2004년 한국산학기술학회 춘계 학술발표논문집

3 Tier Architecture 시스템 구조연구

최성, 김승찬, 유정근, 한정란*
남서울대학교 컴퓨터학과, 협성대학교 경영정보학과*

A Study on the 3Tier Architecture Structure System

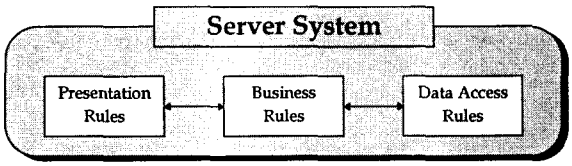
Choi Sung, Yu Jeoung Geun, Kim Seoung Chan, Han Jung Lan*
Dept. of Computer Science, Namseoul University
Dept. of Management Information System, Hyupsung University*

요 약

정보기술 패러다임의 주요 흐름을 살펴보면 오랜 기간동안 주도했던 메인 프레임 위주의 시스템이 90년대 들어서 클라이언트/서버 시스템이 주도하는 환경으로 전환되었으며, 90년대 후반 들어서는 인터넷의 활용증대로 네트웍 컴퓨팅등에 대한 관심이 증대하고 있는 현실이다. 3-tier 구조는 이 시대서 요구하는 시스템 아키텍쳐에 적합하며 발전해 나가 서비스 확장성(scalability)과 안정성, 그리고 효율면에서 유리하므로 대용량 서비스에 적합하다. 본 논문에서는 이제 앞으로 주도할 3-Tier 아키텍처를 제안하였다.

1. 서론

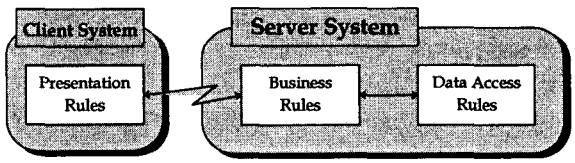
1.1 3-Tier 구조의 시스템 구조의 발전 과정



<그림 1 : 1-Tier 시스템 구조>

1-Tier, 이 방식은 한 개의 시스템 내에 서버와 클라이언트 프로그램이 공존하는 논리적인 클라이언트/서버 시스템이며, 기존의 호스트 기반 시스템들이 이에 해당된다. 이 방식은 시스템 구축에 많은 요소 기술이 필요치 않아, 개발 시간이 적으며, 한 개의 시스템에서 모든 관리를 하므로, 관리와 보안이 쉽다. 하지만, 모든 구성 요소가 한 시스템에서 표현되므로, 서버가 비대해지는 현상이 발생하며, 사용자 수가 많아짐에 따라 서버 부하가 기하 급수적으로 커지게 되고, 새로운 시스템의 도입과 시스템의 확장 시에 쉽게 확장하기가 어려운 단점을 가지고 있다.

이러한 단점을 보안하고자 한 것이 2-tier 클라이언트/서버 구조 시스템이다.



<그림2 : 2-Tier 시스템 구조>

2 계층 클라이언트/서버 구조는 네트웍의 발전과 PC와 같은 지능형 터미널이 도입되면서, 기존의 더미 터미널이 할 수 없었던 기능을 클라이언트로 전이한 방식을 말한다.이 방식은 클라이언트와 서버가 물리적으로 서로 독립된 시스템에 존재하는 형태로 구성되었다. 시스템의 구성이 2 계층 클라이언트/서버 방식으로 발전함으로서, 기존에 서버의 부하가 상당부분 줄어들게 되었으며, 클라이언트가 대량의 자료를 접근하여 가공할 수 있게 됨으로서, 더욱 정교한 자료 분석과 사용자로의 정보 제공이 가능해 졌다.시스템의 크기가 증가하고, 동시에 접근하는 사용자의 수가 증가함에 따라 더 이상 두 계층 클라이언트/서버 방식으로 이를 서비스하기는 힘든 상황이 되었다. 이러한 상황속에 3-tier 구조 시스템이 등장하였다.

2. 본론

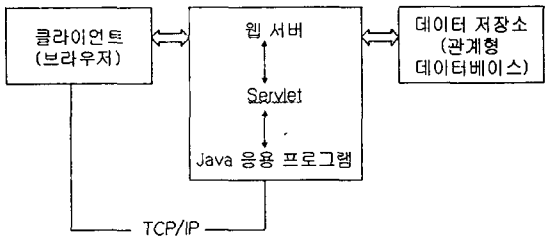
2.1 출현배경

급속도로 발전하는 Client/Server 개발환경의 변화에 따른 개발자들 어려움 과 환경 변화에 부응한 다양한 개발도구 요구, Fat 클라이언트 환경에서 Thin 클라이언트 환경으로의 변화로 볼 수 있다.

소프트웨어 개발 Paradigm의 3-Tier체제로의 변화

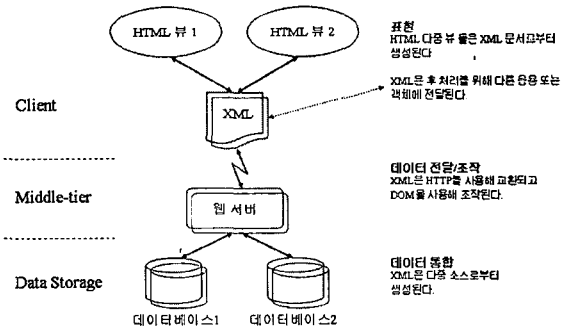
- Presentation Logic, Business Logic, Data Access Logic 의 분리 개발
- 개발자 특성에 따른 표준화 및 Documentation 시간소요
- 업무표현의 분리에 따른 Resource Management-Resource 의 재사용 여부

2.2 3-Tier 프로그램의 구축도구인 JAVA



<그림 3 : 3-tier 구조의 Java>

Java와 XML은 3-tier 프로그램 구축시 최상의 도구를 제공 할수 있으며 Java 는 플랫폼 독립적인 코드 이며 XML 은 플랫폼 독립적인 데이터이다.



<그림 4 : 종합시스템 구조도>

Java를 사용한 Middle-tier 구축 방법으로는 위의 그림과 같이 Servlet 그리고 JSP(Java Server Page), JavaBeans 가 있다.Servlet은 작은 Java 프로그램이며, applet과 유사하다. JSP는 JSP 요소(element)를 포함한 HTML이나 XML 페이지이며, JSP 요소는 JSP 태그로 구분한다. JavaBeans는 JSP 응용구조를 component 형태로 만드는 것이다. JSP 페이지는 servlet 기능을 갖추고, Java 코드와 함께 사용할 수 있는 XML 페이지 쪽으로 이동한다. 그리고, XML 페이지에 Java 코드를 함께 사용하는 다른 방법으로는 XSLT가 있다. 또한 JSP+XML과 XML+XSLT+(Java 코드) 사이에 겹치는 중요한 영역이 생길 것으로 예상되고 있다.

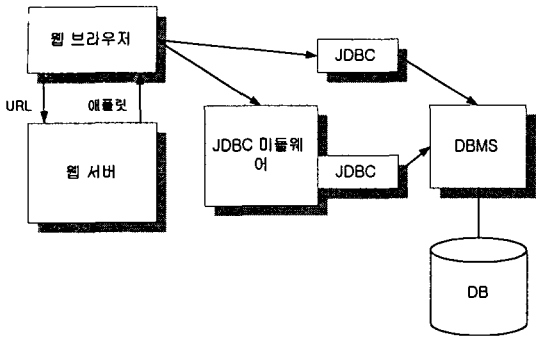
2.3 JDBC에서의 3-Tier 모델

JDBC는 SQL 문장을 실행시키기 위한 자바 APL이며, 자바로 쓰여진 클래스와 인터페이스의 집합으로 구성된다. JDBC는 응용프로그램과 데이터베이스 개발자들을 위한표준 API를 제공하고, 순수 자바 API를 사용하여 데이터베이스 애플리케이션을 작성할 수 있게 해 준다. JDBC를 사용하면 SQL 문장을 데이터베이스로의 전달이 가능하다. JDBC API는 데이터베이스 접근에 대해서 2-tier와 3-tier 모델 모두를 지원한다. 2-tier 모델에서는 자바 애플릿 또는 어플리케이션이 데이터베이스와 직접 통신한다. 이것은 접근한 특정 DBMS와 통신할 수 있는 JDBC 드라이버가 필요하다. 사용자의 SQL문을 데이터베이스로 전달하고, 그 SQL 문의 결과를 사용자에 되돌려 준다.

데이터베이스는 사용자가 네트워크를 통해 연결한 다른 머신상에 위치할 것이며 이것은 사용자의 머신이 클라이언트, 그리고 데이터베이스를 가지고 있는 머신이 서버로써 클라이언트/서버 구성에 속한다.

네트워크는 인트라넷이 될 수 도 있고, 인터넷이 될 수도 있다. 3-tier 모델에서는 명령들을 서비스의 "미들티어(middle-tier)"로 보내고, 그리고 나서 SQL문을 데이터베이스로 보낸다. 데이터베이스는 SQL문을 처리하고 결과를 미들티어로 되돌려주며, 그런다음 그것을 사용자에게 전송한다. MIS 관리자는 미들티어가 기업 데이터로 만들 수 있는 일종의 최신정보와 억세스에 대한 관리를 유지하는 것이 가능하기 때문에 3-tier 모델에 매우 매력을 느낀다. 다른 장점은 미들티어가 있을때에는, 사용자는 미들티어에 의해 해석되어지는 사용하기 쉬운 하이레벨 API를 가지고 적절한 로우레벨 호출을 할 수 있다는 것이다. 결국 많은 경우에서 3-tier 구조는 성능적 이점을 제공할 수 있다. 지금까지 미들티어는 전형적으로 C나 C++과 같은 언어로 작성되어 졌고, 이것은 빠른 성능을 제공했다. 그러나 자바 바이트코드를 유효한 특정 머신 코드로 변환하는 최적화 컴파일러의 도입으로, 미들티어를 자바로 구현하는 것이 가능해졌다. 자바의 견고성, 멀티쓰레딩 그리고 보안 형태들을 이용하는 것이 가능하므로 자바를 이용하는 것은 큰 프러스 요인이 된다. JDBC는 자바 미들티어로부터 데이터베이스

접근을 허용하도록 하는 것이 중요하다.



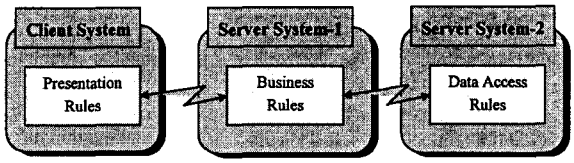
<그림 4 : 서버 시스템구축도>

2.3 3-Tier Architecture

이 방식은 2 계층 클라이언트/서버 방식에서 클라이언트가 수행했던 데이터 가공 작업을 중간층에 존재하는 서버 시스템으로 전이하였다. 중간층에 있는 서버는 클라이언트의 요구에 따라 데이터 관리 서버와 연동하여 클라이언트의 요구를 처리하여 결과를 전달한다. 이 방식에서 작업 수행을 위한 대부분의 로직은 중간 층 서버가 유지하게 된다. 이 방식은 클라이언트와 서버가 시스템이 수행해야 하는 기능을 적당하게 분산하여 가지고 있으므로, 어느 한쪽으로 기능이 비대해지는 현상을 방지할 수 있으며, 서버 측의 확장성이 용이하여, 사용자의 수가 급증하더라도 일정한 시간에 응답을 얻을 수 있어, Mission-Critical 응용에 적합한 특징을 가진다. 서버 측의 레벨이 증가함에 따라 보안과 시스템 관리가 수월해지는 장점도 가지고 있다. 하지만, 이 방식은 초기 설치비용이 크며, 적정 수의 사용자가 사용하지 않으면, 2 계층 방식보다 성능이 떨어질 수 있다. 또한, 한 개의 프로그램을 수정하기 위해 서로 다른 여러 플랫폼에서의 응용 프로그래밍 지식을 가지고 있어야 하는 어려운 점도 있다. 그러므로, 2 계층 방식에서 3 계층 방식으로의 전이는 시스템을 사용하는 환경에 대한 많은 고려가 수반되어야 한다.

3 계층 클라이언트/서버 시스템은 기존의 2 계층 클라이언트/서버 시스템의 문제점을 해결하기 위해 새로 도입된 방식으로, 사용자와 인터페이스 하기 위한 화면 운용(표현 규칙의 운용 : Presentation Rule)을 클라이언트 시스템이 담당하고, 업무 규칙(Business Rule)의 구현과 자료의 접근 규칙(Data Access Rule)의 운용을 서버 시스템에서 담당하는 방법으로 서버와 클라이언트가 작업을 분담하는 형태로 시스템을 운용하는 방식이다.

이 방식에서 서버 시스템은 업무 규칙을 처리하는 부분과 자료 관리를 위한 부분을 물리적으로 나누어 구성될 수도 있다.



<그림 5 : 3-Tier 시스템 구조>

또한 미들티어(3-tier)는 개방시스템 기반의 분산환경의 가장 큰 특징은 업무의 분산처리로 트랜잭션과 애플리케이션이 분산된다는 것이다. 따라서 이기종 환경에서의 분산된 트랜잭션과 애플리케이션을 효율적으로 관리하고 운용하기 위한 미들웨어의 역할과 중요성은 매우 크다. 미들웨어는 높은 성능과 신뢰성을 갖춘 분산 애플리케이션을 구축할 때 사용하는 솔루션으로 이기종 및 분산 환경 하에서 확장성 있는 3-tier 클라이언트/서버 애플리케이션을 개발하는 프레임웍을 제공한다. 소프트웨어 업계에서 떠오르는 핵심 이슈중의 하나가 컴포넌트 기반의 솔루션이다. 미들웨어 역시 컴포넌트를 기반으로 개발의 효율성을 제고하는데 주력하고 있다. 뿐만 아니라 컴포넌트 화와 더불어 논의되고 있는 이슈가 객체지향기술의 통합이다. 사용자들은 객체지향기술의 장점을 충분히 인식하고 있음에도 불구하고 아직까지는 기간업무 애플리케이션에 적용하는 것을 선뜻 결정하지 못하고 있다. 이는 기존의 대규모 클라이언트/서버 환경을 보호하고 기존의 애플리케이션들을 수용할 수 있는 충분히 신뢰할 수 있는 애플리케이션 소프트웨어 플랫폼이 부족하기 때문이라고 생각하기 때문이다. 그런데 최근 객체지향기술을 적용한 미들웨어가 속속 등장하고 있다. 사용자들이 그동안 우려했던 기존 정보시스템환경을 보호함은 기간업무 애플리케이션을 운영하고 있는 사용자들이 안전하게 객체들을 개발하고 구현함으로써 개발기간과 비용을 낮추는 등의 장점을 제공할 수 있게 됐기 때문이다.이제 사용자들은 이기종의 정보시스템 환경 하에서 새로운 애플리케이션 개발뿐만 아니라 관리, 전개 등에 있어서 분산 객체기반 솔루션을 이용함으로써 많은 효과를 얻을 수 있게 된다는 것이다.

3. 결론

계속되는 웹의 발전으로 인하여 사용자들은 웹 브라

우저를 통해 서비스를 요구하는 경향이 많아졌고, 서비스 제공자들은 이를 처리하기 위해 웹 서버가 사용자의 요구를 처리하는 응용 서버(중간 층 서버) 기능을 하도록 노력을 기울이고 있다. 또한, 웹 서버와 기존의 미들웨어를 통합하는 노력도 보인다. 앞으로는 미들웨어의 성능 계선을 위해 클라이언트의 수의 변동에 유기적으로 변하는 미들웨어를 생성하고, 미들웨어(3-tier)를 통하여 아무런 제약도 받지 않고 원하는 데이터를 획득할 수 있으리라 기대된다.

[참고문헌]

[1] 박재진, 손진국 역, Java로 처리하는 JDBC 데이터 베이스

[2] 이상우 저 Web Application for EJB with JSP , 2001

[3] 3 Tier 클라이언트/서버 시스템 구축 기법 mbs 정보화 연구소

[4] 3-Tier Client/Server At Work, Revised Edition

[5] Jeri Edwards 저, 3-Tier Client/Server at Work