UniWeb-웹을 위한 UniSQL/X 데이타베이스 통로+

김평철*

UniWeb-A UniSQL/X Database Gateway for Web

Pyung-Chul Kim

〈요 약〉

대규모 멀티미디어 데이타베이스 서비스 시스템은 웹(Web)의 대규모 멀티미디어 정보 서비스의 우수성과 데이타베이스 시스템의 방대한 데이타베이스 관리 기능을 상호 보완적으로 통합함으로써 구현할 수 있다. 데이타베이스 통로(gateway)는 이러한 통합의 핵심적인 소프트웨어라 할 수 있다. 데이타베이스 통로의 필요성이 인식된 후로 현재 많은 상용 및 연구용 제품이 개발되어 왔다. 데이타베이스 통로의 구조는 데이타베이스 접속을 웹에 어떻게 연동시키는가에 따라 다양하게 나누어지고 구조에 따라 성능도 달라지게 된다.

본 논문에서는 여러가지 데이타베이스 통로의 구조에 대하여 분류체계를 제안하고, 현재 충남대학교에서 개발하고 있는 UniWeb의 설계 및 구현을 기술한다. UniWeb은 웹을 위한 UniSQL/X 데이타베이스 통로로서, DBMS의 최적화 기능을 최대한 활용할 수 있는 구조를 가지고 있으며, 웹 응용 개발을 위한 환경을 지원하고 있다.

+ 본 논문은 한국컴퓨터통신(주) 위탁과제 연구비에 의하여 지원받았음.

* 충남대학교 정보통신공학과

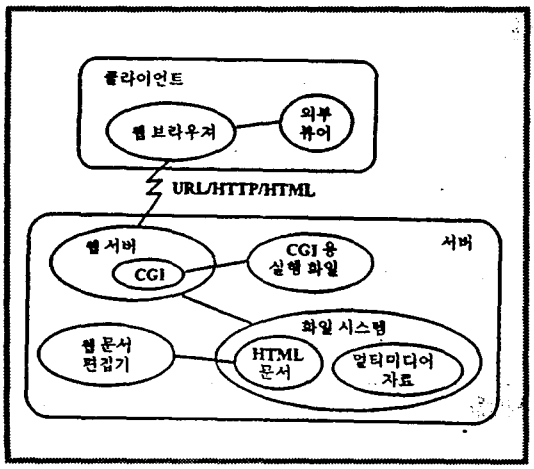
1. 서론

1.1. 월드 와이드 웹

월드 와이드 웹(World-Wide Web)은 스위스의 CERN에서 개발된 하이퍼미디어 방식의 대규모 정보 서비스 시스템으로서, 1992년 발표된 이후 인터넷을 중심으로 급속히 성장하여, 현재는 상업적인 응용에 사용되고 있기도 하다 [3,5]. 기존의 하이퍼미디어 시스템들이 사용자 접속 도구나 편집 기술 및 자료 저장 기법에 치중하여 개발된 반면 웹은 전세계에 광범위하게 퍼져 있는 여러 형식의 자료를 정보 서비스 환경에 접목시키는 데 촛점을 두었다. 이를 위하여 다음과 같은 근본적인 문제를 해결하고 있다.

- 인터넷을 통해 전세계에 퍼져 있는 자료를 식별하기 위한 기법 [2]
- 하이퍼미디어 문서를 기술하는 언어 [4]
- 망을 통해 하이퍼미디어 문서를 전송하는 데 필요한 통신 규약 [1]

〈그림 1〉 웹의 구조



〈그림 1〉은 웹의 구조를 간략하게 나타낸 것이다. 웹에서는 정보가 멀티미디어 자료나 다른 문서를 링크로 갖는 하이퍼미디어 문서로 표현된다. 클라이언트의 웹 브라우저는 개별 하이퍼미디어 문서를 번역하여 사용자 화면에 실연(presentation)한다. 이때, 각 멀티미디어 자료의 실연을 위해 적절한 외부 뷰어(external viewer)를 이용하기도 한다. 예를 들면, MPEG 방식의 동화상을 실연하기 위해 mpeg_play라는 명령어를 이용할 수도 있다. 웹 브라우저는 사용자가 원하는 문서의 주소를 식별하여 해당 웹 서버를 연결한다. 웹 서버는 클라이언트의 요구에 의해 화일 시스템에 저장된 하이퍼미디어 문서와 여기에 링크된 멀티미디어 자료를 전송한다. 이때, 웹 서버와 웹 브라우저간에는 HTTP(hypertext transmission protocol) [1]라는 통신 규약이 적용되고, 하이퍼미디어 문서는 HTML(hypertext markup language) [4]라는 언어로 기술된다. 전세계에 퍼져 있는 각 하이퍼미디어 문서 및 멀티미디어 자료는 고유의 식별자인 URL(uniform resource locator) [2]을 갖는다.

웹 서버는 운영 체제의 일반 프로그램을 호출할 수 있는 CGI(common gateway interface)를 지원한다. 이는 검색, 위치 지정이 가능한 이미지, 폼(form), 그리고 동적으로 생성되는 하이퍼미디어 문서를 다루는 데 자주 이용된다. CGI는 웹 서버로부터 여러 가지 입력 인자를 환경 변수 형태로 받고, 프로그램 수행 결과를 HTML 문서로 변환하여 웹 서버에 전달한다 [8]. 웹 기술은 대규모 멀티미디어 정보 서비스 측면에서 다음과 같은 우수성을 가지고 있다.

- 멀티미디어 데이타베이스를 사용자에게 보여

주는 방식으로 하이퍼미디어 방식을 이용하고 있어서 멀티미디어로 구성된 복잡한 정보의 전달 효과가 높다.

- HTML의 마크업 언어는 실연에 대한 구체적인 내용보다는 문서의 구성을 나타내는 꼬리표 (tag)를 기술하도록 되어 있어 웹 브라우저의 환경에 따라 적절한 실연이 가능하다. 예를 들어, 꼬리표가 〈H1〉인 문장을 그래픽 단말기에서는 굵고 큰 서체로 보일 수 있고, 문자 위주의 단말기에서는 반전된 문자로 보이게 할 수 있을 것이다. 이와 같이 실연과 꼬리표의 구분은 많은 종류의 사용자 플랫폼을 지원할 수 있어서 대규모 서비스에 적합하다.
- HTML 문서는 모두 서버에서 관리된다. 따라서, 서비스되고 있는 정보를 변경하는 것이 매우 용이하다.
- HTML, URL, HTTP 등이 이미 인터넷 표준으로 이미 등록되어 있거나 등록 진행 중이어서 전세계를 대상으로 한 광범위 서비스 구축이 용이하다.
- HTML에는 폼을 기술하는 기능이 있어서 사용자의 입력을 받기 위한 화면 처리를 쉽게 할 수 있다. 이 기능은 질의를 통한 데이타베이스 검색 서비스나, 주문 형태의 서비스를지원하고자 할 때 매우 유용하다.
- 1992년 발표된 이후 기능이 급속히 확장되고 있고, 다양한 관련 제품들이 개발되고 있다. 특히, 상용 웹 브라우저와 웹 문서 편집기가 등장하여 서비스 개발이 용이해지고 있다.

웹은 멀티미디어 정보 서비스 시스템으로서 위와 같은 우수성을 갖는 반면, 아직까지 자료의 저장소로서 데이타베이스에 대한 연결을 직접 지원하지 않고 있어, 대량의 자료를 가진 데이타베이스 서비스를 개발하는 데에는 문제가 많다. 예를 들면, 복잡하고 방대한 멀티미디어 자료를 데이타베이스 관리 시스템의 도움 없이 운영 체제의 화일 시스템에 관리한다는 것은 매우 힘들다. 또한, 데이타베이스 검색을 위한 사용자 접속으로서 HTML의 기능은 몇가지 제약점을 갖고 있다. 예를 들면, HTML의 폼을 이용한 질의 화면 구성은 기존의 데이타베이스 응용 개발도구(예, PowerBuilder나 Visual Basic 등)를 이용하는 것에 비해 매우 단순한 형태를 갖게 된다. 일반적으로 데이타베이스 서비스 시스템은 자료입력을 위한 업무 프로그램과 연동되는데, HTML의 폼 기능은 이러한 목적으로 사용하기에는 부족한 기능이 많다.

복잡한 자료의 실연에 있어서도 HTML의 기능이 부족할 수 있다. 예를 들면, 두 개 이상의 시간적인 개념을 갖는 멀티미디어 자료를 서로 동기화하여 실연한다든가, 혹은 3차원 자료를 서비스하고자 할 때에는 외부 뷰어의 도움을 받아야 한다.

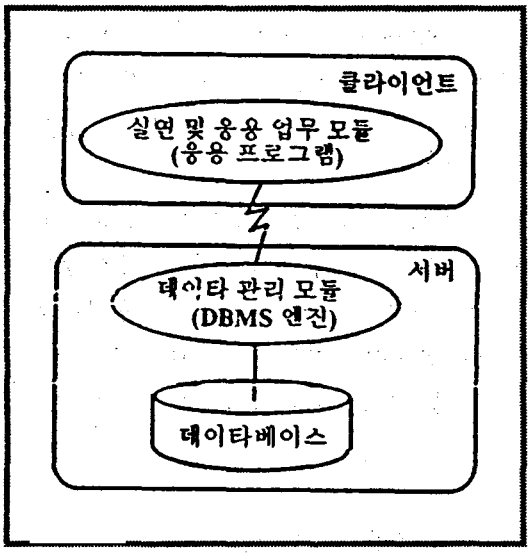
1.2 데이타베이스 시스템

은행의 예입/인출 업무, 항공사의 좌석 예약 업무 등으로 대표되는 클라이언트-서버 데이타베이스 시스템은 크게 세 개의 소프트웨어 모듈로 분리된다. 첫째는 데이타 관리 모듈로서, 특정 응용과 독립되어 데이타베이스의 구축, 검색 등을 지원한다. 일반적으로 DBMS(database management system)엔진이 여기에 해당한다. 둘째는 응용 업무 모듈로서, 각 업무의 데이타 처리절차를 구현하고 있다. 예를 들면, 은행의 예입

업무에 대한 응용 업무 모듈은 데이타 관리 모듈을 이용하여 해당 계좌의 잔금을 변경하는 절차를 구현하게 될 것이다. 셋째는 사용자에게 대화형으로 자료를 입력 받고 결과를 출력하는 실연 모듈이다 [7].

클라이언트-서버 데이타베이스 시스템의 구조는 응용에 따라 여러 가지가 있으나, 현재 가장 많이 사용되고 방식은 〈그림 2〉와 같은 2겹 구조이다 [9].

〈그림 2〉 2겹 클라이언트-서버 데이타베이스 시스템 구조



2겹 구조에서 클라이언트의 응용 프로그램은 여러 가지 응용 개발 도구를 이용하여 쉽게 작성할 수 있다. 클라이언트의 응용 프로그램과 서버의 DBMS 엔진을 접속하기 위해서 여러 가지 중간모(middleware)가 사용되기도 한다. DBMS 엔진은 내장절차(stored procedure)라는 기능을 이용하여 응용 업무 모듈의 일부 기능을 수행할 수도 있다.

한편, 앞으로는 응용 업무 모듈을 실연 모듈에서 분리하여 서버에 두고 여러 클라이언트가 공유하는 3겹 구조가 점점 더 많이 사용될 것으로 보인다. 3겹 구조에서는 응용 업무 모듈이 운영 체제, 트랜잭션 처리 모니터, 그리고 기타 관련된 중간모의 도움을 받아 수행되기 때문에 2겹 구조에 비하여 신축성, 확장성, 그리고 대규모 시스템에서 성능이 좋을 것으로 예상되고 있다 [9, 10].

클라이언트-서버 데이타베이스 시스템은 OLTP(on-line transaction processing)와 같은 주문형 서비스를 구축하는 데 매우 적합하다. 주문형 서비스란 사용자가 일정한 양식을 통해 자료를 입력함으로써 트랜잭션을 발생하고, 서비스 시스템은 이 트랜잭션을 데이타베이스에 수행함으로써 업무가 이루어진다. 데이타베이스 서비스는 주문형 서비스보다는 정보 검색 서비스가 주를 이룬다는 면에서 클라이언트-서버 데이타베이스 시스템의 구조가 적합하지 않을 수도 있다.

멀티미디어 데이타베이스 서비스 시스템은 방대한 자료를 데이타베이스로 관리할 수 있는 기능이 있어야 한다. 클라이언트-서버 데이타베이스 시스템의 DBMS 엔진은 이를 위한 많은 기능을 이미 구현하고 있다. 예를 들면, 많은 상용 DBMS가 다음과 같은 기능을 지원하고 있다.

- 복잡한 자료형과 자료들간의 관계를 모델하기 위한 데이타 모델링 기능
- 사용자 질의의 최적화된 수행
- 트랜잭션 처리 기능
- 파손으로 인한 데이타베이스 회복 기능
- 접근 권한 관리 기능
- 디스크, 화일 및 색인 관리 기능
- 데이타베이스의 시각적인 구축 및 편집 기능

데이타베이스의 접근은 ISO 표준 데이타베이스 언어인 SQL을 통하여 이루어진다. SQL 언어는 비절차적인(non-procedural) 언어로서 데이타베이스의 복잡한 물리적 구조와 관계 없이 질의를 표현할 수 있다. 응용 프로그램 개발자는 일반 프로그래밍 언어에 SQL을 내포하거나, 폼 개발 도구를 이용하여 응용 프로그램을 개발한다. 특히, 폼 개발 도구를 이용하는 것은 생산성이 뛰어난 것으로 알려져 있다.

2겹 혹은 3겹 구조의 클라이언트-서버 데이타베이스 시스템을 멀티미디어 데이타베이스 서비스에 적용하면 데이타베이스가 서버에 해당되고 가입자의 단말기가 클라이언트에 해당될 것이다. 가입자는 클라이언트의 실연 모듈에 의해 정보 서비스를 받게 된다. 멀티미디어 정보의 실연 사양(예를 들면, 화면 배치)은 클라이언트의 응용 프로그램에 코드화 되어 있거나 폼 개발 도구에 의해 작성된 폼 화일에 담겨져 있다. 즉, 응용 프로그램이 수행되거나 폼 화일이 번역되면서 필요한 자료를 서버의 데이타베이스에서 불러와 실연하게 된다. 실연 사양을 응용 프로그램에 코드화하는 방식은 실연 사양 변경이 어렵기 때문에 현실적으로 대규모 데이타베이스 서비스 환경에 적용하는 것은 매우 힘들다.

폼 화일을 이용하는 경우에도 다음과 같은 문제점이 발생한다. 멀티미디어 데이타베이스 서비스에서 자료의 실연은 매우 다양할 수 있다. 예를 들면, 전자 신문의 경우 기사마다 적절한 화면 배치가 다를 수 있다. 반면에 클라이언트-서버 데이타베이스 시스템에서 폼은 응용 업무마다 몇 가지 정형화된 형태를 갖는다. 따라서, 폼을 멀티미디어 데이타베이스 서비스 양식으로 이용하게 되면 매우 많은 종류의 폼이 생성, 관리되어야 할 것이다. 많은 종류의 폼 화일을 클라이언트마다 복사하여 관리하는 것은 비효율적일 수 있다. 특히, 일반적으로 데이타베이스 서비스의 가입자 수는 클라이언트-서버 데이타베이스 환경의 클라이언트 수에 비하여 훨씬 많기 때문에, 폼을 변경하는 것이 매우어렵게 된다. 이러한 문제점을 없애기 위해서는 폼을 서버에 관리하여 공유하는 구조가 바람직하다.

데이타베이스 서비스 가입자가 항상 같은 종류의 폼 번역기를 운영하고 있다고 가정하기는 어렵다. 서비스 범위가 넓어짐에 따라 가입자 환경은 더더욱 다양해질 것이다. 현재 사용되고 있는 폼 화일의 형식은 국제적인 표준이 없고 사용되는 폼 개발 도구마다 다르다. 클라이언트-서버 데이타베이스 시스템을 광범위 데이타베이스 서비스에 적용할 수 있기 위해서는 폼 화일의 형식에 대한 국제적인 표준이 진행되어야 한다.

1.3 웹과 데이타베이스의 통합의 장점

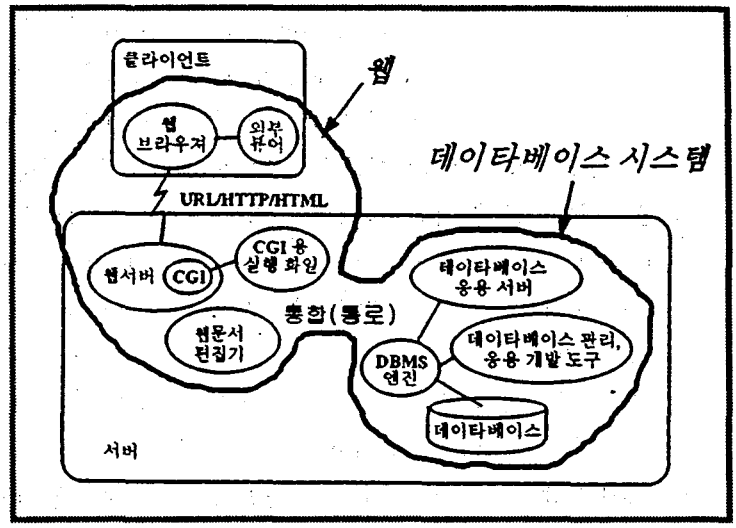
앞에서 살펴본 바와 같이, 대규모 멀티미디어 데이타베이스 서비스 시스템은 클라이언트-서버 데이타베이스 시스템의 방대한 데이타베이스 관리 기능과 응용 개발 도구, 웹의 대규모 서비스 구조가 동시에 필요하다. 〈그림 3〉은 웹와 데이타베이스 시스템의 상호 보완적인 통합의 개념을 표현한 것이다.

멀티미디어 데이타베이스를 DBMS 엔진과 데이타베이스 관리 도구를 이용하여 구축, 관리하고 이를 가입자에게 서비스하기 위해서는 웹의 구조를 이용한다. 데이타베이스 통로(database gateway)는 이와 같은 통합을 구현하는 핵심 부분이다.

위와 같은 상호 보완적인 통합은 클라이언트-서버 데이타베이스 시스템과 웹에서 다음과 같은 장점을 계승 받게 된다.

- 복잡한 자료형과 자료들간의 관계 모델링 용이
- 사용자 질의의 최적화된 수행
- 고성능 트랜잭션 처리
- 파손으로 인한 데이타베이스 회복
- 자료 접근 권한 관리
- 디스크, 화일 및 색인 관리
- 데이타베이스의 시각적인 구축 및 편집
- 하이퍼미디어 방식을 이용한 사용자 접속
- HTML 문서를 이용하여 많은 종류의 플랫폼 지원
- HTML 문서를 서버에 저장하여 대규모 서비스에 관리 용이
- 표준화된 HTML, URL, HTTP 등의 사용으로 광범위 서비스 용이
- HTML의 폼 기능을 이용한 사용자 질의 및 주문 형태 서비스 지원
- 기존의 웹 환경과 데이타베이스 시스템 환경에 접속

〈그림 3〉 데이타베이스 시스템과 웹의 통합 개념



1.4 논문의 구성

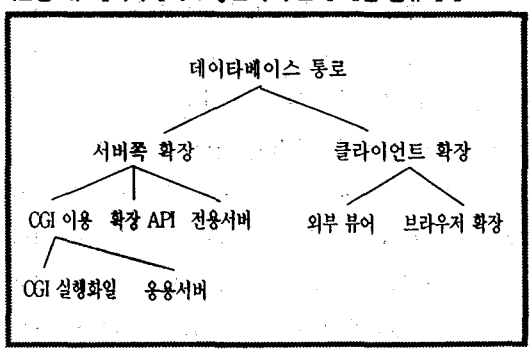
이 논문의 구성은 다음과 같다. 2장에서는 웹와 데이타베이스를 통합하는 핵심 부분이라고 할 수 있는 데이타베이스 통로의 구조에 대한 분류체계를 제안한다. 3장에서는 UniSQL/X 객체관계 DBMS에 연결하여 사용할 수 있는 UniWeb 통로의 설계 및 구현에 대하여 기술하고, 4장에서는 논문의 결론을 맺는다.

2. 데이타베이스 통로의 구조에 대한 분류체계

2.1 분류체계

데이타베이스 통로는 데이타베이스 응용 프로그램이 웹에 어떻게 연동되는가에 따라 〈그림 4〉와 같이 분류할 수 있다.

〈그림 4〉 데이타베이스 통로의 구조에 대한 분류체계



데이타베이스 통로의 구조는, 데이타베이스를 접속하는 프로그램이 웹 서버 쪽에 위치하는 "서버쪽 확장"과, 웹 브라우저 쪽에 위치하는 "클라이언트쪽 확장"으로 크게 나누어질 수 있다.

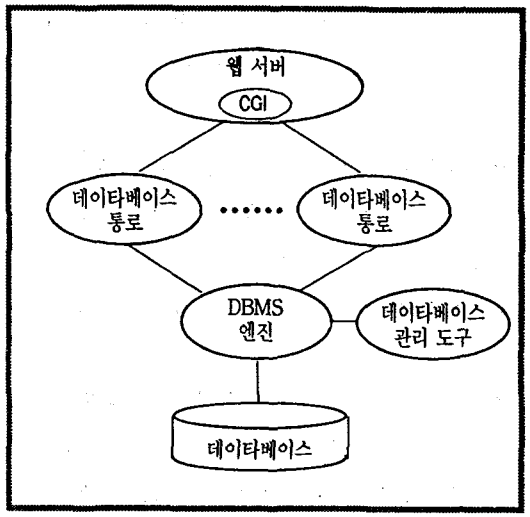
서버쪽 확장은 "CGI 이용", "확장 API", 그리고 "전용 서버" 방식으로 나누어질 수 있다. CGI 이용은 기존의 웹 서버가 지원하는 CGI 기능을 이용하여 데이타베이스를 접근하는 방식을 말하는데, 데이타베이스 응용 프로그램이 CGI용 실행화일 그 자체로 운영되는 "CGI 실행화일" 방식과 데이타베이스 응용 프로그램은 데몬(daemon) 방식으로 운영되고, CGI 실행화일을 통해 질의요구가 전달되는 "응용 서버" 방식으로 다시 분류된다. 확장 API 방식은 웹 서버가 지원하는 확장 API를 이용하여 데이타베이스를 접근하는 구조를 말한다. 그리고, 전용 서버 방식은 특정 DBMS를 접속할 수 있는 기능을 내재하고 있는 웹 서버를 말한다. 이때의 웹 서버는 DBMS 측면에서 보면, HTTP를 지원하는 데이타베이스 응용 프로그램의 하나라고 할 수 있다.

클라이언트쪽 확장은 기존의 웹 브라우저가 지원하는 외부뷰어 접속기능을 이용하여 데이타베이스를 접속하는 "외부뷰어" 방식과, 브라우저 자체에 데이타베이스 접속을 위한 기능을 포함시키는 "브라우저 확장" 방식으로 나눌 수 있다.

2.2 CGI 실행화일 방식

데이타베이스 시스템과 웹의 가장 단순한 통합 방법은 기존의 웹 서버 및 브라우저를 변경하지 않고, CGI 실행화일로 하여금 데이타베이스를 접근하게 하는 방식으로서, 〈그림 5〉와 같은 구조를 갖는다.

〈그림 5〉 CGI용 실행화일 방식의 구조



데이타베이스에 저장된 모든 자료는 데이타베이스 통로를 통해 접근된다. 데이타베이스 통로는 웹 서버의 CGI를 통하여 입력 인자를 받고, 데이타베이스 엔진을 통해 원하는 자료를 검색한 후, 이를 HTML 문서 형태로 변환하여 웹 서버에 넘겨준다. 이때의 데이타베이스 통로는 연결하고자 하는 DBMS의 응용 프로그램 개발 도구(예, ESQL/C, C/C++ API, Perl 등)를 이용하여 작성된다. 즉, 데이타베이스 응용 프로그램의 하나라 할 수 있다.

일반적으로 웹 서버의 CGI와 데이타베이스 통로 사이의 통신은 프로세스 파이프를 이용한다. 즉, 웹 서버는 데이타베이스 통로 프로세스를 생성한 후 프로세스 파이프를 연결하여 데이타베이스 통로 프로세스의 stdout을 결과로 받게 된다. 웹 서버는 하나의 요구를 처리할 때마다 새로운 데이타베이스 통로 프로세스를 생성하고 파이프를 연결한다. 그리고, 요구 처리가 끝나면 이 프로세스를 종료시킨다. 따라서, 데이타베이스 통로는 하나의 요구를 처리하기 위하여 DBMS에 연

결하고 로그인하는 과정을 매번 수행하게 된다.

CGI용 실행화일 방식의 데이타베이스 통로 구조는 다음과 같은 장점을 갖는다.

- 기존의 웹 서버, 웹 브라우저, 그리고 URL, HTTP, HTML 기술을 변경없이 사용할 수 있다.
- 데이타베이스 통로만의 개발/시험이 용이하여 구현이 쉽다.
- 차후, 응용의 확장으로 인하여 데이타베이스 접근 요구사항이 확대될 경우 데이타베이스 통로만을 확장함으로써 해결할 수 있다. 예를 들면, 데이타베이스 통로가 DBMS 뿐만 아니라 정보검색 시스템에 대한 사용자 요구를 처리한다든가, 혹은 분산 데이타베이스 처리를 하고자 할 때, 웹 서버에 변경 없이 확장이 가능하다.

그러나, 이 방식은 성능 면에서 몇가지 문제점을 갖는다. 첫째로, 모든 데이타베이스 자료는 심지어 그것이 단순한 형식을 갖는 하나의 레코드라 할지라도 데이타베이스 통로 프로세스를 통하여 전송되기 때문에 프로세스간 자료 복사, 프로세스 교체 등의 성능 저하 요인을 감수하게 된다.

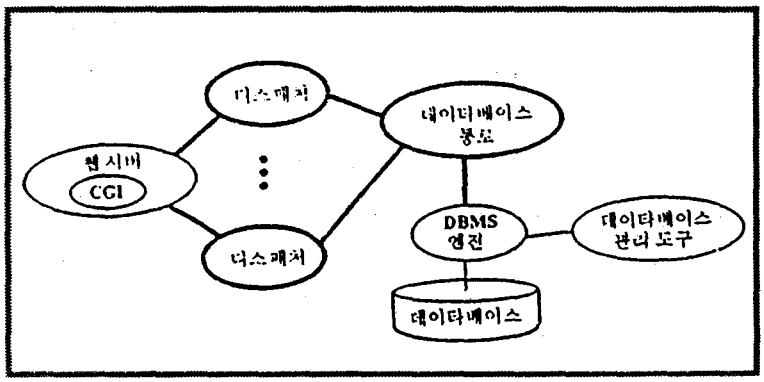
둘째로, 데이타베이스 응용 프로그램이 CGI 실행화일로 구동되기 때문에 수많은 동시 요구가 발생할 경우, 요구 갯수만큼의 데이타베이스 응용 프로세스가 생성되어야 한다. 일반적으로 데이타베이스 응용 프로그램은 그 크기가 크다. 따라서, 동시에 많은 갯수의 응용 프로세스를 생성하게 되면 시스템 자원의 부족과 성능 저하를 초래한다. 이 문제는 대규모 서비스에 치명적일 수 있다.

세째로, CGI용 실행화일은 한번 요구를 수행한 후 종료되기 때문에, 일반적으로 데이타베이스 시스템이 고성능을 위해 지원하고 있는 몇가지 중요한 최적화 기능을 활용하지 못하게 된다. 이러한 최적화 기능은 데이타베이스 응용 프로그램이 유사한 질의를 반복하여 수행하는 동안 데이타베이스와 계속 접속되어 있을 때 효과가 있도록 설계되어 있는 것이 일반적이다.

2.3 응용 서버 방식

CGI용 실행화일 방식의 성능 문제를 해결하기 위해 데이타베이스를 접속하는 프로세스를 데몬 방식으로 바꾼 것이 응용 서버 방식이다. 응용 서버 방식의 데이타베이스 통로는 웹 서버의 CGI에 의하여 구동되는 디스패처(dispatcher) 프로세스와 데이타베이스 검색 및 HTML문서 변환 기능을 수행하는 데몬 방식의 데이타베이스 응용 프로세스로 나누어져 있다. 〈그림 6〉은 이러한 방식의 구조를 보인 것이다.

〈그림 6〉 응용 서버 방식의 구조



디스패처는 웹 서버의 CGI로부터 프로세스 파이프로 연결되어 호출된다. CGI용 실행화일이기 때문에 사용자가 데이타베이스에 접근할 때마다 새로이 실행된다. 디스패처는 사용자가 요구한 질의를 수행할 수 있는 응용 프로세스를 식별하여 사용자의 요구를 전달한다. 데이타베이스 응용 프로세스는 질의를 수행하고, 그 결과를 HTML 형식으로 디스패처에 반환하게 된다. 디스패처는 그 결과를 웹서버에 전달한다.

데이타베이스 응용 프로세스의 주요 기능은 데이타베이스를 검색하여 그 결과를 HTML 문서로 변환하는 것이다. 따라서, 일반적인 데이타베이스 응용 프로그램 개발 도구(예, ESQL/C)를 이용하여 작성하게 된다. 데이타베이스 응용 프로세스는 서비스 시스템이 시작될 때, 웹서버, DBMS 엔진 등과 같이 시작되게 할 수 있다. 즉, 서비스가 시작될 때 DBMS에 접속하고, 서비스가 종료할 때까지 사용자 요구를 하나씩 수행한다.

응용 서버 방식은 기본적으로 CGI 기술을 이용하기 때문에 CGI 실행화일 방식의 장점을 그대로 갖는다. 아울러, CGI 실행화일 방식에서의 여러 가지 성능 문제점을 해결한다.

첫째로, 디스패처는 데이타베이스를 접속하지 않고, 단순히 응용 프로세스를 식별하여 사용자 요구를 넘겨주기 때문에 프로세스의 크기가 매우 작다. 따라서, 동시 요구가 매우 많아도 시스템의 자원을 많이 차지하지 않아 대규모 서비스에서 성능 및 자원부족 문제를 발생시키지 않는다. 데이타베이스를 접근하는 프로세스 갯수는 동시에 서비스되는 사용자의 요구 갯수와 관계가 없다. 데이타베이스 응용 프로세스의 갯수를 시스템의 자원 상태와 사용자 요구의 많고 적음에 따라 적절하게 조정함으로써 대규모 서비스에 대처할 수도 있다.

둘째로, 데이타베이스 응용 프로세스가 하나의 질의요구 때마다 생성/종료되지 않고 서비스가 종료될 때까지 데몬 방식으로 수행되기 때문에, 일반적으로 데이타베이스 시스템이 고성능을 위해 지원하고 있는 몇가지 중요한 최적화 기능을 충분히 활용하게 된다.

대부분의 관계 및 객체관계 DBMS는 반복되는 질의의 성능을 극대화하기 위해 두 가지 최적화 기능을 지원한다. 첫째로, 질의가 처음 수행될 때, 이를 컴파일하고 수행방법을 최적화하여 그 접근계획을 저장해 둔다. 그리고, 같은 질의가 요구되면 바로 접근계획을 실행함으로써 질의의 컴파일 및 최적화 비용을 최소화한다. 둘째로, 한번 접근된 객체는 응용 프로세스의 메모리에 저장해 두었다가, 다시 접근되는 경우 DBMS 엔진에 요구하지 않고 바로 메모리에서 찾는다 (이 기능은 일반적으로 객체관계형 DBMS에서만 지원된다).

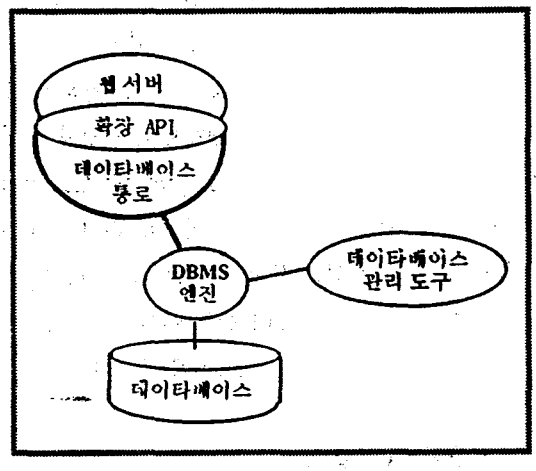
2.4 확장 API 방식

앞에서 기술한 CGI를 이용한 통합 방식의 성능 문제를 해결하기 위하여 웹서버로 하여금 데이타베이스 시스템을 직접 접속하도록 할 수 있다. 웹서버 중에는 서버의 기능을 응용 프로그래머가 확장할 수 있도록 API를 지원하고 있는 것들이 있다. (예, Netscape의 NSAPI). 이러한 API는 CGI와 마찬가지로 데이타베이스 접속 기능과는 관계 없이 범용 응용 확장을 위해 지원되고 있다. 다만, 확장 API를 이용하여 구축된 사용자 응용은 웹 서버에 동적으로 링크되어 하나의 프로세스로 수행되는 것이 일반적이기 때문에, CGI에 비

하여 성능이 우수하다.

확장 API 방식의 데이타베이스 통로는 웹 서버의 확장 API와 DBMS의 API를 이용하여 구축될 수 있다. 〈그림 7〉은 확장 API를 이용한 데이타베이스 통로의 구조를 보인 것이다. 이 방식에서는 웹 브라우저를 비롯한 URL, HTTP, HTML 문서 등 기존의 웹 기술을 그대로 이용할 수 있으나, 웹 서버는 확장 API를 지원하는 것만 이용할 수 있다.

〈그림 7〉 확장 API 방식의 구조



확장 API를 이용한 데이타베이스 통로에서는 데이타베이스의 자료가 DBMS 엔진에서 바로 웹 서버를 통해 전송되므로 프로세스간 자료 복사 및 프로세스 교체로 인한 부가 비용을 줄일 수 있다.

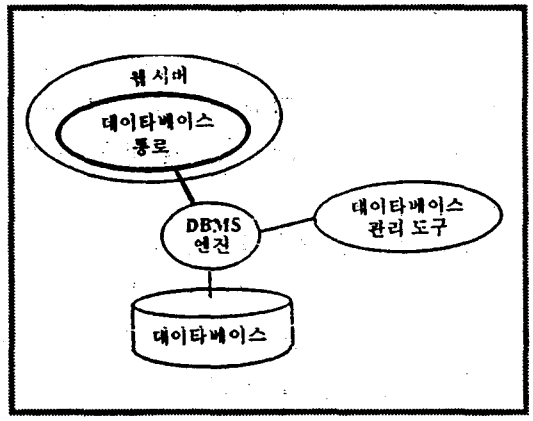
반면에 다음과 같은 단점을 가진다.

- 데이타베이스 통로의 운영 환경이 웹 서버와 같은 프로세스로 연동되어 있어서 개발/
- 시험 등이 단순하지 않고, 새로운 응용으로 확장하고자 할 때 신축성이 떨어진다.
- 웹 서버의 확장 API는 아직까지 표준이 없는 상태이기 때문에 개발된 데이타베이스
- 통로를 다른 웹 서버에서 사용할 수 없는 경우가 많다.

2.5 전용 서버 방식

전용 서버 방식은 웹 서버에 특정 DBMS 엔진을 직접 접속할 수 있는 데이타베이스 통로를 추가하는 방식이다. 〈그림 8〉은 이러한 구조를 보인 것이다. 이 방식은 웹 서버의 확장 API를 이용하는 것과 마찬가지로 웹 브라우저를 비롯한 URL, HTTP, HTML 문서 등 기존의 웹 기술을 그대로 이용할 수 있으나, 웹 서버는 특정 제품을 사용해야 한다.

〈그림 8〉 전용 서버 방식의 구조



DBMS 전용 웹 서버는 DBMS 엔진 측면에서 보면 HTTP 통신규약을 지원하는 데이타베이스 응용 프로그램의 하나로 볼 수 있다. 전용 서버는 API나 특정 스크립트 언어를 지원하여 데이타베이스 응용 프로그램을 작성할 수 있도록 하고 있다.

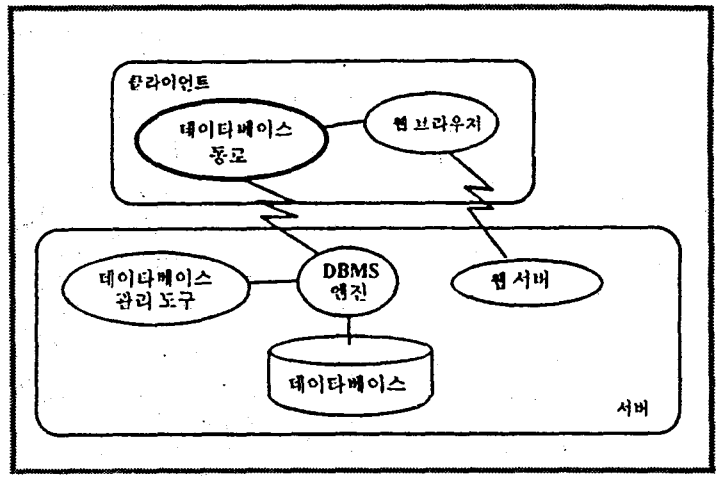
전용 서버 방식은 앞에서 소개한 확장 API 방식의 장단점을 모두 갖는다. 특히, 특정 웹 서버에 종속되기 때문에 차후에 확장하거나 이식하고자 할

때 신축성이 떨어질 수 있다.

2.6 외부 뷰어 방식

웹 브라우저는 MIME 자료 중 자체에서 실연할 수 없는 것은 외부 응용에 연결하여 실연할 수 있도록 지원하고 있다. 이를 데이타베이스 통로로서 활용하는 방식이 외부 뷰어 방식이다. 〈그림 9〉는 외부 뷰어 방식의 데이타베이스 통로를 보인 것이다.

〈그림 9〉 외부 뷰어 방식의 구조



외부 뷰어 방식에서 웹은 서비스를 안내하고, 사용자가 데이타베이스 응용을 선택하면 외부 뷰어를 띄워 주는 역할만 담당한다. 데이타베이스의 접속은 외부 뷰어를 통하여 이루어지게 된다. 외부 뷰어는 자체의 사용자 접속 기능을 가지고 있고, 원격지의 DBMS를 접근하기 위한 통신 규약 등을 가지고 있다. 엄격한 의미에서, 이 구조는 웹 서비스와 DB 서비스를 따로이 지원한다고 할 수 있다.

외부 뷰어로는 이미 많이 개발되어 있는 폼 실행기가 사용될 수 있다. 즉, 사용자는 웹을 통하여 웹 서버에 저장된 폼을 가져오고, 폼 수행기를 통해 이를 실행하는 것이다. 서버에 저장된 폼 자료와 사용자가 이용하는 폼 수행기는 서로 호환성이 있어야 한다.

외부 뷰어 방식에서는 데이타베이스 응용 프로그램이 웹과 관계 없이 기존의 클라이언트-서버 데이타베이스 응용과 같게 개발된다. 따라서, HTML의 폼 기능보다 훨씬 다양한 사용자 질의 화면을 구성할 수 있고, 여러 가지 개발 도구를 이용하여 생산성을 높일 수도 있다.

그러나, 외부 뷰어를 이용하기 때문에, 사용자는 웹 브라우저 이외의 별도의 소프트웨어 기능을 익혀야 한다. 사용자에게는 웹 브라우저 단일 환경에서 모든 서비스를 받는 것이 쉬울 것이다.

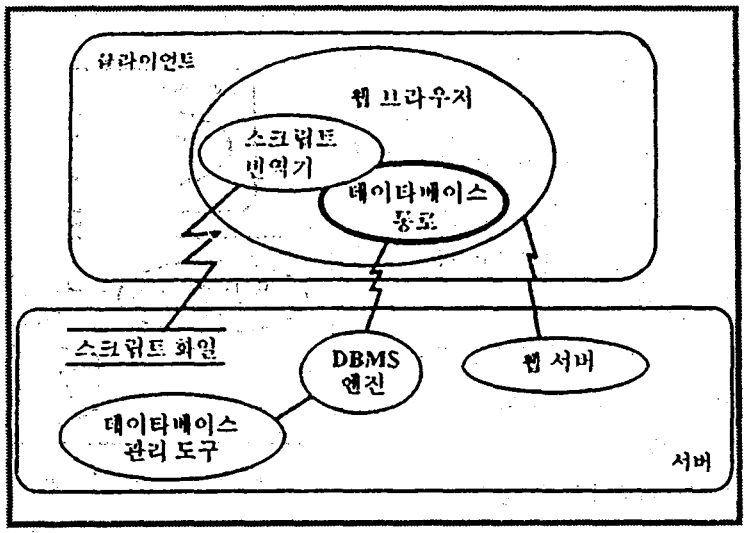
외부 뷰어를 이용하는 방식의 가장 큰 문제점은 대규모 서비스가 힘들다는 것이다. 클라이언트 쪽에 위치하는 외부 뷰어를 이용하여 데이타베이스를 접속하기 때문에, 사용자가 서비스를 받을 수 있으려면 해당 외부 뷰어를 설치하고 있어야 한다. 또한, 외부 뷰어가 데이타베이스를 접속하기 위해서는 클라이언트용 데이타베이스 접속 프로그램도 설치되어 있어야 한다. 현재까지 폼 수행기나 클라이언트용 데이타베이스 접속 프로그램에 대한 표준이 없다. 이러한 표준 없이 클라이언트 환경을 통일하기는 어렵고, 따라서 대규모 서비스를 기대하기는 힘들다.

2.7 브라우저 확장 방식

브라우저 확장 방식은 외부 뷰어를 이용하는 방식에서 데이타베이스를 접속하는 데 이용된 외부 뷰

어의 기능을 웹 브라우저에 포함시킨 것이다. 〈그림 10〉은 브라우저 확장 방식의 데이타베이스 통로 구조를 보인 것이다.

〈그림 10〉 브라우저 확장 방식의 구조



사용자는 웹을 이용하여 서버 쪽에 저장된 데이타베이스 스크립트를 선택한다. 선택된 스크립트가 클라이언트로 전송되어 브라우저가 수행한다. 따라서, 브라우저는 스크립트 번역기를 포함하고 있어야 한다. 스크립트에는 데이타베이스를 접속하는 부분이 있고, 이는 브라우저에 포함된 데이타베이스 통로를 통해 수행된다. 데이타베이스 통로는 DBMS 엔진을 접속하여 질의를 수행한다.

외부 뷰어를 이용하는 방식과의 큰 차이점은 스크립트 번역기와 데이타베이스 통로가 브라우저에 포함되어 있기 때문에 하나의 통합된 환경에서 서비스 받을 수 있다는 것이다.

데이타베이스 검색을 위한 사용자 접속이 스크립트에 의해 정의되기 때문에, HTML의 기능 제약으로부터 벗어날 수 있다. 예를 들면, HTML 폼 기능보다 훨씬 편리한 질의 화면을 지원할 수도 있고, 3차원 자료나 시간적인 개념을 갖는 두 개 이상의 자료를 동기화하여 실연하는 것 등도 가능하게 된다.

브라우저 확장 방식이 대규모 서비스에 사용되기 위해서는 브라우저 쪽에 확장된 부분인 스크립트 번역기와 데이타베이스 통로에 대한 표준이 정해져야 한다. 스크립트 번역기의 경우, 현재 Netscape 2.0에서 JavaScript를 지원하고 있으나 많은 브라우저에서는 이를 지원하고 있지 않다. 특히, 스크립트에서 사용될 수 있는 데이타베이스 접속에 대해서는 거의 개발된 것이 없다고 할 수 있다. 또한, 이 방식에서는 데이타베이스 응용 프로그램이 스크립트를 이용하여 작성되기 때문에, 기존의 클라이언트-서버 데이타베이스 응용 프로그램 개발도구를 이용하지 못한다.

주의할 것은 이러한 문제점이 브라우저 확장 방식의 구조보다는 현재의 기술진도에서 연유한다는 점이다. 웹 브라우저에 스크립트 번역기를 포함하는 것은 데이타베이스 응용과는 별도로 의미가 있기 때문에 조만간 대부분의 브라우저가 하나의 표준 스크립트를 공통적으로 지원할 것으로 기대된다. 아울러, 이러한 스크립트에서 사용할 수 있는 데이타베이스 접속 부분도 각 DBMS 개발자들에 의해 곧 공급되리라 생각한다.

3. UniWeb

UniWeb[11] 현재 충남대학교 정보통신공학과에서 개발하고 있는 UniSQL/X용 웹 데이타베

이스 통로이다. UniSQL/X는 대표적인 객체관계 DBMS의 하나로서 클라이언트-서버 방식의 엔진 구조를 가지고 있으며, 질의어인 SQL/X는 관계 DBMS의 SQL 언어를 객체지향 방식으로 확장한 것이다. UniWeb의 주요 응용은 대규모 멀티미디어 데이타베이스 서비스로서 전자박물관, 전자미술관, 전자신문 등이 대표적인 예라 할 수 있다.

3.1 UniWeb의 구조

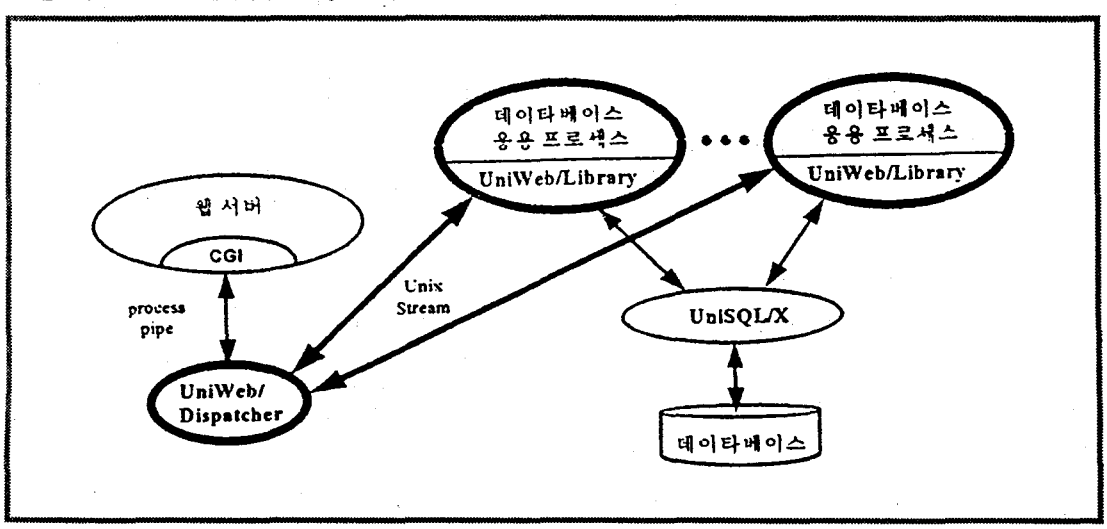
데이타베이스 응용프로그램이 CGI 실행화일 형태로 연결되는 경우에는 데이타베이스 접속시마다 새로운 프로세스가 생성되어 실행된 후 종료된다. 따라서 프로세스의 생성 및 종료 등과 같은 성능 부담이 적지 않고, 특히, 같은 질의의 반복 수행 성능을 극대화하기 위해 DBMS가 지원하고 있는 많은 최적화 기능을 활용하지 못하게 된다.

〈그림 11〉은 UniWeb 데이타베이스 통로의 구조를 보인 것이다. UniWeb 데이타베이스 통로는 웹 서버의 CGI에 의하여 구동되는 디스패처(dispatcher) 프로세스와 데이타베이스 검색 및 HTML문서 변환 기능을 수행하는 데몬 방식의 데이타베이스 응용 프로세스로 나누어져 있어서, CGI 응용 서버 방식이라 할 수 있다. 데이타베이스 응용 프로세스는 UniWeb 라이브러리를 이용하여 작성된다.

UniWeb/디스패처 프로세스

UniWeb/디스패처는 웹 서버의 CGI로부터 프로세스 파이프로 연결되어 호출된다. CGI용 실행화일이기 때문에 사용자가 데이타베이스에 접근할 때마다 새로이 실행된다. UniWeb/디스패처는 사용자가 요구한 URL을 웹 서버로부터 전달받아 PATH_INFO를 이용하여 요구된 질의를 수행할 수 있는 응용 프로세스를 식별한다. UniWeb/디스패처는 웹 서버에서 받은 모든 CGI 환경 변수값을 Unix Stream socket을 이용하여 해당 데이타베이스 응용 프로세스에 전달한다. 데이타베이스 응용 프로세스는 전달된 변수값을 이용하여 질의를 수행하고, 그 결과를 HTML 형식

〈그림 11〉 UniWeb 데이타베이스 통로의 구조



으로 UniWeb/디스패처에 반환하게 된다. UniWeb/디스패처는 그 결과를 웹 서버에 전달한다.

UniWeb/디스패처는 데이타베이스를 접속하지 않고, 단순히 응용프로세스를 식별하여 환경변수값을 넘겨주기 때문에 프로세스의 크기가 매우 작다 (Solaris 2.4에서 실행화일의 크기가 약 15K 바이트 정도). 따라서, 동시 요구가 매우 많아도 시스템의 자원을 많이 차지하지 않아 대규모 서비스에서 성능 및 자원부족 문제를 발생시키지 않는다.

데이타베이스 응용 프로세스

데이타베이스 응용 프로세스의 주요 기능은 데이타베이스를 검색하여 그 결과를 HTML 문서로 변환하는 것이다. 따라서, 일반적인 데이타베이스 응용 프로그램 개발 도구(예, ESQL/C)를 이용하여 작성하게 된다. 웹 사용자의 요구는 환경변수를 통해 전달된다. 데이타베이스 응용 프로세스는UniWeb/디스패처와 통신하거나 전달된 환경변수값 등을 읽어들이기 위해 UniWeb/라이브러리를 이용한다. 아울러, UniSQL/X DBMS를 통하여 질의를 수행하기 위하여 UniSQL/X 클라이언트 라이브러리를 이용하기도 한다.

데이타베이스 응용 프로세스는 서비스 시스템이 시작될 때, 웹서버, DBMS 엔진 등과 같이 시작된다. 일단, 서비스가 시작되면 바로 UniSQL/X에 연결한다. 이때, UniSQL/X의 클라이언트 라이브러리를 통하여 데이타베이스 스키마 일부를 적재하고, UniSQL/X 웍스페이스(workspace)를 초기화하는 등 고성능 질의처리 및 객체 접근을 위한 환경을 준비한다. 데이타베이스 응용 프로세스가 데몬 형태로 실행되기 때문에, 이 환경은 계속되는 사용자 질의 처리에 매우 높은 성능을 지원하게 된다.

UniSQL/X는 반복되는 질의를 위해 두 가지 최적화 기능을 지원한다. 첫째로, UniSQL/X는 질의가 처음 수행될 때, 이를 컴파일하고 수행방법을 최적화하여 그 접근계획을 저장해 둔다. 그리고, 같은 질의가 요구되면 바로 접근계획을 실행함으로써 질의의 컴파일 및 최적화 비용을 최소화한다. 둘째로, 한번 접근된 객체는 응용 프로세스의 메모리(웍스페이스라 부름)에 저장해두었다가, 다시 접근되는 경우 UniSQL/X 엔진에 요구하지 않고 바로 메모리에서 찾는다. 이는 웍스페이스를 관리하지 않는 DBMS에 비하여 수십, 심지어 수천배의 성능 향상을 가져오는 것으로 알려져 있다 [6].

UniWeb에서는 데이타베이스 응용 프로세스가 데몬 형식으로 운영되기 때문에 UniSQL/X가 지원하는 위와 같은 최적화 기능을 모두 활용하게 된다. 만약, 데몬방식으로 운영되지 않고 CGI용 실행화일로 수행된다면 하나의 질의를 처리하고 프로세스가 종료되어 버리기 때문에, 매번 질의를 컴파일하고 최적화할 뿐만 아니라, 항상 DBMS 엔진에서 객체를 가져오게 되어 성능이 크게 저하된다.

UniWeb에서 데이타베이스를 접근하는 프로세스의 갯수는 동시 요구의 갯수와 관계가 없다. 따라서, 대규모 서비스에서도 시스템의 자원에 크게 제한 받지 않을 수 있다. 아울러, 데이타베이스 응용 프로세스의 갯수를 시스템의 자원 상태와 사용자 요구의 많고 적음에 따라 적절하게 조정함으로써 대규모 서비스에 대처할 수 있다.

3.2 UniWeb의 기능

UniWeb을 사용하기 위해서는 먼저,

UniWeb/디스패처가 CGI용 실행화일로 인식되도록 웹 서버의 환경을 설정해야 한다. NCSA 1.5를 사용할 경우 다음과 같은 절차를 수행하면 된다.

1. httpd가 수행 중이면 이를 중단시킨다.
2. "srm.conf" 화일에 다음과 같은 줄을 넣는다 (UniWeb을 /UniWeb 디렉토리에 설치하였다고 가정).

   ScriptAlias /uw/ /UniWeb/bin/
3. httpd를 수행한다.

일반적인 CGI용 실행화일을 지정할 때와 같이 /uw/dispatcher/라는 URL을 통해 디스패처를 호출할 수 있다. 다만, 응용 프로세스를 지정하기 위해서는 URL의 마지막에 응용 이름을 덧붙이면 된다.

예를 들면, 응용 이름이 test__appl이라면, /uw/dispatcher/test__appl이라는 URL을 쓰면 된다(이때, test__appl은 PATH__INFO라는 환경변수를 통하여 UniWeb/디스패처에 전달된다.) UniWeb의 모든 응용 프로세스는 유일한 응용 이름을 갖는데, test__appl이 바로 그 예이다.

HTML 폼을 이용한 사용자 요구는 일반적으로 질의값이 결부되어 있다.

이는 QUERY__STRING이라는 환경변수를 통하여 전달된다. 아래 예제는 사용자에게서 param1 값을 받아서 test__appl이라는 응용 서버를 통해 질의를 처리하고자 할 때에 대한 HTML 폼을 보인 것이다.

```
〈FORM ACTION="http:/uw/dispatcher/test__appl" METHOD=POST〉
〈INPUT NAME="param1" TYPE="text"〉
.....
```

경우에 따라서는 폼을 이용하지 않고, 바로 HREF를 이용하여 URL로 지정할 수도 있다. 아래는 이러한 예이다.

```
HREF="http:/uw/dispatcher/test__appl?param1=value1&...."
```

UniWeb의 데이타베이스 응용 프로그램은 UniSQL/X의 응용 개발 도구(예, ESQL/C, C/C++ API 등)와 UniWeb/라이브러리를 이용하여 작성된다. UniWeb/라이브러리는 사용자에게 기존의 CGI 프로그래밍 환경과 동일한 환경을 지원한다. 즉, 비록, 통신을 통하여 UniWeb/디스패처와 연결되어 있으나, 프로그래머에게 이러한 복잡함을 숨기도록 설계되어 있다. 예를 들면, 응용 프로그램에서 stdout을 통해 출력하는 모든 것은 자동적으로 UniWeb/디스패처에 전달된다.

UniWeb/라이브러리는 UniWeb/디스패처와의 통신을 위한 함수, CGI 환경을 위한 함수, 그리고 기타 오류처리 함수들로 이루어져 있다. 다음은 각 함수를 간략하게 설명한 것이다.

- uw__init__env() - UniWeb 응용 환경으로 설정한다.
- uw__final__env() - UniWeb 응용 환경을 종료한다.
- uw__connect__client() - UniWeb/디스패처로부터 요구를 기다린다.
- uw__disconnect__client() - UniWeb/디스

패처에 결과보내기를 종료한다.

- uw_cgi_init_env() - CGI 환경을 설정한다.
- uw_cgi_final_env() - CGI 환경을 종료한다.
- uw_cgi_find_form_entry() - 폼 입력인자를 검색한다.
- uw_cgi_next_form_entry() - 폼의 다음 입력인자를 찾는다.
- uw_get_error_code() - 현재의 오류코드를 얻는다.
- uw_get_error_message() - 현재의 오류메시지를 얻는다.
- uw_ht_error_message() - 현재의 오류메시지를 HTML로 변환한다.

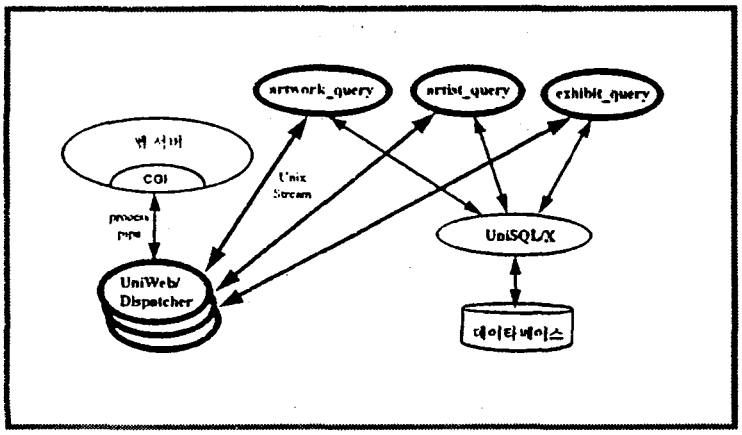
현재, 충남대학교 정보통신공학과에서는 UniWeb 응용 개발의 생산성 향상을 위해 UniWeb을 확장하고 있다. 확장의 주요 기능은 UniWeb/라이브러리 함수를 보다 많은 응용에서 사용할 수 있도록 보강하는 것과, HTML에 SQL/X 기능을 추가하여 프로그래밍 없이 확장된 HTML 작성만으로 응용 개발이 가능하도록 하는 것이다. 특히, 후자는 많은 상용 제품이 제한적으로나마 지원하고 있는 기능으로서 사용자의 응용 개발 생산성 향상에 큰 도움이 될 것으로 기대된다.

3.3 UniWeb 적용 예제: 전자미술관

우리는 최근에 UniWeb과 UniSQL/X을 이용하여 초고속공공응용서비스 중 문화재정보, 전자박물관, 전자미술관 등을 개발 완료하였다. 문화재정보는 문화재관리국의 8700여건의 지정 및 비지정 문화재, 전자박물관은 국립중앙박물관의 1000여건의 유물, 그리고 전자미술관은 국립현대미술관의 1000여건의 소장작품, 1000여명의 미술인, 그리고 200여건의 전시회 정보에 대하여, 서지, 전문(full-text), 이미지, 음성, 동화상 등을 데이타베이스에 저장하여 웹으로 검색할 수 있도록 개발한 것이다. 여기서는 전자미술관의 예를 기술한다.

〈그림 12〉는 전자미술관의 소프트웨어 시스템 구성도를 보인 것이다. 여기서는 데이타베이스 응용 프로세스로서 작품검색 질의를 수행하는 artwork_query, 작가검색 질의를 수행하는 artist_query, 그리고 전시회검색 질의를 수행하는 exhibit_query 등이 있다.

〈그림 12〉 전자미술관의 시스템 구성도

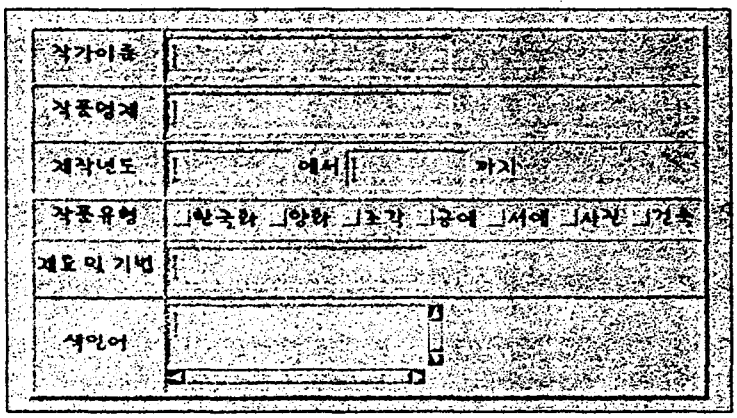


〈그림 13〉과 〈그림 14〉는 작품검색 질의에 사용되는 HTML과 이에 대한 사용자 화면을 각각 나타낸 것이다. 첫번째 줄에 기술되어 있는 바와 같이 FORM의

〈그림 13〉 작품검색을 위한 HTML FORM

```
<FORM ACTION="/uw/dispatcher/artwork_query" METHOD=GET>
<CENTER><TABLE BORDER=3>
<TR> <TD ALIGN=CENTER> 작가이름
    <TD> <INPUT TYPE="text" NAME="name" SIZE=40>
<TR> <TD ALIGN=CENTER> 작품명제
    <TD> <INPUT TYPE="text" NAME="title" SIZE=40>
<TR> <TD ALIGN=CENTER> 제작년도
    <TD> <INPUT TYPE="text" NAME="production_year_from" SIZE=15> 에서
         <INPUT TYPE="text" NAME="production_year_to" SIZE=15> 까지
<TR> <TD ALIGN=CENTER> 작품유형
    <TD> <INPUT TYPE="checkbox" NAME="type_info" VALUE="한국화">한국화
         <INPUT TYPE="checkbox" NAME="type_info" VALUE="양화">양화
         .....
<TR> <TD ALIGN=CENTER> 재료 및 기법
    <TD> <INPUT TYPE="text" NAME="material_tech" SIZE=35>
<TR> <TD ALIGN=CENTER> 색인어
    <TD> <TEXTAREA NAME="ir_query" ROWS=2 COLS=35></TEXTAREA>
</TABLE></CENTER>
</FORM>
```

〈그림 14〉 작품 검색 화면



ACTION은 /uw/dispatcher/ artwork_query 이다. 즉, 이 FORM이 수행 요구되면, CGI를 통하여 UniWeb/디스패처가 호출되고, PATH_INFO에는 artwork_query가 전달된다. UniWeb/디스패처는 QUERY_STRING 환경변수로 전달된 사용자 질의값을 artwork_query 응용 프로세스에게 전달한다.

〈그림 15〉는 artwork_query 응용 프로그램의 구현을 일부 보인 것이다. 먼저, ESQL/C를 이용하여 UniSQL/X에 연결한 후, 서비스가 종료될 때까지 사용자 요구를 받아 질의를 처리하여 결과를 복귀하는 것을 반복한다. 주의할 것은 데이타베이스 연결을 한 번만 한다는 것이다. 함수 process_request()는 사용자가 입력한 값을 이용하여 데이타베이스 질의를 수행하고, 그 결과를 stdout에 출력한다 (stdout은 Unix Stream socket을 통하여 자동적으로 UniWeb/디스패처에 전달된다).

〈그림 16〉과 〈그림 17〉은 artwork_query 응용 프로그램의 수행 결과로 복귀된 HTML과 이를 웹 브라우져 화면에 출력한 예제를 보인 것이다. 〈그림 17〉에서 보듯이 작품명과 작가명은 앵

커로 지원된다. 예를 들면, 작품명 모나리자를 선택하게 되면 〈그림 16〉에서 기술된 바와 같이 artwork_query 응용 프로세스에 oid=1290:6:0이 전달되고, artwork_query는 이를 이용하여 작품 모나리자의 상세한 정보를 출력하게 된다.

UniSQL/X는 객체관계 DBMS로서, 데이타베이스 저장된 모든 객체에 대하여 유일한 객체식별자를 부여하고 있다. 질의를 통하여 해당 객체, 혹은 이와 연관된 다른 객체의 식별자를 알 수 있어서 위의 예와 같은 HTML 문서 출력이 가능하다. 객체식별자는 객체가 저장된 물리적은 주소로서, 매우 신속한 접근을 지원한다. 즉, 위의 예에서와 같이 한 객체에 관련된 다른 객체를 하이퍼미디어 방식으로 검색하는 경우에는 객체식별자를 앵커로 사용하여 매우 빠른 검색을 지원할 수 있다.

〈그림 15〉 artwork_query 응용 프로그램

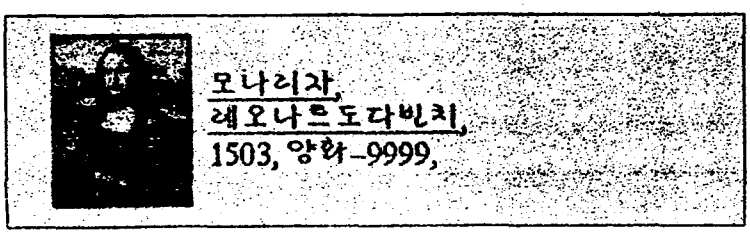
```
void main(int argc, char **argv) {
  uci_startup(argv[0]);
  EXEC SQLX CONNECT 'gallery_db';
  uw_init_env("artwork_query");          /* initialize UniWeb application env */
  for (;;) {
    uw_connect_client();                 /* wait for a client request */
    uw_cgi_init_env();                   /* setup CGI environment */
    process_request();                   /* process the request */
    uw_cgi_final_env();                  /* reset CGI environment */
    uw_disconnect_client();              /* disconnect the client */
  }
}

void process_request(void) {
  ......
  fe_name = uw_cgi_find_form_entry("name"); /* identify the form values */
  ....
  construct the SQL/X statement with form values;
  process the SQL/X statement;
  display the results;
}
```

〈그림 16〉 artwork_query 수행 결과 HTML

```
<TABLE>
<TR>
<TD><A HREF="/gallery/media_data/sample/image.jpg">
    <IMG SRC="/gallery/media_data/sample/image_8.gif" ALIGN=LEFT></A>
<TD><A HREF="/uw/dispatcher/artwork_query?oid=1290:6:0">모나리자</A>,<BR>
    <A HREF="/uw/dispatcher/artist_query?oid=1541:3:0">레오나르도다빈치</A>,<BR>
    1503, 앙화-9999,
</TABLE>
```

〈그림 17〉 artwork_query 수행 결과 화면



4. 맺음말

웹은 대규모 멀티미디어 정보 서비스 시스템으로서 많은 우수성을 갖는 반면, 대량의 자료를 가진 데이타베이스 서비스를 개발하는 데에는 문제가 많다. 반면, 기존의 데이타베이스 시스템은 방대한 데이타베이스 관리 기능과 응용 개발 도구 등의 우수성을 갖고 있으나 폼의 관리 방식과 표준 부재 등의 이유로 대규모 서비스 시스템을 구축하기에는 적합하지 않다. 따라서, 대규모 멀티미디어 데이타베이스 서비스 시스템은 웹와 데이타베이스 시스템 기술의 상호 보완적인 통합이 필요하다. 데이타베이스 통로는 이러한 통합의 핵심적인 소프트웨어라 할 수 있다.

본 논문에는 데이타베이스 통로의 구조에 대한 분류체계를 제안하였다. 데이타베이스 통로의 구조는, 데이타베이스를 접속하는 프로그램이 웹 서버 쪽에 위치하는 "서버쪽 확장"과, 웹 브라우저 쪽에 위치하는 "클라이언트쪽 확장"으로 크게 나누어질 수 있다. 서버쪽 확장은 "CGI 이용", "확장 API", 그리고 "전용 서버" 방식으로 나누어질 수 있고, 이중 CGI 이용은 "CGI 실행화일"과 "응용 서버" 방식으로 다시 분류된다. 클라이언트쪽 확장은 "외부뷰어" 방식과, "브라우저 확장" 방식으로 나눌 수 있다. 각 구조별 사례는 [12]에서 찾을 수 있다.많은 상용 제품이 CGI 실행화일 구조를 지원하고 있으나, 이는 성능 면에서 DBMS의 특징을 충분히 고려한 구조로 볼 수 없고, 특히, 대규모 서비스에서 시스템 자원 부족 및 성능 저하 문제를 발생시킬 것이다. 단기적으로는 현재의 표준 기술만 이용하면서 성능 문제를 최소화한 CGI 응용 서버 방식이, 그리고 장기적으로는 HTML의 실연 기능 제약을 극복하는 브라우저 확장 방식이 주로 사용될 것으로 기대된다. 특히, Java용 데이타베이스 응용 개발 환경은 이 방식을 구현하는 데 큰 역할을 담당할 것이다.

본 논문에서는 UniSQL/X용 데이타베이스 통로인 UniWeb의 설계 및 구현을 기술하였다. UniWeb 데이타베이스 통로는 UniSQL/X가 지원하고 있는 반복질의 수행 성능 향상 및 웍스페이스 관리 기능의 효과를 최대한 활용함과 동시에 웹의 표준기술을 그대로 이용할 수 있도록 하기 위해 CGI 응용 서버 구조로 설계되었다. 우리는 최근에 UniWeb 1.0을 적용하여 초고속공공응용서비스 중 문화재정보, 전자박물관, 전자미술관 멀티미디어 데이타베이스 서비스 시스템을 개발한 바 있다 [13]. 앞으로, 우리는 UniWeb 1.0을 확장하여 응용 개발 생산성을 높이고, 여러 가지 데이타베이스 통로 구조의 성능을 보다 체계적으로 분석할 계획이다.

〈참고자료〉

[1] T. Berners-Lee, Hypertext Transfer Protocol - HTTP/1.0, Internet draft, Dec. 20, 1994. URL: ftp://ds.internic.net/internet-drafts/draft-fielding-http-spec-01.ps.

[2] T. Berners-Lee, Uniform Resource Locators, Internet RFC 1738, Dec. 20, 1994. URL: ftp://ds.internic.net/rfc/rfc1738.txt.

[3] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," Electronic Networking: Research, Applications and Policy, Vol. 1, No. 2, Westport CT, 1992, pp. 52-58. URL: ftp://ftp.w3.org/pub/www/doc/ENRAP_9202.ps.

[4] T. Berners-Lee and D. Connolly, Hypertext Markup Language Specification - 2.0, Internet draft, Feb. 8, 1995. URL: ftp://ds.internic.net/internet-drafts/draft-ietf-html-spec-01.txt.

[5] K. Hughes, Entering the World-Wide Web: A Guide to Cyberspace, Honolulu Community College, Sep. 1993. URL: ftp://ftp.w3.org/pub/www/doc/hughes-guide.ps.

[6] W. Kim, "Object-Oriented Database Systems: Promises, Reality, and Future", Modern Database Systems: The Object Model, Interoperability, and Beyond (Won Kim, ed.), 1995, pp. 255-280.

[7] L. Latham, "Client/Server Computing: Strategic Directions, Tactical Solutions," InSide Gartner Group This Week, Vol. X, No. 20, Gartner Group, May 18, 1994, pp. 1-5.

[8] A. Luotonen and T. Berners-Lee, CERN httpd Reference Manual - A Guide to a World-Wide Web Hypertext Daemon, CERN, May 1994.

[9] R. Schulte, "Two-Tier vs. Three-Tier Trade-Offs," SMS: K-401-1564, SMS Research Note, Gartner Group, Dec. 21, 1994.

[10] R. Schulte, "Three-Tier Software Architecture in Perspective," SMS: K-401-1565, SMS Research Note, Gartner Group, Dec. 21, 1994.

[11] URL: http://grigg.chungnam.ac.kr/~uniweb.

[12] URL: http://grigg.chungnam.ac.kr/~uniweb/documents/www_dbms.html.

[13] URL: http://grigg.chungnam.ac.kr/~kcaf.

[14] URL: http://www.ndev.com/ndc.

[15] URL: http://www.allaire.com

[16] URL: http://gdbdoc.org/letovsky/genera/genera.html.

[17] URL: http://www.ncsa.uiu.edu/SDG/People/jason/pub/gsql/starthere.html.

[18] URL: http://arch-http.hp.eso.org/bfrasmus/wdb/wdb.html.

[19] URL:http://www.02tech.fr.

[20] URL://www.progress.com.

[21] URL://www.nttdata.jp/products_services/network/interserv_e.html.

[22] URL: http://www.navisoft.com.