMap <TriplesMap1>;
      rr:joinCondition[
        rr:child "mun";
        rr:parent "mcode";
      ];
    ];
  ];
  rr:predicateObjectMap [
    rr:predicate ex:hasName;
    rr:objectMap [rr:collumn "name"];
  ].
```

Figure 5.2: Example of Relational Database to RDF Mapping Language (R2RML).

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

is given next. Let $Mun(mcode, name, capital)$ be the scheme of a relation that records data of municipalities. The tuple of each municipality records its *mcode* (primary key), its *name* and its *capital*. The *capital* of a municipality is used, together with the *mcode*, to reference a town recorded in another relation. In relation $Town(mun, tcode, name)$, attribute *mun* records a reference to the municipality it belongs to, attribute *tcode* is a town code, being the pair $(mun, tcode)$ the primary key of this relation, and the *name* of each town is also recorded. An R2RML document that maps tuples of these two relations to RDF is shown in Figure 5.2. The document is composed of two *TripleMaps*, one for each of the above relations. The first *TripleMap* "<*TriplesMap*1>" generates RDF triples for each tuple of relation *Mun*. Clause "*rr*:*logicalTable*" is used to define the set of tuples that is going to be processed, which might be either a whole relation (as in the current document) or the result of a view or the result of an SQL query. Clause "*rr*:*subjectMap*" specifies how the subject of all the triples is generated from the attributes of each tuple. In our example, a template is used to generate the IRIs from the value of attribute *mcode*. Besides, it is also defined that the relevant object is of type "*ex* : *Municipality*". Therefore, a triple of the form "*s* *rdf*:*type* *ex*:*Municipality*" is generated for each tuple, where *s* is the relevant subject generated with the template. Each clause "*rr*:*predicateObjectMap*" is used to generate RDF triples to represent the values of the attributes of the current tuple. Thus, with the first "*rr*:*predicateObjectMap*" of "<*TriplesMap*1>" a triple of the form "*s* *ex*:*hasName* *n*" is generated for each tuple, where *s* is the subject generated with the template specified in the "*rr*:*subjectMap*" and *n* is the value of attribute *name*. On the other hand, the second "*rr*:*predicateObjectMap*" of "<*TriplesMap*1>" is used to generate a reference from the current municipality to its capital town. Thus, for each tuple, a triple of the form "*s* *ex*:*hasCapital* *o*" is generated, where *s* is the subject generated as explained above, and *o* is the IRI of the relevant town, generated with a new template from attributes *mcode* and *capital*. As it may be noticed, the format of the above IRI matches the one used by the template specified in the "*rr*:*subjectMap*" of "<*TriplesMap*2>", which is used to generate the subjects of the triples generated from relation *Town*. To finalize this example, the first "*rr*:*predicateObjectMap*" of "<*TriplesMap*2>" shows how to use a join operation in a "*rr*:*objectMap*" clause to generate object references pointing to the subjects generated by "<*TriplesMap*1>", implementing this way the link between each town and its municipality.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



(a) Illustration of a spatio-temporal meteorological raster coverage

(b) Illustration of direct mapping for two raster cells

(c) Illustration of a spatial temperature raster coverage generated with a C2RML mapping

Figure 5.3: Illustration of geospatial raster coverage to RDF mappings.

5.2 Mapping Geospatial Raster Coverages to RDF

The mappings between geospatial raster coverages and RDF supported by the current implementation of GeoLD are based on adaptions of those defined for relational data by W3C. Thus, a direct mapping is provided as the default mapping and, contrary to what happens with geospatial features, a language based on R2RML, called Coverage to RDF Mapping Language (C2RML), is also defined that enables the programmer to define highly customized mappings.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Direct mapping from geospatial coverages to RDF

To illustrate the application of direct mappings to geospatial coverages, Figure 5.3(a) shows a spatio temporal coverage called *METEO* and Figure 5.3)(b) shows the RDF graph corresponding to the direct mapping of a couple of its raster cells. Each coverage is modeled as a subclass of class "*geo*:*Coverage*", which is in turn subclass of GeoSPARQL class "*geo*:*Feature*". These facts have to be represented in the generated RDF. Besides, in a way similar to that explained above for relational/feature data, in a direct mapping, each cell of the coverage is mapped to a set a RDF triples, all of them with a common subject. Given that the subject of each triple must identify the resource associated to the corresponding raster cell, the IRI must contain the coverage name and the values of the coverage dimensions of the cell. Thus, in the example, the subjects of the triples generated for each cell have the form "*Meteo*/*location*=*c*,*time*=*t*", where *c* is a text representation of the geospatial coordinates of the centroid (geometric center) of the cell and *t* is a text representation of the value of the temporal dimension. For each cell, triples are added to represent: i) the fact that the cell is an instance of the coverage class, ii) the values of all the dimensions for the cell and iii) the values of all the bands of the coverage for the cell. Thus, in the example, a triple "*s rdf*:*typeMETEO*" is added for each cell, where *s* is the subject generated for the cell. Besides, GeoSPARQL properties "*geo*:*hasLocation*" and "*geo* : *asWKT*" are used in triples to represent the point at the centroid of each cell. Similarly, property "*geo*:*hasTime*" is used to represent the value of the temporal dimension of the cell. Finally, properties "*METEO*#*temperature*" and "*METEO*#*humidity*" are used to represent the values of the two bands of the coverage for each cell.

Coverage to RDF Mapping Language (C2RML)

One of the contributions of this Thesis is the definition and implementation in GeoLD of a Coverage to RDF Mapping Language (C2RML). The language is based on the R2RML defined by W3C for relational data. Main differences with R2RML are: i) the use of coverage dimensions (spatial and temporal) to generate either RDF subjects or RDF literals for objects, ii) the use of coverage bands in place of relational attributes and iii) the lack of support for the generation of links from foreign keys, not present in coverages.

Figure 5.4 shows an example of a C2RML document used to transform a coverage "*METEO*" of meteorological data to RDF. The RDF graph corresponding to the RDF triples

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

```
@prefix cr: <http://citius.usc.es/c2rml>
@prefix ex: <http://citius.usc.es/example1>
@prefix geo: <http://www.opengis.net/ont/geosparql#>

<TriplesMap1> a cr:TriplesMap;
  cr:coverage [
    cr:coverageName "METEO";
    cr:dimensionTrim [
      cr:dimName "geo";
      cr:trimStart "<EPSG:23029> POINT(4 -4)";
      cr:trimEnd "<EPSG:23029> POINT(5 -3)";
    ];
    cr:dimensionSlice [
      cr:dimName "time";
      cr:slicePoint t0;
    ];
  ];
  cr:subjectMap [
    cr:template
      "http://citius.usc.es/example1/TempCoverage/Loc={geo}";
    cr:class ex:TempCoverage;
  ];
  cr:predicateObjectMap [
    cr:predicate geo:hasLocation;
    cr:objectMap [ cr:dimension "geo"];
  ];
  cr:predicateObjectMap [
    cr:predicate ex:hasTemp;
    cr:objectMap [ cr:band "Temperature"];
  ].
```

Figure 5.4: Example of Coverage to RDF Mapping Language (C2RML) document.

generated for four neighbour coverage cells by this C2RML document is depicted in Figure 5.3(c). As with R2RML, a mapping is defined by an instance of a "*TriplesMap*". The "*cr*:*coverage*" clause is used to define the portion of the input coverage that is going to be mapped to RDF. Thus, spatial and/or temporal restrictions over the coverage domain are specified through a sequence of dimension trimming and/or slicing operations. A trimming operation defined on the spatial dimension of the coverage filters the cells that are contained in a given trimming rectangle. It is noticed that in the current implementation is has been considered that a cell is contained in the rectangle if its centroid is contained. In the example, a "*cr*:*dimensionTrim*" is used to define a spatial trimming operation. The lower left corner point of the trimming rectangle is specified with clause "*cr*:*trimStart*", whereas the top right

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

corner is given by the "*cr*:*trimEnd*" clause. On the other hand, a slice operation defined on the spatial dimension filters the cells that contain a given slicing point. Given that only one cell (at most) is obtained, special treatment has to be done to points that lie in the borders of the coverage cells. Thus, it is assumed that cells are closed on their bottom and left borders, but they are open on their top and right borders. Therefore, if a point lies in either the top or right border of a cell is considered to be out of the cell and the cell would not be retrieved. However, if a point lies on the top (right) border of a cell, it also lies on the bottom (left) border of a neighbour cell (except for cells in the top and right borders of the coverage), and therefore such a neighbour cell would be obtained. Trimming and slicing operations are also defined for the time dimension. In this case, the trimming filters the time instants that are contained in a trimming period, specified by a start and end instant using again the same "*cr*:*trimStart*" and "*cr*:*trimEnd*" clauses. The slicing filters the instants that match a given slice instant. In the example, a slice operation over the time dimension is defined with the "*cr*:*dimensionSlice*" clause, and the slicing instant is specified with the "*cr*:*slicePoint*" clause.

The syntax to generate subjects, predicates and objects for the result triples in C2RML is very similar to the one used by R2RML. In the example, a "*cr*:*subjectMap*" clause uses an IRI template to generate subjects for all the triples generated for each coverage cell that meets the above spatial and temporal restrictions. Finally, clauses "*cr*:*predicateObjectMap*" are used to generate RDF triples to represent the values of coverage bands at each cell.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

EFFICIENT VECTOR AND RASTER GEOSPATIAL DATA ACCESS FROM GEOSPARQL

This chapter, provides a detailed description of the query processing and optimization artefacts used by GeoLD to enable the access to vector and raster datasets through OGC WFS and WCPS standards. In particular, two new operators enable the submission of WFS and WCPS request to OGC web servers, achieving the access to external geospatial datasets in federated architectures and delegating filtering functionality on those servers. A query optimization algorithm transforms conventional SPARQL query plans to query plans that leverage the use of the above new operators. The chapter includes also the results of a performance evaluation of the system, that shows benefits of the implemented approach with respect to state of the art solutions.

The current implementation is based on three main systems, ARQ[1], the SPARQL Processor of Apache Jena, Petascope [2], the OGC standards-based coverage geospatial server of the raster server Rasdaman [11, 15] and Geoserver[2], an open source geospatial server that implements OGC web services on top of different data storage technologies.

[1] https://jena.apache.org/documentation/query/index.html
[2] https://geoserver.org/

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

6.1 Geospatial Data Access Operators

SPARQL BGP operators are always located at the leafs of the query plans and they enable the generation of variable bindings from graphs based on sequences of triple patterns. However, when the graphs are generated using mappings from geospatial vector features or raster coverages to RDF, and they are accessed through WFS and WCPS web services, more specific operators should be used at the leaf nodes of the query plans to achieve better performance. In particular, GeoLD incorporates a *WFS* operator to access geospatial vector features through OGC WFS services, and a *WCPS* operator to access geospatial raster coverages through OGC WCPS services. In general, those operators generate service requests that retrieve filtered data, which is next used to generate the multisets of variable bindings that serve as input to subsequent non-leaf query plan operators.

The *WFS* operator has the following parameters:

- **FeatureType**: The name of a feature type whose vector features may be accessed thought the WFS service.
- **Bindings**: A set of pairs of the form (v, a), where v is a variable name and a is one of the following: i) The keyword URI, to specify that in each variable binding, the value of variable v will be the URI generated for the current vector feature, according to direct mapping from the feature type to RDF. ii) the name of a property of the feature type, to specify that in each variable binding, the value of variable v will a value of that property. iii) if g is the name of a geometric property of the feature type, then a may be either g_URI to specify that the value of v will be a URI of a blank node associated to the vector geometry, or g_WKT to specify that the value of v will the the Well-Known Text representation of the vector geometry.
- **Filter**: A Boolean expression used to filter the vector features of the feature type.

As with any other operator, to get output data form the *WFS* operator, first its $Open()$ method has to be invoked. Next, successive invocations to $next()$ method enable the retrieval of successive variable bindings. The $Close()$ method informs the operator that the process has finished and therefore, any acquired resource may be liberated. The $Open()$ method generates and submits a WFS request based on the operator parameters. In particular, the first parameter is used to specify the feature type of the request, and the third parameter (the Boolean expression) is used to generate a Filter expression for the request. The current version

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

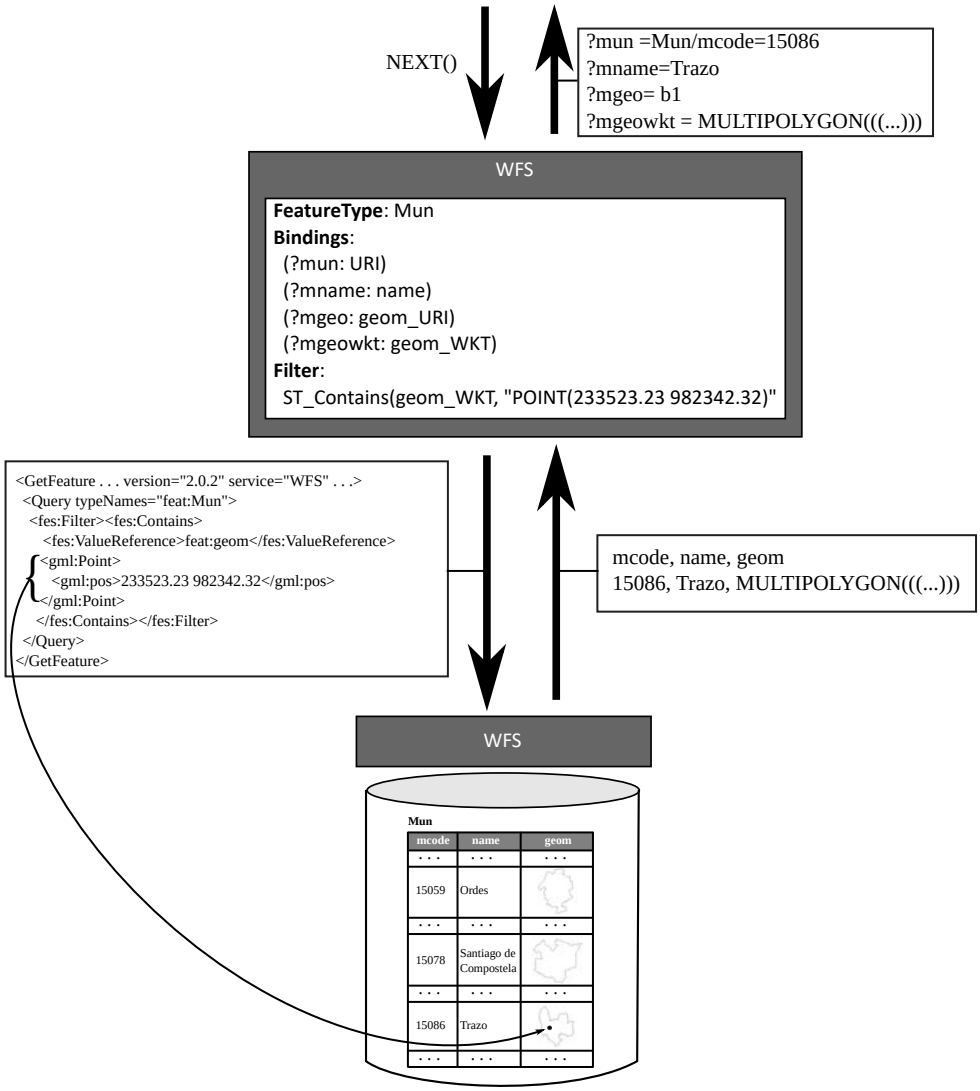


Figure 6.1: Illustration of the GeoLD WFS operator.

of the GeoLD implementation uses *CSV* as the output format for the WFS requests. In each invocation of the $next()$ method, the operator generates a new variable binding using the values of the properties of one of the retrieved vector features, using the configuration given in the second parameter. Figure 6.1 shows a node of a query plan corresponding to a specific instance of the WFS operator. The figure shows the values of the parameters of the operator and illustrates its interface both with its parent operator and with the WFS service. The operator generates a single variable binding generated from a municipality stored in feature type "Mun", that contains a specific query point. The interface with the WFS service shows both the XML encoding of the WFS *GetFeature* request, that includes both the name of the feature type and and filter specification, and the CSV encoding of the *GetFeature* response. The interface with the parent operator shows an invocation of the $next()$ method, that yields a variable binding that assigns values of the vector feature properties to the variables according to the *Bindings* parameter of the operator. Thus, variable $?mun$ is assigned a URI constructed from the vector feature data according to the direct mapping to RDF. Variable $?mgeo$ is assigned a URI $b1$ of a blank node. Variables $?mname$ and $?mgeowkt$ are assigned, respectively, the value of property "name" and the text encoding of geometric property "geom", using the OGC Well-Know Text (WKT) specification.

The access to geospatial raster coverages is provided in GeoLD through a *WCPS* operator. The parameters that configure the behaviour of this operator are basically the same as those of the *WFS* operator. In particular, to create one such operator, the following parameters have to be specified:

- **Coverage**: The name of a coverage accessible through he WCPS service.

- **Bindings**: A set of pairs of the form (v, a), where v is a variable name and a is one of the following: i) The keyword URI, to specify that in each variable binding, the value of variable v will be the URI generated for the current cell of the coverage, according to the available mapping from the coverage structure to RDF. ii) the name of a band of the coverage, to specify that in each variable binding, the value of variable v will the value of that band for the current cell. iii) the name of a dimension of the coverage, i.e., either geo for the spatial dimension or $time$ for the temporal one. As in the case of vector features, the name of the spatial dimension geo must be complemented with either the keyword URI (geo_URI), to use the URI of a blank node, or the keywork WKT (geo_WKT), to use the standard WKT encoding of the cell location.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- **Filter**: A Boolean expression used to filter the cells of the coverage to be retrieved.

As in the case of the *WFS* operator, the *WCPS* operator uses the above parameters to construct and submit a WCPS request that process the referenced coverage to filter the coverage cells accordingly. One variable binding for each retrieved coverage cell is generated by the operator. Contrary to the case of WFS, due to the fact that WCPS is a service designed for processing coverages, and not for filtering coverage cells based on their bands, the implementation of the *WCPS* operator combines a rough filtering and processing done by the WCPS server at the data source with refined filtering done by the *WCPS* operator at the GeoSPARQL engine. In particular, depending on the coverage element involved in the Boolean expression, different strategies are performed.

- If the filter expression contains a geospatial condition, then such condition is used to generate either a trimming or slicing operation over the two geospatial coordinate dimensions of the coverage. Thus, for example, if the spatial dimension of the coverage is tested for intersection with a Point geometry p, the coordinates of p are used to generate integer indexes that are incorporated in a slicing of the coverage over the East and North coordinate dimensions. On the other hand, if the testing involves a geometry with extension (linestring, polygon, etc.), then the Minimum Bounding Rectangle (MBR) of the geometry is used to generate the ranges over the East and North coordinate dimensions incorporated in trimming operations. In this case, the WCPS response would retrieve some coverage cells for which the spatial condition might evaluate to false, and therefore, a refinement of the filtering is required to be done by the *WCPS* operator at the GeoSPARQL engine, which test if each retrieved cell intersects the query geometry.

- If the filter expression contains a condition over the temporal dimension of the coverage, such a condition is used to generate either a trimming or slicing operation over the temporal coordinate dimension of the coverage. As in the case of the geospatial condition, a refinement of the filtering might be required at the GeoSPARQL engine, when time instants contained in the range of the generated trimming operation must be discarded.

- If the expression contains conditions that involve coverage bands, then an strategy different from the above one has to be followed, since WCPS does not support filtering over coverage bands. Notice that in this case, the result of the operator would not be,

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

in the general case, isomorphic to valid coverage. In this case, the rough filtering done at the data source by the WCPS server consists on processing each coverage cell using a SWITCH-CASE expression, retrieving the band value when the condition holds and retrieving a $not - a - number$ value otherwise. Cells with such $not - a - number$ values are discarded by the *WCPS* operator at the GeoSPARQL engine in a refinement filtering step.

Figure 6.2 depicts a *WCPS* leaf node of a query plan, showing the configuration parameters of the operator and illustrating the interface with the WCPS service and with its parent node. The WCPS operator is used to access the cells of a coverage called "Meteo", that records the spatial and temporal variation of meteorological variables. The bindings parameter is used to specify the coverage dimensions and bands used to supply values for the the variables. In particular, the URI of each cell will be retrieved in variable $?s$, the identifier of each cell geometry (RDF blank node) is obtained in variable $sloc$, the WKT encoding of the cell geometry is assigned to variable $?slocwkt$ and variables $?time$ and $?temp$ are used to retrieve, respectively, the time instant obtained from the temporal dimension and the temperature value obtained from the relevant coverage band. The filter condition restricts the output of the operator to coverage cells whose temporal dimension lies inside a specific time range. The above parameters are used to construct the WCPS request shown in the figure. It is noticed that the WCPS request obtains from coverage "Meteo", values of band "Temperature". A trimming operation is performed over the temporal dimension and the results are encoded in CSV format. Such CSV is processed by the *WCPS* operator at the GeoSPARQL engine to generate the variable bindings that are illustrated at the top of the figure. One binding is generated for each retrieved coverage cell.

It is noticed that the CSV obtained from the WCPS service represents an array of values of the requested coverage band, and values of spatial and temporal dimensions associated with each value are omitted. This is possible because of the regular sampling assumed in the definition of spatial and temporal dimensions of geospatial raster coverages. To be able to generate automatically the geospatial and temporal coordinates of the relevant dimensions of each cell, a specific ordering of the data values in the result array must be assumed. In the current implementation, the geospatial and temporal dimensions used in the same order they are defined in the data source coverage or in the C2RML mapping if it exists. Within the geospatial dimension, the ordering of horizontal and vertical subdimensions is the one illustrated in Figure 6.3, which is used in many raster data formats, and used also by the

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

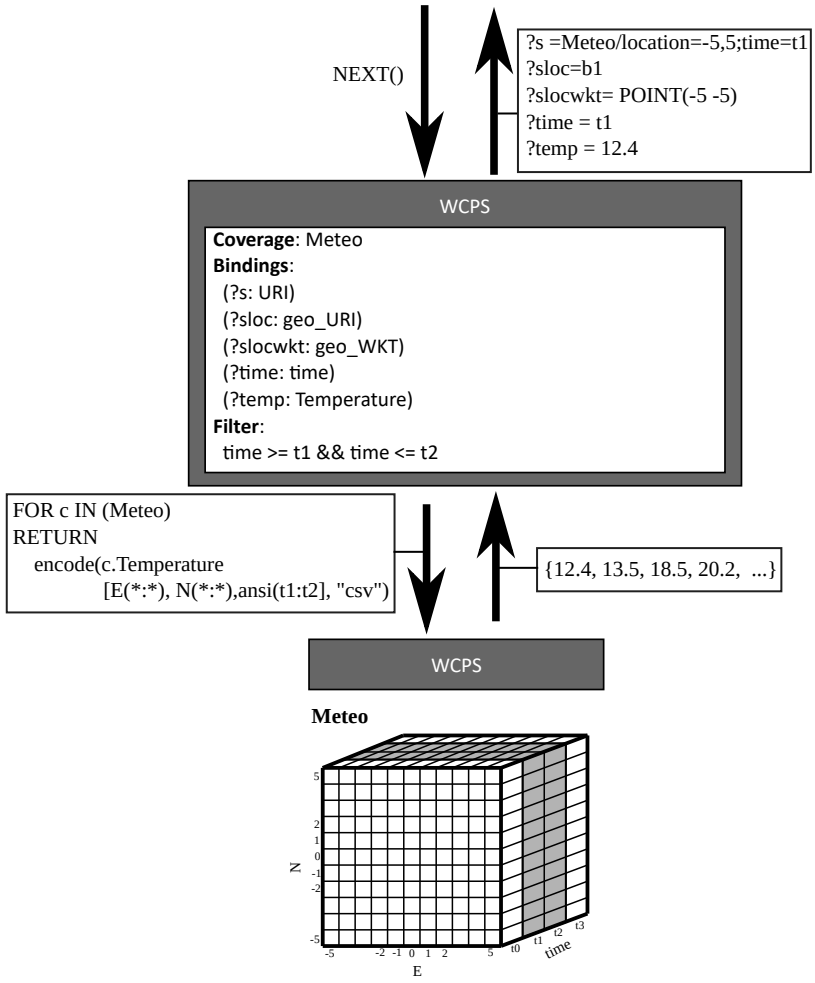


Figure 6.2: Illustration of the GeoLD WCPS operator.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

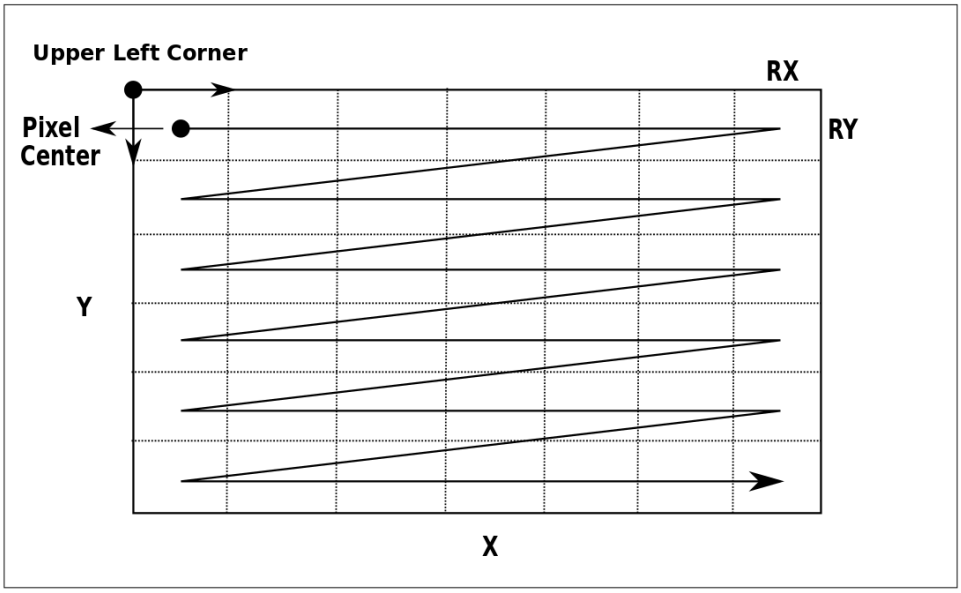


Figure 6.3: Ordering of coverage cells according to the geospatial horizontal and vertical subdimensions.

underlying WCPS implementation of Petascope [2].

6.2 Query Optimization

The use of the new operations in query plans requires the incorporation of new query optimization strategies, that decide when generic BGP and Filter operations may be replaced by these new specialized WFS and WCPS data access operators. In the current GeoLD implementation, conventional general purpose query optimization strategies are first applied and next a new specialized query optimization algorithm replaces parts to the query tree by *WFS* and *WCPS* operators, whenever possible. An example of a general purpose query optimization strategy has already been shown in Figure 3.2, where part of the Filter is moved down in the query plan tree. In general, moving Filter operations as close as possible of the tree leaf nodes is a good heuristic to achieve more efficient query plans. This is also the idea behind the incorporation of the new operators, that move part of the filtering to the data sources.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

The new query optimization algorithm is divided in two phases. In a first phase, a tagging algorithm classifies the nodes of the query plan tree into different categories. BGP and Filter nodes whose vocabulary refer to a single data source element, i.e., feature type or coverage, are tagged with the text "wfs" or "wcps", followed by the name of the feature type or coverage, respectively. All the remainder nodes are tagged with the general tag "sparql". In the second phase, the tagged query plan tree is processed to replace consecutive Filter and BGP nodes tagged with the same data source element by a node of type *WFS* or *WCPS* that provides access to the required data.

Algorithm 1 Data source tagging algorithm

```
1:  function DataSourceTagger(P)
2:      if isBGP(P) then return BGPTagger(P)
3:      else if isFilter(P) then
4:          TaggedChild = DataSourceTagger(child(P))
5:          setChild(P, TaggedChild)
6:          setTag(P, tag(TaggedChild))
7:          return P
8:      else
9:          C = children(P)
10:         for all c ∈ C do
11:             append(pchildren, DataSourceTagger(c))
12:         end for
13:         setChildren(P, pchildren)
14:         setTag(P, "sparql")
15:         return P
16:     end if
17: end function
18: function BGPTagger(N)
19:     classifyTriples(N, dbgps, sbgp)
20:     return createJoin(dbgps, sbgp)
21: end function
```

Algorithm 1 shows the pseudocode of the algorithm of the first phase. The notation used in the pseudocode is introduced in Table 6.1. Function *DataSourceTagger* processes a query plan tree, whose root node is *P*, in dept-first order, to yield a reference to the root node of the relevant tagged query plan tree. In each level of the tree, the tag of a node is decided based on the tags of its children. A specific algorithm is used to tag BGP leaf nodes, whose pseudocode is given in function *BGPTagger*. In line 2 of function *DataSourceTagger*, if the current

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Table 6.1: Notation used in the data source tagging algorithm pseudocode

Notation	Description
$DataSourceTagger(P)$	Returns a tagged version of query plan tree P.
$BGPTagger(P)$	P must be a BGP node. It creates a tagged query tree where triples related to a single data source are isolated into a single BGP, which is tagged with the name of the data source. All the single BGP are combined with JOIN operations.
P	Query plan tree.
N	BGP leaf node.
$isbgp(P)$	Returns true if P is a BGP node.
$isFilter(P)$	Returns true if P is a Filter node.
$child(P)$	It returns the single child node of node P.
$setChild(P1, P2)$	It sets node $P2$ as the single child of node $P1$.
$setTag(P, t)$	It sets text t as the tag for node P.
$tag(P)$	It obtains the tag of node P.
$children(P)$	It returns the collection of child nodes of P.
$append(C, e)$	It appends element e to collection C.
$setChildren(P, C)$	It sets the collection of nodes C as the children of node P.
$classifyTriples(N, dbgps, sbgp)$	It processes a BGP node N to classify its triple patterns. All the triple patterns related to a single data source d are combined in a single BGP node, which is tagged with the name of the data source. All those BGP nodes are appended to the collection $dbgps$. All the triple patterns not related to any OGC data source are combined in BGP node $sbgp$, which is tagged with "sparql".
$createJoin(dbgps, sbgp)$	It creates a tree that joins the results of all the BGP nodes in collection $dbgps$ with the BGP node $sbgp$, using SPARQL JOIN operations.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

node being processed is a BGP node, then function $BGPTagger$ is used to generate a relevant tagged subtree for that BGP node. In lines $3-7$, if the current node is a Filter node, then the node is tagged with the tag already assigned to its child node. All the other types of nodes are tagged with "sparql" and processed in lines $9-15$.

In function $BGPTagger$, the triple patters of leaf BGP nodes are first classified into disjoint subsets (line 19) and next combined using JOIN operations (line 20). The classification of triple patterns done by function $classifyTriples$ works as follows. All triple patterns related to a single geospatial data source (either fetaure type or coverage) are isolated into a single BGP node and appended to the collection $dbgps$. Each such BGP node is tagged with the name of its data source, i.e., with the concatenation of the text "wfs" or "wcps" with the name of the feature type or coverage. All the remainder triple patterns are collected into a singe BGP node $sbgp$, which is tagged with "sparql". After the classification of triple patterns, function $createJoin$ is used to combine all the generated BGPs into a single query plan, using SPARQL JOIN nodes that are tagged with "sparql".

A first example of the result of the application of the above tagging algorithm is given in Figure 6.4, which is the tagged version of the query plan of Figure 3.2. The BGP node at the bottom left side of the query plan is tagged with "wfs Municipality", as all its triple patterns refer to the same $Municipality$ feature type. Similarly, the BGP node at the bottom right side of the query plan is tagged with "wcps Meteo", as all its triple patterns are related to coverage $Meteo$. Each Filter node in the query plan tree is tagged with the same tag of its child. Notice for example the Filter node just on top of the BGP at bottom right side of the query plan, which is tagged with "wcps Meteo". All the remainder nodes are tagged with "sparql". It is also noticed in this example that the three BGP leaf nodes were not split by the $BGPTagger$ function into various BGP nodes, since their triple patterns were homogeneous, in the sense that either they were all related to a single geospatial data source ($Municipality$ feature type or $Meteo$ coverage) or they were all not related to any geospatial data source.

An example of the application of the tagging algorithm that requires the classification of the triples of a BGP is shown in Figure 6.5. The query plan obtains the names of the municipalities whose capital towns are within 30km from the capital of the municipality of "Trazo". The data sources and the mapping to RDF considered in this example are those of Figure 5.2. The BGP node at the bottom left of the query plan obtains the data from the municipalities, which are filtered by name in the Filter node on top of it. Those two nodes are related only to the $Municipality$ feature type, therefore they are tagged with "wfs Municipality". The BGP at

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

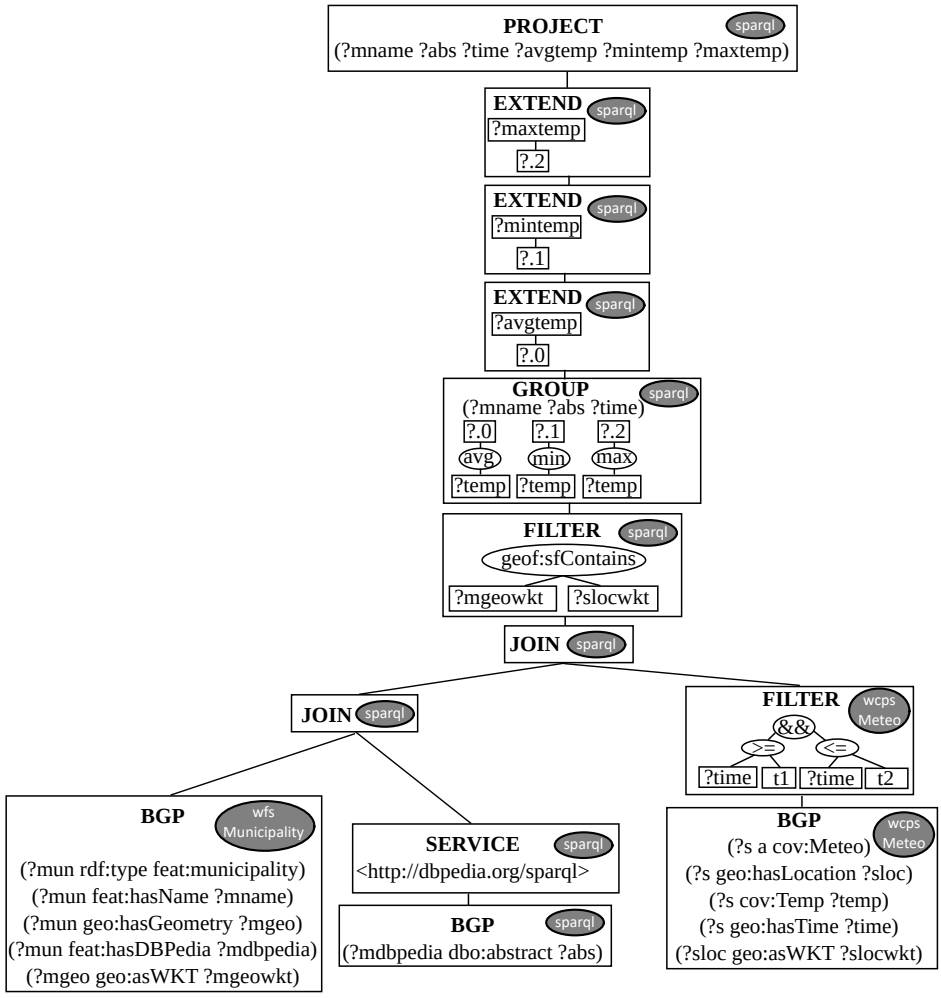


Figure 6.4: Result of the tagging of the query plan of Figure 3.2

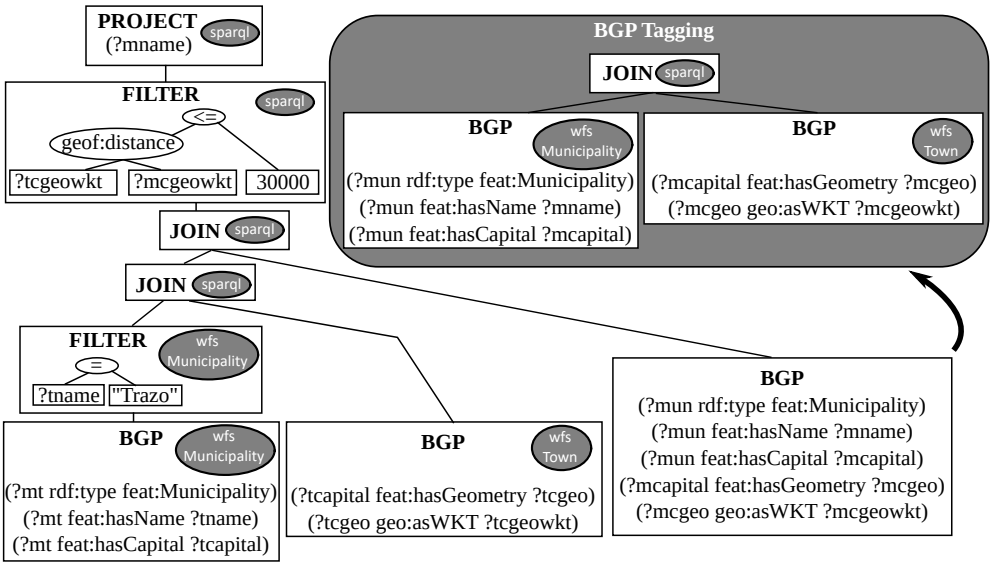


Figure 6.5: Illustration of a tagging phase that requires the classification of triple patterns of a BGP.

the bottom center part of the plan obtains data from towns, therefore it is tagged with "wfs Town". Finally, the BGP at the bottom right side of the query plan has triple patterns related to both feature types, $Municipality$ and $Town$. Therefore, those triple patterns are classified by the $BGPTagger$ function to create the two BGPs shown in the top right part of the figure, which are tagged with the name of the relevant feature type, and combined with a JOIN node. All the remainder nodes of the query plan are tagged with "sparql".

In the second phase of the query optimization process, the tagged query plan produced in the first phase is transformed to a new query plan where data source operators (*WFS* and *WCPS* operator) replace either BGP operators or sequences of Filter and BGP operators. Algorithm 2 shows the pseudocode of this query plan generation algorithm. In line 2, if the current node is a BGP node and it is tagged with the name of a data source element (feature type or coverage), i.e., it is not tagged with "sparql", then the BGP node has to be replaced by a data source operator node, i.e., a *WFS* if the data source element is a feature type or a *WCPS* if the data source element is a coverage. The generation of the data source operator is done by the function $DSOperator$, which must extract the data source name and build the bindings parameter using the tag and triple patterns of the BGP node. The $DSOperator$ must also be

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Algorithm 2 Query plan generation Algorithm

```
1:  function QUERYPLANGENERETOR(P)
2:      if isBGP(P) ∧ tag(P)! = "sparql" then return DSOperator(P)
3:      else if isFilter(P) ∧ tag(P)! = "sparql" ∧ tag(child(P)) = tag(P) then
4:          return DSOperator(P)
5:      else
6:          C = children(P)
7:          for all c ∈ C do
8:              append(pchildren, QueryPlanGeneretor(c))
9:          end for
10:         setChildren(P, pchildren)
11:         return P
12:     end if
13: end function
```

applied if the current node is a Filter just on top of a BGP and both are tagged with the name of the same data source. This is expressed in the condition of line 3. Now, the $DSOperator$ function must also insert a Filter parameter in the data source operator. The remainder lines of the algorithm are used to process all the other cases of current nodes, achieving a depth-first traversal of the query plan tree. In the case of transformations of BGP elements tagged with names of geospatial coverages, independently of the existence of a Filter operation on top of them, and due to the potential use of C2RML to define the mapping from the coverage structure to RDF, it is possible that a Filter parameter has to be included in the result *WFS* operator. This situation happens when a trimming or slicing is used in the C2RML document to define geospatial or temporal dimensions. Those trimming or slicing operations must be included or combined with the existing filters to create a combined Filter parameter of the *WCPS* operator.

As an example of the application of the above algorithm, Figure 6.6 shows the query plan generated for the tagged plan of Figure 6.4. It is noticed that the BGP node at the bottom left part of the tagged query plan is replaced by a *WFS* operator in the optimized plan. The feature type of the *WFS* operator is obtained from the tag of the input BGP node. The bindings parameter of the new *WFS* operator is generated using the names of the variables and the URIs of the predicates of the triple patterns of the input BGP node. To achive this, it is needed to combine also the information provided by the mapping from the feature type to RDF used. Thus, for example, the pair $(?mname : name)$ of the bindings parameter is obtained from the triple pattern $<?mun\ feat{:}hasName\ ?mname>$, considering that the mapping to RDF

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

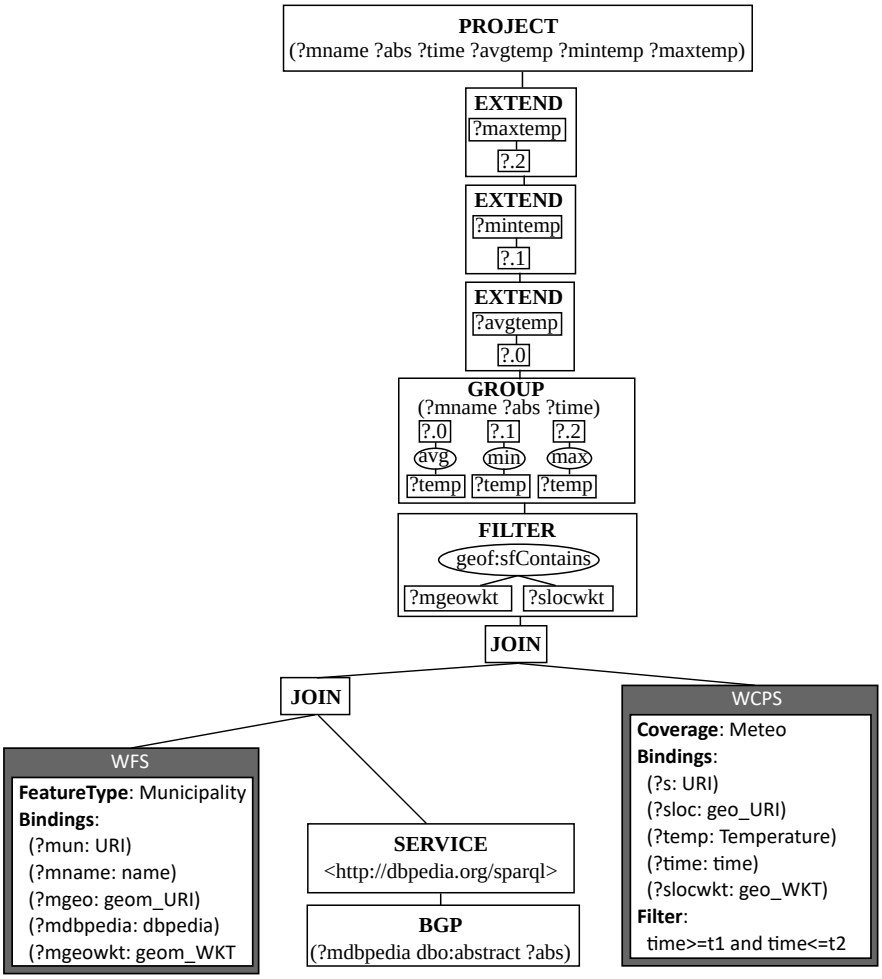


Figure 6.6: Optimized query plan generated for the tagged query plan of Figure 6.4

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

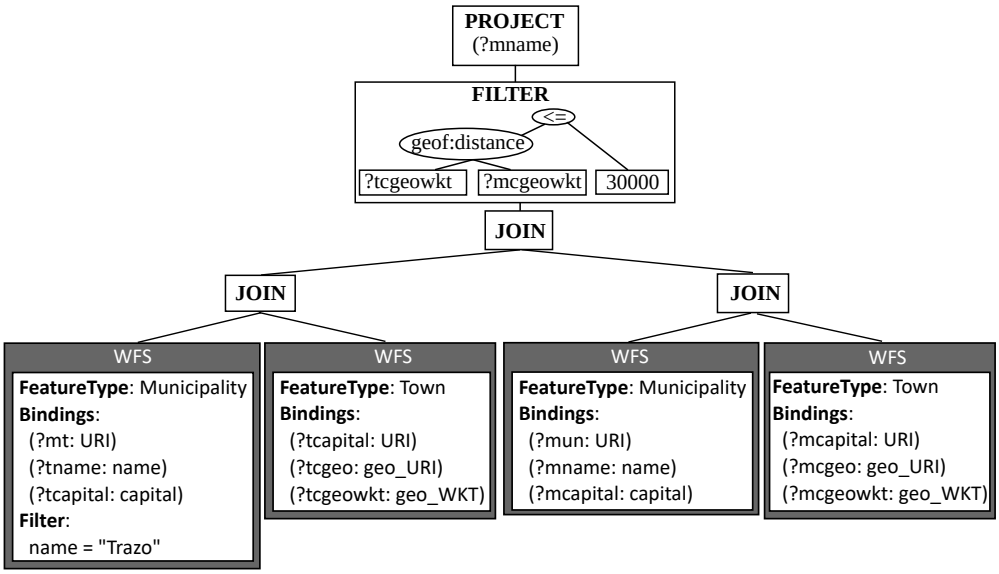


Figure 6.7: Optimized query plan generated for the tagged query plan of Figure 6.5

defines that predicate $feat{:}hasName$ is the result in RDF of the mapping of property *name*. In this case, the *WFS* does not have a Filter parameter, since it has been generated from just a BGP operator without Filter operator. Contrary to this, the sequence of Filter and BGP operators of the bottom right part of the tagged plan are replaced by a *WCPS* operator that includes a Filter parameter.

Another example of the generation of an optimized query plan from a tagged plan is given in Figure 6.7, which shows the result of the application of the second phase of the optimization to the tagged plan of Figure 6.5. Now, it is noticed how the sequence of Filter and BGP operators of the bottom left part of the tagged plan, both tagged with "wfs Municipality", are replaced by an instance of the *WFS* operator, which includes a Filter parameter. All the remainder BGP operators of the tagged plan, all of them tagged with names of feature types, are transformed to instances of the *WFS* operator, always without Filter parameter.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

6.3 Performance Evaluation

The main contribution of GeoLD with respect to state of the art GeoSPARQL implementations is its efficient support for the querying of geospatial raster coverages, accessed through OGC WCPS interfaces. To demonstrate the efficiency of the approach followed by GeoLD, with respect to other approaches of the state of the art, various experiments were undertaken, combining different datasets with different query types.

6.3.1 Evaluation Setup

The datasets, query types and state of the art approaches whose performance is compared with that of GeoLD are described below. Two different types of datasets were used to test query performance with increasing database sizes, testing the system performance scaling both the temporal and geospatial dimensions of the input coverages.

Spatio-temporal Meteorological Coverages : To test the performance of the approaches when the database grows in the time dimension, a spatio-temporal coverage was used, whose bands represent historical numeric weather predictions generated by the meteorological agency of the Spaninsh region of Galicia, MeteoGalicia [3]. The size of the geospatial dimension was that of the original data, i.e., 117×142 cells, and the size of the time dimension was ranging from 10 to 50 time instants. Thus, the overall size ranged from $664K$ to $3.3M$ data elements (spatio-temporal cells).

Spatial DEM Coverages : To test the performance of the approaches when the geospatial dimension of the coverages increases, a spatial coverage representing the Digital Elevation Model (DEM) of Galicia was used, with an original geospatial resolution of 100 meters. Datasets of different sizes were build decreasing the resolution from 100 meters to 800 meters, yielding coverages of sizes that ranged from $72k$ data elements (800 meters) to $4.6M$ data elements (100 meters).

In general, the following 5 types of queries were used to test the query response time of each of the evaluated approaches.

Space Point Query : The query filter for this type of query is a geospatial point. For input spatial coverages the result is a record of band values, and for input spatio-temporal

[3] https://www.meteogalicia.gal/modelos/

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

coverages the result is a time series of records of band values. In both cases, the output restrict to values of cells whose geometry intersects with the input geospatial point, i.e., the query performs a slicing of the input coverage over its spatial dimension.

Time Point Query : The query filter for this type of query is a time instant, and therefore it cannot be applied to coverages without temporal dimension. The result is a spatial coverage that is valid at the query time instant, i.e., the query performs a slicing of the input coverage over its temporal dimension.

Space Range Query : The query filter for this type of query is a geospatial rectangle. The type of the output coverage matches the one of the input coverage. In both cases, the output restrict the cells of the output coverage to those whose geometry lies inside the input query rectangle, i.e., the query performs a trimming operation of the input coverage over its spatial dimension.

Time Range Query : The query filter for this type of query is a time period, and therefore it cannot be applied to coverages without temporal dimension. The result is a spatio-temporal coverage, whose temporal dimension restricts to the subset of the input temporal dimension that lies inside the input time period. In other words, the query performs a trimming of the input coverage over its temporal dimension.

Field Range Query : The query filter is a range of values of one of the coverage bands. The result is a set of triples of the form (s, t, v), such that it exists a cell in the input coverage, whose geometry is s, its valid time is t, its value of the query band is v, and v lies inside the input query range.

Based on the state of the art already described in Chapter 2, the following SPARQL implementations were selected to be compared with the current version of the GeoLD implementation.

STRABON [67] : A GeoSPARQL implementation designed for geospatial vector features, whose data storage technology is based on the PostgreSQL DBMS, extended with PostGIS to support the geometric data types. Raster coverage cells have to be stored in STRABON as vector features of point type. All the types of queries are supported by STRABON.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

SciSPARQL [5, 6] : A SPARQL extension to support scientific array data. Raster coverages are supported by array data elements. The data storage technology is based on textual RDF encodings, which yields response times around two orders of magnitude worse than those of GeoLD.

GeoSPARQL+ [55] : A GeoSPARQL proof of concept implementation that supports both geospatial vector features and raster coverages. Raster coverages are supported by an array data type, which includes both binary and textual encodings. Unfortunately, of the five types of queries described above, only the time point queries could be implemented with the current version of the system, due to the lack of raster data type functions to extract raster cells from raster data type elements.

All the software was installed in a 32 bit virtual machine with Ubuntu 16.04, since SciSPARQL was only available for 32 bit architectures. The hardware specification used for the virtual machine was the following: 4 processors Intel core I7 3.40 GHz and 8 GB of RAM.

6.3.2 Performance testing scaling the temporal dimension

In real application scenarios that deal with historical data, the size of the geospatial dimension of the coverages uses to be static, whereas the temporal dimension increases as new data elements are inserted. This is the case for example of applications that deal with meteorological observations and model results. For these applications, testing how the system performance evolves as new spatial coverages are inserted for new time instants is of great importance. To simulate this scenario, a set of spatio-temporal coverages was generated using the spatio-temporal meteorological coverages described in the previous subsection. In particular, versions of the coverage were created with temporal dimension ranging from 10 to 50 time instants and all the query types were evaluated. All the queries were evaluated with a cold cache system status, to avoid the undesirable effects in the response time of the buffer cache of the PostgreSQL DBMS used by STRABON, which is not present in none of the other solutions.

Figure 6.8(a) shows the SPARQL expression of a time point query, the expression used by the WCPS operator of GeoLD and the response times obtained for each solution with database sizes of increasing number of time instants. The response time is constant with respect to the database size in GeoLD, due to the direct access using the index in the temporal dimension supported by the array data storage technology of Rasdaman. The response time is also constant in the GeoSPARQL+ proof of concept implementation, whose binary raster

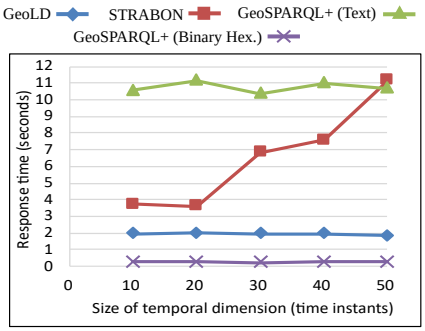
USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
SELECT  ?temp
WHERE  {?s a cov:Meteo; cov:Temp ?temp;cov:hasTime ?time.
          FILTER (?time=t)}
```

WCPS

```
FOR c IN (Meteo)
RETURN encode(c.temperature[E(*:*),N(*:*), ansi(t)],"csv")
```



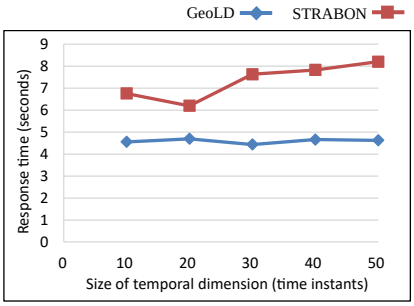
(a) Point Query

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
SELECT  ?temp
WHERE {?s a cov:Meteo; cov:Temp ?temp; cov:hasTime ?time.
          FILTER (?time>=ts  && ?time<=te )}
```

WCPS

```
FOR c IN (Meteo)
RETURN  encode(c.temperature[E(*:*),N(*:*),ansi(ts:te)],"csv")
```



(b) Range Query

Figure 6.8: Response times for temporal queries over spatio-temporal coverages of growing temporal dimension

tile encoding offers clearly better performance than the text encoding. This shows the impact of the use of compact RDF representations, however that is out of the scope of the current Thesis. Regarding STRABON, the relational DBMS-based data storage technology layer provides indexing structures for both 1D temporal dimensions and 2D spatial dimensions. As it is shown in the figure, the performance degenerates with the increase of the database size. Finally, the response times of SciSPARQL are two orders of magnitude worse than those of GeoLD, and that is why they are not shown in the figure.

The expressions and results for time range queries are shown in Figure 6.8(b). Again, the response times of GeoLD are constant with respect to the database size, which is not the case of STRABON. This type of query could not be implemented in GeoSPARQL+, and regarding SciSPARQL, again the response times are two orders around two orders of magnitude worse than those of GeoLD.

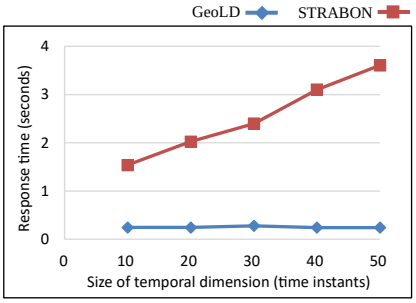
USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
PREFIX geof:<http://www.opengis.net/def/function/geosparql/>
SELECT  ?temp
WHERE {?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp.
   ?sloc geo:asWKT ?slocwkt. FILTER (geof:sfEquals(?slocwkt, p))}
```

WCPS

```
FOR c IN (Meteo)
RETURN encode(c.temperature[E(px),N(py), ansi(*:*)],"csv")
```



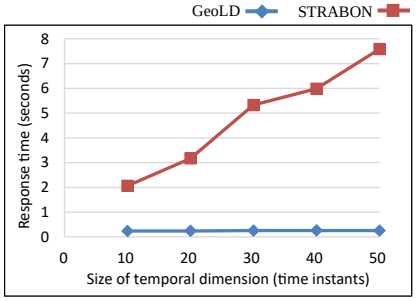
(a) Point Query

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
PREFIX geof:<http://www.opengis.net/def/function/geosparql/>
SELECT  ?temp
WHERE {?s a cov:Meteo; cov:hasLocation ?sloc; cov:Temp ?temp.
 ?sloc geo:asWKT ?slocwkt. FILTER (geof:sfContains(p, ?slocwkt))}
```

WCPS

```
FOR c IN (Meteo)
RETURN encode(c.temperature[E(minpx:maxpx),
                            N(minpy:maxpy), ansi(*:*)],"csv")
```



(b) Range Query

Figure 6.9: Response times for spatial queries over spatio-temporal coverages of growing temporal dimension

The query expressions and response times of spatial point and range queries are shown in Figures 6.9(a) and 6.9(b), respectively. In both cases, GeoLD shows a response time constant with respect to the increase of the size of the temporal dimension, which is not the case of STRABON. Again, the response times of SciSPARQL are two orders of magnitude worse, and they are not shown in the figure. Finally, the spatial queries could not be implemented in GeoSPARQL+ due to the lack of appropriate implementation of raster data type functions to extract cell values from raster tiles. However, a query that retrieves the time instants corresponding to raster tiles that intersect a given query point showed response times of around 4 seconds, which is slower than GeoLD and STRABON, despite of not having to access the raster tiles to obtain the cell values. This is an indication of the lack of focus on efficiency of the current data storage technology used by GeoSPARQL+.

Figure 6.10 shows the generic GeoSPARQL expression of a field range query, together

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
SELECT  ?temp
WHERE {?s a cov:Meteo; cov:Temp ?temp.
          FILTER (?temp>=t1 && ?temp <= t2)}
```

WCPS

```
FOR c IN (Meteo)
RETURN encode(SWITCH
                    CASE c.temperature[E(*:*),N(*:*), ansi(*:*)]>= t1
                      AND c.temperature[E(*:*),N(*:*),ansi(*:*)]<=t2
                              RETURN  (c.temperature)
                    DEFAULT RETURN nan,"csv")
```

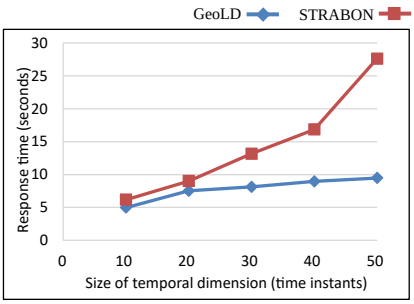


Figure 6.10: Response times of field range queries over spatio-temporal coverages of growing temporal dimension

with the WCPS expression used by the WCPS operator of GeoLD and the response times achieved by GeoLD and STRABON for different sizes of the temporal dimension. Despite of having to bring all the data elements (coverage cells) from the WCPS service to the SPARQL engine, the GeoLD shows better performance than STRABON. None of the two systems have any kind of access method by the coverage bands, therefore, the response times increase with the size of the database. Again, as in the other types of queries, SciSPARQL shows response times two orders of magnitude worse than GeoLD and GeoSPARQL+ does not support the implementation.

Based on the above results, it is clear that GeoLD outperforms the state of the art GeoSPARQL implementations in most cases. Only GeoSPARQL+ with its binary raster encoding shows slightly better performance for time point queries, due to the fact that: i) Considered database sizes are not really big, ii) raster tiles do not have to be accessed in this type of query and iii) retrieving whole raster tiles in compact format is fast. This is an evidence of the importance of the research in compact RDF encodings to achieve efficient SPARQL implementations, but this research objective is out of the scope of the present Thesis.

6.3.3 Performance testing scaling the spatial dimension

To complete the evaluation of GeoLD, a second experiment was done to test the impact in the response time of the increase of the spatial dimension of the input coverage. Thus, space range queries with different sizes of the query rectangle were evaluated over spatial coverages

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GeoSPARQL

```
PREFIX cov:<http://www.mydomain.com/coverages/>
SELECT  ?elev
WHERE {?s a cov:Topo; cov:Elev ?elev;
          cov:hasLocation ?sloc. ?sloc geo:asWKT ?slocwkt.
          FILTER (geof:sfIntersects(slocwkt, r))}
```

WCPS

```
FOR c IN (Topo)
RETURN encode(c.elev
                    [E(minpx:maxpx),
                     N(minpy:maxpy), ansi(*:*)],"csv")
```

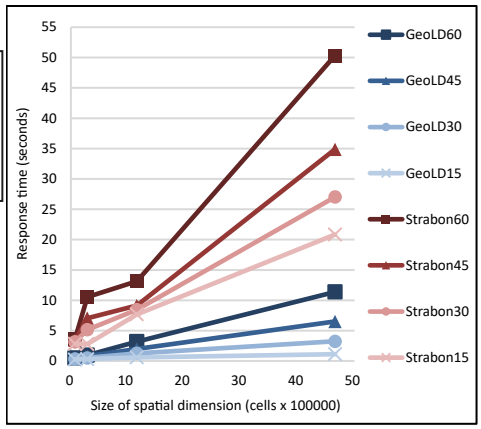


Figure 6.11: Response times for spatial range queries over spatial coverages of growing spatial dimension

of increasing spatial dimension size. In particular, spatial coverages of resolutions of 800, 400, 200 meters were used, with sizes ranging from $72k$ cells to $4.6M$ cells. Query rectangles consisted of squares of increasing size, ranging from $15km$ to $60km$ of side length.

Due to the lack of support for this types of queries provided by GeoSPARQL+ and the low performance shown by SciSPARQL, only the results achieved by GeoLD and STRABON for each query size and for each database size are shown in Figure 6.11. The figure shows also the GeoSPARQL expression of the query, where r represents the query rectangle, and the WCPS expression used the WCPS operator of GeoLD, where $minpx$, $maxpx$, $minpy$ and $maxpy$ represent respectively the minimum and maximum horizontal and vertical coordinates of the query rectangle.

Regarding the response times, as it is shown in the figure, GeoLD outperforms STRABON in all cases, due to the specialized raster data storage and access technology provided by rasdaman.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 7

FEDERATED AGGREGATION OF GEOSPATIAL RASTER COVERAGES OVER VECTOR FEATURES

The query processing and optimization solutions provided by GeoLD improve the efficiency of queries that require the filtering over vector feature types and raster coverages, which are available through standard OGC web services. However, queries that require the combination of vector and raster data, like the one of Figure 3.1, are evaluated using query plans, like the one in Figure 6.6, that are far from being efficient. This is mainly due to two reasons: i) the JOIN operations provided by the SPARQL engine do not exploit the existence of filtering capabilities in the WFS and WCPS operators and ii) the query plan does not make use of the aggregate computation capabilities of the WCPS service. This section elaborates more on this problem, providing new query processing solutions that might be incorporated through relevant query optimization strategies in future versions of GeoLD. The performance of the new solutions are evaluated to show the advantages with respect to the current version of GeoLD.

7.1 Problem Description and GeoLD Solution

The efficient aggregation of raster coverage cells over vector feature geometries requires the efficient implementation of geospatial join operations between these two types of data

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

sources. Furthermore, if a federated architecture is considered, as it is the case of this Thesis, the existence of a communication network between the data sources and the query engine increases the need to push as much processing as possible close to the data storage. To illustrate this, let us consider the query example of Figure 7.1, which is simpler than the running example used in previous chapters, since it does not consider a temporal dimension in the input coverage. In the example, the cells of a coverage of elevation data that lie inside the same municipality (vector feature) are grouped and next the average of the elevation in each municipality is computed. The query and the input and output datasets are illustrated in the figure, which also shows the bottom part of the query plan generated by the current GeoLD implementation.

A quick analysis of the query plan shows that it provides a straightforward approach for the combination of vector and raster data, whose performance if far from being acceptable. In particular, let us first focus on the SPARQL JOIN operator that combines the multisets of variable bindings generated by the WFS and WCPS operators. This general purpose SPARQL JOIN operator does not make any assumption on the underlying datasets and, given that there are not variables in common between the bindings of the two input datasets, its algorithm must generate the Cartesian Product of the two inputs. Such a Cartesian Product is a very large set of variable bindings (it combines each municipality with each cell of the Topo coverage) that is next restricted with the Filter operator, that test predicate $geof{:}sfContains$ between each municipality and raster cell. Clearly, the query plan does not take advantage of the filtering capabilities of the WFS and WCPS operators and beyond that, it does not take advantage of the aggregation capabilities of the WCPS service that is behind the WCPS operator.

A first optimization of the above query plan would come form the incorporation of a new algorithm for the JOIN operator that includes also the spatial filter on top of it (Spatial Join operator) and enables the leveraging of the geospatial filtering capabilities available in the WCPS operator. Broadly speaking, for each vector feature obtained from the WFS operator, the Filter parameter of the WCPS operator is configured to obtain only the cells that are contained in its geometry, avoiding this way the evaluation of the $geof{:}sfContains$ spatial predicate in the SPARQL engine between all the combination of each vector feature with each raster cell. This new join operator and its implementation is described in Section 7.2.

A second optimization is related to a more sophisticated approach, that requires the implementation of a new Group operation that works directly on top of WFS and WCPS operators, whose functionality includes both the spatial join between vector features and raster

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

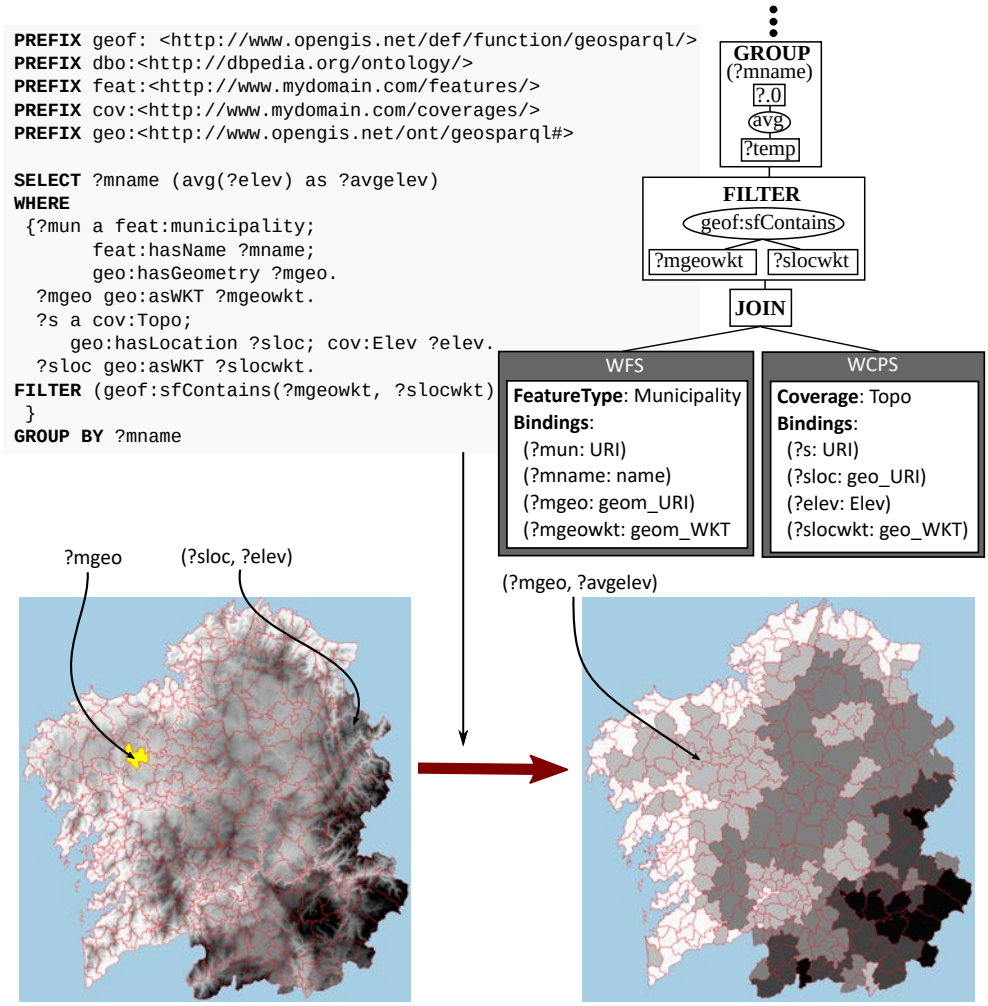


Figure 7.1: Example of aggregation of raster coverages over vector features.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

cells and the group and aggregate of raster cells over vector features. Beyond the efficient implementation of the spatial join of vector features and raster cells, a key feature of this new group operation is its ability to delegate the computation of aggregates to the underlying WCPS service, moving this way an important part of the processing to the data source, which is in general an heuristic that results in more efficient implementations. Broadly speaking, the affected geospatial area is subdivided into tiles that are either: i) completely contained in some vector feature or ii) small enough to be worth to proceed with further subdivision. The above tiles are processed through evaluations of the WCPS operator, whose parameters are configured to either retrieve coverage cells or process aggregates, depending on the type of the tile. A detailed description of this new complex group operator and of its implementation is given in Section 7.3.

7.2 Vector-Raster Indexed Spatial Join

The new Vector-Raster Indexed Spatial Join operator (VRJOIN) has three input parameters.

- A reference to the input WFS operator, that provides access to the vector geospatial feature dataset.
- The name of the variable generated by the WFS operator that records the WKT representations of the vector feature geometries.
- A reference to the input WCPS operator, that provides access to the cells of the raster geospatial coverage.

The result of the evaluation of this new operator is a multiset of variable bindings that combines the bindings generated by the WFS operator with the bindings generated by the WCPS operator, restricting the output to the combinations where the geometry of the vector feature contains the geometry of the raster cell. In other words, the result matches the one produced by a first application of the JOIN operator to the input WFS and WCPS results and a subsequent application of the FILTER operation, using the $geof{:}sfContains$ predicate.

Figure 7.2(a) shows an optimized version of the query plan shown in Figure 7.1, that makes use of this new VRJOIN operator in place of the JOIN and FILTER operations. It is noticed that the VRJOIN operator has been configured to use variable $?mgeowkt$ as the container of the vector feature geometries. The name of the variable containing the raster coverage cell

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

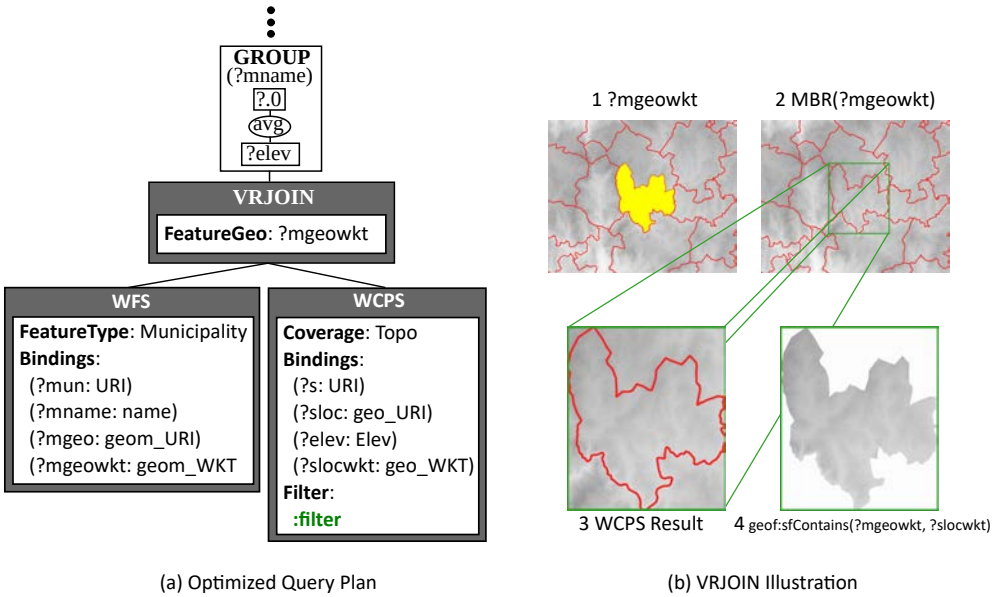


Figure 7.2: Implementation of the new VRJOIN operator.

geometries is not really needed for the implementation of the operation. This is due to the fact that the filter of the WCPS operator is expressed directly using the coverage dimension "geo". It may also be observed that the WCPS configuration includes the name of the coverage and the bindings (mapping of coverage attributes and dimensions to variable names). However the filter will have to be generated using the geometry of each municipality in each WCPS evaluation.

A pseudocode of the algorithm of the VRJOIN operator is provided in Algorithm 3. The input parameters of the operator are shown in Line 1. Parameters WFS and $WCPS$ represent references to the WFS and WCPS operators, parameter v_g is the name of the geometric property of the input vector feature type and parameter $result$ represents the output stream of variable bindings generated by the operator. For each input feature f, first the range filter for the dimension "geo" of the WCPS operator is configured in line 5 using the geometry of f, and next each of the coverage cells contained in f are obtained in line 7, and the relevant join with f is added to the result in line 8. These steps are illustrated in Figure 7.2(b) for our query example and for a specific municipality. First, the geometry of the municipality

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Algorithm 3 Vector-Raster Join Operation

```
1:  procedure VRJOIN(WFS, v_g, WCPS, result)
2:      result.open();
3:      WFS.open()
4:      while f ← WFS.next() do
5:          WCPS.setRangeFilter("geo", f.getValue(v_g));
6:          WCPS.open()
7:          while c ← WCPS.next() do
8:              result.append(JOIN(f, c))
9:          end while
10:         WCPS.close()
11:     end while
12:     WFS.close();
13:     result.close()
14: end procedure
```

is obtained from variable $?mgeowkt$, next the minimum bounding rectangle (MBR) of the municipality is obtained by the WCPS operator to generate the ranges of the two geospatial subdimensions of the Topo coverage needed to specify the required trimming operation in the WCPS request. The WCPS response yields all the cells of the Topo coverage that lie inside the MBR of the municipality. Finally, predicate $geof{:}sfContains$ is used by the WCPS operator in the SPARQL query engine to discard the coverage cells that lie outside the boundaries of the municipality. Those cells are joined with the data of the municipality and submitted to the result of the operation.

7.3 Vector-Raster Spatial Group and Aggregation

The new VRJOIN operation enables pushing geospatial predicate testing down in the query plan tree, performing their evaluation in the WCPS operator, which delegates the filtering using the feature MBR to the WCPS service. However, the capacity of the WCPS service to process aggregate functions is still not exploited by this operator. To move part of the aggregate processing to the WCPS service a new operator has to be defined, that replaces GROUP, FILTER and JOIN operators when they are located on top of WFS and WCPS operators. The description of the functionality and of the implementation of this new operator is provided in the following subsections.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Table 7.1: Examples of decomposable and self-decomposable aggregate functions

Name	Syntax	Definition	Decomposition
Sefl-decomposable functions			
Sum	$sum(S)$	$\sum_i e_i, e_i \in S$	$sum(S_1) + \ldots + sum(S_n)$
Count	$count(S)$	$\|S\|$	$count(S_1) + \ldots + count(S_n)$
Minimum	$min(S)$	$m \in S \| \nexists e \in S, e < m$	$min(min(S_1), \ldots, min(S_n)$
Maximum	$max(S)$	$M \in S \| \nexists e \in S, e > M$	$max(max(S_1), \ldots, max(S_n)$
Square sum	$sqr_sum(S)$	$\sum_i e_i^2, e_i \in S$	$sqr_sum(S_1) + \ldots + sqr_sum(S_n)$
Decomposable functions			
Average	$avg(S)$	$\mu = \frac{\sum_i e_i}{\|S\|}, e_i \in S$	$sum(S)/count(S)$
Variance	$var(S)$	$v = \frac{\sum_i (e_i - \mu)^2}{(\|S\|-1)}, e_i \in S$	$(count(S) * sqr_sum(S) - sum(S)^2)/$ $(count(S) * (count(S) - 1))$
Std. dev.	$std_dev(S)$	$s = \sqrt{\frac{\sum_i (e_i - \mu)^2}{(\|S\|-1)}}, e_i \in S$	$sqrt($ $(count(S) * sqr_sum(S) - sum(S)^2)/$ $(count(S) * (count(S) - 1)))$

7.3.1 New Operators

Before we start with the description of the functionality of the VRGROUP operation, it has to be said that not all the aggregate functions will be supported. In particular, due to details related to the implementation, only *decomposable* aggregate functions, as they are defined in [63], may be used in the VRGROUP operation.

If S is a multiset of values of some data type then an aggregate function f is *decomposable* if it may be expressed as a combination

$$f(S) = g(f1(S), f2(S), \ldots, f_n(S)),$$

where g is a function and each f_i is a *self-decomposable* aggregate function. The application of a *self-decomposable* function f_i to multiset S, $f_i(S)$, may be further decomposed into applications of the same functions f_i to subsets $S_j \subset S$, which may next be combined with some other function h. More precisely,

$$f_i(S) = h(f_i(S_1), \ldots, f_i(S_m), S_1 \cup \ldots \cup S_m = S.$$

Each of the above aggregate functions f_i is called a *primitive aggregate function*. Table 7.1 provides well-known examples of *decomposable* and *selt-decomposable* aggregate functions. Examples of aggregate functions that are not *decomposable* are those that compute $percentiles$ such as the $median$. Counting distinct values in a set cannot be expressed as a *decomposable*

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

function.

The functionality of the WCPS operator must be extended to support the delegation of the evaluation of aggregate functions to the underlying WCPS service. To achieve this, in the new version of the operator, the *Bindings* parameter may contain either references to dimensions and bands, if the operator is used to query coverage cells, or expressions involving aggregate functions if the operator is used to evaluate aggregates. In this latter case, if the *Filter* parameter contains some spatial predicate, its operand must be either a point or a rectangle, to be directly supported by either trimming or slicing operations over the geospatial dimension. In general, if the WCPS operator is used to evaluate aggregates, them:

- The *Bindings* parameter must be a set of pairs of the form $(v_i : f_i(b_i))$, where each v_i is a variable name, each f_i is an aggregate function supported by the underlying WCPS engine and each b_i is a coverage band.
- The *Filter* parameter may only contain geospatial conditions of either of the two following forms: i) $geof{:}Contains(r, geo)$, ii) $geof{:}Equals(p, geo)$, where $r(p)$ is a geometry of type $rectangle(point)$, and geo denotes the geospatial dimension of the coverage.

If $(minpx, minpy, maxpx, maxpy, mint, maxt)$ are the real coordinates that define the ranges of the geosaptial and temporal filter and $f1(b1), f2(b2), \ldots fn(bn)$ are the aggregate functions used in the *Bindings* parameter, then the WCPS service expression generated is as follows:

```
FOR c IN (CoverageName) RETURN
 encode({1:f1(c.b1[E(minpx:maxpx),N(minpy:maxpy),ansi(mint:maxt)])
         2:f2(c.b2[E(minpx:maxpx),N(minpy:maxpy),ansi(mint:maxt)])
         ...
         n:fn(c.bn[E(minpx:maxpx),N(minpy:maxpy),ansi(mint:maxt)])
        },"csv")
```

Based on the above, the new Vector-Raster Group operator (VRGROUP) has four input parameters.

- A reference to the input WFS operator, that provides the access to the vector geospatial feature dataset.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- The name of the variable generated by the WFS operator that records the WKT representations of the vector feature geometries.
- A reference to the input WCPS operator, that provides access to the cells of the raster geospatial coverage, and also enables the evaluation of aggregates on bands of those cells.
- A set of aggregate bindings of the form $(v : f(b))$, where v is a variable of the bindings generated by the operator and $f(b)$ is an aggregate expression evaluated on the cells of the input coverage for each input vector feature.

The result of the evaluation of the operator is a multiset of variable bindings that includes all the variables generated by the WFS operator and also all the variables defined in the aggregate parameter. For each input vector feature, the raster coverage is queried through the WCPS to compute all the aggregate expressions. As an example, let us consider again the query of Figure 7.1 that computes the average of elevation above sea level for each municipality. The bottom part of an optimized query plan that uses the new operation VRGROUP is shown in Figure 7.3. It is first noticed that the VRGROUP operator works directly on top of the WFS and WCPS operators. The *FeatureGeo* parameter of the operator has the name of the variable where the WFS operator records the geometry of each of its output vector features, i.e., $?mgeowkt$. The *Aggregates* parameter specifies that variable ?0 of the output will contain the average of the $Elev$ band of the input coverage. Thus, each variable binding of the output of the VRGROUP operator will include the following variables:

- Variable $?mun$ containing the URI of each municipality.
- Variable $?mname$ containing the name of each municipality.
- Variable $mgeo$ containing the URI of the blank node corresponding to the geometry of each municipality.
- Variable $?mgeowkt$ containing the WKT representation of the geometry of each municipality.
- Variable ?0 containing the average of the elevation in each municipality.

To achieve the above result, the VRGROUP operator must configure and use the WCPS operator to obtain the average of band $Elev$ inside the geometry of each municipality. To

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

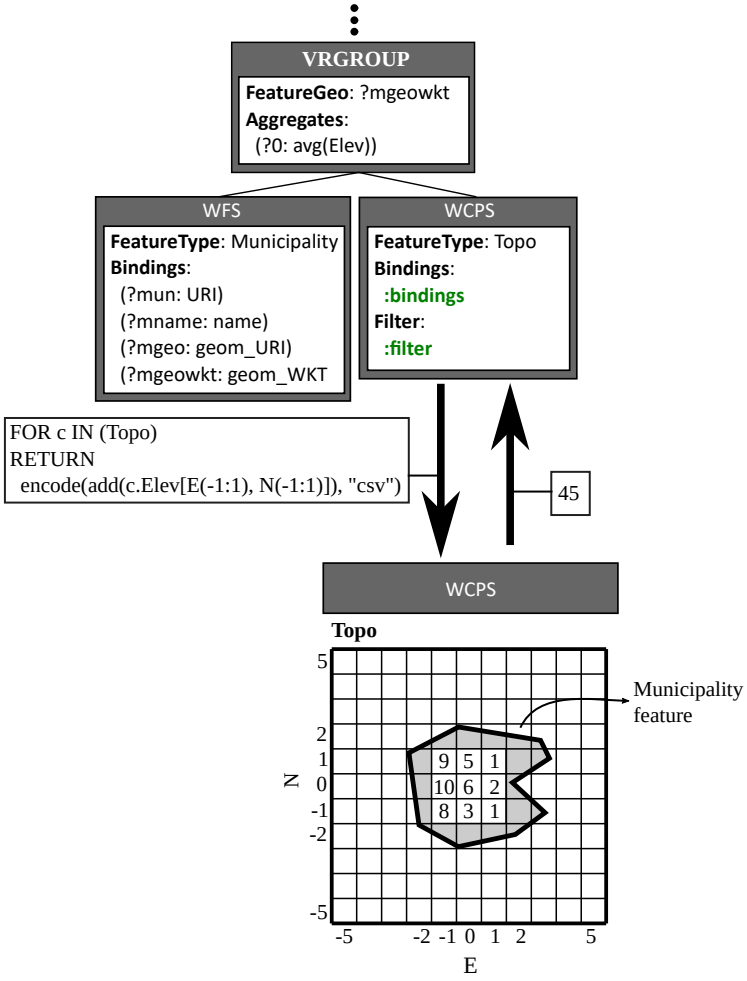


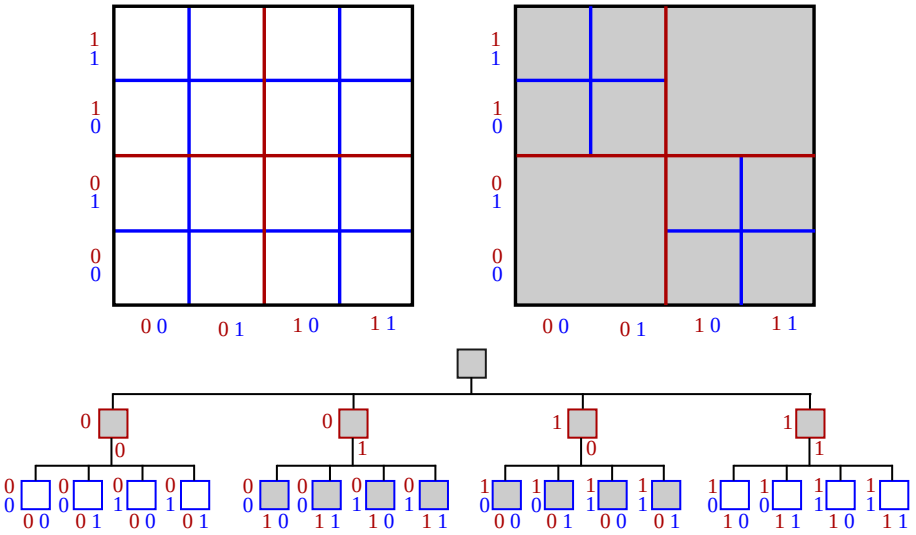
Figure 7.3: Optimized query plan with VRGROUP operation.

achieve this efficiently, the data traffic between the WCPS service and the SPARQL engine should be minimized. Therefore, the implementation should cover as much of the result area as possible with WCPS requests that obtain already aggregated data. In Figure 7.3, it is shown how the VRGROUP operator is using the WCPS operator to generate a WCPS request that computes the sum of the elevation inside a big square that lies inside of a municipality. Thus, with a minimum amount of data transfer, the *sum* has been evaluated in a large area contained in the municipality. The remainder area has to be completed using other WCPS requests that obtain the coverage cells. The question to be replied to get an optimum implementation is how to get an optimal partition of the total geospatial area in rectangles (WCPS requests), such that big rectangles lie inside each municipality, to delegate the evaluation of aggregates to the WCPS service. The current version of the implementation of this operator uses a quadtree structure to partition the space into WCPS requests. Details of the implementation are provided in the next subsection.

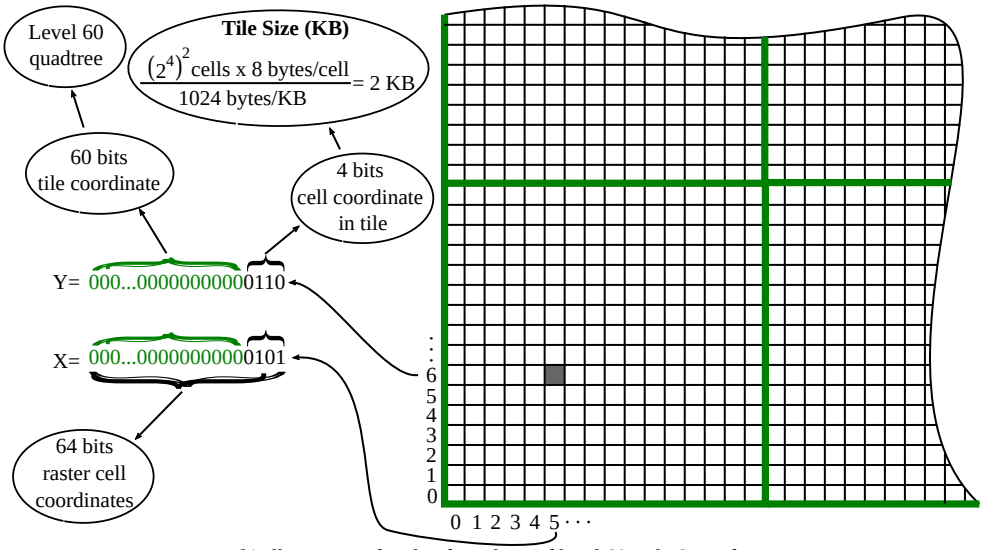
7.3.2 Efficient Implementation

The current implementation of the GROUP operator makes use of a quadtree structure to partition the space occupied by the input vector features into squares. Each square is used to configure the WCPS Filter parameter to obtain either cell data or aggregated data from the input coverage. A quadtree is a tree where each internal node has exactly four children, each of them representing a quadrant of the parents square. Figure 7.4(a) shows a quadtree of three levels and the geometric representation of each of its nodes. The root node (black node and square, is divided into four children (red nodes and squares), and subsequently, each red node is divided into four children (the blue nodes and squares). It is noticed that for nodes of level 1 (red nodes), one bit is sufficient to represent horizontal and vertical coordinates of each node. For nodes of level 2, two bits are needed for each coordinate. First bit (drawn in red color in the figure) matches the bit of the parent node in both horizontal and vertical coordinates. In general, at any level n of the tree, the first p bits starting from the left may be used to obtain the coordinates of the ancestor of level p. Depending on the data represented in each node, a node may be subdivided further or not. Thus for example, in Figure 7.4(a), the nodes filled with grey color represent a quadtree where nodes of level 1 (red nodes) with coordinates $(0, 0)$ and $(1, 1)$ are not subdivided. If we consider coverage cells as leaf nodes of the maximum level of a quadtree, then any quadtree defines a partition of the coverage cells into squares, where the number of cells in a leaf node is determined by the level of such node.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



(a) Quadtree with 3 levels



(b) Illustration of a tile of quadtree of level 60. Tile Size of 2KB.

Figure 7.4: Illustration of a Quadtree and calculation of Quadtree maxium level from WCPS service Minimum Request Size (MRS).

Let us assume now that 64 bits (typical length of a word in current hardware architectures) are used to represent the integer coordinates of the cells of the coverages. In a quadree of level n, n bits are used to represent each leaf node and $64 - n$ bits may be used to represent cell coordinates inside each leaf node. Thus, the maximum number of levels allowed for the quadtree determines the minimum size of a tile represented by a quadtree node, and therefore, the minimum size of a filter square of a WCPS request. Figure 7.4(b) shows a leaf node of a quadtree of level 60, that contains a coverage tile of 16×16 cells (2KB of data assuming only one band of 8 bytes per cell). It is shown how the first 60 bits of a cell coordinate are used to identify the tile and the remainder 4 bits are used to identify each cell inside the tile. Thus, given an objective Minimum Request Size (MRS), in KBs, the number of bits required for the cell coordinates inside the response tile is obtained by the formula:

$$b = \left\lceil \frac{\log\left(\sqrt{\frac{MRS*1024}{8}}\right)}{log(2)} \right\rceil,$$

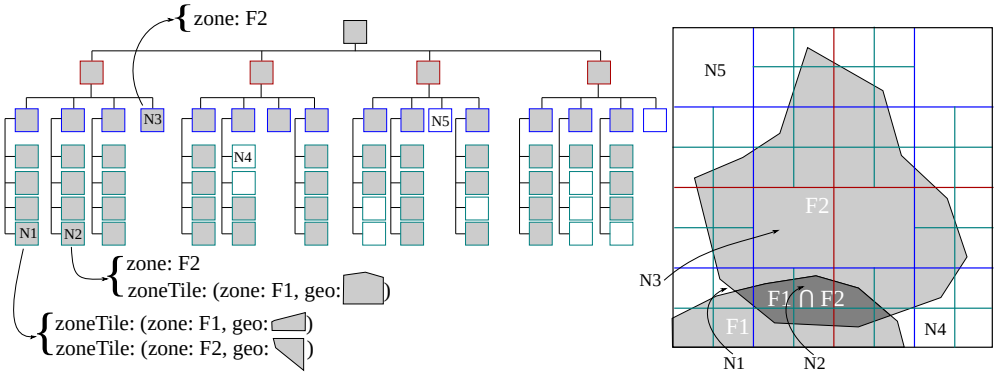
and therefore, the maximum quadtree level is obtained as:

$$MaxLevel = 64 - b.$$

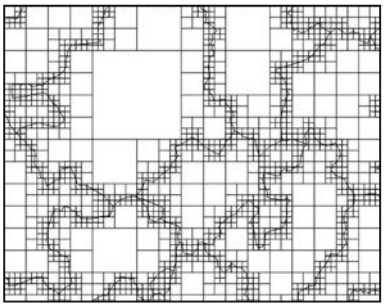
The VRGROUP implementation generates a quadtree based on the geometries of the input vector features. Figure 7.5(a) shows a quadtree where two vector features, $F1$ and $F2$, have been inserted. There are three types of quadtree leaf nodes:

- Nodes completely outside of any vector feature, like nodes $N5$ and $N4$ in Figure 7.5(a). These nodes area ignored during the quadtree processing.
- Nodes completely inside one or more vector features, like node $N3$ in Figure 7.5(a). Each such node records a reference to an input vector feature, which is called a *zone*. Given that their geometry is completely contained inside one or various input *zones*, during the quadtree processing, these nodes generate WCPS operator executions that evaluate the involved primitive aggregate functions on the input coverage, to accumulate the results in each *zone*.
- Nodes of the maximum level of the quadtree, i.e., corresponding to tiles of size MRS, like nodes $N1$ and $N2$ in Figure 7.5(a). These nodes partially intersect with at least one input feature, therefore, aggregate data may not be obtained from the input coverage for them. The part of a *zone* that is contained in the node is called a *zoneTile*. A *zoneTile* references an input vector feature, but it also records the intersection between the feature geometry and the node. This part of the vector feature geometry will be used to test

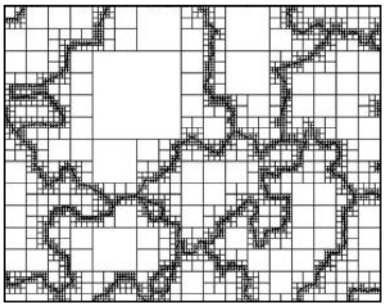
USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



(a) Illustration of the quadtree structure used for implementation of the VRGROUP operator



(b) Quadtree Tiles for MRS = 128KB (57 levels)



(c) Quadtree Tiles for MRS = 32KB (58 Levels)

Figure 7.5: Illustration of the quadtree structure used in the implementation of the VRGROUP operator.

if obtained coverage cells are inside the feature, faster than if the whole vector feature geometry were used. In addition to the *zonteTiles* of the intersecting features, these nodes might algo include *zones*, corresponding to vector features that might entirely contain the node geometry. Thus, node $N1$ of Figure 7.5(a) records two *nodeTiles*, one referencing a piece of feature $F1$ and another referencing a piece of feature $F2$. On the other hand, node $N2$ contains a *nodeTile* referencing feature $F1$, since $N2$ is only partially intersecting $F1$, but also a *zone* referencing feature $F2$, since $F2$ completely contains $N2$.

Based on the above, when a new vector feature is inserted in a quadtree, leaf nodes that partially intersect the feature might be further subdivided if the maximum level has not been

Algorithm 4 Insertion of Vector Features in the Quadtree

```
1:  function INSERTZONE(zt, Q)
2:      if isLeaf(Q) then
3:          if contains(zt, getGeometry(Q)) then addZone(zt, Q); return Q;
4:          else if level(Q) = maxLevel(Q) then addZoneTile(zt, Q); return Q;
5:          else
6:              children ← split(Q);
7:              for all c ∈ children do
8:                  if intersects(zt, getGeometry(c)) then
9:                      c ← insertZone(intersection(zt, getGeometry(c)), c);
10:                 end if
11:             end for
12:             return newNonLeafNode(children);
13:         end if
14:     else
15:         for all c ∈ children(Q) do
16:             if intersects(zt, getGeometry(c)) then
17:                 c ← insertZone(intersection(zt, getGeometry(c)), c);
18:             end if
19:         end for
20:         return Q;
21:     end if
22: end function
```

reached. Algorithm 4 provides a pseudocode for the insertion of a *zoneTile zt* in a quadtree Q. If Q is not a leaf node then zt is recursively inserted in each of its four children, in case zt intersects with the child node. The intersection of zt with the geometry of the child node is computed before the insertion (see Line 17). This way, the geometry of each *zoneTile* is always contained in its node. The computation of these intersections ensures that the number of points in the border of *zoneTiles* is smaller as it is pushed down in the tree. This is important to speed up the point in polygon evaluations that are required to test if a coverage cell is contained or not inside a vector feature. If Q is a leaf node that is completely contained in zt, then zt has to be added to the list of *zones* of Q (see line 3). Otherwise, if Q is already at the maximum level (see line 4), then zt has to be added to the list of *zoneTiles* of Q, since zt partially intersects Q and Q may not be split further. If none of the two above cases hold, then Q has to be split into four children, and zt has to be inserted in the intersecting children. This is performed in lines 6 – 12.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

The geometries of the leaf nodes of a specific area of two quadtrees constructed from the municipalities of the Spanish region of Galicia are shown in Figures 7.5(b) and 7.5(c). The value of MRS used to generate the quadtree shown in Figure 7.5(b) is 128KB, which yields a maximum quadtree level of 57. On the other hand, one more level of depth was allowed to construct the quadtree shown in Figure 7.5(c), whose leaf node tiles have a size of 32KB.

Algorithm 5 Quadtree Processing

```
1:  procedure PROCESSQTREE(Q, WCPS)
2:      if isLeaf(Q) then
3:          if zoneTiles(Q) ≠ ∅ then
4:              WCPS.setBindings(getQueryBindings(Q));
5:              WCPS.setRangeFilter("geo", getGeometry(Q));
6:              WCPS.open();
7:              while c ← WCPS.next() do
8:                  for all z ∈ zones(Q) do accumulateCell(c, z);
9:                  end for
10:                 for all zt ∈ zoneTiles(Q) do
11:                     if contains(zt, c) then accumulateCell(c, zt);
12:                     end if
13:                 end for
14:             end while
15:         else
16:             if zones(Q) ≠ ∅ then
17:                 WCPS.setBindings(getAggregateBindings(Q));
18:                 WCPS.setRangeFilter("geo", getGeometry(Q));
19:                 WCPS.open();
20:                 if c ← WCPS.next() then
21:                     for all z ∈ zones(Q) do accumulateAggregate(c, z);
22:                     end for
23:                 end if
24:             end if
25:         end if
26:     else
27:         for all c ∈ children(Q) do
28:             processQTree(c, WCPS)
29:         end for
30:     end if
31: end procedure
```

Once the quadtree is build, using the vector features, it is processed with a depth-first

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

traversal, where a WCPS operator evaluation is generated for each leaf node. The pseudocode of this depth-first traversal of the quadtree is given in Algorithm 5. Quadtree Q is recursively processed as follows. If Q is not a leaf node, then each of its children have to be processed (lines $27 - 29$). Otherwise, the use of the WCPS operator to either query coverage cells or evaluate primitive aggregate functions will depend on whether the set of *zoneTiles* of Q is empty or not. If the set of *zoneTiles* is empty (see lines $16 - 24$), then all the intersecting vector features are referenced through *zones*, i.e., the node is completely contained in all the intersecting vector features. In this case, WCPS is configured to perform an evaluation of primitive aggregate functions in line 17, the Filter parameter is set in line 18 to perform a spatial range filter using the node geometry. The results of the aggregate evaluations are accumulated in each *zone* of the node in line 21. On the other hand, if the set of *zoneTiles* of Q is not empty, it means that at least an intersecting vector feature does it only partially, and each of the cells of the coverage have to be tested to decide if they are either inside or outside the vector feature. Thus, the WCPS operator is configured to get all the cells inside the geometry of Q (see lines $4 - 5$). Next, for each cell c obtained by the WCPS, the values of the involved bands are accumulated in each *zone* of Q (see lines $8 - 9$), and they are accumulated also in those *zoneTiles* of Q that contain c (see lines $10 - 13$). Thus, after the quadtree has been processed, each input vector feature has accumulated all the values of the self-decomposable primitive aggregate functions needed to compute the required decomposable aggregate functions.

The pseudocode of the VRGROUP operator, that makes use of the above quadtree algorithms is shown in Algorithm 6. In line 2, the quadtree structure is initialized, using the value configured for the MRS. In this line, the maximum quadtree level is computed from the MRS. Lines $5 - 10$ obtain the vector features from the WFS operator, create *zones* by combining the features with the information of the aggregate functions that have to be computed (line 6) and create *zoneTiles* to be inserted in the quadtree (lines $8 - 9$). The created zones are maintained in a list that will be processed to produce the operator result (see line 7). After the creation of the quadtree and the insertion of all the vector features, the quadtree may be processed in line 12 to accumulate in each *zone* the values of the primitive functions. Finally, lines $14 - 16$ compute the aggregate functions from the primitive aggregated data and combine such results with the vector feature data in the operator result.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Algorithm 6 Vector-Raster Group Operation

```
1:  procedure VRGROUP(WFS, v_g, WCPS, F, result)
2:      Q ← createQTree(MRS);
3:      WFS.open();
4:      vectorZones ← ∅
5:      while f ← WFS.next() do
6:          z ← createVectorZone(f, F)
7:          append(z, vectorZones)
8:          zt ← createZoneTile(z, f.getValue(v_g));
9:          Q ← insertZone(zt, Q)
10:     end while
11:     WFS.close()
12:     processQTree(Q, WCPS)
13:     result.open()
14:     for all z ∈ vectorZones do
15:         append(computeAggregates(z), result)
16:     end for
17:     result.close()
18: end procedure
```

7.4 Performance Evaluation

In this section it is shown how the use of the new VRJOIN and VRGROUP operators enables obtaining query plans with improved performance. In particular, the use of VRJOIN brings response times of around one order of magnitude faster than those of GeoLD with FILTER and JOIN operators. Furthermore, the use of VRGROUP enables achieving response times between one and two orders of magnitude faster than those obtained with VRJOIN. The following subsections describe the evaluation setup, show the results of a first experiment aimed at determining the best value for the MRS parameter and finally show the results of the performance comparison of the query plans generated with the three approaches.

7.4.1 Evaluation Setup

The query plans and datasets used in the evaluation experiments undertaken are described below. The structure of all the queries used in all the experiments matches the one of the query shown in Figure 7.1. The main difference will the the vector feature dataset used and the spatial resolution used for the raster coverage dataset. Three different query plan structures

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

are considered.

JOIN-FILTER-JOIN These query plans follow the structure of the query plan shown in Figure 7.1. These are the query plans currently supported by the GeoLD optimizer. They incorporate WFS and WCPS operators, but they do not use any of the two operators introduced in the present chapter. Therefore, the spatial join between vector features and raster cells is supported by the sequence of a conventional SPARQL JOIN operator followed by a FILTER operator. The calculation of the aggregates is done with a conventional SPARQL GROUP operator.

VRJOIN These query plans follow the structure of the query plan of Figure 7.2(a). In these query plans, the JOIN and FILTER operators are replaced by a more specific VRJOIN operator. However, aggregate calculations are still performed by the conventional SPARQL GROUP operator, thus aggregated computing may not be delegated to the WCPS service.

VRGROUP These query plans follow the structure of the query plan of Figure 7.3. Now the VRGROUP operator performs the spatial join and the aggregate computation using the capabilities provided directly by the WFS and WCPS operators, delegating part of the aggregate evaluation to the WCPS service.

The raster coverages used for the evaluation match the spatial DEM coverages already described in Subsection 6.3.1. Regarding vector feature datasets, three different levels of administrative divisions of the Spanish region of Galicia were used. The main characteristics of all these datasets are shown in Figure 7.6. In particular, for each vector dataset, the figure shows the number of features and the average number of points per feature. Province features are few and simple (few points per feature). Shires are not many, but each of them is quite complex in terms of number of points in the border. Finally, municipalities is the larger dataset, although their vector geometries are simpler than those of shires. Regarding raster coverages, four different coverages were generated from the DEM, whose spatial resolutions and number of cells (pixels) are shown in the figure. The hardware and base software configuration is the one already described in Subsection 6.3.1.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Vector Feature Datasets

Dataset	Features	Points per Feature (avg)
Provinces	4	768
Shires	53	1692
Municipalities	314	809

Raster Coverage Datasets

Resolution	Size (pixels)
200	1056 x 1103
100	2112 x 2207
50	4225 x 4415
25	8451 x 8830

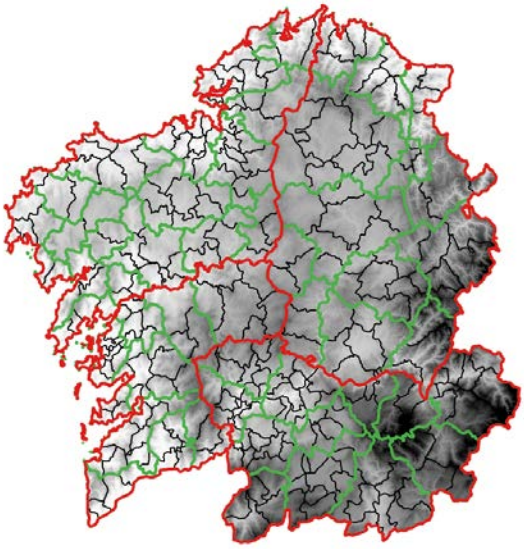
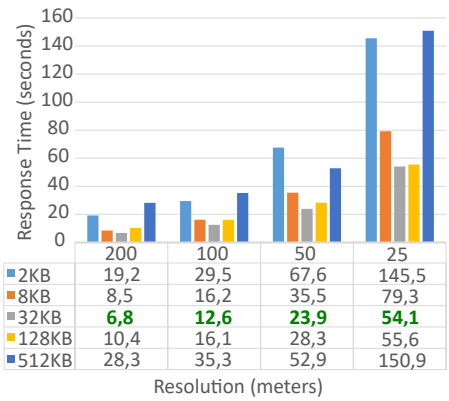


Figure 7.6: Datasets used for the evaluation of the VRJOIN and VRGROUP operators.

7.4.2 Minimum Request Size

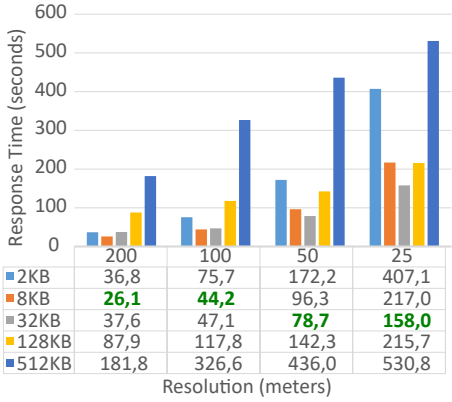
The objective of the first experiment was the finding of the best Minimum Request Size (MRS) parameter for the *VRGROUP* operator, and for each combination of input datasets. Thus, a query plan with the structure of that of Figure 7.3 was evaluated for each combination of vector feature dataset (Provinces, Shires and Municipalities) with raster coverage dataset (resolutions 200, 100, 50 and 25). In general, the results show how the best MRS size is obtained as as compromise between the many WCPS request that have to be done when the MRS is small and the many point in polygon tests to be done for each request when the MRS is large. The results for the vector feature dataset of Provinces are shown in Figure 7.7(a). For all the raster coverage resolutions, the best MRS is always $32KB$. However, as it may be observed in the figure, for bigger cell sizes, a smaller MRS of $8KB$ is closer to be optimum, whereas for smaller cell sizes, the larger MRS of $128KB$ is closer to be optimum. The reason of this better performance of larger MRSs when smaller cells are used for the same vector features is that, the same MRS with smaller cell size implies request rectangles with smaller area. Thus, to maintain the optimum area that keeps the best compromise between the number of request and number of point in polygon tests, it migh be necesary to increase the MRS. This is clearly

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



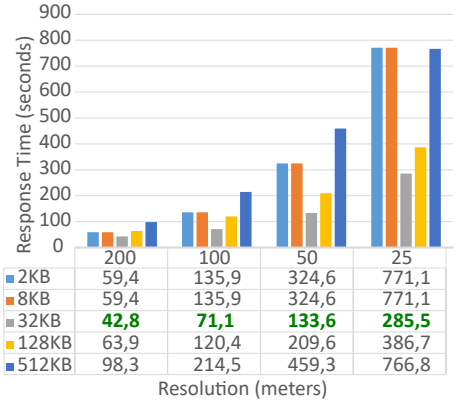
	200	100	50	25
2KB	19,2	29,5	67,6	145,5
8KB	8,5	16,2	35,5	79,3
32KB	**6,8**	**12,6**	**23,9**	**54,1**
128KB	10,4	16,1	28,3	55,6
512KB	28,3	35,3	52,9	150,9

(a) Provinces



	200	100	50	25
2KB	36,8	75,7	172,2	407,1
8KB	**26,1**	**44,2**	96,3	217,0
32KB	37,6	47,1	**78,7**	**158,0**
128KB	87,9	117,8	142,3	215,7
512KB	181,8	326,6	436,0	530,8

(b) Shires



	200	100	50	25
2KB	59,4	135,9	324,6	771,1
8KB	59,4	135,9	324,6	771,1
32KB	**42,8**	**71,1**	**133,6**	**285,5**
128KB	63,9	120,4	209,6	386,7
512KB	98,3	214,5	459,3	766,8

(c) Municipalities

Figure 7.7: Determination of the best MRS for the implementation of the VRGROUP opertor.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

observable when Shires are used as input vector feature dataset. The results are shown in Figure 7.7(b). Now, the best MRS for large cells of 200 and 100 meters is $8KB$. But when the cell size is decreased to 50 and 25 meters, then the optimum MRS is $32KB$. Notice that Shires, compared to Provinces, have smaller area, but at the same time have many more points per feature in their borders, therefore, the point in polygon tests are much more expensive, and therefore, the optimum moves towards the direction of having more request and less point in polygon tests, i.e., towards smaller MRS. Finally, Figure 7.7(c) shows the results for the vector feature dataset of Municipalities. Again, the smaller number of points per feature yield better performance when the MRS is $32KB$ in all the cases, as with Provinces.

7.4.3 Performance Testing

The second experiment compares the performance of the three query plans with all the combinations of input datasets. The Minimum Request Size (MRS) used for the implementation of the VRGROUP operator was $32KB$, the one that showed better performance in most cases in the first experiment.

Figure 7.8 shows how the response time evolves with respect to the raster coverage resolution for each of the vector feature datasets and for each query plan. In general, it may be observed that the VRGROUP query plan has always the best performance. Besides, the query plan with the VRGROUP operation is the only one that enables the evaluation using raster coverages of 25 meters of resolution with reasonable response times.

The graph of the left side of Figure 7.8(a) shows how the VRJOIN operator enables obtaining query plans that run with response times much better than those offered by the JOIN-FILTER-GROUP query plans of GeoLD, when the vector feature dataset of Provinces is used. In particular, the performance improvement increases with the size of the raster coverage, reaching around 3 times faster with the coverage of 50 meters. Beyond that, the graph of the right side of the same figure shows how the VRGROUP operation enables the generation of query plans of around 200 times faster than those generated with operation VRJOIN, with coverages of 50 meters of resolution.

The results of the comparison of the three query plans for all the coverages and for the Shires vector feature dataset are shown in Figure 7.8(b). The left side of the figure shows results similar to those of Provinces when the VRJOIN query plans are compared with the JOIN-FILTER-GROUP query plans. On the other hand, the rifht side of the figure shows that

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

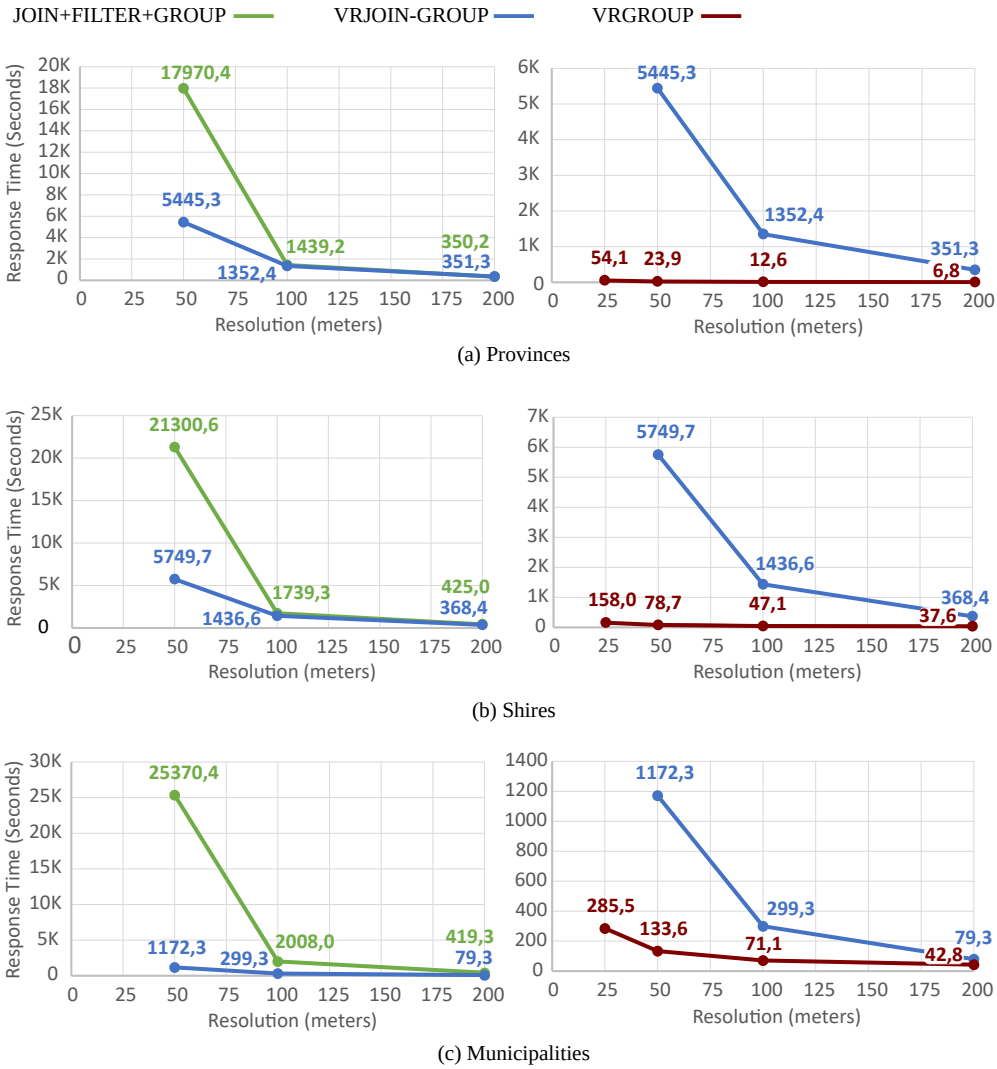


Figure 7.8: Comparison of the performance of the three implemented approaches for the group and aggregation of coverage data over vector features.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

now VRGROUP query plans are only around 70 times faster than VRJOIN plans for coverages of 50 meters.

The results of the comparison for the Municipalities vector feature dataset are shown in Figure 7.8(c). The left side of the figure shows that the difference between VRJOIN plans and JOIN-FILTER-GROUP plans have increased to reach around 20 times faster. On the other hand, the difference of VRJOIN and VRGROUP query plans is now of only 8 times faster.

In general, the response times of VRGROUP query plans are much better than those of VRJOIN query plans, which in turn are much better than those of JOIN-FILTER-GROUP query plans. The difference in resonse times between VRJOIN and JOIN-FILTER-GROUP query plans increases with the raster coverage resolution and increases also with the number of vector features in the datasets. On the other hand, the difference between VRGROUP and VRJOIN query plans increases with the resolution of the raster coverages and increases also with the average area of the vector features, as large vector features bring more opportunities to delegate aggregate processing to the WCPS service along larger query rectangles.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

CHAPTER 8

CONCLUSIONS AND FURTHER WORK

Some conclusions and issues related to further work are discussed below.

8.1 Conclusions

The design and implementation of GeoLD responds to the need of advances in the efficient query of data obtained from *Raster Coverages* through linked data standards. The system enables efficient access to heterogeneous environmental data sources, including *Vector Features* and *Raster Coverages*, available through standard OGC web services, through the use of a GeoSPARQL end point. The main advantages of the proposed solution with respect to the state of the art geospatial linked data technologies may be resumed as follows.

- The system enables the querying of *Vector Features* directly available through OGC WFS services. Conventional and geospatial filters of the input datasets are delegated to the underlying web service, which enables improving the query response time.
- The system enables the querying of *Raster Coverages* directly available through OGC WCPS services. The performance of the queries that include geospatial and/or temporal filter of the input *Coverages* are improved by the automatic incorporation of subsetting patterns in the WCPS request on the spatial and/or temporal dimensions.
- *Raster Coverages* may be efficiently queried by just using GeoSPARQL predicates and functions, without the need to use specific raster data types and large lists of raster functions.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

- The system incorporates a new Coverage to RDF Mapping Language (C2RML) that enables the customization of the mapping from the *Raster Coverages* to RDF, by using vocabularies defined by the programmer.
- New SPARQL operations, proposed in the Thesis and used by the system to generate query plans, enable achieving the efficient aggregation of *Raster Coverage* data over the zones defined by *Vector Features*, again, without forcing the user to have additional skills related to new raster functions.
- The query performance reached by GeoLD outperforms state of the art GeoSPARQL implementations and also GeoSARQL extensions specifically designed for the support of array or raster data. Furthermore, the new Join and Group operations enable the generation of query plans that achieve response times orders of magnitude faster than those generated with conventional Join and Group SPARQL operations.

The main drawbacks of the current version of GeoLD are related to improvements that might be incorporated in future versions and that are described in the following section.

8.2 Further Work

Future research lines may be resumed as follows.

- The optimum value for the Minimum Request Size (MRS) in the implementation of the VRGROUP operation depends on various factors that include the system hardware and network and input dataset characteristics. The current implementation uses a constant MRS that might be tuned by the system administrator. However, a cost model that provides a good estimation of the expected query response time for a specific MRS would enable the determination of the best MRS for each query.
- The in-memory representation of variable bindings of the current version of GeoLD is the one provided by the underlying Jena ARQ engine. It is expected that the incorporation of new compact representations that might take into account the specificities of the raster data would bring additional performance improvements, specially when large subsets of *Raster Coverages* are retrieved from the underlying services. These representations would take advantage of the regular sampling nature of the geoesptial and temporal dimensions of the *Raster Coverages*.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

- The current version of GeoLD provides new SPARQL operators to perform dataset scan and filtering for both *Vector Features* and *Raster Coverages* available through OGC web services. The system incorporates also specific implementations of Join and Group operations to combine and aggregate vector and raster data. Other spatial indexing and join strategies, available in the area of spatial query processing, might yield good query response times in some queries, thus they should be implemented and tested.

- Current query optimization of GeoLD identifies query subtrees that might be replaced by instances of the new WFS and WCPS operators. New strategies should be incorporated in the optimizer to leverage the implementation of the remainder operations, including the new implementations of Join and Group.

- The Technology Readiness Level (TRL) of the current implementation of GeoLD lies between TRL3 and TRL4. That means that an experimental proof of concept has been implemented. A more exhaustive evaluation, with more datasets of different sizes has to be done to definitely reach TRL4 (technology validated in lab). Further work on the implementation should be done to reach a higher TRL. A reasonable effort to face technology transfer objectives should bring the software to TRL6, i.e., a demonstration of the technology in some relevant environment.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] Adel M. Abdallah and David E. Rosenberg. A data model to manage data for water resources systems modeling. *Environmental Modelling & Software*, 115:113 – 127, 2019.

[2] Andrei Aiordăchioaie and Peter Baumann. Petascope: An open-source implementation of the ogc wcs geo service standards suite. In *International Conference on Scientific and Statistical Database Management*, pages 160–168. Springer, 2010.

[3] Riccardo Albertoni, David Browning, Simon Cox, Alejandra Gonzalez Beltran, Andrea Perego, and Peter Winstanley. Data catalog vocabulary (dcat) - version 2. W3c candidate recommendation., World Wide Web Consortium., 2019.

[4] Riccardo Albertoni and Antoine Isaac. Data on the web best practices: Data quality vocabulary. W3c working group note., World Wide Web Consortium., 2016.

[5] Andrej Andrejev and Tore Risch. Scientific sparql: semantic web queries over scientific data. In *2012 IEEE 28th International Conference on Data Engineering Workshops*, pages 5–10. IEEE, 2012.

[6] Andrej Andrejev, Salman Toor, Andreas Hellander, Sverker Holmgren, and Tore Risch. Scientific analysis by queries in extended sparql over a scalable e-science data store. In *2013 IEEE 9th International Conference on e-Science*, pages 98–106. IEEE, 2013.

[7] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and Juan Sequeda. A direct mapping of relational data to rdf. W3c recommendation., World Wide Web Consortium., 2012.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[8] Nikolaos Athanasis, Kostas Kalabokidis, Michail Vaitis, and Nikolaos Soulakellis. Towards a semantics-based approach in the development of geographic portals. *Computers & Geosciences*, 35(2):301 – 308, 2009.

[9] Omer Saud Azeez, Biswajeet Pradhan, Helmi Z. M. Shafri, Nagesh Shukla, Chang-Wook Lee, and Hossein Mojaddadi Rizeei. Modeling of co emissions from traffic vehicles using artificial neural networks. *Applied Sciences*, 9(2), 2019.

[10] J.L.P. Barker and C.J.A. Macleod. Development of a national-scale real-time twitter data mining pipeline for social geodata on the potential impacts of flooding on communities. *Environmental Modelling & Software*, 115:213 – 227, 2019.

[11] Peter Baumann. Management of multidimensional discrete data. *VLDB J.*, 3(4):401–444, 1994.

[12] Peter Baumann. Web coverage processing service (wcps) language interface standard. https://www.ogc.org/standards/wcps, 2009. Online; accessed April-2022.

[13] Peter Baumann. The ogc web coverage processing service (wcps) standard. *Geoinformatica*, (14):447–479, 2010.

[14] Peter Baumann. Ogc wcs 2.0 interface standard- core: Corrigendum. Ogc interface standard., Open Geospatial Consortium Inc., 2012.

[15] Peter Baumann, Andreas Dehmel, Paula Furtado, Roland Ritsch, and Norbert Widmann. The multidimensional database system rasdaman. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 575–577, 1998.

[16] Peter Baumann, Eric Hirschorn, and Joan Masó. Ogc coverage implementation schema. Ogc implementation standard, Open Geospatial Consortium Inc., 2017.

[17] Konstantina Bereta, Guohui Xiao, and Manolis Koubarakis. Ontop-spatial: Ontop of geospatial databases. *Journal of Web Semantics*, 58:100514, 2019.

[18] Tim Berners-Lee. Linked-data design issues. W3C design issue document, 2009.

[19] Mike Botts and Alexandre Robin. Opengis sensor model language (sensorml) implementation specification. https://www.ogc.org/standards/sensorml, 2007. Online; accessed July-2022.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[20] Dan Brickley and R.V. Guha. Rdf schema 1.1. W3c recommendation., World Wide Web Consortium., 2014.

[21] Paul G. Brown. Overview of scidb: Large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 963–968, New York, NY, USA, 2010. ACM.

[22] Arne Bröring, Christoph Stasch, and Johannes Echterhoff. Ogc sensor observation service interface standard. version 2.0. Opengis implementation standard., Open Geospatial Consortium Inc., 2012.

[23] Agustina Buccella, Alejandra Cechich, and Pablo Fillottrani. Ontology-driven geographic information integration: A survey of current approaches. *Computers & Geosciences*, 35(4):710 – 723, 2009. Geoscience Knowledge Representation in Cyberinfrastructure.

[24] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub. The geojson format. Standards track, Internet Engineering Task Force (IETF), 2016.

[25] Eliseo Clementini and Paolino Di Felice. A comparison of methods for representing topological relationships. *Information Sciences - Applications*, 3(3):149–178, 1995.

[26] European Commission. Geodcat application profile for data portals in europe, geodcat-ap v1.0.1. https://joinup.ec.europa.eu/release/geodcat-ap/101, 2016. Online; accessed July-2022.

[27] European Commission. Dcat application profile for data portals in europe, dcat-ap v1.1. https://joinup.ec.europa.eu/release/dcat-ap/12, 2018. Online; accessed July-2022.

[28] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The ssn ontology of the w3c semantic sensor network incubator group. *Journal of Web Semantics*, 17:25 – 32, 2012.

[29] Simon Cox. Observations and measurements - xml implementation. Opengis implementation standard, Open Geospatial Consortium Inc., 2011.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[30] Simon Cox. Observations and measurements. verion 2.0. The opengis abstract specification., Open Geospatial Consortium Inc., 2013.

[31] Richard Cyganiak and Dave Reynolds. The rdf data cube vocabulary. W3c recommendation., World Wide Web Consortium., 2014.

[32] Souripriya Das, Seema Sundara, and Richard Cyganiak. R2rml: Rdb to rdf mapping language. W3c recommendation., World Wide Web Consortium., 2012.

[33] Jeff de la Beaujardiere. Opengis web map server implementation specification. Opengis implementation specification., Open Geospatial Consortium Inc., 2006.

[34] Emmanuel Devys, Ted Habermann, Chuck Heazel, Roger Lott, and Even Rouault. Ogc geotiff standard. https://docs.ogc.org/is/19-008r4/19-008r4.html, 2019. Online; accessed July-2022.

[35] Ben Domenico. Ogc network common data form (netcdf) core encoding standard version 1.0. http://www.opengis.net/doc/IS/netcdf/1.0, 2011. Online; accessed July-2022.

[36] H. Dong, M. Halem, and S. Zhou. Social media data analytics applied to hurricane sandy. In *2013 International Conference on Social Computing*, pages 963–966, Sep. 2013.

[37] J. Dutta, F. Gazi, S. Roy, and C. Chowdhury. Airsense: Opportunistic crowd-sensing based air quality monitoring system for smart city. In *2016 IEEE SENSORS*, pages 1–3, Oct 2016.

[38] Prabal Dutta, Paul M. Aoki, Neil Kumar, Alan Mainwaring, Chris Myers, Wesley Willett, and Allison Woodruff. Common sense: Participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 349–350, New York, NY, USA, 2009. ACM.

[39] Bakinam T. Essawy, Jonathan L. Goodall, Hao Xu, and Yolanda Gil. Evaluation of the ontosoft ontology for describing metadata for legacy hydrologic modeling software. *Environmental Modelling & Software*, 92:317 – 329, 2017.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[40] R. Foley. Integrated spatial data infrastructure. In Rob Kitchin and Nigel Thrift, editors, *International Encyclopedia of Human Geography*, pages 507 – 511. Elsevier, Oxford, 2009.

[41] Mark Gahegan, Junyan Luo, Stephen D. Weaver, William Pike, and Tawan Banchuen. Connecting geon: Making sense of the myriad resources, researchers and concepts that comprise a geoscience cyberinfrastructure. *Computers & Geosciences*, 35(4):836 – 854, 2009.

[42] Fabien Gandon and Guus Schreiber. Rdf 1.1 xml syntax. W3c recommendation., World Wide Web Consortium., 2014.

[43] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, November 2011.

[44] Andrea Ghermandi and Michael Sinclair. Passive crowdsourcing of social media in environmental research: A systematic map. *Global Environmental Change*, 55:36 – 47, 2019.

[45] John Graybeal, Anthony W. Isenor, and Carlos Rueda. Semantic mediation of vocabularies for ocean observing systems. *Computers & Geosciences*, 40:120 – 131, 2012.

[46] Jennifer Green, Catherine Dolbear, Glen Hart, John Goodwin, and Paula Engelbrecht. Creating a semantic integration system using spatial data. In *Proceedings of the 2007 International Conference on Posters and Demonstrations-Volume 401*, pages 70–71. CEUR-WS. org, 2008.

[47] W3C SPARQL Working Group. Sparql 1.1 overview. W3c recommendation., World Wide Web Consortium., 2013.

[48] Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos Lorentzos, Enrico Nardelli, Markus Schneider, and Jose R. R. Viqueira. *Chapter 4: Spatio-temporal Models and Languages: An Approach Based on Data Types*, pages 117–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[49] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. Semantic sensor network ontology. W3c recommendation., World Wide Web Consortium., 2017.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[50] Steve Harris and Andy Seaborne. Sparql 1.1 query language. W3c recommendation., World Wide Web Consortium., 2013.

[51] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space (1st edition)*. Morgan & Claypool, 2011.

[52] John Herring. Topic 1: Feature geometry. The opengis abstract specification., Open Geospatial Consortium Inc., 2001.

[53] John Herring. Simple feature access - part 1: Common architecture. Opengis implementation standard., Open Geospatial Consortium Inc., 2011.

[54] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer (second edition). W3c recommendation., World Wide Web Consortium., 2012.

[55] Timo Homburg, Steffen Staab, and Daniel Janke. Geosparql+: Syntax, semantics and system for integrated querying of graph, raster and vector data. In Jeff Z. Pan, Valentina Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *The Semantic Web – ISWC 2020*, pages 258–275, Cham, 2020. Springer International Publishing.

[56] Jeffery S. Horsburgh, Anthony K. Aufdenkampe, Emilio Mayorga, Kerstin A. Lehnert, Leslie Hsu, Lulin Song, Amber Spackman Jones, Sara G. Damiano, David G. Tarboton, David Valentine, Ilya Zaslavsky, and Tom Whitenack. Observations data model 2: A community information model for spatially discrete earth observations. *Environmental Modelling & Software*, 79:55 – 74, 2016.

[57] Jeffery S. Horsburgh, David G. Tarboton, David R. Maidment, and Ilya Zaslavsky. A relational model for environmental and water resources data. *Water Resources Research*, 44(5), 2008.

[58] Jeffery S. Horsburgh, David G. Tarboton, Michael Piasecki, David R. Maidment, Ilya Zaslavsky, David Valentine, and Thomas Whitenack. An integrated system for publishing environmental observations data. *Environmental Modelling & Software*, 24(8):879 – 888, 2009.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[59] ISO 19115-1:2014. Geographic information – metadata – part 1: Fundamentals, iso/tc 211 geographic information / geomatics. https://www.iso.org/standard/53798.html, 2014. Online; accessed July-2022.

[60] ISO/IEC JTC 1/SC 32 Data management and interchange. Iso/iec 13249-3:2016. information technology — database languages — sql multimedia and application packages — part 3: Spatial. Iso standard., International Organization for Standardization (ISO)., 2016.

[61] ISO/TS 19139-1:2019. Geographic information – xml schema implementation – part 1: Encoding rules, iso/tc 211 geographic information/geomatics. https://www.iso.org/standard/67253.html, 2019. Online; accessed July-2022.

[62] Krzysztof Janowicz, Sven Schade, Arne Bröring, Carsten Keßler, Patrick Maué, and Christoph Stasch. Semantic enablement for spatial data infrastructures. *Transactions in GIS*, 14(2):111–129, 2010.

[63] Paulo Jesus, Carlos Baquero, and Paulo Sergio Almeida. A survey of distributed data aggregation algorithms. *IEEE Communications Surveys Tutorials*, 17(1):381–404, 2015.

[64] Liangcun Jiang, Peng Yue, Werner Kuhn, Chenxiao Zhang, Changhui Yu, and Xia Guo. Advancing interoperability of geospatial data provenance on the web: Gap analysis and strategies. *Computers & Geosciences*, 117:21 – 31, 2018.

[65] Cliff Kottman and Carl Reed. Topic 5: Features. The opengis abstract specification., Open Geospatial Consortium Inc., 2009.

[66] Manolis Koubarakis, Mihai Datcu, Charalambos Kontoes, Ugo Di Giammatteo, Stefan Manegold, and Eva Klien. Teleios: A database-powered virtual earth observatory. *Proc. VLDB Endow.*, 5(12):2010–2013, August 2012.

[67] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: A semantic geospatial dbms. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2012*, pages 295–311, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[68] Kostis Kyzirakos, Ioannis Vlachopoulos, Dimitrianos Savva, Stefan Manegold, and Manolis Koubarakis. Geotriples: A tool for publishing geospatial data as rdf graphs using r2rml mappings. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272*, ISWC-PD'14, page 393–396, Aachen, DEU, 2014. CEUR-WS.org.

[69] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. Prov-o: The prov ontology. W3c recommendation., World Wide Web Consortium., 2013.

[70] M. Lutz, J. Sprado, E. Klien, C. Schubert, and I. Christ. Overcoming semantic heterogeneity in spatial data infrastructures. *Computers & Geosciences*, 35(4):739 – 752, 2009. Geoscience Knowledge Representation in Cyberinfrastructure.

[71] M. Lutz, J. Sprado, E. Klien, C. Schubert, and I. Christ. Overcoming semantic heterogeneity in spatial data infrastructures. *Computers & Geosciences*, 35(4):739 – 752, 2009.

[72] Xiaogang Ma, Emmanuel John M. Carranza, Chonglong Wu, Freek D. van der Meer, and Gang Liu. A skos-based multilingual thesaurus of geological time scale for interoperability of online geological maps. *Computers & Geosciences*, 37(10):1602 – 1615, 2011.

[73] Xiaogang Ma, Chonglong Wu, Emmanuel John M. Carranza, Ernst M. Schetselaar, Freek D. van der Meer, Gang Liu, Xinqing Wang, and Xialin Zhang. Development of a controlled vocabulary for semantic interoperability of mineral exploration geodata for mining projects. *Computers & Geosciences*, 36(12):1512 – 1522, 2010.

[74] Xiaogang Ma, Jin Guang Zheng, Justin C. Goldstein, Stephan Zednik, Linyun Fu, Brian Duggan, Steven M. Aulenbach, Patrick West, Curt Tilmes, and Peter Fox. Ontology engineering in provenance enablement for the national climate assessment. *Environmental Modelling & Software*, 61:191 – 205, 2014.

[75] Nicolas Maisonneuve, Matthias Stevens, Maria E. Niessen, and Luc Steels. Noisetube: Measuring and mapping noise pollution with mobile phones. In Ioannis N. Athanasiadis, Andrea E. Rizzoli, Pericles A. Mitkas, and Jorge Marx Gómez, editors, *Information Technologies in Environmental Engineering*, pages 215–228, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[76] Frank Manola and Eric Miller. Rdf primer. W3c recommendation., World Wide Web Consortium., 2004.

[77] Seth J.K. Mason, Sean B. Cleveland, Pol Llovet, Clemente Izurieta, and Geoffrey C. Poole. A centralized tool for managing, archiving, and serving point-in-time data in ecological research laboratories. *Environmental Modelling & Software*, 51:59 – 69, 2014.

[78] Tom Narock and Hayden Wimmer. Linked data scientometrics in semantic e-science. *Computers & Geosciences*, 100:87 – 93, 2017.

[79] S. Nativi, M. Craglia, and J. Pearlman. Earth science infrastructures interoperability: The brokering approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1118–1129, June 2013.

[80] Stefano Nativi, Paolo Mazzetti, Mattia Santoro, Fabrizio Papeschi, Max Craglia, and Osamu Ochiai. Big data challenges in building the global earth observation system of systems. *Environmental Modelling & Software*, 68:1 – 26, 2015.

[81] Douglas Nebert, Uwe Voges, and Lorenzo Bigagli. Ogc catalogue services 3.0 - general model. Ogc implementation standard., Open Geospatial Consortium Inc., 2016.

[82] Douglas D. Nebert. *Developing Spatial Data Infrastructures: The SDI Cookbook*. Global Spatial Data Infrastructure, 2004.

[83] Regina Obe and Leo Hsu. *PostGIS in action*. Simon and Schuster, 2021.

[84] OGC. Topic 6: Schema for coverage geometry and functions. version 7.0.0. The opengis abstract specification., Open Geospatial Consortium Inc., 2006.

[85] Kostas Patroumpas, Giorgos Giannopoulos, and Spiros Athanasiou. Towards geospatial semantic data management: strengths, weaknesses, and challenges ahead. In Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan, editors, *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, pages 301–310. ACM, 2014.

[86] Matthew Perry and John Herring. Ogc geosparql - a geographic query language for rdf data. Ogc implementation standard., Open Geospatial Consortium Inc., 2012.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[87] Clemens Portele. Opengis geography markup language (gml) encoding standard. version 3.2.2. Opengis standard., Open Geospatial Consortium Inc., 2016.

[88] Eric Prud'hommeaux and Gavin Carothers. Rdf 1.1 turtle: Terse rdf triple language. W3c recommendation., World Wide Web Consortium., 2014.

[89] Robert G. Raskin and Michael J. Pan. Knowledge representation in the semantic web for earth and environmental terminology (sweet). *Computers & Geosciences*, 31(9):1119 – 1125, 2005. Application of XML in the Geosciences.

[90] Manuel A. Regueiro, José R.R. Viqueira, Christoph Stasch, and José A. Taboada. Semantic mediation of observation datasets through sensor observation services. *Future Generation Computer Systems*, 67:47 – 56, 2017.

[91] Manuel A. Regueiro, José R.R. Viqueira, José A. Taboada, and José M. Cotos. Virtual integration of sensor observation data. *Computers & Geosciences*, 81:12 – 19, 2015.

[92] Camilo Restrepo-Estrada, Sidgley Camargo de Andrade, Narumi Abe, Maria Clara Fava, Eduardo Mario Mendiondo, and João Porto de Albuquerque. Geo-social media as a proxy for hydrometeorological data for streamflow estimation and to improve flood monitoring. *Computers & Geosciences*, 111:148 – 158, 2018.

[93] Philippe Rigaux, Michel O. Scholl, and Agnes Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[94] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

[95] Jon Sege, Marwan Ghanem, Waseem Ahmad, Hanadi Bader, and Yoram Rubin. Distributed data collection and web-based integration for more efficient and informative groundwater pollution risk assessment. *Environmental Modelling & Software*, 100:278–290, 2018.

[96] Kristin Stock, Tim Stojanovic, Femke Reitsma, Yang Ou, Mohamed Bishr, Jens Ortmann, and Anne Robertson. To ontologise or not to ontologise: An information model for a geospatial knowledge infrastructure. *Computers & Geosciences*, 45:98 – 108, 2012.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

[97] Sebastián Villarroya, José R. R. Viqueira, Manuel A. Regueiro, José A. Taboada, and José M. Cotos. Soda: A framework for spatial observation data analysis. *Distributed and Parallel Databases*, 34(1):65–99, Mar 2016.

[98] José Ramon Rios Viqueira, Nikos A. Lorentzos, and Nieves R. Brisaboa. Survey on spatial data modelling approaches. In Yannis Manolopoulos, Apostolos Papadopoulos, and Michael Vassilakopoulos, editors, *Spatial Databases: Technologies, Techniques and Trends*, pages 1–22. Idea Group, 2005.

[99] José R. R. Viqueira, Sebastián Villarroya, David Mera, and José A. Taboada. Smart environmental data infrastructures: Bridging the gap between earth sciences and citizens. *Applied Sciences*, 10(3), 2020.

[100] Panagiotis A. Vretanos. Opengis web feature service 2.0 interface standard. Ogc implementation standard., Open Geospatial Consortium Inc., 2010.

[101] Panagiotis A. Vretanos. Ogc filter encoding 2.0 encoding standard. Ogc implementation standard., Open Geospatial Consortium Inc., 2014.

[102] John Wainwright and Mark Mulligan. *Environmental Modelling: Finding Simplicity in Complexity*. John Wiley & Sons, Ltd., 2013.

[103] Chengbin Wang, Xiaogang Ma, and Jianguo Chen. Ontology-driven data integration and visualization for exploring regional geologic time and paleontological information. *Computers & Geosciences*, 115:12 – 19, 2018.

[104] Ruo-Qian Wang, Huina Mao, Yuan Wang, Chris Rae, and Wesley Shaw. Hyper-resolution monitoring of urban flooding with social media and crowdsourcing data. *Computers & Geosciences*, 111:139 – 147, 2018.

[105] Peng Yue, Liping Di, Wenli Yang, Genong Yu, and Peisheng Zhao. Semantics-based automatic composition of geospatial web service chains. *Computers & Geosciences*, 33(5):649 – 665, 2007.

[106] Peisheng Zhao, Liping Di, Genong Yu, Peng Yue, Yaxing Wei, and Wenli Yang. Semantic web-based geospatial knowledge transformation. *Computers & Geosciences*, 35(4):798 – 808, 2009.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Figures

Fig. 1.1 Illustration of an application example. 3

Fig. 2.1 Examples of Environmental Data Sources. 10
Fig. 2.2 Examples of Features and Coverages. 14
Fig. 2.3 Classes of geometric objects defined in the Simple Feature Access (SFA) OGC Standard. 18
Fig. 2.4 Architecture of main services provided by Spatial Data Infrastructures (SDI). 22
Fig. 2.5 Illustration of slicing and trimming operations on a *Raster Coverage*. 23
Fig. 2.6 Example of RDF Graph. 27

Fig. 3.1 Example of a GeoSPARQL query that combines vector features with a raster coverage. 40
Fig. 3.2 Query Plan and Optimization. 43

Fig. 4.1 Extended SPARQL Query Engine Architecture 50

Fig. 5.1 Direct mapping from geospatial features to RDF. 57
Fig. 5.2 Example of Relational Database to RDF Mapping Language (R2RML). . . . 58
Fig. 5.3 Illustration of geospatial raster coverage to RDF mappings. 60

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Fig. 5.4 Example of Coverage to RDF Mapping Language (C2RML) document. . . . 62

Fig. 6.1 Illustration of the GeoLD WFS operator. 67
Fig. 6.2 Illustration of the GeoLD WCPS operator. 71
Fig. 6.3 Ordering of coverage cells according to the geospatial horizontal and vertical sub-dimensions. 72
Fig. 6.4 Result of the tagging of the query plan of Figure 3.2 76
Fig. 6.5 Illustration of a tagging phase that requires the classification of triple patterns of a BGP. 77
Fig. 6.6 Optimized query plan generated for the tagged query plan of Figure 6.4 . . . 79
Fig. 6.7 Optimized query plan generated for the tagged query plan of Figure 6.5 . . . 80
Fig. 6.8 Response times for temporal queries over spatio-temporal coverages of growing temporal dimension 84
Fig. 6.9 Response times for spatial queries over spatio-temporal coverages of growing temporal dimension 85
Fig. 6.10 Response times of field range queries over spatio-temporal coverages of growing temporal dimension 86
Fig. 6.11 Response times for spatial range queries over spatial coverages of growing spatial dimension 87

Fig. 7.1 Example of aggregation of raster coverages over vector features. 91
Fig. 7.2 Implementation of the new VRJOIN operator. 93
Fig. 7.3 Optimized query plan with VRGROUP operation. 98
Fig. 7.4 Illustration of a Quadtree and calculation of Quadtree maxium level from WCPS service Minimum Request Size (MRS). 100
Fig. 7.5 Illustration of the quadtree structure used in the implementation of the VRGROUP operator. 102
Fig. 7.6 Datasets used for the evaluation of the VRJOIN and VRGROUP operators. . 108
Fig. 7.7 Determination of the best MRS for the implementation of the VRGROUP opertor. 109
Fig. 7.8 Comparison of the performance of the three implemented approaches for the group and aggregation of coverage data over vector features. 111

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

List of Tables

Tab. 6.1 Notation used in the data source tagging algorithm pseudocode 74

Tab. 7.1 Examples of decomposable and self-decomposable aggregate functions . . . 95

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

This Thesis proposes advances in various components of a new GeoSPARQL query engine, called GeoLD, that enables the efficient integrated querying of heterogeneous environmental datasets, including Vector Features and Raster Coverages, which are available through standard web services of geospatial data infrastructures. The proposed solution enables the access to Raster Coverages by providing a new Coverage to RDF Mapping Language (C2RML), which enables the programmer to incorporate specific vocabularies during the definition of the mapping between the coverage data schema and RDF. New SPARQL operators and a new query optimization strategy provide the algorithms required to reach query response times orders of magnitude faster than those of state of the art technologies. Additionally, contrary to existing approaches, raster data querying is achieved without the need to use large lists of specific raster manipulation functions, which simplifies the programming task