서버 아키텍쳐 입문

서버의 유래와 다양한 사례 소개

Version 1.0 2015/02/22

Sunny Kwak

sunykwak@daum.net

Agenda

컴퓨터 역사 이해

클라이언트/서버 이전

컴퓨터 역사 상식 #1

- 최초의 컴퓨터 애니악(ENIAC)은 서버인가?
  - No. '서버'라는 말은 1980년대 후반부터 사용하게 된 용어.
- 1980년대 이전의 컴퓨터 종류는 어떤게 있었는가?
  - 메인프레임, 미니컴퓨터, 워크스테이션 등이 있었음.
- 메인프레임(Mainframe)이란?
  - 1960년대, 기업과 정부기관에서 인구조사(Census), 각종 통계, 미션 크리티컬(mission critical) 업무, 대규모 데이터 처리에 활용할 수 있는 컴퓨터를 IBM 등의 컴퓨터 회사에서 생산했음.
  - 메인프레임이라는 용어의 유래는 CPU와 메모리를 감싸고 있는 금속성 캐비넷(cabinet)을 메인 프레임(main frame)이라고 부른데서 유래함.
  - 메인프레임의 구조(설계)는 60년대에 확립된 이후, 계속 진화 중.
  - 메임프레임의 특징은 컴퓨터 하드웨어와 운영체제 및 소프트웨어를 단일 기업에서 모두 설계 및 개발하고, 타 회사 제품과의 호환성은 없는 폐쇄 구조임. 단, 단일 컴퓨터의 성능이 매우 높기 때문에 여전히 고성능 컴퓨터를 필요로 하는 기업들에서 사용중임.
- 미니컴퓨터(Minicomputer), 워크스테이션(Workstation)
  - 미니컴퓨터는 메인프레임 보다 적은 성능을 내지만, 메인프레임을 도입하기에는 예산이 부족한 작은 기업을 위한 컴퓨터.
  - 워크스테이션은 주로 개인이나 소규모 팀에서 사용할 수 있는 고성능 컴퓨터를 의미하나, PC 성능의 발전으로 근래에는 워크스테이션급이라는 용어는 거의 사용 안함.

컴퓨터 역사 상식 #2

- 개인용 컴퓨터(Personal Computer)의 도래
  - 1981년 IBM 사에서 개인용 컴퓨터 'IBM 5150', 1984년 애플 사에서 매킨토시(?)를 출시
  - IBM 5150은 최초의 개인용 컴퓨터가 아니었으나, 컴퓨터 하드웨어 설계를 공개함으로써 다양한 기업들에서 개인용 컴퓨터를 설계할 수 있게 하고, 시장의 확산과 생산 단가 인하를 유도했음.

- 유닉스(UNIX) 운영체제
  - 1970년대 초반 벨 연구소 직원인 켄 톰슨, 데니스 리치, 더글러스 매클로리 등이 처음 개발.
  - 다양한 시스템 간에 이식할 수 있다는 잇점과 C 언어의 간결함 덕분에 기존 메인프레임 기반의 대형 컴퓨터 회사에 대항하는 기업들이 새로운 운영체제 기반의 컴퓨터를 출시함.

- 컴퓨터 다운사이징(computer downsizing)
  - 많은 사무실에서 PC를 활용해 문서 작성 및 간단한 처리 – 회계 장부를 입력하는 등 – 를 하게 되고, 점점 높아지는 PC를 이용해 메인프레임(혹은 미니컴퓨터)의 기능을 대체하자는 아이디어가 생겨남.
  - PC가 서버 기능의 일부를 수행하도록 하고, 서버는 데이터 저장소 역할을 맡겨 전체 시스템 도입 비용을 절감하자는 캠페인이 시작됨. (다운사이징은 경영 혁신 용어로도 사용되었음.)
  - 서구에서는 1980년대 후반, 한국에서는 1990년대 초중반부터 다운사이징이 도입되었음.

- 클라이언트/서버
  - 개인용 컴퓨터와 중앙 컴퓨터를 연결하는 네트워크 구조
  - 다운사이징 운동과 함께 클라이언트/서버 구조가 나온 이후, '서버 컴퓨터'라는 용어가 사용됨.

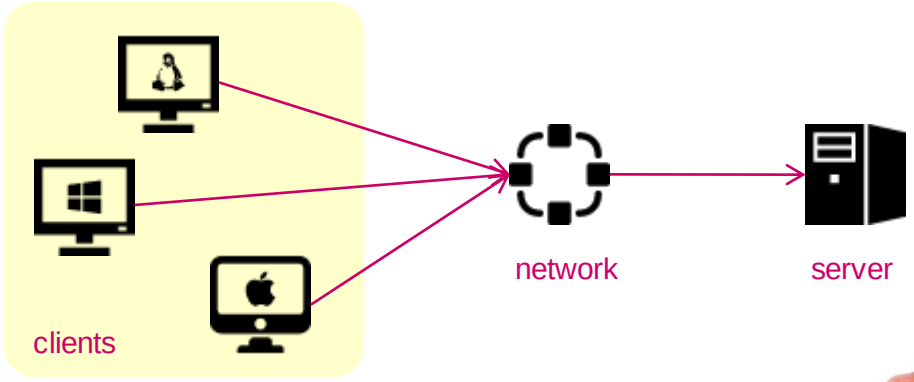
2계층 부터 N 계층까지...

클라이언트/서버 구조

클라이언트/서버 구조

- **Client/Server Architecture**
  - 클라이언트 (일반적으로 GUI를 사용하는 어플리케이션)를 서버에서 분리하는 **네트워크 구조**이다. 각각의 클라이언트 소프트웨어 인스턴스는 서버에 요청을 전송할 수 있다.
  - 하나의 서버에 복수의 클라이언트가 접속하게 된다. (일대다 관계)

- **서버 유형**
  - 어플리케이션 서버 (게임, 채팅, 메신저, 증권 거래 서버 등)
  - 파일 및 FTP 서버
  - 터미널 서버
  - 메일, DNS 서버



클라이언트/서버 기능

- **서버 기능 (혹은 특성)**
  - 수동적, 서비스 제공자 (Passive, Slave)
  - 클라이언트 요청을 처리하기 위해 대기.
  - 요청(request)을 처리한 후, 결과를 클라이언트에 회신(reply)

- **클라이언트 기능**
  - 능동적, 의뢰자 (Active, Master)
  - 서버가 수행할 수 있는 요청을 전송함.
  - 회신(응답)이 반환될 때까지 기다림

- **널리 알려진 클라이언트**
  - 가장 보편적인 클라이언트는 인터넷을 통해 웹 서버와 통신하여 웹 페이지를 다운로드하고 사용자에게 보여주는 웹 브라우저이다.

N-계층 구조

- **2 계층 구조 (2-tier architecture)**
  - 클라이언트와 서버 등 2개의 노드(node)로 구성된 구조(architecture)를 2계층 구조라고 부른다.
  - 2 계층 구조에서 서버는 단지 데이터를 저장하는 역할만을 수행하며, 클라이언트가 모든 처리(processing)을 담당한다.

- **2계층 구조의 한계(문제)**
  - 클라이언트(PC 등)의 상대적인 성능이 향상되면서 다양한 처리를 클라이언트로 이전할 수 있으나, 데이터의 무결성(integrity)을 유지(관리)하기가 어렵다.
  - 비즈니스 로직(business logic)을 클라이언트에 두기 어려운 경우도 있다. 예를 들어, 사용자 간의 메시지를 주고 받아야 할 경우 서버는 데이터를 저장하는 역할만 수행하기 때문에 클라이언트 간에 직접 통신을 해야 한다.

- **N 계층 구조 (N-tier architecture)**
  - 2 계층 구조의 한계를 극복하기 위해, 3개 이상의 노드를 네트워크 상에서 구성하는 방식이 N 계층 구조이다. N 계층은 3개, 4개 혹은 더 많은 노드로 구성된다.
  - N 계층 구조가 2 계층 구조를 완전히 대체하는 하는 것은 아니다. FTP, Telnet 서비스 등은 여전히 2계층 구조로 동작한다.

3계층 구조

- **3계층 어플리케이션**
  - 정보, 중간, 클라이언트 등 3개의 계층으로 구성된다.

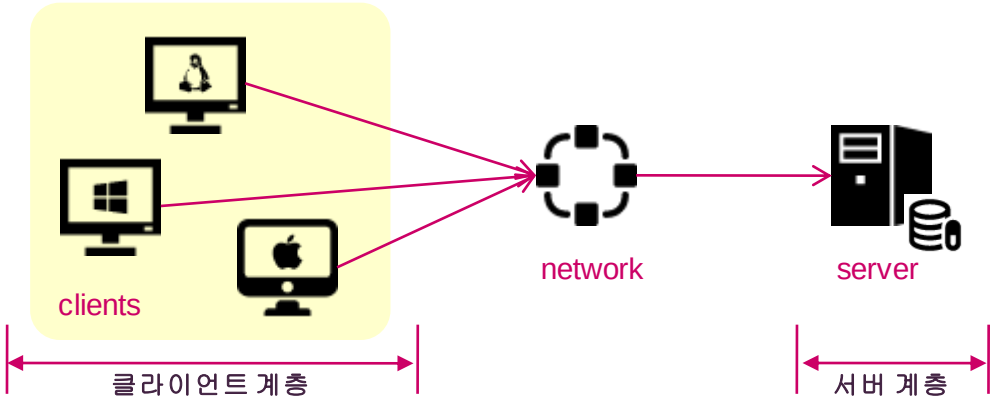
- **정보 계층 (Information tier)**
  - 데이터 계층(data tier) 혹은 최하위 계층(bottom tier)이라 부른다.
  - 어플리케이션을 위한 데이터를 관리한다.
  - 일반적으로 관계형 데이터베이스(Relational Database)를 이용해 데이터를 저장한다.

- **중간 계층 (Middle tier)**
  - 어플케이션 계층(applicatoin tier)으로 부르기도 한다.
  - 비즈니스 로직(business logic) 및 프리젠테이션 로직(presentation logic)을 구현한다.
  - 어플리케이션 클라이언트와 데이터 간의 상호작용을 제어한다.
  - 정보 계층의 데이터와 어플리케이션 클라이언트 간의 매개자(intermediary) 역할을 수행한다.

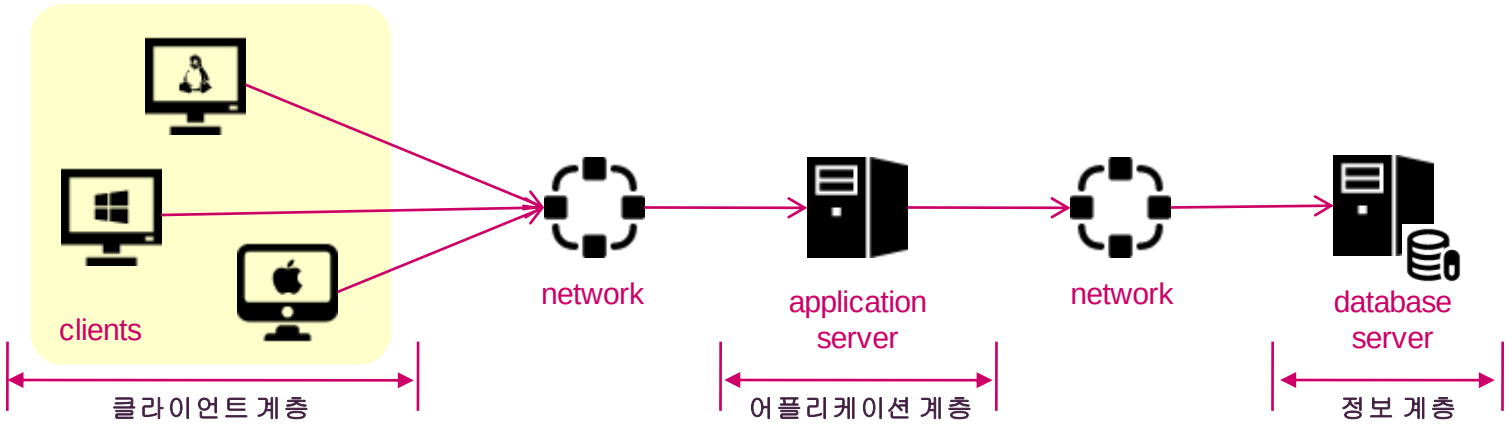
- **클라이언트 계층 (Client tier)**
  - 최상위(top) 계층으로 부르기도 한다.
  - 어플리케이션의 사용자 인터페이스 역할을 수행한다.
  - 중간 계층과 상호작용을 통해 요청을 전달하고 정보 계층에서 데이터를 조회한다.

2 계층 vs 3 계층

2 계층 구조 (2-tier architecture)



3 계층 구조 (3-tier architecture)



단순 2 계층부터, 클라우드 환경 까지

서버 아키텍쳐 사례

인터넷 2계층 구조

- **인터넷 기반의 2계층 서비스**
  - FTP (File Transfer Protocol), e-mail, DNS(Domain Name Server) 등 인터넷 상에서 널리 사용되는 서비스들은 2 계층 구조로 동작한다.
  - 초기의 인터넷 서비스 또한 2계층 구조로 동작했다. (단순 HTML 문서 전송)

- **클라이언트 및 서버의 역할**
  - 서버는 데이터를 저장, 클라이언트는 정보를 처리하고 서버로 전송하는 역할 수행

- **동작 방식**
  - 서버 컴퓨터 상에서 클라이언의 요청을 처리하는 프로세스가 실행된다.
    (서버 상의 소프트웨어를 데몬(daemon)이라고 부른다.)
  - 클라이언트 컴퓨터 상에는 서버 프로세스와 통신하는 전용 클라이언트 프로그램이 설치된다.
    FTP는 파일질라(filezilla), e-mail은 아웃룩(outlook) 등의 프로그램이 유명하다.
  - 클라이언트 프로그램은 서버 프로세스와 TCP/IP 네트워크를 통해 통신하며, 대게 전용 프로토콜을 이용해 데이터를 송수신한다. (FTP 프로토콜, SMTP, POP3 프로토콜 등)

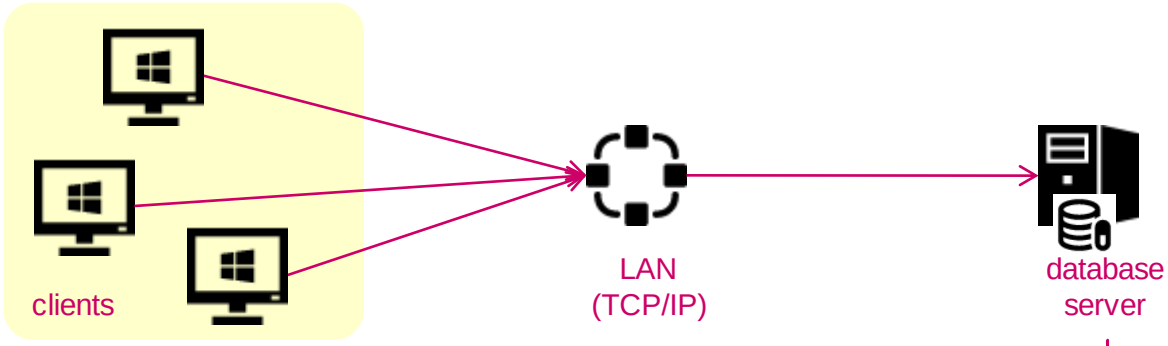
인트라넷 2계층 구조

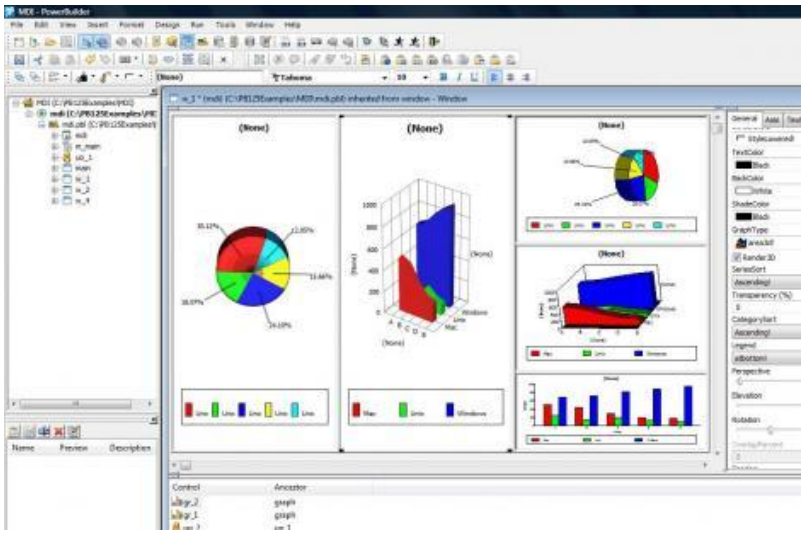
- **인트라넷 기반의 2계층 구조**
  - 인트라넷(intra-net)은 공공 및 사기업 등 각종 기관에서 조직 내의 컴퓨터들을 연결한 네트워크를 말한다. 초기 클라이언트/서버 구조는 기업 내의 PC와 서버 컴퓨터를 인트라넷으로 연결해 2계층으로 설계 및 운영하였다.

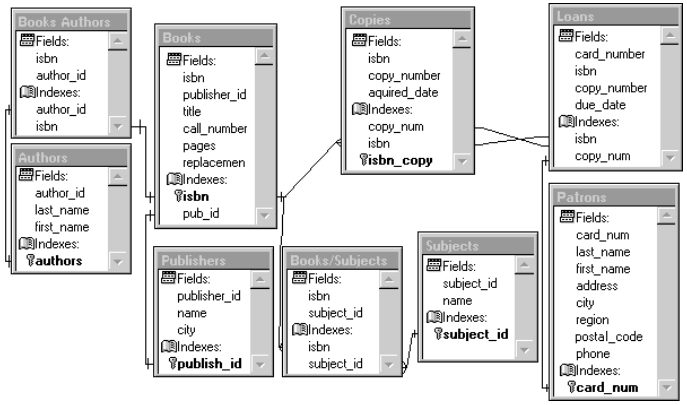
- **GUI 기반의 클라이언트**
  - 기업(및 공공기관) 업무용 클라이언트는 FTP, 이메일과 다르게 복잡한 기능 (데이터 목록 조회, 검색, 각종 계산, 자료 입력 등, 챠트 및 그래프 출력)을 수행해야 하므로, GUI (Graphic User Interface)를 구현할 수 있는 개발 도구(tool)를 이용해 작성하였다.
  - Delphi (Boland), Power Builder (Sybase), Visual Basic (Microsoft) 등의 개발 도구가 사용되었으며, 각기 Object Pascal, Powerscript, Visual Basic 언어를 이용해 코딩하였다.
  - GUI 개발이 손쉽도록 마우스 조작 만으로 손쉽게 메뉴, 버튼, 그리드(grid, 표 양식), 텍스트 입력 박스 등을 디자인할 수 있다.

- **인트라넷 서버**
  - 초기 2계층 구조에서 클라이언트는 데이터베이스 서버와 직접 통신했다.
  - 관계형 데이터베이스(Relational Database) 제품들이 많이 사용되었고, 오라클(Oracle), 인포믹스(Informix), 사이베이스(Sybase), MS-SQL 등이 있다.

2계층 구조 예시


clients
LAN
(TCP/IP)
database
server




Books Authors
Fields:
isbn
author_id
Indexes:
author_id
isbn
Authors
Fields:
author_id
last_name
first_name
Indexes:
authors
Books
Fields:
isbn
publisher_id
title
call_number
pages
replacemen
Indexes:
isbn
pub_id
Copies
Fields:
isbn
copy_number
aquired_date
Indexes:
copy_num
isbn
isbn_copy
Loans
Fields:
card_number
isbn
copy_number
due_date
Indexes:
isbn
copy_num
Patrons
Fields:
card_num
last_name
first_name
address
city
region
postal_code
phone
Indexes:
card_num
Publishers
Fields:
publisher_id
name
city
Indexes:
publish_id
Books/Subjects
Fields:
isbn
subject_id
Indexes:
isbn
subject_id
Subjects
Fields:
subject_id
name
Indexes:
subject_id

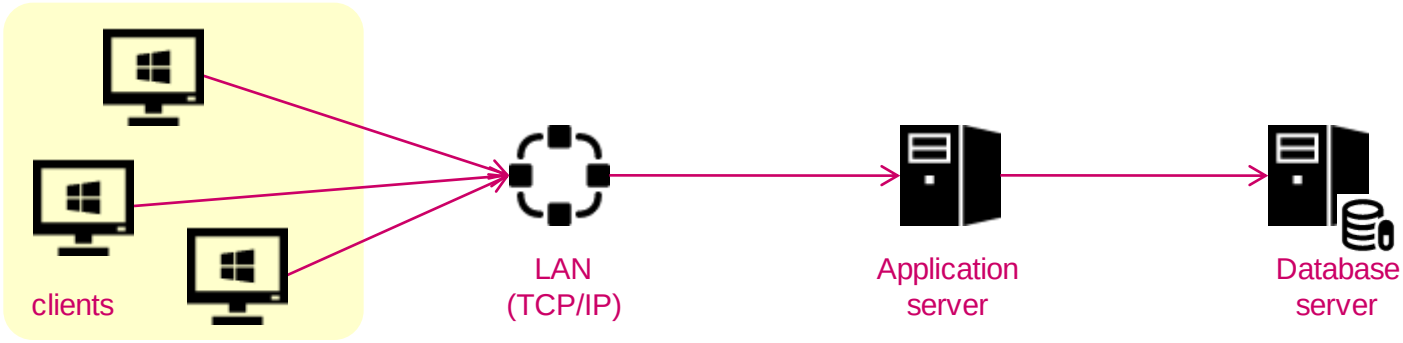
3계층 구조

- **2계층 구조의 한계**
  - 2계층 구조에서는 클라이언트가 사용자 인터페이스(User Interface), 각종 비즈니스 처리 (입력, 데이터의 가공, 편집 및 계산 등)를 담당하고, 서버는 데이터의 저장 및 조회를 담당한다.
  - 데이터의 증가, 조직의 거대화, 업무 복잡성의 증가 등으로 인해 클라이언트에서 처리할 수 없는 비즈니스 처리가 증가한다. (개인용 컴퓨터 처리 속도와 메모리 용량은 서버 보다는 항상 낮다.)

- **1차 대안 : Stored Procedure 활용**
  - 비즈니스 처리(business processing)을 전담하는 함수들을 데이터베이스 서버 내에 포함시키는 것이다. 데이터베이스 서버가 자료의 보관 및 각종 계산 처리를 모두 담당한다.
    (데이터베이스 서버 내에서 동작하는 함수를 stored procedure라고 한다.)
  - 그러나, 데이터베이스 서버 하드웨어의 용량(CPU, memory 등)을 증설하는 것에 물리적인 한계가 있으므로, 데이터베이스 서버의 부하(load)를 줄이기 위한 방법이 모색되기 시작한다.

- **2차 대안 : Application Server 도입**
  - 클라이언트와 데이터베이스 사이에 비즈니스 처리를 전담하는 어플리케이션 서버를 설치하고, Stored Procedure 내에서 구현하던 기능을 어플리케이션 서버에서 실행하는 것이다.
  - 네트워크 구조는 보다 복잡하고 각각의 서버 간에 통신을 수행하는데 따른 추가 비용(시간)이 증가하나, 각각의 서버가 단일 기능을 처리함으로서 병목(bottle-neck) 현상 혹은 데이터베이스 서버의 과부하(overload) 현상을 해결할 수 있다.

3계층 구조 예시



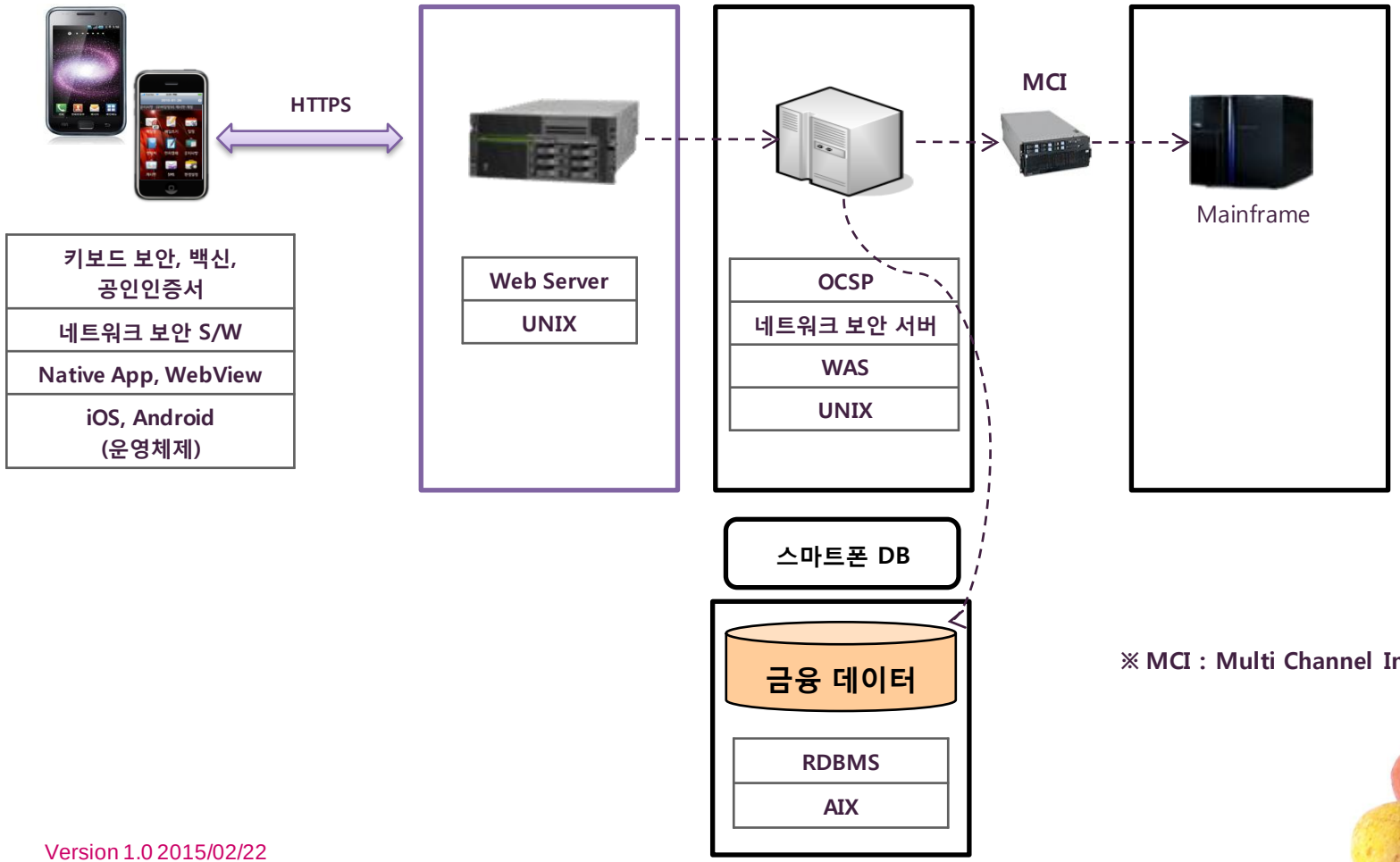
- **2 계층 모델 vs. 3 계층 모델**
  - 2계층 구조에서는 클라이언트가 사용자 인터페이스와 각종 비즈니스 처리 (데이터의 가공, 편집 및 계산 등)를 담당하지만, 3계층 구조에서는 주로 사용자 인터페이스만 처리한다.
  - 2계층 구조에서는 클라이언트가 직접 데이터베이스에 쿼리를 전송하지만, 3계층에서 어플리케이션 서버가 쿼리를 요청하고, 쿼리 결과를 이용해 데이터를 조작한다.
  - 2계층 구조는 모든 클라이언트가 항상 데이터베이스와 연결되기 때문에 클라이언트 수를 늘리는데 한계가 있다. 반면에 3계층 구조에서는 어플리케이션 서버가 데이터베이스 쿼리를 수행하기 때문에 데이터베이스 측면에서 연결(connection) 수가 줄어들어 보다 효율적인 운용이 가능하다.

N계층 구조

- **Web Server & WAS(Web Application Server)**
  - 웹 브라우저로 전송하는 컨텐츠(contents) 중에는 CSS(Cascading Style Sheet), 이미지, 자바 스크립트 파일 등 정적인 요소도 존재하며, 이러한 컨텐츠를 전송하는 기능은 웹 서버(Web Server)가 전담한다.
  - 웹 기반의 환경에서 어플리케이션 서버 역할을 수행하는 서버를 WAS라고 하며, WAS 는 서블릿(Servlet) 등의 자바 컴포넌트를 이용해 비즈니스 로직을 수행한다.
  - 클라이언트와 데이터베이스 사이에 2계층이 존재하므로 전체 4계층으로 구성된다.

- **금융 시스템 구조**
  - 금융 시스템은 '계정계(account system)', '정보계(information)' 등 코어 뱅킹 시스템 (core banking system) 과 채널(channel)이 분리되어 있어서 네크워크 구조가 복잡하다.
  - 채널(channel)은 고객이 금융 시스템을 사용하기 위해 접속하는 다양한 외부 접속 환경을 의미한다. 예를 들자면, 은행 창구의 통합 터미널, 스마트폰 앱, 인터넷 뱅킹, ATM 장비 등이 채널의 유형이다.

스마트폰 뱅킹 구조 사례



※ MCI : Multi Channel Interface

참고 자료

- IBM PC
  - http://ko.wikipedia.org/wiki/IBM_PC
- Mainframe computer
  - http://en.wikipedia.org/wiki/Mainframe_computer
- 이미치 출처
  - http://www.iconarchive.com
  - http://icons8.com