

Fastcampus

Computer Science Extension School

Python Basic\_Day4



# 보충학습

## Switch

| 패킷 충돌 방지를 위해 패킷의 목적지를 기준으로 1:1 연결

# 무선공유기

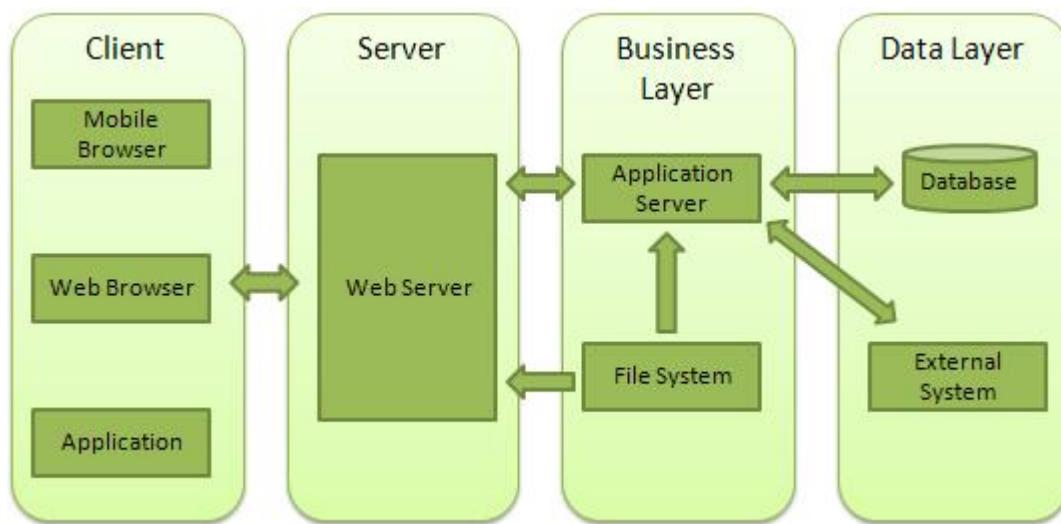
SOHO Router => Router + AP

# Network L2,3,4 Switch

- L2 Switch: Switching hub(MAC Address)
  - 패킷의 MAC Address를 읽어 스위칭
  - 데이터 링크 간 스위칭
  - 브릿지: 받은 데이터를 모든 포트로 전송
  - 스위치허브: 받은 데이터를 특정 포트로 전송
- L3 Switch: Router(IP Address)
  - IP 주소를 읽어 스위칭
  - 다른 네트워크간 스위칭
  - L2 + 라우팅
- L4 Switch: Load Balancer(IP Address, TCP/UDP Port Number)
  - 트래픽을 여러 서버로 분산(서버 1대처럼 사용)
  - L3 + 가상 랜, 그룹화, 로드밸런싱
  - http=80, https=443, ftp=21, smtp=25, ..
  - [https://ko.wikipedia.org/wiki/TCP/UDP의\\_포트\\_목록](https://ko.wikipedia.org/wiki/TCP/UDP의_포트_목록)

# Web Programming

# Web architecture



## 웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>Static Header</h1>
<div>Static Contents</div>
</body>
</html>
```

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달

## 웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>{% Dynamic Header %}</h1>
<div>{% Dynamic Contents %}</div>
</body>
</html>
```

- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨

## 웹 개발 패턴의 변화

```
<html>
<head>
<script src="https://unpkg.com/vue"></script>
</head>
<body>
<h1>{{ header }}</h1>
<div id="app">
  {{ message }}
</div>
<script>
var app = new Vue({
  el: '#app',
  data: {
    message: '안녕하세요 Vue!'
  }
})
</script>
</body>
</html>
```

- 2010 ~ 현재: JavaScript!! (Dynamic Web Client)

# 웹 개발의 현재

javaScript

# Client-side

- HTML/CSS, javaScript
- jQuery, AJAX
- Front-end Web Framework
  - AngularJS
  - React.js
  - Vue.js
- CSS Framework
  - Bootstrap
  - Foundation

# Server-side

- Depends on Language
  - PHP: Laravel
  - javaScript: Node.js(Express.js)
  - Java: Spring
  - C++, C#: [ASP.net](#)
  - Python: Django, Flask
  - Golang: itself
  - Ruby: Ruby on Rails

# Database

- RDBMS
  - MySQL
  - PostgreSQL
  - MariaDB
- noSQL
  - MongoDB
  - CouchDB
  - Redis

# URI, URL, URN

## URI

- Uniform Resource Information
- `https://www.example.com/post/how-to-make-url`

## URL

- Uniform Resource Locator
- `https://www.example.com/post/`

## URN

- Uniform Resource Name
- `www.example.com/post/how-to-make-url`

# API

## Application Program Interface

- 응용프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스
- [Windows API](#)
- [python/C API](#)

# Web API

웹서버 혹은 웹브라우저를 위한 API

# REST API

RE presentational S tate T ransfer

A pplication P rogramming I nterface

Resource - URI

Verb - HTTP method

Representations - 표현

# So, REST is

- HTTP URI + HTTP method

[Yahoo Finance](#)

[json api](#)

# Roy Fielding



- 2000년 UC Irvine의 박사 학위 논문 "Architectural Styles and the Design of Network-based Software Architectures" 발표

# Characteristics of REST

- 범용성(HTTP가 가능하면 OK)
- 리소스 중심 API 명세(URI를 읽는 것으로 이해 가능)
- Stateless(클라이언트의 상태를 신경쓰지 않음)

# pros and cons of REST

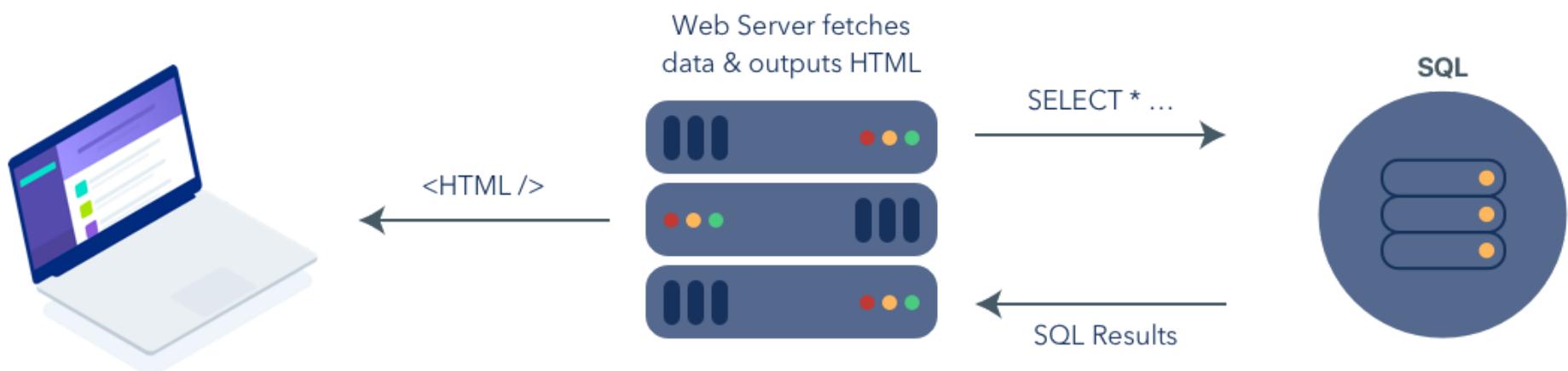
pros:

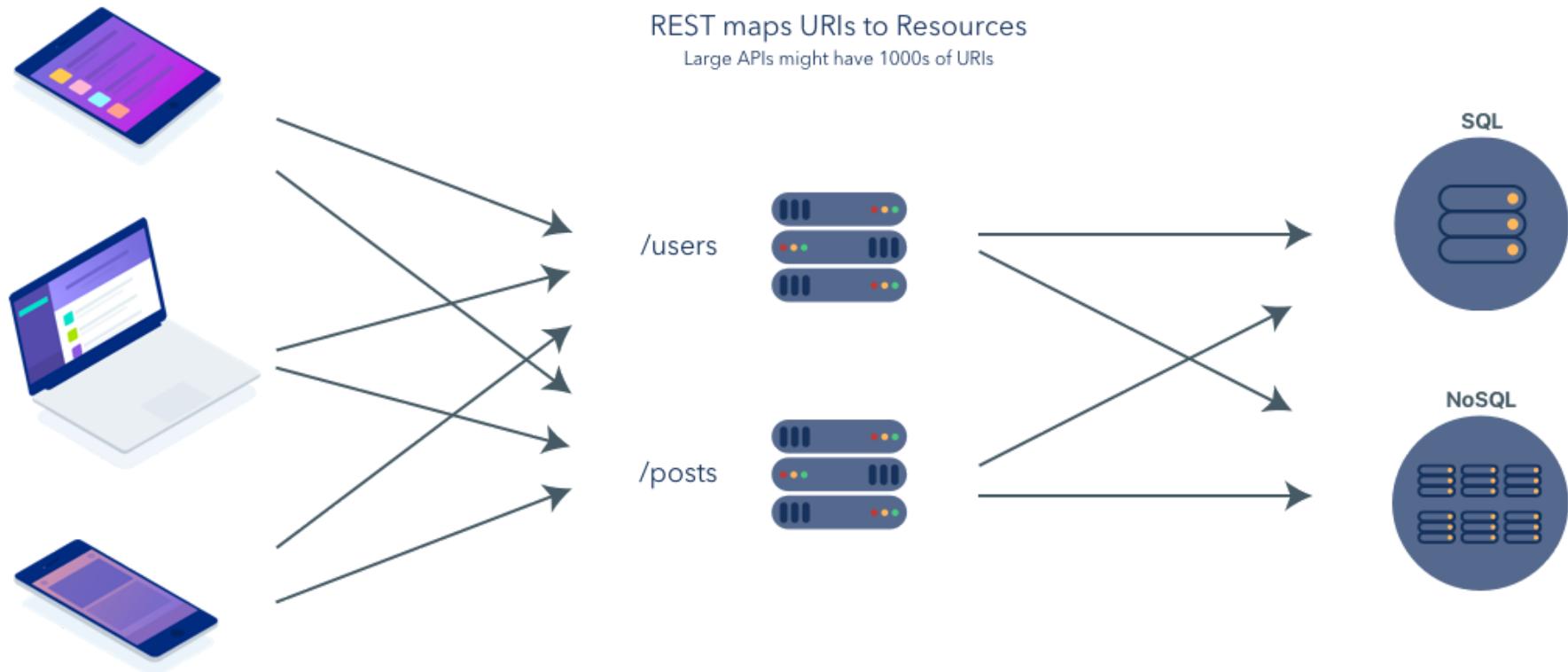
- 스펙없이 기존의 HTTP를 이용해 요청을 처리할 수 있다.

cons:

- 사용할 수 있는 메소드가 4개다
- 표준이 없다

## Before REST





# CRUD

Create

Read

Update

Delete

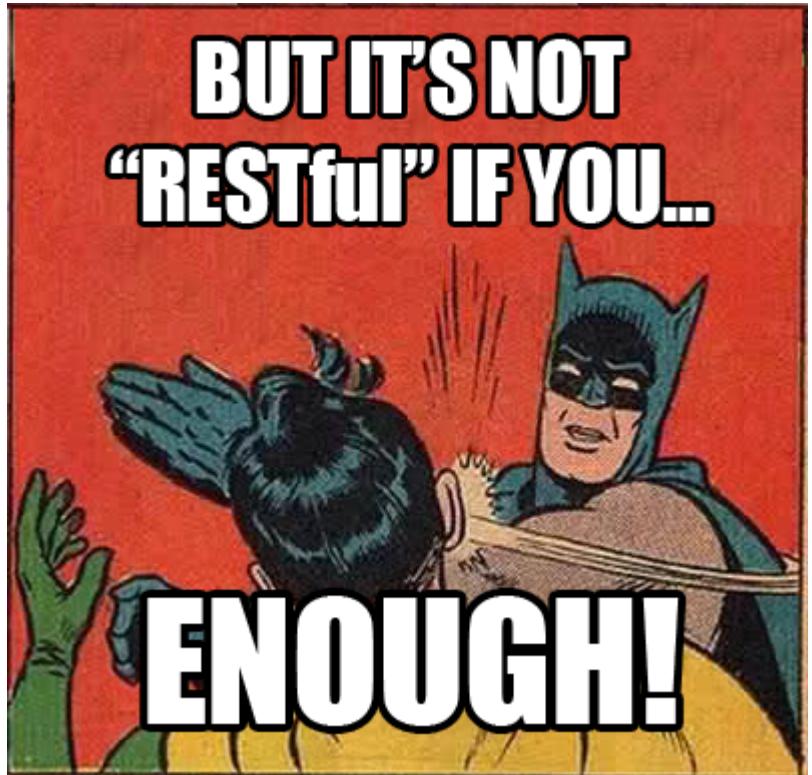
# HTTP Response code

200, 201 - Success

400, 404 - Not found

500 - Server error

[more info..](#)



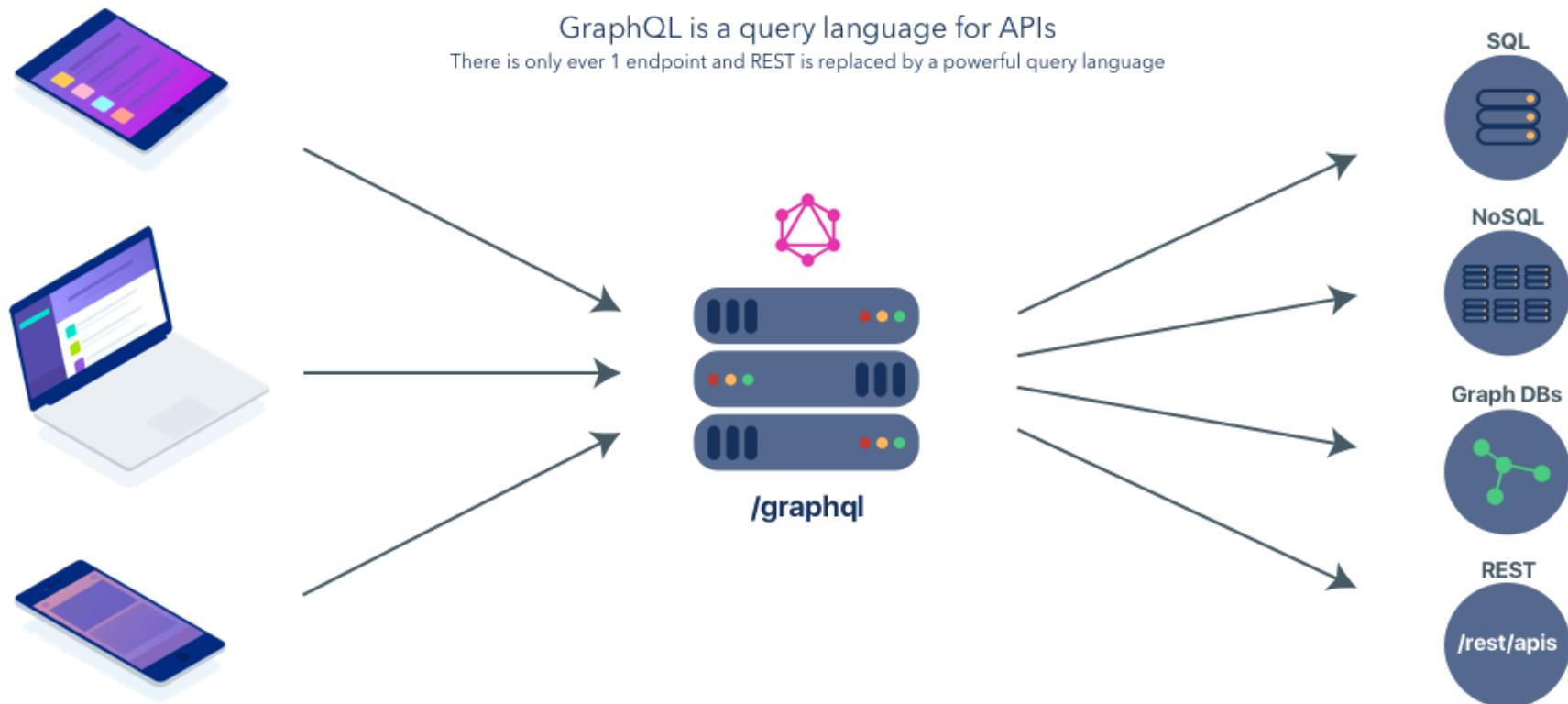
# API의 미래

## GraphQL

# GraphQL

- Open-sourced by Facebook
- Alternative to REST for building APIs
- create strong contract between Client and Server

# GraphQL



## data

- 컴퓨터가 처리할 수 있는 문자, 숫자, 소리, 그림 따위의 형태로 된 정보.
- Latin "Datum"의 복수형 "Data"에서 유래

# Database

- 체계화된 데이터의 모임
- 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음

# DB?? DBMS??

DBMS(DataBase Management System)

- 데이터의 모임인 Database를 만들고, 저장, 관리 할 수 있는 기능을 제공하는 응용 프로그램
- Oracle, Mysql, MariaDB, DB2, MS SQL Server, ..

# DBMS의 조상님



# DBMS의 조상님

## dBASE

- 마이크로컴퓨터용 최초의 DBMS
- 1979년 Ashton이 개발
- SQL이 아닌 독자 스크립트 언어로 실행 -> dbf 파일 생성

# Characteristics

- 데이터의 무결성
- 데이터의 중복 방지
- 보안(추상화, 접근권한)
- 성능 향상
- 프로그램 수정과 유지 보수 용이

# Differences between DataBase & File System

자기기술성

File System

- .hwp -> 한글
- .doc -> Microsoft Word
- .xls -> Microsoft Excel

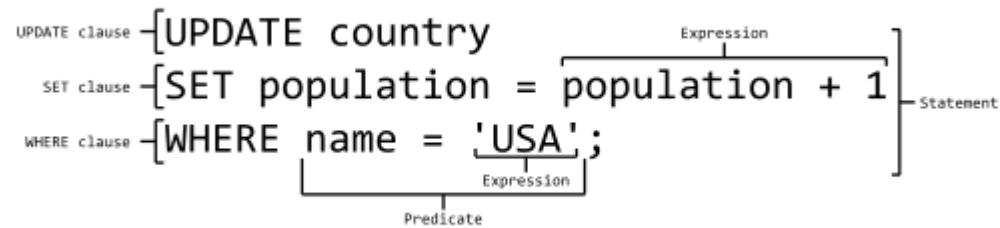
DB

- Only SQL(RDBMS)



# SQL(Structured Query Language)

데이터 관리를 위해 설계된 특수 목적의 프로그래밍 언어



# SQL - 데이터 정의언어

데이터를 정의

CREATE - DB 개체 정의

DROP - DB 개체 삭제

ALTER - DB 개체 정의 변경

# SQL - 데이터 조작언어

데이터 검색, 등록, 삭제, 갱신

INSERT - 행, 테이블 데이터 삽입

UPDATE - 테이블 업데이트

DELETE - 특정 행 삭제

SELECT - 테이블 검색 결과 집합

# SQL - 데이터 제어언어

데이터 액세스 제어

GRANT - 작업 수행권한 부여

REVOKE - 권한 박탈

# RDBMS vs NoSQL

구분	RDBMS	NoSQL
형태	Table	Key-value, Document, Column
데이터	정형 데이터	비정형 데이터
성능	대용량 처리 시 저하	잦은 수정 시 저하
스키마	고정	Schemeless
장점	안정적	확장성, 높은 성능
유명	Mysql, MariaDB, PostgreSQL	MongoDB, CouchDB, Redis, Cassandra

# RDBMS

[PostgreSQL Docs](#)

[MariaDB Docs](#)

name	age
John	17
Mary	21

```
rdb =  
{  
    name: [John, Mary],  
    age: [17, 21]  
}
```

## Table == Relation

Primary Key	Attribute1	Attr2	Attr3	Attr4
Tuple1				
Tuple2				
Tuple3				
Tuple4				

# NoSQL

## MongoDB Docs

```
nosql =  
[  
    {  
        name:John,  
        age:17  
    },  
    {  
        name:Mary,  
        age:21  
    },  
    ...  
]
```

# Document vs Key-value

```
document
{
    key: value,
    key: {
        key: value,
        key: value
    }
}
```

```
key-value
{
    key: value,
    key: value,
    key: value
}
```

# How to Design Database?

# Schema

- Database의 구조와 제약조건에 대한 전반적인 명세 기술
- Database의 Blueprint
- 외부(서브)스키마, 개념스키마, 내부스키마로 구성

## 외부(서브) 스키마

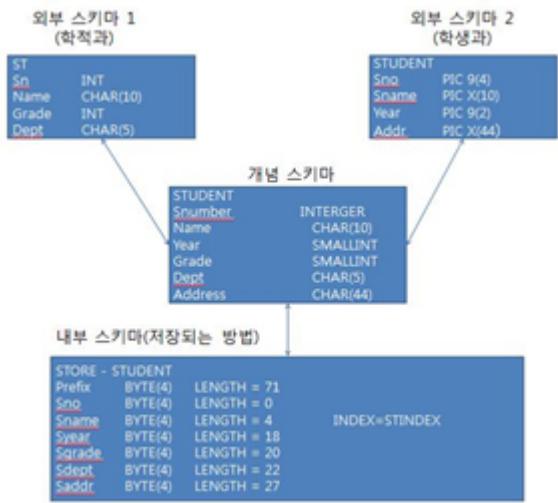
- 프로그램 사용자가 필요로 하는 데이터베이스의 논리적인 구조를 정의

## 개념 스키마

- 조직 전체의 관점에서의 구조와 관계를 정의
- 외부 스키마의 합과 그 사이의 데이터의 관계 등등
- 일반적인 스키마의 정의

## 내부 스키마

- 저장장치의 입장에서 데이터베이스가 저장되는 방법을 기술



# Design Database

Primary Key	Attribute1	Attr2	Attr3	Attr4
Tuple1				
Tuple2				
Tuple3				
Tuple4				

# SQLite

# SQLite with python

- for windows:
  - <https://www.sqlite.org/2018/sqlite-tools-win32-x86-3230100.zip>
  - <https://www.sqlite.org/download.html>
- for mac:

## SQLite - check sqlite version

```
$ python
>> import sqlite3
>> sqlite3.version
>> sqlite3.sqlite_version
```

## SQLite - Create table

```
$ sqlite3 users.db  
sqlite> .tables  
sqlite> .exit  
  
$ vi users.db
```

# SQLite - Create table & Insert User

```
$ sqlite3 user.db
SQLite version 3.16.0 2016-11-04 19:09:39
Enter ".help" for usage hints.
sqlite> CREATE TABLE user (
...>     id integer primary key autoincrement,
...>     name text not null,
...>     age integer,
...>     lang text);
sqlite> INSERT INTO user ( name, age, lang )
...> VALUES('Fastcampus', 3, 'Python');
sqlite> .tables
user
sqlite> SELECT * FROM user;
1|Fastcampus|3|Python
sqlite> .exit
```

## SQLite - insert data

- `sqlite3.connect` 메소드를 이용해서 DB 파일에 연결한 후 'Connection' 객체를 생성한다.
- Connection 객체를 통해 Cursor 객체를 생성한다.
- 'Cursor' 객체의 `execute` 메소드를 통해서 query를 실행한다.
- 'Connection' 객체의 `commit`을 이용하여 변경된 내용을 commit한다.
- DB와의 연결을 닫는다.

# Small Project

Flask와 sqlite를 사용해 동아리 주소록 만들기

Back-end: Flask

Database: sqlite

Front-end: HTML

# Advanced Web Crawling

# Selenium

# Web Crawling with Selenium

```
$ pip install selenium
```

# Web Crawling with Selenium

```
import os
import math
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

ch_driver = webdriver.Chrome()
ch_driver.get("https://www.data.go.kr/")
```

# Web Crawling with Selenium

```
input_area = ch_driver.find_element_by_id("search-key")
input_area.send_keys("지하철")
#from selenium.webdriver.common.keys import Keys
#input_area.send_keys(Keys.ENTER)
search_button = ch_driver.find_element_by_class_name(
    'search-btn')

search_button.click()
ch_driver.implicitly_wait(1)
```

# Web Crawling with Selenium

```
ch_driver.execute_script("doIndexSearch('DATA');")
n = math.ceil(int(ch_driver.find_element_by_xpath(
    "//*[@id='sub-main']/div[4]/div[2]/
    div[2]/div[1]/div/div[1]/span").text) / 10)
print(n)

RESULTS_LOCATOR = "//*[@id='file-list-wrapper']/div/div[1]/a"
WebDriverWait(ch_driver, 10).until( \
    EC.visibility_of_element_located((\
        By.XPATH, RESULTS_LOCATOR)))

results = ch_driver.find_elements(By.XPATH, RESULTS_LOCATOR)
for item in results:
    print(item.text, item.get_attribute("href"))
```

# Web Crawling with Selenium

```
def get_data():
    RESULTS_LOCATOR = \
        "//*[@id='file-list-wrapper']/div/div[1]/a"

    WebDriverWait(ch_driver, 10).until( \
        EC.visibility_of_element_located((By.XPATH, \
            RESULTS_LOCATOR)))

    results = ch_driver.find_elements(By.XPATH, RESULTS_LOCATOR)
    for item in results:
        print(item.text)

for i in range(1,n+1):
    print("page",i)
    ch_driver.execute_script(
        "javascript:doPaging({});".format(i))
    get_data()
```