J. of Basic Sciences, Cheju Nat'l Univ.
16(1), 73~87, 2003

기초과학연구 제주대학교
16(1), 73~87, 2003

XML 문서 검색과 데이타베이스 검색의 성능 비교

강 미 경 · 박 소 정 · 이 정 훈
제주대학교 자연과학대학 전산통계학과

요 약

인터넷을 통해 전달되는 문서의 개수가 기하급수적으로 증가함에 따라 사용자가 요구하는 문서를 보다 효과적으로 저장 및 검색하는 것이 필수적이다. XML(eXtensible Markup Language)은 사용자가 필요로 하는 XML 정보를 일반 관계 데이터베이스에 XML형태로 저장하고 효율적으로 검색할 수 있게 해준다. 본 논문에서는 XML 성능 평가의 일환으로써 C#프로그램을 이용한 데이터베이스 검색과 XML 문서 검색의 성능을 비교하였다. 데이터베이스 검색을 수행하기 위하여 DataReader와 DataSet을 이용하였고, XML 문서 검색을 수행하기 위해서는 XmlTextReader를 이용하였다. 검색을 위한 데이터 생성은 난수를 이용하여 대량의 학생정보 데이터를 가상으로 만들었다. 성능평가 결과 소량의 데이터 정보에서 검색을 수행하는 데는 XML 문서 검색이 월등하게 빨랐으나 데이터양이 많아질수록 XML 문서 검색 성능은 현격히 저하되어 검색 성능의 한계를 드러내었다. 따라서 대용량의 데이터를 처리할 경우는 데이터베이스 시스템을 이용하여 저장 및 검색을 수행하여야겠지만 1만개 이하의 소량의 레코드를 유지하는 경우에는 XML 문서를 이용하여 검색하는 것이 효율적이다.

1. 서 론

인터넷을 통해 전달되는 문서의 개수가 기하급수적으로 증가함에 따라 사용자가 요구하는 문서를 보다 효과적으로 저장 및 검색하는 것이 필수적이다. 인터넷을 통해 전달되는 문서로는 HTML이나 XML(eXtensible Markup Language) 혹은 SGML(Standard Generalized Markup Language)과 같은 구조화된 문서 등이 있다. 다양한 정보를 저장하고 관리하기 위한 표준화 방법과 이기종 시스템간의 정보 교환과 다양한 검색환경을 제공하기 위해 국제표준(ISO 8879)으로 제정된 SGML은 마크업 언어로써 문서의 논리구조와 내용을 기술할 수 있지만 문법이 복잡하며 인쇄를 위한 표준 언어이다. 그리고 HTML 기반의 웹 문서는 오직 디스플레이 위주로 재가공하여 사용할 수 없으며 의미정보를 표현하지 못해 인터넷상에서 의미 정보를 검색하는데 큰 문제점을 갖고 있다. XML은 HTML과 SGML이 가지고 있는 다양한 기능들과 구조적인 표현 능력, 그리고 사용의 용이성 등의 장점을 가지면서 특히 인터넷상에서 의미정보를 전달하고 질의할 수 있는 막강한 장점을 지닌 언어로 1996년 웹의 문서 표준으로 제안되었고 웹언어 표준화단체인 W3C(The World Wide Web Consortium)이 1998년 2월에 표준버전 1.0으로 인정한 문서표준이다[1].

XML은 본질적으로 다른 언어를 기술하기 위한 언어, 즉 메타언어이다. HTML은 태그의 종류가 한정되어 있는 반면 XML은 다른 사용자가 작성한 태그들의 의미를 파악할 수 있도록, 또는 XML 문서가 태그 정의를 참조할 수 있도록 하는 선언 파일인 DTD(Document Type Definition)를 통해 문서의 내용 또는 데이터에 관련된 독자적인 태그를 사용자가 직접 정의할 수 있으며 혹은 CSS(Cascading Style Sheets) 태그를 다른 사람들이

사용하도록 할 수 있다.[2]

XML은 사용자가 필요로 하는 XML 정보를 일반 관계 데이터베이스에 XML 형태로 저장하고 XML을 이용하여 효율적으로 검색할 수 있게 해준다[3]. 예를 들어 현재와 같은 EDI(Electronic Data Interchange) 전자상거래 환경에서는 서로 다른 데이터 구조(스키마)를 지니는 데이터베이스간의 데이터 교환이 필수적으로 발생하게 되는데 이때 XML은 각기 다른 데이터베이스에 자신의 데이터를 XML 데이터로 변환 저장하고, 저장된 XML 데이터를 검색할 수 있게 해준다.

XML은 SGML의 복잡한 문법이 생략되며 인터넷을 기반으로 하고 있어 인터넷상의 정보를 효과적으로 표현하고 교환할 수 있는 표준이 되고 있다. 또한 정형화되지 않은 데이터들을 저장할 수 있는 표준 방법으로 인식되고 있다. 그리고 다양한 분야의 XML 문서들을 효과적으로 저장하고 검색할 수 있는 XML 문서 저장 검색 시스템이 개발되어 왔다. XML 문서를 쉽게 검색한다면 사용자는 정보를 손쉽게 얻을 수 있을 것이다. 따라서 기존의 데이터베이스를 이용한 검색과 XML을 이용한 검색을 비교 평가해 보고 좀더 나은 XML 문서 검색 시스템을 연구해 볼 필요가 있다[4].

XML 응용 개발을 위한 표준 API(Application Programming Interface)로 이벤트 기반의 SAX(Simple API for XML)와 객체 모델 기반의 DOM(Document Object Model)이 있는데 SAX 방법은 이벤트에 접근한 문서 내에 특정 엘리먼트를 만났을 때 지정된 이벤트를 발생시켜 이를 이용하여 필요한 정보에 접근하는 방법이며 DOM 방법은 XML 문서를 파싱하여 문서의 구조정보와 컨텐츠 모두를 객체로서 메모리에 올려놓고 문서 전체에 대한 구조 정보를 트리에 기반한 객체로 이용 가능하여 XML 문서를 구조적으로 변경하거나 검색하는 작업 등에 적합하나 크기가 큰 XML 문서의 사용은 그에 따른 메모리 사용량이 증가한다는 단점이 있다. DOM을 이용하여 사용자는 문서내의 이벤트 및 문서 구조, 문서 내용, 문서의 보안 레벨 설정 등과 같은 문서의 모든 것에 대해 정의, 조작, 변경, 접근할 수 있다[3].

이 논문에서는 다량의 데이터 생성뿐만 아니라 생성된 데이터를 기반으로 하여 닷넷 프레임워크에 포함된 클래스를 기반으로 MS-SQL 데이터베이스를 이용한 검색과 XML 문서를 이용한 검색 속도를 비교한다. 2절에서는 관련 연구에 대하여 알아보고 3절에서 Testbed 구축 방법, 성능평가 프로그램 및 여러 가지 검색 조건에 대한 성능 평가 결과에 대하여 기술하고 마지막으로 4절에서 결론을 맺는다.

2. 관련 연구

XML이 등장한 이후 XML을 이용해서 데이터를 표현할 수 있게 되고 기존에 데이터베이스를 이용하는 곳에도 XML을 사용할 수 있게 되었다. 데이터베이스와 XML은 서로 다른 장점을 가지고 있다. 데이터베이스는 데이터를 우수하게 관리하는 반면, XML은 데이터를 우수하게 변환하고 표현할 수 있기 때문에 서로 상호 보완하는 작용이 있다. XML과 데이터베이스, 데이터 검색에 관련된 연구를 정리하면 다음과 같다.

저장하부구조로서 객체관계 데이터베이스 시스템(ORDBMS: Object-Relational DataBase Management System)뿐만 아니라 관계형 데이터베이스 시스템(RDBMS)을 이용하여 XML 문서 저장 시스템의 성능을 평가하는 한 연구에서는, 어떠한 XML 문서도 저장할 수 있는 DTD 독립적인 방법이 아니라, 특정 응용프로그램에서 사용될 수 있는 DTD 유무에 따른 검색 시간을 실험하였다. 검색 시간은 질의의 종류를 달리하면서 측정하였으며 질의는 한 테이블 내에서 검색 가능한 단순 질의와 여러 테이블의 조인을 통해서만 결과값을 얻을 수 있는 조인 질의 등 질의의 종류를 다양하게 하였다. 실험결과 질의 요구에 대한 응답 시간이 DTD 독립적인 경우보다 DTD 의존인 경우에 더 작은 것을 알 수 있었다. 이러한 결과는 DTD 의존적인 문서는 DTD에 따라 스키마가 고정되므로, 훨씬 다양한 질의를 수행할 수 있을 뿐만 아니라 빠른 접근이 가능하기

때문이며 똑같은 결과를 얻을 수 있는 질의라도 그 질의의 종류에 따라 검색 시간은 크게 차이를 보일 수 있기 때문이다. 또한 저장하부시스템을 달리하여 관계형 데이터베이스는 오라클, 객체 관계형 데이터베이스는 오디세우스를 이용하여 DTD에 독립적인 XML 문서 저장 시스템 하에서 성능 비교 평가를 하였다. 두 하부 시스템의 저장 시간을 측정한 결과 ORDBMS인 오디세우스에 저장하는 것이 좀더 적은 시간을 요구하고 있는 것을 볼 수 있었다. XML 문서는 Semistructured Document라서 구조적인 정보를 포함하고 있으며 실제 RDBMS의 경우 이러한 구조를 표현하기가 쉽지 않으며 반대로 ORDBMS는 객체적 특성을 이용하여 XML 같은 구조적 질의처리를 쉽게 할 수 있다. 마지막으로 DBMS를 이용하는 Edge 기법을 이용하는 타 시스템과 비교한 결과 성능이 더 뛰어나다는 결과를 얻었다[4].

대부분의 자료 검색 시스템의 경우 텍스트 위주의 검색 방법이지만 SGML과 XML은 DTD로 문서의 논리 구조를 표현할 수 있다. 즉 DTD를 통해 데이터베이스를 구성함으로써 구조 검색이 가능하게 되었다. TheoSEARCH, XSearcher, STEER 등이 이에 해당되는 시스템들이며 DTD에서 문서구조를 나타내는 엘리먼트로 구조검색을 지원하고 있다. 또한 각 검색 시스템들은 구조검색과 내용검색을 모두 지원하고 있다. XSL(eXtensible Stylesheet Language)를 이용한 XML 문서 검색에 관한 연구 중 XSL의 Transformation 기능을 이용한 XML 문서의 구조검색과 내용검색을 수행한 논문에서는 XSL의 패턴을 이용해서 문서 구조를 표현할 수 있는 트리를 생성하며 사용자에게 편리한 인터페이스를 제공함으로써 질의를 쉽게 하도록 했다. XSL의 Transformation 기능에서 DTD의 엘리먼트를 나타내는 Pattern을 기반으로 XML 문서 검색 시스템을 구성하였다. Pattern의 특성상 문서의 구조 검색이 가능하게 되어 기존의 검색보다 정확하고 다양한 조건으로 검색할 수 있다[6]. DTD가 없는 Well-formed XML 문서일 경우 DTD를 새로 만들어 데이터베이스에 저장하는 것보다 XSL의 패턴을 이용해 데이터베이스를 구성하는 것이 더 적합하다. 검색 시스템에 일반적인 내용 검색 방법도 포함하여 다양한 검색 기능을 가지게 했다[5].

문서구조화와 정보검색을 위한 연구로 CD-ROM이나 하드디스크와 같은 보조기억장치에 존재하는 개인의 정보를 데이터베이스나 검색엔진을 이용하지 않고 정보를 효율적으로 처리할 수 있는 XML 기반의 로컬 검색 시스템을 설계하고 구현한 사례가 있다. 이를 위해서 정보를 효율적으로 관리하고 저장할 수 있는 인덱스를 XML 문서로 작성한다. 이러한 XML 문서의 태그를 이용해서 사용자가 원하는 문서 파일을 스크립트(Script) 언어를 사용해서 검색한다. 또한 정보의 양이 증가할 경우 검색 시간을 단축하기 위해서 XML 문서를 분리해서 저장하였다[6].

XML 질의어 처리 및 XML의 데이터 모델과 데이터베이스에 저장하기 위한 방안들에는 여러 가지가 있다. XML 데이터를 데이터베이스에 저장하여 질의하기 위한 방법은 많이 연구되어 왔다. 이러한 방법들은 XML을 위한 새로운 데이터베이스 시스템을 만들거나 기존의 데이터베이스 시스템에 저장하는 것들이다. 기존의 데이터베이스 시스템에 저장할 때 데이터 모델의 이질성으로 인하여 이를 저장하기 위한 다양한 방법이 제시되었으며 이는 기존의 데이터베이스 시스템의 특성을 잘 이용하여 XML 질의 수행을 빠르게 하도록 고안되었다. XML 데이터는 그래프 기반의 비정형 데이터 모델에 매핑되어 질의가 수행될 수 있기 때문에 비정형 질의 언어를 XML 질의에 이용할 수 있다. 이러한 질의 언어로는 UnQL, Lore 등이 있다. 이 밖에도 XML에 특화된 질의 언어로는 XML-QL과 XQL(XML Query Language)이 제안되었다[7].

XML 저장 기법은 크게 세가지로 구분할 수 있는데 파일 시스템, 데이터베이스 시스템, 저장 관리기가 그것이다. 이 중 관계 데이터베이스 시스템을 이용한 방법은 XML 문서의 저장을 위한 구조 정보를 관계 데이터베이스에 테이블 형태로 구성하고 저장한다. 관계 데이터베이스를 이용하는 방법은 RDBMS의 우수한 성능을 충분히 활용할 수 있고, 기존 응용 시스템의 데이터를 함께 사용할 수 있는 장점을 가진다. 그러나 검색 시 다수의 테이블에 대한 고비용의 조인 연산을 수행해야 하며 검색 결과를 추출하는 데 많은 시간과 노력이 필요하고, 집합값을 지원하지 않는 등의 단점을 가지고 있다[4].

XML 문서의 저장은 다음과 같은 과정을 거치게 된다. 일단 XML 문서는 파서에 의하여 파싱되며, 이는 DOM 트리의 형태로 결과가 얻어진다. 파싱의 결과로 얻어진 DOM 트리는 저장관리기에 의하여 테이블에 매핑되어 저장된다[1]. 저장 관리기는 문서의 루트 노드부터 DFS(Depth First Search) 방식으로 노드를 순회하며 각 노드의 정보는 해당 테이블에 저장되는데, 노드의 타입이 엘리먼트이면 Element 테이블에 엘리먼트 이름과 부모 엘리먼트의 OID(Object IDentifier), 경로 등의 정보가 저장되며 애트리뷰트가 있는지를 체크하여 애트리뷰트는 Attribute 테이블에 애트리뷰트가 나열된 순서와 함께 애트리뷰트의 이름과 그 값을 저장한다. 이러한 과정을 자식노드를 재귀적으로 순회하면서 모든 엘리먼트를 데이터베이스에 저장한다. XML 문서에 대한 검색으로는 문서가 포함하고 있는 내용을 검색하는 내용 기반 검색과 XML 데이터가 가지고 있는 구조 정보인 순서나 부모, 자식, 조상, 자손, 형제와 같은 계층 정보에 대한 검색인 구조 기반 검색이 필요하며, 내용 기반 검색과 구조 기반 검색을 혼합한 내용+구조 검색 기능이 요구된다[4].

닷넷 프레임워크에서 데이터베이스를 사용하기 위한 기술을 'ADO.NET'이라고 한다. 이것은 ODBC(Open DataBase Connectivity), OLEDB(Object Linking and Embedding DataBase), JDBC(Java DataBase Connectivity) 등과 같은 기존의 기술과 마찬가지로 데이터베이스에 접근해서 필요한 작업을 수행할 수 있도록 구성되었다. 또한 레코드 정보를 XML로 표현해서 저장 및 이용할 수 있도록 필요한 것들을 지원하는 특성을 가지고 있다. XML을 이용하면 데이터베이스와의 연결을 유지하지 않아도 되는 장점을 갖게 된다. ADO.NET에 있는 네임스페이스의 클래스를 통해서 원하는 데이터베이스에 접근한 후에 자료를 검색하거나 수정 및 삭제하는 등의 작업을 진행할 수 있다. 우선 데이터베이스에 연결한 후에 그 결과로 이용되는 데이터의 형태에 따라서 Command나 DataAdapter를 이용한다. Command를 이용하는 경우에는 DataReader를 통해서 데이터를 사용할 수 있고, DataAdapter를 이용하는 경우에는 DataSet을 통해서 XML로 표현된 데이터를 사용할 수 있다.

XML 문서를 읽는다는 것은 사람이 읽는 것이 아니라 XML 관련 응용 프로그램이 읽는다는 것을 의미한다. XML 문서의 구조는 사람뿐만 아니라 응용 프로그램의 입장에서도 쉽게 파악할 수 있도록 구성되어 있다. 닷넷 프레임워크에 포함된 클래스를 이용해서 XML 문서를 읽는 방법 중 XmlTextReader 클래스는 유효성을 검사하지 않는 반면 빠르게 실행된다. XmlTextReader 클래스의 Read() 함수가 XML 문서의 선언문부터 마지막 엘리먼트까지 읽으면서 다음에 읽을 내용이 있으면 True 값을 리턴하고 없으면 False 값을 리턴한다. 이렇게 분석한 정보는 XmlTextReader 객체에 보관되고 프로퍼티를 통해서 분석한 내용은 응용프로그램이 적절하게 사용하게 된다. 그 객체를 읽어들여 XPathDocument와 XPathNavigator를 이용하면 원하는 정보의 위치를 탐색할 수 있다.

3. 성능 평가

이 절에서는 데이터 생성 Testbed를 구축하는 방법과 닷넷 프레임워크에 포함된 클래스를 기반으로 MS-SQL 데이터베이스를 이용한 검색과 XML 문서를 이용한 검색 속도를 비교하는 프로그램 및 여러 가지 검색 조건과 결과에 대하여 정리한다.

3.1 데이터 생성을 위한 Testbed 구축 방법

생성필드는 ID, 학번, 성명(한글), 학과코드, 학과(학부)명, 입학년도, 성별, 주민등록번호 8가지이며 최소 500에서 최대 5,000,000(오백만)개의 데이터까지 데이터 개수를 다양하게 변화시켜가며 생성하였으며, 데이터

가 제대로 생성되었는지를 확인하기 위하여 Microsoft Excel/Access에서 필터와 함수 등을 이용하여 중복여부와 정상출력 여부를 살펴본 결과 이상이 없었다. 5만개의 데이터를 생성하는데 걸린 시간을 clock함수를 이용하여 체크한 결과 0.41초 정도 걸렸다.

표 1. 데이터 생성 필드

①ID	⑧학번	②성명	③학과코드	④학과명	⑤입학년도	⑥성별	⑦주민등록번호
일련번호	10자리	3또는 4자 한글	두자리	공백이 없는 문자열	4자리	1:남자 2:여자	999999-9999999

① ID : 생성된 순서로 레코드의 개수만큼 자동 증가

② 성명(한글) : 대한민국 성씨 154개(2자 포함)가 저장된 파일과, 이름으로 쓸 수 있는 글자 431개가 저장된 파일을 읽어와서 배열에 저장하여, 성(한 글자 또는 두 글자)과 이름 두자를 랜덤하게 추출해내어 결합한다. 즉 3자리나 4자리의 성명이 나올 수 있다.

③④ 학과코드, 학과(학부)명 : 제주대학교 학과(학부) 55개를 가나다 순으로 2자리의 코드(01~55)를 매긴 파일을 읽어와서 배열에 저장한 후, 랜덤하게 추출해내어 학과 코드와 해당 학과명을 출력한다.

⑤ 입학년도 : 1980~2003년 사이에서 가능한 24개의 입학년도를 랜덤하게 추출해낸다.

⑥ 성별 : 1과 2중 하나를 생성해낸다.

⑦ 주민등록번호 : 입학년도에서 19를 빼어 생년을 계산하고, 1과 12 사이의 난수를 발생시켜 월을 구하고, 생성시킨 년(윤년인지 아닌지), 월에 따라 최대일수를 계산하여 난수를 발생시킨다. 주민등록번호 뒷부분 중 첫 번째 코드는 발생시킨 성별 난수를 가져오고, 세자리 코드와 두자리 코드를 난수로 발생시키고 난 뒤, 마지막 체크 코드를 알고리즘에 따라 규칙에 맞는 값을 생성해낸다.

⑧ 학번 : 앞에서 생성한 입학년도 4자리와 학과코드 2자리를 결합한다. 마지막 4자리 학과내에서의 번호는 입학년도와 학과코드별로 생성시킨 데이터의 순으로 일련번호를 부여하였다. 따라서 동일 입학년도, 동일 학과 내에서는 9999명까지는 학번이 중복될 수 없다.

3.2 생성된 데이터를 MS-SQL 데이터베이스 레코드로 변환시키는 방법

Access/Excel/Text 파일 등을 MS-SQL 데이터베이스로 변환시키기 위해서는 먼저 Microsoft SQL Server의 엔터프라이즈 관리자를 실행시킨 후에 DTS(Data Transformation Services) Wizzard(데이타 가져오기/내보내기)를 수행한다. 다음으로 데이타 원본 선택란에서 변환을 원하는 소스타입(Text, Micosoft Access, Microsoft Excel 등)과 변환할 파일을 선택한 후 Microsoft SQL Server의 데이터베이스를 대상으로 선택한다. 실행 시기를 결정한 후 변환을 실행하면 진행률과 현재 상태가 표시되며 변환 완료 후에는 해당 데이터베이스 서버에 접속하여 정상적으로 테이블이 생성되었는지 확인해 볼 수 있다.

3.3 MS-SQL 데이터베이스 데이터를 XML 문서로 변환시키는 방법

DataSet에 보관된 정보는 XML 문서 형태로 쉽게 출력할 수 있다. 그리고 데이터베이스의 데이터를 DataSet으로 변환하고 간단하게 XML 문서로 출력할 수 있다. 또한 데이터베이스에 있는 테이블의 구조에 따라서 XSD(XML Schema Definition)도 출력할 수 있다. 다음은 데이터베이스에 있는 데이터를 DataAdapter를 이용해서 DataSet으로 생성한 후에 DataSet의 내용을 XML 파일로 출력하는 소스이다.

```
string connStr = "server=localhost; database=xml; user id=sa; pwd=password";
SqlConnection con = new SqlConnection(connStr);
con.Open();
SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM db_table_name",con);
adapter.TableMappings.Add("Table", "학생정보");
DataSet ds = new DataSet();
adapter.Fill(ds);
ds.WriteXmlSchema("ran_out_50000.xsd");
ds.WriteXml("ran_out_50000.xml");
```

소스코드 1. 데이터베이스의 자료를 DataSet으로 생성한 후 XML 파일로 출력하는 소스

3.4 MS-SQL 데이터베이스를 이용한 검색과 XML 문서를 이용한 검색 비교

3.4.1 MS-SQL 데이터베이스를 이용한 데이터 검색 방법

ADO.NET에서 지원하는 DataReader를 이용하여 데이터베이스의 자료를 검색하는 프로그램은 데이터베이스에 연결해서 자료를 검색할 수 있도록 쿼리를 실행하고 검색된 자료를 읽어서 화면에 출력하도록 구성된다. 사용하는 데이터베이스 서버는 MS-SQL Server이며 작성된 프로그램에서 수행하는 작업은 소스코드 2와 같이 총 7단계를 거치게 된다.

```
(1)  string connStr = "server=localhost; database=xml; user id=sa; pwd=password";
     SqlConnection con = new SqlConnection(connStr);
     /* 데이터베이스에 접속하기 위한 문자열을 구성,
        서버에 접속하기 위해서 SqlConnection 객체 생성 */
(2)  con.Open();
     /* Connection 객체에 Open() 메소드를 호출해서 SQL Server에 연결 */
(3)  string query="SELECT * FROM out1000000 where 학과명='전산통계학과' and 성별='2'";
     /* 쿼리 구성 */
(4)  SqlCommand cmd = new SqlCommand(query);
     /* 쿼리를 실행하기 위해 Command 객체를 생성 */
(5)  cmd.Connection = con;
     /* Command 객체에 쿼리를 실행할 때 필요한 Connection 객체를 설정 */
(6)  SqlDataReader reader = cmd.ExecuteReader();
     /* 쿼리를 실행하기 위해서 Command 객체의 ExecuteReader() 메소드를 호출 */
(7)  while(reader.Read()){ Console.WriteLine("..."); ... }
     /* DataReader 객체에서 Read() 메소드를 호출해서 검색된 레코드에 접근 */
```

소스코드 2. DataReader를 이용해서 데이터베이스의 자료를 검색하는 소스

검색 속도를 체크하기 위해서는 데이터베이스에 연결해서 검색한 후 출력하는 총 시간 뿐만 아니라 각 단계별로 시간을 체크하여 어느 단계에서 어느 정도의 시간이 걸리는지를 알아보았으며 TimeCheck 함수를 작성하여 각 단계의 start time과 stop time을 DateTime.Now를 이용하여 체크하였다.

DataSet을 이용하면 데이터베이스의 데이터를 보관할 수 있고 필요할 때마다 데이터베이스에 접속하지 않으면서 DataSet에서 원하는 데이터를 이용할 수 있다. DataSet은 말 그대로 데이터의 집합으로 데이터베이스의 자료를 읽어서 객체 형태로 보관한다. 따라서 데이터베이스와 연결이 종료되어도 보관되어 있는 값을 이용

할 수 있다. 즉 데이터베이스 자료의 복사본을 객체 형태로 보관하고 있는 것이다. DataSet을 이용해서 데이터베이스의 자료를 검색하는 총 8단계는 다음과 같다.

```
①   string connStr = "server=localhost; database=xml; user id=sa; pwd=password";
    SqlConnection con = new SqlConnection(connStr);
②   con.Open();
③   SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM out1000000 where 학과
    명='전산통계학과' and 성별='2'",con);
    /* 데이터를 검색하기 위한 쿼리를 문자열로 표현하고 DataAdapter 객체 생성 */
④   adapter.TableMappings.Add("Table", "학생정보");
    /* DataAdapter에 '학생정보'라는 이름을 테이블명으로 매핑 */
⑤   DataSet ds = new DataSet();
    /* DataSet 객체 생성 */
⑥   adapter.Fill(ds);
    /* DataAdapter 객체의 Fill() 메소드를 호출해서 DataSet에 검색된 데이터 구성 */
⑦   DataTable customerTable = ds.Tables["학생정보"];
    /* DataSet 객체에서 '학생정보' 테이블 정보에 접근함.
       DataSet에서는 테이블 정보를 DataTable 객체로 표현함 */
⑧   foreach (DataRow row in customerTable.Rows){ Console.WriteLine("..."); ... }
    /* DataTable의 레코드 정보는 DataRow 객체로 표현하는데, DataTable에 있는 모든
       DataRow 객체를 foreach 문을 이용해서 참조함 */
```

소스코드 3. DataSet을 이용해서 데이터베이스의 자료를 검색하는 소스

결과를 살펴보면 DataReader와 DataSet에 큰 차이가 없다. DataSet은 데이터베이스에서 자료를 가져온 후에 보관하고 있는 상태에서 사용한다. 그래서 같은 데이터를 여러 번 검색해서 사용할 경우 데이터베이스에 다시 접속하지 않아도 된다. 반면 DataReader는 자료를 검색하고 있는 동안 데이터베이스와 연결되어야 한다. 또한 DataSet을 이용하면 보관중인 데이터를 좀 더 쉽게 XML 문서로 변환할 수 있다. 그러나 검색된 데이터양에 비례하여 DataSet을 구성하는데 시간이 소요되므로 대량 정보에서의 검색 수행에 있어서는 DataReader가 유용하다.

3.4.2 XML 문서를 이용한 데이터 검색 방법

XmlTextReader 클래스를 이용해서 XML 문서를 분석하는 각 단계는 다음과 같다.

```
①   XmlTextReader reader = new XmlTextReader("out300000.xml");
    /* XmlTextReader 클래스를 이용해서 객체 생성, 분석할 파일명 전달 */
②   System.Xml.XPath.XPathDocument doc = new XPathDocument(reader);
    /* XPathDocument 객체 생성 */
③   System.Xml.XPath.XPathNavigator nav = doc.CreateNavigator();
    /* XPathNavigator 객체 생성 */
④   nav.MoveToRoot(); nav.MoveToFirstChild(); nav.MoveToFirstChild();
    /* <NewDataSet> 엘리먼트의 첫 번째 <학생정보> 엘리먼트로 이동 */
⑤   do{
            nav.MoveToFirstChild();          /* 하위 엘리먼트로 이동 */
            do {
                    if(nav.Name=="학과명" && nav.Value=="전산통계학과"){
```

```
                                nav.MoveToNext();
                                nav.MoveToNext();
                                if(nav.Value=="2") {
                                        nav.MoveToParent();
                                        nav.MoveToFirstChild();
                                        Console.WriteLine("ID : {0}", nav.Value);
                                        nav.MoveToNext(); ... 중략(나머지 엘리먼트검색) ...
                                }
                        }
                } while(nav.MoveToNext());      /* 전산통계학과 여학생의 데이터 접근 */
                nav.MoveToParent();
        } while(nav.MoveToNext());              /* 계속해서 다음 <학생정보> 엘리먼트로 이동 */
```

소스코드 4. XmlTextReader 클래스를 이용해서 XML 문서를 분석하는 소스

3.4.3 MS-SQL 데이터베이스 검색과 XML 문서 검색 속도 비교

표 2. 실험환경

CPU	Pentium IV 1.7GHz
RAM	512MB
HDD	5G 여유공간
OS	Windows 2000 Professional
데이터베이스	MS-SQL Server
Program	데이터 생성(Visual C++ 6.0), 검색속도 측정(C#)
기본 검색 조건	전산통계학과 여학생

기본 검색 조건으로 주어진 전산통계학과 여학생의 비율은 전체 데이터에 대하여 약 0.91%를 차지한다. 다음과 같이 전체 레코드의 개수를 다양하게 증가시키면서 DataReader, DataSet, XmlTextReader를 이용한 방법에 따른 검색 속도를 체크하여 초단위로 정리하면 다음과 같다.

표 3. 검색 속도 체크 결과 (단위:초)

레코드 수	500	10.000	20.000	30.000	40.000	50.000	60.000	70.000	100.000	200.000
DataReader	0.407	0.453	0.64	0.719	1.015	1.422	2.078	3.578	4.718	8.75
DataSet	0.531	0.625	0.718	0.797	1.282	2.844	3.016	3.484	3.938	9.203
XmlTextReader	0.047	0.609	1.578	2.516	4.062	5.203	7.454	8.609	13.719	36.828

레코드 수가 아주 극소인 경우에는 XmlTextReader가 월등하게 빠른 속도로 검색할 수 있지만, 레코드 수가 10.000개 이하인 경우 데이터베이스를 이용한 방법과 XML 문서를 이용하는 방법이 별 차이가 없음을 그림 1을 통하여 알 수 있다. 그러나 레코드의 개수가 10.000개를 기준으로 그 이상이 되면 XmlTextReader가 검색하는데 걸리는 시간이 점점 데이터베이스를 이용하는 경우와 차이가 많이 나게 되며 DataReader와 DataSet 사이에는 별 차이가 없는 것을 볼 수 있다.

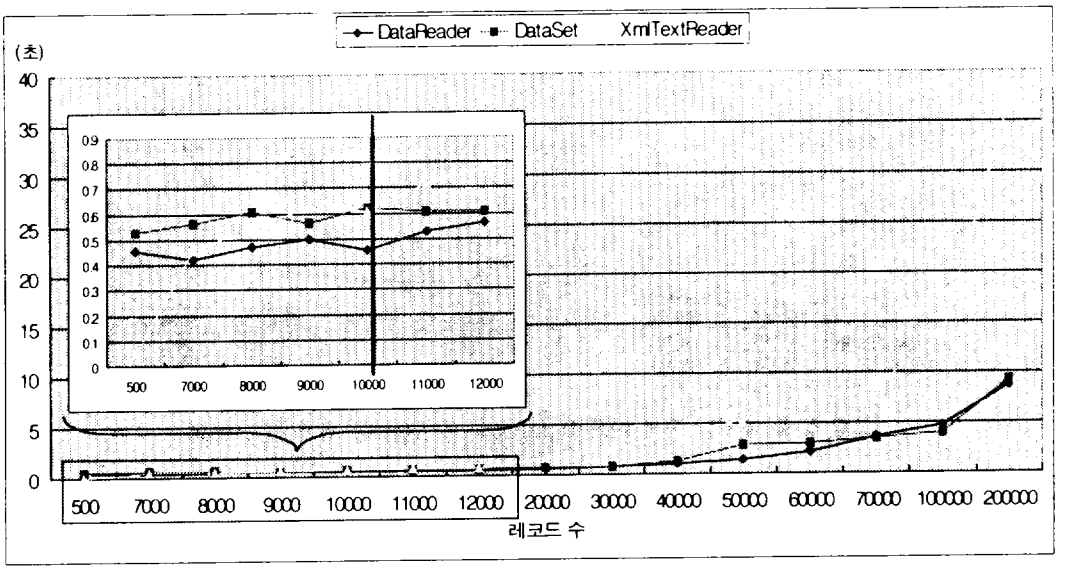


그림 1. 총 레코드 수에 따른 DataReader. DataSet. XmlTextReader 검색 속도 비교

검색된 레코드의 수에 따른 검색 속도비교 결과는 그림 2와 같다.

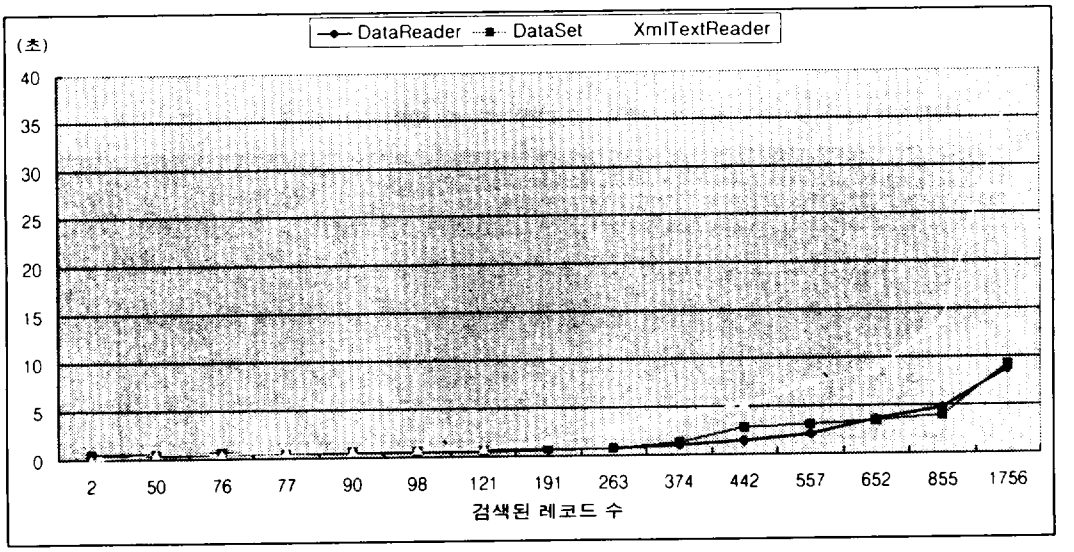
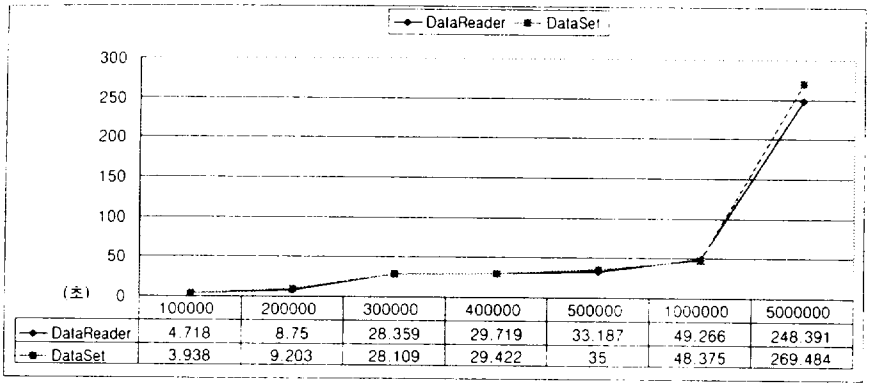


그림 2. 검색된 레코드 수에 따른 DataReader. DataSet. XmlTextReader 검색 속도 비교

데이터의 양이 더 많은 경우에도 DataReader와 DataSet 사이에 검색 속도 차이가 없는지를 알아보기 위하여 레코드의 개수를 1만. 2만. 3만. 4만. 5만. 100만. 500만개로 변화시켰다. 검색 속도 체크 결과는 그림 3에서와 같다. 즉 100만개의 레코드의 경우까지도 거의 유사하다는 것을 알 수 있으며 레코드의 개수가 500만이었을 때는 DataSet의 경우가 8.49%정도 검색시간이 더 드는 결과를 볼 수 있었다.



	100000	200000	300000	400000	500000	1000000	5000000
DataReader	4.718	8.75	28.359	29.719	33.187	49.266	248.391
DataSet	3.938	9.203	28.109	29.422	35	48.375	269.484

그림 3. 총 레코드 수에 따른 DataReader와 DataSet 검색 속도 비교

다음으로는 DataReader. DataSet. XmlTextReader에 따른 각 단계 수행에 걸리는 시간을 체크하여 총 검색 시간에 영향을 미치는 부분은 어느 단계인지 알아보았다.

첫 번째로 DataReader의 경우 소스코드 2의 총 7단계 중 2 Conn_open. 6 exe_reader 7 read&print 단계가 대부분을 차지함을 알 수 있다. 그리고 레코드의 수가 커짐에 따라 검색되는 레코드의 수가 약 0.91%비율에 맞추어 증가하며 그에 따라 DataReader 객체에서 Read() 메소드를 호출해서 검색된 레코드에 접근하는 시간이 늘어나게 되지만. SQL 서버에 연결하는 시간이나 쿼리를 실행하기 위해서 Command 객체의 ExecuteReader() 메소드를 호출하는 시간은 거의 비슷하며 나머지 시간들은 그래프에서 확인할 수 없을 정도로 미미한 시간이었다.

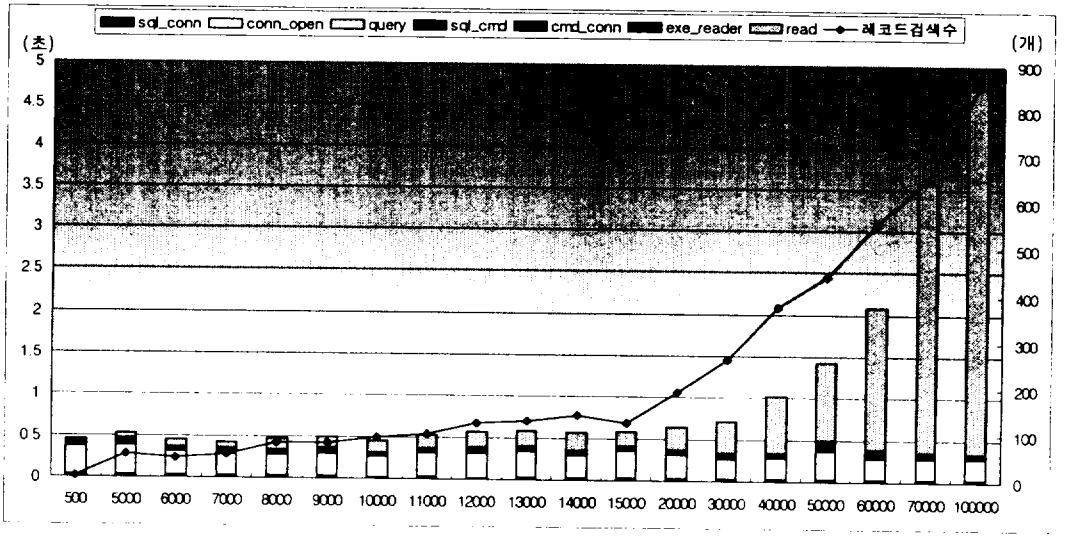


그림 4. DataReader를 이용한 검색의 단계별 소요 시간 및 검색된 레코드수

두 번째로 DataSet의 경우 소스코드 3의 총 8단계 중 2 Conn_open. 6 fill. 8 read&print 단계가 대부분을 차지함을 알 수 있다. 즉 SQL 서버에 연결하는 시간과 DataAdapter 객체의 Fill() 메소드를 호출해서

DataSet에 검색된 데이터를 구성하는 시간. 그리고 DataTable에 있는 모든 DataRow 객체를 foreach 문을 이용해서 참조하는 시간임을 보인다.

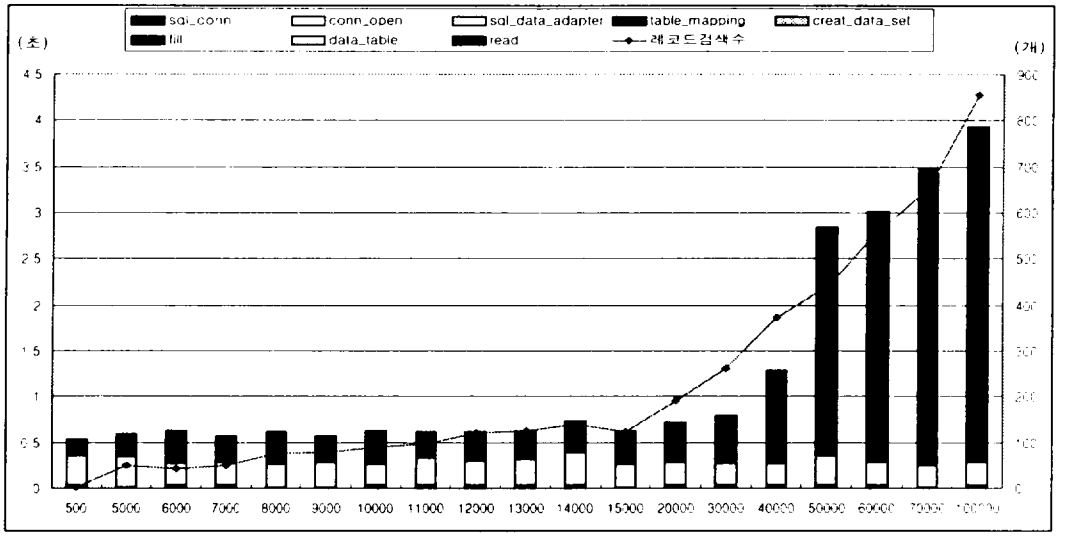


그림 5. DataSet을 이용한 검색의 단계별 소요 시간 및 검색된 레코드수

세 번째로 XmlTextReader의 경우 소스코드 4의 총 5단계 중 ②create__XpathDoc__obj와 ⑤navigation 단계가 대부분을 차지함을 알 수 있는데 이 중에서도 레코드의 수가 커짐에 따라 XPathDocument 객체를 생성하는 시간과 트리구조의 각 엘리먼트를 검색하며 조건에 맞는 레코드를 검색하는데 걸리는 시간이 거의 비례하여 증가함을 볼 수 있다.

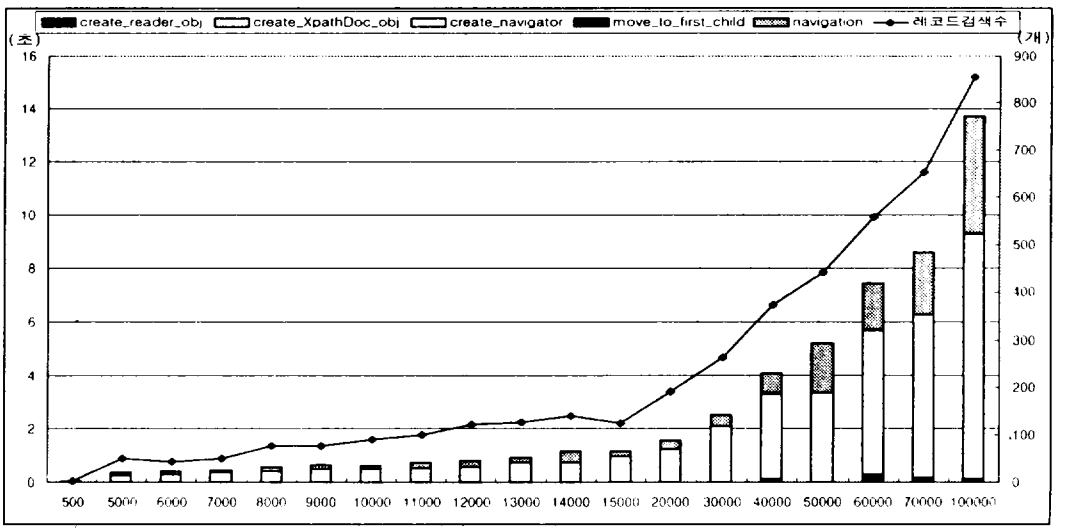
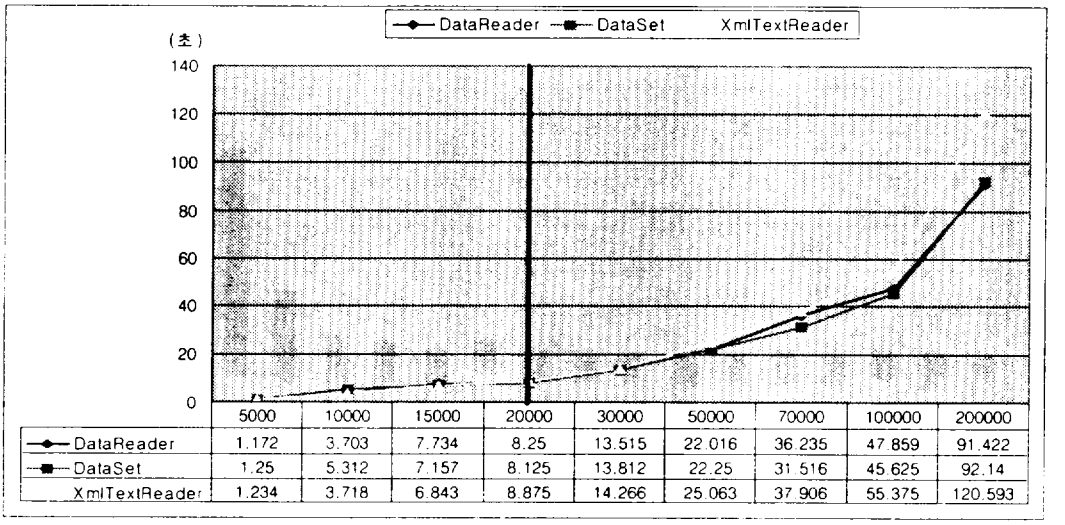


그림 6. XmlTextReader를 이용한 검색의 단계별 소요 시간 및 검색된 레코드수

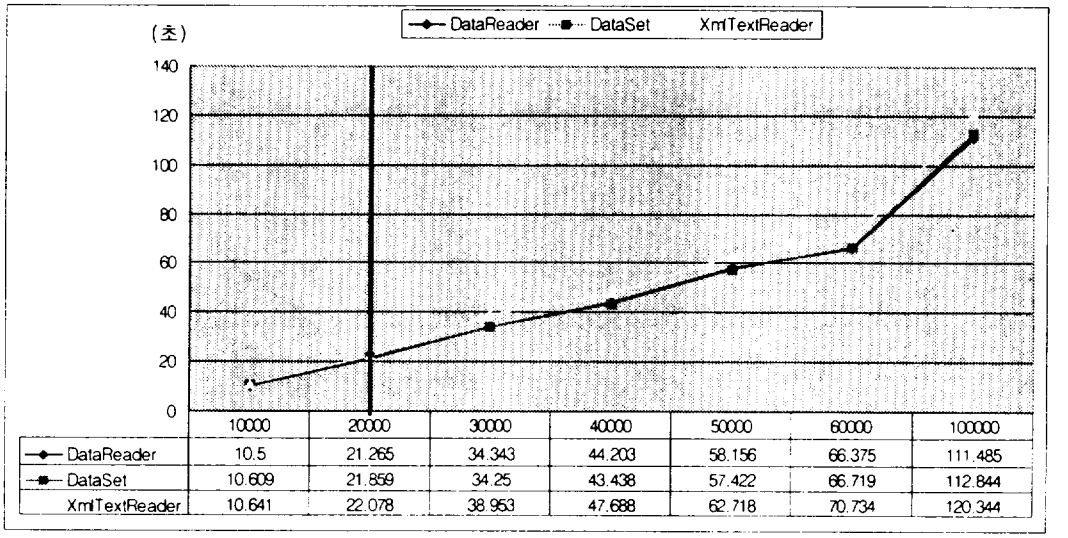
전체 레코드 수에 대한 검색된 레코드 수의 비를 변경하였을 때의 검색 성능 변화 추세에 대하여 측정한 결과는 다음과 같다. 위에서는 검색의 비율이 0.91%일 때 전체 레코드 수에 따른 검색시간 및 추이를 조사하였다. 그림 7은 검색 조건을 입학년도가 1984년과 1994년인 경우로 설정하여 검색비를 8.33%로 고정시켜 전체 레코드 수에 따른 검색시간 변화 추이를 나타낸 것이다.



	5000	10000	15000	20000	30000	50000	70000	100000	200000
DataReader	1.172	3.703	7.734	8.25	13.515	22.016	36.235	47.859	91.422
DataSet	1.25	5.312	7.157	8.125	13.812	22.25	31.516	45.625	92.14
XmlTextReader	1.234	3.718	6.843	8.875	14.266	25.063	37.906	55.375	120.593

그림 7. 검색 비율이 0.91%일 경우 전체 레코드 수에 따른 검색시간 및 추이

그림 8은 검색 조건을 입학년도가 1994년부터 1998년까지로 설정하여 검색비를 20.83%로 고정시켜 전체 레코드 수에 따른 검색시간 변화 추이를 나타낸 것이다.



	10000	20000	30000	40000	50000	60000	100000
DataReader	10.5	21.265	34.343	44.203	58.156	66.375	111.485
DataSet	10.609	21.859	34.25	43.438	57.422	66.719	112.844
XmlTextReader	10.641	22.078	38.953	47.688	62.718	70.734	120.344

그림 8. 검색 비율이 20.83%일 경우 전체 레코드 수에 따른 검색시간 및 추이

결과적으로 전체 레코드 수에 대한 검색 레코드의 개수의 비율을 0.91%, 8.33%, 20.83%로 변화시켰을 때의 검색 시간을 하나의 그림에서 비교하면 다음과 같다. 그림 9에서 볼 수 있듯이 검색비가 증가할수록 검색 시간 차이가 확연하게 드러나 세 개의 그룹을 이루고, 또한 각각에 대하여 DataReader와 DataSet은 거의 유사한 검색 시간을 보이지만 전체 레코드 개수가 증가할수록 XmlTextReader 검색시간이 15,000개 전후로 데이터베이스 검색에 비하여 증가폭이 커진다. 그리고 전체 레코드 수가 200,000개 정도가 되면 검색비가 0.91%일 때는 데이터베이스 검색에 비하여 XML 문서 검색시간이 4배 정도나 소요되며 검색비가 8.33%일때는 1.3배 정도가 소요됨을 알 수 있다. 뿐만 아니라 검색비가 증가할수록 데이터베이스 검색과 XML 문서 검색시간의 차이는 줄어드는 결과를 확인할 수 있었다.

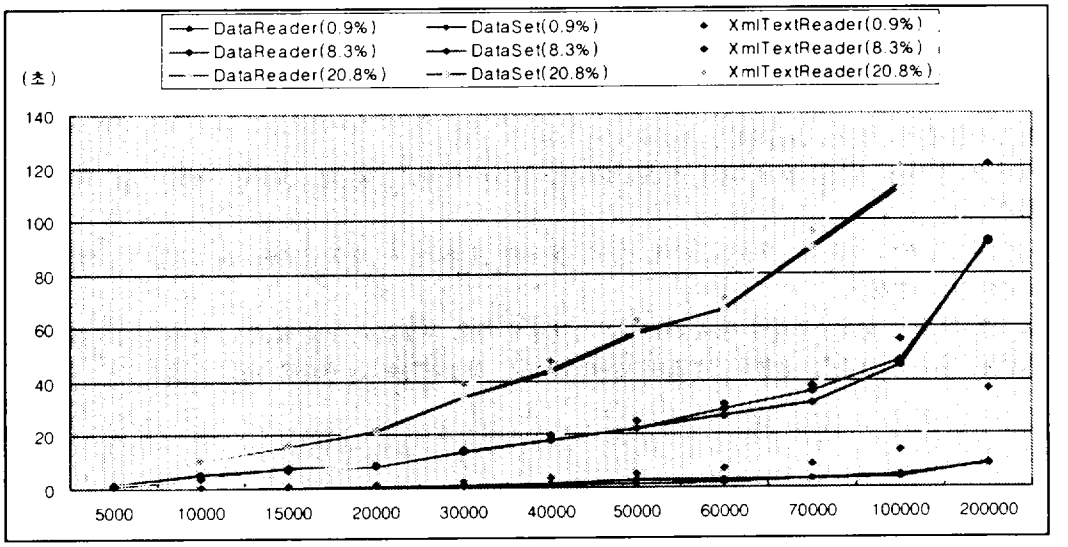


그림 9. 검색 비율(0.91%, 8.33%, 20.83%)에 따른 DataReader, DataSet, XmlTextReader 검색 속도 비교

4. 결 론

본 논문에서는 난수를 이용하여 다량의 학생정보 데이터를 생성한 후, C# 프로그램을 이용한 데이터베이스 검색과 XML 문서 검색의 성능을 평가하였다. 소량의 데이터 정보에서 검색을 수행하는 데는 XML 문서 검색이 월등하게 빨랐다. 그러나 데이터양이 많아질수록 XML 문서 검색 성능은 현격히 저하되어 검색 성능의 한계를 드러내었다. 검색 데이터양을 전체 레코드 수의 0.91%, 8.33%, 20.83%로 변화시켰을 때 약 15,000개 전후에서부터 데이터베이스 검색 성능에 비하여 좋지 못한 결과를 보였다. 또한 거의 유사한 성능을 보이는 DataReader와 DataSet의 성능은 레코드의 개수가 500만이었을 때, DataSet의 경우가 8.49%정도 검색시간이 더 드는 결과를 볼 수 있었다. 세 가지 검색 방법, DataReader와 DataSet, 그리고 XmlTextReader 모두 레코드를 검색해서 읽어온 후 결과를 출력하는데 걸리는 시간은 유사하였으며 검색한 레코드의 수에 비례하였다. DataReader의 경우 나머지 단계의 소요시간은 레코드 수에 영향을 받지 않고 거의 일정한 값을 유지하였고, DataSet의 경우는 DataAdapter 객체의 Fill() 메소드를 호출해서 DataSet에 검색된 데이터를 구성하는 시간이 검색한 레코드의 수에 비례하여 증가하였으며 나머지 단계는 일정한 값을 유지하였다. 마지막으로

XmlTextReader에서는 XpathDocument Object를 생성하는데 걸린 시간이 검색된 레코드 수에 거의 정비례하였다. 따라서 대용량의 데이터를 처리할 경우는 XML 문서 기반의 웹서비스를 제공하더라도 데이터베이스 시스템을 이용하여 저장 및 검색을 수행하여야겠지만 1만개 이하의 소량의 레코드를 유지하는 경우에는 XML 문서를 이용하여 검색하는 것이 더욱 효율적이다.

참고문헌

[1] Bray. T.. Paoli. J.. and Sperberg-McQueen. C.M.. Extensible Markup Language(XML). http://www.w3.org/ TR/ PR-xml-971208. Dec. 1997.

[2] Laurent S.. XML Element of Style. McGraw-Hill. 2000.

[3] 김성규. 연역 객체 지향 데이터베이스 언어 구현을 통한 XML 데이터 처리에 관한 연구. 정보처리학회논문지. 제9-D권. 제6호. Dec. 2002.

[4] 박민경. 홍의경. XML 문서 저장 시스템의 성능 평가. 정보기술연구소 논문집. 제4집. pp. 29-36. Aug. 2002.

[5] 김충성. 김용성. XSL를 이용한 XML 문서 검색에 관한 연구. 한국정보과학회 가을 학술발표논문집. Vol.26. No.2. 1999.

[6] 맹성현. 주종철. 문서구조화와 정보검색. 정보과학회지 특집 정보검색. Vol. 16. No. 8. pp. 6-15. 1998.

[7] 박상원. 정재목. 정태선. 김형주. XML과 데이터베이스. 정보과학회지. 제19권. 제1호. 2001.

Performance Evaluation of XML Document Retrieval and Database Retrieval

Mi-Kyung Kang, So-Jeong Park, Jung-Hoon Lee
Department of Computer Science and Statistics, Cheju National University

Abstract

It is necessary to efficiently retrieve and save the web documents requested by a user, according to the intensive increasement in the number of document transmitted via the Internet. XML (eXtensible Markup Language) saves the information selected by a user into a relational database as an instance of the XML form, enabling efficent retrieval of the saved information. This paper measures the performance of XML document retrieval in addition to that of database retrieval using a C# program, aiming at evaluating the performance of XML as an alternative way to cope with the increased Internet document. For the database retrieval, DataReader and DataSet classes are used, while XmlTextReader is used for XML document retrieval. We assume that data retrieval distributes randomly and is targeted to a large volume of student information virtually created for this experiment. The experiment result shows that XML document retrieval is superior to database retrieval for a small amount of data, but not efficient as the data amount increases, exposing the a limit of retrieval performance. The database system is desirable for the retrieval of large data volume. On the contrary, XML is also an efficient way for the retrieval of less than 10,000 records.