

종합설계 1팀 넋지 핵심 기술 발표

윤형로(팀장), 표승수, 정기욱, 황선기, 김영주, 이민기

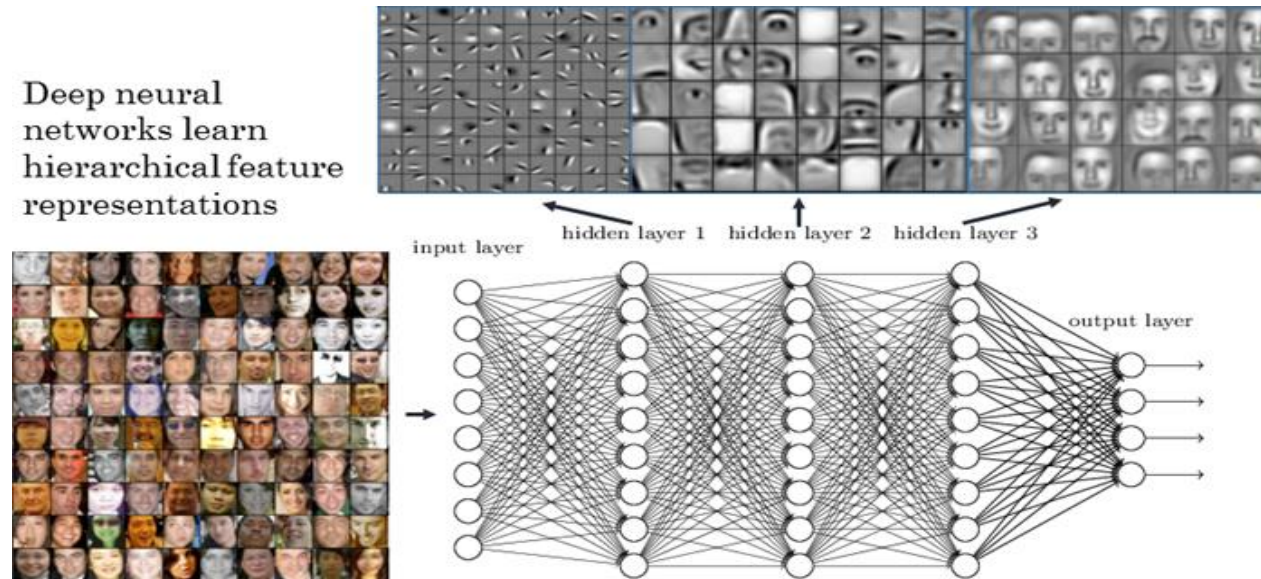
〈 목차 〉

1. 이미지 학습을 위한 원리
2. On Device ML Model & ML Model Demo Test
3. Database 설계1
4. Database 설계2
5. Django RESTful API서버 & MySQL 연동
6. AWS 구축 및 Django, MySQL Setting

이미지 학습을 위한 원리

201403672 황선기

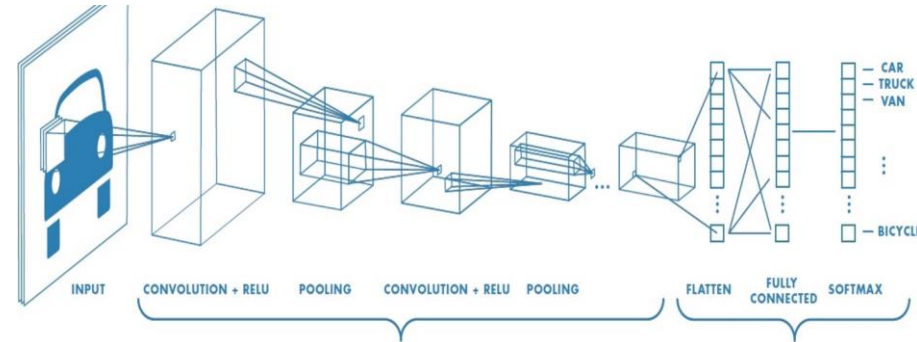
■ 머신러닝 - CNN



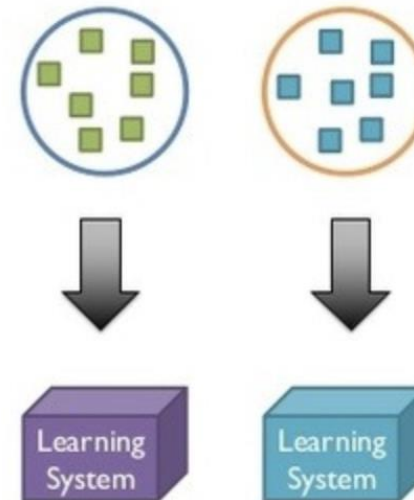
- 인간의 뇌가 사물을 인식하는 레이어를 모방하여 컴퓨터가 똑같이 인식할 수 있게 하는 방법 ex) 숫자 8을 인식하는 경우

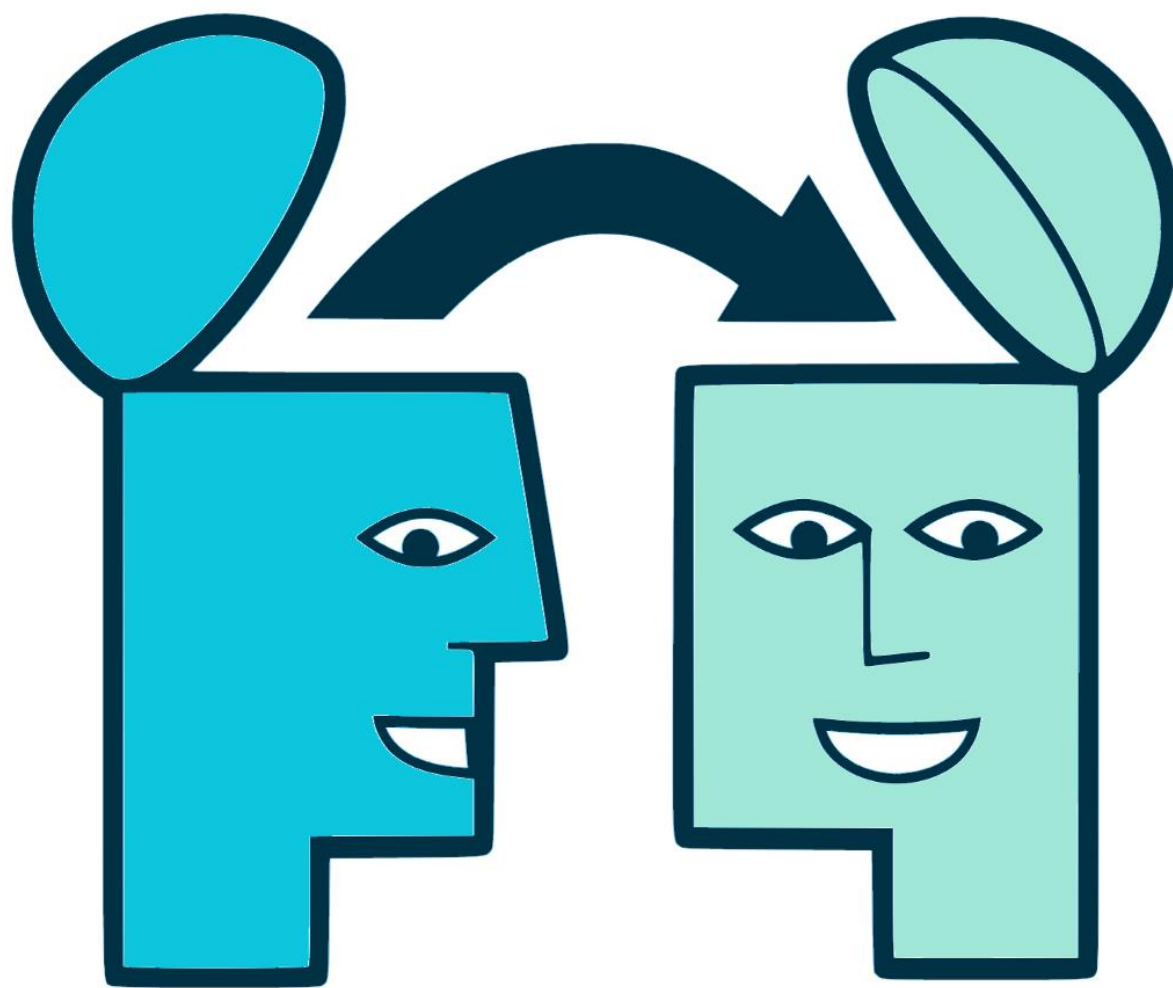
■ 전통적인 머신러닝 학습을 사용하지 않은 이유

- CNN에 여러 레이어를 거쳐 학습 시키면 시간이 많이 걸린다.
- 데이터의 양이 많이 필요하다.
- 하드웨어가 필요하다.(GPU, CPU, RAM)
- 학습이 독립적이다. 새로운 모델을 만들기 위해서는 많은 소요가 일어난다.
- 정확도 측면에서 transfer learning보다 낮다.

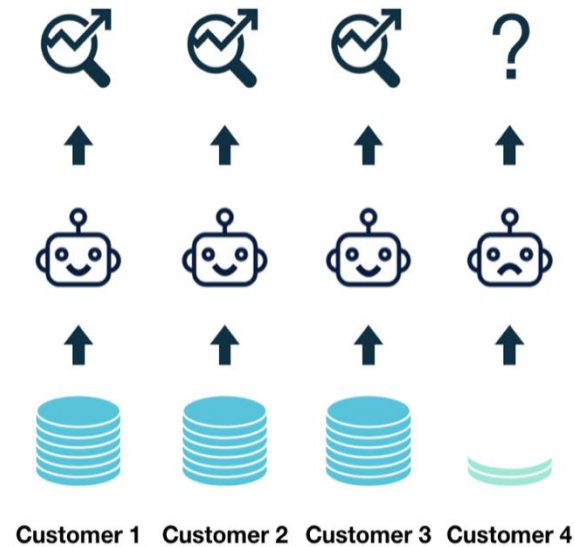


Traditional Machine Learning (ML)

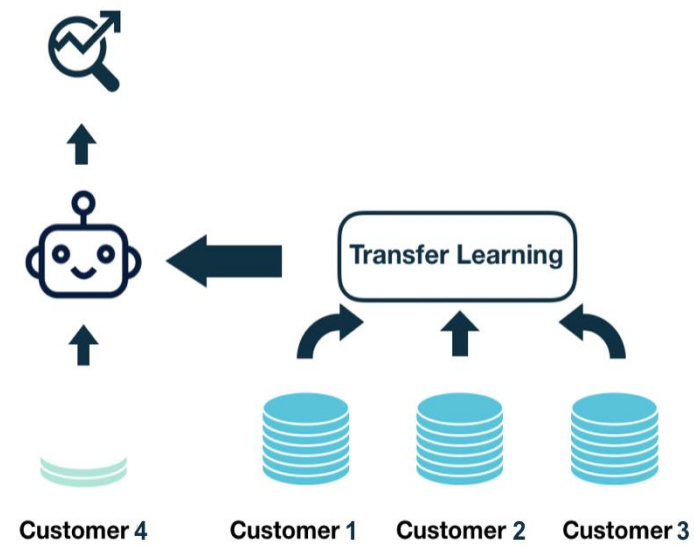




▪ Transfer learning



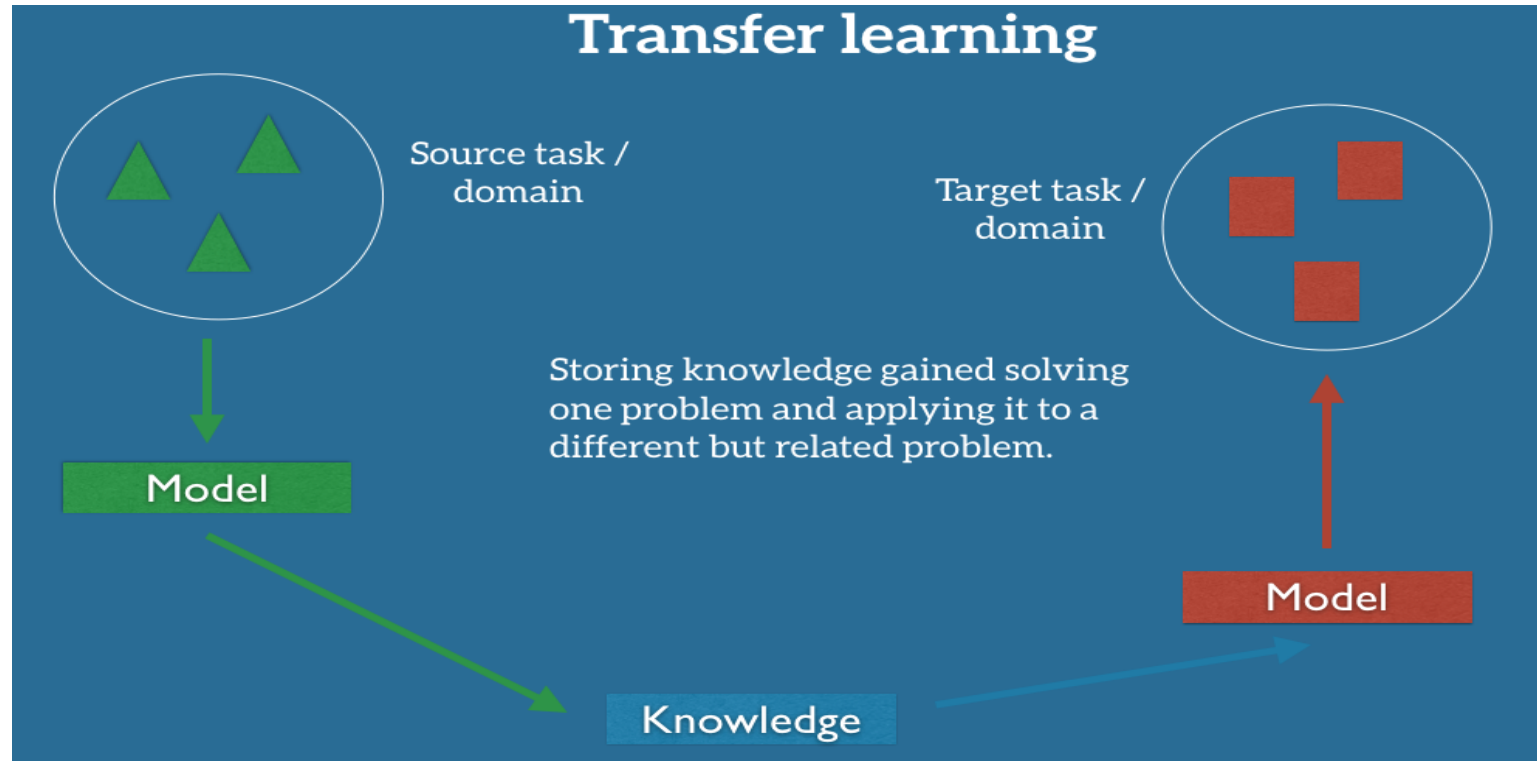
Without Transfer Learning



With Transfer Learning

- 이전에는 독립적으로 모델 학습을 진행하였다.
- 하지만 전이 학습을 통하여 적은 데이터 셋을 사용하더라도 새로운 데이터를 인식할 수 있다.

- Pre-trained ML model을 사용해서 새로운 모델을 만들어 내는 기법

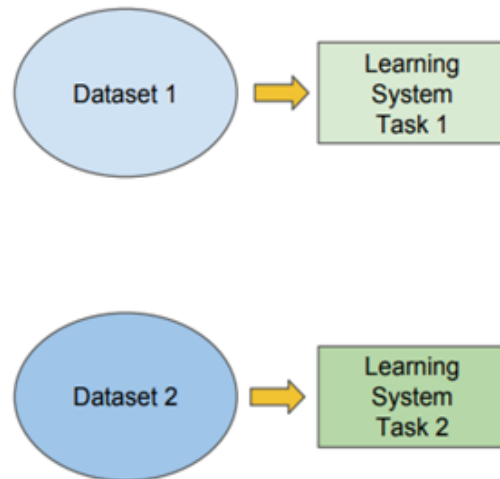


- 미리 만들어진 ML 모델에서부터 그 지식을 활용하여 새로운 도메인 영역을 구별한다.
- Ex) 카페 레시피 머신 러닝 모델의 지식을 이용하여 레스토랑 레시피 머신 러닝 모델 만들기

Transfer Learning Comparison

Traditional ML

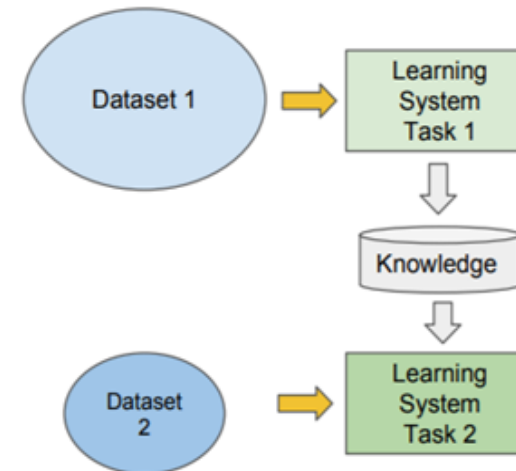
- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



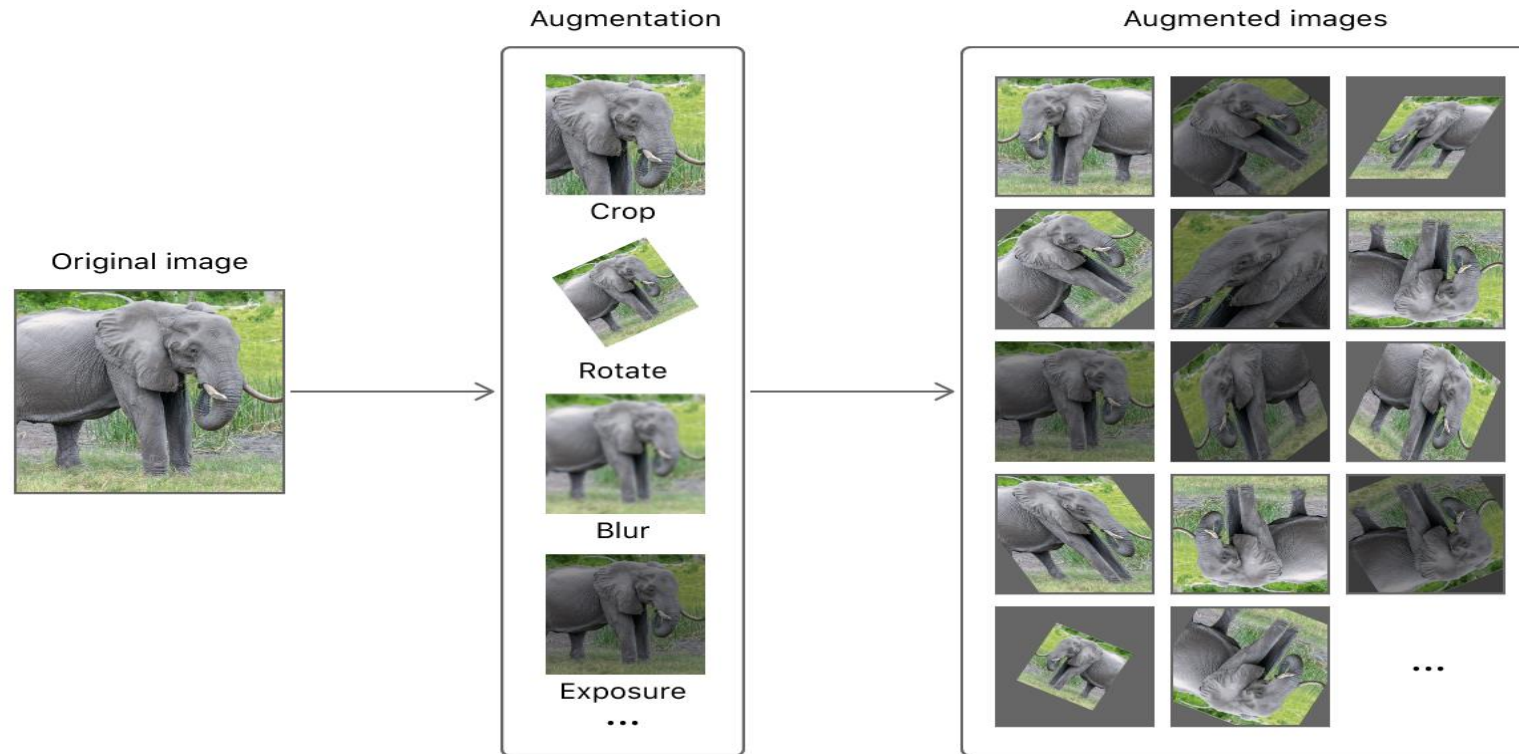
vs

Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data

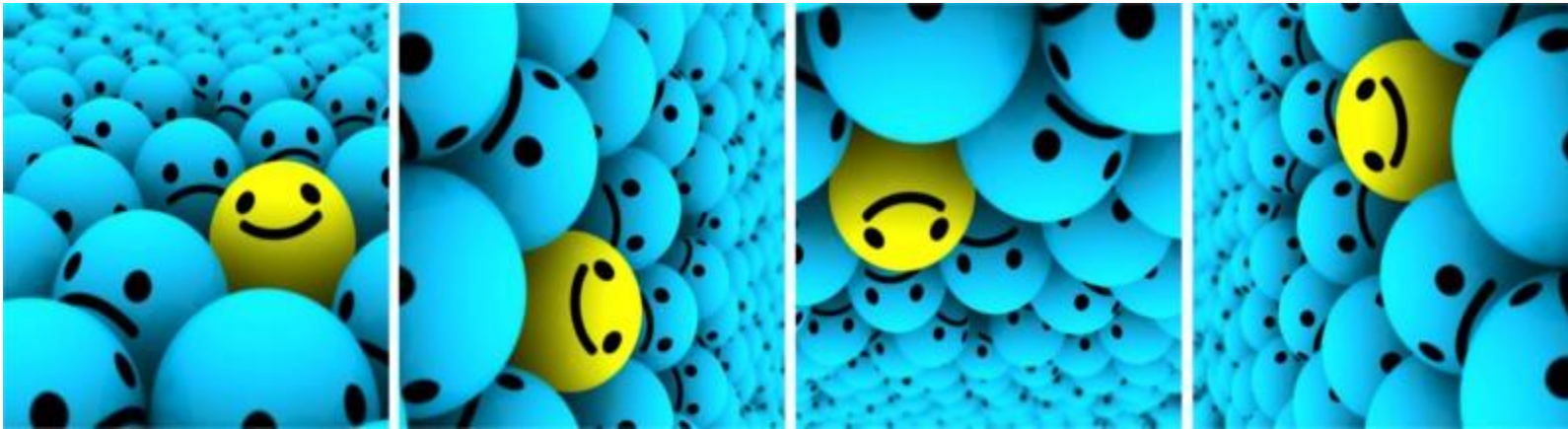


■ Data Augmentation



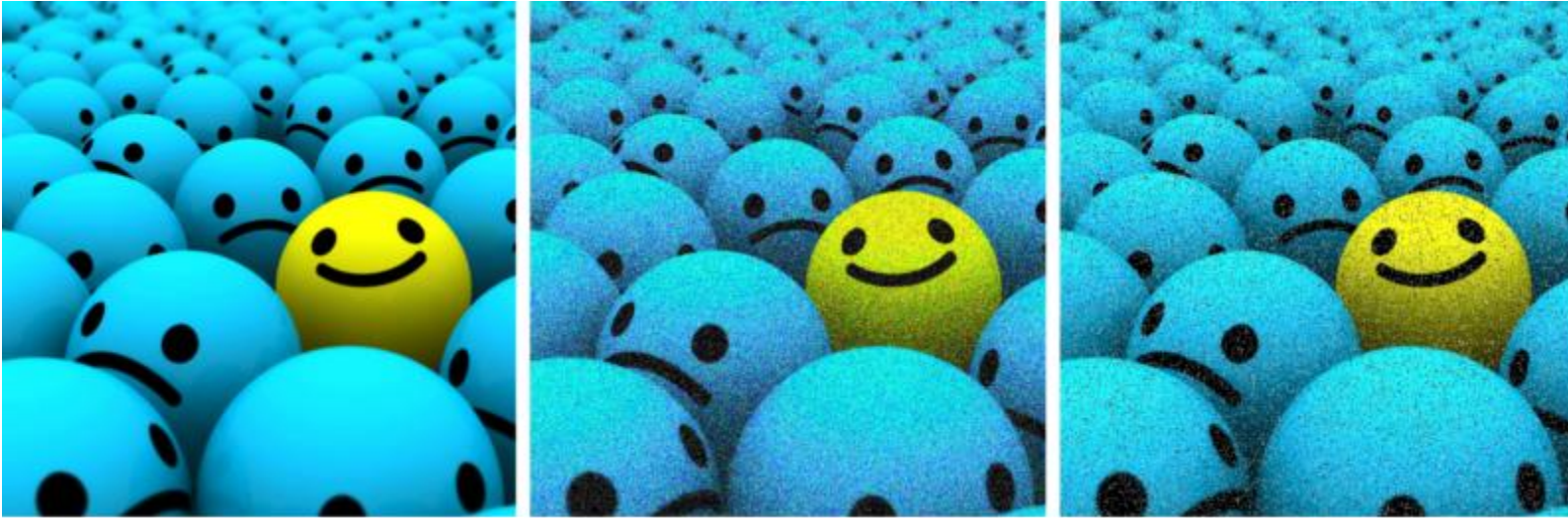
- 하나의 이미지는 다양한 조건으로 존재할 수 있다.
- 냉장고의 경우 찍는 각도와 어둑기에 따라서 상황이 달라진다.

■ Rotation



- 위 사진은 경우는 직각으로 회전된 정사각형의 모습입니다.
- 세밀한 각도에 따라 이미지가 달라집니다.
- 냉장고의 경우 직는 각도에 따라 다양한 이미지가 존재합니다.

■ Exposure



- 위의 이미지는 적절히 노이즈를 준 이미지이다.
- 냉장고의 상황과 짝는 상황에서 일어나는 경우를 보강해준다.
- 어두울 수 있고 깨질 수가 있다.

■ 시험 결과

▼ Machine Learning Model

Name Advanced

Type Image Classifier

Size 197 KB

Author 정기욱

Description A model trained using CreateML

License Unspecified

▼ Model Class

 Advanced ➕

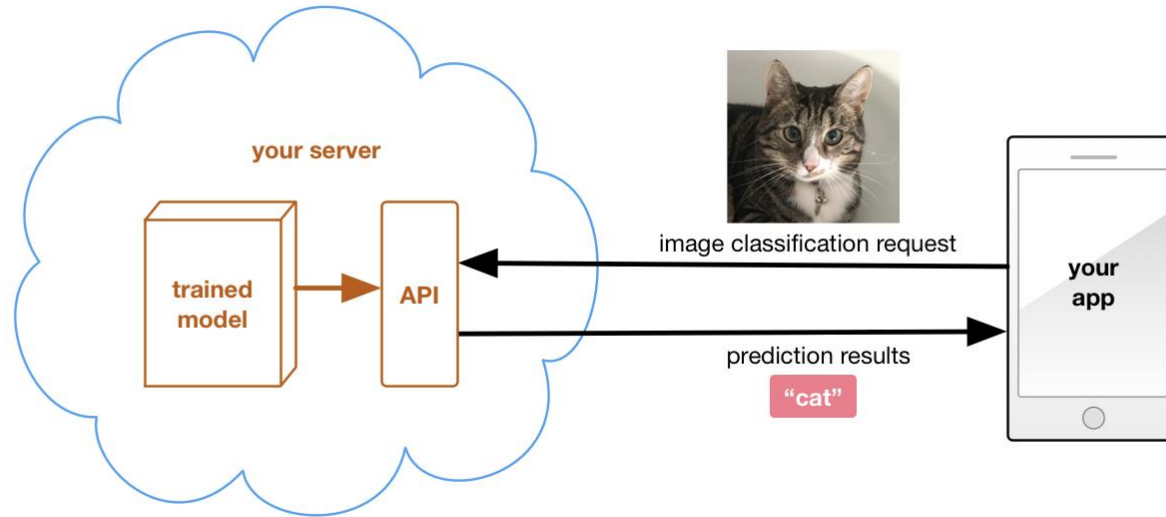
Automatically generated Swift model class

- Apple이 제공하는 머신 러닝 모델을 활용했습니다.
- 기존의 지식을 활용하여 실제 추가적인 카테고리를 만들었습니다.

On Device ML Model & ML Model Demo Test

201303063 정기욱

▪ Inference on the server



- 머신 러닝 모델을 서버에 두고 클라이언트 앱과 서버가 통신하는 방법
- 객체를 카메라를 통해 인식시킨 후 Restful API 형태로 HTTP 요청을 주고 응답을 받음 (Request & Response)
- 서버는 클라이언트 앱에게 영상을 받은 뒤 분석 한 이 후 응답으로 Label(물체의 이름)과 Confidence(정확도)를 보내줌

▪ Upsides & Downsides

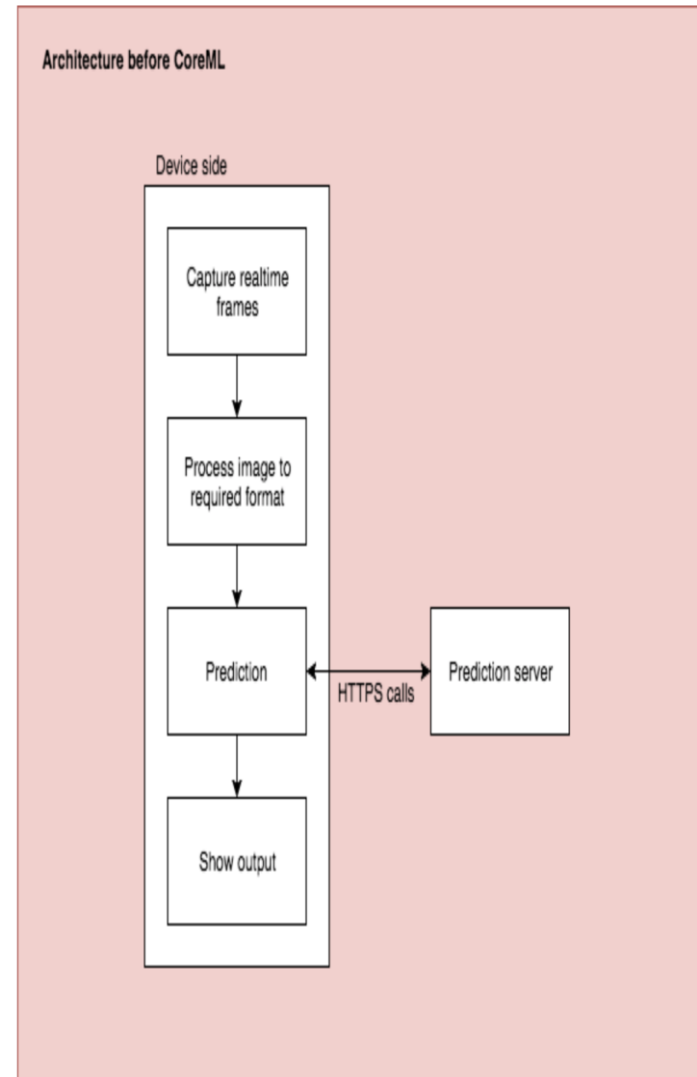
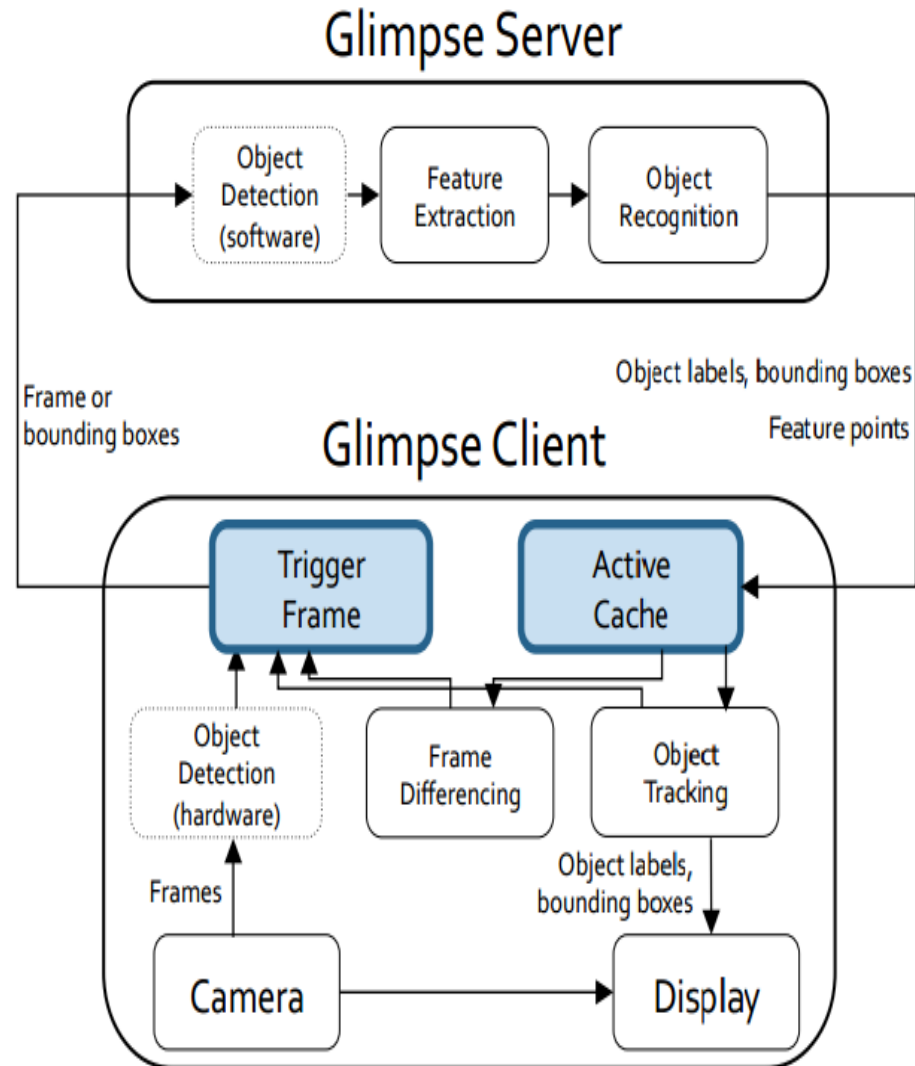
Upsides

- 머신 러닝 모델이 서버 측에 있어서 추가학습에 있어서 클라이언트 앱에 영향을 끼치지 않고 이뤄질 수 있다. (사용자는 추가 학습된 모델을 위해 앱을 업데이트 할 필요 없다.)
- 서버에 존재하는 머신 러닝 모델은 다양한 플랫폼과 결합하기 쉽다.
(Web, Android, iOS 등)

Downsides

- 반드시 네트워크 연결환경(4G, Wifi 등) 이 필요하다.
- 서버에 대한 관리가 필요하다.(해커의 공격, 서버 다운관리, 보안 문제 등)
- 클라이언트 앱을 많은 사람이 사용할 경우 서버 측에 과부하가 될 수 있다. (동영상 http 전송)
- 요청을 보내고 응답을 받는 방식으로 사용자가 기다려야하는 불편함을 겪을 수 있다.

■ Architecture



■ Demonstration



- 두잉 랩에서 만든 '다이어트 카메라' 어플
- 이 어플은 사진을 찍어 HTTP 요청을 서버로 보내주기 때문에 객체를 인식하는데 약 2~3초의 시간이 걸림
- Real time Object Recognition(실시간 객체 인식)의 경우 동영상으로 인식 시켜야 하기 때문에 더 많은 소요가 예상됨.

- Inference on the device



- 머신 러닝 모델을 클라이언트 앱에 내장해서 그 모델을 기반으로 디바이스의 CPU와 GPU를 사용한 연산으로 물체를 추론하는 방법
- 서버에 카메라로 인식한 객체에 대한 요청을 보내지 않음.
(객체를 인식한 뒤 얻은 라벨을 서버로 요청을 보냄)
- 서버는 디바이스에서 추론된 라벨 요청을 받아서 레시피를 찾아 다시 디바이스에게 응답을 보내줌.

▪ Upsides & Downsides

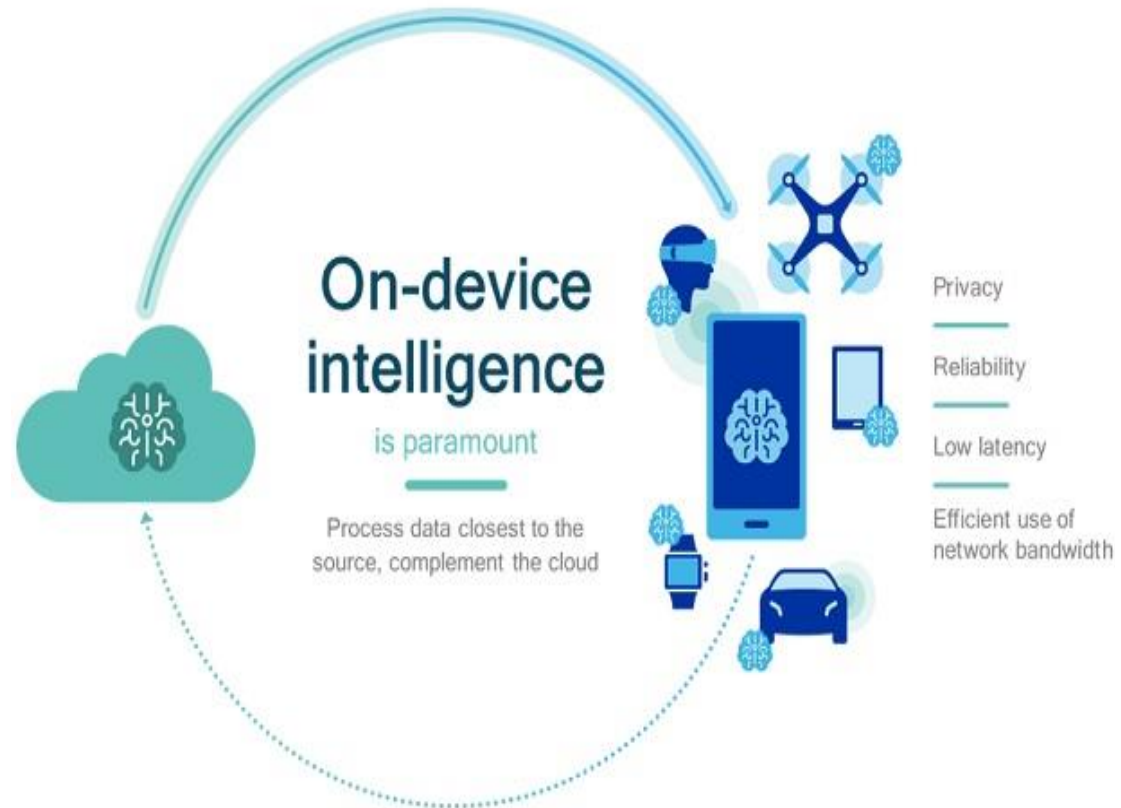
Upsides

- 객체 인식에 통신이 필요하지 않아 빠르다.
- 객체를 인식하는 과정까지는 네트워크가 필요 없다.
- 통신에 대한 부담이 없기 때문에 서비스가 커졌을 때, 서버 관리에 부담이 적다.
- 마찬가지로, 서버에 ML모델이 집중된 방식이 아니기 때문에 서버 비용 문제에서 비교적 자유롭다.

Downsides

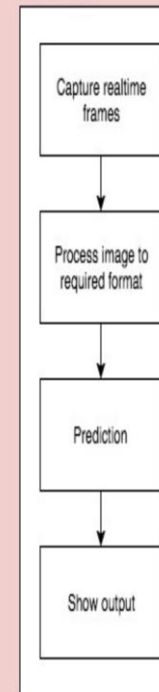
- 앱 사이즈가 커질 수 있다.
- 새로운 모델에 대한 업데이트는 사용자가 지속적으로 업데이트를 해줘야 한다.
- iOS와 Android처럼 다른 플랫폼은 각각의 프레임워크를 사용해 머신 러닝 모델을 넣어줘야 한다.
- 디바이스의 CPU를 사용한 연산이기 때문에 일정 수준의 배터리가 필요하다.

■ Architecture



Architecture with CoreML

Device side



▪ What to infer?

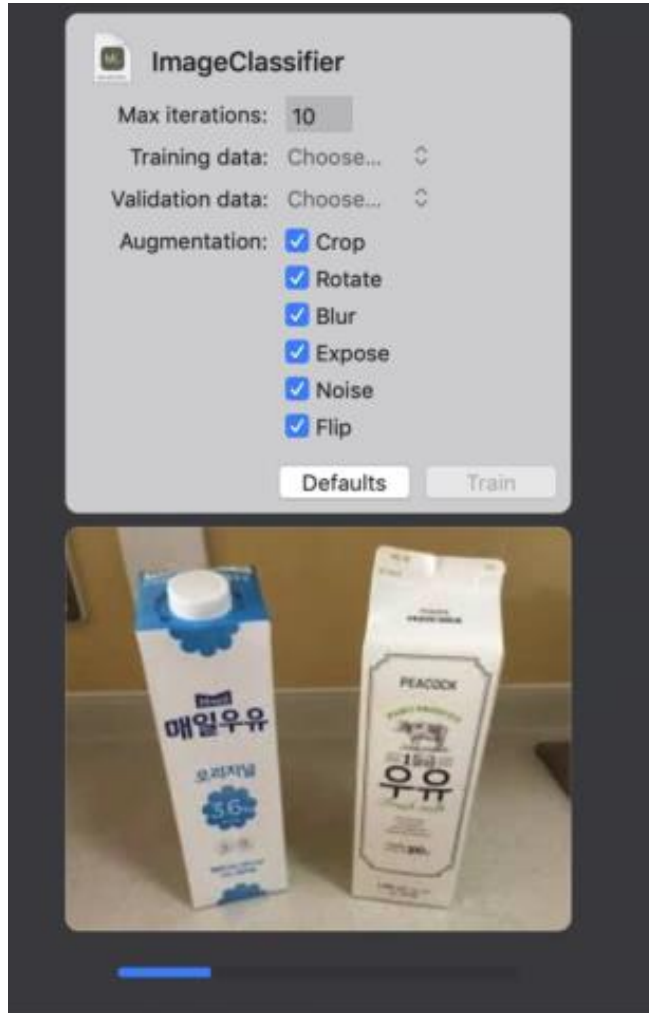
Can

- 상표가 붙은 기성품
- 냉장고 속에서 육안으로 확인 할 수 있는 재료
- 투명한 저장용기에 담긴 김치와 반찬
- 불투명한 저장용기에 담겼지만 카메라 각도를 조정하거나 뚜껑으로 식별할 수 있는 재료

Can't

- 불투명한 용기에 담긴 재료
- 제 3자가 육안으로 식별했을 때 인지 할 수 없는 재료
- 주재료가 아닌 양념과 같은 첨가물, 냉장보관 하지 않는 소스 (간장, 설탕, 소금 및 각종 소스류)
- 일반적으로 많이 사용하지 않는 식재료 (삭스핀, 달팽이 등등..)

■ ML Model Training



➤ 13개의 카테고리(우유, 김치, 된장, 돼지고기, 소고기, 양파, 대파, 갈치, 고등어, 달걀, 닭 등등)

➤ 카테고리당
각 11장의 사진

➤ 6가지의 Data
Augmentation

➤ 10번의 반복 횟수

➤ $13 * 11 * 10 * 6$

➤ 대략 8500장의
이미지를 Training

- **ML Model Training**

Raw Images Processed	Augmented Images	Elapsed Time	Percent Complete
VPA info: plugin is INTEL, AVD_id = 1080020, AVD_api.Create:0x1171bbcf1			
1	65	7.86s	0.5%
2	130	16.98s	1.25%
3	195	23.31s	2%
4	260	32.10s	2.75%
5	325	40.85s	3.5%
7	455	56.27s	4.75%
8	520	1m 5s	5.5%
9	585	1m 13s	6.25%

•

•

•

133	8645	17m 32s	94.25%
134	8710	17m 38s	95%
135	8775	17m 44s	95.5%
136	8840	17m 50s	96.25%
137	8905	17m 56s	97%
138	8970	18m 2s	97.75%
139	9035	18m 9s	98.5%
140	9100	18m 17s	99.25%
141	9165	18m 24s	100%

-----+-----+-----+-----+

Extracting image features from validation data set.
Analyzing and extracting image features.

- 약 9000장의 이미지를 학습하는데 20분 남짓한 시간이 소요됨.
- 즉, 한 카테고리당 1분 30초 내외에 시간소요 (13개 카테고리 학습)

■ ML Model Training Result

```
*****PRECISION RECALL*****
-----
Class           Precision(%)  Recall(%)
Chicken         nan          0.00
Egg             100.00       100.00
Green Onion     83.33        100.00
Hairtail        75.00        60.00
Kimchi          100.00       100.00
Mackerel        33.33        80.00
Meat            83.33        100.00
Milk            50.00        100.00
Onion           55.56        100.00
Pork            44.44        80.00
Red Pepper Paste nan          0.00
Soybean         nan          0.00
Tofu            nan          0.00
```

- Training ML Model에 테스트 데이터를 넣은 결과 위와 같은 결과가 도출됨.
- 닭, 고추장, 된장, 두부는 적절한 학습이 되지 않음
(이 후, 개선점에서 이유 및 추가사항 설명)

■ Demonstration



- 위의 ML Model은 냉장고 환경을 고려하지 않고 만든 프로토 타입
- 실제 모델은 냉장고 환경에 맞춤 데이터를 사용 할 예정

■ Improvement Point



- 한 가지 카테고리에 통일되지 않은 학습을 시키면 학습되지 않음
- 해결책으로 각각 다른 카테고리로 취급하여 학습시킬 예정
(고추장, 기성품 고추장, 두부, 기성품 두부)
- 또한, Data Augmentation에서 냉장고 환경에 필요한 부분과 필요 없는 부분 그리고 반복횟수를 줄여 학습 시간을 조절 할 예정

Database 설계1

201403506 표승수

순서

1. 요구분석
2. 개념적 설계
3. 논리적 설계
4. 물리적 설계

1. 요구분석

- 1) 사용자는 자신의 냉장고에 있는 재료를 사진으로 찍으면, 재료가 데이터화 되어 서버로 보낸다.
- 2) 보내진 재료를 토대로 서버는 만들 수 있는 메뉴 리스트를 뽑아서 준다.
- 3) 메뉴 리스트에는 메뉴 이름과 전체 재료, 레시피 그리고 그와 관련한 이미지들이 포함되어 있다.
- 4) 레시피에서 부족한 재료에 대해서는 사용자에게 배송업체로 연결해줄 수 있는 링크도 함께 보내준다.
- 5) 사용자가 이용한 레시피는 최근 10개까지 저장해둔다.

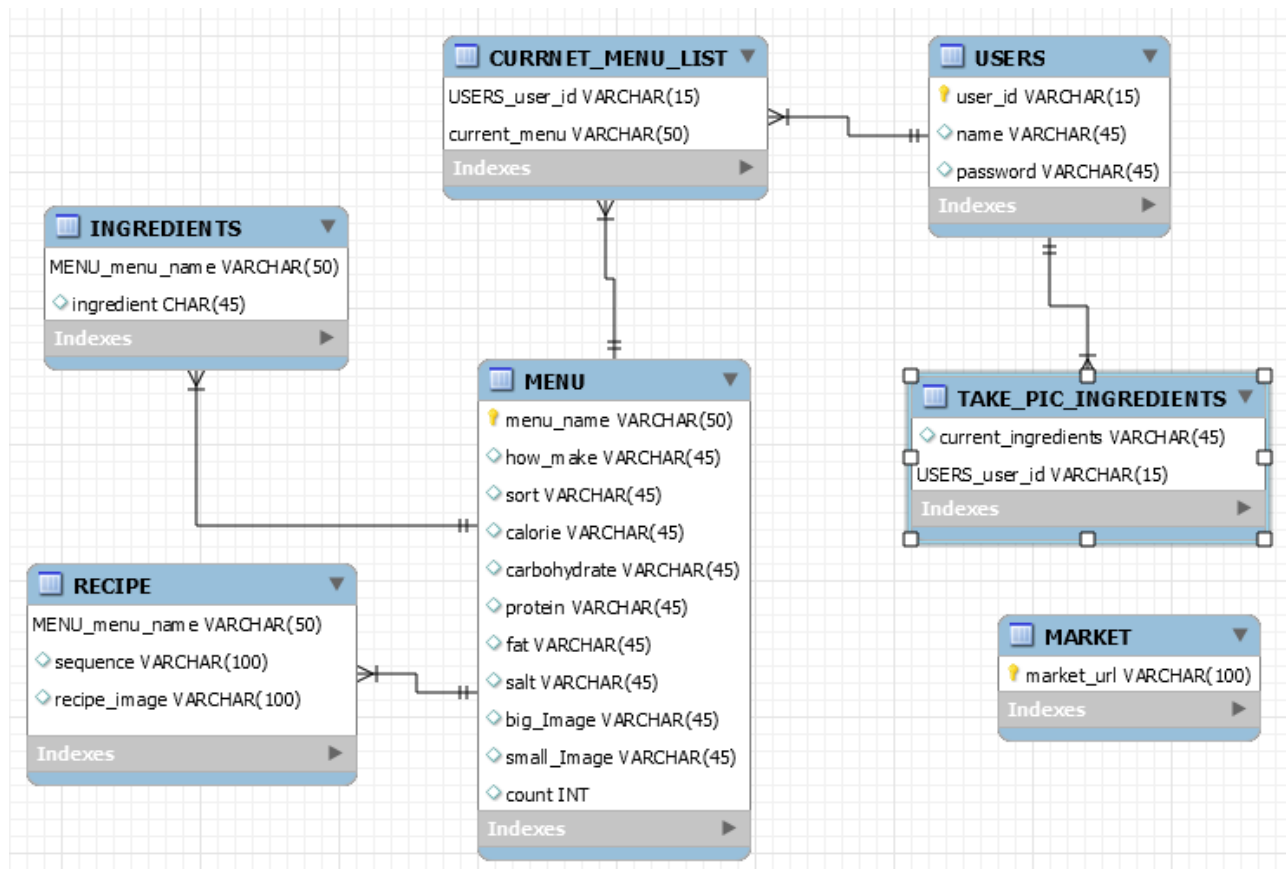
2. 개념적 설계

Entity	Attributes
사용자	사용자id, 사용자 이름
메뉴	메뉴이름, 요리 대분류, 중분류, 칼로리, 지방, 단백질, 음식 사진
재료	재료 이름
레시피	레시피 설명, 관련 사진
최근 이용한 메뉴(최근 10가지)	사용자 이름, 메뉴 이름
사용자가 전송할 재료	재료 이름
마켓	마켓url

3. 논리적 설계

- 사용자는 메뉴 테이블에 접근하여 여러 메뉴를 열람하고 최근 10개 메뉴를 저장해둔다.
->多 대多 관계
- 재료 테이블은 메뉴 테이블을 참조한다. -> 1대 多 관계
- 레시피 테이블은 메뉴 테이블을 참조한다.-> 1대 多 관계
- 사용자는 카메라를 이용해 인식한 재료를 잠시 저장하여, 가능한 메뉴들을 고를 수 있다.
->사용자와 사용할 재료 테이블은 1 대 多 관계

3. 논리적 설계



4. 물리적 설계(메뉴 테이블)

```
CREATE TABLE IF NOT EXISTS `mydb`.`MENU` (  
  `menu_name` VARCHAR(45) NOT NULL,  
  `how_make` VARCHAR(15) NULL DEFAULT NULL,  
  `sort` VARCHAR(15) NULL DEFAULT NULL,  
  `calorie` int NULL DEFAULT NULL,  
  `carbohydrate` int NULL DEFAULT NULL,  
  `protein` int NULL DEFAULT NULL,  
  `fat` int NULL DEFAULT NULL,  
  `salt` int NULL DEFAULT NULL,  
  `big_Image` VARCHAR(100) NULL DEFAULT NULL,  
  `small_Image` VARCHAR(100) NULL DEFAULT NULL,  
  `count` INT NULL, PRIMARY KEY (`menu_name`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

4. 물리적 설계(재료 테이블)

```
CREATE TABLE IF NOT EXISTS `mydb`.`INGREDIENTS` (  
  `MENU_menu_name` VARCHAR(45) NOT NULL,  
  `ingredient` CHAR(45) NULL,  
  PRIMARY KEY (`MENU_menu_name`),  
  INDEX `fk_INGREDIENTS_MENU1_idx` (`MENU_menu_name` ASC) VISIBLE,  
  CONSTRAINT `fk_INGREDIENTS_MENU1`  
  FOREIGN KEY (`MENU_menu_name`)  
  REFERENCES `mydb`.`MENU` (`menu_name`)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8;
```

4. 물리적 설계(레시피 테이블)

```
CREATE TABLE IF NOT EXISTS `mydb`.`RECIPE` (  
  `MENU_menu_name` VARCHAR(45) NOT NULL,  
  `sequence` VARCHAR(100) NULL DEFAULT NULL,  
  `recipe_image` VARCHAR(100) NULL,  
  PRIMARY KEY (`MENU_menu_name`),  
  CONSTRAINT `fk_RECIPE_MENU1`  
  FOREIGN KEY (`MENU_menu_name`)  
  REFERENCES `mydb`.`MENU` (`menu_name`)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE)  
  
ENGINE = InnoDB  
  
DEFAULT CHARACTER SET = utf8;
```

4. 물리적 설계(사용자 테이블)

- `CREATE TABLE IF NOT EXISTS `mydb`.`USERS` (
 `user_id` VARCHAR(15) NOT NULL,
 `name` VARCHAR(45) NULL,
 `password` VARCHAR(45) NULL,
 PRIMARY KEY (`user_id`))
ENGINE = InnoDB;`

4. 물리적 설계(최근 메뉴 리스트)

- ```
CREATE TABLE IF NOT EXISTS `mydb`.`CURRNET_MENU_LIST` (
 `USERS_user_id` VARCHAR(15) NOT NULL,
 `current_menu` VARCHAR(45) NOT NULL,
 PRIMARY KEY (`USERS_user_id`, `current_menu`),
 INDEX `fk_USERS_has_MENU_MENU1_idx` (`current_menu` ASC) VISIBLE,
 INDEX `fk_USERS_has_MENU_USERS1_idx` (`USERS_user_id` ASC) VISIBLE,
 CONSTRAINT `fk_USERS_has_MENU_USERS1`
 FOREIGN KEY (`USERS_user_id`)
 REFERENCES `mydb`.`USERS` (`user_id`)
 ON DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `fk_USERS_has_MENU_MENU1`
 FOREIGN KEY (`current_menu`)
 REFERENCES `mydb`.`MENU` (`menu_name`)
 ON DELETE NO ACTION ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

## 4. 물리적 설계(사용자가 사용할 재료)

```
CREATE TABLE IF NOT EXISTS `mydb`.`TAKE_PIC_INGREDIENTS` (
 `current_ingredients` VARCHAR(45) NULL,
 `USERS_user_id` VARCHAR(15) NOT NULL,
 PRIMARY KEY (`USERS_user_id`),
 CONSTRAINT `fk_TAKE_PIC_INGREDIENTS_USERS1`
 FOREIGN KEY (`USERS_user_id`)
 REFERENCES `mydb`.`USERS` (`user_id`)
 ON DELETE CASCADE
 ON UPDATE CASCADE)
ENGINE = InnoDB;
```

## 4. 물리적 설계(마켓url)

```
CREATE TABLE IF NOT EXISTS `mydb`.`MARKET` (
 `market_url` VARCHAR(100) NOT NULL,
 PRIMARY KEY (`market_url`))
ENGINE = InnoDB;
```

## 4. 물리적 설계(뷰)

```
-- View `mydb`.`MENU_INGREDIENTS` --
DROP TABLE IF EXISTS `mydb`.`MENU_INGREDIENTS`;
USE `mydb`;
CREATE OR REPLACE VIEW `MENU_INGREDIENTS` AS
select i.ingredient
from MENU m, INGREDIENT I
where m.menu_name = i.MENU_menu_name;
```

메뉴 테이블과 재료 테이블을 조인한  
MENU\_INGREDIENT 뷰를 만들어 DB에  
좀 더 쉽게 접근할 수 있다.

```
-- View `mydb`.`MENU_RECIPE` --
DROP TABLE IF EXISTS `mydb`.`MENU_RECIPE`;
USE `mydb`;
CREATE OR REPLACE VIEW `MENU_RECIPE` AS
select r.sequence, r.recipe_image
from MENU, RECIPE
where m.menu_name = r.MENU_menu_name;
```

메뉴 테이블과 레시피 테이블을 조인한  
MENU\_RECIPE 뷰를 만들어 DB에 좀 더  
쉽게 접근할 수 있다.



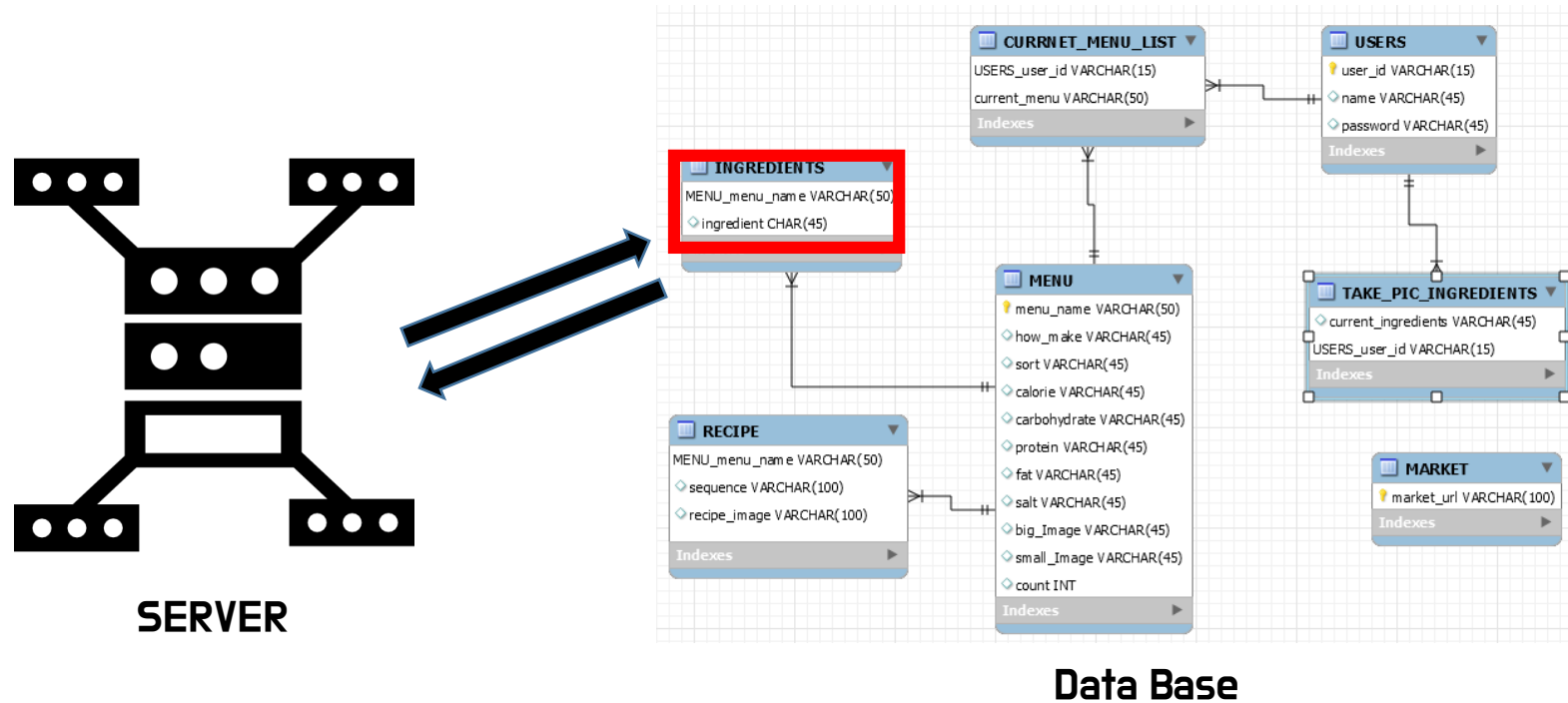
# Database 설계2

201402168 윤형로

## 순서

1. DB의 논리적 흐름도
2. RECIPE 테이블 구조 1
3. RECIPE 테이블 구조 2
4. 이상적 데이터 타입 JSON

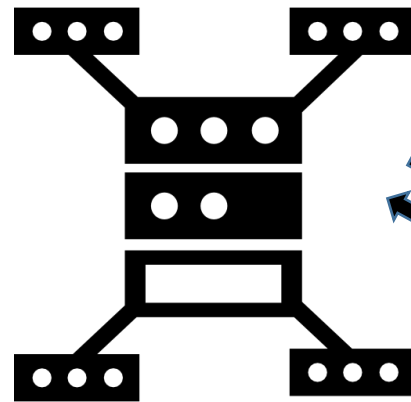
## ▪ DB의 논리적 흐름도



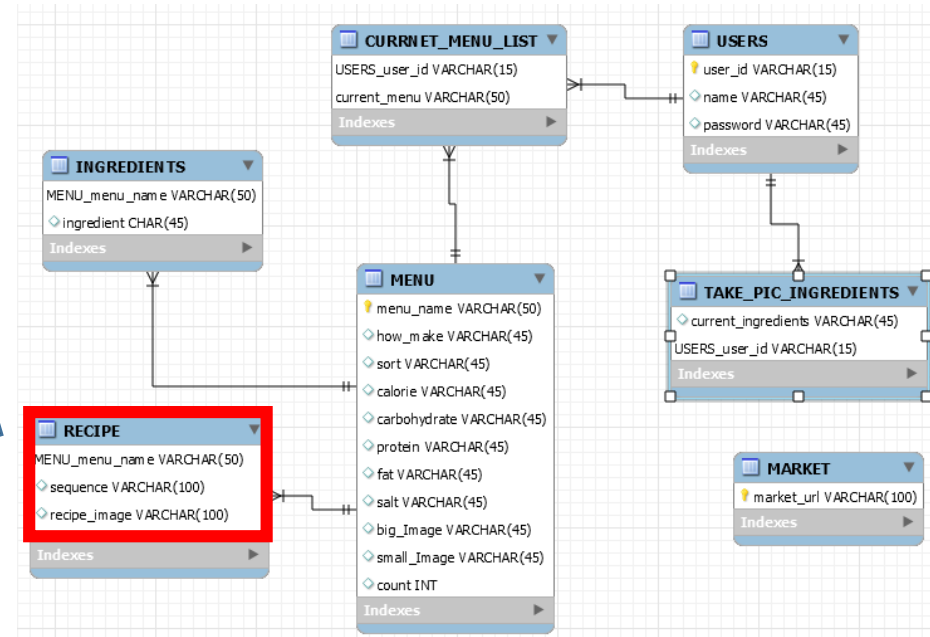
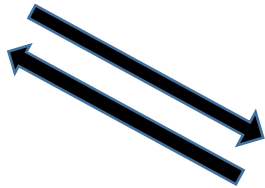
### 1. 서버에서 구현한 알고리즘을 통해 query 전달받음

(select menu\_name , ingredient from INGREDIENTS, MENU where ingredient = 사용자가 선택한 재료 )

## ▪ DB의 논리적 흐름도



SERVER



Data Base

2. 사용자가 선택한 Menu의 Recipe 에 대한 새로운 query문을 통해 데이터 전달받음

(Select \* from RECIPE.MENU where menu\_name = 된장찌개)

## RECIPE 테이블 구조 2가지

| MENU | RECIPE |
|------|--------|
| 된장찌개 | 레시피1   |
| 된장찌개 | 레시피2   |
| 김치찌개 | 레시피1   |
| 된장찌개 | 레시피3   |
| -    | -      |
| -    | -      |
| -    | -      |

**VS**

[illegible]

## ▪ RECIPE 테이블 구조 1

| MENU | RECIPE |
|------|--------|
| 된장찌개 | 레시피1   |
| 된장찌개 | 레시피2   |
| 김치찌개 | 레시피1   |
| 된장찌개 | 레시피3   |
| -    | -      |
| -    | -      |
| -    | -      |

- 장점 : Null값을 없게 하여 메모리를 절감

- 단점

1. 중복된 Tuple 값으로 인하여  
메모리 늘어남(Null값 보다는 적음.)

2. 데이터 탐색 시간이 늘어남

## ▪ RECIPE 테이블 구조 2

| Menu | RECIPE1 | RECIPE2 | RECIPE3 | RECIPE4 | RECIPE5 | RECIPE6 | RECIPE7 | RECIPE8 | RECIPE9 |
|------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 된장찌개 | 1.      | 2.      | 3.      | 4.      | 5       | Null    | Null    | Null    | Null    |
| 김치찌개 | 1.      | 2.      | 3.      | 4.      | 5.      | 6.      | 7.      | 8.      | 9.      |
|      |         |         |         |         |         |         |         |         |         |

- 장점 : 중복된 값이 없고 한 번에 데이터 서칭이 가능하다.
- 단점 : Null 값으로 인한 메모리 낭비가 심하다.

- 이상적 데이터 타입 Json
- MySQL 에서 사용이 가능
- 쿼리 안에서 함수를 사용, 입력, 삭제 수정 가능
- 인덱스 추가가 가능함 → 집합 단위로 들어온 데이터에 순서를 부여하여 데이터 서칭 속도 줄이고 Null값도 없어 메모리 줄어든다.

```
1 "COOKRCP01": {
2 "RESULT": {
3 "MSG": "정상처리되었습니다.",
4 "CODE": "INFO-000"
5 },
6 "total_count": "1198",
7 "row": [
8 {
9 "INFO_WGT": "",
10 "MANUAL01": "1. 고구마는 깨끗이 씻어서 껍질을 벗기고 4cm
11 정도로 잘라준다.a",
12 "MANUAL02": "2. 찜기에 고구마를 넣고 20~30분 정도 삶아
13 주고, 블렌더나 체를 이용하여 잘 으깨어 곱게 만든다.b",
14 "INFO_ENG": "205",
15 "MANUAL03": "3. 고구마와 물을 섞어 끓이면서 찹쌀가루로
16 농도를 맞추고 설탕을 넣어 맛을 낸다.c",
17 "MANUAL04": "4. 찹쌀 팬에 노릇하게 볶아 다져서 고구마
18 죽에 섞는다. 기호에 따라 고구마를 튀겨 얹어 먹어도 좋다
19 .",
20 "MANUAL05": "",
21 "ATT_FILE_NO_MK": "http://www.foodsafetykorea.go.kr
22 /uploadimg/cook/10_00017_1.png",
23 "MANUAL06": "",
24 "MANUAL07": "",
25 "MANUAL08": "",
26 "MANUAL09": "",
27 "RCP_PARTS_DTLS": "고구마죽\n고구마 100g(2/3개), 설탕 2g
28 (1/3작은술), 찹쌀가루 3g(2/3작은술), 물 200ml(1컵), 잣
29 8g(8알)"
30 }
31]
32 }
```

Json 타입으로 저장된 데이터

# Django RESTful API서버 & MySQL 연동

201400588 김영주



## ▪ Django REST Framework 란?

Django 안에서 RESTful API서버를 쉽게 구축할 수 있도록 도와주는 오픈 소스 라이브러리



- **URI** : 수행 대상이 되는 리소스를 정의. 슬래시(/)로 구분.
- **Method**: 리소스에 대한 행위를 정의.(GET, POST, DELETE 등)

GET /Food/

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
[
 {
 "Food_name": "계란찜",
 "ingredient1": "계란",
 "ingredient2": "대파",
 "ingredient3": "소금"
 },
 {
 "Food_name": "짜장면",
 "ingredient1": "국수면",
 "ingredient2": "춘장",
 "ingredient3": "완두콩"
 }
]
```

Body 부분 json 형식으로 저장  
{ "key": "value" }

## ▪ Django REST Framework 설정1

### ➤ serializers.py

```
serializers.py x
1 import ...
3
4
5 class FoodSerializer(serializers.HyperlinkedModelSerializer):
6 class Meta:
7 model = Food
8 fields = ('Food_name', 'ingredient1', 'ingredient2', 'ingredient3')
```

json 형식의  
key 부분 설정

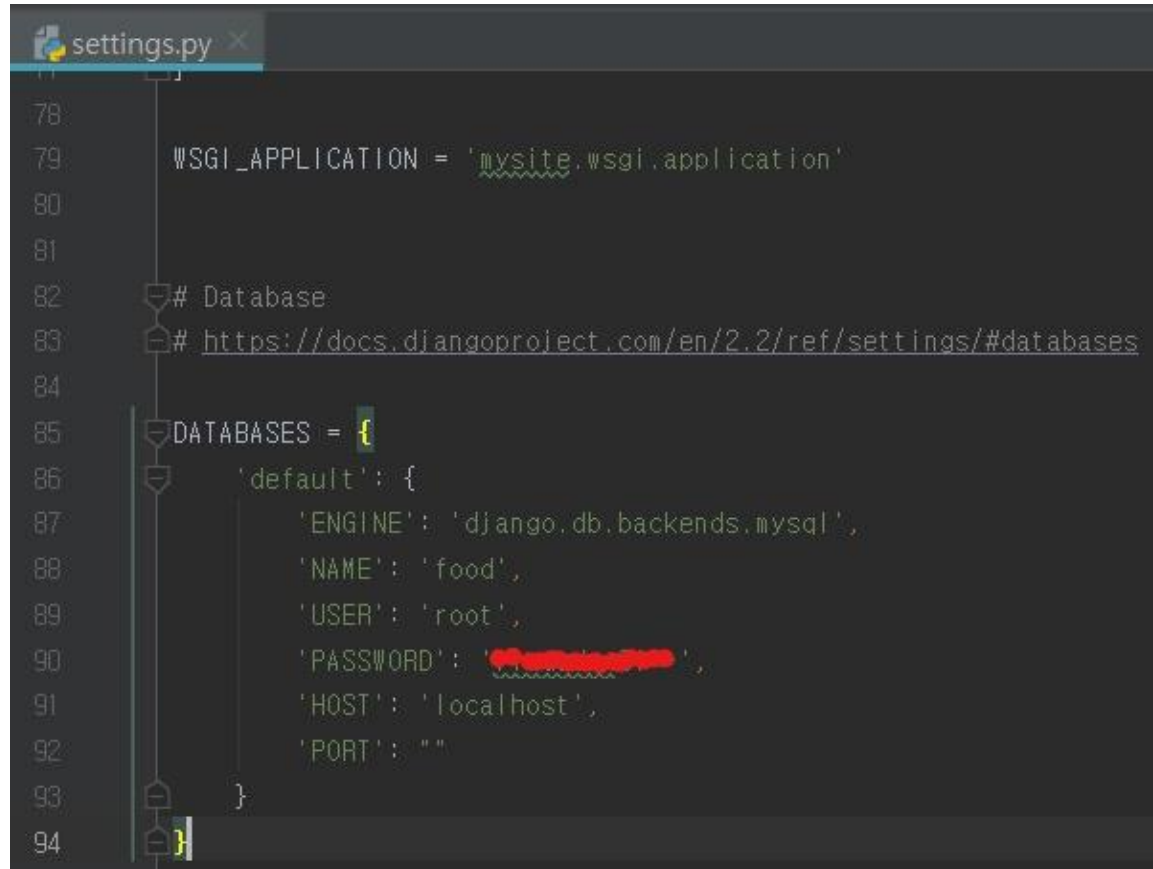
### ➤ models.py

```
models.py x
1 from django.db import models
2
3 # Create your models here.
4 class Food(models.Model):
5 Food_name = models.CharField(max_length=30)
6 ingredient1 = models.CharField(max_length=30)
7 ingredient2 = models.CharField(max_length=30)
8 ingredient3 = models.CharField(max_length=30)
9
```

Food Class 설정  
각 key에 대한 value type 지정

## ▪ Django REST Framework 설정2

### ➤ settings.py



```
78
79 WSGI_APPLICATION = 'mysite.wsgi.application'
80
81
82 # Database
83 # https://docs.djangoproject.com/en/2.2/ref/settings/#databases
84
85 DATABASES = {
86 'default': {
87 'ENGINE': 'django.db.backends.mysql',
88 'NAME': 'food',
89 'USER': 'root',
90 'PASSWORD': 'root@12345678',
91 'HOST': 'localhost',
92 'PORT': ''
93 }
94 }
```

MySQL과의 연동을 위한 DATABASES 부분 설정

## ▪ Django REST Framework 설정3

### ➤ MySQL에 테이블 자동 생성

```
mysql> desc myapp_food;
```

| Field       | Type        | Null | Key | Default | Extra          |
|-------------|-------------|------|-----|---------|----------------|
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| Food_name   | varchar(30) | NO   |     | NULL    |                |
| ingredient1 | varchar(30) | NO   |     | NULL    |                |
| ingredient2 | varchar(30) | NO   |     | NULL    |                |
| ingredient3 | varchar(30) | NO   |     | NULL    |                |

5 rows in set (0.00 sec)

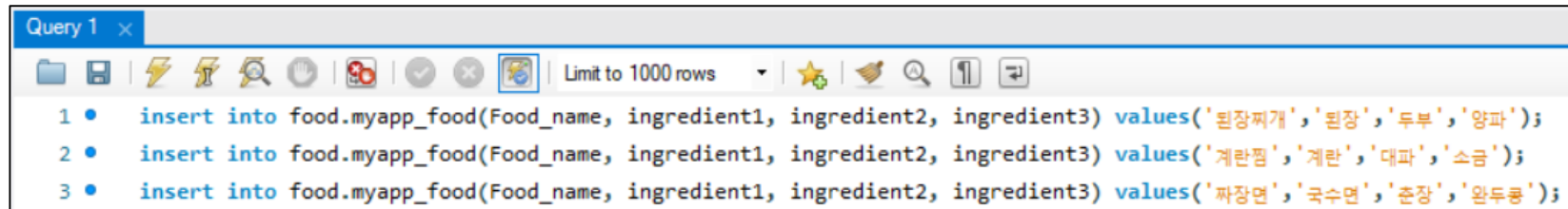
**python manage.py makemigrations**

**python manage.py migrate** 입력하면 MySQL과 연동

Django에서 설정한 Key값이 DB에서 Field 값으로 지정

## ▪ Django REST Framework 설정4

### ➤ DB 테이블에 모의로 데이터 삽입



```
Query 1 x
Limit to 1000 rows
1 • insert into food.myapp_food(Food_name, ingredient1, ingredient2, ingredient3) values('된장찌개', '된장', '두부', '양파');
2 • insert into food.myapp_food(Food_name, ingredient1, ingredient2, ingredient3) values('계란찜', '계란', '대파', '소금');
3 • insert into food.myapp_food(Food_name, ingredient1, ingredient2, ingredient3) values('짜장면', '국수면', '춘장', '완두콩');
```

### MySQL workbench에서 데이터 삽입



```
mysql> select * from myapp_food;
```

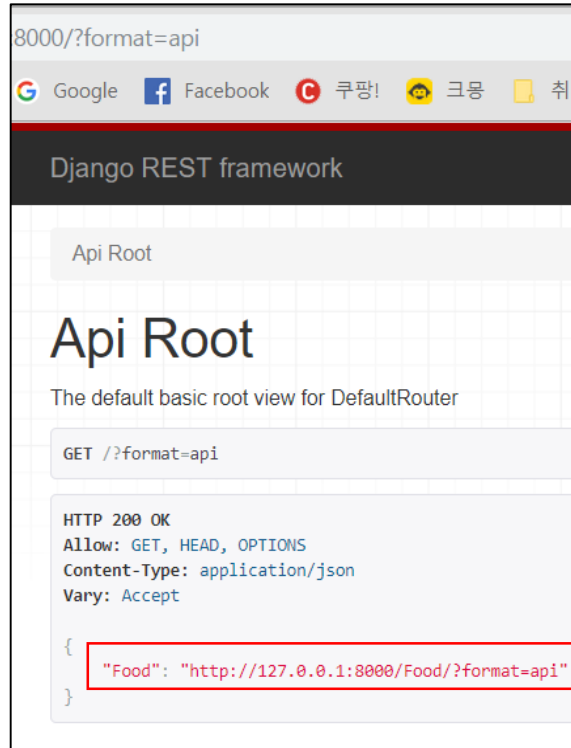
| id | Food_name | ingredient1 | ingredient2 | ingredient3 |
|----|-----------|-------------|-------------|-------------|
| 1  | 된장찌개      | 된장          | 두부          | 양파          |
| 2  | 계란찜       | 계란          | 대파          | 소금          |
| 3  | 짜장면       | 국수면         | 춘장          | 완두콩         |

```
3 rows in set (0.00 sec)
```

### MySQL Command Line에서 테이블 조회

## ▪ Django REST Framework 설정5

### ➤ RESTful API와 DB 연동 확인

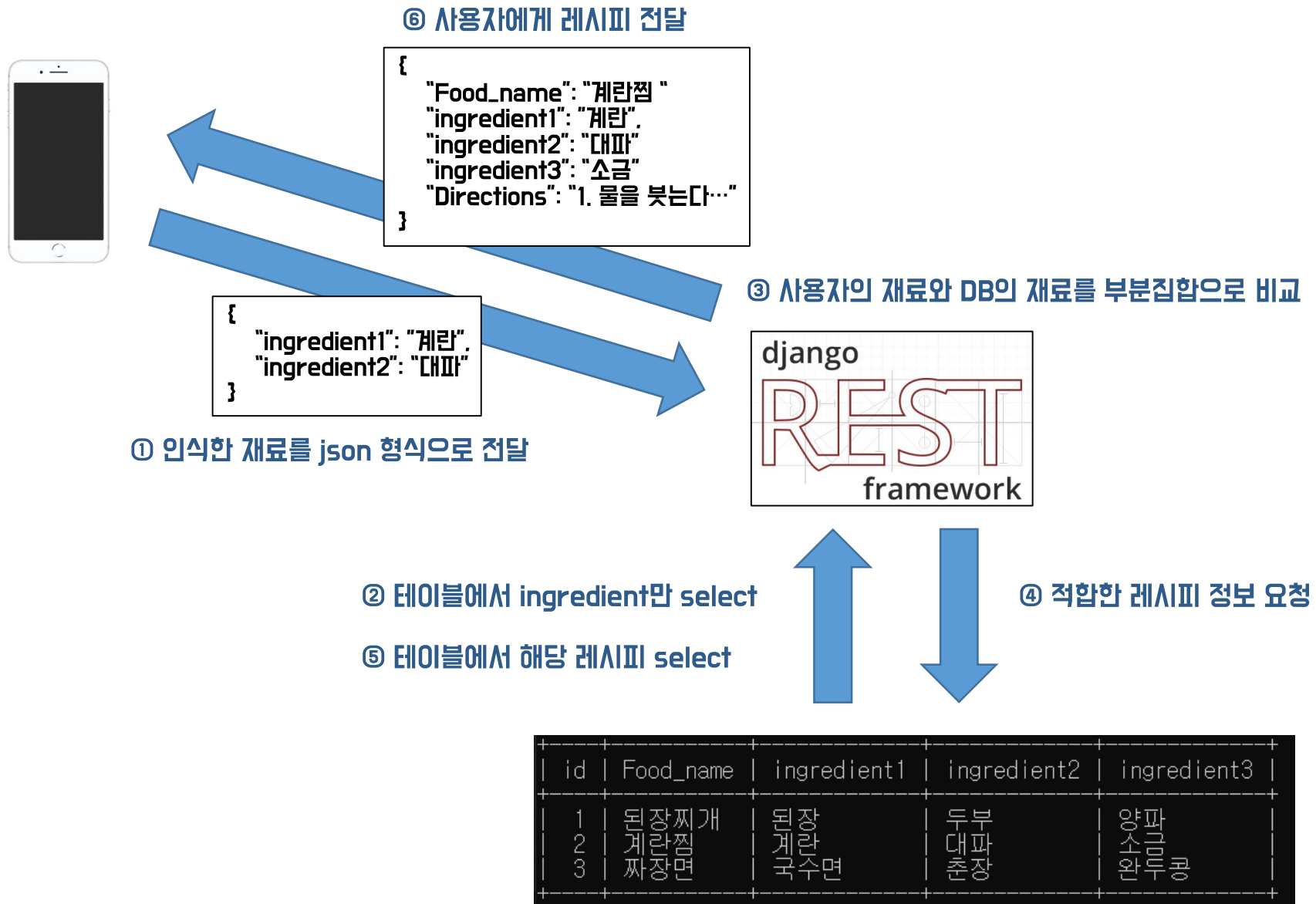


Food 모델로 진입



MySQL DB에  
저장된 내용 반영

## ▪ Django REST Framework 작동 과정



**AWS 구축 및**

**Django, MySQL Setting**

**201402275 이민기**



## ▪ AWS에 Python 및 Django 설치

### ➤ Xshell에 설치된 Django Project

```
ubuntu@ip-172-31-20-75:~/app/django/firstproject/myvenv$ ls
bin include lib lib64 manage.py project pyenv.cfg share
ubuntu@ip-172-31-20-75:~/app/django/firstproject/myvenv$ cd project/
ubuntu@ip-172-31-20-75:~/app/django/firstproject/myvenv/project$ ls
__init__.py settings.py urls.py wsgi.py
```

## ▪ AWS에 Python 및 Django 설치

### ➤ Settings.py에서 Host를 변경

```
"""
Django settings for project project.

Generated by 'django-admin startproject' using Django 1.11.20.

For more information on this file, see
https://docs.djangoproject.com/en/1.11/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.11/ref/settings/
"""

import os

Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

Quick-start development settings - unsuitable for production
See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '(nj1-f)=j304%@@3qx8n*3aonsw_*rr_%$u4w%i7u5v-8y=5sy'

SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ALLOWED_HOSTS = ['ec2-15-████████.ap-northeast-2.compute.amazonaws.com', '15-████████-3']

Application definition

INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
]

INSERT
```

## ▪ AWS에 Python 및 Django 설치

### ➤ 8000번 포트를 추가하여 웹서비스를 위한 환경 셋팅

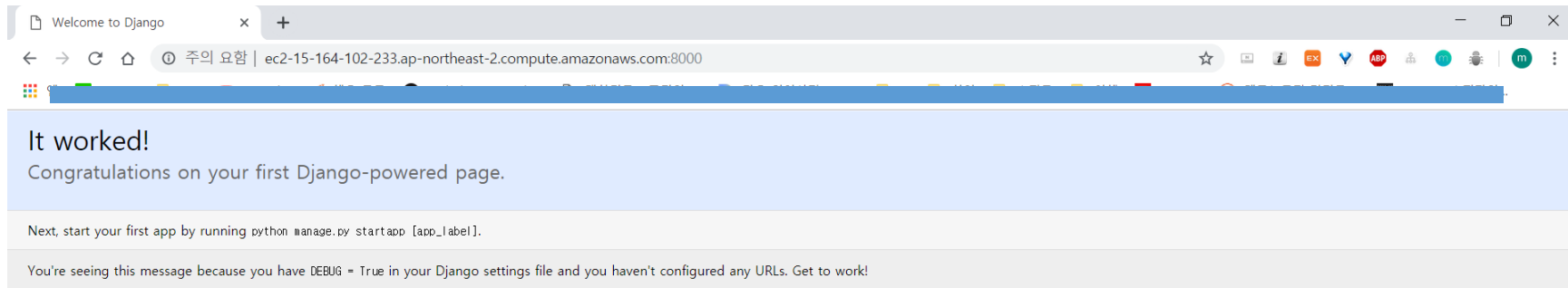
|                                                                                                                                 |     |      |        |           |             |   |
|---------------------------------------------------------------------------------------------------------------------------------|-----|------|--------|-----------|-------------|---|
| 사용자 지정 T                                                                                                                        | TCP | 8000 | 사용자 지정 | 0.0.0.0/0 | django port | ✕ |
| <div>규칙 추가</div> <p>참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보로 새 규칙이 생성됩니다. 이렇게 하면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 잠시 중단될 수 있습니다.</p> |     |      |        |           |             |   |

취소

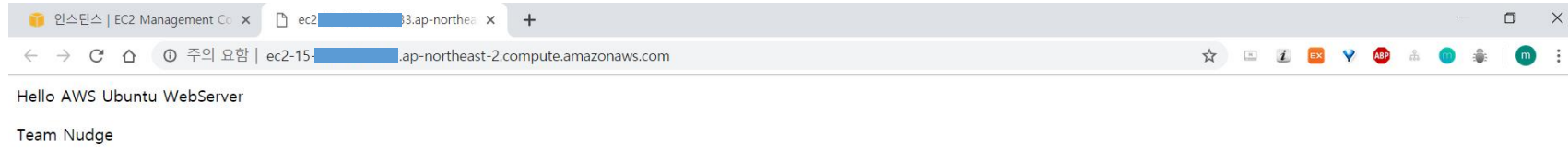
저장

## ▪ AWS에 Python 및 Django 설치

### ➤ 8000번 포트를 추가하여 웹서비스를 위한 환경 셋팅



## ■ AWS Ubuntu WebServer 구현



## ▪ AWS MySQL 설치 및 구현

```
ubuntu@ip-172-31-0-75:~$ sudo apt-get install mysql-server
```

```
root@ip-172-31-0-75:~# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.26-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
mysql>
```