The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 3, pp.121-127, Jun. 30, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.3.121
JIIBC 2017-3-14

다차원 데이터를 위한 시멘틱 웹 연구

A Study on Semantic Web for Multi-dimensional Data

김정준*

Jeong-Joon Kim*

요 약 최근 공간 데이터와 같은 2차원 데이터를 위한 Semantic Web에 대한 연구가 활발하게 진행되고 있다. 2차원 Semantic Web은 기존의 Geospatial Web과 Semantic Web이 접목되어 다양한 지리 공간 정보와 일반 웹 상의 방대한 비공간 정보를 효율적으로 연계 및 통합하여 제공할 수 있는 지능적인 지리정보 웹 서비스 기술이다. 하지만 다차원 데이터 처리를 위한 연구는 전체적으로 부족한 편이며, 관련 표준 역시 제정되어 있지 않다. 따라서 본 논문에서는 그동안 진행되었던 Ontology 처리 기술과 관련된 다양한 기반 이론 및 기술들을 적용하여 다차원 데이터 처리가 가능한 온톨로지, 질의, 추론에 대한 내용을 제안하였다. 또한 각각 제안한 내용을 다차원 질의가 필요한 가상 시나리오에 적용해보았다.

Abstract Recently, it has been actively Semantic Web studies for 2-dimensional data of the spatial data. 2-dimensional Semantic Web, are fused existing Geospatial Web and the Semantic Web, and integrate with the efficient cooperation of the vast non-spatial information on a variety of geospatial information and general Web, it is possible to provide it is a Web services technology of intelligent geographic information. However, in the research for multi-dimensional data processing, and in those who are missing overall, relevant standards also not been enacted. Therefore, in this paper, by applying a variety of base of the theory and technology related to this to take place the Ontology processing technology, multi-dimensional data processing is possible ontology, question, and suggested the contents of the reasoning. Also, we tried to apply what you have proposed respectively to the multi-dimensional query virtual scenario necessary.

Key Words : Semantic Web, Multi-dimensional Data, Ontology

Ⅰ. 서 론

Semantic Web 은 RDF, OWL과 같은 온톨로지 언어들을 사용하여 의미기반의 데이터 분석을 함으로써 기존의 웹이 갖는 한계점을 극복하였으며, 차세대 웹의 표준으로 인식되고 있다. Semantic Web은 별도의 웹이 아니라 현재 웹의 확장이며, 컴퓨터와 사람들이 협력 작업을 할 수 있도록 의미가 잘 정의된 웹이다[1,2]. 그리고 더 나아가 웹의 발달로 인해 점차 방대해지는 웹의 2차원 공간 정보를 효율적으로 연계 및 통합하기 위한 연구가 진행되면서, Geo Semantic Web이 제시되었다[3,4]. Geo Semantic Web은 기존의 Geospatial Web과 Semantic Web이 접목되어 다양한 지리 공간 정보와 일반 웹 상의 방대한 비공간 정보를 효율적으로 연계 및 통합하여 제

*정회원, 한국산업기술대학교 컴퓨터공학과
접수일자: 2016년 12월 28일, 수정완료: 2017년 4월 17일
게재확정일자: 2017년 6월 9일

Received: 28 December, 2016 / Revised: 17 April, 2017 / Accepted: 9 June, 2017
*Corresponding Author: jjkim@kpu.ac.kr
Dept. of Computer Eng., Korea Polytechnic University, Korea

공할 수 있는 지능적인 지리정보 웹 서비스 기술이다. 이러한 Geo Semantic Web 표준 개발을 위해 OGC에서는 공간 질의 언어인 Geo SPARQL[1]에 대한 표준을 제안하였다[5,6]. 그리고 ISO 및 OGC 표준을 따르는 표현을 제공하고 다양한 공간 속성과 타입을 지원하기 위해서 Geospatial Vocabulary의 확장에 대한 필요성으로 GeoRSS, Geo OWL 표준을 제시하였다[7,8]. 하지만 현재 다차원 데이터 처리를 위한 연구는 전체적으로 부족한 편이며, 관련 표준 역시 제정되어 있지 않다[9,10]. 따라서 본 논문에서는 그동안 진행되었던 Ontology 처리 기술과 관련된 다양한 기반 이론 및 기술들을 적용하여 다차원 데이터 처리가 가능한 온톨로지, 질의, 추론에 대한 내용을 제안하였다. 또한 각각 제안한 내용을 다차원 질의가 필요한 가상 시나리오에 적용해보았다.

II. 관련 연구

1. Geo SPARQL

SPARQL은 2008년 W3C에서 제시한 Query Language로 분산되어 있는 Semantic Web의 데이터에 효과적으로 접근하고 탐색하기 위해 제안되었다.

이에GeoSPARQL은 2010년 OGC에서 제시한 표준 Geometry 타입과 이를 처리하기 위한 표준 공간 연산자에 대한W3C에서 제시된 Semantic Web 아키텍처의 표준 질의 언어인 SPARQL을 확장하는 방안을 제시하고 있다. GeoSPARQL은 SPARQL과 마찬가지로 RDF 트리플에 대한 패턴을 기술함으로써 원하는 RDF 트리플 정보를 검색할 수 있으며, relate 연산자를 제외한 공간 관계 연산자의 경우에는 기존 SPARQL의 술어를 확장한 Spatial Predicate로 지원한다. 그리고 공간 관계 연산자 중 하나인 relate와 공간 분석 연산자의 경우에는 기존 SPARQL의 Filter Function을 확장하여 지원한다.

다음 Table 1은 GeoSPARQL에서 지원하는 연산자의 spatial Predicate와 Filter Function을 보여준다.

표 1. GeoSPARQL에서 지원하는 연산자
Table 1. Operators of GeoSPARQL

Spatial Predicate	Filter Function
ogc:equals	ogc:relate
ogc:disjoint	ogc:distance
ogc:intersects	ogc:buffer
ogc:touches	ogc:convexHull
ogc:crosses	ogc:intersection
ogc:within	ogc:union
ogc:contains	ogc:difference
ogc:overlaps	

표 1에서Spatial Predicate에는 8개의 공간 관계 연산자를 지원하며, 결과로써 TRUE, FALSE를 반환한다. Filter Function은 1개와 공간 관계 연산자와 6개의 공간 분석 연산자를 지원한다.

다음은 Overlap 연산자를 사용한 GeoSPARQL질의 예를 보여준다.

```
SELECT ?Block1 ?Block2
WHERE{
?Block1 rdf:type my:CommercialParcel
?Block2 rdf:type my:Street
?Block1 rdf:overlap ?Block2}
```

Block1의 타입은 상업 구역이고, Block2의 타입은 도로일 때 block1과 block2에 대해 Overlap 질의를 하여 상업 구역과 도로가 겹치는 경우 그에 따른 각각의 상업 구역과 도로를 반환하는 질의가 가능하다.

2. Spatial SWRL

현재 Spatial SWRL 관련 표준 제정은 없는 상태이다. 그러나 2011년 프랑스에서 열린 International Conference on Geospatial Semantics의 Integration of Spatial processing and knowledge Processing through the Semantic Web Stack 에서 Spatial SWRL 관련 내용을 살펴 볼 수 있다. 이 논문에서 제안한 Spatial SWRL은 8개의 Spatial SWRL built-in을 제안하였다. 이 8개의 built-in은 4개의 geoprocessing 기능과 나머지 4개의 georelationship 기능으로 이루어진다. 이 built-in요소는 데이터베이스에서 사용할 수 있게 하기 위해 SQL문을 사용한다. 그 후에 SQL문을 Translation engine에 의해 처리한 뒤 Translated Built-ins를 얻게 된다. 두 처리 과정 모두 공간 데이터베이스에 질의하기 위해서 SQL 문을 사용한다.

2.1 Geoprocessing Built-ins

Geoprocessing Built-ins은 두 가지 기능을 가진다. 첫 번째 기능으로는 geometry 기능을 반환하고 두번째 기능은 feat:Feature, sa:hasSpatialRelations, sa:sptialOperation sequence를 통해 온톨로지를 조정한다.

표 2. 지오처리 빌트인
Table 2. GeoProcessing built-ins

Functions	Class	Object Property	Data Property	Built-ins
Buffer	sa:sp_Buffer	sa:hasBuffer	sa:hasBufferDistance	Buffer(?x, b, ?y)
Union	sa:sp_Union	sa:hasUnion	-	Union(?x, ?y1, y2)
Intersection	sa:sp_Intersection	sa:hasIntersection	-	Intersection(?x, ?y1, y2)
Difference	sa:sp_Difference	sa:hasDifference	-	Differerence(?x, ?y1, y2)

Spatial built-in을 기본 SWRL 구문으로 바꾸기 위한 과정은 다음과 같다.

```
feat:Feature(?x) ^ Buffer(?x, b, ?y) ---(1)
```

```
SELECTBuffer(geom::Feature, bufferDistance --(2)
```

데이터베이스에서 built-in 요소를 쓰기 위해 SQL문을 사용한다. (2)에서 쓰인 geom은 feat:Feature의 geometry object이다. geometry는 특정한 bufferDistance를 가진 각 object의 buffer를 가리킨다.

그 후에 Spatial built-in 문을 SWRL문으로 translation한 것은 다음과 같다.

```
feat:Feature(?x)^sa:hasbuffer(?x,
?y)^sa:sp_buffer(?y)^sa:hasBufferDistance(?y, ?b) ---(3)
```

sa:sp_Buffer class는 객체를 인스턴스화 시키고 bufferDistance와 bufferGeometry 를 저장한다. 따라서 Spatial built-in을 feat:Feature, sa:hasSpatialRelations, sa:spatial Operation sequence형태의 SWRL문으로 translation되었다.

표 3. 지오처리를 위한 SQL 문장 실행
Table 3. SQL statements executions of geoprocessing built-ins for the spatial enrichment

Built-ins	SQL statements	Translated Built-ins
Swrlbspatial:Buffer(?x, b, ?y)	SELECT Buffer(geom::Feature, bufferDistance) Result: Populated in the knowledge base as individuals of class sa:sp_Buffer.	sa:hasBuffer(?x,?y) ^sa:p_Buffer(?y) ^ sa:hasBufferDistance(?y, b)
Swrlbspatial:Union(?x, ?y1, ?y2)	Select Union(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Union.	sa:sp_Union (?x) ^ sa:hasUnion(?x, ?y1) ^ sa:hasUnion(?x, ?y2)
Swrlbspatial:Intersection(?x, y1, ?y2)	Select Intersection(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Intersection.)	sa:sp_Intersection(?x) ^ sa:hasIntersection(?x, ?y1) ^ sa:hasIntersection(?x, ?y2)
Swrlbspatial:Difference(?x, y1, ?y2)	Select Difference(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Difference.	sa:sp_Difference(?x) ^ sa:hasDifference(?x, ?y1) ^ sa:hasDifference(?x,?y2)

2.2 Georelationship Built-ins

Georelationship built-ins는 object property에만 의존하기 때문에 Geoprocessing built-ins보다 훨씬 간단하다.

표 4. 지오관계
Table 4. Georelationship built-ins

Functions	Class	Object Property	Built-ins
Disjoint	-	sa:hasDisjoint	Disjoint(?x,?y)
Touches	-	sa:hasTouches	Touches(?x,?y)
Within	-	sa:hasWithin	Within(?x,?y)
Overlaps	-	sa:hasOverlaps	Overlaps(?x,?y)

Spatial built-in을 기본 SWRL 구문으로 바꾸기 위한 과정은 다음과 같다.

```
feat:Feature(?x ^ feat:Feature(?y) ^ Touch(?x, ?y) --(4)
```

Touch연산자는 x변수와 y변수가 인접한 지역을 Boolean 형태로 값을 반환해준다. 그러나 SQL문을 사용하여 파라미터 값을 사용하면 더 구체적인 결과를 낼 수 있다.

```
SELECT Touch(geom::Feature1, geom::Feature2) --(5)
```

이SQL문은 feature1 geometry와 feature2 geometry가 인접한 여부를 ture 값과 false 값으로 반환해준다.

```
select Feature2
From spTable
WHERE Touch(geom::Feature1, geom::Feature2) -- (6)
```

이 SQL문은 spTable로부터 feature1에 인접한 feature2 를 반환한다.

여기서 spTable이란 geometry를 데이터베이스 시스템에 저장하고 공간적으로 주석된 것을 의미한다.

```
feat:Feature(?x) ^ hasTouch(?x, ?y) ^ feat:Feature(?y) ---(7)
```

마지막 단계로 Touch(?x, ?y)연산자를 SWRL문을 얻기 위해 feat:Feature, sa:hasSpatialRelations, feat:feature sequence 형태로 나눠진다.

표 5. 지오관계를 위한 SQL 문장 실행
Table 5. SQL statements executions of georelationship built-ins for the spatial enrichment

Built-ins	SQL statements	Translated Built-ins
swrlbspatial:Disjoint(?x, ?y)	SELECT Feature2 FROM spTable WHERE Disjoint(geom::Feature1, geom::Feature2)	sa:hasDisjoint(?x, ?y)
swrlbspatial:Touches(?x,?y)	SELECT Feature2 FROM spTable WHERE Touch(geom::Feature1, geom::Feature2)	sa:hasTouch(?x, ?y)
swrlbspatial:Within(?x, ?y)	SELECT Feature2 FROM spTable WHERE Within(geom::Feature1, geom::Feature2)	sa:hasWithin(?x, ?y)
swrlbspatial:Overlaps(?x,?y)	SELECT Feature2 FROM spTable WHERE Overlap(geom::Feature1, geom::Feature2)	sa:hasOverlaps(?x, ?y)

Ⅲ. 시스템 구현 및 시나리오

1. Operator for Query

다차원 질의를 위해 연산자를 제안하였다. 제안된 연산자는 시간을 포함하는 다차원 질의가 가능하다.

표 6. 관계 연산자
Table 6. Relation Operators

ST_Relation Operators	Explanation
ST_Equlas(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B가 같은지 여부 반환
ST_Disjoint(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B가 만나지 않는지 여부 반환
ST_Intersects(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B가 만나는지 여부 반환
ST_Touches(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B의 경계가 만나는지 여부 반환
ST_Crosses(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B가 교차하는지 여부 반환
ST_Within(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 B가 A를 포함하는지 여부 반환
ST_Contains(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A가 B를 포함하는지 여부 반환
ST_Overlaps(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B가 겹치는지 여부 반환

표 6에서 관계 연산자로 Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps 로 8개를 정의하였다.

이와 같은 관계 연산자를 통한 ST_FILTER의 적용으로 통합 질의가 가능하다. 다음은 본 시스템에서 지원하는 syntax를 보여준다.

```
SELECT [ATTRIBUTE]
WHERE
{ST_Geometry A rdfs:type ST_Geometry
ST_Geometry B rdfs:type ST_Geometry
ST_FILTER [ST_Relation Operator (ST_Geometry A,
ST_Geometry B) ]}
```

다음은 ST_Relation Operator의 Within연산자를 이용한 질의 예시이다.

```
SELECT *
WHERE
{vin:CoteDOrRegion rdfs:type vin:Region
vin:ClosDeVougeot rdfs:type vin:Winery
ST_FILTER [ST_Within (vin:CoteDOrRegion, vin:ClosDeVougeot)]}
```

이 예제는 vin:CoteDOrRegion 지역과 vin:ClosDeVougeot의 관계를 입증하기 위해 ST_Relation Operator의 Within연산자를 사용한다. vin:CoteDOrRegion 지역에 vin:ClosDeVougeot winery가 포함되는지 여부를 ST_FILTER를 사용하여 각 시간 단위 마다 그 결과를 result로 반환 가능하다.

표 7. 시간 연산자
Table 7. Time Operators

ST_Analysis Operators	Explanation
ST_Intersection(ST_Geometry A, ST_Geometry B	각 시간 단위 마다 객체 A와 B의 교집합을 반환
ST_Union(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B의 합집합을 반환
ST_Difference(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B의 차집합을 반환
ST_Distance(ST_Geometry A, ST_Geometry B)	각 시간 단위 마다 객체 A와 B의 거리를 반환

표 7 의 시간 연산자로는 Intersection, Union, Difference, Distance 로 4개를 정의하였다.

다음은 본 시스템에서 지원하는 syntax를 보여준다.

```
SELECT [ATTRIBUTE]
WHERE
{ST_Geometry A rdfs:type ST_Geometry
ST_Geometry B rdfs:type ST_Geometry
ST_FILTER [ST_Analysis Operator (ST_Geometry A,
ST_Geometry B) ]}
```

다음은 ST_ Analysis Operator의 intersection연산자를 이용한 질의 예시이다.

```
SELECT ?region
WHERE
{vin:CoteDOrRegion rdfs:type vin:Region
?region rdfs:type vin:Region
ST_FILTER [intersection (vin:CoteDOrRegion, ?region)]}
```

이 예제는 vin:CoteDOrRegion 지역과 다른 지역간의 관계를 입증하기 위해 ST_Analysis Operator의 intersection 연산자를 사용한다. vin:CoteDOrRegion 지역과 다른 지역의 교집합을 ST_FILTER를 사용하여 각 시간마다 그 결과를 result로 반환 가능하다.

2. Inference Rule

공간의 독립적 추론이 아닌 통합적인 추론을 고려하였다. 따라서 본 논문에서는 다차원 통합 추론을 위해 OpenGIS “Simple Features Specification for SQL”에서 명시하는 공간 연산자를 확장한 다차원 연산자들을 각각 관련연구 Geoprocessing Built-ins와 Georelationship Built-ins 추론알고리즘에 적용하여 다차원 통합 연산자

를 제안하였다.

표 8. 다차원 연산자
Table 8. Multi-dimensional Operator

ST_Relationship Built-ins	SQL statements	Translated Built-ins
ST_Disjoint(?ST_GeometryA,?ST_Geometry B)	SELECT Feature_2 FROM stTable WHERE Disjoint(ST_Geometry::Feature1, ST_Geometry::Feature2)	st:hasDisjoint(?ST_GeometryA, ?ST_GeometryB)
ST_Touches(?ST_GeometryA,?ST_Geometry B)	SELECT Feature_2 FROM stTable WHERE Touches(ST_Geometry::Feature1, ST_Geometry::Feature2)	st:hasTouch(?ST_GeometryA, ?ST_GeometryB)
ST_Within(?ST_GeometryA,?ST_Geometry B)	SELECT Feature_2 FROM stTable WHERE Within(ST_Geometry::Feature1, ST_Geometry::Feature2)	st:hasWithin(?ST_GeometryA, ?ST_GeometryB)
ST_Overlaps(?ST_GeometryA,?ST_Geometry B)	SELECT Feature_2 FROM stTable WHERE Overlaps(ST_Geometry::Feature1, ST_Geometry::Feature2)	st:hasOverlap(?ST_GeometryA, ?ST_GeometryB)
ST_Intersects(?ST_GeometryA,?ST_Geometry B)	SELECT Feature_2 FROM stTable WHERE Intersects(ST_Geometry::Feature1, ST_Geometry::Feature2)	st:has Intersects(?ST_GeometryA, ?ST_GeometryB)

표 8은 본 논문에서 제안하는 연산자를 관련연구 Georelationship Built-ins 추론알고리즘에 적용하여 새로운 다차원 연산자를 제안한 것이다. 기존 관련연구에서는 4개의 공간 관계 연산자를 제안하였으나 본 논문에서는 ST_Intersect 연산자를 추가하여 총 5개의 연산자를 제안한다.

3. Scenario

본 논문에서 제안한 다차원 데이터 처리를 검증하기 위하여 다음과 같은 시나리오를 구상하였다.

"대한민국에서 지진이 일어나 많은 사상자가 발생하였다. 현재시간에 진료 가능한 병원을 표시하라."

이와 같은 질의를 수행했을 때 해당 지역에 대한 시공간 데이터와 병원에 대한 시공간 데이터를 이용하여 현재 위치와 가장 가까운 병원을 표시할 수 있다. 그리고 현재 시간에 진료 가능한 병원 정보 역시 확인할 수 있다. 가상 시나리오 적용을 위해 지진, 병원에 관한 시공간 온톨로지를 구성하였다.

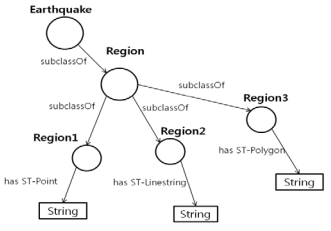

그림 1 . 지진 온톨로지
Fig. 1. Earthquake Ontology

그림 1을 보면 ontology 타이틀로는 지진을 온톨로지로 나타내었으며 Region이라는 클래스안에 지진 피해 지역인 Region1, Region2, Region3 로 피해지역 3개가 포함되어 있는 것을 알 수 있다. 지진이 일어난 지역을 공간 타입으로 설정하였다. 또한 지진이 일어난 시간을 시간 타입으로 설정하여 시공간 타입인 ST-point, ST-linestring, ST-polygon 3가지 타입을 이용하여 나타내었다.

```
…
<owl:Ontology rdf:about="">
<rdfs:comment>
        Earthquake OWL   Ontology
</rdfs:comment>
    </owl:Ontology>

<owl:Class rdf:ID="Earthquake">
        <rdfs:subClassOf   rdf:resource="#Region"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#occur"/>
                <owl:allValuesFrom rdf:resource="#Region1"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

<owl:ObjectProperty rdf:ID="occur">
        <rdfs:subPropertyOf   rdf:resource="#part"/>
        <rdfs:domain   rdf:resource="#Earthquake"/>
        <rdfs:range   rdf:resource="#Region"/>
    </owl:ObjectProperty>

 <owl:DatatypeProperty   rdf:ID="size">
        <rdfs:domain   rdf:resource="#Earthquake"/>
        <rdfs:range   rdf:resource="&xsd:Integer"/>
    </owl:DatatypeProperty>

 <owl:DatatypeProperty   rdf:ID="time">
        <rdfs:domain   rdf:resource="#Earthquake"/>
        <rdfs:range   rdf:resource="&xsd:datetime"/>
    </owl:DatatypeProperty>

<owl:Class rdf:ID="Region1">
   <geometry>
     <ST_point>
       <time>
        <instant>
         2011-12-20 11:20:12
        </instant>
       </time>
       <point>
         37.497942 127.027621
       </point>
     </ST_point>
   </geometry>
</owl:Class>

<owl:Class rdf:ID="Region2">
   <geometry>
     <ST_Linestring>
       <time>
        <instant>
         2011-12-20 11:28:12
        </instant>
       </time>
       < linestring>
         37.449021 127.147006,   37.412041 127.254981
       </linestring>
     </ST_ Linestring >
   </geometry>
</owl:Class>

<owl:Class rdf:ID="Region3">
   <geometry>
     <ST_polygon>
       <time>
        <instant>
         2011-12-20 11:30:15
        </instant>
       </time>
       <polygon>
         37.394253 126.956821,   37.424707 126.908569
37.478488 126.864289
       </polygon>
     </ST_polygon>
   </geometry>
</owl:Class>
…
```

위의 예는 ST-OWL의 지진 ontology의 부분적인 예시를 보여준다. ST-OWL은 ST-Geometry type을 적용하여 표현하였으며 시간과 공간을 함께 표현하여 시간과 공간을 통합한 OWL이라 할 수 있다. ST-OWL의 예로는 Region1은 2011년 12월 20일 11:20:12에 서울특별시 강남구에서 일어난 지진지역을 point geometry를 적용하

여 위도, 경도로 나타내었다. Region2는 2011년 12월 20일 11:28:12에 경기도 성남시와 광주시에서 일어난 지진지역을 linestring geometry를 적용하여 위도, 경도로 나타내었다. Region3은 2011년 12월 20일 11:30:15에 경기도 안양시, 광명시, 성남시에서 일어난 지진지역을 polygon geometry를 적용하여 위도, 경도로 나타낸 것을 보여준다.

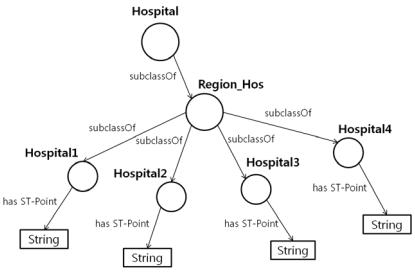

그림 2. 병원 온톨로지
Fig 2. Hospital Ontology

그림 2를 보면 ontology 타이틀로는 병원으로 나타내었으며 Region_Hos라는 클래스안에 병원으로 Hospital1, Hospital2, Hospital3, Hospital4 로 4개가 포함되어 있는 것을 알 수 있다. 병원 위치를 공간 타입으로 설정, 병원의 진료 시간을 시간 타입으로 설정하여 시공간 타입인 ST-point타입을 이용하여 나타내었다.

```
…
<owl:Ontology rdf:about="">
<rdfs:comment>
        Hospital OWL Ontology
</rdfs:comment>
    </owl:Ontology>

<owl:Class rdf:ID="Hospital">
        <rdfs:subClassOf   rdf:resource="#Region_Hos"/>
          <rdfs:subClassOf>
              <owl:Restriction>
                  <owl:onProperty    rdf:resource="#location"/>
                  <owl:allValuesFrom    rdf:resource="#Region1"/>
              </owl:Restriction>
          </rdfs:subClassOf>
    </owl:Class>

<owl:ObjectProperty    rdf:ID="location">
          <rdfs:subPropertyOf rdf:resource="#part"/>
          <rdfs:domain rdf:resource="#Hospital"/>
          <rdfs:range rdf:resource="#Region_Hos"/>
    </owl:ObjectProperty>

 <owl:DatatypeProperty    rdf:ID="point">
          <rdfs:domain rdf:resource="#Hospital"/>
          <rdfs:range rdf:resource="&xsd;Integer"/>
    </owl:DatatypeProperty>

 <owl:DatatypeProperty    rdf:ID="time">
          <rdfs:domain    rdf:resource="#Hospital"/>
          <rdfs:range rdf:resource="&xsd;datetime"/>
    </owl:DatatypeProperty>

<owl:Class rdf:ID="Hospital1">
    <geometry>
      <ST_point>
        <time>
          <period>
            2000-12-11 09:00:00, now
          </period>
        </time>
        <point>
          37.35277 127.123264
        </point>
      </ST_point>
    </geometry>
</owl:Class>
…
```

위의 예는 ST-OWL의 병원 ontology의 부분적인 예시를 보여준다. ST-OWL Hospital1은 2000년 12월 11일 09:00:00에 개원하여 현재까지 즉 , now로 진료 시간을 설정하였으며 위치는 경기도 성남시 분당구에 위치한 서울대병원으로 point geometry를 적용하여 위도, 경도로 나타낸 것을 보여준다.

```
SELECT   ?Hospital
WHERE
{
?Region   rdfs:type   Earthquake:Region
?Region   Earthquake:Region   ?Region1
?Region_Hos   rdfs:type   Hospital:Region_Hos
ST_FILTER  [ST_intersection  (Region1  ,    Hospital:Region_Hos)]
}
```

위와 같이 시공간 SPARQL에서 Earthquake:Region의 Type을 Region이라는 변수로 나타내었으며 Region 안에서 Region1을 새로운 변수로 지정하였다. 또한 Hospital:Region_Hos의 Type을 Region_Hos라는 변수로 설정하였다. Region1과Region_Hos의 교차하는 병원을 알기 위하여 ST_FILTER의 분석연산자인 intersection을 사용하였음을 나타낸다. 하지만 시공간 SPARQL에서 단순히 지진피해지역과 병원이 교차하는 지역을 출력한다. 이 한계점을 극복하기 위하여 본 논문에서 제안한 시공간 통합 연산자를 이용하여 시공간 온톨로지의 질의 결과를 풍부하게 할 수 있으며 시나리오에 부합한 정보를 얻을 수 있다.

```
 ?x = ST_1(2011,2013, 37.449021 127.147006)
 ?y = ST_2(2011,2013, 37.412041    127.254981)
?z = ST_3(2000,2013, 37.35277, 127.123264)
?r = st:hasTouch(ST_1,ST_2,ST_3)
```

위의 표는 시공간 추론 변수 정의를 나타낸다. x변수와 y변수는 지진 ST-OWL에 Region2를 사용하였다. Region2는 2011년 12월 20일 11:28:12에 경기도 성남시와 광주시에서 일어난 지진지역을 linestring geometry를 적용하여 위도, 경도로 나타낸 것이다. z변수는 병원 ST-OWL의 병원 ontology의 부분적인 예시인 ST-OWL Hospital를 사용하였다. 이 병원은 2000년 12월 11일 에 개원하였고 위치는 경기도 성남시 분당구에 위치한 서울대병원으로 point geometry를 적용하여 위도, 경도로 나타내었다. r변수에는 본 논문에서 제안한 시공간 통합 관계 연산자 중 하나인 st:hasTouch를 사용하여 x,y,z변수로 나타내었다.

```
Region(?x)  ^  Region(?y)  ^Hospital(?z)  ^        st:hasTouch(x,  y,  z)  →  available_
hospital
```

위의 표는 추론 문법으로 지진피해지역을 다차원 연

산자 st:hasTouch를 이용하여 x 지진피해 지역과 y지진피해지역에 인접한 z병원지역을 나타내는 추론 규칙이다. 이 추론 규칙을 통하여 지진피해지역에서 인접한 병원을 찾는 질의의 결과를 풍부하게 할 수 있으며 본 시나리오에 부합한 정보를 얻을 수 있다는 이점이 있다.

V. 결 론

최근 Geo Semantic Web에 대한 연구가 활발하게 진행되고 있다. Geo Semantic Web은 기존의 Geospatial Web과 Semantic Web이 접목되어 다양한 2차원 지리 공간 정보를 효율적으로 연계 및 통합하여 제공할 수 있는 지능적인 지리정보 웹 서비스 기술이다. 하지만 다차원 데이터에 대한 처리 및 추론에 대한 연구는 부족한 편이며, 관련 표준 또한 제정되어 있지 않다. 또한 현재까지의 대부분 연구들은 시간과 공간이 분리된 독립적 추론만이 가능하다는 문제점이 있다. 따라서 본 논문에서는 그동안 진행되었던 Ontology 처리 기술과 이와 관련된 다양한 기반 이론 및 기술들을 적용하여 다차원 데이터 처리가 가능한 온톨로지, 질의, 추론에 대한 내용을 제안하였으며 시나리오를 통하여 검증하였다.

References

[1] GeoOWL Geospatial Ontology language Document Overview, W3C Recommendation (2006)
[2] SPARQL Query language Document Overview, W3C Recommendation (2008)
[3] GeoSPARQL Geospatial Query language Document Overview, OGC Candidate (2009)
[4] Matthew Perry and Prateek Jain and Amit P.Sheth, SPARQL-ST: Extending SPARQL to Support Spatiotemporal Queries, (2011)
[5] SWRL Rule Document Overview, W3C Recommendation (2004)
[6] Ashish Karmacharya, Christophe Cruz2, Frank Boochs, Franck Marzani, Use of Geospatial Analyses for Semantic Reasoning (2010)
[7] Ashish Karmacharya, Christophe Cruz, Frank Boochs, Franck Marzani, Integration of Spatial processing and knowledge Processing through the Semantic Web Stack (2011)
[8] Myung-Jae Lim, Myung-Gwon Kim, Ki-Young Lee, "WPAN Based Semantic-Web Health Monitoring," Journal of IIBC, Vol. 13, No. 6., 2013. pp. 167-172
[9] Yang-Hui Cho, Jae-Pyo Park, Seung-Min Yang, "Grid-Based Key Pre-Distribution for Factory Equipment Monitoring" Journal of the Institute of Internet, Broadcasting and Communication, Vol. 16, No. 6, pp. 147-152, Dec 2016.
DOI: https://doi.org/10.7236/JIIBC.2016.16.6.147
[10] Sun-Jin Oh, "Design of a Smart Application for Remote Diagnosis in Ubiquitous Computing Environment", Journal of the Institute of Internet, Broadcasting and Communication, Vol. 16, No. 4, pp. 82-87, Aug 2016.
DOI: https://doi.org/10.7236/JIIBC.2016.16.4.81

저자 소개

김 정 준(정회원)

- 2003년 2월 : 건국대학교 컴퓨터공학과 학사
- 2005년 2월 : 건국대학교 컴퓨터공학과 석사
- 2010년 8월 : 건국대학교 컴퓨터공학과 박사
- 2010년 9월 ~ 2012년 8월: 건국대학교 컴퓨터공학과 강의교수
- 2012년 9월 ~ 2016년 2월: 건국대학교 컴퓨터공학과 조교수
- 2016년 3월 ~ 현재: 한국산업기술대학교 컴퓨터공학과 조교수

<주관심분야 : Database Systems, BigData, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), etc.>

※ 이 성과는 2017년 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2017R1A2B4011243)