NoSQL

대용량 데이터를 저장 할 수 있는 단순한 형태의 데이터베이스

대표적인 오픈소스 소프트웨어 - mongoDB(document 스타일)

mongoDB - document DB

스키마 제약 없이 (k,v)의 형태로 데이터를 저장하는 비관계형 DBMS

저장되는 데이터는 반정형(json, xml 등)의 문서(document) 형태로 저장

모든 형태의 데이터 저장이 가능하다.

GrideFS - mongoDB안에 비정형 데이터를 저장할 수 있는 파일시스템

1개의 파일을 최대 2GB까지 저장 가능

각 파일은 chuck(=block) 단위로 저장 (1chuck=256kb)

chucks = chuck에 관한 정보

GridFS의 예시

mongodb 기본 명령어

```
use<DB명> - DataBase 생성
show dbs - 현재 생성되어 있는 DataBase를 보여준다.
db - 현재 사용하고 있는 DataBase를 보여준다.
db.createCollection - 컬렉션 생성 및 보기 (*컬렉션 = table)
show collections - 현재 생성된 collection을 보여준다.

db.컬렉션명.insert({Key:value}) - 해당 컬렉션에 데이터 삽입(컬렉션이 없어도 가능)
db.컬렉션명.find() - 해당 컬렉션에 저장된 값을 확인
  *db.컬렉션명.find({}, {확인할 key:1}) - 해당 key를 가지는 값만 보기

db.document명.remove({key:value}) - 해당 document 삭제
db.컬렉션명.drop() - 해당 collection 삭제
db.dropDatabase() - DataBase 삭제
```

mongodb - python 연동 기본절차

```
#1.test.txt 파일이 저장된 경로로 들어가 pip install pymongo를 실행해준다.
#라이브러리 적용
from pymongo import MongoClient
from gridfs import GridFS
from bson import objectid

#mongodb에 python_test데이터 베이스 접속
db = MongoClient().python_test
#해당 python_test라는 db가 없을 경우 자동으로 python_test라는 db를 생성해 줌

#python_test에 파일을 저장할 GridFS 객체 생성
fs = GridFS(db)

#c:\ai\workspace\mongodb\test.txt를 읽을 객체 f 생성
with open("c:/ai/workspace/mongodb/test.txt", 'rb') as f:
    fs.put(f, filename='test.txt')

#GridFS에 저장된 파일 조회
db.fs.files.find()
list(db.fs.files.find()) #리스트로 조회

#GridFS에 저장된 test.txt파일을 읽을 객체 f생성
f = fs.get_last_version(filename='text.txt')

#test.txt파일의 내용을 읽어서 data 저장
data = f.read()
data

#data에 저장된 내용을 utf-8로 인코딩하여 출력
data.decode('utf=8') #문자 -> 숫자 (encoding) <=> (decoding)

list(db.fs.files.find())
[{'_id': ObjectId('608f52085372321039ae27ed'), #objectid - 중복되지 않는 숫자
  'filename': 'test.txt',                      #파일명
  'md5': '0b97c2704cfb0e1892bb47a0b62bef96',
  'chunkSize': 261120,                         #256kb로 쪼갠 데이터를 byte단위로 보여줌
  'length': 35,                                #35kb
```

pymongo 이미지 파일 저장

```
#라이브러리 설치
from pymongo import MongoClient
from gridfs import GridFS
from bson import objectid
from gridfs import GridFSBucket

#mongodb python_test 데이터 베이스 접속
db = MongoClient().python_test

#GridFS에 파일을 저장할 객체 생성
fs = GridFS(db)

#pymongo 이미지 파일 저장
import urllib.request
#다운받을 이미지 파일 주소 가져오기
url = 'https://perfectacle.github.io/images/spring-boot-docker-image-optimization/thumb.png'

#이미지 파일 확장자 저장
image_type = url.split('.')[-1]  #'png'

#이미지의 타입 설정
content_type = 'image/{}'.format(image_tpye) #'image.png'

#이미지의 이름 리턴
image_name = url.split('/')[-1] #'thumb.png'

#이미지 소스를 불러와 객체에 저장
image = urllib.request.urlopen(url).read()

#이미지 파일을 저장할 객체 생성
bucket = GridFSBucket(db)

#이미지 파일을 GridFS에 저장할 객체 저장
```

```
grid_in = bucket.open_upload_stream(image_name, metadata={'ContentType':content_type})

#이미지의 내용을 GridFS에 저장
grid_in.write(image)

#이미지 저장 종료
grid_in.close()
```

mongoDB에 저장된 이미지 파일 출력하기

```
#기본 mongodb 접속 라이브러리 설치
from pymongo import MongoClient
from gridfs import GridFS
from bson import objectid
from gridfs import GridFSBucket

#이미지를 출력할 라이브러리 설치
from PIL import Image
from io import BytesIO
from IPython.display import display

#mongodb 접속
db = MongoClient().python_test #이미지 파일이 저장된 db 접속
fs = GridFS(db)

#mongodb의 이미지 파일의 인덱스 값을 불러와 저장
file_detail = list(db.fs.files.find()[2]) #이미지 파일이 2번 인덱스에 저장된 경우

#파일명 조회
file_detail['filename']

#피일명이 일치하는 마지막 파일의 내용을 객체에 저장
docker = fs.get_last_version(filename=file_detail['filename'])

#이미지 파일의 소스를 불러와 객체 저장
docker_source = docker.read()

#이미지 출력하기
image_docker = Image.open(BytesIO(docker_source))
image_docker #=display(image_docker)
```

네이버 API를 이용한 고양이, 강아지 이미지 가져오기

```
#네이버 API를 이용한 강아지, 고양이 이미지 불러오기
#불러온 이미지를 SVM 이용해 분류하기

#api를 이용한 고양이, 강아지 사진 가져오기
#https://openapi.naver.com/v1/search/image - 네이버 json 요청 변수 URL

#라이브러리 설치
import requests

#네이버 API 인증 정보(로그인)
client_id = "My Client Id"
client_secret = "My Client Secret"
login = {'X-Naver-Client-Id':client_id, 'X-Naver-Client-Secret':client_secret}

#네이버 고양이 이미지 검색 실행주소 변수 설정 (기본적으로 open API ID, PW가 필수적으로 필요함)
url = 'https://openapi.naver.com/v1/search/image?query=고양이'

#url 실행 - 200 - 성공 / 그 외 - 실패
result = requests.get(url, headers=login)  #성공 시 다음으로

#네이버 open API 접속 후 검색하는 기능 함수 정의
def get_api_result(keyword, display, start):
    url = "https://openapi.naver.com/v1/search/image?query=" + keyword \
        + "&display=" + str(display) \
        + "&start=" + str(start)
    result = requests.get(url, headers=login)
    return result.json()

#키워드에 맞는 이미지 링크만을 불러오는 함수 정의
```

```
def call_and_print(keyword, total_page=10):
    link_list = []
    for page in range(1, total_page+1):
        display = 100
        start = ((page-1)*display)+1
        json_obj = get_api_result(keyword, display, start)

        for item in json_obj['items']:
            link_list.append(item['link'])
    return link_list

#고양이 이미지 불러오기
keyword='고양이'
link1=call_and_print(keyword)

#강아지 이미지 불러오기
keyword='강아지'
link2=call_and_print(keyword)

#mongoDB에 이미지 파일 저장하기

#mongodb 라이브러리 임포트
from pymongo import MongoClient
from gridfs import GridFS
from bson.objectid import ObjectId
from gridfs import GridFSBucket

#mongodb DataBase 접속
db = MongoClient().python_test

#GridFS 파일을 저장할 객체 생성
fs = GridFS(db)

#이미지 파일을 저장할 GridFSBucket 객체 생성
bucket = GridFSBucket(db)

import urllib.request

#link1 - 고양이 이미지를 다운로드 할 url 1000개가 저장된 리스트
for url in link1:
    try:
        image = urllib.request.urlopen(url).read()   #url에서 고양이 이미지를 다운로드하여 image 변수에 저장
        image_name = url.split('/')[-1]              #split 후 파일명.확장자 추출
        if image_name.find('?') != -1:               #image에 ?가 있으면 ?를 기준으로 split 후 확장자 추출
            image_name = image_name.split('?')[0]
        image_type = 'jpg'                           #이미지의 기본 타입
        if image_name.find('.') != -1:
            image_type = image_name.split('.')[-1]   #image_name(파일명.확장자)에서 .을 기준으로 split 후 확장자 리턴
        content_type = 'image/{}'.format(image_type) #확장자content_type에 대입
        grid_in = bucket.open_upload_stream(image_name, metadata={'ContentType':content_type,'Type':'cat'}) #이미지 파일 정보 저장
        grid_in.write(image)  #이미지 내용 GridFS에 저장
        grid_in.close()       #이미지 저장 종료
        print('image_type :',image_type, 'content_type :',content_type, 'image_name :',image_name)
    except:
        print('Error')

#link2 - 강아지 이미지를 다운로드 할 url 1000개가 저장된 리스트 (위 과정과 동일)
for url in link2:
    try:
        image = urllib.request.urlopen(url).read()
        image_name = url.split('/')[-1]
        if image_name.find('?') != -1:
            image_name = image_name.split('?')[0]
        image_type = 'jpg'
        if image_name.find('.') != -1:
            image_type = image_name.split('.')[-1]
        content_type = 'image/{}'.format(image_type)
        grid_in = bucket.open_upload_stream(image_name, metadata={'ContentType':content_type,'Type':'Dog'})
        grid_in.write(image)
        grid_in.close()
        print('image_type :',image_type, 'content_type :',content_type, 'image_name :',image_name)
    except:
        print('Error')
```

저장된 고양이, 강아지 이미지 지도학습

```
#mongoDB에 저장된 이미지 일기
from PIL import Image
import urllib.request
from io import BytesIO
import numpy as np

image_list = list(db.fs.files.find())
image_list

#이미지의 가로 세로 크기를 설정할 변수 생성
im_width = 200
im_height = 200

#mongoDB에서 불러온 이미지를 저장할 리스트 생성
images = []
labels = []

#불러온 이미지를 수치화하여 array에 저장
for file_detail in image_list:
    file_name = file_detail['filename']
    f = fs.get_last_version(filename=file_name)
    data = f.read()
    if len(data)>0:
        im = Image.open(BytesIO(data))  #data(이미지소스)를 이미지로 변환해서 im에 저장
        im = im.convert('L')  #이미지를 흑백으로 저장
        im = im.resize((im_width, im_height)) #크기 설정
        im = np.array(im).flatten()/255.0 #이미지를 2차원 배열로 변환 / flatten() - 2차원 배열을 1차원으로 변환(svm은 한줄짜리만 모델링이 가능하기 때문)
        im = im.astype('float32') #배열을 float으로 변환
        images.append(im) #최정으로 변화된 이미지 데이터 삽입
        label = 1 if 'cat' in file_detail['metadata']['Type'] else 0  #file_detail['metadata']['type']가 cat이면 1 아니면 0
        labels.append(label)

images_arr = np.array(images)
labels_arr = np.array(labels)

#학습을 위한 train, test split
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(images_arr, labels_arr)

#svm을 이용한 학습
from sklearn import svm
```

Cassandra - Key-ValueDB

대표적인 Key-Value 형태의 NoSQL 소프트웨어

읽기, 쓰기가 매우 빠르다는 것이 특징 (검색엔진을 사용하는 기업에서 주로 사용)