극대용량 서지 링크드 데이터 구축의 효율성을 위한 RDF 트리플 저장소 접근 최소화에 관한 연구

Research on Minimizing Access to RDF Triple Store for Efficiency in Constructing Massive Bibliographic Linked Data

이 문 호(Moon-Ho Lee)*
최 성 필(Sung-Pil Choi)**

〈목 차〉

초 록

본 논문에서는 세계 최대 규모의 생의학 분야 서지 데이터베이스인 MEDLINE 전체를 링크드 데이터로 변환·구축하는 효율적인 방안을 제시한다. 이를 위해서 우선 MEDLINE 레코드 구조를 세부적으로 분석하여 적합한 RDF 스키마를 도출하고 각 레코드를 도출된 스키마에 유효한 RDF 파일로 변환하는 과정을 거친다. 본 논문에서는 변환된 레코드 단위의 모든 RDF 파일을 병합하여 이를 단일 RDF 트리플 저장소에 저장할 때 주어 URI 중복 확인 절차를 효율화하는 이중 일괄 등록 방법을 적용한다. 이 방법을 통해서 RDF 파일 단위로 링크드 데이터를 순차적으로 구축하는 방법과 비교했을 때 주어 URI 중복 제거를 위한 RDF 트리플 저장소 접근 횟수가 26,597,850회에서 2,400회로 감소하는 결과를 가져왔다. 따라서 본 연구의 결과는 대용량 서지 레코드 집합을 링크드 데이터로 변환하는 과정에서의 비효율성을 제거하고 신속성과 시의성을 확보할 수 있는 중대한 계기를 제공할 것으로 기대한다.

키워드: MEDLINE, 링크드 데이터, RDF 스키마, RDF 트리플 저장소, 이중 일괄 등록

ABSTRACT

In this paper, we propose an effective method to convert and construct the MEDLINE, the world's largest biomedical bibliographic database, into linked data. To do this, we first derive the appropriate RDF schema by analyzing the MEDLINE record structure in detail, and convert each record into a valid RDF file in the derived schema. We apply the dual batch registration method to streamline the subject URI duplication checking procedure when merging all RDF files in the converted record unit and storing it in a single RDF triple storage. By applying this method, the number of RDF triple storage accesses for the subject URI duplication is reduced from 26,597,850 to 2,400, compared with the sequential configuration of linked data in units of RDF files. Therefore, it is expected that the result of this study will provide an important opportunity to eliminate the inefficiency in converting large volume bibliographic record sets into linked data, and to secure promptness and timeliness.

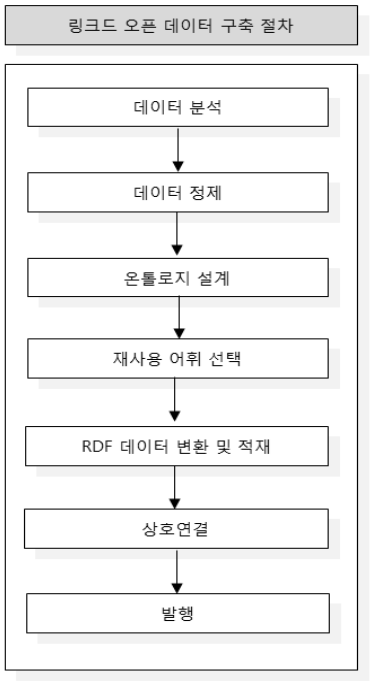
Keywords: MEDLINE, Linked data, RDF Schema, RDF triple store, Dual batch registration

* 경기대학교 대학원 문헌정보학과 박사과정(best.conv2@gmail.com) (제1저자)
** 경기대학교 문헌정보학과 조교수(spchoi@kyonggi.ac.kr) (교신저자)
•논문접수: 2017년 8월 20일 •최초심사: 2017년 8월 29일 •게재확정: 2017년 9월 19일
•한국도서관정보학회지 48(3), 233-257, 2017. [http://dx.doi.org/10.16981/kliss.48.201709.233]

Ⅰ. 서론

2006년 Berners-Lee가 제시한 링크드 데이터(Linked Data)의 4가지 원칙은 다음과 같다. 첫째, 모든 사물(things)의 이름을 URI(Uniform Resource Identifier)로 사용한다. 둘째, 사물의 이름을 찾을 수 있도록 HTTP URI를 사용한다. 셋째, 누군가가 RDF(Resource Description Framework)나 SPARQL(Simple Protocol And RDF Query Language) 같은 표준을 사용해 URI를 찾을 때 유용한 정보를 제공받는다. 넷째, 다른 URI에 대한 연결을 포함하므로 더 많은 사물을 발견할 수 있다(Berners-Lee 2006). 위와 같은 원칙을 바탕으로 링크드 데이터는 특정 자원과 자원 사이의 연결점(endpoint)을 제공하고 이러한 연결점들은 SPARQL 등과 같은 구조적 질의 언어를 통해 고수준의 검색 및 정보 접근을 실현할 수 있다.

한국정보문화진흥원(2014)은 우선 오픈 데이터(Open Data)를 "모든 사람이 누구나 자유롭게 데이터를 활용하고, 재설계, 재생산할 수 있는 데이터"로 정의했고, 링크드 오픈 데이터(Linked Open Data, LOD)를 "Linked Data(링크드 데이터)와 Open Data(오픈 데이터)의 핵심 개념을 포괄하는 개념으로 링크드 데이터 발행 원칙에 맞춰 데이터를 개방하는 것을 의미한다." 라고 했다. 한국정보문화진흥원(2014)은 우리나라의 공공 부문과 민간 부문에서 다



〈그림 1〉 링크드 오픈 데이터 구축 절차

수의 링크드 오픈 데이터 구축 사례를 보고하고 있다. RDF 변환 대상인 데이터 집합의 규모는 구축 사례별로 최소 약 12,000 건부터 약 610만 건까지 다양하다. 이 구축 사례들에 나온 링크드 오픈 데이터 구축 절차를 정리해 도식화하면 다음 <그림 1>과 같다.

<그림 1>의 RDF 데이터 변환 및 적재 부분은 원시 데이터 집합을 RDF 데이터로 변환한 후, RDF 트리플 저장소(RDF triple store)에 적재하는 과정이다. 적재 시간은 일반적으로 RDF 트리플 저장소에 저장된 트리플 규모에 비례해 증가하게 된다. 따라서 이 저장소의 적재 속도가 링크드 데이터 구축 시간을 좌우하는 중요한 요소로 판단할 수 있다. 중복된 데이터가 포함된 대용량 데이터 집합이라면 주어가 URI인 RDF 데이터의 중복 여부 처리를 고려할 필요가 있다. 대용량 서지 데이터 집합에서 중복되는 데이터는 대표적으로 저널과 이슈이다. RDF 데이터로 변환해 중복 처리하지 않은 채 RDF 트리플 저장소에 저장한 후, 저널의 URI가 주어인 RDF 데이터를 살펴보면 동일한 데이터가 중복되어 있다. 조회할 때 중복 제거된 후의 이 RDF 데이터가 최신이어야 하고, 또한 다른 LOD와 상호 연결됐을 때도 최신이어야 한다. 이와 같이 링크드 데이터 구축 이후의 중복된 RDF 데이터에 대한 작업이 수반되므로 비효율적으로 이어진다. 따라서 <그림 1>의 RDF 데이터 변환 및 적재 부분에서 주어가 URI인 모든 RDF 데이터가 최신이고 중복되지 않도록 우선적으로 처리해야 한다. 이를 위해서 링크드 데이터를 구축하는 과정에서는 RDF 트리플 저장소에 SPARQL로 매번 조회해 URI이 중복됐는지 확인한 후, 등록하거나 갱신한다. 이것은 중복 검사를 어떻게 구현할지, 실시간이나 배치 방식 중 어떤 것으로 처리할지 등에 따라 구축하는 데에 걸리는 시간을 좌우하므로, 이 시간을 상당히 단축시키되 주어가 URI인 RDF 데이터가 최신인지 검증하는 방법이 연구되어야 한다.

본 논문에서는 22,673,666 건의 대용량 데이터 집합이자 세계 최대 규모의 생의학 서지 데이터베이스인 MEDLINE 데이터를 RDF 데이터로 변환해 약 1,684,852,355건의 트리플로 링크드 데이터를 구축한다. 이 과정에서 RDF 트리플 저장소 접근을 최소화하는 이중 일괄 등록 방법 즉, RDF 트리플 저장소에 조회하고 삭제와 등록하는 과정을 줄인 후, 중복된 URI가 주어인 RDF 데이터 일괄 제거 및 일괄 재등록하는 방법을 제안한다.

Ⅱ. 관련 연구

미국 국립의과학도서관(National Library Of Medicine, NIH)에서 제공하는 생의학 서지 데이터베이스인 MEDLINE은 1946년 이후 5,600건 이상의 저널에 수록된 2,400만 건 이상의 서지 정보를 제공한다. PubMed(http://www.pubmed.gov)는 MEDLINE, 원문이 수록된

PMC(PubMed Central), 책의 컨텐츠를 온라인으로 제공하는 NCBI(National Center for Biotechnology Information)의 BookSelf 등 2,700만 건 이상의 문헌을 무료로 검색할 수 있는 온라인 검색엔진이다(NIH 2017). MEDLINE은 미국 국립의과학도서관이 통제하는 시소러스인 MeSH(Medical Subject Heading, 의학 주제 표목)로 색인되어 있으므로 PubMed에서 재현율과 정확도를 고려한 검색 방법으로 적절한 문헌을 찾을 수 있다. 미국 국립의과학도서관은 XML(Extensible Markup Language) 형식의 MEDLINE 데이터 파일과 DTD 파일, 샘플 파일 등의 다운로드와 관련 문서, FAQ 등을 웹페이지로 제공한다. MEDLINE 데이터의 특성상 XML로 분할하고, 요소를 추출하고, RDF 트리플 저장소나 DBMS 같은 저장소에 등록해 구축하는 일련 과정과 MEDLINE 데이터를 데이터마이닝, 언어 네트워크 분석, 전문검색 등의 텍스트 처리를 위한 원시 데이터로 다룰 수 있으므로 문헌정보학 관점에서 연구할만한 가치가 있다.

PubMed의 검색 조건이 복잡하기 때문에 다양한 사용자의 검색 요구에 적합한 응용에 관한 다수의 연구에서 텍스트 파일인 MEDLINE 데이터를 활용했었다. Zhiyong(2011)은 PubMed와 적용된 기술을 연대표로 소개하고, PubMed와 호환되는 도구인 28개를 각각 검색 결과 랭킹, 토픽 클러스터링, 시맨틱과 시각화로 결과 강화, 검색 인터페이스와 검색 경험을 개선하기 위한 도구로 분류하고 있다. 전형적인 PubMed 검색에 이 도구를 활용하는 사례를 도식화하고 PubMed와 호환되는 도구와 비교한 후, 검색, 결과 분석, 인터페이스, 사용성에 대한 제안을 했다.

Chen 외 6인(2010)은 MEDLINE 데이터를 포함한 다수의 데이터 집합을 RDF로 변환한 후, Bio2RDF, LODD(Linking Open Drug Data), DBPedia 등의 다른 LOD와 상호연결하는 LOD인 Chem2BioRDF를 구축하는 방법을 제안했다. 이 연구에서는 RDF 트리플 저장소인 D2R 서버와 SPARQL endpoint 외에 사용자 친화형 SPARQL 생성기, RDF 패싯 브라우저, 시각화 플러그인 등의 고유 기능을 기술하고 있다. 이외 PubMed Centeral 데이터를 RDF화해 웹서비스를 제공하는 방법에 관한 연구가 있었다(Castro, McLaughlin, and Garcia 2013).

Kilicoglu 외 5인(2008)은 MEDLINE의 인용에서 추출한 의미론적 서술 요약 및 시각화, 구조화된 여러 자원과 연결하는 웹 애플리케이션인 Semantic MEDLINE에 관한 연구를 했다. Lin(2016)은 MEDLINE의 초록과 TREC 2007 데이터를 대상으로 여러 검색 알고리즘을 적용해 전문 검색이 초록 검색보다 더 효과적인지에 관한 연구를 했다.

위의 선행 연구들에서는 MEDLINE 데이터를 기반으로 PubMed 사용자에게 효율적인 서비스를 제공하는 방법에 목적을 두고 있는 것으로 분석되었다.

그 반면, MEDLINE 데이터 자체를 구조화하여 이를 데이터베이스에 체계적으로 적재하려는 시도도 있었다. Oliver E 외 3인(2004)는 MEDLINE 데이터를 상용 데이터베이스인 DB2와

Oracle로 적재하는 방법에 관한 연구를 수행했다. DB2의 경우 2003년의 MEDLINE 데이터인 500개의 XML 파일을 대상으로 XML 분할한 후 Java로 14개의 테이블에 적재했다. Oracle의 경우 2002년 MEDLINE 데이터인 396개를 XML 파일로 분할한 후 Java로 적재하는 방법과 Perl로 CSV 파일로 저장한 후 Oracle의 SQL*Loader로 CSV 파일을 적재하는 방법으로 나눠 처리했다. 소요 시간은 DB2의 경우 76시간, Oracle의 경우 196시간과 132시간이었고, 디스크 용량은 DB는 46.3GB, Oracle은 37.7GB와 31.6GB이었다. 이 연구에서는 DB2와 Oracle의 모든 14개 테이블의 주키(primary key)가 PMID(PubMed Identifier)이므로 관련 데이터가 중복될 수 있어 처리가 필요한 것으로 분석되었다. 예로 저널 테이블에 동일한 저널 데이터가 여러 개가 존재하면 SQL을 이용해 데이터 중복을 처리하는 방법을 들 수 있다.

MEDLINE 데이터 외에 RDF 데이터를 대상으로 저장하거나 질의하는 방법에 관한 다수의 연구가 있었다. 문현정 외 3인(2007)은 데이터 중복을 최소화해 대용량의 RDF 데이터를 효율적으로 저장할 수 있는 관계 정보와 데이터 정보를 분리시킨 저장 구조를 제안했고, 관계형 데이터베이스를 이용한 추론 규칙으로 질의 성능을 개선하는 방법을 제시하였다. 정준원 외 4인(2006)은 RDF 데이터를 관계형 데이터베이스에 저장하고 처리하는 기법을 제시하고, 연관 데이터의 빠른 처리를 지원하기 위한 시멘틱 정보 기반 캐쉬 기법 및 사용자 중심의 온톨로지 검색 인터페이스를 제안했으며, 실제 시스템에 적용한 후 다른 온톨로지 시스템과 비교해 효율적으로 동작함을 확인하는 연구를 했다. 전명중, 홍진영, 박영택(2016)은 Spark를 기반으로 SPARQL 질의문을 SparkSQL로 변환해 질의 처리 속도를 향상시키는 질의 엔진을 구축하는 방법을 제안하였다. 김천중 외 5인(2014)는 동적 RDF 데이터 환경에서의 질의 성능을 개선하기 위해 빈번하게 사용되는 질의 데이터의 빈도를 고려해 분할 및 서버에 분산시킴으로써 특정 서버에 집중된 데이터 집중을 해결하는 RDF 동적 분할 기법에 관한 연구를 했다. 대용량 RDF 데이터를 대상으로 한 이 선행 연구들에서는 주로 질의 성능을 개선하는 데에 중점을 두고 있다.

지금까지 기술한 선행 연구들에서 보듯이 MEDLINE 데이터가 매우 방대하기 때문에 RDF 트리플 저장소의 특성을 고려해 링크드 데이터를 구축하는 절차 외에 구축 시간, 최적화 등과 관련된 부분에 관한 선행 연구가 아직 부족한 실정이다. 따라서 본 연구에서는 MEDLINE 데이터를 링크드 데이터로 구축하는 과정에서의 RDF 트리플 저장소 접근 최소화, URI가 주어인 RDF 데이터 중복 제거에 목적을 두고 구축 시간을 단축하는 방법에 관한 가이드라인을 제시하는 연구를 수행한다. 또한, 실시간으로 처리하는 일반적인 구축 방법과 비교했을 때 RDF 트리플 저장소 접근 횟수가 얼마나 감소됐는지를 확인함으로써 링크드 데이터 구축에 상당한 시간이 걸리는 대용량 집합을 처리함에 있어서 본 논문에서 제안하는 방법이 효율적임을 입증하고자 한다.

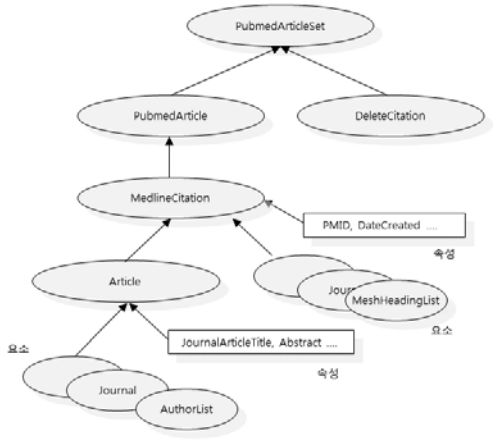
Ⅲ. 연구 설계

이 장에서는 MEDLINE 데이터 구조를 정의한 DTD에서 발췌한 논문과 관련된 요소, 이들 요소를 활용해 도출한 RDF 스키마의 클래스와 속성, 링크드 데이터로 변환할 때의 주어 URI 명명 규칙, RDF 트리플 접근을 최소화하기 위한 실험 시나리오 선정 과정을 차례대로 기술한다.

1. MEDLINE 데이터 구조 및 RDF 스키마 도출

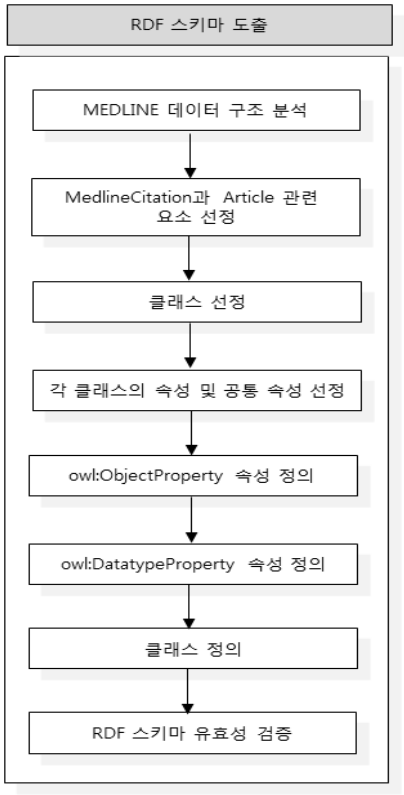
XML 형식인 MEDLINE 데이터 구조는 DTD(Document Type Definition)로 정의되고 있다. 이 DTD의 논문 관련 요소의 일부를 발췌한 것과 각 요소 간의 관계를 도식화한 것을 <그림 2>에 제시한다.

```
<!-- Set-level elements -->
<!ELEMENT PubmedArticleSet ((PubmedArticle | PubmedBookArticle)+, DeleteCitation?) >

<!-- Document-level elements -->
<!ELEMENT PubmedArticle (MedlineCitation, PubmedData?)>
<!ELEMENT DeleteCitation (PMID+) >

<!-- Sub-Document wrapper elements -->
<!ELEMENT MedlineCitation (PMID, DateCreated, DateCompleted?, DateRevised?, Article, MedlineJournalInfo, ChemicalList?, SupplMeshList?, CitationSubset*, CommentsCorrectionsList?, GeneSymbolList?,MeshHeadingList?, NumberOfReferences?, PersonalNameSubjectList?, OtherID*, OtherAbstract*, KeywordList*, CoiStatement?, SpaceFlightMission*, InvestigatorList?, GeneralNote*)>
<!ELEMENT Article (Journal,ArticleTitle,((Pagination, ELocationID*) | ELocationID+), Abstract?,AuthorList?, Language+, DataBankList?, GrantList?,PublicationTypeList, VernacularTitle?, ArticleDate*) >
```



<그림 2> MEDLINE 데이터의 DTD에서 발췌한 논문과 관련된 일부 요소들

<그림 2>에 보듯이 논문과 관련된 요소는 MedlineCitation과 Article이다. MedlineCitation 요소는 Article의 부모 요소인 논문 관련 메타 정보가 정의되어 있고, Article 요소는 논문, 저널, 저자, 이슈, 후원 등과 관련된 요소가 정의되어 있음을 확인할 수 있다. 본 연구에서는 MEDLINE 데이터의 MedlineCitation과 Article 관련 요소들을 선정한 후, 이 요소들과 관계를 활용해 RDF 스키마의 클래스와 속성을 정의하는 방향으로 다음 <그림 3>과 같이 RDF 스키마를 도출했다.



<그림 3> RDF 스키마 도출 과정

이 RDF 스키마 도출 과정에서 최종적으로 다음 절에 기술할 7개의 클래스, 5개의 owl:ObjectProperty 속성, 35개의 owl:DatatypeProperty 속성들을 정의하고, 클래스와 속성 간의 관계와 속성 간의 관계를 rdfs:domain과 rdfs:range 속성으로 제한했다. 마지막에서는 도출한 RDF 스키마의 유효성을 검증하기 위해 W3C RDF Validator (https://www.w3.org/RDF/Validator/)에서 수행했다.

2. RDF 스키마 명세

가. 클래스와 속성 목록

도출한 RDF 스키마는 7개의 클래스(논문, 저널, 이슈, 저자, 저자 소속기관, 후원, 후원 기관)와 40개의 속성으로 구성된다. 다음 <표 1>은 URI를 갖는 자원(resource)의 타입인 클래스를 나열한 것이다.

<표 1> 클래스 목록

클래스	rdf:type
논문	pubmed:Article
저널	pubmed:Journal
이슈	pubmed:Issue
저자	foaf:Person
저자 소속기관	foaf:Organization
후원	pubmed:Grant
후원 기관	foaf:Organization

다음 <표 2>는 두 클래스의 객체(instance) 즉, 자원 간의 관계를 표현하는 owl:ObjectProperty 속성 타입으로 정의한 객체 관계 속성을 나열한 것이다. <표 2>의 rdfs:domain과 rdfs:range는 속성의 범위를 제한하는 속성이다. 속성의 정의역을 정의하는 속성인 rdfs:domain은 속성의 주어인 클래스를 명시하고, 속성의 치역을 정의하는 속성인 rdfs:range는 목적어인 클래스를 명시한다.

<표 2> owl:ObjectProperty 속성

rdfs:domain	속성(owl:ObjectProperty)	rdfs:range
pubmed:Article	pubmed:hasJournal	pubmed:Journal
pubmed:Journal	pubmed:hasIssue	pubmed:Issue
pubmed:Article	pubmed:hasAuthor	foaf:Person
foaf:Person	pubmed:hasAffiliation	foaf:Organization
pubmed:Article	pubmed:hasGrant	pubmed:Grant
pubmed:Grant	pubmed:hasAgency	foaf:Organization

다음 <표 3>은 클래스의 객체에 특정 데이터 타입을 연결시키는 owl:DatatypeProperty 속성 타입을 이용해 정의한 데이터 타입 속성을 나열한 것이다. 이 속성은 owl:Object Property와 동일하게 rdfs:domain과 rdfs:range로 제한된다. pubmed로 시작하는 속성의

목적어에 해당하는 데이터 타입은 xsd:string 즉, 문자열로 정의되어 있다.

<표 3> owl:DatatypeProperty 속성

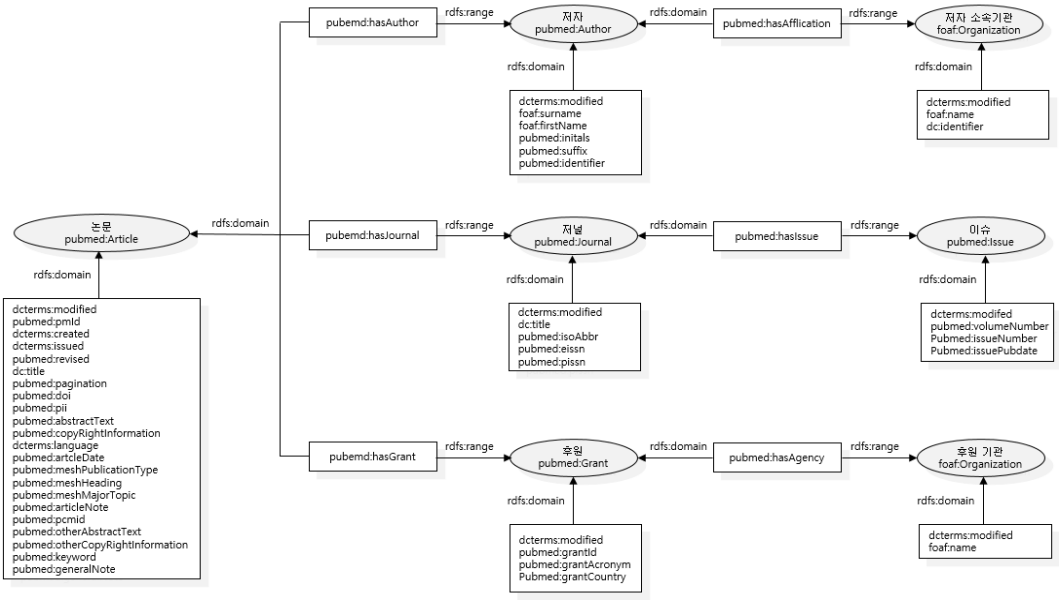
rdfs:domain	속성(owl:DatatypeProperty)	rdfs:range
pubmed:Article 논문	dcterms:modified	
	dcterms:created	
	dcterms:issued	
	pubmed:revised	xsd:string
	pubmed:pmId	xsd:string
	dc:title	
	pubmed:pagination	xsd:string
	pubmed:doi	xsd:string
	pubmed:pii	xsd:string
	pubmed:abstractText	xsd:string
	pubmed:copyRightInformation	xsd:string
	pubmed:articleDate	xsd:string
	pubmed:meshHeading	xsd:string
	pubmed:meshMajorTopic	xsd:string
	pubmed:articleNote	xsd:string
	pubmed:pmcId	xsd:string
	pubmed:otherAbstractText	xsd:string
	pubmed:otherCopyRightInformation	xsd:string
pubmed:Journal 저널	dcterms:modified	
	dc:title	
	pubmed:isoAbbr	xsd:string
	pubmed:eissn	xsd:string
	pubmed:pissn	xsd:string
pubmed:Issue 이슈	dcterms:modified	
	pubmed:volumeNumber	xsd:string
	pubmed:issueNumber	xsd:string
	pubmed:issuePubdate	xsd:string
foaf:Person 저자	dcterms:modified	
	foaf:surname	
	foaf:firstName	
	pubmed:initals	xsd:string
	pubmed:suffix	xsd:string
	foaf:name	
	dc:identifier	
foaf:Organization 저자 소속기관	dcterms:modified	
	foaf:name	
	dc:identifier	
pubmed:Grant 후원	dcterms:modified	
	pubmed:grantId	xsd:string
	pubmed:grantAcronym	xsd:string
	pubmed:grantCountry	xsd:string
foaf:Organization 후원 기관	dcterms:modified	
	foaf:name	

<표 1>, <표 2>, <표 3>의 pubmed, dc, dcterms, foaf, xsd, rdfs는 자원의 속성이나 자원 간의 관계를 표현하는 어휘(vocabulary)의 네임스페이스(namespace)에 대한 약자다. 다음 <표 4>는 각 어휘의 네임스페이스를 나열한 것이다. 그 중에서 pubmed는 본 연구에서 도출한 RDF 스키마를 명세하기 위한 어휘의 약자이다.

<표 4> 네임스페이스 목록

어휘 약자	네임스페이스
pubmed	http://pubmed.kgu.ac.kr/ontology/
dc	http://purl.org/dc/elements/1.1/
dcterms	http://purl.org/dc/terms/
foaf	http://xmlns.com/foaf/0.1/
xsd	http://www.w3.org/2001/XMLSchema#string
rdfs	http://www.w3.org/2000/01/rdf-schema#

RDF 스키마의 클래스와 속성 간의 관계를 <표 2>와 <표 3>의 rdfs:domain 속성과 rdfs:range 속성을 이용해 간략히 도식화한 것을 다음 <그림 4>에 제시한다.



<그림 4> rdfs:domain과 rdfs:range로 제한된 클래스와 속성 간의 관계

<그림 4>에서 우선 클래스 간의 관계를 살펴보면 논문을 최상위 클래스로 간주했을 때 논

문, 저널, 이슈, 저자, 저자 소속기관, 후원, 후원 기관 클래스 간의 관계가 계층적임을 추론할 수 있다. 클래스와 속성 간의 관계를 rdfs:domain으로 제한했으므로 속성의 주어가 어떤 클래스인지 알 수 있다.

나. 주어 URI 명명 규칙

MEDLINE 데이터를 RDF 스키마에 매핑해 RDF 데이터로 변환하는 과정에서 URI를 부여하는데, 이때 이 URI을 쉽게 식별해야 한다. 따라서 본 연구에서는 DTD의 요소를 참조해 <표 5>의 주어 URI 명명 규칙을 적용한다. 논문, 저널, 이슈는 유일하게 식별할 수 있는 요소가 MELINE 데이터의 DTD에 있으므로 해당 요소 값을 이용한다. 그 반면, 저자, 저자 소속기관, 후원, 후원 기관은 이들 요소가 존재하지 않으므로 고유키를 활용한다.

<표 5> 주어 URI 명명 규칙

클래스	DTD 요소	URI
논문	PMID	http://pubmed.kgu.ac.kr/resource/article/[PMID]
저널	ISSN, ISSNType	ISSNType 요소의 값이 Print이면 http://pubmed.kgu.ac.kr/resource/journal/p[ISSN]
		ISSNType 요소의 값이 Electronic이면 http://pubmed.kgu.ac.kr/resource/journal/e[ISSN]
이슈	Volume, Issue	http://pubmed.kgu.ac.kr/resource/issue/p[ISSN]_v[Volume]_i[Issue]
		http://pubmed.kgu.ac.kr/resource/issue/e[ISSN]_v[Volume]_i[Issue]
저자	없음	http://pubmed.kgu.ac.kr/resource/person/[고유키]
저자 소속기관	없음	http://pubmed.kgu.ac.kr/resource/org/[고유키]
후원	없음	http://pubmed.kgu.ac.kr/resource/grant/[고유키]
후원 기관	없음	http://pubmed.kgu.ac.kr/resource/org/[고유키]

예외적으로 다음 <그림 5>와 같이 저널 정보에 ISSN이 없고, 이슈 정보에 권호수가 없는 MEDLINE 데이터가 존재할 수 있다. 이 경우를 고려해 저널 클래스의 주어 URI 명명 규칙을 http://pubmed.kgu.ac.kr/resource/journal/[고유키]로 정의하고, 이슈 클래스의 주어 URI 명명 규칙을 http://pubmed.kgu.ac.kr/resource/issue/[고유키]로 정의한다.

```
<PubmedArticleSet>
    <PubmedArticle>
    <MedlineCitation Status="MEDLINE" Owner="NLM">
      <PMID Version="1">5830790</PMID>
      ...(중략)...
```

```
        <Article PubModel="Print">
          <Journal>
            <JournalIssue CitedMedium="Print">
              <PubDate>
                <MedlineDate>1964-1965</MedlineDate>
              </PubDate>
            </JournalIssue>
            <Title>The Proceedings of the Cardiff Medical Society</Title>
            <ISOAbbreviation>Proc Cardiff Med Soc</ISOAbbreviation>
          </Journal>
          ...(중략)...
      </MedlineCitation>
      <PubmedData>
      ...(중략)...
      </PubmedData>
    </PubmedArticle>
</PubmedArticleSet>
```

<그림 5> ISSN과 권호수가 모두 존재하지 않는 MEDLINE 데이터 예시

3. 실험 시나리오 선정

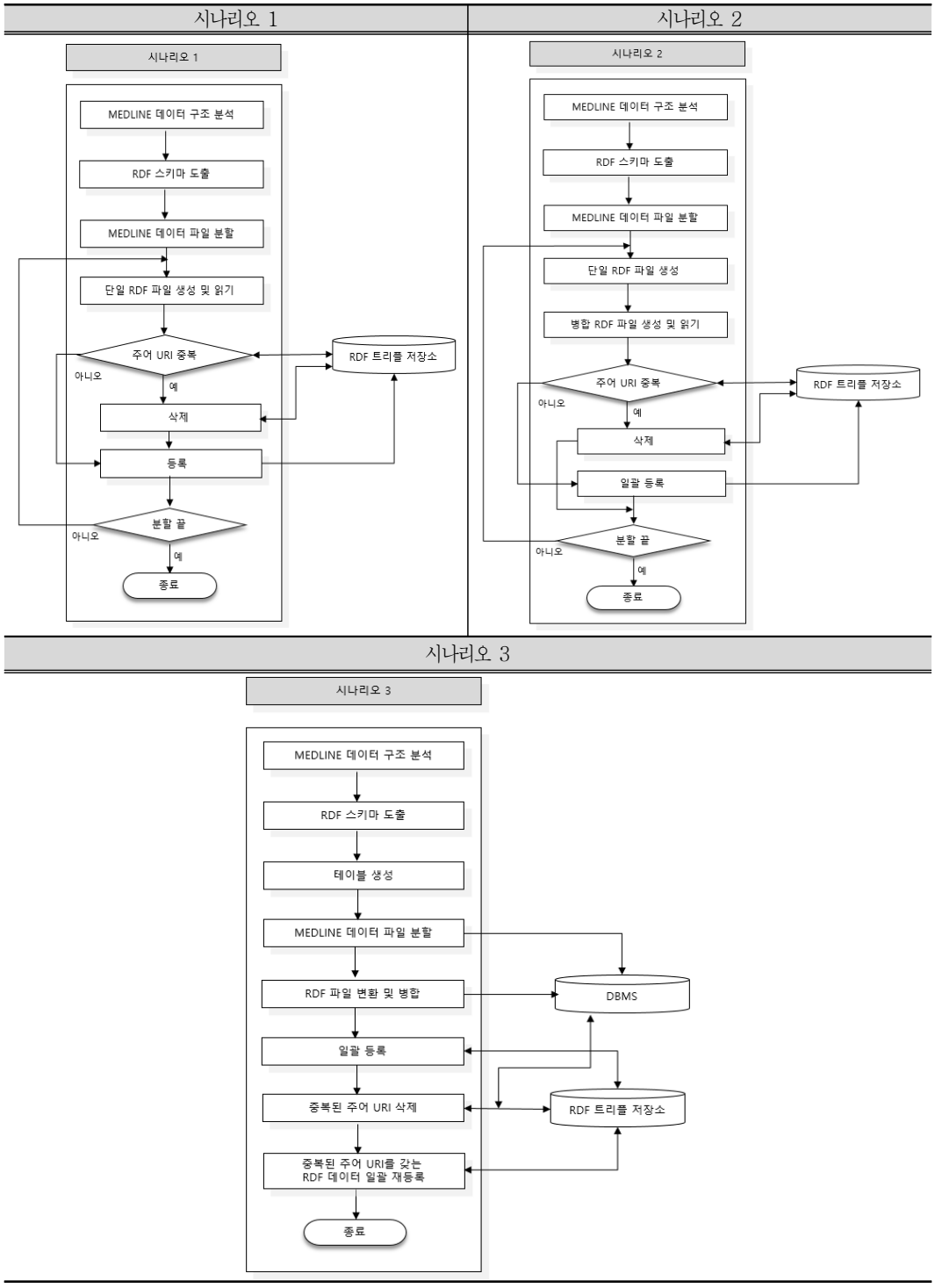
MEDLINE 데이터를 RDF로 변환해 링크드 데이터 구축을 완료한 후, RDF 트리플 저장소에서 주어 URI를 갖는 RDF 데이터가 최신이고 유일하게 존재해야 한다. 따라서 이를 위한 고려사항을 정리해 다음 <표 6>에 제시한다.

<표 6> 링크드 데이터 구축 시 고려사항

항목	구분	
구축 방식	실시간	비실시간
일괄 처리 모드	온라인	오프라인
URI 중복 조회/삭제	즉시	후처리
원본 파일 분할	예	아니오
RDF 파일 저장	예	아니오
RDF 파일 처리 단위	개별	병합
DBMS 활용	예	아니오

<표 6>의 항목을 기준으로 조합하면 다수의 구축 시나리오가 나올 수 있다. 본 연구의 실험 환경은 JDK 1.8, Jena, Virtuoso Jdbc이고, DBMS와 RDF 트리플 저장소는 각각 MySQL 5.7과 OpenLink사의 Virtuoso 7.2이다. RDF 파일을 후처리하기 위해 보존한다는 전제에서 이 실험 환경에 적합한 세 가지 시나리오를 다음 <표 7>에 제시한다.

〈표 7〉 링크드 데이터 구축 시나리오



<표 7>에서 보듯이 MELINE 데이터 파일 분할까지는 모든 시나리오에서 동일하다. 시나리오 1에 따르면 단일 RDF 파일을 읽으면서 주어 URI 중복 조사한 후, 중복 여부에 따라 삭제 후 등록이나 등록 중 하나를 수행한다. 시나리오 2에 따르면 병합 RDF 파일을 읽으면서 주어 URI 중복 조사한 후 중복되면 삭제한다. 그 후 일괄 등록한다. 시나리오 3에 따르면 먼저 병합 RDF 파일을 일괄 등록한다. 그 다음 중복된 모든 주어 URI를 일괄 삭제한 후, 중복된 주어 URI를 갖는 RDF 데이터들을 일괄 재등록한다.

세 가지 시나리오 중 주어 URI 중복 조사 횟수와 RDF 트리플 저장소 접근 횟수가 가장 적은 것은 시나리오 3이다. 따라서 본 연구에서는 시나리오 3을 선정해 실험한다.

Ⅳ. 실험 및 결과 분석

3장에서 선정한 링크드 데이터 구축 시나리오를 기준으로 이중 일괄 등록 방법을 적용하는 과정을 상세하게 기술한다. 구축 절차는 이중 일괄 등록하기 위해 활용하는 테이블 생성, 대용량 XML 파일을 멀티스레딩 기법으로 분할, 분할된 XML 파일을 대상으로 RDF 파일 변환 및 병합, RDF 트리플 저장소에 일괄 등록, 중복된 URI가 주어인 RDF 데이터를 일괄 재등록하는 과정으로 구성된다. 이 구축 절차가 끝나면 결과를 분석해 고찰한다.

1. 테이블 생성

이중 일괄 등록 방법을 실험에 적용하기 위해서 XML 분할과 RDF 파일 변환 상태 관리 및 중복된 모든 주어 URI의 최신 RDF 데이터 등의 관련 정보를 활용할 수 있는 저장소가 필요하다. 따라서 본 연구에서 PUBMED_FILE과 PUBMED_URI 테이블을 생성했다. 중복된 URI를 조사하기 위한 PUBMED_FILE 테이블은 XML 파일 분할 상태와 RDF 파일 변환

〈표 8〉 PUBMED_FILE 테이블의 주요 컬럼

컬럼명	설명
MEDLINE_NM	MEDLINE 디렉터리 이름
SPLIT_STATUS	XML 파일 분할 상태
RDF_TRANS_STATUS	RDF 파일 변환 상태
RDF_INSERT_STATUS	RDF 트리플 저장소 등록 상태
ARTICLE_URI	논문 URI
JOURNAL_URI	저널 URI
ISSUE_URI	이슈 URI
RDF_TRIPLE_CNT	RDF 파일당 트리플 건수

상태, RDF 트리플 저장소 등록 상태 등의 메타 정보를 저장한 테이블이다. 그 반면, 중복된 URI가 주어인 RDF 데이터를 가져오기 위한 PUBMED_URI 테이블은 RDF 파일 변환 과정에서 추출한 저널 URI과 이슈 URI, 각 URI가 주어인 최신 RDF 데이터를 저장한 테이블이다. 이 두 테이블의 세부 구조를 <표 8>과 <표 9>에 제시한다.

<표 9> PUBMED_URI 테이블의 주요 컬럼

컬럼명	설명
JOURNAL_URI	저널 URI
JOURNAL_TRIPLES	저널 URI이 주어인 RDF 데이터
ISSUE_URI	이슈 URI
ISSUE_TRIPLES	이슈 URI이 주어인 RDF 데이터

2. MEDLINE 데이터 파일 분할

MEDLINE 데이터는 XML 형식인 텍스트 파일을 압축한 gz 포맷이며, 2017년 8월 기준으로 다음 <표 10>과 같이 게시되어 있다. 본 연구에서는 Annual Baseline에 게시된 892개의 gz 포맷 압축 파일을 내려 받아 사용한다. 다운로드한 파일의 총 용량은 20.7GB이다.

<표 10> MEDLINE 데이터

구분	파일 개수
Annual Baseline	892개
Daily Update Files	343개

MEDLINE의 각 데이터 파일을 압축 해제하면 평균 450MB의 용량인 XML이 하나만 있다. 이 XML 파일에는 평균 25,418 건의 논문 관련 정보가 모두 들어 있다. 이 XML 파일을 분할하기 위해서 DOM 파서와 SAX 파서 중 하나를 사용할 수 있다. DOM 파서는 XML을 전체 읽은 후 파싱한 각 태그 정보를 사용하기 때문에 저용량의 XML 파일을 처리하기 쉽다. 그 반면, SAX 파서는 읽어 들인 태그를 기준으로 XML 파일을 파싱하므로 대용량의 XML 파일을 처리하는 데에 적합하다. 따라서 멀티스레딩 기법을 통한 병행처리(concurrent processing)로 892개의 MEDLINE 데이터 파일을 압축 해제한 후, SAX 파서를 사용해 XML 파일을 분할한다.

이 XML 파일을 분할하는 과정에서 평균 25,418개의 작은 XML 파일을 생성한다. HDD의 쓰기/읽기 속도가 SSD에 비해 상대적으로 느리고, 특히 파일의 크기가 작고 개수가 많을수록 처리 시간이 증가하는 것으로 알려졌다. 따라서 본 연구에서는 처리 속도를 향상시키기 위해서 HDD에 있는 MEDLINE 데이터의 압축 파일을 SSD의 임시 디렉토리에 복사한 후, 압축을

해제하고 XML 파일을 분할한다. 이 과정을 마치면 XML 분할 관련 정보를 PUBMED_FILE 테이블에 등록한다. XML 파일 분할 과정에서 어떠한 원인으로 오류가 발생해 중단될 수 있다. 이때 PUBMED_FILE 테이블의 SPLIT_STATUS 컬럼의 값을 활용하면 그 시점에서 재시작할 수 있다.

다음 절에 기술할 분할된 XML 파일을 RDF 파일로 변환 및 병합하는 과정을 모두 완료하면 다시 압축해 HDD에 복사한다. XML 분할부터 RDF 파일 병합까지 8개의 멀티스레드로 처리했을 때 소요 시간은 약 9시간 6분이었다.

3. RDF 파일 변환 및 병합

XML 파일 분할 과정이 끝나면 MEDLINE 디렉터리에 있는 모든 XML 파일 목록을 가져온 후, 각 XML 파일을 읽은 다음에 RDF 스키마에 맞춰 트리플로 변환한다. 이때 S(주어)-P(술어)-O(목적어)로 구성된 N-Triples 형식의 O에 해당하는 값이 문자열이면 큰따옴표로 묶여 있다. 이 값 안에 큰따옴표나 사선이 있을 시 이스케이프로 처리해야 한다. 트리플 변환 과정을 마치면 XML 파일명 뒤에 “.nt”를 붙여 저장한다. 이 nt 확장자는 N-triples 형식의 약자를 뜻한다. 이 N-Triples 형식의 RDF 파일 내용의 일부를 예로 보여주기 위해 S-P-O 형식으로 정리한 것을 <표 11>에 제시한다.

<표 11> 변환된 RDF 파일 내용

S	P	O
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://pubmed.kgu.ac.kr/ontology/Article>
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://purl.org/dc/terms/modified>	"2017-07-21 04:15:22"
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://pubmed.kgu.ac.kr/ontology/pmId>	"27899033"^^<http://www.w3.org/2001/XMLSchema#string>
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://purl.org/dc/terms/created>	"2016-11-30"
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://pubmed.kgu.ac.kr/ontology/revised>	"2016-12-03"^^<http://www.w3.org/2001/XMLSchema#string>
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://purl.org/dc/elements/1.1/title>	"Novel antibacterial polymeric nanocomposite for smart co-delivery of anticancer drugs."@en
<http://pubmed.kgu.ac.kr/resource/article/27899025>	<http://pubmed.kgu.ac.kr/ontology/pagination>	"1-12"^^<http://www.w3.org/2001/XMLSchema#string>

저널 URI와 이슈 URI가 주어인 RDF 데이터를 PUBMED_URI 테이블에 저장할 때 해당 URI의 존재 여부와 무관하게 삭제한 후 등록한다. 변환 완료된 모든 RDF 파일을 단일 RDF 파일로 병합한다. XML 파일을 RDF 파일로 변환해 저장하는 이유는 다음의 두 가지이다.

첫째, XML 파일을 읽어 분할한 후 트리플로 변환하는 과정에서 URI가 주어인 RDF 데이터가 RDF 트리플 저장소에 존재하는지 확인하기 위해 먼저 조회한다. 존재하면 삭제한 후에 등록한다. N개의 멀티스레드로 이 과정을 수행할 경우 중복 처리하기 위해 RDF 트리플 저장소에 2N번 접근해야 한다. 따라서 RDF 트리플 저장소에 접근하지 않도록 RDF 파일로 저장한 후 병합하는 것이다.

둘째, SPARQL의 패턴매칭 기반 regex() 함수를 사용해 검색할 때, 대용량 RDF 트리플 저장소이면 조회 성능이 저하될 수 있다. 이를 보완하기 위한 방법은 전문검색엔진을 이용한 전문검색이며, RDF 파일을 원시 데이터 파일로 간주해 색인하는 것이다.

모든 XML 파일을 RDF 파일로 변환해 병합을 완료한 후의 통계는 다음 <표 12>와 같으며, PUBMED_FILE과 PUBMED_URI 테이블에 SQL로 질의해 이 통계를 얻었다.

<표 12>의 통계는 MEDLINE 데이터 집합 내의 논문 건수, 저널 URI와 이슈 URI의 중복 건수와 관련된 세 가지 중요한 사실을 제시하고 있다.

〈표 12〉 PUBMED_FILE과 PUBMED_URI를 이용한 통계

테이블	항목	총 건수
PUBMED_FILE	논문 URI	22,673,666건
	저널 URI	413,735건
	이슈 URI	2,015,936건
	중복된 논문 URI	0건
	중복된 저널 URI	21,427건
	중복된 이슈 URI	1,483,086건
	총 트리플 건수	1,949,304,499건
	RDF 파일당 평균 트리플 건수	85.97건
PUBMED_URI	중복되지 않은 저널 URI	413,735건
	중복되지 않은 이슈 URI	2,015,936건

첫째, 중복된 논문 URI의 총 건수가 0이므로 MEDLINE 데이터 집합 안에 중복된 논문이 없음을 의미한다. 이것은 PUBMED_URI 테이블을 생성할 때 논문 URI, 논문 URI가 주어인 RDF 데이터에 대한 두 컬럼을 추가하지 않은 이유다.

둘째, 각 중복된 저널 URI와 이슈 URI의 총 건수는 21,427건과 1,483,086건이다. 일괄 등록이 완료된 후의 RDF 트리플 저장소에서 중복된 저널 URI과 이슈 URI가 주어인 RDF 데이터를 일괄 삭제한 후 일괄 재등록해야 함을 보여준다.

셋째, 각 중복되지 않은 저널 URI와 이슈 URI의 총 건수는 링크드 데이터를 구축한 결과가 정상임을 판단하는 기준이 된다.

4. RDF 트리플 저장소에 일괄 등록

RDF 변환 및 병합 완료한 후의 RDF 파일 개수는 MEDLINE 데이터 내의 파일 개수와 동일해야 한다. 이를 확인한 후, 이 모든 병합된 RDF 파일 목록을 읽어 RDF 트리플 저장소에 순차적으로 일괄 등록한다. 이때 각 RDF 파일의 내용을 차례대로 등록하므로 중복된 트리플이 RDF 트리플 저장소에 존재할 수 있다. 여기서 중복은 트리플의 S와 P가 동일하고 O가 다른 경우를 의미한다. 순차적으로 처리하는 이유는 병합된 RDF 파일 당 평균 트리플 건수가 약 218만 건이기 때문이다. 또한 멀티스레딩 기법으로 일괄 등록을 하면 적재에 따른 부하로 인해 트랜잭션 오류가 발생할 수도 있고, RDF 트리플 저장소에 적재된 트리플의 총 건수에 따라 적재 속도가 느려질 수도 있다는 점을 고려했다. 일괄 등록을 프로그램적으로 처리하기 위해 RDF 트리플 저장소인 Virtuoso의 BulkUpdateHandler API를 사용한다. 참고로 오프라인 방식의 일괄 등록도 지원한다. Viruoso의 명령행 도구인 isql에서 우선 ld_dir() 함수나 ld_dir_all() 함수에 디렉토리, 파일명, 확장자 패턴, 그래프 IRI(International Resource Identfier) 등을 인자로 지정해 실행한 후, rdf_loader_run() 함수를 실행한다.[1] 이것은 RDF 파일 개수가 많을수록 일일이 입력해야 하는 번거로움이 있고, 소요 시간을 정확하게 계산하기가 어려우므로 본 연구에서 배제했다.

모든 병합된 RDF 파일을 RDF 트리플 저장소에 순차적으로 일괄 등록하는 데에 소요된 시간은 약 45시간 26분이었다. RDF 트리플 저장소에 등록된 트리플 총 건수는 1,684,852,355건이며, 다음 <그림 6>의 SPARQL 질의를 실행해 조회했다.

```
select count(*)
from <http://pubmed.kgu.ac.kr/resource/>
where {
  ?s ?p ?o
}
```

<그림 6> 총 트리플 건수를 조회하는 질의

<그림 6>의 질의로 조회한 총 트리플 건수와 <표 11>의 총 트리플 건수가 차이가 나는 이유는 저널의 URI와 이슈의 URI가 중복됐기 때문이다. 논문 URI, 저널 URI, 이슈 URI의 총 건수를 구하는 SPARQL 질의와 실행 결과를 각각 <그림 7>과 <표 13>에 제시하며, <표 12>와 동일하게 나왔다.

1) https://www.openlinksw.com/iODBC/main/Main/VirtBulkRDFLoader

```
select count(?article_uri),
       count(?journal_uri),
       count(?issue_uri)
where
{
  { ?article_uri a <http://pubmed.kgu.ac.kr/ontology/Article>}
  union
  { ?journal_uri a <http://pubmed.kgu.ac.kr/ontology/Journal>}
  union
  { ?issue_uri a <http://pubmed.kgu.ac.kr/ontology/Issue>}
}
```

(그림 7> 논문 URI, 저널 URI, 이슈 URI의 총건수 조회 질의

<표 13> 논문 URI, 저널 URI, 이슈 URI의 총 건수 조회 결과

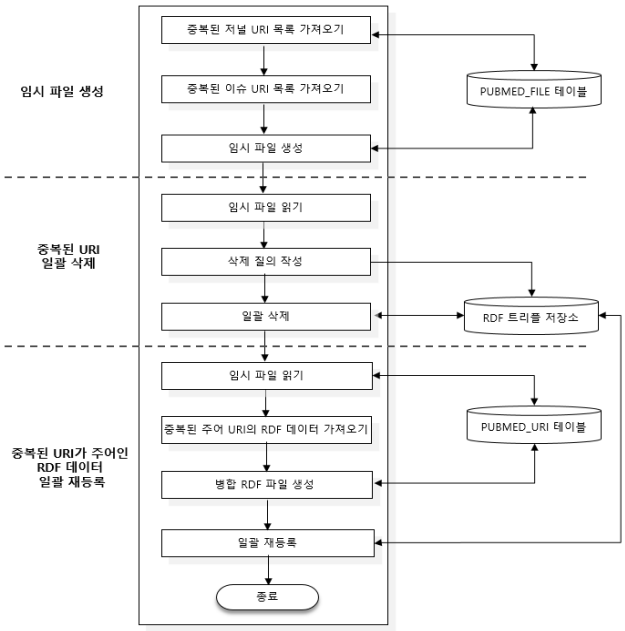
항목	총 건수
논문 URI	22,673,666건
저널 URI	413,735건
이슈 URI	2,015,936건

5. RDF 데이터 일괄 재등록

RDF 트리플 저장소에 일괄 등록한 후의 다음 과정은 임시 파일 생성, 중복된 URI 일괄 삭제, 중복된 URI가 주어인 RDF 데이터 일괄 재등록으로 구성된다. 이 과정을 다음 <그림 8>에 제시하고, 차례대로 기술한다.

첫째, PUBMED_FILE 테이블에서 중복된 저널 URI와 이슈 URI 목록을 가져와 텍스트 포맷인 임시 파일을 생성해 저장한다. 이것은 PUBMED_FILE에 다시 조회할 때의 시간을 줄이기 위한 방법이다.

둘째, 임시 파일을 다시 읽어 가져온 URI 목록을 대상으로 <그림 9>와 같이 SPARQL의 DELETE와 IN을 이용해 일괄 삭제하는 질의를 작성한다. 그 다음에 RDF 트리플 저장소에 삭제 질의를 수행한다. 단, IN의 특성상 삭제 질의를 수행할 때의 URI 최대 개수는 RDF 트리플 저장소마다 다를 수 있다. 본 연구에서는 일괄 삭제를 위한 질의를 1,000건마다 생성한 후, RDF 트리플 저장소에 모든 질의를 순차적으로 수행했다.



<그림 8> 중복된 URI가 주어인 RDF 데이터 일괄 재등록

```
with <http://pubmed.kgu.ac.kr/resource/>
delete { ?s ?p ?o }
where { ?s ?p ?o .
    filter(?s IN (
        <http://pubmed.kgu.ac.kr/resource/journal/e1098-1128>,
        <http://pubmed.kgu.ac.kr/resource/journal/e1098-1136>,
        ....(중략)....
        <http://pubmed.kgu.ac.kr/resource/journal/p8756-971x>,
        <http://pubmed.kgu.ac.kr/resource/journal/p8756-9787>
      )
    ).
}
```

<그림 9> 중복된 저널 URI과 이슈 URI의 RDF 데이터를 일괄 삭제하기 위한 질의

셋째, 임시 파일을 또 다시 읽어 가져온 URI 목록을 조회하면서 PUBMED_URI 테이블에 있는 해당 URI의 RDF 데이터를 가져온다. 그 다음에 단일 RDF 파일에 덧붙이는 방식으로 병합한 후 저장한다. 이 병합된 RDF 파일을 RDF 트리플 저장소에 일괄 재등록한다.

중복된 URI가 주어인 RDF 데이터를 일괄 삭제한 후, 일괄 재등록하는 데 소요된 시간은 약 5시간 25분이었다. 총 트리플 건수는 1,677,703,788건이며, 약 14%의 중복된 저널 URI와 이슈 URI가 주어인 RDF 데이터인 트리플이 삭제됐다.

6. 결과 및 분석

본 연구에서는 링크드 데이터 구축 과정에서 RDF 트리플 저장소에 대한 접근 최소화에 목적을 두고, 중복된 URI가 주어인 RDF 데이터를 최종적으로 제거했다. 총 구축 시간은 59시간 57분이며, 일수로 환산하면 약 2.5일이다. 총 소요 용량은 153GB이며, 그 중에서 RDF 트리플 저장소에 등록한 후의 데이터베이스 소요 용량은 110GB이다. 이 총 구축 시간과 총 소요 용량은 RDF 스키마의 클래스/속성 정보에 기인한 트리플 개수 증감, RDF 트리플 저장소의 적재 성능, HDD와 SSD 사용 여부와 읽기/쓰기 속도, 멀티스레딩 기법 적용 여부 등에 따라 달라질 수 있다.

본 연구에서 제안한 이중 일괄 등록 방법이 RDF 트리플 저장소 접근 횟수가 적음을 보여주기 위해 앞에서 기술한 다른 두 시나리오와 비교한 것을 <표 14>에 제시한다.

<표 14> RDF 트리플 저장소 접근 횟수 비교

시나리오 구분	처리과정	접근 횟수
단일 RDF 파일 대상	RDF 트리플 저장소에 개별 등록	22,673,666회
	저널 URI 중복 조사	413,735회
	저널 URI가 주어인 RDF 데이터 삭제	21,427회
	이슈 URI 중복 조사	2,015,936회
	이슈 URI가 주어인 RDF 데이터 삭제	1,473,086회
	총합	26,597,850회
병합 RDF 파일 대상	RDF 트리플 저장소에 일괄 등록	892회
	저널 URI 중복 조사	413,735회
	저널 URI가 주어인 RDF 데이터 삭제	21,427회
	이슈 URI 중복 조사	2,015,936회
	이슈 URI가 주어인 RDF 데이터 삭제	1,473,086회
	총합	3,925,076회
제안한 방법 (이중 일괄 등록)	RDF 트리플 저장소에 일괄 등록	892회
	중복된 저널 URI가 주어인 RDF 데이터 일괄 삭제	22회
	중복된 이슈 URI가 주어인 RDF 데이터 일괄 삭제	1,484회
	중복된 저널 URI가 주어인 RDF 데이터를 모두 병합한 후, RDF 트리플 저장소에 일괄 재등록	1회
	중복된 이슈 URI가 주어인 RDF 데이터를 모두 병합한 후, RDF 트리플 저장소에 일괄 재등록	1회
	총합	2,400회

<표 14>에 보듯이, 단일 RDF 파일이나 병합 RDF 파일을 대상으로 RDF 트리플 저장소에 실시간으로 접근해 순차적으로 중복된 URI를 확인하고 삭제나 등록하는 과정에서 접근 횟수가 상당함을 확인했다. 멀티스레딩 기법을 통해서 병행 처리할 경우에도 접근 횟수는 동일하지만, RDF 트리플 저장소에 적재된 트리플 총 개수가 대용량일수록 순차적 처리보다 더 느려질 가능성을 배제할 수 없다. 링크드 데이터를 구축하는 과정에서 중복된 URI가 주어인 RDF 데이터를 처리하는 부분이 수반됨에도 불구하고 RDF 트리플 저장소에 단순히 의존한다면 구축 소요 시간을 예측하기가 어려워진다. 이와 같이 링크드 데이터 구축 시 RDF 트리플 저장소의 성능도 중요한 사항이지만, 대용량 데이터 집합을 대상으로 구축한다면 프로그램적으로 처리할 수 있는 방법이 필요하다.

따라서 본 연구에서 제안한 이중 일괄 등록 방법은 RDF 트리플 저장소 접근 횟수가 2,400회로서 기존 시나리오의 26,597,850회와 3,925,076회에 비해 상당히 감소했으며 효율적임을 입증했다. RDF 트리플 저장소가 멀티스레딩 기법으로 인한 영향을 최대한 받지 않도록 RDF 트리플 저장소 접근 횟수를 최소화시키는 방향으로 RDF 데이터를 순차적으로 등록과 삭제 즉, 동시에 작업하지 않은 채 일괄 처리 방식으로 수행했기 때문이다. 이로써 구축 시간이 상당히 걸리는 부분을 해결할 수 있었다. 또한 오프라인이 아닌 온라인으로 일괄 처리를 했으므로, 오프라인에서 수작업으로 해야 하는 번거로움을 없앨 수 있다. 특히 온라인 일괄 등록 방식을 지원하는 모든 RDF 트리플 저장소를 활용할 수 있으므로, 특정 RDF 트리플 저장소에 종속되지 않는 부가적인 효과를 기대할 수 있다.

MEDLINE 데이터처럼 파일인 원시 데이터 집합이 대용량이고, 중복된 URI 처리가 필요한 경우에 유용하며, 상당한 시간이 소요되는 링크드 데이터를 구축하는 과정의 대안이 될 수 있다.

V. 결론

본 연구에서는 링크드 데이터를 구축할 때 MEDLINE 데이터를 RDF 파일로 변환 및 병합해 RDF 트리플 저장소에 등록하는 과정에서 URI 중복 조회 절차를 효율화한 이중 일괄 등록 방법을 제안하였다. 이로써 다음과 같이 기대할 수 있다.

첫째, 링크드 데이터를 구축하는 과정에서 RDF 트리플 저장소 접근을 최소화하기 위해 일괄 등록과 일괄 삭제 시 모두 순차적으로 처리하므로, RDF 트리플 저장소 적재 부하를 줄일 수 있는 대안이 될 수 있다.

둘째, RDF 파일 변환 과정에서 중복된 URI가 주어인 RDF 데이터를 DBMS에 저장하므로 최종적인 RDF 데이터를 보관하고 있다. 따라서 제안한 방법으로 하면 RDF 트리플 저장소에

적재된 RDF 데이터가 최신임을 보장할 수 있다.

셋째, 주어가 URI인 RDF 데이터나 URI를 DBMS에 저장하는 방식으로 확장하면, DBMS의 SQL과 SPARQL을 사용해 원시 데이터 집합과 RDF 트리플 저장소의 데이터 간 정합성을 검증할 수 있다.

넷째, DBMS에 저장된 정보와 변환한 RDF 파일을 모두 원시 데이터 집합으로 볼 수 있다. 전문 검색에 취약한 RDF 트리플 저장소 대신에 검색의 편의성을 위해 전문검색엔진을 사용할 경우 색인기로 색인하는 과정에서 XML 파일 대신에 RDF 파일을 활용하므로 그만큼 처리 시간을 단축할 수 있다.

본 연구의 실험 과정과 결과를 토대로 향후에 다음과 같이 URI 명명 규칙의 한계를 보완하고, 중복된 URI가 주어인 최신 RDF 데이터 자동 관리 및 다른 RDF 트리플 저장소 적용을 연구하고자 한다.

첫째, RDF 파일 변환 과정에서 MEDLINE 데이터의 특성상 정제되지 못한 데이터인 경우 저널 정보에 ISSN이 없거나 이슈 정보에 권호수가 존재하지 않으면 사전에 정의한 URI 명명 규칙에 따라 고유키를 부여한다. RDF 파일로 다시 변환하는 과정에서 기존 저널 URI나 이슈 URI와 일치하지 않아 RDF 트리플 저장소에 신규 등록한다. 결국 기존 URI이 주어인 RDF 데이터는 의미 없는 데이터로 남는다. 이것은 URI 명명 규칙의 한계이다. 따라서 변환 정보가 들어 있는 DBMS와 연계하는 방법 등으로 새로운 URI로 대치해 RDF 데이터를 등록하거나 기존 URI을 유지하되 해당 RDF 데이터를 최신으로 유지하는 방법으로 URI 명명 규칙을 보완해 연구해야 한다.

둘째, MEDLINE 데이터에서 중복된 것이 저널과 이슈임을 사전에 알고 있기 때문에 저널과 이슈가 주어인 URI의 최신 RDF 데이터를 관리할 수 있었다. 그 반면, MEDLINE 데이터가 아닌 다른 대용량 데이터 집합을 대상으로 링크드 데이터를 구축할 때 중복된 URI의 클래스가 다수일 수 있다. 이때 PUBMED_URI 테이블에 컬럼들을 일일이 추가해야 하므로 관리가 용이하지 않을 수 있다. 따라서 PUBMED_URI 테이블의 컬럼들을 고정시키는 방향으로 재설계한 후, 중복된 URI가 주어인 최신 RDF 데이터를 자동으로 관리해야 할 것이다. 이를 위해 RDF 스키마 도출 과정에서 중복된 URI의 클래스 목록을 파악한 후, RDF 파일 변환 및 병합 과정에서 추출된 URI의 클래스와 비교해 관련 정보를 PUBMED_URI 테이블에 자동으로 저장하는 방안을 연구해야 한다.

셋째, 본 연구에서 실험했던 RDF 트리플 저장소 대신에 RDF 파일을 프로그램적으로 일괄 등록할 수 있는 API를 제공하는 다른 RDF 트리플 저장소인 Apache TDB 등으로 대치할 수 있다. 본 연구의 동일한 실험 환경과 결과를 기준으로 다른 RDF 트리플 저장소를 사용했을 때의 접근 횟수와 총 트리플 건수 등의 차이점이 없음을 확인하고, 해당 RDF 저장소의 적재 성능에 따라 링크드 데이터 구축 시간이 달라지는지 연구해야 한다.

참고문헌

김천중, 김기연, 윤종현, 임종태, 복경수, 유재수. 2014. 대규모 RDF 데이터의 분산 저장을 위한 동적 분할 기법.『정보과학회논문지』, 41(12): 1126-1135.

문현정, 성정환, 김영지, 우용태. 2007. 대용량 RDF 데이터의 처리 성능 개선을 위한 효율적인 저장구조 설계 및 구현.『한국전자거래학회지』, 12(3): 251-268.

전명중, 홍진영, 박영택. 2016. SparQLing : SparkSQL 기반 대용량 트리플 데이터를 위한 SPARQL 질의 시스템 구축.『정보과학회논문지』, 43(4): 450-459.

정준원, 정호영, 김종남, 임동혁, 김형주. 2005. RDF 기반의 온톨로지 처리시스템.『정보과학회논문지 : 컴퓨팅의 실제 및 레터』, 11(4): 381-392.

한국정보화진흥원. 2014. 2014「링크드 오픈 데이터 국내 구축 사례집」. 서울: 한국정보문화진흥원.

Berners-Lee, Tim. 2006. Linked Data, <https://www.w3.org/DesignIssues/LinkedData.html> [citied 2017. 8. 7].

NIH. 2017. Fact Sheet MEDLINE, PubMed, and PMC(PubMed Central): How are they different?, <https://www.nlm.nih.gov/pubs/factsheets/dif_med_pub.html> [cited 2017. 8. 7].

Oliver E, Bhalotia G, Schwartz AS, Altman RB, Hearst MA. 2004. "Tools for loading MEDLINE into a local relational database." *BMC Bioinformatics*, 5(1): 146.

Zhiyong Lu. 2011. PubMed and beyond: a survey of web tools for searching biomedical literature. Database, 2011.

Chen, B., Ding, Y., Wang, H., Wild, D. J., Dong, X., Sun, Y., & Sankaranarayanan, M. 2010. "Chem2bio2rdf: A Linked Open Data Portal for Systems Chemical Biology." *In Web Intelligence and Intelligent Agent Technology* (WI-IAT), 1: 232-239.

Kilicoglu, H., Fiszman, M., Rodriguez, A., Shin, D., Ripple, A., & Rindflesch, T. C. 2008. Semantic MEDLINE: a web application for managing the results of PubMed Searches, in: Proc. 3rd International Symposium in Semantic Mining in Biomedicine, European Bioinformatics Institute, Hinxton, 2008: 69-76.

Lin, J., 2009. "Is searching full text more effective than searching abstracts?." *BMC bioinformatics*, 10(1): 46.

Castro, L.J.G., McLaughlin, C. and Garcia, A., 2013. "Biotea: RDFizing PubMed Central in support for the paper as an interface to the Web of Data." *Journal of biomedical semantics,* 4(1): S5.

국한문 참고문헌의 영문 표기
(English translation / Romanization of reference originally written in Korean)

National Information Society Agency. 2014. *2014 domestic case study of linked open data construction.* Seoul: Nation Information Society Agency.

Mun Hyeon Jeong, Sung Jung Hwan, Kim Young Ji and Woo Yong Tae. 2007. "A Design and Implementation of Efficient Storage Structure for a Large RDF Data Processing." *The Jounal of Society for e-Business Studies,* 12(3): 251-268.

Jun-Won Jung, Ho-Young Jung, Jong-Nam Kim, Dong-Hyuk Lim, Hyoung-Joo Kim. 2005. "A RDF based Ontology Management System." *Journal of KIISE : Computing Practices and Letters,* 11(4): 381-392.

MyungJoong Jeon, JinYoung Hong and YoungTack Park. 2016. "SPARQL Query Processing System over Scalable Triple Data using SparkSQL Framework." *Journal of KIISE,* 43(4): 450-459.

Cheon Jung Kim, Ki Yeon Kim, Jong Hyeon Yoon, Jong Tae Lim, Kyoung Soo Bok, Jae Soo Yoo. 2014. "A Dynamic Partitioning Scheme for Distributed Storage of Large-Scale RDF Data." *Journal of KIISE,* 41(12): 1126-1135.