

Fastcampus

Computer Science SCHOOL

Database Basic

# 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

CURSOR	<-- -->	UP	DOWN	DELETE	Insert Mode: Ins
Char:	< >	Field: ↑	↓	Char: Del	Exit/Save: ^End
Word:	Home End	Page: PgUp	PgDn	Field: ^Y	Abort: Esc
		Help: F1		Record: ^U	Memo: ^Home

FIRSTNAME	Claire
LASTNAME	Buckman
ADDRESS	8307 Santa Anita Blvd
CITY	Oxnard
STATE	CA
ZIPCODE	93034
PHONE	(555)456-9059

EDIT	<C:>	CLIENTS	Rec: 1/49
------	------	---------	-----------

# 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

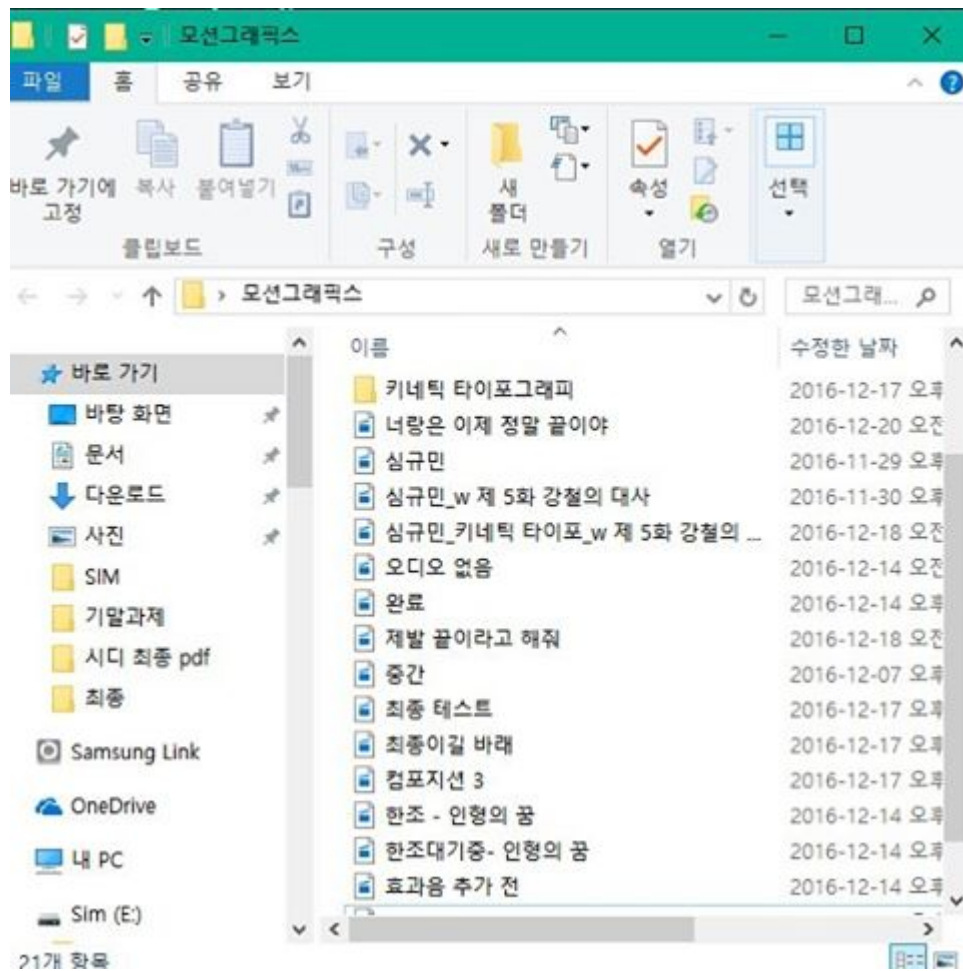
- .hwp -> 한글
- .doc -> Microsoft Word
- .xls -> Microsoft Excel

DB

- Only SQL(RDBMS)







# SQL (Structured Query Language)

데이터 관리를 위해 설계된 특수 목적의 프로그래밍 언어

The diagram illustrates the components of an SQL UPDATE statement. It shows the following code with annotations:

```
UPDATE clause [UPDATE country  
SET clause [SET population = population + 1  
WHERE clause [WHERE name = 'USA';
```

Annotations and groupings:

- A bracket on the right groups the three clauses under the label "Statement".
- A bracket above "population + 1" is labeled "Expression".
- A bracket below "name = 'USA'" is labeled "Expression".
- A bracket below the "WHERE" clause is labeled "Predicate".

# 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 처리시 저하	찾은 수정시 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

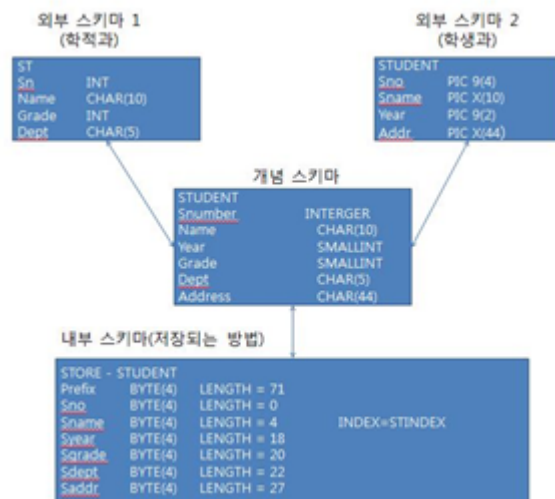
## 개념 스키마

- 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

## 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.
- Connection객체를 통해 Cursor 객체를 생성한다.
- 'Cursor' 객체의`execute` 메소드를 통해서 query를 실행한다.
- 'Connection' 객체의 `commit`를 이용하여 변경된 내용을 commit한다.
- DB와의 연결을 닫는다.



# Small Project

Flask와 sqlite를 사용해 동아리 주소록 만들기

Back-end: Flask

Database: sqlite

Front-end: HTML

# Beautiful Soup

## Web Scraping with Beautiful Soup

```
$ pip install beautifulsoup4
```

```
$ pip install beautifulsoup4
```

```
Collecting beautifulsoup4
```

```
  Downloading beautifulsoup4-4.5.1-py3-none-any.whl (83kB)
```

```
    100% |██████████████████████████████████████████████████████████████████████████████| 92kB 153kB/s
```

```
Installing collected packages: beautifulsoup4
```

```
Successfully installed beautifulsoup4-4.5.1
```

## Web Scraping with BeautifulSoup

```
$ pip list
DEPRECATION: The default format will switch to
the [list] section) to disable this warning.
beautifulsoup4 (4.5.1)
pip (9.0.1)
setuptools (20.10.1)
urllib3 (1.19.1)
```

# Web Scraping with BeautifulSoup

```
>>> import urllib
>>> from bs4 import BeautifulSoup
>>> html = """
... <html><head><title>The Dormouse's story</title></head>
... <body>
... <p class="title"><b>The Dormouse's story</b></p>
...
... <p class="story">Once upon a time there were three little sisters; and their names were
... <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
... <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
... <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
... and they lived at the bottom of a well.</p>
...
... <p class="story">...</p>
... """
>>> soup = BeautifulSoup(html, 'html.parser')
>>> print(soup.prettify())
```

# Web Scraping with BeautifulSoup

```
import urllib
from bs4 import BeautifulSoup
html = """

uglified html code

"""
soup = BeautifulSoup(html, "html.parser")
print(soup.prettify())
```

# Web Scraping with BeautifulSoup

```
curl https://www.rottentomatoes.com
```

```
import urllib.request
from bs4 import BeautifulSoup

url = "https://www.rottentomatoes.com"
html = urllib.request.urlopen(url)
source = html.read()
html.close()

soup = BeautifulSoup(source, "html.parser")
print(soup)

table = soup.find(id="Top-Box-Office")
print(table)
```

## Web Scraping with BeautifulSoup

```
all_tr = table.find_all("tr")

for tr in all_tr:
    all_td = tr.find_all("td")
    score = all_td[0].find("span", attrs={"class": "tMeterScore"})
    movie_name = all_td[1].a.text
    amount = all_td[2].a.text
    print(score, movie_name, amount)
```



# Web Scraping with BeautifulSoup

```
>>> import urllib.request
>>> from bs4 import BeautifulSoup
>>> url = "https://www.rottentomatoes.com"
>>> html = urllib.request.urlopen(url)
>>> source = html.read()
>>> html.close()
>>> soup = BeautifulSoup(source, "html.parser")
>>> table = soup.find(id="Top-Box-Office")
>>> all_tr = table.find_all("tr")
>>> for tr in all_tr:
...     all_td = tr.find_all("td")
...     score = all_td[0].find("span", attrs={"class": "tMeterScore"}).text
...     movie_name = all_td[1].a.text
...     amount = all_td[2].a.text
...     print(score, movie_name, amount)
```

# Web Scrapping with BeautifulSoup

```
...
69% Sing
                                $41.5M

95% Fences
                                $10.2M

40% Why Him?
                                $10.1M

16% Assassin's Creed
                                $8.1M

12% Collateral Beauty
                                $4.1M

73% Fantastic Beasts and Where to Find Them
                                $4.0M
```

So, Let's Scrap Naver

# Advanced Web Crawling

# Selenium

# Web Crawling with Selenium

```
$ pip install selenium
```

# Web Crawling with Selenium

```
import os
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
```

## Web Crawling with Selenium

```
ff_driver = webdriver.Firefox()
ff_driver.get("https://www.google.co.kr/")

query = ff_driver.find_element_by_id("lst-ib")
query.send_keys("스타크래프트")

ff_driver.find_element_by_name("btnK").click()
ff_driver.implicitly_wait(10)
```



# Web Crawling with Selenium

```
RESULTS_LOCATOR = "//div/h3/a"
WebDriverWait(ff_driver, 10).until(
    EC.visibility_of_element_located((By.XPATH, RESULTS_LOCATOR))
page1_results = ff_driver.find_elements(By.XPATH, RESULTS_LOCATOR)
for item in page1_results:
    print(item.text)
```