DOI 수집 API 개발에 관한 연구

A Study on the Development of DOI Lookup API

김 선 태* · 예 용 희**
Sun-Tae Kim · Yong-Hee Yae

차 례

초 록

CrossRef는 DOI 식별자 및 학술메타정보를 수집할 수 있는 다양한 쿼리(OpenAPI)를 제공하고 있으며, 이를 활용해 출판사 및 정보서비스 기관에서 다양한 서비스를 개발하도록 독려하고 있다. 본 연구에서는 CrossRef에서 제공하는 쿼리를 면밀히 분석하였으며, 다양한 방법으로 DOI를 자동수집 하는 API를 개발하였다. 또한 자체적으로 학술메타데이터를 보유한 기관이 개발된 API를 활용할 수 있는 방법을 제시하였다.

키 워 드

DOI, CrossRef, OpenAPI, OpenURL

* 한국과학기술정보연구원 서비스개발팀 선임연구원
(Senior Researcher, Knowledge Asset Team, Korea Institute of Science and Technology Information, stkim@kisti.re.kr)
** 한국과학기술정보연구원 서비스개발팀 책임연구원
(Principal Researcher, Knowledge Asset Team, Korea Institute of Science and Technology Information, yaeyh@kisti.re.kr)
• 논문접수일자 : 2008년 1월 9일
• 게재확정일자 : 2008년 3월 14일

ABSTRACT

CrossRef provides a various queries(OpenAPIs) which can be used for DOI & meta data lookup. CrossRef encourages publishers and library societies to develop diverse system by using the queries. In this thesis, CrossRef's queries are analyzed and DOI Lookup API which could automatically lookup the DOI by various methods was developed. I proposed that how institutions having their own meta data can use the developed API.

KEYWORDS

DOI, CrossRef, OpenAPI, OpenURL

1. 서 론

1.1 연구배경

2000년 초 학술분야의 출판사 대표들이 모여 CrossRef를 운영하는 Publishers International Linking Association(PILA)을 결성하였다. CrossRef는 출판사들에 의해 공동으로 운영되는 독립된 조직으로서, 출판사간 협력을 이끌어내 연구자가 주요 학술자료를 손쉽게 획득 할 수 있도록 하는 것을 주요한 목적으로 한다. CrossRef는 학술 출판물에 부여된 DOI를 공식적으로 등록하는 에이전시 역할을 수행하고 있으며, 출판사간 인용 링킹 시스템(cross-publisher citation linking system)을 운영함으로써, 연구자가 특정 출판사의 플랫폼에서 제공되는 인용문헌 링크를 이용해 타겟(Target) 출판사에서 제공하는 전자원문 서비스를 받을 수 있도록 하고 있다. 물론 인용된 전자원문의 이용가능 여부는 타겟 출판사의 이용권한 정책에 부합 되어야만 한다.

이렇듯 많은 출판사가 참여하여 운영되는 CrossRef는 출판사가 보유하고 있는 전자자원에 대한 이용률을 더욱 높이고자 하는 것이 궁극적인 목적일 것이다. 따라서 참여하는 출판사는 DOI가 부여된 자원이 참조한 레퍼런스에 대해서도 DOI 및 메타데이터 정보를 함께 제공하고 있다. 이로써 DOI를 통해 출판사간 참조링킹이 가능해졌으며, 후방향 링킹(Backward Linking) 뿐 아니라 전방향 링킹(Forward Linking)서비스까지 가능하게 되었다. 이러한 서비스들의 핵심에는 DOI 식별자가 있다. 구글스칼라나 마이크로소프트사의 라이브서치 등 학술포털에서 사용하는 링킹의 메커니즘과 출판사 플랫폼에서 내부적으로 운영되는 링킹의 핵심은 DOI 식별자이다. 또 하나의 활용사례를 들면, OpenURL로 전달된

메타정보가 DOI 뿐일 경우 링킹서버는 DOI를 CrossRef에 보내고 해당되는 메타정보를 전달받게 된다. 다음으로 메타정보를 분석해 라이선스 유무를 판단하여 이용자에게 서비스페이지를 만들어 서비스 하게 된다. 이러한 서비스는 정보자원 간 원활한 연계서비스를 가능하도록 하는 주요한 기능이다.

링킹의 핵심 알고리즘으로 DOI는 주요한 역할을 하고 있다. 정보서비스를 하는 기관에서는 보유하고 있는 메타데이터를 활용한 DOI 수집 프로세스를 통해, 기 구축된 메타데이터 및 학술지 전거-이명 정보를 보강하고, 수집된 DOI를 활용한 원문서비스를 이용자에게 제공 할 수 있게 되었다.

하지만 CrossRef와 DOI에 대한 국내·외 선행연구는 CrossRef 서비스와 DOI 자체에 대한 조사 및 특정 쿼리방식(파이프방식)에 국한되어 있어서, 실제 DOI를 수집하기 위해 사용될 수 있는 다른 타입의 쿼리에 대한 분석 및 API 개발이 필요하였다.

본 연구에서는 학술정보 서비스기관에서 CrossRef이 제공하는 쿼리를 개별적으로 분석 할 필요 없이, 손쉽게 활용할 수 있는 DOI 자동수집 API를 개발하고, 이를 활용할 수 있는 방법을 제시한다.

1.2 연구내용 및 구성

학술정보가 효율적으로 유통되도록 하기 위해 CrossRef는 다양한 쿼리(OpenAPI)를 제공하고 있다. 따라서 DOI를 수집하여 가치가 부가된 새로운 서비스를 학술정보 이용자들에게 제공하는 것이 가능해졌다. 학술메타정보를 활용해 DOI 식별자를 검색할 수 있을 뿐 아니라, DOI를 활용해 CrossRef에 구축된 메타정보를 검색할 수 있다. 본 연구에서는 먼저 DOI 자동수집 API의 오류를 최소화 하기위해 CrossRef에서 제공하는 6가지의 쿼리를 면밀히 검토하였다. 각각의 특징을 분석했으며, 파라미터 및 활용방법을 비교하였다. DOI 자동수집 API개발 시에는 DOI를 활용한 메타데이터 수집 모듈은 개발하지 않았으며, CrossRef에서 제공하는 XML 쿼리 업로드 로직을 일부 활용하였다.

2장에서는 CrossRef가 제공하는 쿼리에 대한 분석내용을 기술하였으며, 3장에서는 각각의 API에 대해 특징 및 활용방법을 기술하였다. 4장에서는 결론 및 향후 연구계획에 대해 기술하였다.

2. CrossRef 쿼리분석

CrossRef에서 제공하는 쿼리의 종류는 아래와 같이 6가지이다. 각각은 활용방법과 목적이 상이하다. OCI 방식은 Telnet 프로토콜을 사용하고, 나머진 모두 HTTP 프로토콜을 사용하고 있다.

•XML Queries

•Journal Pipe'd Queries

•Conference proceeding & books

Pipe'd Queries

•The HTTP Queries Interfaces

•OCI(Open Channel Interface)

•OpenURL Queries

다음은 각각의 쿼리를 분석한 내용이다.

2.1 XML Queries

XML 형태로 쿼리를 작성하여 CrossRef 시스템에 POST 방식으로 전송을 하면, CrossRef 시스템의 큐에 저장이 된 후 쿼리에 명시된 이메일로 매핑결과(resolving result)가 전송된다. 아래 〈그림 1〉은 XML 쿼리의 내용과 주요 엘리먼트의 목적을 보여 주고 있다. 첫 번째 〈doi_batch_id〉 엘리먼트는 쿼리를 전송한 후에 CrossRef의 'Show my submission queue' 서비스를 통해 작업내용을 추적할 수 있는 아이디를 입력한다. doi_batch_id값은 쿼리를 전송하는 로컬기관에서 관리하는 ID값으로 본 프로젝트에서는 doi_batch_id의 최종 값을 보관하는 파일을 생성해 관리하도록 하였다.

2.2 Journal Pipe'd Queries

'|' 구분자에 의해 메타데이터를 일정한 순서에 맞게 조합해 전송하는 방식이다. 필수필드는 ISSN이나 저널명, 첫 번째 저자의 성 혹

<?xml version = "1.0" encoding="UTF-8"?>
<query_batch version="1.0" xmlns = "http://www.crossref.org/qschema/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<head>
<email_address>kkoscher@crossref.org</email_address>
<doi_batch_id>trackingid1</doi_batch_id>
</head>
<body>
<query key="MyKey1" enable-multiple-hits="false"
forward-match="false">
<journal_title match="fuzzy">Chem Commun</journal_title>
<author match="exact">Mouton</author>
<issue match="exact">8</issue>
<volume/>
<first_page>863</first_page>
<year>2001</year>
</query>
</body>
</query_batch>

Query Match Alerts

The 'forward-match' attribute is used to enable Query Match Alerts. This feature eliminates the need for users to repeatedly poll CrossRef for queries that do not initially return a DOI. When a query is marked to enable alerts the CrossRef system will automatically send an email containing the query results to the specified address

Process Tracking

Tracking IDs allow you inspect the status of a batch job uploaded to the system. By executing an HTTP request in the following form you can retrieve information on when a job was processed

http://doi.crossref.org/servlet/submissionDownload?usr=<USER>&pwd=<PWD>&doi_batch_id=trackingid1&type=result

Query results are not saved by the system, so the tracking ID function can not return the results of a query job.

Multiple Hits

Setting the enable-multiple-hits attribute to "true" instructs the system to return several results if it is unable to reduce the candidate matches to a single item. Normally, if a query results in an ambiguous result set the system returns no data at all. This behavior is necessary since most processes querying CrossRef are automated and are incapable of deciding amongst multiple DOI records. However, in an editorial environment when a person is involved in the query multiple results may be valuable.

〈그림 1〉 XML Query

〈표 2〉 저널 쿼리포맷(10개 필드)

필드	필드확장	값에 대한 설명
ISN	ISSN	구분자로 ','을 사용하여 인쇄 및 전자전널의 ISSN을 기술
TTL	Journal Title	완전한 저널명 및 축약형 저널명(조합가능)
NAM	First Author	첫번째 저자의 성
VID	Volume ID	볼륨(숫자, 문자 모두 가능)
IID	Issue ID	이슈(숫자, 문자 모두 가능)
PID	Page ID	페이지(정수사용, 페이지 번호에서 선행하는 0은 제외함, 페이지 앞에 page, PP 등은 제외함)
YNO	Year	년도(숫자를 사용하며, 문자는 오류를 발생함. 두 자리 숫자는 19XX로 간주함)
TYP	Type	전문(full_text), 초록(abstract_only) 혹은 서지레코드(bibliographic_record)
KEY	Unique key	결과값에 그대로 반환되는 값으로 쿼리결과를 받아 전송되었던 쿼리와의 매핑작업에 사용할 수 있음.
DOI	DOI	결과값으로 전송되는 DOI(전송되는 쿼리에서는 항상 공백으로 남아 있음)

출처 : http://www.crossref.org/04intermediaries/25query_spec.html

은 시작페이지이며, 나머지 필드는 옵션이다. 가능한 많은 옵션 필드가 쿼리에 포함 될수록 정확한 매핑이 가능해 진다. 〈표 2〉를 준수하는 포맷으로 전송될 데이터는 아래와 같이 구성된다.

예 : 01291831,01291831|International Journal of Modern Physics C | Huang|11||287|2000|full_text||

2.3 Conference proceeding & books Pipe'd Queries

컨퍼런스 프로시딩은 컨퍼런스 행사, 프로시딩 출판, 프로시딩 각각의 컨퍼런스 논문들의 정보를 포함하고 있을 수 있다. VTL은 프로시딩 타이틀과 컨퍼런스 두문자의 조합을 찾는데 사용되며, STL은 사용되지 않으므로 빈칸을 유지한다. 컨퍼런스 프로시딩은 볼륨과 판차 정보를 가지고 있지 않다. 따라서 EID필드는 빈칸을 유지한다.

도서는 chapter와 section에 대한 타이틀 정보뿐 아니라 서명, 칼럼, 판차 메타정보를 가지고 있다. 또한 시리즈 서명 정보도 갖고 있을 수 있다(〈표 3〉 참조). 아래는 예를 보여준다.

예 : 078037293X||10th IEEE International Conference on Fuzzy Systems (Cat No01CH37297) FUZZY-01|Ha|1||332|2001|||

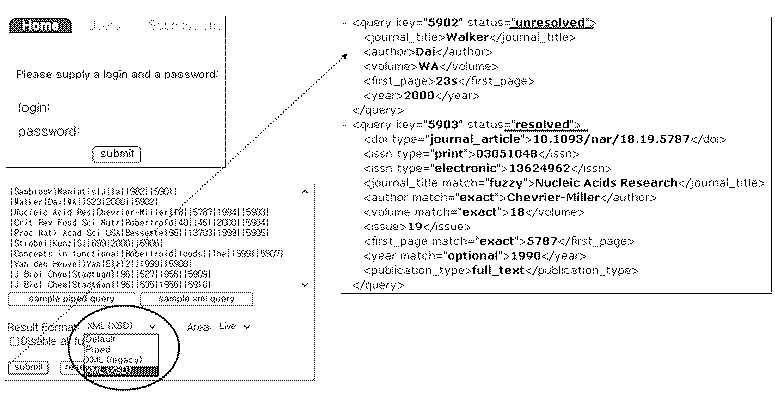
〈표 3〉 단행본/프로시딩 쿼리포맷(12개 필드)

필드	필드확장	값에 대한 설명
ISN	ISSN	구분자로 ‘,’을 사용하여 인쇄 및 전자 ISSN을 기술
STL	Serial Title	완전한 시리얼 및 축약형 시리얼명(조합가능)
VTL	Volume Title	완전한 도서명
NAM	First Author	첫번째 저자의 성
VID	Volume ID	볼륨(숫자, 문자 모두 가능, 예 : fall, Q1)
EID	Edition	판차(숫자, 문자 모두 가능, 도서만 사용)
PID	Page	페이지(정수사용, 페이지 번호에서 선행하는 0은 제외함, 페이지 앞에 page, PP 등은 제외함)
YNO	Year	년도(숫자를 사용하며, 문자는 오류를 발생함. 두 자리 숫자는 19XX로 간주함)
CNO	Component #	도서나 컨퍼런스, 프로시딩을 구성하는 Chapter, section or part
TYP	Type	전문(full_text), 초록(abstract_only) 혹은 서지레코드(bibliographic_record)
KEY	Unique key	결과값에 그대로 반환되는 값으로 쿼리결과를 받아 전송되었던 쿼리와의 매핑작업에 사용할 수 있음.
DOI	DOI	결과값으로 전송되는 DOI(전송되는 쿼리에서는 항상 공백으로 남아 있음)

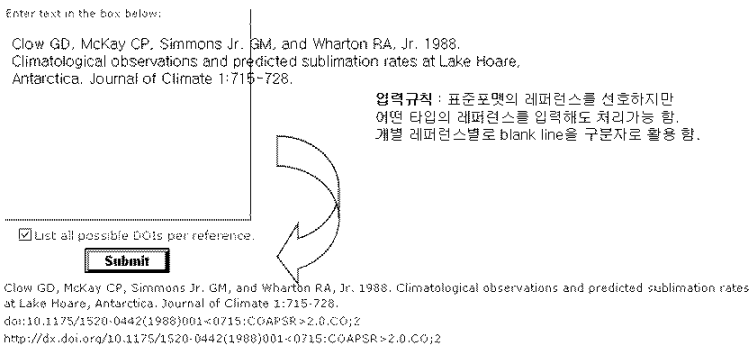
출처 : http://www.crossref.org/04intermediaries/25query_spec.html

2.4 HTTP Query Interfaces

아래의 http://doi.crossref.org 웹 인터페이스에서 CrossRef의 계정을 이용하여 로그인을 하고 쿼리를 입력한 후 결과를 기다린다. 〈그림 2〉 원의 내용과 같이 현재 결과 뷰는 크게 4가지 포맷으로 제공하고 있으며, XML(XSD)를 선호하고 있다. 쿼리 엘리먼트의 상태 속성



〈그림 2〉 동기화된 방식의 HTTP인터페이스



〈그림 3〉 입력된 레퍼런스 텍스트를 이용한 DOI LookUp

〈표 4〉 비동기 방식 인터페이스의 파라미터

폼 필드	설명	가능한 값	필수 여부	디폴트
operation	전송타입에 따라 다름	•doMDUpload : For metadata submissions •doXSDMDUpload : same as doMDUpload •doQueryUpload : For Query submissions •doDOIQueryUpload : For DOI query submissions •Submit Batch File : same as doMDUpload (for backward compatibility)	X	doMDUpload
login_id	CrossRef 제공 아이디		O	
login_passwd	CrossRef 제공 패스워드		O	
area	제출을 위한 지정분야	•live •test	X	live
Content parts				
fname	Submission contents		O	

출처 : http://www.crossref.org/04intermediaries/25query_spec.html#Anchor-Th-63393

값으로 매핑 결과를 알 수 있다.

http://www.crossref.org/guestquery 에서는 하나의 쿼리를 처리하는 DOI LookUp 서비스를 제공하고 있다(〈그림 3〉 참조).

제공하는 인터페이스 중, 쿼리를 업로드 한 후 메일을 통해 DOI LookUp 결과를 통보받는 비동기화 방식도 제공하고 있다. 비동기 방식을 사용 할 경우 POST 방식을 사용하는데, 파

라미터로 넘겨야 하는 것 중 파일명 파라미터(fname)만 빼고 GET방식으로 보낸다(<표 4> 참조). 아래 예는 파일명 파라미터(fname)를 뺀 GET방식의 URL을 보여 주고 있다.

예 : http://doi.crossref.org/servlet/deposit?operation=doMDUpload&login_id=atypon&login_passwd=_atypon_&area=live

2.5 Open Channel Interface (OCI)

CrossRef에 계정이 있어야 사용할 수 있는 서비스로서 Telnet과 흡사한 서비스이다. 특정포트에 연결되어 쿼리를 전송하고 결과 값을 돌려받는 서비스이다. 많은 양의 쿼리를 사용할 경우는 업로드 방법을 사용하는 것을 권고 하고 있다.

<그림 4>의 방법은 실제 접속자가 보는 화면의 내용과 프로그램의 내부 모습을 보여 주고 있다. 실제 쿼리를 전송하는 순서와 응답하는 순서가 같지 않음을 알 수 있다. 이것은 스레드를 통해 복수개의 쿼리를 처리하고 있음을 알 수 있다.

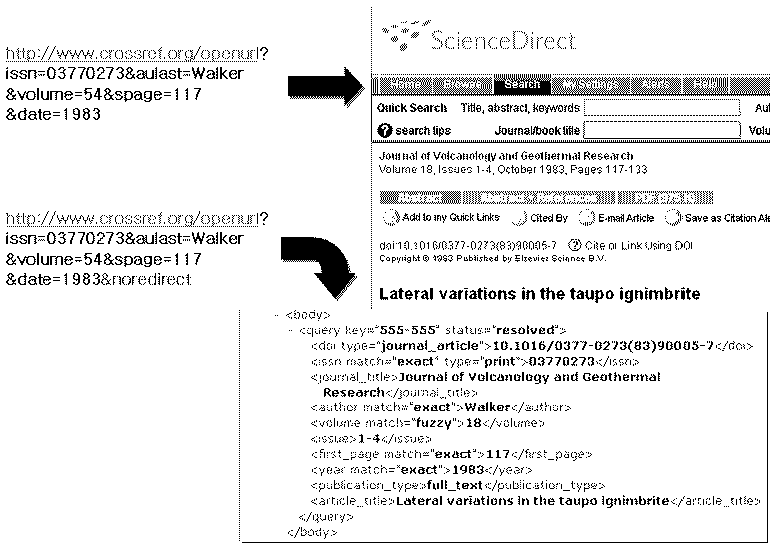
2.6 OpenURL Queries

CrossRef에서는 OpenURL을 받아들여 처리하는 링킹서버를 운영 중에 있다. 링킹서버는 DOI 시스템과의 연동으로 더욱 양질의 서비스를 만들어 낼 수 있게 되었다. CrossRef 링킹서버 사용은 특별한 계정이 없이 가능하나 나쁜 의도로 남용되는 사용을 막고자 지속적인 모니터링이 수행되고 있다. 따라서 모니터링 없이 자유로운 사용을 원할 경우 pid 파라미터에 '이용자아이디:패스워드' 형태의 값을 넘겨야 한다. <그림 5>는 'noredirect' 파라미터를 통해 결과 값을 HTML과 XML로 제공받을 수 있음을 보여준다.

```
1) [root@cr2 root]$ telnet 172.20.1.17 8081
2) Trying 172.20.1.17...
3) Connected to crl.crossref.org (172.20.1.17).
4) Escape character is '^]'.
5) H:USR=crefteat;PWD=******
6) AUTHORISED
7) |curr opin struct Biol|Zwick|||10||2|242|2000||KEY1|
8) 0959440X|Current Opinion in Structural Biology|Zwick|||10|2|242|2000|full_text|KEY1|10.1016/S0959- 440X(00)00075-6
9) |nature|Croll|386||463|1997||KEY2|
10) |cell|Glickman|94|515|1998||KEY3|
11) |trends cell biol|Schwechheimer|11||420|2001||KEY4|
12) |mol cell|Kohler|7||1143|2001||KEY5|
13) 00928674|Cell|GLICKMAN|94|5|515|1998|full_text|KEY3|10.1016/S0092-8674(00)81603-7
14) 00280836,14764679|Nature|Croll|386|6624|463|1997|full_text|KEY2|10.1038/386463a0
15) 09628924|Trends in Cell Biology|Schwechheimer|11|10|420|2001|full_text|KEY4|10.1016/S0962-8924(01)02091-8
16) 10972765|Molecular Cell|KOHLER|7|6|1143|2001|full_text|KEY5|10.1016/S1097-2765(01)00274-X
```

```
1) socket = new Socket(host, port);
2) out = new PrintWriter(socket.getOutputStream(), true);
3) in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
4) out.println("H: usr=<username>;pwd=<pwd>");
5) String line = in.readLine();
6) out.println(qData);
7) line = in.readLine();
```

<그림 4> OCI방식 쿼리



〈그림 5〉 CrossRef Resolver활용 예

3. DOI Lookup API 개발

자바의 입·출력 스트림 클래스를 이용하여 DOI 수집 API를 개발하였으며, 개발 시 사용한 자바 버전은 1.6.0 이다. 작성된 프로그램은 아래와 같이 크게 3 가지이다.

- Piped Query를 이용한 자동 LookUp 프로그램(DOILookUp.java)
- XML Query 자동 작성 및 CrossRef client API연동 프로그램(MakeXMLQuery.java)
- OCI방식을 이용한 자동 LookUp 프로그램(OCILookUp.java, OCILookUpUser.java)

Piped Query형태는 일반적인 OpenAPI 서비스방식에서 많이 사용되는 형태이므로 query spec에 대한 분석 작업이 용이하였고, MakeXMLQuery.java 작성은 CrossRef에서 제공하는 XSD파일의 스펙을 분석하고 개발에 착수하였다. 초기 구현 시에는 XML Query만을 작성하고 단순한 로그를 쌓는 정도의 API 수준이었지만, CrossRef에서 제공하는 SubmissionUploader API와 연동되도록 기능을 보강하였다. 즉, XML Query를 자동생성 후 CrossRef에 자동 전송할 수 있게 하여, 작업자의 수고를 덜도록 구현하였다. OCI 방식을 이용한 프로그램은 소켓통신을 통해 쿼리를 전송하고, 결과 값을 받도록 구현하였다.

3.1 Piped Query를 이용한 자동 LookUp 프로그램

먼저 Piped Query를 이용한 API 구현내용부터 살펴보도록 하겠다. DOILookUp.java 프로그램 실행 시 명령어 라인에 아래와 같이 입력을 한다.

〔실행방법〕
java DOILookUp [대상리스트파일명] [결과파일명] [CrossRef-ID] [CrossRef-Password]

여기에서 '대상리스트 파일명'에는 LookUp을 할 메타데이터가 들어있다. 메타데이터의 제일 첫 번째 칼럼에는 ISSN, 두 번째 칼럼은 저널명, 세 번째 칼럼은 volume, 네 번째 칼럼은 issue, 다섯 번째 칼럼은 시작페이지, 여섯 번째 칼럼은 출판년도, 일곱 번째 칼럼은 소속기관의 시스템에서 사용하는 키 값이다. 일곱 번째 칼럼은 LookUp 결과를 소속기관 시스템에 반영할 때 사용될 수 있다. 예를 들어 DOI를 관리하는 테이블이 있을 경우 해당 식별자를 이용하여 DOI를 구축할 수 있다. 이후 링킹서비스를 제공할 때 기관 시스템에서 사용하는 키 값을 이용해 다른 테이블과의 조인으로 다양한 서비스를 만들어 낼 수 있을 것이다. 아래 〈그림 6〉은 대상파일의 내용을 보여준다. API의 내부 로직은 다음과 같이 구성되어 있다.

- 입력받은 파일들을 위해 스트림클래스 생성
- 파일을 읽기 위해, BufferedReader 클래스객체 생성
- 대상파일에서 라인별로 읽어 아래 프로세스들을 처리
- 넘겨온 파라미터로 기본적인 쿼리의 틀을 생성
- 토근으로 분리하기 위해 객체 String

```
0006-2960   Biochemistry   41  1   1   2002   11826227
0006-2960   Biochemistry   41  1   8   2002   11826228
0006-2960   Biochemistry   41  1   15  2002   11826229
0006-2960   Biochemistry   41  1   21  2002   11826230
0006-2960   Biochemistry   41  1   42  2002   11826232
0006-2960   Biochemistry   41  1   52  2002   11826233
0006-2960   Biochemistry   41  1   61  2002   11826234
0006-2960   Biochemistry   41  1   78  2002   11826236
0006-2960   Biochemistry   41  1   86  2002   11826237
0006-2960   Biochemistry   41  1   94  2002   11826238
0006-2960   Biochemistry   41  1   111 2002   11826239
0006-2960   Biochemistry   41  1   120 2002   11826240
0006-2960   Biochemistry   41  1   131 2002   11826241
0006-2960   Biochemistry   41  1   144 2002   11826242
0006-2960   Biochemistry   41  1   151 2002   11826243
0006-2960   Biochemistry   41  1   162 2002   11826244
0006-2960   Biochemistry   41  1   170 2002   11826245
```

〈그림 6〉 대상리스트파일 내용

Tokenizer 객체생성

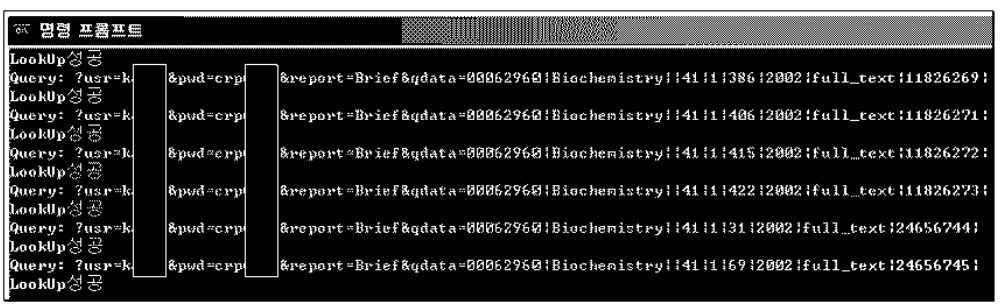
- URL 오브젝트의 openConnection() 메소드를 통한 컨넥션 생성
- 입력스트림 생성 후 Lookup결과 파싱 → 파싱결과 저장

아래 〈그림 7〉은 프로그램이 실행되고 있는 모습을 보여주고 있다. 직사각형으로 볼 수 없는 부분은 프로그램을 사용하는 기관의 CrossRef 계정 정보로 출력될 것이다.

아래 〈그림 8〉은 프로그램이 종료된 후 파라미터로 지정한 결과 파일의 내용을 살펴 본 것이다. 3번째 필드에 '첫 번째 저자의 성'에 대한 정보가 추가되어 있는 것을 알 수 있다. 아홉번째 필드에는 파라미터로 넘긴 로컬기관의 식별자를 CrossRef 시스템이 그대로 보내준 것을 알 수 있다. 마지막 필드에는 매핑된 DOI가 있음을 알 수 있다.

3.2 XML Query 자동 작성 및 CrossRef client API 연동 프로그램

위에서 설명한 파이프 쿼리방식과 달리 다음과 같이 XML로 쿼리를 작성해 CrossRef시



〈그림 7〉 LookUp 실행화면

```
00062960|Biochemistry|Zuiderweg|41|1|1|2002|full_text|11826227|10.1021/bi011870b
00062960|Biochemistry|Wilkinson|41|1|8|2002|full_text|11826228|10.1021/bi0158391
00062960|Biochemistry|Sørensen|41|1|15|2002|full_text|11826229|10.1021/bi011718+
00062960|Biochemistry|Sanchez|41|1|21|2002|full_text|11826230|10.1021/bi010930a
00062960|Biochemistry|Muller|41|1|42|2002|full_text|11826232|10.1021/bi015668k
00062960|Biochemistry|Dhanvantari|41|1|52|2002|full_text|11826233|10.1021/bi015698n
00062960|Biochemistry|Zhang|41|1|61|2002|full_text|11826234|10.1021/bi0117955
00062960|Biochemistry|Mine|41|1|78|2002|full_text|11826236|10.1021/bi011299g
00062960|Biochemistry|Kim|41|1|86|2002|full_text|11826237|10.1021/bi0113824
00062960|Biochemistry|DeRose|41|1|94|2002|full_text|11826238|10.1021/bi0114170
00062960|Biochemistry|Merli|41|1|111|2002|full_text|11826239|10.1021/bi0115386
00062960|Biochemistry|Sato|41|1|120|2002|full_text|11826240|10.1021/bi0117448
00062960|Biochemistry|Takoucheva|41|1|131|2002|full_text|11826241|10.1021/bi011041q
00062960|Biochemistry|Rabilloud|41|1|144|2002|full_text|11826242|10.1021/bi0114776
00062960|Biochemistry|Jarstfer|41|1|151|2002|full_text|11826243|10.1021/bi011649z
```

〈그림 9〉 XML 쿼리

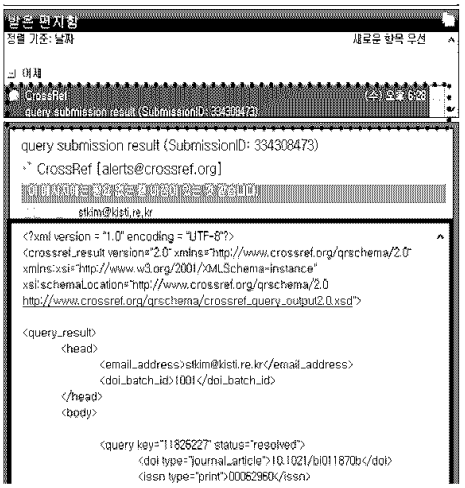
스템의 매핑기능을 호출하는 방식이다. 개발된 XML쿼리 API는 3.1절에서 설명한 API와 실행 방법이 같다. 다만 대상리스트 파일이 매핑된 결과물이 아니고, CrossRef에 전송하기 위한 XML 파일인 점이 다르다. 〈그림 9〉는 프로그램의 산출물인 결과파일의 모습을 보여주고 있다.

[실행방법]

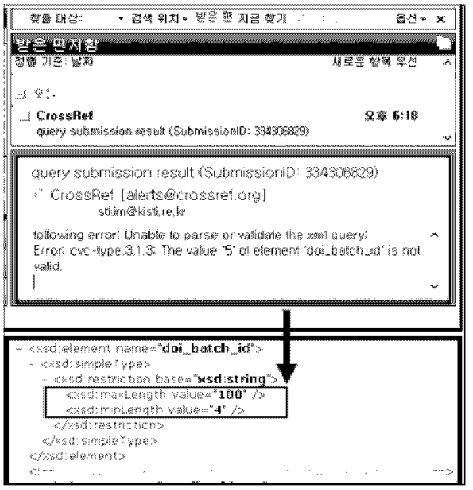
java MakeXMLQuery [대상리스트파일명] [결과파일명] [CrossRef-ID] [CrossRef-Password]

API의 내부 로직은 다음과 같이 구성되어 있다.

- 입력받은 파일들을 위해 스트림클래스 생성
- doi_batch_id.txt 파일로 부터 배치작업 ID 획득
- XML 쿼리 헤더부 작성
- 대상파일에서 라인별로 읽어 아래 프로세스들을 처리
- 넘겨온 파라미터로 기본적인 쿼리틀 생성
- 기관의 시스템에서 사용하는 고유한 key 값으로 쿼리 엘리먼트 key 속성 값 작성
- 토근으로 분리하기 위해 객체 String Tokenizer객체생성
- 결과 파일 기록
- doi_batch_id값 파일에 저장
- CrossRef 업로드 프로그램 실행



〈그림 10〉 XML쿼리 전송결과



〈그림 11〉 유효하지 않은 XML쿼리 전송시 오류 반환

〈그림 10〉은 XML 쿼리내용을 매핑한 후 결과를 이메일로 송부된 결과를 보여주고 있다. 〈doi_batch_id〉 엘리먼트를 통해 전송한 XML 쿼리가 무엇이었는지 알 수 있다.

〈그림 11〉은 doi_batch_id의 값으로 유효하지 않은 아이디값을 전송했을 경우 CrossRef에서 제시한 XML 쿼리의 DTD(Data Type Definition)에 위반하기 때문에 오류 내용을 메일로 반환한 모습을 보여주고 있다. doi_batch_id 값의 길이가 최소 4자이상 100자 이하여야 함을 나타내고 있다.

3.3 OCI 방식을 이용한 자동 LookUp 프로그램

위에서 설명한 프로그램들과 작동방식이 유사하지만 소켓을 이용해 구현하였다는 점이 다르다. CrossRef 시스템에 안내되어 있는 IP주소를 대상으로 할 경우 오류가 발행하여 CrossRef의 호스트 이름(doi.crossref.org)을 사용해 보았더니 정상적으로 작동하였다. 8081포트는 유효한 포트이다. 프로그램 구성은 아래와 같다.

- OCI LookUp.java : 서버와 소켓 연결, 쿼리전송, 모든 쿼리처리 후 소켓을 닫음
- OCI LookUpUser.java : 서버의 쿼리처리 결과 화면출력 및 파일출력 담당, 서버가 접속을 끊었을 경우 후 처리 작업

[실행방법]

java OCI LookUp [대상리스트파일명] [결과파일명] [CrossRef-ID] [CrossRef-Password]

API의 내부 로직은 다음과 같이 구성되어 있다.

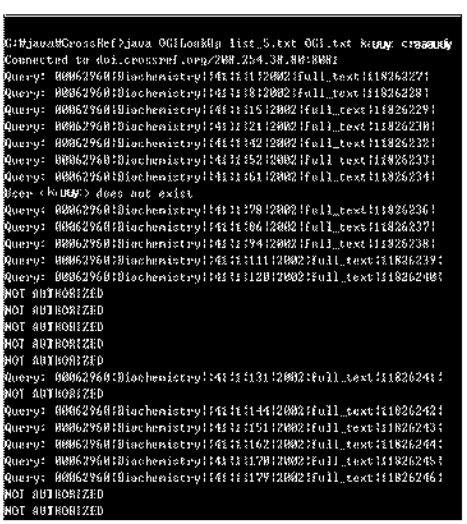
- 입력받은 파일들을 위해 스트림클래스 생성
- 키보드에서 데이터를 읽어내기 위한 Reader 생성
- 화면으로 데이터를 출력하기 위한 Writer 생성
- OCILookUpUser클래스를 이용하여, 네트워크에서 전송된 데이터를 화면에 출력
- 쓰레드를 실행
- 파일을 읽기 위해, BufferedReader 클래스객체 생성
- CrossRef 호출
- 결과 파일 기록

프로그램을 실행시켜 보니 〈그림 12〉와 같이 '사용한 계정이 존재하지 않는다'는 문제가 발생하였다. '파이프 쿼리, XML 쿼리에서 사용하는 계정이 존재하지 않는다'는 리턴 값은 쉽게 납득할 수 있는 결과가 아니었다.

아래 〈그림 13〉은 OCI LookUp 결과의 로그 내용이다. 위 그림과 같은 내용을 확인할 수 있다. 실제 CrossRef 서비스에서 제공하고 있는 쿼리스팩 페이지에 노출된 IP정보 등이 유효하지 않은 것이 확인되었으며, OCI 방식은 데몬(Daemon)의 형태로 실행되면서, 대용량 데이터를 처리하는 프로세스에서 사용하지 말 것을 권고하고 있다. 파이프나 XML 쿼리가 더 유용하게 쓰일 수 있으며, 인증상의 문제이므로 더 이상의 실험은 하지 않았다.

이상으로 개발된 API는 〈표 5〉와 같다.

정보서비스 기관에서 DOI 수집 API를 사용하기 위해서는 대상리스트 파일포맷에 준하



〈그림 12〉 OCI LookUp 결과

〈그림 13〉 OCI LookUp 로그파일 내용

〈표 5〉 개발된 API 입력값과 출력값 비교

API명	입력 값	출력 값
파이프 쿼리를 이용한 자동 LookUp 프로그램 (DOILookUp.java)	대상리스트파일 결과파일명 CrossRef ID CrossRef Password	Lookup결과 텍스트파일
XML 쿼리 자동 작성 및 CrossRef Client API 연동 프로그램(MakeXMLQuery.java)		CrossRef업로드용 XML파일
OCI방식을 이용한 자동 LookUp 프로그램 (OCILookUp.java, OCILookUpUser.java)		Lookup결과 텍스트파일

* 개발된 API는 모두 실행방법이 아래와 같이 동일하다.
java API명 [대상리스트파일명] [결과파일명] [CrossRef-ID] [CrossRef-Password]

는 파일을 생성 후 기관의 CrossRef 아이디와 패스워드를 이용하여 API를 실행하면 된다. 혹은 API의 메인함수의 이름을 간단히 변경하여 기관에서 개발하고 있는 프로그램에서 호출 후 사용하여도 무방하다.

4. 결론 및 향후 계획

본 연구에서는 DOI를 수집할 수 있도록 CrossRef이 제공하는 6가지 유형의 쿼리(XML 방식, 저널 파이프 방식, 컨퍼런스 프로시딩 & 단행본 파이프 방식, HTTP 쿼리방식, OCI방식, OpenURL방식)를 면밀하게 분석하였다. 또한 기관이 보유하고 있는 학술 메타데이터를 활용하여 CrossRef의 DOI를 자동 수집하는 API를 개발하였으며, 그 활용방법을 제시하였다. 개발된 API는 실행방법이 모두 동일하며, 어떤 프로그램에서든지 쉽게 호출하여 사용될 수 있도록 개발하였다. API의 입력 값은 대상리스트파일, 결과파일명, 아이디(CrossRef ID), 패스워드(CrossRef password)로 모두 동일하며, 출력 값은 텍스트와 XML파일의 형태를 취하도록 하였다.

다량의 학술 메타데이터를 보유하고 있는 정보서비스 기관이 DOI를 수집하기 위해 각각의 쿼리 및 프로그램을 개별적으로 분석·개발해야 하는 수고를 덜 수 있게 되었다. 개발된 API는 CrossRef 리졸빙시스템(resolving system)과의 복잡한 인터페이스를 내부에 가지고 있을 뿐 API 사용자는 사용에 대한 부담이 전혀 없도록 설계되었다. 또한 기관이 개별적으로 분석·개발 시 갖게 될 오류를 최소화하였다. 예를 들어, OCI 방식을 이용한 자동 LookUp 프로그램 개발과정에서 CrossRef에서 제공하는 IP 주소정보가 잘못된 것을 발견하게 되어, 이후 OCI 관련 시스템을 개발하는데 오류가 발생하지 않게 되었다.

쿼리 작성 시 사용된 대상리스트 파일에서 특정 필드의 값이 빠져있는 DOI 조회결과인

텍스트와 XML에 자동으로 채워져 반환되는 것을 〈그림 8〉을 통해 알 수 있다. 이것은 반환된 결과 값을 활용해 기 구축된 메타데이터를 재가공할 수 있음을 의미한다. 예를 들어 저널명을 통해 ISSN을 구축할 수 있으며, 반환된 저널명을 활용하여 저널전거(이명) 데이터베이스를 구축하는데 활용될 수 있을 것이다.

향후에는 LookUp으로 획득한 메타데이터를 활용할 수 있도록 분석기에 대한 연구를 진행할 것이다. 예를 들어, 웹 인터페이스로 전송된 XML 파일과 텍스트를 처리하는 분석기와 메일로 전송된 내용을 자동처리하는 분석기 개발이 필요하다. 또한 전송된 학술정보자원 간 연계의 중요성이 높아지고, 자원에 대한 유일한 식별자(DOI, PMID 등)를 수집하기 위한 수요가 증가함에 따라, DOI 식별자 뿐 아니라 NCBI(National Center for Biotechnology Information)와 같은 서비스에서 제공하는 데이터베이스 내의 식별자들을 수집하는 '글로벌 식별자 자동 수집기'에 대한 연구를 진행할 것이다. NCBI에서 제공하는 데이터베이스 중 PubMed 서비스의 경우 CrossRef처럼 LookUp을 위한 쿼리를 제공하고 있지 않기 때문에, 웹 크롤링(crawling) 방식의 수집기를 개발해야 한다. 또한 출력되는 결과 값이 CrossRef와 상이하기 때문에 별도의 분석기에 대한 설계 작업이 뒤따라야 할 것이다. 이렇듯 서비스에 종속적인 수집기 개발에는 많은 어려움이 있을 수 있으나, 연구자들에게 최적의 학술자원 연계서비스를 제공하기 위해서는 반드시 개발되어야 할 것이다.

마지막으로, 개발된 API를 활용하여 학술정보 유통서비스 분야의 개발 담당자가 보다 나은 서비스를 만들어내길 기대한다.

참고문헌

김선태. 2002. 「NDSL(National Digital Science Library) 시스템을 기반으로 한 DOI Lookup시스템 개발에 관한 연구」. 석사학위논문, 전북대학교 정보과학대학원, 정보과학과.

Brand, A. 2004. "Publishers Joining Forces through CrossRef." Serials review, 30(1): 3-9.

CrossRef 시스템 문서. [cited 2007.10.23]. 〈http://doi.crossref.org/doc/userdoc.html〉.

CrossRef OpenURL resolver 설명. [cited 2007.9.21]. 〈http://www.crossref.org/02publishers/openurl_info.html〉.

CrossRef의 OpenURL서비스 설명. [cited 2007.9.20]. 〈http://www.crossref.org/03libraries/16openurl.html〉.

David Sidman, Tom Davidson. 2001. "A Practical Guide to Automating the Digital Supply Chain with the Digital Object Identifier (DOI)." Content

Directions, 17(2).

DOI 시스템 프락시 서버와 쿠키설정. [cited 2007.8.11]. 〈http://www.doi.org/doi_proxy/appropriate_copy.html〉.

DOI Name Information and Guidelines. June 11, 2007 [cited 2007.12.20]. 〈http://www.crossref.org/02publishers/doi-guidelines.pdf〉.

DOI-X Prototype 정보. [cited 2007.8.27]. 〈http://meta.doi.org/〉.

Free DOI LookUp. [cited 2007.10.29]. 〈http://www.crossref.org/guestquery〉.

Herbert Van de Sompel. 1999. "Reference Linking in a Hybrid Library Environment." D-Lib Magazine, 5(10).

Pentz, E. 2006. "CrossRef at the crossroads." Learned Publishing, 19(4): 250-258.

XML query XSD. [cited 2007.10.15]. 〈http://www.crossref.org/qschema/crossref_query_input2.0.xsd〉.

예제 정보. [cited 2007.10.30]. 〈http://doi.crossref.org/doc/samples.zip〉.