

1주차 교육 세미나

2023. 07. 20
DQA1-3 이윤재

I. 운영체제

II. 개발 언어, 개발 툴, 컴파일러

III. 네트워크, 클라이언트와 서버

IV. 데이터베이스

V. 프레임워크와 라이브러리

VI. 형상관리, 협업 툴

1. 운영체제

- OS 개념과 종류
- 운영체제의 구성
- 스케줄링과 메모리 관리
- Linux와 Unix의 차이

OS 개념과 종류

운영체제란?

운영체제 또는 **오퍼레이팅 시스템**은 사용자의 하드웨어, 시스템 리소스를 제어하고 프로그램에 대한 일반적 서비스를 지원하는 시스템 소프트웨어이다.

OS 개념과 종류

운영체제의 종류

- LINUX
- WINDOWS
- MAC
- TMAXOS

OS 개념과 종류

운영체제의 역할

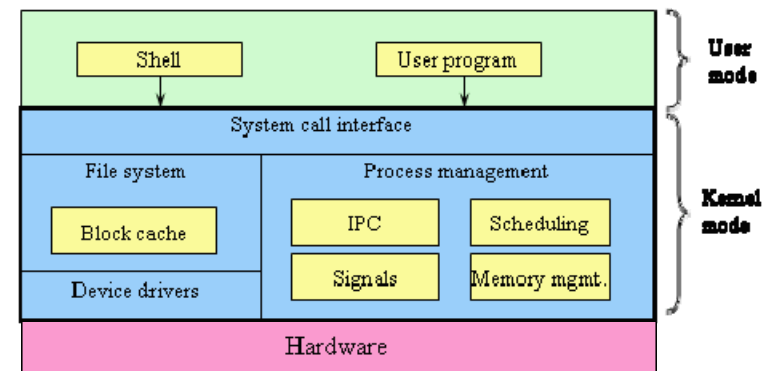
- 사용자에게 컴퓨터를 사용하기 쉬운 환경 제공.
- 컴퓨터 시스템 하드웨어 및 소프트웨어 자원을 여러 사용자 간에 효율적 할당, 관리 보호하는 것.
- 제어 프로그램으로서 사용자 프로그램의 오류나 잘못된 자원 사용을 감시하는 것과 입출력 장치 등의 자원에 대한 연산과 제어를 관리.



OS 구성

운영체제의 구조

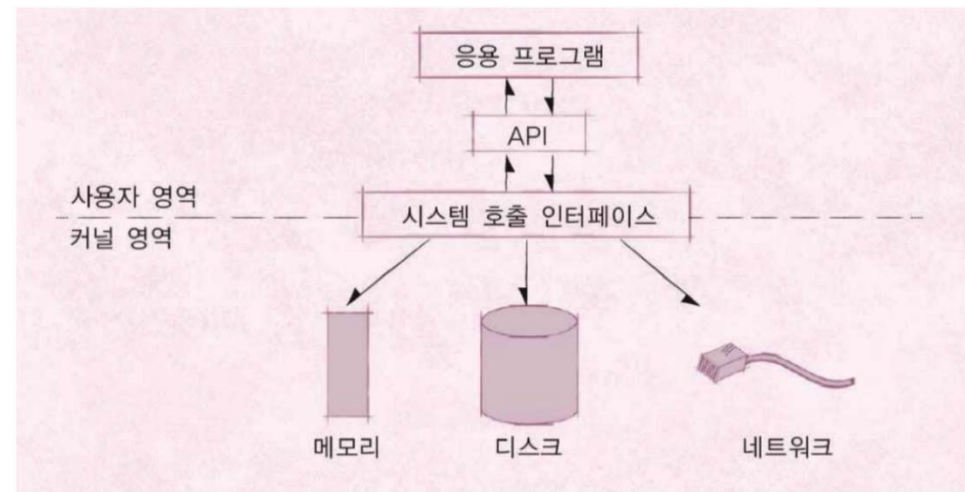
- 셸 / 프로그램
- 커널
 - SYSTEM CALL
 - 메모리 관리
 - 스케줄링
- 하드웨어



OS 구성

운영체제 구조에서의 흐름 예시

1. 응용프로그램에서 API를 통해 SYSTEM CALL 호출.
2. 시스템 콜 인터페이스에서 SYSTEM CALL에 해당하는 커널 함수를 찾아 실행.
3. 커널에서 인터럽트 요청.
4. 인터럽트 발생.
5. 하드웨어 I/O 발생.



스케줄링과 메모리 관리

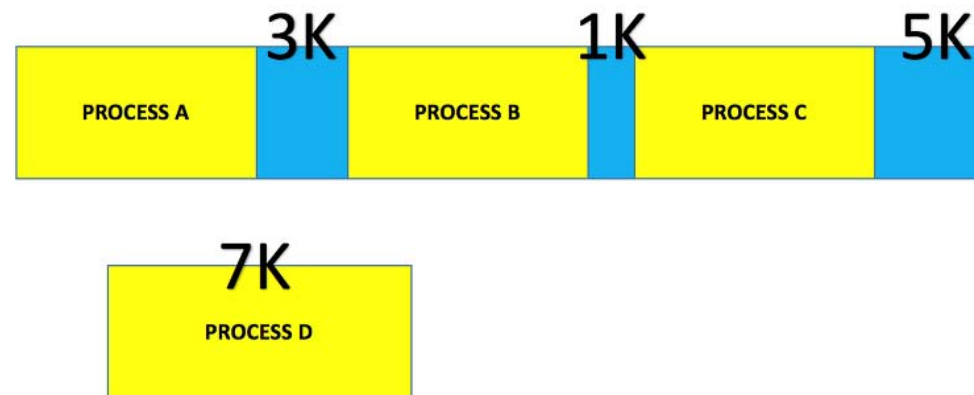
스케줄링

- 한정된 CPU 자원을 효율적으로 프로세스에게 할당하기 위한 모듈.
- 프로세스는 스케줄러에 의해 CPU 자원을 할당 받는다.
- 선점형 스케줄링과 비선점형 스케줄링이 있다.

스케줄링과 메모리 관리

메모리 관리

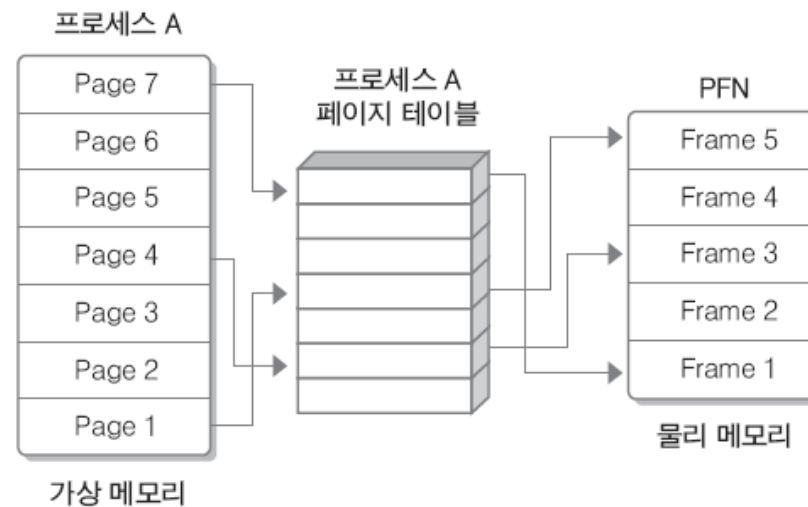
- 외부 단편화 문제
 - 프로세스가 메모리를 연속적으로 할당 받았을 때, 프로세스 메모리간의 빈 공간으로 인해 메모리 낭비가 심해진다.



스케줄링과 메모리 관리

메모리 관리

- 메모리를 논리적으로 분할하여 관리함으로써, 외부 단편화 문제를 막고 낭비되는 메모리를 줄일 수 있다.



Linux와 Unix의 차이

	Unix	Linux
제조사	Solaris(Oracle), AIX(IBM)	커뮤니티에 의해 개발
가격	대부분 유료	무료
장점	기술지원	유닉스와 호환 가능 공개 운영체제의 접근성
단점	Cli 를 사용하기 때문에 사용자 친화적이지 않음	기술지원과 보상이 없음
예시	OS X, Solaris	Ubuntu, Fedora, Debian

2. 개발언어, 개발 툴, 컴파일러

- 프로그래밍 언어
- 컴파일러
- 개발 툴

프로그래밍 언어

프로그래밍 언어란

프로그래밍 언어(programming language)는 컴퓨터 시스템을 구동시키는 소프트웨어를 작성하기 위한 형식언어이다.

프로그래밍 언어

저수준 언어와 고수준 언어

- 저수준 언어 : 컴퓨터 친화적인 언어로, 컴퓨터의 작동 원리를 파악하고 있어야 사용할 수 있음.
 - 기계어
 - 어셈블리 언어
- 고수준 언어 : 인간 친화적인 언어로, 인간이 이해하기 쉬운 문법으로 작성할 수 있음.
 - Java
 - C
 - Python

컴파일러

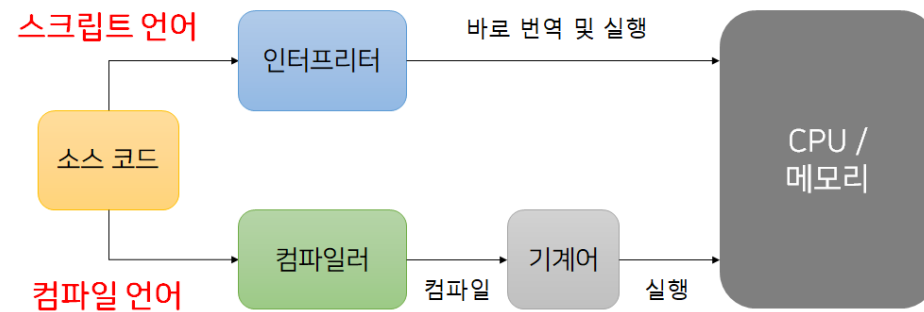
컴파일러란

컴파일러(compiler, 순화 용어: 해석기, 번역기)는 특정 프로그래밍 언어로 쓰여 있는 문서를 다른 프로그래밍 언어로 옮기는 언어 번역 프로그램을 말한다.

컴파일러

인터프리터와 컴파일러

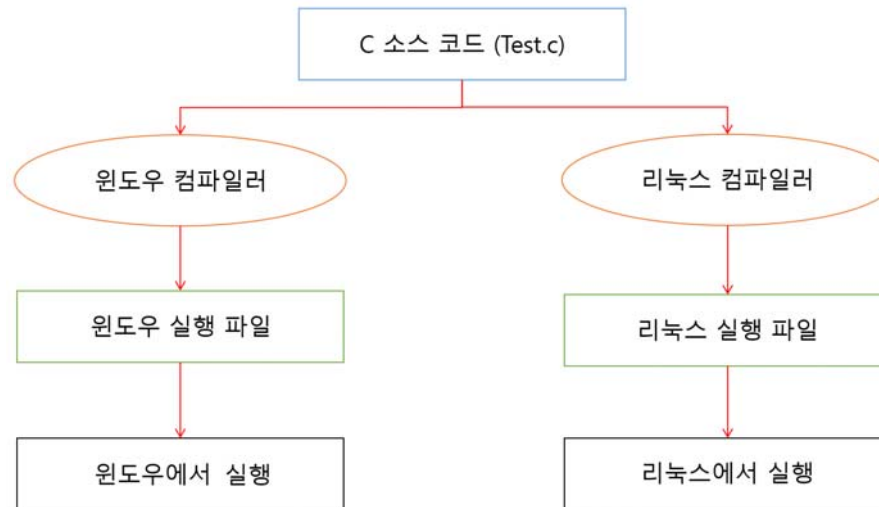
- 인터프리터 : 코드를 한 줄씩 읽으며 즉시 결과를 생성하는 프로그램.
- 컴파일러 : 코드를 한번에 모두 읽어 기계어로 변환하는 프로그램.



컴파일러

OS와 컴파일러

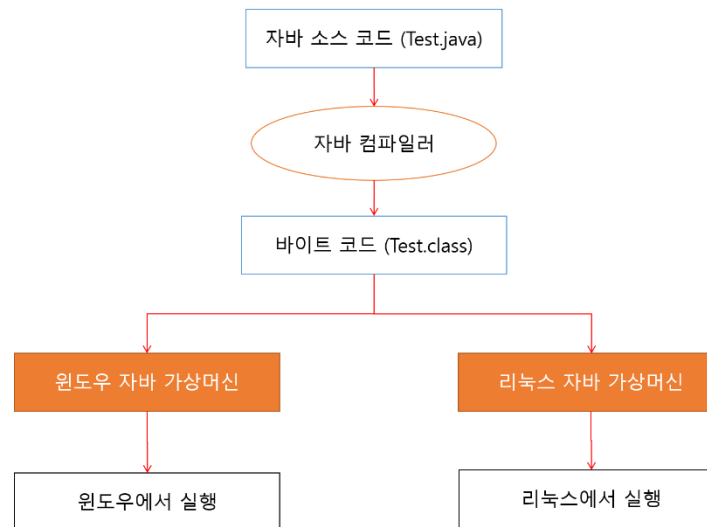
- OS에 맞는 컴파일러를 사용하여 실행 파일을 생성해야 한다.
- 타 OS의 컴파일러를 사용한 실행파일의 경우 해당 OS에서 실행시킬 수 없다.



컴파일러

JVM의 경우

- 자바 컴파일러에 의해 생성된 바이트 코드를 OS 종속적인 JVM이 실행파일로 변환하여 실행.
- 타 OS에서 생성된 바이트 코드를 공유하면, OS에 맞게 설치된 JVM에서 바이트 코드 실행.



개발 툴

통합개발환경(IDE)

- 프로그래머가 소프트웨어 코드를 효율적으로 개발하도록 돕는 소프트웨어 애플리케이션.
- 몇몇 언어에 특화된 개발 환경을 지원하기 때문에 적절한 IDE를 선택하는 것이 중요하다.



C, C++, C#



JAVA, KOTLIN,
SCALA,
SPRING



JAVA, KOTLIN,
ANDROID 앱
개발

3. 네트워크, 클라이언트 서버

- LAN, WAN
- OSI 7 LAYERS
- 클라이언트와 서버

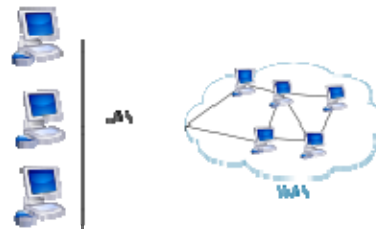
LAN, WAN

LAN

근거리 통신망, 로컬 영역 네트워크는 네트워크 매체를 이용하여 집, 사무실, 학교 등의 건물과 같은 가까운 지역을 한데 묶는 컴퓨터 네트워크이다.

WAN

광역 통신망은 드넓은 지리적 거리/장소를 넘나드는 통신 네트워크 또는 컴퓨터 네트워크이다.

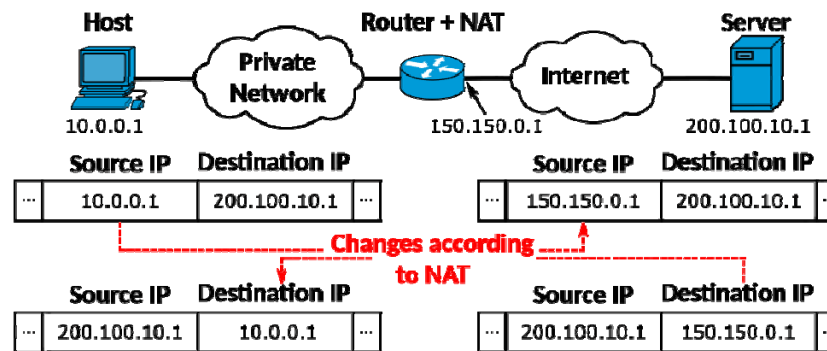


LAN, WAN

NAT

NAT(Network Address Translation)란, IP 패킷에 있는 출발지 및 목적지의 IP 주소와 TCP/UDP 포트 숫자 등을 바꿔 재기록하면서 네트워크 트래픽을 주고 받게 하는 기술.

하나의 IP 주소로 여러 Host가 연결될 수 있어 IP 주소를 절약하고, 외부망으로부터 호스트의 IP를 숨길 수 있는 장점이 있다.



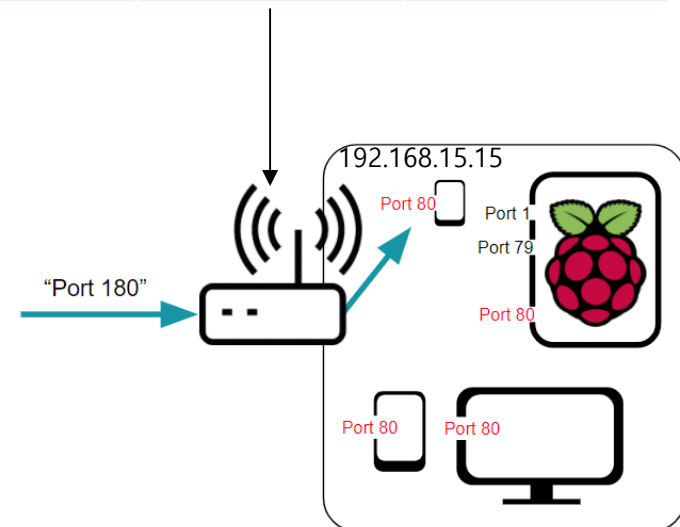
LAN, WAN

포트포워딩

포트포워딩은 NAT의 응용으로서 목적지 IP 주소와 포트 번호의 결합을 통해 목적지에 도달할 수 있도록 한다.

외부망에서 들어온 패킷의 포트 번호와 외부 포트 번호를 비교하여 일치하는 Host가 있으면 패킷을 전달.

내부 ip	외부 포트	내부 포트
192.168.15.15	180	80



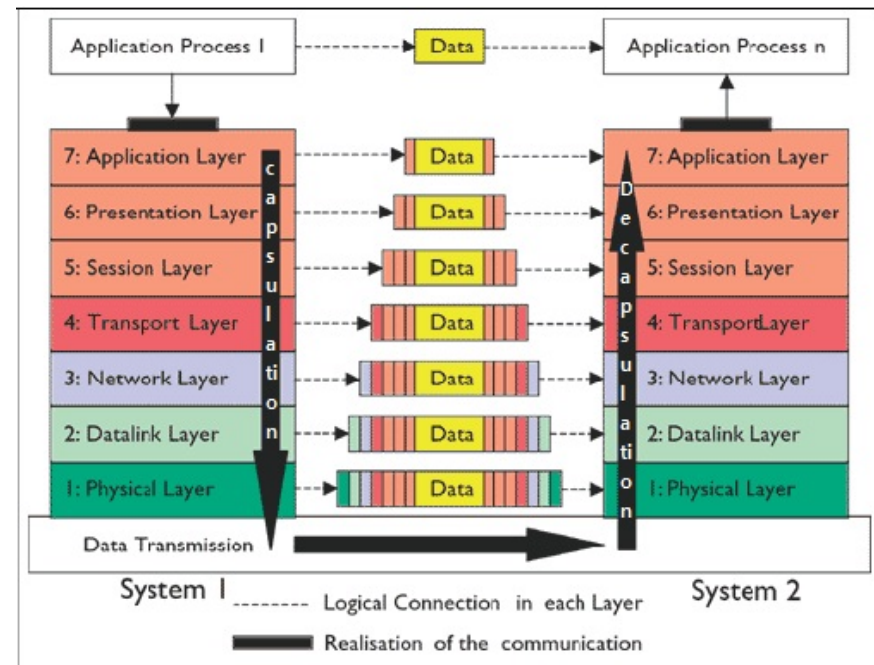
OSI 7 LAYERS

OSI 7 LAYERS

OSI 7 계층은 네트워크에서 통신이 일어나는 과정을 7단계로 나눈 것을 말한다.

효과

- 흐름을 한눈에 알아보기 쉬움.
- 특정한 곳에 문제가 생기면 다른 단계의 장비와 소프트웨어를 건들이지 않아도 됨



OSI 7 LAYERS

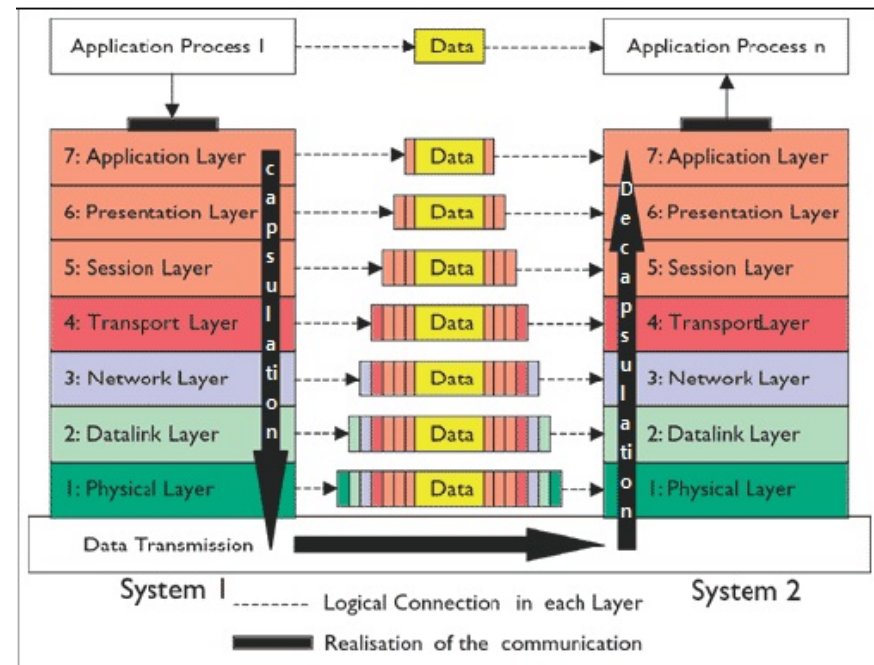
Application LAYERS

응용 프로세스와 직접 관계하여 일반적인 응용 서비스를 수행한다.

데이터 전송 단위 : 메시지

프로토콜

- HTTP
 - 웹 상에서 웹 서버 및 웹브라우저 상호 간의 데이터 전송을 위한 응용계층 프로토콜.



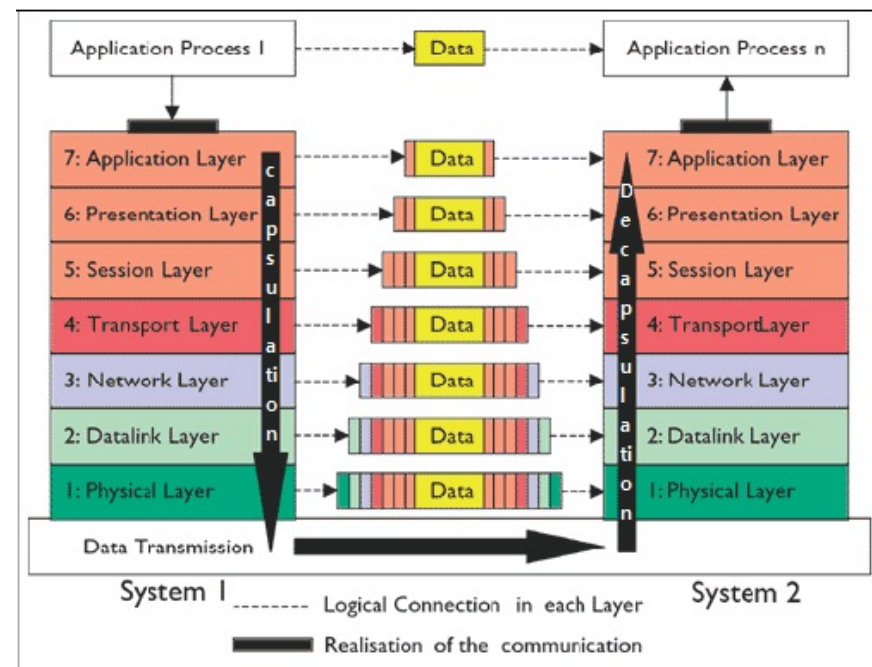
OSI 7 LAYERS

Presentation LAYERS

데이터 표현이 상이한 응용 프로세스의 독립성을 제공하고, 암호화한다.

코드 변환, 구문 검색, 데이터 압축 및 암호화 등의 기능 수행.

데이터 전송 단위 : 메시지



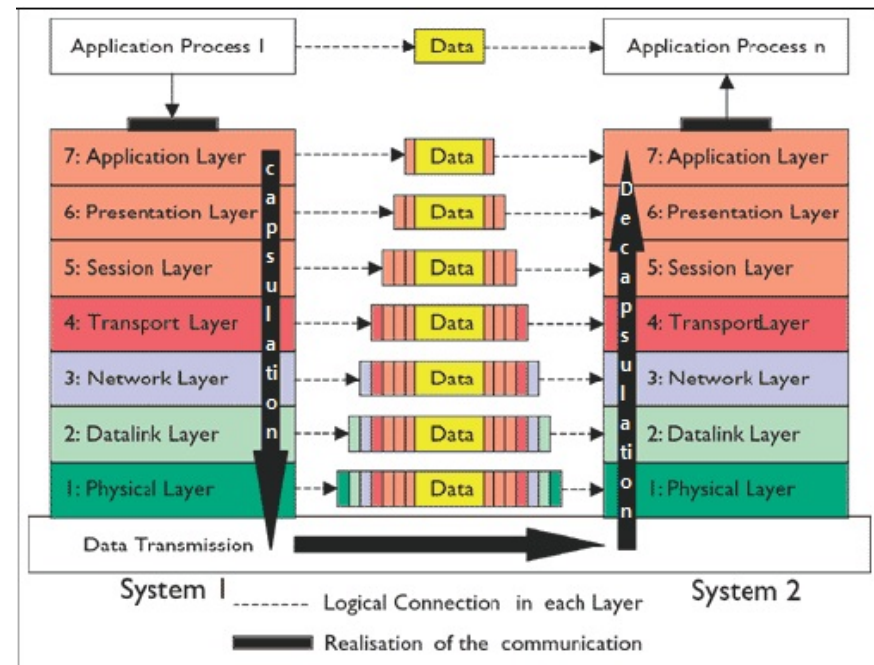
OSI 7 LAYERS

Session LAYERS

데이터가 통신하기 위한 논리적인 연결을 말한다.

데이터 전송 단위 : 메시지

- WinSock
 - 유닉스 등에서 TCP/IP 통신시 사용하는 Socket을 Windows에서 그대로 구현한 것.



OSI 7 LAYERS

Transport LAYERS

통신을 활성화하기 위한 계층.
보통 TCP 프로토콜을 이용하며, 포트를 열어
서 응용 프로그램들이 전송을 할 수 있게
한다.

데이터 전송 단위 : 세그먼트

프로토콜

- TCP
 - 전송제어프로토콜
 - 데이터의 전달을 보장하고 순서를 보장함.
- UDP
 - 비연결성이고 신뢰성이 없으며, 순서화되지 않은 Datagram 서비스 제공.

Source port address(16bit)		Destination port address(16bit)	
Sequence number(32bit)			
Acknowledge number(32bit)			
Data Offset (4bit)	Reservation (6bit)	TCP flag (6bit)	Window size(16bit)
Checksum(16bit)			Urgent pointer(16bit)
Option			
Option			
Option			
Option			padding
Data			

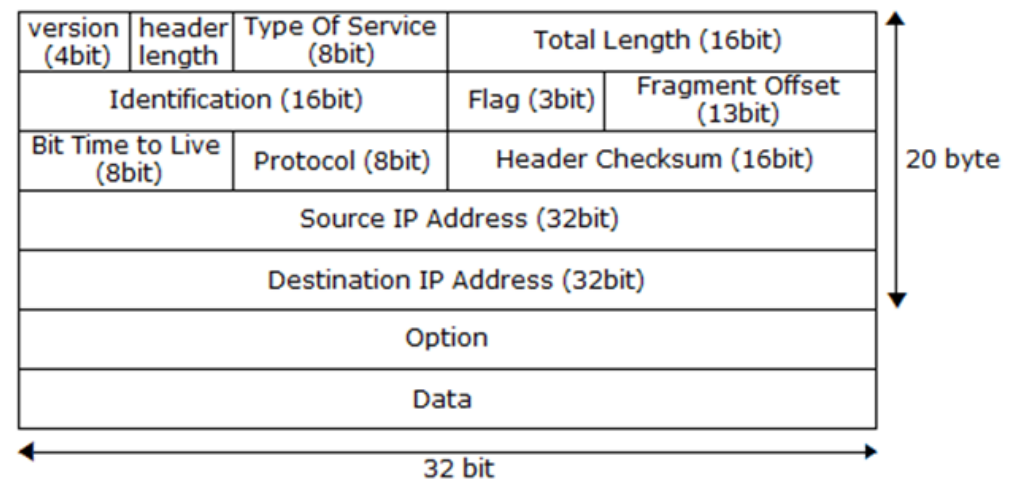
OSI 7 LAYERS

Network LAYERS

TCP/IP 상에서 네트워크 주소를 정의하고 IP 패킷의 전달 및 라우팅을 담당하는 계층.

프로토콜

- IP
 - IP 주소 지정
 - IP 패킷의 라우팅 대상이 됨.
 - 신뢰성과 흐름제어를 TCP와 같은 상위 계층에 의존.



OSI 7 LAYERS

Datalink LAYERS

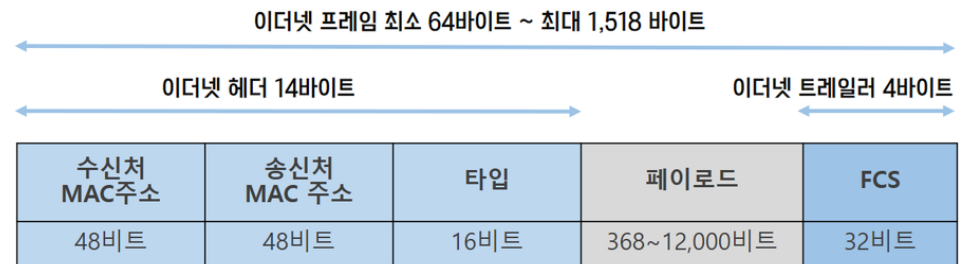
물리계층을 통해 송수신되는 정보의 오류와 흐름을 관리하여 안전한 정보의 전달을 수행할 수 있도록 도움.

MAC 주소를 통해 통신하고, 에러검출/흐름 제어 기능을 수행함.

데이터 전송 단위 : 프레임

프로토콜

- 이더넷
 - CSMA/CD
- ARP



OSI 7 LAYERS

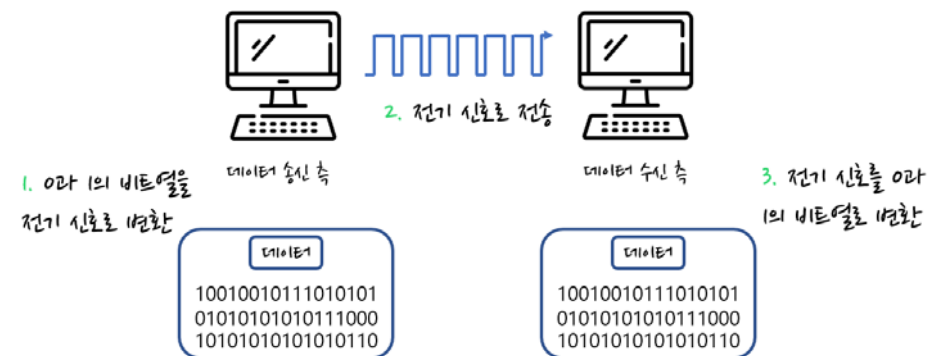
Physical LAYERS

데이터를 전기적 신호로 변환해서 주고받는 기능.

데이터 전송 단위 : 비트

주요 장비

- 리피터
- 허브



클라이언트와 서버

클라이언트

클라이언트는 서버의 서비스를 받아 사용하는 장치, 프로그램을 말한다.

클라이언트의 종류

- 데스크톱
- 노트북
- 모바일 어플리케이션
- 웹 브라우저

클라이언트와 서버

서버

서버는 네트워크를 통해 클라이언트에게 서비스를 제공하는 시스템이다.

서버의 종류

- Web Server
- WAS
- DB Server
- File Server

클라이언트와 서버

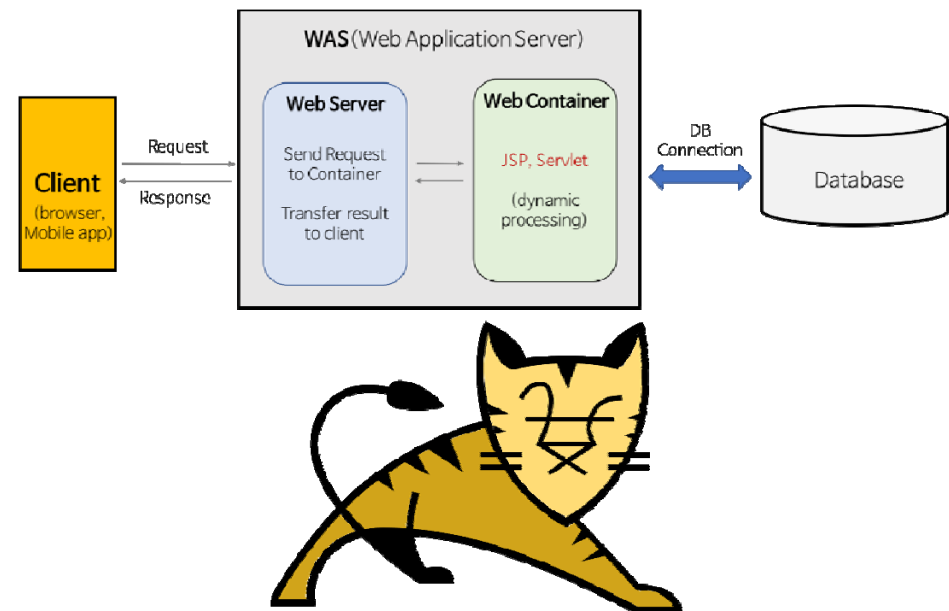
Web Server와 WAS

Web Server

- 웹 브라우저 클라이언트로부터 HTTP 요청을 받아 정적인 콘텐츠(.html .jpeg .css 등) 을 제공하는 컴퓨터 프로그램.

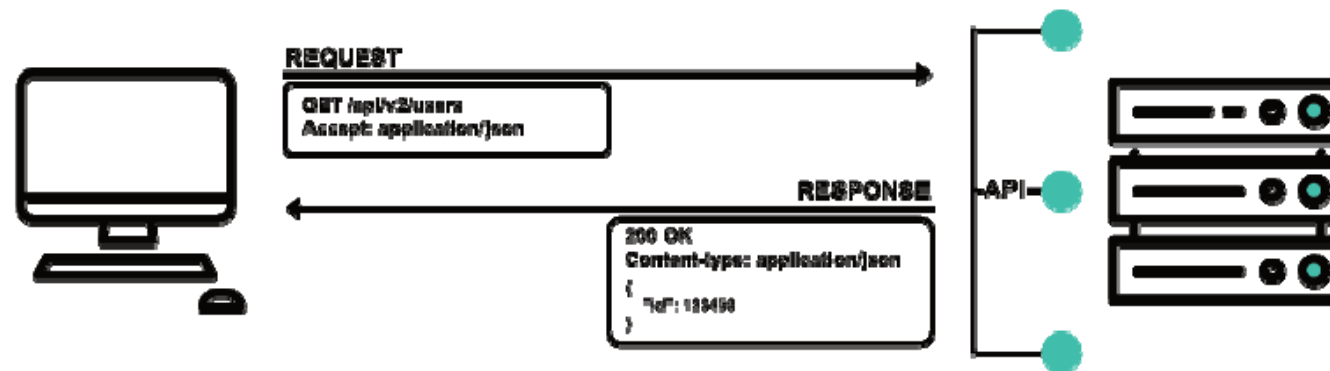
WAS(Web Application Server)

- DB 조회나 다양한 로직 처리를 요구하는 동적인 콘텐츠를 제공하기 위해 만들어진 Application Server.



클라이언트와 서버

클라이언트와 웹 서버간 통신



클라이언트와 서버

API

API(Application Programming Interface)는 컴퓨터나 컴퓨터 프로그램 사이의 인터페이스이다.

API 의 종류

- Private API : 내부 API로, 기업이나 연구 단체 등에서 자체 제품과 운영 개선을 위해 단체 내부에서만 사용.
- Public API : 개방형 API로, 모두에게 공개된다. 그 중 접속하는 대상에 대한 제약이 없는 경우를 Open API라고 한다.

클라이언트와 서버

REST API

REST API는 다양하게 생성될 수 있는 API의 URL 형태를 통일시키기 위한 규칙.

REST API 디자인 가이드

1. URI는 정보의 자원을 표현해야 한다(행위가 들어가면 안됨).
2. 자원에 대한 행위는 HTTP Method(GET,POST,PUT,DELETE)로 표현한다.

회원정보를 가져오는 URI

GET /members/show/1	(x)
GET /members/1	(o)

회원을 추가할 때

GET /members/insert/2	(x)
POST /members/2	(o)

4. 데이터베이스

- RDBMS
- 테이블
- 인덱스

RDBMS

데이터베이스란

데이터베이스 관리 시스템은 다수의 사용자들이 데이터베이스 내의 데이터를 접근할 수 있도록 해주는 소프트웨어 도구의 집합이다.

데이터베이스는 여러 사람이 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합이다.

RDBMS

관계형 데이터베이스

관계형 데이터베이스란 Key와 Value로 이루어진 간단한 데이터의 관계를 테이블로 나타낸 것을 말한다.

order_no	product_id	amount	create_dt

Primary Key

Value

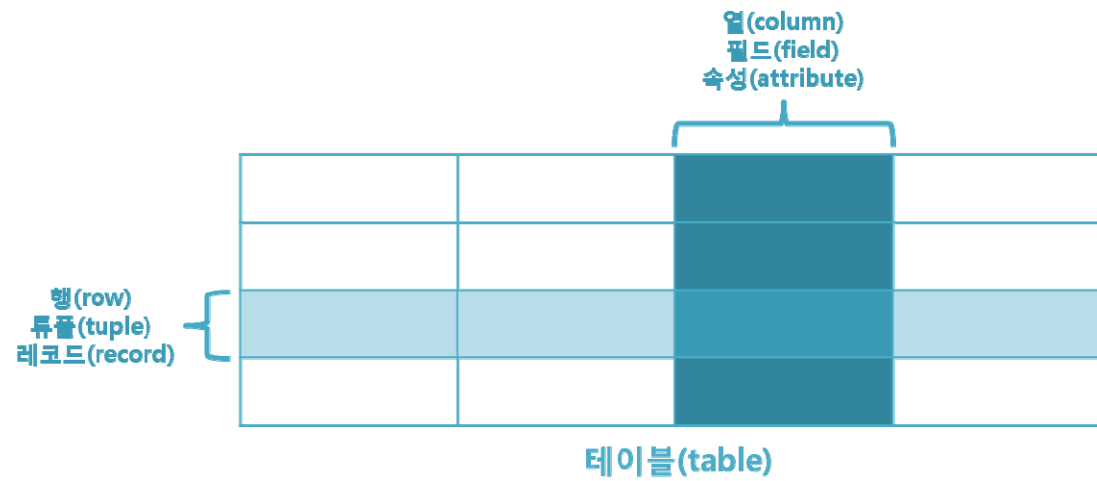
RDBMS

관계형 데이터베이스의 특징

- 정규화를 통해 테이블간 관계를 정의해 데이터 정합성을 지킬 수 있다.
- 트랜잭션 ACID를 지원하여 데이터의 안정성을 높일 수 있다.

테이블

테이블의 구조



테이블

DDL(데이터 정의 언어)

데이터베이스 스키마를 정의하거나 조작하기 위해 사용한다.

SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 다음 명령어로 정의, 변경, 삭제한다.

- CREATE
- ALTER
- DROP
- TRUNCATE

테이블

DML(데이터 조작 언어)

데이터를 조작(조회, 추가, 변경, 삭제) 하기 위해 사용한다.

- SELECT
- INSERT
- DELETE
- UPDATE

테이블

DCL(데이터 제어 언어)

데이터베이스에 접근하고 객체들을 사용하도록 권한을 주고 회수하는 명령어.

- GRANT
- REVOKE

테이블

TCL(트랜잭션 제어 언어)

트랜잭션을 제어하는 언어.

- COMMIT
- ROLLBACK
- SAVEPOINT
- TRANSACTION

테이블

트랜잭션

트랜잭션이란 작업을 수행하는 논리적인 단위이다.

질의어 하나하나가 트랜잭션이 아닌, 작업에 맞는 질의어를 모아놓은 것이 트랜잭션이다.

테이블

트랜잭션 ACID

- 원자성(Atomicity)
 - 트랜잭션은 반영이 되거나 안되거나 둘 중 하나여야 한다.
- 일관성(Consistency)
 - 트랜잭션이 적용될 때, 데이터 무결성을 위해 미리 정해진 제약조건을 벗어나지 않아야 한다.
- 독립성(Isolation)
 - 다른 트랜잭션에 영향을 받아서는 안 된다.
- 영구성(Durability)
 - 트랜잭션 적용 후 시스템 오류가 발생해도 데이터 저장을 보장해야 한다.

테이블

격리 수준

- Read Uncommitted : 다른 트랜잭션에서 Commit 되지 않은 데이터를 참조할 수 있다.
 - Dirty Read, Non-Repeatable Read, Phantom Read
- Read Committed : 다른 트랜잭션에서 Commit된 데이터만 참조할 수 있다.
 - Non-Repeatable Read, Phantom Read
- Repeatable Read : 트랜잭션에 진입하기 이전에 Commit된 내용만 참조할 수 있다.
 - Phantom Read
- Serializable : 트랜잭션에 진입하면 락을 걸어 다른 트랜잭션이 접근할 수 없다.

테이블

제약조건

- 기본키 제약 조건 : 컬럼 값은 반드시 존재해야하며(NOT NULL) 유일해야함(UNIQUE).
- 고유키 제약 조건 : 중복값 입력 금지(NULL 값은 가능).
- NOT NULL 제약 조건 : NULL 값 입력 금지.
- 체크 제약 조건 : 조건으로 설정된 값만 입력 허용.
- 외부 참조키 제약 조건 : 다른 테이블의 컬럼을 조회해서 무결성 검사.

테이블

KEY의 종류

- 슈퍼 키 : 유일성을 만족하는 키
- 복합 키 : 2개 이상의 속성을 사용한 키
- 후보 키 : 유일성과 최소성을 만족하는 키. 기본 키가 될 수 있는 후보이기 때문에 후보키라고 불린다.
- 기본 키 : 후보 키에서 선택된 키. NULL값이 들어갈 수 없으며 중복값 입력이 불가하다.
- 대체 키 : 후보 키 중에 기본 키로 선택되지 않은 키
- 외래 키 : 다른 테이블의 기본 키를 참조하는 속성

테이블

정규화

정규화란 이상 현상을 제거하기 위해서 데이터베이스를 올바르게 설계해 나가는 과정

이상 현상의 종류

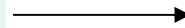
- 삽입 이상 : 데이터를 삽입하기 위해 불필요한 데이터도 함께 삽입해야 하는 문제
- 갱신 이상 : 중복 튜플 중 일부만 변경하여 데이터가 불일치하게 되는 문제
- 삭제 이상 : 튜플을 삭제하면 꼭 필요한 데이터까지 같이 삭제되는 데이터 손실 문제

테이블

제 1 정규화

- 모든 속성은 반드시 하나의 값을 가져야 한다.

<u>order_id</u>	<u>user_id</u>	status
1	1	1,2
2	2	1



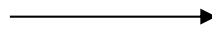
<u>order_id</u>	<u>user_id</u>	status
1	1	1
1	1	2
2	2	1

테이블

제 2 정규화

- 모든 속성은 반드시 식별자 속성 전부에 종속되어야 한다.

<u>order_id</u>	<u>user_id</u>	user_name
1	1	LEE
2	2	JANG
3	3	KIM



order_id	user_id
1	1
2	2
3	3

user_id	user_name
1	LEE
2	JANG
3	KIM

테이블

제 3 정규화

- 식별자가 아닌 모든 속성 간에는 서로 종속될 수 없다.

<u>order_id</u>	product_id	product_name
1	1	Desk
2	2	Chair
3	3	Mouse



order_id	product_id
1	1
2	2
3	3

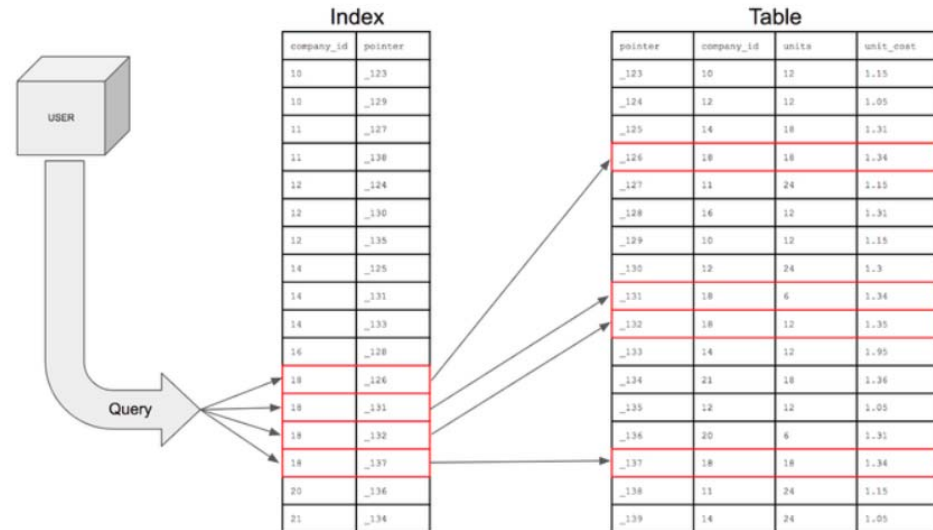
product_id	product_name
1	Desk
2	Chair
3	Mouse

인덱스

인덱스의 의미와 효과

데이터를 빠르게 조회하기 위해 Key 값으로 미리 정렬된 구조.

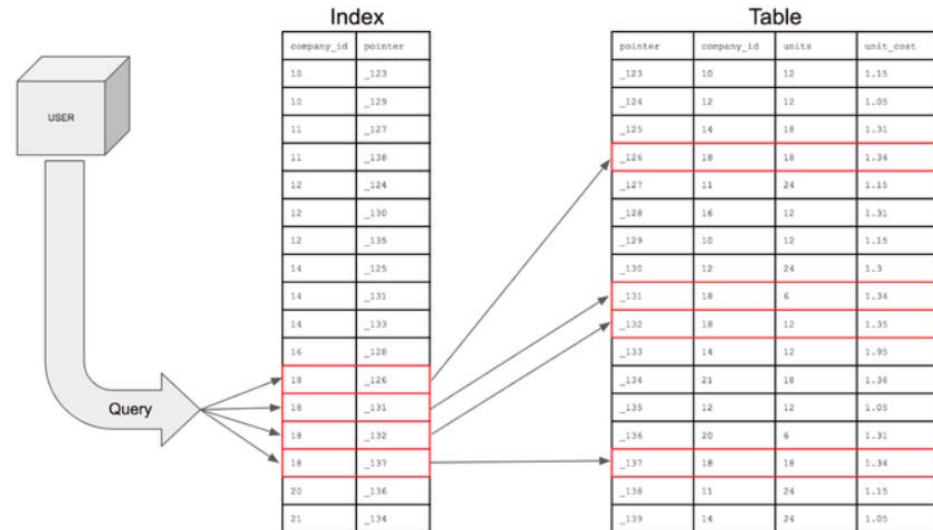
1. Key 값으로 미리 정렬되어 있으므로 Key 값에 해당하는 조건으로 검색 시 빠르게 조회할 수 있음.
2. 인덱스의 key값만 조회 시 디스크에 접근할 필요 없음.
3. 정렬되어 있기 때문에 정렬연산을 회피할 수 있다.



인덱스

인덱스의 단점

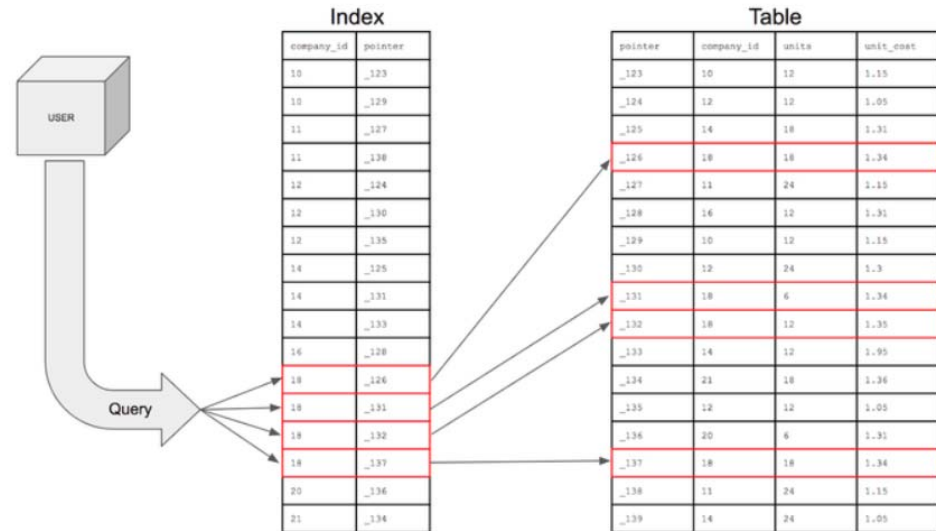
1. 데이터 삽입, 삭제가 이뤄졌을 때 인덱스도 함께 적용되어야 하기 때문에 속도가 느려짐.
2. 추가 저장 공간 필요.
3. 데이터 중복도가 높을 경우, range scan 범위가 늘어나 비효율적일 수 있음.



인덱스

효과적인 인덱스 사용

1. 데이터 삽입, 삭제가 자주 일어나지 않는 컬럼.
2. 데이터 중복도가 낮은 컬럼.
3. 정렬된 결과에 자주 사용되는 컬럼.



5. 프레임워크와 라이브러리

- 라이브러리
- 프레임워크

라이브러리

라이브러리

라이브러리는 주로 소프트웨어를 개발할 때 컴퓨터 프로그램이 사용하는 비휘발성 자원의 모임이다.

예시 :

- C++ 표준 템플릿 라이브러리(STL)
- Node.js에서 npm으로 설치한 모듈
- HTML의 클라이언트 사이드 조작을 단순화하는 JQuery



프레임워크

프레임워크

- 프레임워크는 작업(work)의 구조(frame)가 정해져 있는 라이브러리라고 볼 수 있다.
- 앱/서버 emdd의 구동, 메모리 관리, 이벤트 루프 등의 공통된 부분은 프레임워크가 알아서 관리.

예시 :

- Java 서버 개발에 사용되는 Spring
- Python 서버 개발에 사용되는 Django, Flask

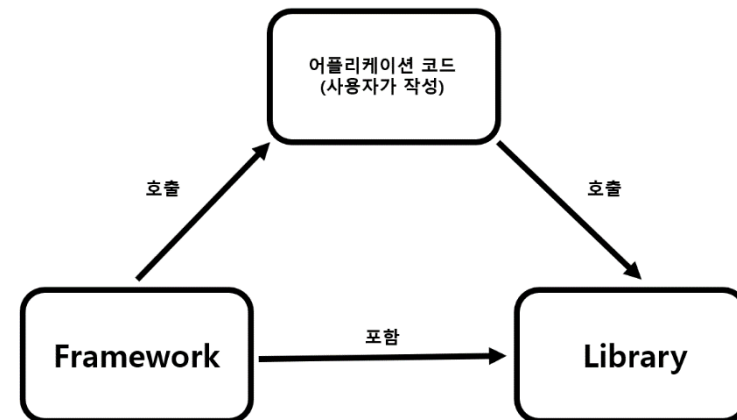


프레임워크

라이브러리와 프레임워크

프레임워크와 라이브러리의 차이점은 제어 흐름의 권한이 어디에 있는가 이다.

- 라이브러리
 - 사용자는 애플리케이션 코드의 흐름을 직접 제어해야 한다.
- 프레임워크
 - 프레임워크가 짜 놓은 틀에서 애플리케이션 코드가 수동적으로 동작



6. 형상관리와 협업 툴


- 형상관리
- Git
- Svn

형상관리

형상관리

소프트웨어의 변경사항을 체계적으로 추적하고 통제하는 것.

- 문서나 파일의 변경 원인과 변경 사항을 기록하고, 통제
- 협업의 경우, 다른 사람이 변경한 코드를 쉽게 파악할 수 있기 때문에 형상관리는 필수이다.



항의하기 버튼 눌렀을때 기본은 사라지노록 수정

2.0.1 빌드 버전 올림 **버전관리**

글자 opacity 제거
버전 업로드
미리보기 추가 이미지 업로드
UI 개선 **변경 관리**

v2.0 Merge pull request #20 from guardianspowermask/Detail

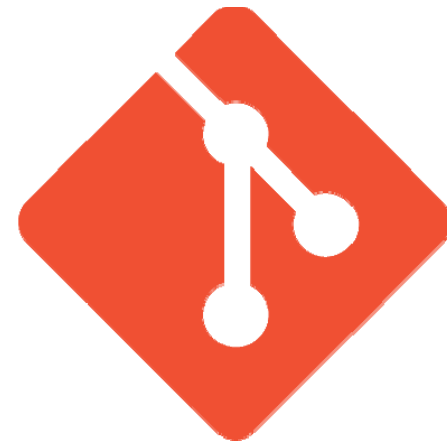
```
src 00 ~3,5 +3,13 @@ disableCommentingDefaultMessage = Comentários foram fechados nessa história.  
3 3  
4 4 reaction-labelRespect = Respeitar  
5 5 reaction-labelActiveRespected = Respeitado  
6 ~ reaction-sortLabelMostRespected = Respeitados  
+ reaction-sortLabelMostRespected = Mais Respeitados  
7 +  
8 + comment-count =  
9 + <span class="{ $numberClass }">{ $number }</span>  
10 + <span class="{ $textClass }">{ $number }->  
11 + [one] Comentário  
12 + * [other] Comentários  
13 + </span>  
14 +  
15 staff-label = Staff
```

Git

Git

개발자가 중앙 서버에 접속하지 않은 상태에서도 코딩 작업을 할 수 있도록 지원하는 버전 관리 시스템

- 팀 개발을 위한 분산 환경 개발에 최적화
- 원격 서버 Git Repository에 push 하지 않은 채 여러 branch 생성이 가능함
- 로컬에서 변경 이력을 관리할 수 있기 때문에, Git 저장소만 있으면 리모트 복구가 언제든지 가능.



SVN

SVN

서버와 클라이언트로 구분되어, 개발과정에서 사용하는 파일들을 관리하기 위한 시스템

- 중앙집중식 형상 관리
- 최초 1회에 한해 파일 원본을 저장하고 이후에는 실제 파일이 아닌 원본과 차이점을 저장하는 방식
- 다수의 개발자가 하나의 저장소에 커밋



SVN

Git과 SVN

Git과 SVN의 차이는 버전 관리를 로컬에서 하느냐 중앙 서버에서 하느냐 이다.

- Git은 레포지토리에서 Pull해오면 모든 이력을 로컬에서 조회할 수 있다.
- Git은 로컬에 개인적인 이력을 가질 수 있지만, SVN은 로컬에 이력을 가질 수 없다.
- SVN은 크기가 큰 파일 관리에 유리

