Journal of The Korea Society of Computer and Information
Vol. 23 No. 10, pp. 45-55, October 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.10.045

R2RML Based ShEx Schema

Ji-Woong Choi*

Abstract

R2RML is a W3C standard language that defines how to expose the relational data as RDF triples. The output from an R2RML mapping is only an RDF dataset. By definition, the dataset has no schema. The lack of schema makes the dataset in linked data portal impractical for integrating and analyzing data. To address this issue, we propose an approach for generating automatically schemas for RDF graphs populated by R2RML mappings. More precisely, we represent the schema using ShEx, which is a language for validating and describing RDF. Our approach allows to generate ShEx schemas as well as RDF datasets from R2RML mappings. Our ShEx schema can provide benefits for both data providers and ordinary users. Data providers can verify and guarantee the structural integrity of the dataset against the schema. Users can write SPARQL queries efficiently by referring to the schema. In this paper, we describe data structures and algorithms of the system to derive ShEx documents from R2RML documents and presents a brief demonstration regarding its proper use.

▸Keyword: ShEx, SHACL, RDF, SPARQL, Direct Mapping, R2RML

I. Introduction

W3C는 지난 10여 년간 데이터의 웹 (the Web of Data) 즉, 웹상에 데이터베이스를 구축하기 위한 제반 기술들의 권고안을 발표해 옴과 동시에 이를 확산시키기 위한 LOD(Linking Open Data)[1] 프로젝트를 진행해 오고 있다. 데이터의 웹에서 데이터 표현 모델의 표준은 RDF[2]이며 표준 질의어는 SPARQL[3]이다. RDF 모델은 트리플(주어-술어-목적어)들의 집합으로 정의되며 하나의 트리플을 노드-아크-노드로 해석하는 단순한 그래프 모델이다. 이 모델은 서로 다른 트리플에 출현하는 동일한 노드를 모두 중첩시킴으로써 그래프를 확장해 나간다. RDF의 데이터의 표현은 단순하지만 모델의 확장에 있어서는 유연하다는 특성은 기존의 다양한 데이터 모델로 표현된 데이터를 데이터의 웹에서 하나의 통일된 방식으로 표현하기 위한 데이터 모델에 요구됐던 특성에 기인한다. 기존 데이터 모델에 대한 RDF 표현을 용이하게 할 목적으로 발표한 W3C의 대표적인 권고안으로는 Direct Mapping[4], R2RML[5], CSV2RDF[6] 등이 있다. [4,5]는 RDB, [6]은 CSV 데이터를 RDF로 표현하기 위한 규격과 절차에 관한 명세이다.

RDB에서 스키마의 역할은 두 가지로 요약할 수 있다. 첫째, 무결성 제약조건의 집합으로서의 스키마이다. 이것은 RDBMS로 하여금 데이터베이스가 항상 무결한 상태이도록 하는 기준으로서의 역할이다. 둘째, 데이터베이스 구조에 대한 청사진으로서의 스키마이다. 이것은 데이터베이스 구조를 설명하는 역할로서 사용자들의 효율적인 SQL 질의문 작성을 돕는다. RDF의 경우 최근까지 RDF 데이터셋에 대한 스키마 역할을 담당할 표준화된 기술이 존재하지 않았다. 이로 인해 데이터 공급자 입장에서는 RDF 데이터셋의 구조를 검증(validation)하는데 있어서 어려움을 겪었으며 데이터 사용자 입장에서는 원하는 데이터를 획득하기 위한 정확한 SPARQL 질의문을 완성하기까지 상당한 인내가 필요하였다[7-9]. 이러한 문제로 인하여 W3C는 2017년에 RDF 데이터셋에 대한 스키마를 명세할 수 있는 언어인 SHACL(Shapes Constraint Language)[10]을 개발하였다. SHACL은 앞서 언급한 스키마의 첫 번째 역할에 해당하는 무결성 제약조건 집합으로서의 스키마를 작성할 수 있도록 고안된 언어이다. 따라서 SHACL 문서에 명세한 제약조건을 기준으로 RDF 그래프의 구조를 검증(validation)할

• First Author: Ji-Woong Choi, Corresponding Author: Ji-Woong Choi
*Ji-Woong Choi (iamjwchoi@gmail.com), School of Computer Science and Engineering, Soongsil University
• Received: 2018. 09. 03, Revised: 2018. 10. 01, Accepted: 2018. 10. 02.
• The source code of this work is hosted on https://github.com/jwchoi/OWL4RDB/tree/ShExForR2RML2.

수 있게 되었다. 그러나 SHACL을 앞서 언급한 스키마의 두 번째 역할로 사용하기에는 구문이 간결치 못하다는 단점이 있다. 그 이유는 SHACL 문서가 RDF 문서로 간주되도록 설계되었기 때문이다. 즉, SHACL에서는 하나의 제약조건을 기술하는데 하나의 트리플이 소요된다. 때문에 복수개의 제약조건이 동일 대상에 대한 것이라 하더라도 복수개의 분리된 문장으로 기술해야한다. 이러한 신택스는 RDF 그래프의 구조를 신속하게 파악하고자 하는 인간의 입장에서 볼 때 적합하지 않다. W3C는 SHACL의 이러한 단점에 대한 대안으로 인간이 이해하기에 보다 간결한 구문을 제공하는 ShEx(Shape Expression)[11] 이라는 언어를 개발 중이다. ShEx는 현재 권고안으로 발표되지 않았을 뿐 신택스와 검증기(validator)는 개발된 상태이기 때문에 활용 가능한 수준에 와있다.

본 연구는 관계형 데이터베이스로부터 R2RML 문서의 명세에 따라 생성된 RDF 데이터셋에 대한 ShEx 스키마를 자동 생성하는 방법을 제안한다. R2RML 문서로부터 생성된 ShEx 스키마는 앞서 언급한 RDB 스키마의 두 가지 역할을 RDF 그래프에 대해 충실히 수행할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구와 배경 지식을 기술한다. 3장에서는 R2RML 문서를 분석하여 ShEx 스키마를 자동으로 생성 및 출력하는 과정과 방법을 상세히 기술한다. 4장에서는 3장의 내용을 구현한 시스템의 데모를 수행하며, 5장에서는 결론을 맺는다.

II. Related Works

본 장에서는 본 연구가 제안하는 시스템의 입력과 출력에 해당하는 R2RML과 ShEx 문서의 구조와 의미를 개략적으로 소개함으로써 본 장 이후에 대한 이해를 돕고자 한다.

1. R2RML: RDB to RDF Mapping Language

R2RML은 RDB에서 RDF로의 매핑을 정의할 수 있는 언어이다. R2RML 문서는 logical table들과 triples map들로 구성된다. logical table은 RDF로의 매핑 원천이 되는 RDB 데이터를 지정하는 부분으로서 테이블명, 뷰(view)명, SQL 질의문을 지정할 수 있다. logical table이 테이블 혹은 뷰일 경우 해당 테이블 혹은 뷰 데이터 전체가 매핑 대상이며, SQL 질의문일 경우 해당 질의문의 결과 셋이 매핑 대상이 된다. triples map은 RDF 트리플을 생성하는 규칙을 정의하는 부분이다. 1개의 triples map은 1개의 logical table, 1개의 subject map, 복수개의 predicate-object map으로 구성된다. triples map 구성요소로서의 logical table은 해당 triples map이 정의한 트리플 생성 규칙을 적용할 대상으로 선정된 RDB 데이터이다. triples map 내의 logical table은 미리 정의된 하나의 logical table을 참조하는 형식 혹은 triple map 내부에서 직접 정의하는 형식 모두 가능하다. triples map의 매핑 규칙은 선정된 logical table의 각각의 행에 대하여 적용된다. 각각의 행으로부터 생성될 트리플의 주어는 subject map에 정의된 생성 규칙을 따르며 술어와 목적어는 predicate-object map에 정의된 생성 규칙을 따른다. 따라서 1개의 행으로부터 생성되는 트리플의 개수는 predicate-object map의 개수와 동일하지만 이들은 모두 공통된 주어를 갖게 된다.

트리플의 주어는 IRI 혹은 blank node이다. IRI 주어를 생성하고자 할 때는 subject map 정의 시 IRI 템플릿을 명시해야 한다. 템플릿에는 logical table의 컬럼명이 포함되며 RDF 생성 시 컬럼명은 컬럼값으로 치환된다. predicate-object map은 predicate map과 object map으로 구성된다. 트리플의 술어는 IRI 만이 가능하다. 따라서 predicate map 정의는 단순히 술어로 사용할 IRI를 직접 명시하면 된다. 트리플의 목적어는 리터럴(literal), blank node, IRI 모두 가능하다. 리터럴 생성을 위한 object map 정의는 직접 리터럴을 명시하는 방법과 컬럼명을 명시하는 방법이 있다. 후자는 RDF로 매핑될 때 각각의 행에서 취한 컬럼값을 목적어 리터럴로 사용하는 반면 전자는 문서에서 명시한 리터럴을 목적어로 사용하므로 모든 행 변환에서 항상 같은 목적어가 생성된다. IRI 목적어 생성을 위한 object map은 두 가지 방법이 가능하다. 첫 번째는 컬럼명이 포함된 IRI 템플릿을 명시하는 방법이다. 이것은 IRI 주어를 생성하는 subject map 정의와 동일하다. 즉, 템플릿 내 컬럼명이 컬럼값으로 치환된 IRI가 목적어가 된다. 두 번째는 다른 triples map 정의에 의해 생성된 주어 IRI를 현재 생성코자 하는 트리플의 목적어로 취하기 위한 방법이다. 이러한 object map을 referencing object map이라고 하며 1개의 parent triples map과 복수개의 join condition으로 구성된다. parent triples map에서는 참조할 triples map을 명시하며 join condition에서는 현재 triples map의 logical table과 parent triples map의 logical table에 존재하는 컬럼명을 하나씩 짝지어 준다. 이러한 명세에 의해 RDF 변환 시 두 logical table을 실제로 조인함으로써 행들의 쌍을 구한다. 마지막으로 parent triples map에 의해 생성된 트리플의 주어가 blank node라면 그 경우의 referencing object map은 blank node 생성을 위한 object map이 될 수 있다.

그림 1은 RDF 1.1 Turtle[12]로 표현된 R2RML 문서의 일부다. R2RML 문서는 RDF 문서가 되도록 설계되었다. 그림 1에서 <#TriplesMap1>과 <#TriplesMap2>가 triples map이다. <#TriplesMap2>는 일부만 그림에 포함되어 있다. <#EmpTableView>는 logical table이며 SQL 질의문이 정의되어 있다. <#EmpTableView>는 <#TriplesMap1>의 logical table로 사용되고 있다. <#TriplesMap2>은 내부에서 테이블 DEPT를 logical table로 정의하고 있다. <#TriplesMap1>의 subject map은 컬럼 EMPNO를 포함한 IRI 템플릿을 명시하고 있다. ex:name을 술어로 컬럼 ENAME을 목적어로 명시한 predicate-object map은 컬럼값을 목적어 리터럴로 사용하는 예이다. ex:job을 술어로 컬럼 JOB을 포함하는 템플릿을 목적어로 명시한 predicate-object map은 컬럼값을 치환하여 생성한 IRI를 목적어로 취하는 예이다. ex:department를 술어로 명시한 predicate-object map은 다른 tiples map의 주어를 목적어로 취하는 예이다. 이 예에서는

<#TriplesMap2>의 주어를 목적어로 취한다. 그림 1에서 rr:child의 목적어인 “DEPTNO”는 <#TriplesMap1>의 logical table인 <#EMPTableView>의 컬럼이며 rr:parent의 목적어인 “DEPTNO”는 <#TriplesMap2>의 logical table인 DEPT 테이블의 컬럼이다.

```
<#EmpTableView> rr:sqlQuery """
SELECT EMPNO, ENAME, JOB, DEPTNO FROM EMP;
""".
<#TriplesMap1>
 rr:logicalTable <#EmpTableView>;
 rr:subjectMap [
  rr:template "http://ex.com/emp/{EMPNO}";
  rr:class ex:Employee;
 ];
 rr:predicateObjectMap [
  rr:predicate ex:name;
  rr:objectMap [ rr:column "ENAME" ];
 ];
 rr:predicateObjectMap [
  rr:predicate ex:job;
  rr:objectMap [rr:template "http://ex.com/job/{JOB}"];
 ];
 rr:predicateObjectMap [
  rr:predicate ex:department;
  rr:objectMap [
   rr:parentTriplesMap <#TriplesMap2>;
   rr:joinCondition [
    rr:child "DEPTNO";
    rr:parent "DEPTNO";
   ];
  ];
 ].
<#TriplesMap2>
 rr:logicalTable [ rr:tableName "DEPT" ];
```

Fig. 1. Example of R2RML Document

2. ShEx: Shape Expressions Language

하나의 ShEx 문서가 표현하고 있는 내용을 논리적 수준에서ShEx 스키마라고 한다. ShEx 스키마는 어떤 RDF 그래프의 구조에 대한 설명이자 제약조건이다. ShEx 스키마는 셰이프 표현식(Shape Expression)의 모음이다. 셰이프 표현식은 노드 제약조건(node constraint)과 셰이프 제약조건(shape constraint)으로 분류할 수 있다. 셰이프 제약조건은 간략히 셰이프(shape)라고도 한다.

노드 제약조건은 RDF 노드의 종류와 값의 범위를 제약한다. 노드의 종류는 IRI, blank node, 리터럴 중 하나일 수 있다. ShEx는 리터럴 노드에 한하여 데이터타입을 추가적으로 제약할 수 있도록 한다. 이 때 사용되는 데이터타입은 XML 스키마 데이터타입[13]이다. 이는 RDF가 XML 스키마 데이터타입을 리터럴을 위한 데이터타입으로 차용하기 때문이다. 노드 제약조건은 IRI와 리터럴 노드에 대하여 값의 범위를 제약할 수 있는 facet이라는 부속 제약조건을 포함할 수 있다. IRI 노드의 경우 IRI 문자열 패턴을 제약하는 facet이 있다. ShEx에서는 이를 위해 XPath 정규 표현식[14]를 사용한다. 리터럴 노드의 경우 데이터타입에 따라 사용할 수 있는 facet이 달라진다. 예를 들어, 문자열 리터럴에 대해서는 문자열의 길이와 패턴을 제약할 수 있다. 숫자형 리터럴에 대해서는 허용되는 자릿수 또는 이상, 이하 등에 해당하는 ShEx 어휘를 활용한 값의 범위에 대한 제약이 가능하다. 마지막으로, ShEx에서는 열거형 표현식 facet이 가능하며 이 facet은 IRI 그리고 모든 타입의 리터럴 노드 제약조건에서 사용할 수 있다.

그림 2는 샘플 ShEx 스키마이다. 그림 2에서 첫 번째 라인은 하나의 노드 제약조건이다. ex:age는 노드 제약조건의 레이블이라고 하며 문서 내에서 식별자 역할을 한다. 레이블이 있는 노드 제약조건은 그림 2에서처럼 @ex:age 형식으로 문서 내에서 참조될 수 있다. 이 노드 제약조건은 어떤 노드가 13이상 20이하의 정수 리터럴이어야 함을 설명한다.

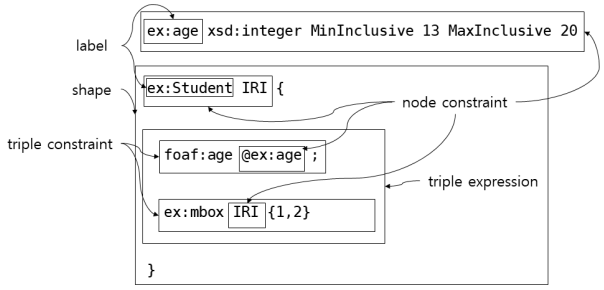

Fig. 2. Example of ShEx Schema

셰이프는 어떤 노드를 중심으로 그 노드가 어떤 구조 속에 있어야 하는지 까지 제약한다. 그림 2에서 2~5 라인이 하나의 셰이프이다. ex:Student는 셰이프의 레이블로서 @ex:Student 형식으로 다른 셰이프에 의해 참조될 수 있다. 중괄호 이전 까지는 노드 제약조건으로 해석될 수 있다. 즉, 중괄호 이전 파트에서는 어떤 노드의 특성만을 제약하며 중괄호 파트에서는 그 노드의 그래프 내에서의 형상을 제약한다. 중괄호 파트는 트리플 표현식(tripple expression)이라고 한다. 그림 2의 트리플 표현식은 2개의 트리플 제약조건(tripple constraint)으로 구성되어 있다. 이 셰이프는 IRI인 어떤 노드가 foaf:age 프로퍼티에 의해 ex:age 노드 제약조건을 만족하는 노드 1개와 주어로서 연결되어야 하며 ex:mbox 프로퍼티에 의해 IRI이기만 하면 되는 노드 1~2개와 주어로서 연결되어야 함을 의미한다. 트리플 제약조건의 마지막 요소는 cardinality이며 cardinality가 명시되지 않았을 시는 1을 의미한다.

III. The Proposed Method

본 장에서는 R2RML 문서의 명세에 따라 생성된 RDF 그래프에 대한 ShEx 스키마를 생성하기 위한 절차를 단계별로 상세히 기술한다. 본 연구에서는 R2RML 모델 구축 단계, ShEx 스키마 모델 구축 단계, ShEx 파일 출력 단계를 통해 ShEx 파일을 생성한다.

3.1 Phase I: Building R2RML Model

R2RML 모델은 R2RML 문서의 의미 구조를 표현하는 모듈이다. R2RML 문서는 RDF 문서이다. 따라서 RDF 수준에서의 문서 파싱 및 트리플 단위의 탐색을 바탕으로 파악한 문서의 의미 구조를 시스템에서 복원한 것이 R2RML 모델이다. R2RML 모델은 구축과

정에서 RDBMS에도 접근한다. 목적은 R2RML 문서에서 사용된 컬럼의 SQL 데이터타입과 널(null) 허용여부 정보를 수집하기 위함이다. 컬럼의 SQL 데이터타입은 노드 제약조건 정의 시 사용될 리터럴의 데이터타입을 도출하는데 필요하다. 컬럼의 널 허용여부는 트리플 제약조건의 cardinality를 결정하는데 사용된다.

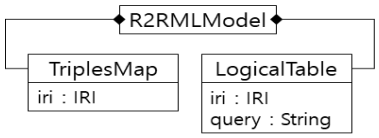

Fig. 3. Structure of R2RML Model

그림 3은 R2RML 모델의 구조이다. R2RMLModel이 R2RML 모델에 해당하는 클래스이다. R2RMLModel은 R2RML 문서에서 참조 가능하도록 선언된 triples map과 logical table에 대응하는 TriplesMap과 LogicalTable 클래스의 집합으로 구성된다. TriplesMap과 LogicalTable은 문서에 명세된 참조를 위한 식별자 IRI를 속성으로 갖는다. LogicalTable은 SQL 질의문을 속성으로 갖는다. R2RML 문서에 테이블 혹은 뷰 이름이 명세되었어도 결과적으로 RDF와 ShEx 생성을 위해 필요한 것은 그 이름의 모든 데이터를 가져오는 SQL 질의문이기 때문이다.

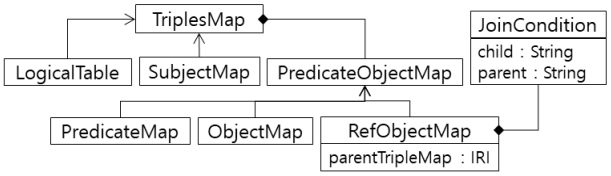

Fig. 4. Structure of Triples Map

그림 4는 TriplsMap 클래스의 구조이다. 이 구조는 R2RML 문서에서의 triples map에 대한 의미 구조를 따른다. TriplesMap은 각각 1개씩의 LogicalTable과 SubjectMap 그리고 복수개의 PredicateObjectMap에 대한 참조를 갖는다. PredicateObjectMap에는 1개씩의 PredicateMap, ObjectMap, RefObjectMap에 대한 참조가 선언되어 있다. 다만, predicate object map에는 하나의 object map만이 선언될 수 있으므로 하나의 PredicateObjectMap은 ObjectMap 혹은 RefObjectMap 중에서 하나의 참조만 갖는다. RefObjectMap은 참조하는 triples map의 IRI를 저장하기 위한 parentTripleMap 속성과 join condition 목록을 저장하기 위한 속성으로 구성되어 있다. JoinCondition 클래스는 child 컬럼명과 parent 컬럼명을 저장하기 위한 속성을 갖는다.

subject map, predicate map, object map의 정의부에는 공통적으로 rr:constant, rr:column, rr:template, rr:termType, rr:language, rr:datatype 이라는 R2RML 술어들이 출현할 수 있다. 이러한 특성을 논리적으로 표현하기 위해 R2RML 사양은 이들 세 개의 map에 대한 상위 개념으로서 term map을 정의하고 있다. term map 술어들의 사용 가능한 목적어는 rr:termType의 경우 rr:IRI, rr:BlankNode, rr:Literal 중 하나, rr:constant의 경우 rr:termType을 준수하는 상수값, rr:column의 경우 컬럼명, rr:template의 경우 컬럼명을 포함한 템플릿 문자열, rr:language의 경우 문자열 리터럴에 대한 언어 태그, rr:datatype의 경우 XML 스키마 데이터타입이다. 본 연구에서는 이러한 R2RML 사양의 논리 구조를 반영하여 SubjectMap, PredicateMap, ObjectMap이 TermMap 클래스를 상속하도록 설계하였다. PredicateMap과 ObjectMap 클래스는 추가적인 속성이 존재하지 않지만, SubjectMap은 rr:class 술어의 목적어를 저장할 속성이 추가되어 있다. rr:class 술어의 목적어는 RDF 변환 시 생성된 IRI 주어에 대한 rdf:type 술어의 목적어가 된다.

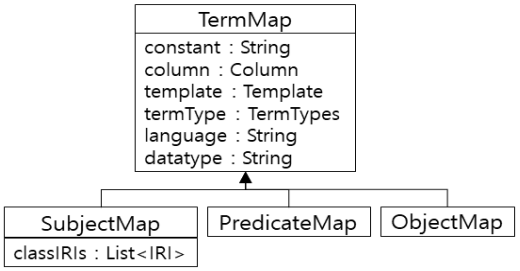

Fig. 5. Structure of Term Map

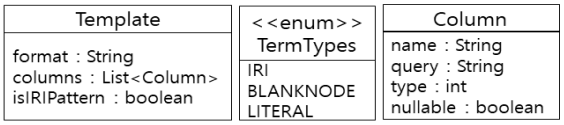

Fig. 6. Classes for Term Map

그림 6은 TermMap에서 참조하는 클래스들이다. 열거형 TermTypes에 선언된 상수들은 각각 rr:IRI, rr:BlankNode, rr:Literal에 대응한다. Column 클래스는 컬럼명 뿐 아니라 컬럼명의 출처인 SQL 질의문 그리고 SQL 데이터타입과 널 허용여부를 속성으로 한다. Template 클래스의 속성은 템플릿 문자열, 템플릿 문자열 내에 포함된 컬럼들의 정보 그리고 템플릿 문자열이 IRI 문자열을 생성해 내는지 아닌지 여부로 구성된다.

3.2 Phase 2: Building ShEx Schema Model

이 절에서는 R2RML 모델을 입력으로 하여 생성되는 ShEx 스키마 모델의 구조와 생성절차를 기술한다. 그림 7은 ShEx 스키마 모델의 구조를 묘사하는 클래스 다이어그램이다. 클래스 ShExSchema가 ShEx 스키마 모델에 해당하는 클래스이다. ShEx 스키마의 정의대로 ShExSchema 또한 노드 제약조건에 해당하는 NodeConstraint 클래스와 셰이프 제약조건에 해당하는 Shape 클래스의 모음으로 구성된다. Shape 클래스는 트리플 제약조건에 해당하는 TripleConstraint 클래스의 집합을 속성으로 갖도록 설계하였다.

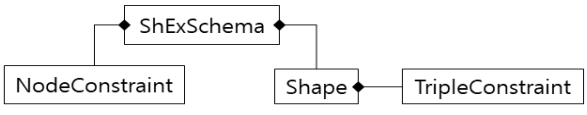

Fig. 7. Structure of ShEx Schema Model

Table 1. Input for Building ShEx Schema Model

Output	Input	
ShEx Schema Model	R2RML Model	ShEx Schema Model
NodeConstraint	ObjectMap	
Shape (Type 1)	IRI of TriplesMap, SubjectMap	
Shape (Type 2)		set of Type 1 Shapes
TripleConstraint (Type I)	classIRIs of SubjectMap	
TripleConstraint (Type 2)	PredicateMap, ObjectMap	
TripleConstraint (Type 3)	PredicateMap, RefObjectMap	

표 1은 ShEx 스키마 모델을 구성하는 클래스들의 객체가 R2RML 모델의 어떠한 요소를 입력으로 하여 생성되는지를 요약한다. NodeConstraint 객체는 R2RML 모델의 ObjectMap 객체와 1:1 대응이다. 즉, 본 연구의 최종 산출물인 ShEx 문서에 존재하는 레이블이 있는 노드 제약조건은 모두 목적어 노드를 설명한다. Shape 객체는 R2RML 모델의 TriplesMap의 식별자인 IRI와 그 TripesMap의 SubjectMap으로부터 생성된다. 사실상 TriplesMap과 Shape의 대응수는 1:1이다. TriplesMap의 IRI는 Shape의 레이블 생성을 위해 사용되며 SubjectMap은 Shape의 중괄호 이전 파트 즉, 주어 노드에 대한 제약조건 생성을 위해 사용된다. Type 1 TripleConstraint 객체는 의미상 R2RML 문서에 있는 rr:class 술어의 목적어 IRI로부터 생성되는 것이다. 이 TripleConstraint는 묵시적으로 술어가 rdf:type으로 결정된 것이기 때문에 트리플 제약조건의 목적어 생성에 필요한 rr:class 술어의 목적어 IRI만을 필요로 한다. 표 1에서 Type 2와 3 TripleConstraint 생성을 위한 입력은 R2RML 문서에 있는 각각의 predicate-object map으로부터 하나씩의 TripleConstraint가 생성됨을 보여주는 것이다.

표 1에서 Type 2 Shape의 경우 입력이 R2RML 모델 요소가 아닌 Type 1 Shape의 집합이다. 우선 Type 2 Shape의 존재 이유를 먼저 설명하고자 한다. 서로 다른 triples map에 의해 생성된 트리플들 사이에 동일한 주어를 공유하게 되는 경우가 있다. 즉, 하나의 주어를 공유하는 트리플들이 여러 triples map으로부터 생성된 경우이다. 이러한 상황은 서로 다른 triples map의 subject map에 정의된 템플릿들이 컬럼명만 제외하면 동일할 경우에 컬럼 값에 따라 발생할 수 있다. 본 연구에서는 (Type 1) Shape을 triples map에 일대일로 대응시켰기 때문에 이러한 경우의 트리플들에 대한 구조를 설명할 수 있는 Shape은 존재하지 않게 된다. 예를 들어, triples map의 집합 {A, B, C}의 원소들의 subject map 템플릿이 컬럼명만 제외하고 동일하다면, {A}, {B}, {C}, {A, B}, {B, C}, {C, A}, {A, B, C} 라는 각각의 부분집합으로부터 동일한 주어가 발생할 수 있다. 원소의 개수가 1인 부분집합으로부터 생성된 트리플의 구조는 Type 1 Shape으로 설명할 수 있다. 그 외의 부분집합으로부터 생성된 트리플의 구조를 설명하기 위한 Shape이 추가로 필요하며 이러한 용도의 Shape이 바로 Type 2 Shape이다. 본 연구에서는 Type 2 Shape의 생성을 위한 입력으로 주어를 공유케 하는 triples map의 집합 대신 이에 대응하는 Type 1 Shape의 집합을 사용하였다. 예를 들면, triples map A로부터 생성한 Type 1 Shape을 S^A라고 한다면, 부분집합 {A, B}로부터 생성된 트리플을 설명할 수 있는 Type 2 Shape을 생성하기 위한 입력으로서의 Type 1 Shape의 집합은 $\{S^A, S^B\}$이 된다.

```
function buildShExSchemaModel(r2rml)
input: R2RML Model r2rml
output: ShEx Schema Model schema
01  schema ← create an empty ShEx Schema Model.
02  TMs ← get the set of TriplesMaps in r2rml.
03  for each TM ∈ TMs {
04    iri ← get the IRI of TM.
05    SM ← get the subject map of TM.
06    shape1 ← create a type 1 shape with iri and
            SM.
07    clsIRIs ← get classIRIs of SM.
08    tc1 ← create a type 1 triple constraint with
          clsIRIs.
09    add tc1 to shape1.
10    POMs ← get the set of predicate-object maps.
11    for each POM ∈ POMs {
12      PM ← get the predicate map of POM.
13      OM ← get the object map of POM.
14      ROM ← get the referencing object map of
15            POM.
16      if OM != null {
17        nc ← create a node constraint with OM.
18        add nc to schema.
19        tc2 ← create a type 2 triple constraint with
              PM and OM.
20        add tc2 to shape1.
21      }
22      if ROM != null {
23        tc3 ← create a type 3 triple constraint with
              PM and ROM.
24        add tc3 to shape1.
25      }
26    }
27    add shape1 to schema.
28  }
29  IRISet ← create an empty set for IRI of TriplesMap.
30  for each TM ∈ TMs {
31    if IRI of TM ∉ IRISet {
32      shape1 ← find the type 1 shape created from
                TM.
33      shape1s ← get a set of type 1 shapes to
                  share subjects from schema.
34      add the IRIs of every shape in shape1s to
        IRISet.
35      subsets ← get a set of subsets contain more
                  than 2 elements of shape1s.
36      for each subset ∈ subsets {
37        shape2 ← create a type 2 shape with
                   subset.
38        add shape2 to schema.
39      }
40    }
41  }
42  return schema
```

Fig. 8. Algorithm for Building ShEx Schema Model

그림 8은 ShEx 스키마 모델을 구축하는 과정을 보여준다. 3~28 라인에서 Type 2 Shape을 제외한 모든 ShEx 스키마 모델 구성요소를 생성한다. 4~9 라인에서는 TriplesMap 객체로부터 표 1에 기술된 필요한 입력 요소들을 추출한 후 Type 1 Shape과 Type 1 TripleConstraint 객체를 생성하고 TripleConstraint 객체를 Shape 객체에 추가한다. 10 라인에서는 TriplesMap 객체로부터 PredicateObjectMap 객체의 집합을 추출한다. 16~21 라인은 이 집합의 한 원소인 PredicateObjectMap 객체가 ObjectMap 객체를 소유했을 경우의 처리로서 이때는 NodeConstraint 객체와 Type 2 TripleConstraint 객체를 생성하고 NodeConstraint 객체는 ShExSchema 객체에 TripleConstraint 객체는 Shape 객체에 추가한다. 22~25 라인은 그 PredicateObjectMap 객체가 RefObjectMap 객체를 소유했을 경우를 처리한다. 이때는 Type 3 TripleConstraint 객체를 생성하고 Shape 객체에 추가한다. 라인 27에서는 1개의 TriplesMap으로부터 생성되어 TripleConstraint 객체들이 모두 추가된 Shape 객체를 ShExSchema 객체에 추가한다.

Type 2 Shape 생성을 위한 처리는 다음과 같다. 32라인에서 TriplesMap 객체로부터 생성된 Type 1 Shape 객체를 ShEx Schema 객체에 질의하여 찾아낸다. 33라인에서 ShEx Schema 객체에 질의하여 그 Shape 객체가 설명할 수 있는 RDF 주어 노드를 함께 설명할 수 있는 모든 Type 1 Shape들의 집합을 얻어낸다. 이 집합에는 질의된 Shape 객체도 포함되어 있다. 이 집합에 포함된 Shape이 다시 질의될 경우 동일한 집합이 다시 얻어지므로 이를 방지하기 위한 장치가 29 라인에서 선언된 IRISet이다. 34 라인에서는 33라인에서 얻은 집합의 원소 Shape들로부터 자신을 생성케 한 TriplesMap의 IRI를 추출하여 모두 IRISet에 등록해 둠으로써 동일한 집합이 얻어지지 않게 한다. 35 라인에서는 33 라인에서 얻은 집합에 대한 원소 개수가 2 이상인 모든 부분집합을 구해낸다. 각각의 부분집합을 입력으로 하여 37 라인에서 Type 2 Shape 객체를 생성, 38 라인에서 ShEx Schema 객체에 추가한다. Type 2 Shape 객체는 생성 시 입력된 Type 1 Shape 객체들 소유의 TripleConstraint 객체들의 합집합을 구한 후 이를 자신 소유의 TripleConstraint 집합으로 설정한다.

3.2.1 Setting Attributes of NodeConstraints

본 절에서는 ObjectMap 객체를 기반으로 NodeConstraint 객체가 속성값을 결정하는 방법을 설명한다. 표 2는 NodeConstraint의 속성 목록임과 동시에 각 속성이 ObjectMap의 어떤 속성값을 바탕으로 생성되는지를 요약한다.

Table 2. Attributes of NodeConstraint

NodeConstraint	ObjectMap
id	
nodeKind	termType
valueSet	language
datatype	datatype
xsFacet	template

id 속성값은 출력단계에서 노드 제약조건의 레이블이 된다. id 속성값은 ObjectMap에 의존하지 않고 유일한 값이 되도록 시스템에서 생성한 값이다. nodeKind 속성값은 BNODE, IRI, LITERAL 중 하나이며 termType 속성값인 BLANKNODE, IRI, LITERAL에 각각 대응한다. valueSet 속성은 language 속성값 앞에 “@” 문자를 덧붙인 결과를 취한다. 예를 들어, language 속성값이 “en-us”라면 ShEx에서 사용하는 언어태그 형식인 “@en-us”으로 변환된다. 참고로, valueSet 속성은 출력 단계에서 ShEx 열거형 표현식을 생성한다. ShEx는 열거형으로 언어태그 제약을 표현할 수 있도록 한다. NodeConstraint의 datatype 속성값은 기본적으로 ObjectMap의 동명 속성에 값이 존재할 경우 그대로 취한다. 다만, ObjectMap의 column 속성값이 존재하고 datatype 속성값은 존재하지 않을 경우 column의 type 속성 즉, 그 컬럼의 SQL 데이터타입에 대응하는 XML 스키마 데이터타입을 NodeConstraint의 datatype의 값으로 취한다. SQL과 XML 간 데이터타입 대응관계는 R2RML 사양에 정의된 [15]를 따른 것이다. xsFacet 속성은 template 속성의 format 문자열을 ShEx에서 사용할 수 있는 정규표현식 신택스[14]로의 문자열 조작 후 그 결과를 값으로 할당받는다.

3.2.2 Setting Attributes of Shapes

Shape 객체의 속성은 표 3에 나열된 3개의 속성과 TripleConstraint 객체들의 집합으로 구성된다. 표 3의 3개의 속성은 출력단계에서 중괄호 이전 파트 즉, 주어 노드에 대한 노드 제약조건 파트 생성에 사용되며 TripleConstraint 집합은 중괄호 파트 생성에 사용된다. id 속성값은 IRI이며 출력단계에서 셰이프의 레이블이 된다. 본 연구에서는 Type 1 Shape의 id IRI와 TriplesMap의 IRI가 네임스페이스는 달리하지만 fragment는 동일도록 생성하였다. 사용자가 셰이프와 triples map의 대응관계를 용이하게 파악하도록 할 의도이다. Type 2 Shape의 id는 입력 Type 1 Shape들의 id fragment를 합성하였다. 이 또한 사용자가 Type 2 셰이프가 어떤 Type 1 셰이프들로부터 생성된 것인지를 쉽게 파악하도록 할 의도이다. nodeKind 속성값은 주어 노드에 대한 설명이기에 IRI 혹은 BNODE만 가능하며 SubjectMap의 termType 속성값 IRI, BLANKNODE에 각각 대응하여 결정된다. xsFacet 속성은 SubjectMap의 template 속성의 format 문자열을 정규표현식 신택스로의 문자열 조작 후 그 결과를 값으로 할당받는다. Type 2 Shape의 생성을 위해 입력된 Type 1 Shape들의 nodeKind와 xsFacet 속성값은 모두 같아야 한다. 왜냐하면 동일한 주어 노드를 설명할 수도 있는 셰이프들이기 때문이다. 따라서 그들로부터 생성된 Type 2 Shape 또한 두 속성의 값이 같아야 한다. 따라서 본 구현에서는 입력된 첫 번째 Shape의 두 속성값을 생성되는 Shape의 속성값으로 사용하였다.

Table 3. Attributes of Shape

NodeConstraint	Input of R2RML Model
id	IRI of TriplesMap
nodeKind	termType of SubjectMap
xsFacet	template of SubjectMap

3.2.3 Setting Attributes of TripleConstraints

TripleConstraint 클래스는 predicate, valueExpr, cardinality 속성으로 구성되어 있다. 출력단계에서 predicate은 트리플의 술어, valueExpr은 트리플의 목적어, cardinality는 구조에 부합하는 트리플의 개수를 설명하는 신택스 출력에 사용된다.

Type 1 TripleConstraint의 predicate는 “rdf:type”으로 결정된다. valueExpr은 생성 인자인 classIRIs 집합의 원소들을 취한다. 이 집합의 원소들은 출력단계에서 열거형 표현식의 원소로 출력된다. cardinality는 입력 IRI 집합의 크기를 값으로 취한다. 본 연구에서는 이 타입의 트리플 제약조건이 셰이프 내에서 한 번만 출력 되도록 설계하였다. 따라서 Type 2 Shape에 포함된 Type 1 TripleConstraint의 valueExpr은 생성 인자 Shape들의 Type 1 TripleConstraint의 valueExpr의 합집합이다.

Type 2 TripleConstraint의 predicate 속성값은 생성 인자 PredicateMap의 constant 속성값인 술어 IRI와 같다. valueExpr 속성값은 생성 인자 ObjectMap을 ShExSchema에 질의하여 그 ObjectMap으로부터 생성된 NodeConstraint의 template 값 유무에 따라 달라진다. template 값이 존재하면 NodeConstraint의 id를 획득한 후 “@id값” 형식의 문자열을 valueExpr의 값으로 할당한다. template 값이 존재하지 않으면 NodeConstraint의 문자열화 함수의 호출 결과 즉, 노드 제약조건 표현식 자체를 valueExpr의 값으로 할당한다. cardinality의 값 결정 방식은 ObjectMap의 상태에 따라 달라진다. 첫 번째로 ObjectMap의 column 속성에 값이 존재할 경우 그 컬럼이 null을 허용한다면 cardinalty는 “?”, 즉, 0 또는 1, 그렇지 않다면 “” 즉, 디폴트 1을 의미하는 길이 0인 문자열이다. 두 번째로 ObjectMap의 template 속성의 값이 존재할 경우 템플릿에 포함된 컬럼들 중 하나라도 null을 허용한다면 cardinality는 “?”, 모든 컬럼이 null을 허용하지 않는다면 “”이다.

Type 3 TripleConstraint의 predicate 속성은 생성 인자 PredicateMap의 constant 속성값인 술어 IRI를 그대로 취한다. valueExpr 속성값은 생성 인자 RefObjectMap의 속성인 parentTriplesMap을 ShExSchema에 질의하여 그 triples map IRI로부터 생성된 Shape 객체를 찾아내서 그 Shape의 id를 획득한 후 “@id값” 형식으로 조작 후 할당된다. 이 타입의 cardinality는 “*” 즉, 0개 이상이다. 이는 조인 결과에 따라 대응되는 행의 개수가 달라지기 때문이다.

3.3 Phase 3: Writing ShEx Schema Model

본 절에서는 이전 단계에서 구축한 ShEx 스키마 모델을 문서로 출력하기 위한 단계를 설명한다. ShEx 스키마 모델을 구성하는 클래스들에는 자신의 속성값을 ShEx 문법에 맞게 배열하여 ShEx 제약조건 문자열을 생성시키는 함수들이 정의되어 있다. 따라서, ShEx 문서는 ShExSchema의 문자열화 함수 호출 결과인 하나의 ShEx 스키마에 해당하는 문자열을 단순히 파일에 출력함으로써 획득된다. ShExSchema의 문자열화 함수는 내부에서 자신에게 등록된 NodeConstraint와 Shape 객체들을 1회씩 방문하여 그 객체들의 문자열화 함수를 호출함으로써 노드 제약조건과 셰이프 제약조건 문자열들을 획득한다. Shape 객체들의 문자열 함수 또한 내부에서 자신이 포함하고 있는 TripleConstraint 객체들의 문자열화 함수를 1회씩 호출하며 그 결과인 트리플 제약조건 문자열들을 셰이프 제약조건의 일부의 트리플 표현식 구성에 사용한다.

```
① id ‘[’ valueSetValue+ ‘]’ xsFacet
② id datatype xsFacet
③ id nodeKind xsFacet
```

Fig. 9. Available Node Constraint Syntaxes

	predicate	valueExpr	cardinality
Type 1:	a	‘[’ classIRI+ ‘]’	sizeOfClassIRIs
Type 2:	predicate	‘@’nid	‘?’|‘’
Type 2:	predicate	nodeConstraint	‘?’|‘’
Type 3:	predicate	‘@’sid	‘*’

Fig. 10. Available Triple Constraint Syntaxes

```
id nodeKind xsFacet? ‘{’
 ( tripleConstraint
 |
 ‘(’ tripleConstraint (‘|’ tripleConstraint)+ ‘)’
 )+
‘}’
```

Fig. 11. Shape Constraint Syntax

그림 9는 NodeConstraint가 생성할 수 있는 3가지 구조의 노드 제약조건 신택스이다. ①~③은 생성에 대한 우선순위이기도 하다. 즉, valueSet의 값이 존재하면 ①의 신택스를 생성한다. valueSet의 값은 존재하지 않지만 datatype의 값이 존재하면 ②를 생성한다. ①과 ②의 생성이 가능하지 않을 때 ③을 생성한다. ①에서 valueSetValue는 valueSet 내의 각각의 원소를 의미한다. ShEx에서 대괄호(‘[’‘]’) 부분은 열거형 표현식에 해당한다. ①~③ 모두 xsFacet의 값이 존재할 경우에만 위의 포맷으로 출력되며 그렇지 않을 경우 id와 xsFacet이 제거된 형태로 트리플 제약조건 내부에 출력된다.

TripleConstraint는 늘 “predicate valueExpr cardinality” 형식의 문자열을 출력한다. 그림 10은 이해를 돕기 위하여 타입에 따라 출력되는 각 속성의 값을 구체화 하였다. Type 1에서 ‘a’는 ‘rdf:type’과 동일한 의미이다. classIRI는 classIRIs 집합의 원소, sizeOfClassIRI는 classIRIs 집합의 크기를 의미한다. Type 2에서 nid는 노드 제약조건의 id를 의미하고 nodeConstraint는 id와 xsFacet이 없는 포맷의 노드 제약조건 표현식을 의미한다. Type 3에서 sid는 셰이프 제약조건의 id를 의미한다.

그림 11은 Shape이 생성하는 셰이프 제약조건 신택스이다. 그림에서 xsFacet은 주어 IRI의 정규 표현식이므로 nodeKind가 IRI인 경우에 한하여 출력한다. 중괄호 내부는 기본적으로 트리플 제약조건들을 나열함으로써 완성된다. 다만, 특정 조건

에 부합하는 Type 3 TripleConstraint의 경우 그림 11의 라인 4 표현식을 생성한다. 이 표현식의 형식은 괄호 안에 복수개의 트리플 제약조건을 OR('|') 연산자로 연결한다. 이 표현식은 타겟 트리플이 괄호안의 후보 트리플 제약조건들 중 최소 1개만 만족시켜도 표현식을 만족시킨 것으로 판정된다. ShEx에서는 이 표현식을 "multiElementOneOf" 트리플 표현식이라 한다. 이 표현식을 생성하는 Type 3 TripleConstraint의 조건은 valueExpr의 값 즉, 그림 10에서 sid가 의미하는 셰이프가 Type 2 Shape의 생성에 사용된 셰이프일 경우이다. 이 경우의 트리플 제약조건이 설명하고자 했던 트리플의 목적어는 RDF 생성 양태에 따라 sid 셰이프 혹은 sid 셰이프가 포함되어 생성된 Type 2 셰이프들 중 하나의 구조일 수 있다. 이러한 이유로 각각의 그 모든 셰이프들의 참조로 valueExpr 파트를 치환한 트리플 제약조건들로 구성된 multiElementOneOf 트리플 표현식을 생성하였다.

IV. Demonstration

본 장에서는 본 연구에서 제안한 방법을 구현한 시스템의 유용성을 데모를 통해 밝히고자 한다.

표 4는 ISWC로 명명한 학술대회 정보를 저장하고 있는 데이터베이스의 규모를 요약한다. 이 데이터베이스는 본 데모를 위해 MariaDB[16] 상에 구축하였다. 표 5는 이 데이터베이스로부터 RDF 데이터셋을 생성하는 규칙을 명세한 R2RML 문서 정보를 요약한다. 이 문서에는 9개의 triples map이 정의되어 있다. 표 5의 첫 번째 열에서 같은 셀에 묶여있는 triples map들은 주어를 공유하는 트리플들을 생성할 가능성이 있는 것들이다. 즉, Type 2 Shape을 생성케 하는 것들이다. 표 5의 두 번째 열은 각각의 triples map에 정의되어 있는 predicate object map의 개수이다. 표 5의 세 번째 열은 각각의 triples map에 정의되어 있는 predicate object map들 중에서 referencing predicate object map이 포함되어 있을 경우 그 referencing predicate object map 내에서 사용된 parent triples map의 이름이다.

Table 4. ISWC Database Summary

Table	Rows	Columns
conferences	2	6
organizations	9	10
papers	7	7
persons	10	10
rel_paper_topic	13	3
rel_person_organization	10	2
rel_person_paper	9	2
rel_person_topic	21	2
topics	15	4

Table 5. R2RML Document Summary

Triples Map	Predicate Object Maps	Parent Triples Map
Person, Person_research_interests, Person_has_affiliation	1	
	1	Concept
	1	Organization
Concept	2	Concept
PostalAddress	3	
Organization	1	
Document, Document_Creator, Document_subject	3	
	1	Person
	1	Concept

Table 6. Count of Subjects per Triples Map Group

Triples Map Group	Subject Nodes
Concept	15
Organization	9
PostalAddress	2
Person, Person_research_interests, Person_has_affiliation	8
Person, Person_has_affiliation	1
Person	1
Document, Document_Creator, Document_subject	2
Document, Document_Creator	1
Document	1

본 데모에서 표 5의 R2RML 문서로부터 RDF 데이터셋을 생성하기 위해 연구 [17]의 구현물인 [18]의 R2RML 프로세서를 사용하였다. 이 프로세서는 154개의 트리플로 구성된 RDF 문서를 생성하였다. 이 문서에 출현하는 주어 IRI는 총 40개였다. 즉, 이 문서는 154개의 트리플을 40개의 주어를 기준으로 분류할 수 있다. 표 6은 RDF 주어 노드들을 기준으로 한 40개의 트리플 그룹에 대한 생성 출처 triples map이 무엇인지를 보여준다. 표 6의 첫 번째 열에서 복수개의 triples map 이름이 포함된 행이 존재하므로 이것은 복수개의 triples map에 의해 생성된 트리플들이 공통된 주어를 설명하게 된 케이스가 발생했음을 보여준다.

Table 7. Generated Shape List

Shape
PersonShape, Person_has_affiliationShape, Person_research_interestsShape, PersonPerson_has_affiliationShape, PersonPerson_research_interestsShape, Person_has_affiliationPerson_research_interestsShape, PersonPerson_has_affiliationPerson_research_interestsShape
DocumentShape, Document_CreatorShape, Document_subjectShape, DocumentDocument_CreatorShape, DocumentDocument_subjectShape, Document_CreatorDocument_subjectShape, DocumentDocument_CreatorDocument_subjectShape
OrganizationShape
PostalAddressShape
ConceptShape

표 5의 R2RML 문서를 입력으로 하여 본 연구의 시스템을 실행한 결과 17개의 셰이프와 1개의 레이블을 갖는 노드 제약조건으로 구성된 ShEx 문서가 생성되었다. 표 7은 생성된 셰이프 목록이다. Type 1 Shape의 셰이프명은 생성 출처 triples map의 이름 뒤에 'Shape'을 덧붙인 형식이며 Type 2 Shape의 셰이프명은 생성 출처 triples map들의 모든 이름을 나열한 후 'Shape'을 덧붙인 형식이다. 각각의 triples map으로부터 1개씩의 Type 1 셰이프와 모든 경우를 대비한 Type 2 셰이프가 올바르게 생성되었음을 알 수 있다.

```
map:Person
  rr:logicalTable [ rr:sqlQuery """
    SELECT PerID AS ID, Email AS foaf_mbox, FirstName AS
foaf_name, LastName AS foaf_name2, Address FROM
persons
  """ ];
  rr:subjectMap [
    rr:template 'http://data.example.com/person/{"ID"}';
    rr:class dc:Person;
  ];
  rr:predicateObjectMap [
    rr:predicate foaf:name;
    rr:objectMap [
      rr:template '{"foaf_name"} {"foaf_name2"}';
      rr:termType rr:Literal;
    ]
  ].
map:Person_has_affiliation
  rr:logicalTable [ rr:sqlQuery """
   SELECT persons.PerID AS ID_foaf_Person,
rel_person_organization.OrganizationID AS conf_has_affiliation
    FROM persons, rel_person_organization, organizations
    WHERE persons.PerID = rel_person_organization.PersonID
    AND rel_person_organization.OrganizationID = organizations.OrgID
  """ ];
  rr:subjectMap [
    rr:template
                  'http://data.example.com/person/{"ID_foaf_Person"}';
    rr:class foaf:Person;
  ];
  rr:predicateObjectMap [
    rr:predicate conf:has_affiliation;
    rr:objectMap [
      rr:parentTriplesMap map:Organization;
      rr:joinCondition [
        rr:child 'conf_has_affiliation';
        rr:parent 'OrgID';
      ]
    ]
  ].
map:Document
  rr:logicalTable [ rr:sqlQuery """
    SELECT PaperID AS ID, Title AS dc_title,
           Year AS dc_date,
           (SELECT Name || ', ' || location
            from conferences
            where conferences.ConfID=papers.conference)
                          AS conf_conference
    FROM papers
    WHERE papers.Year > 2002
  """ ];
  rr:subjectMap [
    rr:template 'http://data.example.com/document/{"ID"}';
    rr:class foaf:Document;
  ];
  rr:predicateObjectMap [
    rr:predicate conf:conference;
    rr:objectMap [ rr:column 'conf_conference' ; ]
  ];
  rr:predicateObjectMap [
    rr:predicate dc:date;
    rr:objectMap [ rr:column 'dc_date' ; ]
  ];
  rr:predicateObjectMap [
    rr:predicate dc:title;
    rr:objectMap [ rr:column 'dc_title' ;]
  ].
map:Document_Creator
  rr:logicalTable [ rr:sqlQuery """
    SELECT papers.PaperID AS ID_foaf_Document,
           rel_person_paper.PersonID AS dc_creator
    FROM papers, rel_person_paper, persons
    WHERE papers.PaperID = rel_person_paper.PaperID
    AND rel_person_paper.PersonID = persons.PerID
    AND papers.Year > 2002
  """ ];
  rr:subjectMap [
    rr:template
    http://data.example.com/document/{"ID_foaf_Document"}';
    rr:class foaf:Document;
  ];
  rr:predicateObjectMap [
    rr:predicate dc:creator;
    rr:objectMap [
      rr:parentTriplesMap map:Person;
      rr:joinCondition [
        rr:child 'dc_creator';
        rr:parent 'PerID';
      ]
    ]
  ].
```

Fig. 12. Some Triples Maps in Table 4

그림 13은 표 5의 R2RML 문서로부터 생성된 ShEx 문서의 일부다. 그림 13에 포함된 셰이프와 노드 제약조건으로 본 연구에서 고려한 ShEx 구문들이 올바르게 생성되었음을 설명할 수 있다. 그림 12는 그림 13의 제약조건들을 생성케 한 triples map들만 포함하고 있다.

```
foo:PersonShape_Obj1 LITERAL /^(.*) (.*)$/

foo:PersonShape IRI
 /^http:₩/₩/data₩.example₩.com₩/person₩/(.+)$/ {
  a [ dc:Person ] ;
  foaf:name @foo:PersonShape_Obj1
}

foo:Person_has_affiliationShape IRI
 /^http:₩/₩/data₩.example₩.com₩/person₩/(.+)$/ {
  a [ foaf:Person ] ;
  conf:has_affiliation @foo:OrganizationShape *
}

foo:PersonPerson_has_affiliationShape IRI
 /^http:₩/₩/data₩.example₩.com₩/person₩/(.+)$/ {
  foaf:name @foo:PersonShape_Obj1 ;
  conf:has_affiliation @foo:OrganizationShape * ;
  a [ dc:Person foaf:Person ] {2}
}

foo:DocumentShape IRI
 /^http:₩/₩/data₩.example₩.com₩/document₩/(.+)$/ {
  a [ foaf:Document ] ;
  dc:date xsd:integer ? ;
```

```
  dc:title xsd:string ? ;
  conf:conference xsd:integer ?
}

foo:Document_CreatorShape IRI
 /^http:₩/₩/data₩.example₩.com₩/document₩/(.+)$/ {
  a [ foaf:Document ] ;
  (
     dc:creator @foo:PersonShape *
    |
     dc:creator @foo:PersonPerson_has_affiliationShape *
    |
     dc:creator @foo:PersonPerson_research_interestsShape *
    |
     dc:creator
@foo:PersonPerson_has_affiliationPerson_research_interestsShape *
  )
}
```

Fig. 13. The Part of Generated ShEx Schema

그림 13의 첫 라인은 레이블을 갖는 노드 제약조건의 예이다. 이 노드 제약조건은 triples map Person의 유일한 object map으로부터 생성되었다. 이 object map에 정의된 template 속성으로 인해 NodeConstraint 객체에 xsFacet 속성이 존재하게 되어 레이블을 갖는 노드 제약조건이 생성 되었다. 출력된 xsFacet은 정규 표현식이며 template에서 컬럼값으로 치환되는 부분인 중괄호부를 “(.*)”으로 치환한 결과이다. 이 노드 제약조건은 Person 셰이프의 술어가 foaf:name인 트리플 제약조건의 valueExpr 자리에서 참조되고 있다. 이 트리플 제약조건은 Type 2 트리플 제약조건 중 노드 제약조건을 참조하는 형식의 출력 예이다.

그림 12에서 Person과 Person_has_affiliation triples map은 subject map의 template의 값이 컬럼값 치환부를 제외하고 동일함을 알 수 있다. 이로 인해 이 두 triples map은 Type 2 Shape을 생성하게 되는 것이다. 그 결과로 이 두 triples map에 의해 생성된 그림 13에서의 첫 세 개 셰이프의 중괄호 이전 파트가 동일하게 된 것이다. Type 2 셰이프의 한 사례인 PersonPerson_has_affiliationShape의 중괄호 파트에 나열된 트리플 제약조건들은 자신을 생성케 한 Type 1 셰이프인 Person과 Person_has_affiliation 셰이프 내의 트리플 제약조건들을 모두 성공적으로 포함시켰음을 보인다. 이 셰이프의 세 번째 트리플 제약조건은 Type 1 셰이프들에 속해 있던 Type 1 트리플 제약조건들이 Type 2 셰이프 내에서 하나의 Type 1 트리플 제약조건으로 수렴한 사례이다. 이 셰이프의 두 번째 트리플 제약조건은 Type 3 트리플 제약조건의 사례이다.

그림 13에서 DocumentShape 셰이프의 2~4번째 트리플 제약조건은 Type 2 트리플 제약조건 중 노드 제약조건을 트리플 제약조건의 valueExpr 자리에 직접 출력하는 사례이다. 이처럼 본 연구에서는 xsFacet이 없는 단항 노드 제약조건은 트리플 제약조건 내부에서 직접 출력하도록 하였다.

그림 13에서 Document_CreatorShape 셰이프는 트리플 표현식 multiElementOneOf를 포함하는 사례이다. 이 트리플 표현식은 그림 12의 triples map Document_Creator의 유일한 referencing object map에 parent triples map으로서 지정된 triples map이 Person인 것에 기인한다. 즉, triples map Person이 Type 2 Shape의 생성을 유발하기 때문이다. 따라서 이 셰이프는 잠재적으로 가능한 dc:creator의 모든 목적어 구조를 설명할 수 있도록 triples map Person으로부터 생성된 Type 1 셰이프인 Person 뿐만 아니라 Person 셰이프를 기반으로 생성된 Type 2 셰이프들에 대한 참조로 구성된 트리플 표현식을 준비해 놓은 것이다.

그림 14는 ShEx2 온라인 검증기[19] 상에서 본 데모를 위해 생성한 RDF 데이터셋을 본 연구의 시스템이 출력한 ShEx 문서를 통해 구조 검증을 수행하는 스냅샷이다. 그림 14의 좌상단 박스에는 ShEx 문서 전체가 로딩되어 있는 모습이며 우상단 박스에는 RDF 문서 전체가 로딩된 상태이다. 그림 14의 하단부는 Query Map Editor 편집모습으로서 이 편집기에서 검증 대상 노드와 제약조건을 짝지어줄 수 있다. 그 과정 중 팝업 메뉴가 표시되어 있으며 18개의 제약조건이 나열되어 있다. 로딩한 ShEx 문서에 문법적 오류가 있다면 이 팝업 메뉴는 디스플레이되지 않기 때문에 본 연구의 시스템이 생성한 ShEx 문서에 문법 오류가 존재하지 않음을 알 수 있다.

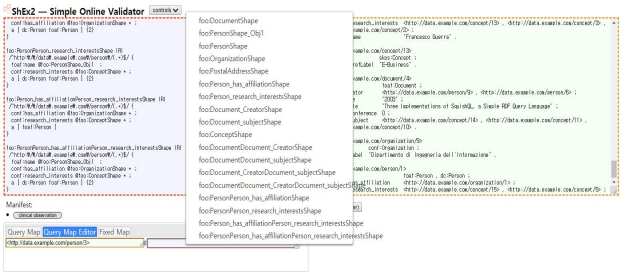

Fig. 14. Validation on the ShEx2 Validator

```
<http://data.example.com/person/3>
  a                    foaf:Person , dc:Person ;
  conf:has_affiliation <http://data.example.com/organization/1> ;
  foaf:name            "Jim Blythe" .
```

Fig. 15. Target Node for Validation

Fig. 16. Validation Results

그림 15는 검증에 선택된 타겟 노드이며 triples map Person과 Person_has_affiliation로부터 생성된 유일한 트리플 그룹이다. 이 노드의 구조는 PersonPerson_has_affiliation 셰이프로만 설명 가능해야 한다. 이 주장을 검증기에서 확인한 결과가 그림 16이다. 첫 두 행에는 ✗가 표시된 반면 마지막 행에는 ✓가 표시되었다. 이 결과는 타겟 노드가 첫 두 행의 셰이프 제약조건을 각각 만족시키지 못했으나 마지막 행의 셰이프 제약조건은 만족시켰음을 의미한다. 그림 16에서 1행은 PersonShape, 2행은 Person_has_affiliation, 3행은 PersonPerson_has_affiliation 셰이프가 검증 셰이프이다.

V. Conclusion and Future Work

본 논문을 통해 RDB로부터 R2RML 매핑에 의해 생성한 RDF 그래프에 대한 ShEx 스키마를 자동 생성하는 방법을 상세히 소개하였다. RDF 그래프에 대한 스키마 정보를 생성하는 기존 연구들은 그 목적에 따라 크게 두 부류로 요약할 수 있다. 첫 번째 부류는 최근 연구를 기준으로 연구 [9]와 같이 데이터셋을 공개하기 이전 단계에서 요구되는 구조 검증 수단으로서의 스키마를 생성하는 연구들이다. 두 번째 부류는 최근 연구를 기준으로 연구 [8]과 같이 공개된 데이터셋에 대한 구조를 사용자들에게 직관적으로 이해시킬 수단으로서의 스키마를 생성하는 연구들이다. 즉, RDF 모델에 대해서는 스키마의 두 가지 역할이 통합되지 못하고 나뉘어져 진행되어 왔다. W3C는 이러한 문제를 극복하기 위한 수단으로서 RDF에 대한 구조 검증과 설명이 동시에 가능한 ShEx 언어를 제안하였다. 본 연구는 이러한 특성을 갖는 언어로 스키마를 표현하기 때문에 본 연구를 통해 자동 생성된 스키마는 RDF 데이터셋에 대한 두 가지 목적의 스키마 역할을 모두 충실히 수행할 수 있다. 한편, R2RML 매핑은 RDB 데이터로부터 RDF 데이터셋을 획득케 하는 표준방법일 뿐 아니라 활용빈도가 높은 기술이다. 일반적으로 자동 생성되는 구조적인 RDF 데이터셋은 다양한 방식으로 수집된 데이터를 우선 RDB에 저장한다. 이 과정에서 데이터 정제 효과가 발생하기 때문이다. 이 후 RDB 데이터는 R2RML 매핑에 의해 RDF로 변환하여 공개된다. 본 연구를 통해 제안한 기술은 이러한 방식으로 생성되는 RDF 데이터셋에 대한 스키마를 자동 생성해주기 때문에 파급 효과가 클 것으로 기대된다. 마지막으로 본 연구를 기반으로 웹 브라우저 상에서 RDF 데이터셋의 탐색과 SPARQL 질의문 작성을 직관적으로 수행할 수 있도록 하는 도구를 개발할 예정이다.

REFERENCES

[1] Linking Open Data W3C SWEO Community Project, *https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData*

[2] RDF 1.1 Concepts and Abstract Syntax, *http://www.w3.org/TR/rdf11-concepts/*

[3] SPARQL 1.1 Update, *https://www.w3.org/TR/sparql11-update/*

[4] A Direct Mapping of Relational Data to RDF, *http://www.w3c.org/TR/rdb-direct-maipping/*

[5] R2RML: RDB to RDF Mapping Language, http://www.w3.org/TR/r2rml/

[6] Generating RDF from Tabular Data on the Web, *http://www.w3c.org/TR/csv2rdf/*

[7] D. Tomaszuk, "RDF Validation: A Brief Survey," Proceedings of the 13st International Conference on Beyond Databases, Architectures and Structures, pp. 344-355, 2017.

[8] S. Han, L. Zou, J. X. Yu, and D. Zhao, "Keyword Search on RDF Graph - A Query Graph Assembly Approach," Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 227-236, 2017.

[9] D. Arndt, B. D. Meester, A. Dimou, R. Verborgh, and E. Manneus, "Using Rule-Based Reasoning for RDF Validation," Proceedings of International Joint Conference on Rules and Reasoning, pp. 22-36, 2017.

[10] SHACL, *http://www.w3c.org/TR/shacl/*

[11] ShEx, *http://shex.io/shex-semantics/*

[12] RDF 1.1 Turtle, *http://www.w3c.org/TR/turtle/*

[13] XML Schema Part 2: Datatypes Second Edition, *http://www.w3c.org/TR/xmlschema-2/*

[14] XPath 3.1 Regular expression, *http://www.w3c.org/TR/xpath-functions-31/#regex-syntax*

[15] Natural Mapping of SQL Values, *https://www.w3.org/TR/2012/REC-r2rml-20120927/#natural-mapping*

[16] MariaDB, *http://mariadb.com*

[17] N. Konstantinou and D.E. Spanos, "Creating Linked Data from Relational Databases," Springer, pp. 73-102, Sept. 2015.

[18] R2RML Parser, https://github.com/nkons/r2rml-parser/

[19] ShEx2 - Simple Online Validator, https://rawgit.com/shexSpec/shex.js/master/doc/shex-simple.html

Authors

Ji-Woong Choi received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Soongsil University, Korea, in 2001, 2003 and 2011, respectively Dr. Choi joined the faculty of the School of Computer Science and Engineering at Soongsil University, Seoul, Korea, in 2013. He is currently an Assistant Professor in the School of Computer Science and Engineering, Soongsil University. He is interested in information system.