정보자원의 개방과 연계를 위한 SRU, SRU Record Update 프로토콜 연구

A Study on SRU & SRU Record Update Protocol for Openness and Linkage of Resources

이 지 원(Ji-Won Lee)*

〈 목 차 〉

Ⅰ. 서 론
Ⅱ. SRU 프로토콜 기능과 특징
1. SRU 프로토콜 개요
2. SRU 프로토콜 기능과 데이터 요소
3. CQL의 특징과 표현
4. 검색 프로토콜 비교
Ⅲ. SRU Record Update 프로토콜 기능과 특징
1. SRU Record Update 개요
2. SRU Record Update 기능과 데이터 요소
3. 갱신 작업 및 데이터 요소
4. 갱신 프로토콜 비교
Ⅳ. 프로토콜 구현 사례
1. The European Library
2. OCLC WorldCat
3. 한국교육학술정보원 종합목록
Ⅴ. 결론 및 제언

초 록

다양한 가상공간에 산재하는 많은 분산자원들을 보다 효과적으로 활용하기 위하여 여러 프로토콜들이 개발되어 왔다. 이 연구는 분산 정보자원 검색과 갱신을 위한 SRU, SRU Record Update 프로토콜의 개발 배경, 기능, 구성 요소 등을 살펴봄으로 새로운 표준 프로토콜에 대한 이해를 넓히기 위함이다. 또한 다른 프로토콜과의 비교 및 실제 구현 사례 등을 통하여 자관의 정보자원을 외부에 효과적으로 제공하고, 외부 정보자원을 적절히 활용하려는 도서관 및 정보서비스 기관들에게 상호운용성 보장을 위한 실제적인 방안을 제시하기 위함이다.

키워드: SRU 프로토콜, SRU Record Update 프로토콜, 웹서비스 프로토콜, 상호운용성

ABSTRACT

Several protocols have been developed to efficiently utilize a great number of distributed resources. This paper investigated the background, operations and elements of SRU and SRU Record Update protocol, compared them with other protocols, and reviewed their implementation cases. The purpose of this pager is to broaden the understanding of the two new standards and to provide a practical guide to ensure their interoperability for libraries and information service centers which want to expose their own contents and to access to external resources.

Keywords: SRU Protocol, Search/Retrieval via URL, SRU Record Update Protocol, Web Service Protocol, Interoperability

* 한국교육학술정보원 선임연구원(jwlee@keris.or.kr)
• 접수일: 2009년 8월 23일 • 최초심사일: 2009년 8월 25일 • 최종심사일: 2009년 9월 21일

Ⅰ. 서 론

도서관 및 정보서비스 기관 등에서 자관이 소유하고 있는 물리적인 형태의 자원뿐만 아니라 다양한 외부 자원을 활용하는 것에 관심과 노력을 기울인다는 것은 이제 모두가 인지하는 사실이다. 매일 새로워지는 정보통신기술은 이러한 활용에 가장 큰 영향을 미쳤으며, 다양한 가상공간에 산재하는 많은 분산자원들을 보다 효과적으로 활용하고 상호운용성을 보장하기 위하여 여러 표준 프로토콜들이 개발되어 왔다.

Z39.50 프로토콜은 미국 국가표준이자 국제표준(ISO 23950)으로 이기종간의 정보 검색을 위해 표준화된 통신규칙을 규정한 것이다. 특히 도서관 분야에서 표준화된 방식으로 원격지의 자원 검색을 하는데 널리 사용되어 왔다. Z39.50 프로토콜은 1988년 제1판이 발표된 이래 30년 이상 사용된 안정적인 프로토콜이지만, 웹 환경이 보편화 되면서 그 구현에 있어 최신성과 편리성이 뒤떨어진다는 인식이 확산되어 왔다. 이러한 배경하에 2000년부터 ZING(Z39.50 International: Next Generation)이라는 차세대 Z39.50 프로토콜을 위한 5가지의 연구 및 프로젝트가 수행되었다. 2006년 ZING의 공식 홈페이지가 없어지면서 ZING이라는 명칭은 더 이상 사용되지는 않지만, ZING의 핵심 연구 주제였던 SRU(Search/Retrieve via URL Service) 프로토콜 홈페이지[1]가 ZING의 다른 영역인 CQL과 ZeeRex의 내용을 포함하며 그 역할을 지속하고 있다.

SRU 프로토콜이 원격지의 정보자원 검색을 위한 것이라면, SRU Record Update 프로토콜은 원격지의 정보자원 갱신을 위한 것이다. 종합목록과 같이 정보자원이 하나 이상의 기관에서 작성된 데이터의 집합이라면 이 자원의 최신성, 정확성을 보장하기 위해서는 여러 기관의 데이터 변경사항이 신속하고 정확하게 반영되어야 한다. SRU Record Update 프로토콜은 이를 위하여 개발된 갱신 프로토콜이다. 명칭에서도 알 수 있듯이 SRU 프로토콜과 그 기반을 같이 하고 있다.

미국 국회도서관(이하 LC)에서 두 프로토콜을 유지, 관리하고 있으며, SRU 편집위원회 중심으로 지속적인 개정 작업과 활용 확대를 위한 여러 가지 활동들을 하고 있다. SRU, SRU Record Update 프로토콜은 Z39.50 프로토콜의 기본 개념은 유지하면서 기술적인 측면에서는 웹을 기반으로 하여 구현의 장벽을 낮추었다는데 가장 큰 의의가 있다. 즉 표준만이 가지는 상호운용성을 보장하는 동시에 XML, HTTP 등의 웹 기반 기술을 사용함으로써 구현의 편리성 및 확장성을 가질 수 있게 된 것이다.

이 연구는 분산 정보자원 검색과 갱신을 위한 새로운 프로토콜인 SRU, SRU Record Update의 개발 배경, 기능, 구성 요소 등을 살펴봄으로 표준 프로토콜에 대한 이해를 넓히기 위함이다. 또한 다른 프로토콜과의 비교 및 실제 구현 사례 등을 통하여 자관의 정보자원을 외부에 효과적으로 제공하고, 외부 정보자원을 적절히 활용하려는 도서관 및 정보서비스 기관들에게 상호운용성 보장을

1) SRU 홈페이지, 〈http://www.loc.gov/standards/sru/〉 [cited 2009. 6. 22].

위한 실제적인 방안을 제시하기 위함이다.

Ⅱ. SRU 프로토콜 기능과 특징

1. SRU 프로토콜 개요

SRU 프로토콜은 정보검색의 표준을 정의한 REST(REpresentational State Transfer) 방식의 웹 서비스 프로토콜이다. 2002년 11월 제1판, 2004년 2월 제1.1판이 발표되었고, 현재 2007년 7월에 발표된 제1.2판이 최신판이다. 차세대 Z39.50 프로토콜을 개발하기 위하여 ZING이라는 명칭으로 연구가 시작되었을 당시는 SOAP(Simple Object Access Protocol) 방식의 웹 서비스 프로토콜[2]인 SRW와 같이 다루어 졌으나, 현재는 SRU가 대표 명칭으로 사용되고 SRW는 SRU의 하나의 변형인 "SRU via HTTP SOAP"이라는 명칭으로 다루어지고 있다.[3]

LC에서 운영하고 있는 SRU 공식 홈페이지에서는 SRU 프로토콜 명세서, 프로토콜과 관련된 구성요소들(Context Sets, Schemas, Profiles 등), 개발 도구 및 사례, SRU 관련 서지사항 및 각종 소식 등을 제공하고 있다. SRU 편집위원회가 프로토콜의 개정에 주된 책임을 맡고 있으며, SRU 구현에 관심 있는 누구나 참여할 수 있는 listserv를 운영하고 있다.

LC, OCLC, BL, 유럽연합 도서관, 네덜란드 국립도서관(National Library of the Netherlands) 등의 기관에서 SRU 프로토콜을 구현하였고, OCLC, Indexdata[4] 등에서는 SRU 프로토콜 관련 소프트웨어와 소스를 다운로드받을 수 있게 공개하였다.

2. SRU 프로토콜 기능과 데이터 요소

SRU는 SerachRetrieve, Scan, Explain 세 가지의 기능을 제공하며, 각 기능에서 사용하는 요청과 응답에 대한 파라미터들이 정의되어 있다.[5]

---

2) 클라이언트가 서버에 요청시 그리고 서버에서 클라이언트로 결과를 보낼 때 모두 SOAP 형식으로 처리한다. SOAP은 XML과 HTTP 등을 기반으로 하여 다른 컴퓨터에 있는 데이터나 서비스를 호출하기 위한 통신 규약이다.

3) Ray Denenberg, "Search Web Services - The OASIS SWS Technical Committee Work," *D-Lib Magazine*, Vol.15, No.1/2(Jan/Feb 2009), 〈http://www.dlib.org/dlib/january09/denenberg/01denenberg.html〉 [cited 2009. 7. 2].

4) 도서관 등을 위한 정보 검색 소프트웨어 개발 업체로 덴마크 코펜하겐에 본사가 있으며, Z2950 프로토콜 개발을 위한 YAZ toolkit과 Z39.50과 SRU 프로토콜의 지원이 모두 가능한 YAZproxy 등을 개발하였다.

5) *SRU Version 1.2 Specifications*, 2007. 〈http://www.loc.gov/standards/sru/specs/〉 [cited 2009. 6. 22].

가. SearchRetrieve 기능

SRU의 핵심 기능이면서 필수 지원 기능으로, 제1판부터 포함되었다. 원하는 레코드 검색을 위하여 클라이언트는 원격지 서버로 질의문을 포함하여 관련 파라미터 값들을 전달하고, 서버는 이에 적합한 레코드를 검색하여 클라이언트로 보낸다.

〈표 1〉 searchRetrieve 요청 파라미터

명 칭	유 형	필수여부
operation	*xsd:string*	필수
version	*xsd:string*	필수
query	*xsd:string*	필수
startRecord	*xsd:integer*	선택
maximumRecords	*xsd:integer*	선택
recordPacking	*xsd:string*	선택
recordSchema	*xsd:string*	선택
recordXPath	*xsd:string*	선택
resultSetTTL	*xsd:integer*	선택
sortKeys	*xsd:string*	선택
stylesheet	*xsd:anyURI*	선택
extraRequestData	*xmlFragment*	선택

① operation : 'searchRetrieve'라는 문자열이 파라미터 값이다.

② version : 요청에서 현재 적용하고 있는 판 정보이면서 동시에 응답에서 지원되기를 바라는 판 정보를 나타낸다.

③ query : 검색을 원하는 질의문을 CQL로 표현한 것이다.

④ startRecord : 검색된 레코드 집합 중에서 첫 번째 레코드의 위치를 나타낸다. 0보다 커야 하며, 생략된 경우 초기 설정값은 1이다.

⑤ maxiumRecords : 서버로부터 응답을 원하는 레코드의 수이다. 0보다 커야 하며, 생략된 경우 초기 설정값은 서버에 의해 결정된다.

⑥ recordPacking : 서버로부터 응답되어지는 레코드의 형태를 나타내며 'xml'과 'string'이 선택가능한 값이다. 초기 설정값은 'xml'이다.

⑦ recordSchema : 서버로부터 응답되어지는 레코드에 적용되는 스키마를 지정한다. URI 식별자 형태이거나 서버가 지정한 접두어 형태이다. 생략된 경우 초기 설정값은 서버에 의해 결정된다.

⑧ recordXPath : 레코드 스키마 중에서 요청하는 특정 부분을 지정한다. 1.1판에서만 사용한다.

⑨ resultSetTTL : 서버에서 응답한 결과집합(result set)이 유지되기 원하는 시간을 지정하는

것으로, 초 단위로 표현한다. 생략된 경우 초기 설정값은 서버에 의해 결정된다.

⑩ sortKeys : 서버에서 응답되어지는 레코드에 적용되기 원하는 정렬 기준을 지정한다. 1.1판에서만 사용한다.

⑪ styleSheet : 서버에서 응답되어지는 레코드에 포함될 스타일시트의 URL을 지정한다.

⑫ extraRequestData : 기타 추가적인 정보의 요청 내용을 지정한다.

다음은 SRU searchRetrieve 요청의 예이다.

http://z3950.loc.gov:7090/voyager?version=1.1&operation=searchRetrieve&query=dinosaur&maximumRecords=1&recordSchema=dc

〈표 2〉 searchRetrieve 응답 파라미터

명 칭	유 형	필수여부
version	*xsd:string*	필수
numberOfRecords	*xsd:integer*	필수
resultSetId	*xsd:string*	선택
resultSetIdleTime	*xsd:integer*	선택
records	*sequence of records*	선택
nextRecordPosition	*xsd:integer*	선택
diagnostics	*sequence of diagnostics*	선택
extraResponseData	*xmlFragment*	선택
echoedSearchRetrieveRequest	*echoedSearchRetrieveRequest*	선택

① version : 현재 적용하고 있는 판 정보로서, 클라이언트에서 요청한 판보다 같거나 낮아야 한다.

② numberOfRecords : 결과집합에 포함된 레코드의 수이다.

③ resultSetId : 클라이언트가 요청한 질의가 수행되어 검색된 결과집합의 고유한 식별자이다.

④ resultSetIdleTime : 결과집합이 유지되는 시간으로, 초 단위로 표현한다.

⑤ records : 클라이언트가 요청한 질의에 해당되는 일련의 레코드들이다. 집합이다. 하위 요소로 〈recordSchema〉, 〈recordPacking〉, 〈recordData〉, 〈recordIdentifier〉, 〈recordPosition〉, 〈extraRecordData〉가 있다.

⑥ nextRecordPosition : 마지막 레코드 다음 레코드의 결과집합내의 위치를 나타낸다. 남아있는 레코드가 없으면, 이 요소는 생략된다.

⑦ diagnostics : 클라이언트의 요청을 처리하는 과정에서 생긴 오류에 대한 결과를 보여준다. 하위 요소로 〈uri〉, 〈details〉, 〈message〉가 있는데, 〈uri〉는 진단 결과 기록에 대한 고유의 식별

자를 나타내기 위한 요소로서, 〈uri〉 요소값이 "info:srw/diagnostic/1/"로 시작하면 표준 진단 목록이 적용된 것이다.

⑧ extraResponseData : 서버에서 제공하는 기타 추가적인 정보를 포함한다.

⑨ echoedSearchRetrieveRequest : 클라이언트에서 받은 요청 내용을 XML 형태로 다시 작성하여 포함한다.

나. Scan 기능

Scan 기능은 SearchRetrieve 기능과 같이 검색을 원하는 레코드 전체를 요청하는 것이 아니라, 원하는 색인에 해당하는 색인어의 정렬 리스트를 요청 및 응답하기 위한 기능이다. 도서의 권말색인과 같은 개념으로, 예를 들어 원하는 서명에 대한 Scan 요청을 하면, 서버는 관련된 서명 리스트들만 보내는 것이다. 브라우징을 원하는 색인과 색인어를 CQL로 표현한 scanClause 등의 파라미터를 포함하여 요청하며, 클라이언트에서 요청한 색인어에 해당하는 용어 리스트를 포함하는 terms 등의 파라미터 사용하여 응답한다.

다. Explain 기능

클라이언트가 SRU 서버가 제공하는 이용가능한 정보를 확인할 수 있는 기능이다.[6] 이 정보를 확인함으로 클라이언트는 적절한 설정 및 인터페이스 구성을 할 수 있게 된다. 응답의 핵심 요소인 record는 ZeeRex 형식을 적용한다.

ZeeRex는 Z39.50 또는 SRU 서버에 관한 정보를 기술하는데 사용되는 XML 스키마를 정의한 것이다. SRU 서버를 위한 ZeeRex 스키마 상위 요소는 서버 연결 정보를 나타내는 〈serverInfo〉, 데이터베이스에 관한 정보를 나타내는 〈databaseInfo〉, ZeeRex 레코드 자체에 관련된 정보를 나타내는 〈metaInfo〉, 데이터베이스 탐색을 위해 사용 가능한 색인에 관련된 정보를 나타내는 〈indexInfo〉, 스키마에 관련된 정보를 나타내는 〈schemaInfo〉, 서버의 설정과 관련된 정보를 나타내는 〈configInfo〉가 있다.[7]

3. CQL의 특징과 표현

CQL(Contextual Query Language)[8]은 정보검색시스템에서 질의를 표현하기 위해서 사용하는 언어로서 2007년 7월 발표된 제1.2판이 최신판이다. SRU 프로토콜에서 서버에 요청하는 질의

6) Explain 요청을 이용하지 않고, 어떤 파라미터도 포함하지 않은 SRU 기본 URL 요청만으로도 서버 설명 정보를 제공받을 수 있다.

7) *An Overview of ZeeRex*, 2004, 〈http://explain.z3950.org/overview/index.html〉 [cited 2009. 6. 22].

8) 제1.1판에서는 Common Query Language이 공식 명칭이었으나, 제1.2판으로 개정되면서 명칭도 변경되었다.

구성에 사용되며, SRU 프로토콜과 함께 유지, 관리되고 있다.[9]

질의 언어는 크게 두 가지 종류로 나눌 수 있는데, 하나는 강력하고 표현이 풍부하지만 비전문가는 쉽게 읽고 쓸 수 없는 언어로서, SQL, PQF, XQuery 등이 해당된다. 다른 하나는 복잡한 개념을 표현하기에는 충분하지 않으나 단순하고 직관적인 언어로서 CCL 또는 구글과 같은 검색엔진에서 사용하는 질의 언어이다. CQL은 두 가지 유형의 질의 언어의 장점을 모두 수용하여, 단순하면서도 직관적인 동시에 표현의 풍부함도 유지하고자 개발되었다.

질의 구성에 있어 다양한 분야에서 자체적으로 사용하는 의미들을 수용하고 상호운영성을 보장하기 위하여 CQL은 Context Sets을 이용한다. Context Set은 XML의 이름공간(Namespace)과 유사한 개념으로서, 질의문에서 사용할 수 있는 요소들을 자체적으로 정의한 것이다. 색인명, 관계표현(Relation), 관계표현 수정기호(Relation Modifier), 불리언 수정기호(Boolean Modifier)들이 정의될 수 있는데, 반드시 네 가지 요소들이 모두 포함되어야 하는 것은 아니며, 대부분의 Context Set은 색인명만이 정의되어 있다. Context Set은 URI 형식의 고유한 식별기호(Identifier)를 가지고 있으며, 또한 이에 해당하는 접두어(prefix)도 가지게 된다. CQL문에서는 보통 prefix.value 형식으로 표현하며 접두어가 생략된 경우는 서버에서 지정한 Context Set을 사용하는 것이다. SRU 홈페이지는 현재 CQL Context set, Dublin Core Context Set, MARC Context Set 등 18개의 Context Set이 등록되어 있다. CQL Context Set은 접두어로 cql을 사용하며, 여러 도메인이나 프로토콜에서 일반적으로 적용할 수 있는 색인명, 관계표현, 관계표현 수정기호, 불리언 수정기호를 정의한 것이다.

CQL 질의문은 관계표현 기호, 불리언 연산기호 등을 이용하여 다양한 표현이 가능하다. 다음은 간단한 CQL 질의문의 예이다.

```
dc.title any fish
dc.title any fish or dc.creator any sanderson
dc.title any fish sortBy dc.date/sort.ascending
> dc = "info:srw/context-sets/1/dc-v1.1" dc.title any fish
```

마지막 예에서는 사용하는 접두어(dc)에 해당하는 Context Set 고유 식별기호를 포함한 경우이다.

9) *CQL: Contextual Query Language(SRU Version 1.2 Specifications)*, 2007. <http://www.loc.gov:8081/standards/sru/specs/cql.html> [cited 2009. 6. 22].

4. 검색 프로토콜 비교

도서관 및 정보서비스기관에서의 분산자원 검색은 Z39.50 프로토콜의 출현으로 본격화되었으며, Z39.50 프로토콜의 기반의 검색은 지난 30여 년간 다양한 기관, 다양한 목적으로 폭넓게 사용되었다.

그러나 웹 기반의 네트워크 환경이 일반화되면서 Z39.50 프로토콜을 적용하지 않고, 자체 메타검색 방식을 개발하여 분산자원 검색에 이용하는 것이 일반적인 방식이 되었다. 이는 HTTP를 기반으로 하는 웹 환경에서는 자체 메타검색 방식이 Z39.50 프로토콜에 비해 구현이 용이하기 때문이다. 또한 Z39.50 프로토콜은 검색 대상이 Z39.50 서버가 구축된 시스템만으로 한정되지만, 자체 메타검색 방식은 웹 페이지에 검색 인터페이스를 가지는 시스템을 검색 대상으로 할 수 있기 때문에 그 범위가 더 광범위하다는 장점을 가진다.

하지만 자체 메타검색 방식은 표준 검색 프로토콜을 적용하지 않았기 때문에, 검색 대상에 대한 정보(검색 항목, 연산자, 디스플레이 항목 등)를 해당 검색 인터페이스의 개별적인 분석을 통해서 알아내야 하고 또한 이러한 정보가 변경되는 경우에도 변경된 내용을 쉽게 알 수가 없다. 따라서 개발 수준에 따라 검색 효율성에 대한 신뢰도의 편차가 많이 존재하며, 검색 대상 정보에 대한 불확실성으로 인해 안정적인 메타검색 서비스를 제공하는데 한계가 있었다.[10] 초기 메타검색이 이러한 한계가 많았다면 최근에는 포털 사이트, 인터넷 서점, 정보서비스 기관들이 외부에서 자신들의 정보자원 검색을 용이하게 할 수 있도록 Open API를 공개함으로써 비교적 안정적인 검색을 가능케 하고 있다. 하지만 이러한 웹 서비스 Open API도 대부분 각 기관에서 자체적으로 정의하여 공개한 것으로, 제공 방식에 있어 매우 다양한 형태로 나타난다. Open API가 확산되는 한편으로 분산 정보자원 검색을 위한 웹 서비스 표준 프로토콜의 개발이 진행되었는데, 그 중 하나가 앞서 살펴본 SRU 프로토콜이고, 또 다른 하나가 OpenSearch이다.

SRU 프로토콜은 Z39.50 프로토콜의 기본 개념은 계승하였지만, XML 형식으로 레코드 구문(Syntaxes)을 통일시키고 서버와 데이터베이스를 단일 구조로 정의하였으며, 사람이 쉽게 해석할 수 있는 CQL이라는 질의언어를 사용하며 질의 방식도 Z39.50 프로토콜의 속성 벡터 형태가 아닌 질의문 내에 색인명이 나타나는 단순 색인 형태로 바뀌었다.

OpenSearch는 웹 사이트나 검색엔진의 검색 결과를 표준적인 방식으로 제공하기 위하여 Amazon의 검색엔진 자회사 A9에서 개발한 것이 그 시작이다. 2005년 3월 Opensearch 1.0이 Web 2.0 컨퍼런스에서 발표되었고, 현재 공식 홈페이지에 OpenSearch 1.1 draft 4가 공개되어 있다. OpenSearch 명세서는 크게 OpenSearch을 지원하는 검색엔진을 XML 형태로 정의하는 것에 관한 OpenSearch description document, 검색 요청 파라미터 등을 설명하는 OpenSearch URL

10) 이지원, "분산 자원 검색과 갱신을 위한 프로토콜에 관한 연구," 제12회 정보관리학회 학술대회 논문집, 2005, pp.281-282.

template syntax, 질의문에 사용할 수 있는 요소들을 설명하는 OpenSearch Query element, 그리고 응답에 사용되는 요소들을 설명하는 OpenSearch response elements의 4가지 기본사항 및 확장하여 사용하는 방법들에 대한 설명을 포함하고 있다. Google, Yahoo, Naver 등 국내외 주요 포털 사이트를 비롯하여 많은 웹 사이트가 OpenSearch 방식의 검색이 가능하며, Internet Explorer 7 버전 이상과 Firefox 2.0 버전 이상의 웹 브라우저, A9.com, OSfeed와 같이 메타검색 웹사이트(Search aggregation websites)가 OpenSearch를 지원하는 클라이언트의 기능을 가지고 있다.[11]

SRU 프로토콜과 OpenSearch는 웹 환경에 적합한 표준 검색 웹 서비스라는 공통점을 가진다. 그러나 OpenSearch는 단순 키워드 검색만을 기본적으로 사용하고 검색 결과도 레코드의 제목과 레코드에 대한 URL을 RSS, Atom 형식으로만 제공하고 레코드의 구조에 대해서는 정의하지 않는다. 따라서 구조화되지 않은 문서들에 대한 단순한 검색에 적합하다. 이에 반해 확장 가능한 Context Set을 정의할 수 있는 CQL 질의언어를 사용하고, 검색 결과에 사용하는 레코드 스키마에 제한을 두지 않은 SRU 프로토콜은 잘 구조화된 문서의 검색, 다양한 이용자의 제어가 필요한 검색에 적합하다.[12] OpenSearch가 보다 단순한 방식의 표준 검색이라면, SRU 프로토콜은 더 정교하고 다양한 방식의 검색이 가능하다.

Ⅲ. SRU Record Update 프로토콜 기능과 특징

1. SRU Record Update 개요

SRU Record Update 프로토콜은 SRU 프로토콜에 대한 계속적인 확장의 결과이다. SRU 프로토콜이 개발되면서, 다른 프로토콜과의 관련성에 대한 연구 및 추가적인 기능들이 꾸준히 제안되었고 이를 반영한 작업들이 지속적으로 진행되었다. 이러한 활동 가운데 갱신(Update) 기능에 대한 필요성이 제기되었고, 기존의 관련된 기술들(WIKI, ATOM, REST/CRUD, WSRS, XML update)을 검토한 결과 적합한 것이 없다는 결론을 내리고, 새로운 프로토콜을 개발하기로 하였다.

SRU Record Update 프로토콜은 2004년 6월 ZING update 프로토콜이라는 명칭으로 초안이 제안되었으며, 같은 해 10월 SRW Editorial Board Meeting에서 검토되었다. 2006년 9월 SRU Record Update 프로토콜로 수정된 초안이 다시 개발, 검토되었고, 2007년 6월 제1판이 발표되었다.

SRU Record Update 프로토콜은 SRU를 기반으로 하여 개발된 것이나, 반드시 SRU를 적용하

11) OpenSearch Homepage, 〈http://www.opensearch.org/〉 [cited 2009. 7. 27].

12) Ralph LeVan, op.cit. ; Ray Denenberg, "OpenSearch and SRU: A Continuum of Searching," *Information Technology and Libraries*, Vol.25, No.3(2006), p.153.

고 있어야 하는 것은 아니며 OAI 프로토콜이나 자체 HTML 인터페이스를 가진 데이터베이스 관리에도 활용할 수 있도록 독립적으로 존재한다. SRU의 새로운 기능(operation)으로 포함될 것인가 별도의 프로토콜로 만들어질 것인가의 논의에서 후자의 방법으로 결정되어 개발된 이유가 여기에 있다.[13]

2. SRU Record Update 기능과 데이터 요소

SRU Record Update은 Explain과 Update 두 가지의 기능을 제공한다. Explain은 서비스에 대한 설명을 제공하며, 실질적인 작업은 Update 기능을 통하여 이루어진다.

다음은 Update 기능과 관련하여 데이터 요소들이 사용된 Namespaces와 요청과 응답 데이터 요소 정의 및 주요 요소에 대한 설명이다.[14]

〈표 3〉 Update Operation 사용 Namespaces

접두어	Namespace URI	설 명
srw	http://www.loc.gov/zing/srw/	SRW 스키마
ucp	info:lc/xmlns/update-v1	Update 추가 요소
diag	http://www.loc.gov/zing/srw/diagnostic/	SRW 진단 스키마

〈표 4〉 Update Operation 요청 파라미터

명 칭	유 형	필수여부
srw:version	xsd:string Value="1.0"	Mandatory
ucp:action	xsd:string	Mandatory
ucp:recordIdentifier	xsd:string	Optional
ucp:recordVersions	sequence	Optional
ucp:recordVersion		Mandatory
ucp:versionType	xsd:string	Mandatory
ucp:versionValue	xsd:string	Mandatory
srw:record		Optional
srw:recordPacking	xsd:string	Mandatory
srw:recordSchema	xsd:string	Mandatory
srw:recordData	srw:stringOrXmlFragment	Mandatory
srw:extraRecordData	srw:extraDataType	Optional
srw:extraRequestData	srw:extraDataType	Optional

13) SRU Listserv Archives, 〈http://sun8.loc.gov/listarch/zng.html〉 [cited 2009. 6. 22] ; SRU Record Update, 2007, 〈http://www.loc.gov/standards/sru/record-update/〉 [cited 2009. 6. 22].

14) *Ibid.*

① action : 요청에서 사용되는 핵심 파라미터이다. 제공되어지는 정보와 함께 서버가 수행하여야 하는 작업(action)이 무엇인지 결정한다. 기본 프로파일에 정의된 작업 유형은 다음의 세 가지이다. 향후 새로운 작업이나 기본 세 가지 작업의 확장을 위한 새로운 식별자가 만들어질 수 있다.

식별자(Identifier)	설 명
info:srw/action/1/create	신규 레코드 생성
info:srw/action/1/replace	현재 레코드를 신규 레코드로 교체
info:srw/action/1/delete	현재 레코드 삭제

② recordIdentifier : 요청과 응답에서 사용되며, 레코드를 식별하는 수단이다. 문자열 형태의 식별자일 수도 있고, 결과 집합(result set)에 대한 연결 또는 결과 집합 내에서의 위치나 쿼리가 될 수도 있다. SRU 1.1 레코드 구조를 사용하는 경우에 제공되며, SRU 1.2 레코드 구조를 사용하는 경우에는 recordIdentifier가 포함되는 형태이기 때문에 별도의 recordIdentifier 파라미터는 생략하도록 권장한다.

③ record : 요청과 응답에서 사용되며, 작업에 필요한 실제 레코드 데이터로서 SRU 레코드 구조와 의미를 동일하게 사용한다. 프로파일과 작업 종류에 따라 각 요청과 응답에서 실제 레코드의 필요 여부가 달라진다. 레코드 자체가 포함될 수도 있고, 레코드로의 연계 정보일 수도 있다.

〈표 5〉 Update Operation 응답 파라미터

명 칭	유 형	필수여부
srw:version	xsd:string	Mandatory
ucp:operationStatus	xsd:string	Mandatory
ucp:recordIdentifier	xsd:string	Optional
ucp:recordVersions	sequence	Optional
ucp:recordVersion		Mandatory
ucp:versionType	xsd:string	Mandatory
ucp:versionValue	xsd:string	Mandatory
srw:record		Optional
srw:recordPacking	xsd:string	Mandatory
srw:recordSchema	xsd:string	Mandatory
srw:recordData	srw:stringOrXmlFragment	Mandatory
srw:extraRecordData	srw:extraDataType	Optional
srw:diagnostics	sequence	Optional
diag:diagnostic		Mandatory
diag:uri	xsd:anyURI	Mandatory
diag:details	xsd:string	Optional
diag:message	xsd:string	Optional
srw:extraResponseData	srw:extraDataType	Optional

① operationStatus : 요청에 대한 작업 결과에 대한 값으로, 정의된 값은 success(서버가 요청된 작업을 성공하였음), fail(서버가 요청된 작업을 성공하지 못했으며, 그 사유에 대하여 추가 정보가 제공될 수 있음), partial(작업의 일부만 성공하였으며, 그 사유에 대하여 추가 정보가 제공될 수 있음), delayed(서버에서 작업이 아직 끝나지 않았음) 4가지이다.

② diagnostics : 클라이언트의 요청을 처리하는 과정에서 생긴 오류에 대한 결과를 진단하여 제공하는 것이다. 현재 URI "info:srw/diagnostic/12"를 사용하는 65개의 진단 목록이 정의되어 있다.

3. 갱신 작업 및 데이터 요소

앞 절에서 살펴보았듯이, SRU Record Update 작업 요청에는 생성, 교체, 삭제의 세 가지 유형이 있다. 기본 프로파일에서는 세 가지 유형에 대하여 요청과 응답에서 사용되어지는 요소에 대하여 정의하고 있다. 작업 유형은 필요에 따라 더 추가될 수 있으며, 그 경우 추가 요소가 요구되어질 수도 있다. 작업 유형의 추가는 세 가지 유형 외의 것일 수도 있지만, 세 가지 유형이 다른 작업 절차를 가지게 되는 경우도 해당된다. 기본 프로파일의 작업 유형별 관련 요소의 정의는 다음과 같다.[15]

가. 생성 작업(Create Action)

구 분		필수여부	설 명
요청	RecordIdentifier	선택	서버에서 신규 레코드의 식별자로 사용하도록 요청하는 것임
	RecordVersions	선택	'숫자'만 포함될 수 있으며, 서버에서 신규 레코드 첫 버전 정보로 요청된 버전 정보를 사용하도록 하기 위함
	Record	선택	레코드를 포함한다면 반드시 생성되는 레코드가 제공되어야 하며, 만일 레코드가 포함되지 않는다면 서버에 빈 레코드 공간을 생성하여 차후 편집이 가능한 참조 정보를 제공해 주도록 요청하는 것
응답	RecordIdentifier	선택	SRU 1.1 레코드 구조를 사용한다면, RecordIdentifier가 제공되어지는 것을 권장함
	RecordVersions	선택	
	Record	선택	서버가 레코드를 어떤 방식으로라도 변경했다면 레코드를 제공하는 것을 권장함

나. 교체 작업(Replace Action)

구 분		필수여부	설 명
요청	RecordIdentifier	필수	교체되는 레코드를 식별함
	RecordVersions	선택	교체되는 레코드를 더 정확히 식별함
	Record	필수	

15) *Ibid.*

응답	RecordIdentifier	선택	SRU 1.1 레코드 구조를 사용한다면, RecordIdentifier가 제공되어지는 것을 권장함
	RecordVersions	선택	
	Record	선택	서버가 레코드를 어떤 방식으로라도 변경했다면 레코드를 제공하는 것을 권고함

교체 요청의 정확성을 위하여 edit replace structure의 사용이 필요할 수도 있다. edit replace structure는 extraRequestData 요소에 포함되며 교체되는 내용에 대한 요청의 의도를 명확하게 표현하기 위하여 사용된다.

데이터 요소	유 형	필수 및 반복 여부
dataIdentifier	xsd:string	선택, 반복불가
oldValue	xsd:string	editRelaceType이 R 또는 D인 경우 필수, I인 경우 생략, 반복불가
newValue	xsd:string	editRelaceType이 I인 경우 필수, D인 경우 생략, R인 경우 선택, 반복불가
editReplaceType	xsd:string values : I=Insert D=Delete, R=Replace	필수, 반복불가

다. 삭제 작업(Delete Action)

현재 존재하는 레코드 전체를 삭제하기 위한 작업이며, 레코드의 일부분만을 삭제하고자 한다면 edit replace structure를 사용하여 교체 작업으로 요청하여야 한다.

구 분		필수여부	설 명
요청	RecordIdentifier	선택	삭제되는 레코드를 식별하며, 만일 RecordIdentifier가 없다면 Record는 반드시 존재하여야 함
	RecordVersions	선택	삭제되는 레코드를 더 정확히 식별함
	Record	해당시 필수	RecordIdentifier가 없는 경우에는 필수 RecordIdentifier가 있는 경우에도 함께 쓰일 수 있으나, 서버에서 반드시 cross-check를 할 필요는 없음
응답	RecordIdentifier	선택	SRU 1.1 레코드 구조를 사용한다면, RecordIdentifier가 제공되어지는 것을 권장함
	RecordVersions	선택	삭제되는 레코드를 더 정확히 식별함
	Record	선택	삭제된 레코드를 포함함

4. 갱신 프로토콜 비교

SRU Record Update 프로토콜과 같이 자원 특히 메타데이터의 갱신을 위한 프로토콜로 Z39.50 프로토콜 갱신 확장서비스와 OAI-PMH 프로토콜이 있다.

Z39.50 프로토콜 확장서비스(Extended-Services)는 Z39.50-1995(제3판)에 추가된 것으로,

결과 집합 저장(Persistent Result Set), 자료 주문(Item Order) 등 6가지 작업(task)와 함께 데이터베이스 갱신(Database Update) 작업이 포함되어 있으며, 표준문서에 확장서비스 요청과 응답에 사용되는 공통 파라미터 및 고유 파라미터가 정의되어 있다.

또한 Z39.50 프로토콜 갱신 확장서비스와 관련된 프로파일로 UCP(Union Catalog Profile)가 있다. UCP는 Z39.50 프로토콜의 갱신 확장서비스를 이용하여 분산 환경에서 데이터베이스 관리를 효과적으로 할 수 있도록 갱신 관련정보를 추가적으로 정의한 것으로, 1996년 처음 개발되었으며, 1999년 국제 등록 프로파일로 승인되었다. UCP에는 서지, 전거, 소장 데이터의 갱신, 일시 정보(date/time stamp)를 이용한 동시 갱신 제어, 통합과 전역변경, 처리 결과에 따른 메시지 등이 정의되어 있다.[16]

그러나 갱신 확장서비스와 UCP를 실제 적용한 사례는 많지 않은데, 이것은 표준 프로토콜과 프로파일이 복수 계층(확장서비스, 태스크패키지) 사용, 일괄/온라인 갱신, 단일/복수 레코드 갱신 등을 정의하고 있기 때문에 실질적으로 구현하기에 복잡하기 때문이다. KERIS 종합목록시스템은 데이터베이스 갱신을 위하여 현재 Z39.50 프로토콜 갱신 확장서비스를 적용하고 있으나, UCP Profile은 적용하지는 않으며, 자체적으로 갱신 작업 유형 및 갱신 요청 방법을 정의하였다.[17]

SRU Record Update 프로토콜은 Z39.50 프로토콜 갱신 확장서비스와 UCP의 기본 개념은 유지하고 있으나, SRU 프로토콜과 마찬가지로 웹 환경에 적합하고 구현이 용이하다는 점에서 Z39.50 프로토콜 갱신 확장서비스와 차이가 있다.

OAI-PMH(Open Archives Initiative-Protocol for Metadata Harvesting; 이하 OAI 프로토콜)은 개방 아카이브(archives)의 메타데이터를 수집(harvesting)하기 위한 프로토콜이다. 아카이브는 자원 특히 디지털 자원의 저장소를 의미하며, 리포지토리(repository)라고 부르기도 한다. 2001년 1월 제1판이 발표되었고, 2002년 6월 제2판이 발표되었다. OAI 프로토콜은 SP(Service Provider)와 DP(Data Provider)의 요청/응답 방식을 규정한 것이다. SP는 DP들로부터 메타데이터를 수확하여 검색, 브라우징, 원문제공 등과 같은 부가가치 서비스를 제공하고, DP는 자체적으로 각종 디지털 자원을 수집, 보유하여 SP의 수확 요청에 대응하여 적합한 메타데이터를 제공한다.

OAI 프로토콜은 XML 형식을 적용하고 웹 환경을 기본으로 개발되었다는 점에서 SRU Record Update와 유사하다. 그러나 SRU Record Update의 경우 클라이언트의 역할을 하는 로컬시스템에서 서버의 역할을 하는 센터시스템으로 갱신을 원하는 데이터를 보내는 Push 방식인 반면, OAI 프로토콜은 데이터를 통합하여 서비스한다는 점에서 센터시스템과 유사한 역할을 하는 SP가 단위시스템인 DP로부터 데이터를 수확해 오는 Pull 방식이다. 또한 SRU Record Update가 단일 레코드의

16) Janifer Gatenby, *Lesley Bezear and Judith Pearce, Union Catalogue Profile*, 1999, 〈http://www.nla.gov.au/ucp/irp.html〉 [cited 2009. 6. 22].

17) 한국교육학술정보원, *KERIS Z39.50 Configuration Guide Ver 2.2*, 2008, 〈http://www.riss4u.net/libn_ch/ulist/download/KZ_GUIDE_Ver2.2.pdf〉 [cited 2009. 7. 27].

갱신(추가, 교체, 삭제)을 즉각 요청하고 이에 대한 처리결과를 바로 받아보는 온라인 대화식 방식이라면, OAI 프로토콜은 복수의 레코드를 일정(schedule)에 따라 일괄로 처리하는 방식이다.[18]

OAI 방식의 데이터 수집은 데이터 작성자가 각각 별도의 업로딩 작업을 하기 않기 때문에 SRU Record Update 방식에 비하여 수집이 용이한 반면, 갱신 작업에 있어 정확한 요청을 하기 어렵고, 데이터의 품질을 보장하기 어렵다는 단점이 있다.

Ⅳ. 프로토콜 구현 사례

1. The European Library

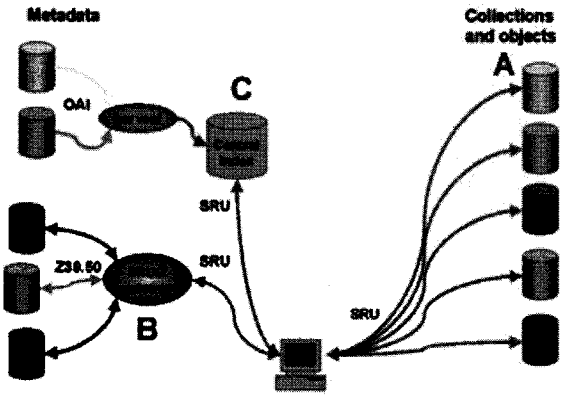
The European Library(이하 TEL)는 유럽 국가도서관의 자원들을 통합하여 제공하는 서비스[19]이다. 먼저 유럽 국가 도서관장협의회(Conference of European National Librarinans, 이하 CENL) 주관으로 유럽 국가도서관의 접근과 이용을 확대하여 유럽 시민과 연구자들을 위한 새로운 정보서비스 시스템을 개발한다는 목표 아래 2001년 2월부터 2003년 7월까지 TEL 구축 프로젝트가 수행되었고, 2005년 3월 실 서비스가 개시되었다. 9개 국가도서관으로 시작한 TEL은 매해 그 범위를 확대하여 48개 국가도서관의 1억 5천 만 건 데이터를 35개의 언어 인터페이스로 서비스하기에 이르렀다. 서비스되고 있는 자원은 서지데이터와 도서 원문, 포스터, 사진, 지도, 시청각 자료 등의 디지털 자원을 모두 포함하며, 특히 디지털 자원의 확대에 많은 지원이 이루어질 계획이다.[20]

TEL의 특징 중에 하나는 장서(collections)라는 단위로 특정 주제나 특정 분야의 자료, 또는 특정 OPAC을 다루고 있다는 것이다. 참여 도서관은 장서 단위로 자료를 TEL에 추가할 수 있고, 이용자는 필요에 따라 전체를 통합하거나 개별 장서 단위로 접근할 수 있다.[21]

TEL은 유럽의 다양한 참여 도서관의 자원을 통합하여 서비스한다는 특성상 상호운용성의 확보가 매우 중요한 요소이다. TEL 포털 서비스를 위한 검색 모델 선정을 위한 사전 검토 작업 결과 단순하면서도 계속적인 장서의 추가와 변경에 신속히 대응할 수 있다는 장점을 가진 SRU 프로토콜이 기반 프로토콜로 채택되었다. 또한 Z39.50 프로토콜과 OAI 프로토콜과도의 상호호환성을 확보하는 방안도 함께 고려되었다. 〈그림 1〉은 TEL 검색의 기본 구조이다. A방식은 SRU 프로토콜

18) Eric L. Morgan, "An Introduction to the Serarch/Retrieve URL Service(SRU)," *Ariadne Issue* 40, 2004, 〈http://www.ariadne.ac.uk/issue40/morgan/〉 [cited 2009. 7. 27].
19) TEL Homepage, 〈http://www.theeuropeanlibrary.org〉 [cited 2009. 7. 27].
20) Fleur Stigter, Sally Chambers, and Louise Edwards, "The Europena Library - gateway to the resources of Europe's national libraries," *IFLA Journal*, Vol.34, No.3(2008), p.258.
21) 표순희, 이재윤, "유럽의 국가 도서관 통합 서비스에 대한 연구," 정보관리연구, 제36권, 제3호(2005. 9), p.84.

을 지원하는 참여기관의 서버에 직접 접근하여 검색하는 것이다. B방식은 Z29.50 프로토콜만을 지원하는 참여기관의 경우 SRU/Z 게이트웨이를 통해 두 가지 프로토콜의 호환을 가능하게 하고, 이를 다시 SRU 프로토콜로 검색해 오는 방식이다. 마지막 C방식은 OAI 프로토콜을 지원하는 참여기관의 경우 메타데이터를 수집해 와서 종합 색인(Central Index)를 구축하고, 마찬가지로 이를 다시 SRU 프로토콜로 검색해 오는 것이다.[22]



(출처 : http://www.theeuropeanlibrary.org/handbook/)

〈그림 1〉 The European Library 기본 구조

TEL의 성공 사례는 SRU 프로토콜이 여러 기관에 산재된 데이터에 대한 메타검색, 다양한 언어로 구성된 대량의 데이터 집합들에 대한 검색에 매우 적합하다는 것을 보여주고 있다.

2. OCLC WorldCat

OCLC는 SRU 프로토콜이 개발되는 초기부터 이에 관심을 가지고 연구 및 테스트를 계속하여 왔고, SRU 프로토콜 관련 소프트웨어와 소스도 공개하였다. OCLC의 대표적 데이터베이스인 WorldCat을 비롯하여 OCLC 회원 기관 레지스트리 등을 SRU 프로토콜을 통해 검색할 수 있다.[23]

SRU 프로토콜을 통해 WorldCat 검색에 제공되는 레코드 스키마는 MARCXML과 더블린 코아 형식이며, 서명, 저자, 출판년, OCLC 제어번호 등으로의 정렬도 가능하다. frbrGrouping 파라미터가 "on"(default)일 경우 검색 결과를 유사한 판으로 묶어서 해당 판들의 첫 번째 결과만을

22) The Europeanl Library Handbook Homepage, 〈http://www.theeuropeanlibrary.org/handbook/〉 [cited 2009. 7. 27].

23) OCLC WorldCat은 접근성 확대를 위하여 SRU 프로토콜 방식 외에도 OpenSearch 방식과 식별자(OCLC 제어번호, ISBN) 이용한 검색 API 방식을 제공하고 있다.

대표 레코드로 제공한다. 시스템에서 제공하는 기본값(defaults)은 키워드 검색, MARCXML 형식, 시작 위치 1, 최대 레코드 개수 10, 그리고 적합성 기준으로의 정렬이다. 검색 요청 필수 파라미터인 version, operation도 version=1.1, operation= searchRetriev인 경우에는 생략하여도 기본값으로 적용한다. 다음은 저자키워드 *mann*, 서명키워드 *faustus*에 대한 검색 요청 예이다.

http://www.worldcat.org/webservices/catalog/search/sru?query=srw.au+%3D+%22mann%22+and+srw.ti+%3D+%22faustus%22&wskey=[key]

마지막 wskey 파라미터 값은 검색 요청에 대한 인증키 값을 위한 것이다.[24]

SRU Record Update 프로토콜은 OCLC WorldCat과 네덜란드 종합목록(이하 NCC : Nederlandse Centrale Catalogus)과의 데이터 실시간 갱신을 위하여 처음으로 구현되어, 2008년 2월부터 실제 적용되고 있다. SRU Record Update 프로토콜을 적용하기 이전에는 새로운 데이터가 일괄처리(batches) 방식으로 추가되었고 따라서 처리가 될 때까지 즉 NCC의 데이터가 WorldCat에 반영될 때까지는 일정 기간이 소요되었다. 5개월간의 운영기간 동안 NCC의 약 150만건의 소장정보와 23만건의 서지데이터가 추가되었고, 두 데이터베이스간의 갱신 처리 시간은 평균 5초 이내였다. SRU Record Update 프로토콜은 갱신의 신속성 뿐 아니라 포괄성도 보장하는데, 이는 일괄처리 과정에서 생길 수 있는 데이터 누락의 위험을 제거하고 오류 데이터를 최소화할 수 있도록 즉각적으로 검증 결과를 제공하기 때문이다. 2009년 1월 두 번째 SRU Record Update 프로토콜의 구현이 호주국가종합목록과 WorldCat간에 이루어졌다.

OCLC는 전용 클라이언트인 Connextion을 이용하지 않고 데이터의 일괄처리 방식으로 WorldCat에 기여하는 비중이 커짐에 따라 SRU Record Update 프로토콜을 이용하는 새로운 연계 방식에 매우 긍정적인 평가를 내리고 있으며, 이를 활성화시킬 방안을 모색하고 있다.[25]

3. 한국교육학술정보원 종합목록

한국교육학술정보원(Korea Research & Education Information Service, KERIS) 종합목록은 1998년 운영을 시작하였고, 운영 초기부터 Z39.50 프로토콜을 이용한 검색 및 데이터 업로딩을 지원하였으며, 이용의 편의성 때문에 대부분의 기관이 전용 클라이언트인 UNICAT 대신에 자관 도서관

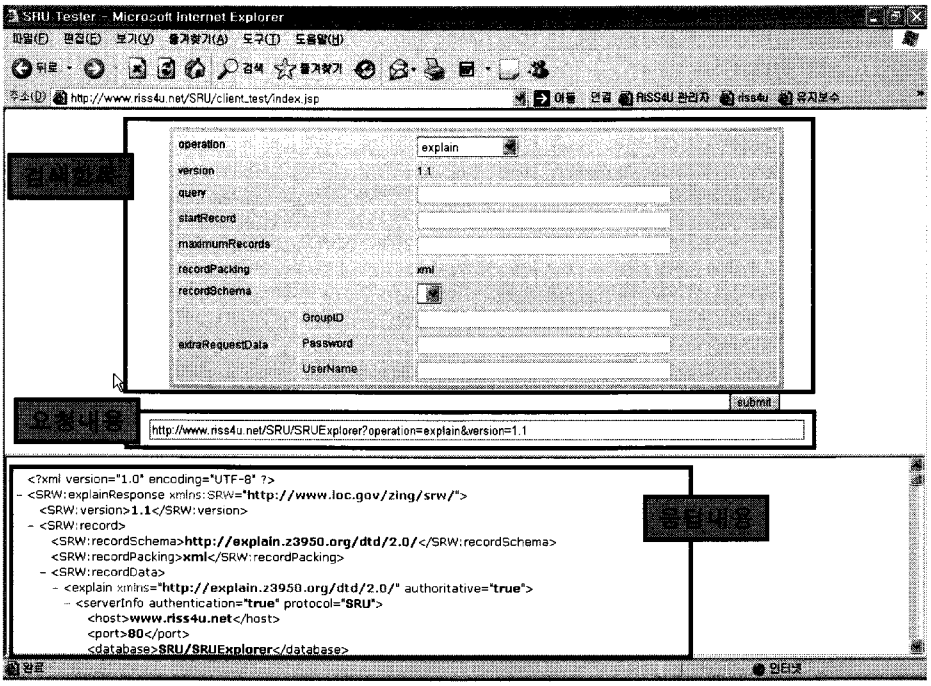
24) WorldCat Search API, 〈http://worldcat.org/devnet/wiki/SearchAPIDetails〉 [cited 2009. 7. 27].

25) Janifer Gatenby, *SRU Update and the Dutch Union Catalogue*, 2008, 〈http://community.oclc.org/metalogue/archives/2008/07/sru-update-and-the-dutch-union.html〉 [cited 2009. 7. 27] ; OCLC, *SRU Update and WorldCat*, 2009, 〈http://www.oclc.org/enews/2009/08/en_sru.htm〉 [cited 2009. 7. 27].

시스템의 Z39.50 클라이언트 기능을 사용하여 공동편목에 참여하고 있다.

웹 기반의 표준 프로토콜도 지원하고자 2006년 SRU 프로토콜 서버를 개발하였으며, 이와 관련한 구성 지침서를 공개하였다.[26] 종합목록 SRU 서버는 Explain과 SearchRetrieval 기능을 지원하며 제1.1판을 준수하고 있다. 레코드 스키마는 서지레코드 전체 내용에 대해서는 MARCXML 형식, 검색리스트용 기본 서지사항에 대해서는 자체 정의한 형식을 제공하고 있다. 질의문 구성을 위한 Context Set은 Z39.50 BIB-1 Context Set과 일부 요소에 대해서는 자체 정의하여 사용하고 있다. 종합목록 기관회원 인증을 위해서 extraRequestData 파라미터도 사용하고 있다.

또한 종합목록 SRU 서버를 통한 검색을 테스트하고, 요청과 응답 내용을 확인할 수 있도록 테스트 페이지도 제공하고 있다.



〈그림 2〉 KERIS 종합목록 SRU 검색 테스트 페이지

아직까지 SRU 프로토콜을 사용하여 종합목록을 검색하는 기관은 확인되지 않았는데, 이는 국내 도서관시스템 상당부분이 클라이언트 시스템 기반으로 만들어져 있으며 이미 Z39.50 프로토콜이 구현되어 있어, 웹 기반 기술을 전제로 하는 SRU 프로토콜 적용 필요성이 적기 때문이다. 또한

26) 한국교육학술정보원, *KERIS SRU Configuration Guide Ver 1.0*, 2006, 〈http://www.riss4u.net/libn_ch/ulist/download/SRU_Conf_Guide_uni.pdf〉 [cited 2009. 7. 27].

공동편목을 위해서는 데이터 업로드가 가능하여야 하는데, SRU Record Update 프로토콜이 최근에야 공식 발표가 되어 아직은 지원하지 못하고 있기 때문이다. 향후 웹 기반 도서관시스템이 일반화되고, SRU Record Update 프로토콜도 지원되면 활용이 많아지리라 예상한다.

V. 결론 및 제언

인터넷, 웹의 출현과 정보통신기술의 발전은 정보자원의 폭발적인 증가와 더불어 이에 대한 접근과 활용을 가능케 하였다. 대부분의 도서관과 정보서비스 기관은 다양한 외부 정보자원들을 함께 또한 내부 자원들과 통합하여 어떻게 효과적으로 이용자에게 제공할 것인가에 큰 관심을 가지고 지속적으로 그 방법을 개발하고 실행하여 왔다. 또한 최근 참여, 개방, 연계, 통합이라는 모토 아래 진행되고 있는 Library 2.0 물결 가운데 많은 기관들은 보유하고 있는 자원들을 적극적으로 개방하여 더 많이 활용시키고자 노력하고 있다.

2002년 처음 공식 발표된 SRU 프로토콜은 정보검색의 표준을 정의한 웹 서비스 프로토콜이다. 도서관 분야에서 널리 사용되어 온 Z39.50 프로토콜의 기본 개념은 유지하면서 XML, HTTP 등의 웹 기반 기술을 사용함으로써 구현의 편리성 및 확장성을 가질 수 있게 하였다. 또 다른 웹 서비스 검색 프로토콜인 OpenSearch와 비교하면 질의 언어와 레코드 스키마의 확장이 가능하기 때문에 더 정교하고 다양한 방식의 검색이 가능하고 잘 구조화된 문서의 검색 및 이용자의 다양한 제어가 필요한 검색에 적합하다. 미국, 유럽의 여러 정보서비스 기관에서 SRU 프로토콜을 지원하는 데이터베이스 등을 공개하고 있다.

SRU Record Update 프로토콜은 원격지의 정보자원에 대한 갱신을 표준적으로 정의하기 위해서 수 년간의 검토와 개발을 거쳐 2007년에 공식 발표된 갱신 프로토콜이다. 개발 초기부터 주도적으로 참여해 온 OCLC PICA[27]가 자신들이 개발한 네덜란드, 호주 종합목록 시스템과 OCLC WorldCat과의 실시간 갱신 작업을 위하여 SRU Record Update 프로토콜을 적용하여 성공적이라는 평가를 내리고 있다.

국내에서는 아직까지 두 프로토콜에 대한 인식이 부족한 편이고, 실제 구현한 사례도 거의 없다. 표준을 적용한다는 것이 장기적으로는 상호운용성을 보장하고 신뢰성과 확장성을 가진다는 장점을 가지지만, 적용을 위한 개발 단계에서는 제약사항이 될 수도 있고 따라서 표준 준수를 고려하지 않는 경우에 비해 시간과 노력을 더 많이 필요로 할 수도 있다. 시스템을 구현하는데 있어 충분한 계획과 개발 기간을 갖기가 어려운 국내 현실을 고려할 때, 표준에 대한 관심과 연구 및 사례 공유가 더욱 필요하다고 생각한다. 특히 정부 주도의 사업과 같이 많은 기관과의 연계를 전제로 하는

27) 유럽 PICA가 OCLC에 통합되었을 초기에 사용한 명칭이나, 현재 PICA 명칭은 사용하지 않는다.

정보자원을 구축하거나 개선할 경우에는 초기 계획 단계부터 표준에 대한 검토가 반드시 필요할 것이다.

SRU 프로토콜을 적용한 사례에서 TEL의 경우 OAI-PMH나 Z39.50과 같은 다른 프로토콜과 연동하여 활용하고 있었고, OCLC의 경우 WorldCat 검색의 여러 방안 중에 또 하나의 선택사항으로 제공하고 있음을 확인할 수 있었다. 국내에서도 자체 정의한 Open API 방식과 더불어 선택의 폭을 넓히고 보다 신뢰성 있고 확장 가능한 연계 방식을 제공한다는 측면에서 SRU 프로토콜 적용을 고려하여야 할 것이다. 또한 외부의 정보자원에 대한 갱신 작업이 지속적으로 필요한 시스템의 경우 특히 현재 일괄처리 방식으로 갱신 작업을 하고 있는 경우에는 정보자원 최신성과 품질의 신뢰성을 제고한다는 측면에서 SRU Record Update 프로토콜의 적용을 검토할 필요가 있다.

이 연구가 아직은 관심이 적은 두 표준 프로토콜에 대한 이해를 넓히고, 또한 자관의 정보자원을 외부에 효과적으로 제공하며, 외부 정보자원을 적절히 활용하려는 도서관 및 정보서비스 기관들에게 상호운용성 보장을 위한 실제적인 도움이 되기를 바란다.

〈참고문헌은 각주로 대신함〉