2020년 2학기 데이터관리와 분석

프로젝트#1: Conceptual DB design & DB implementation

12조
2017-13918 황재훈
2017-10854 최의현
2017-18868 조형찬
2017-15751 장준혁
2017-10089 윤성진

목차

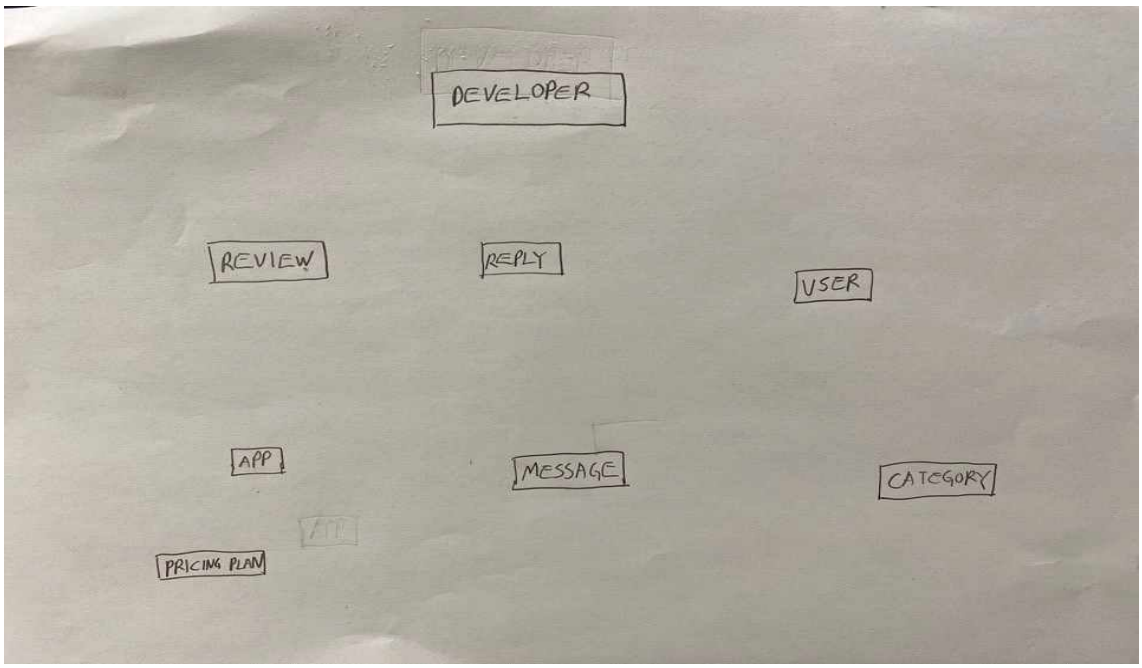
I. ER diagram 도식화
1) Entity Type 그리기
2) Weak Entity Type 그리기
3) Relationship Type 그리기
4) 최종 ER diagram

Ⅱ. Relational DB model 구현
1) Regular Entity Type mapping
2) Weak Entity Type mapping
3) Binary 1:1 Relationship Type mapping
4) Binary 1:N Relationship Type mapping
5) Binary M:N Relationship Type mapping
6) Multivalued Attributes mapping
7) N-ary Relationship Type mapping
8) 최종 Relation DB model

III. 데이터 입력
1) 프로그램 준비 작업
2) R2-1: Create Schema
3) R2-2: Create Table
4) R2-3: Insert Data
5) R2-4: Add Constraint Key (Foreign Key)

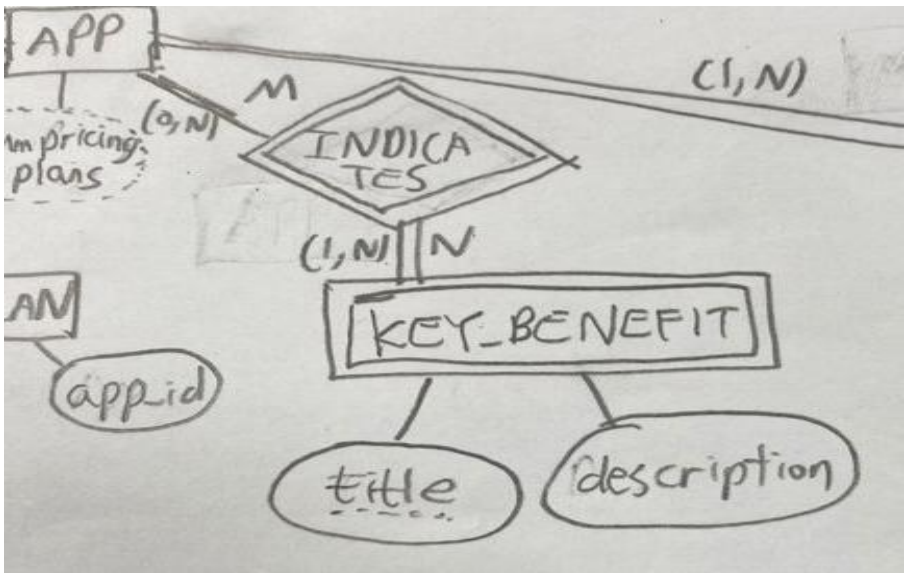
I. ER diagram 도식화

1) Entity Type 그리기



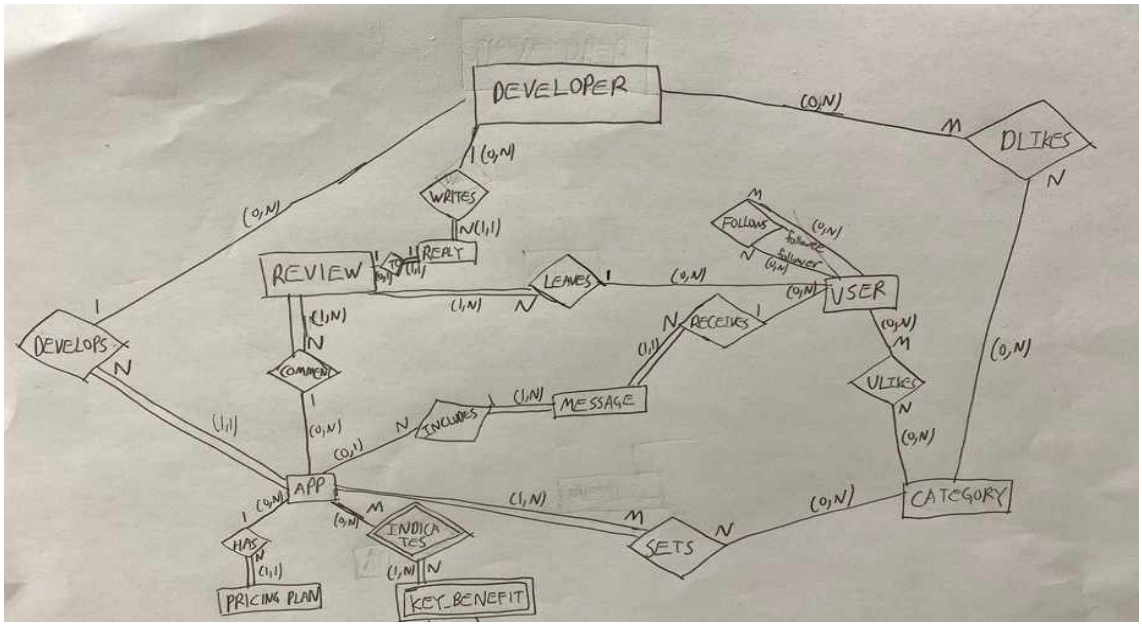
문제에서 명시된 조건으로 entity를 그리면 위와 같이 8개의 entity를 그릴 수 있다. (DEVELOPER, REVIEW, REPLY, USER, APP, MESSAGE, CATEGORY, PRICING_PLAN)

2) Weak Entity Type 그리기



weak entity type인 KEY_BENEFIT은 자신만의 key attribute가 존재하지 않는다. 한 앱의 주요 혜택들은 모두 다른 이름을 가지므로 KEY_BENEFIT에서 partial key로 title 선정하였고 KEY_BENEFIT과 APP을 Weak Relationship Type인 INDICATES로 연결해주었다.

3) Relationship Type 그리기

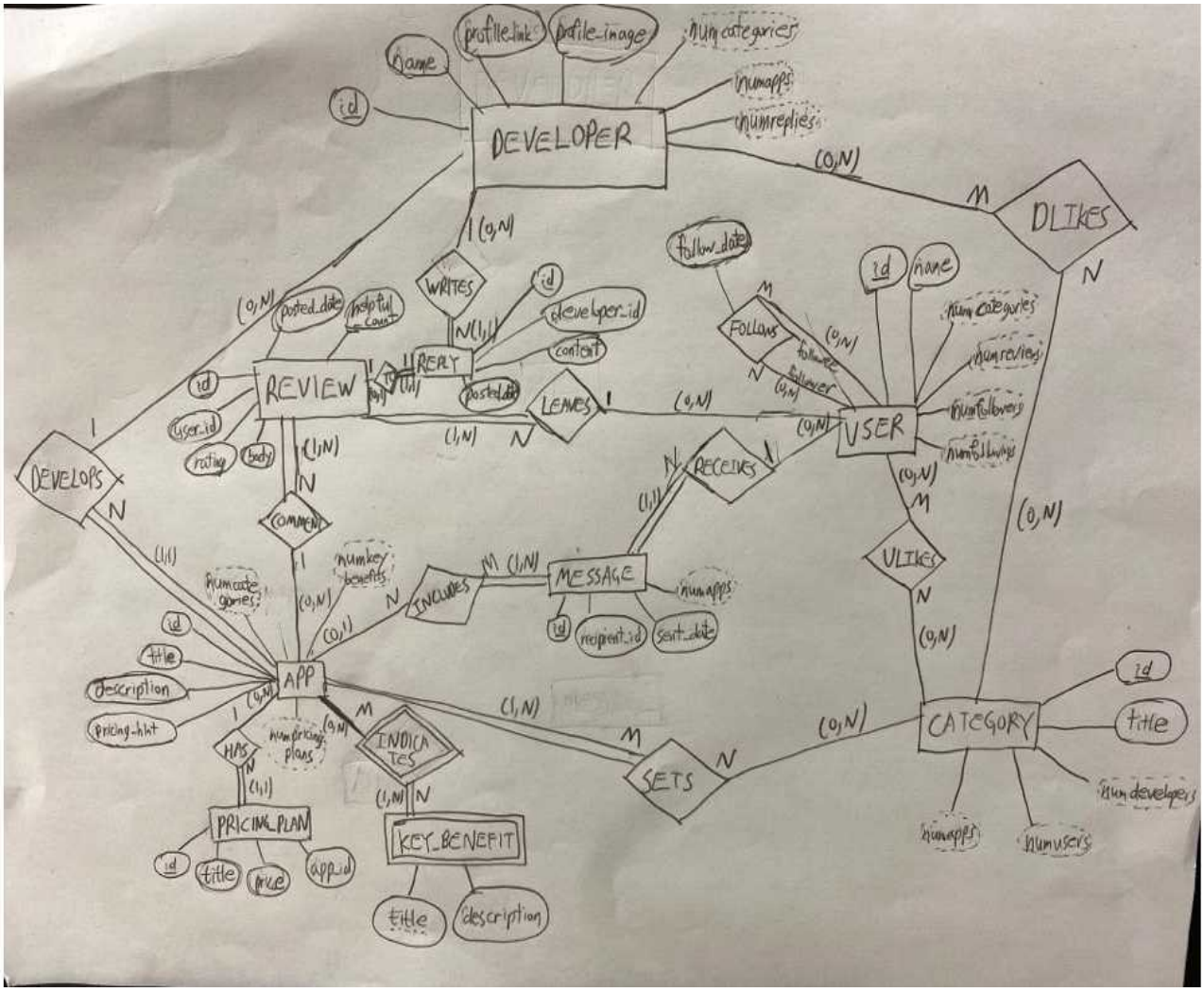


entity type 간의 여러 개의 relationship type이 존재한다. ER diagram에서 존재하는 Weak Relationship Type을 제외한 relationship type은 다음과 같다.

Relationship type	Entity type	Entity type
DEVELOPS	DEVELOPER	APP
HAS	APP	PRICING_PLAN
SETS	APP	CATEGORY
DLIKES	CATEGORY	DEVELOPER
ULIKES	USER	CATEGORY
WRITES	DEVELOPER	REPLY
TO	REVIEW	REPLY
COMMENT	REVIEW	APP

INCLUDES	APP	MESSAGE
RECEIVES	MESSAGE	USER
FOLLOWS	USER	USER
LEAVES	REVIEW	USER

4) 최종 ER diagram



최종 ER 다이아그램은 다음과 같다. relationship type을 그린 상태에서 attribute를 넣었는데 각 entity에 대해 넣은 attribute는 다음과 같다. 실선으로 된 동그라미는 derived attribute이다. relationship(FOLLOWS)에도 attribute가 있다.

Relationship	Attribute
FOLLOWS	follow_date

Entity	Attribute
DEVELOPER	id
	name
	profile_link
	profile_image
	numcategories (derived)
	numapps (derived)
	numreplies (derived)

Entity	Attribute
CATEGORY	id
	title
	numdevelopers(derived)
	numusers(derived)
	numapps(derived)

Entity	Attribute
USER	id
	name

	numcategories(derived)
	numreviews(derived)
	numfollowers(derived)
	numfollowings(derived)

Entity	Attribute
PRICING_PLAN	id
	title
	price
	app_id

Entity	Attribute
MESSAGE	id
	recipient_id
	sent_date
	numapps(derived)

Entity	Attribute
REVIEW	id
	user_id
	rating

	body
	posted_date
	helpful_count

Entity	Attribute
REPLY	id
	developer_id
	contend
	posted_date

Entity	Attribute
APP	id
	title
	description
	pricing_hint
	numcategories(derived)
	numkeybenefits(derived)
	numpricingplans(derived)

위와 같이 entity에 해당하는 relationship을 넣어서 전체 ER diagram을 완성했다.

Ⅱ. Relational DB model 구현

총 7단계를 거쳐 relational DB model을 구현했다.

1) Regular Entity Type mapping

ER diagram의 Regular Entity Type 8개를 아래의 그림과 같이 mapping 했다. 모든 Entity에서 'id'가 key attribte 이므로 각 relation에서 'id'를 primary key로 설정했다.

DEVELOPER

id	name	profile_image	profile_link

USER

id	name

APP

id	title	developer_id	description	pricing_hint

CATEGORY

id	title

REPLY

id	review_id	developer_id	content	posted_date

REVIEW

id	user_id	rating	body	helpful_count	posted_date	app_id

MESSAGE

id	recipient_id	sent_date

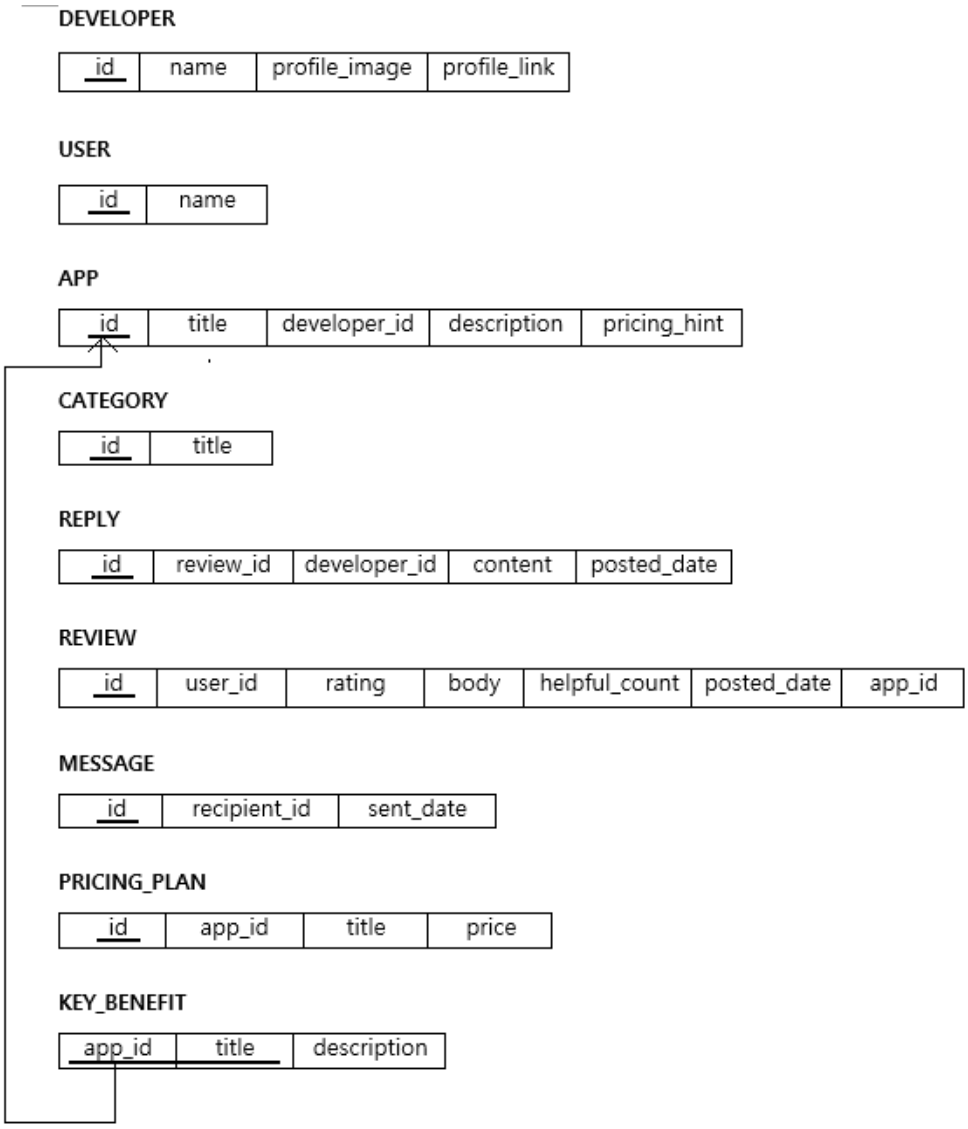
PRICING_PLAN

id	app_id	title	price

2) Weak Entity Type mapping

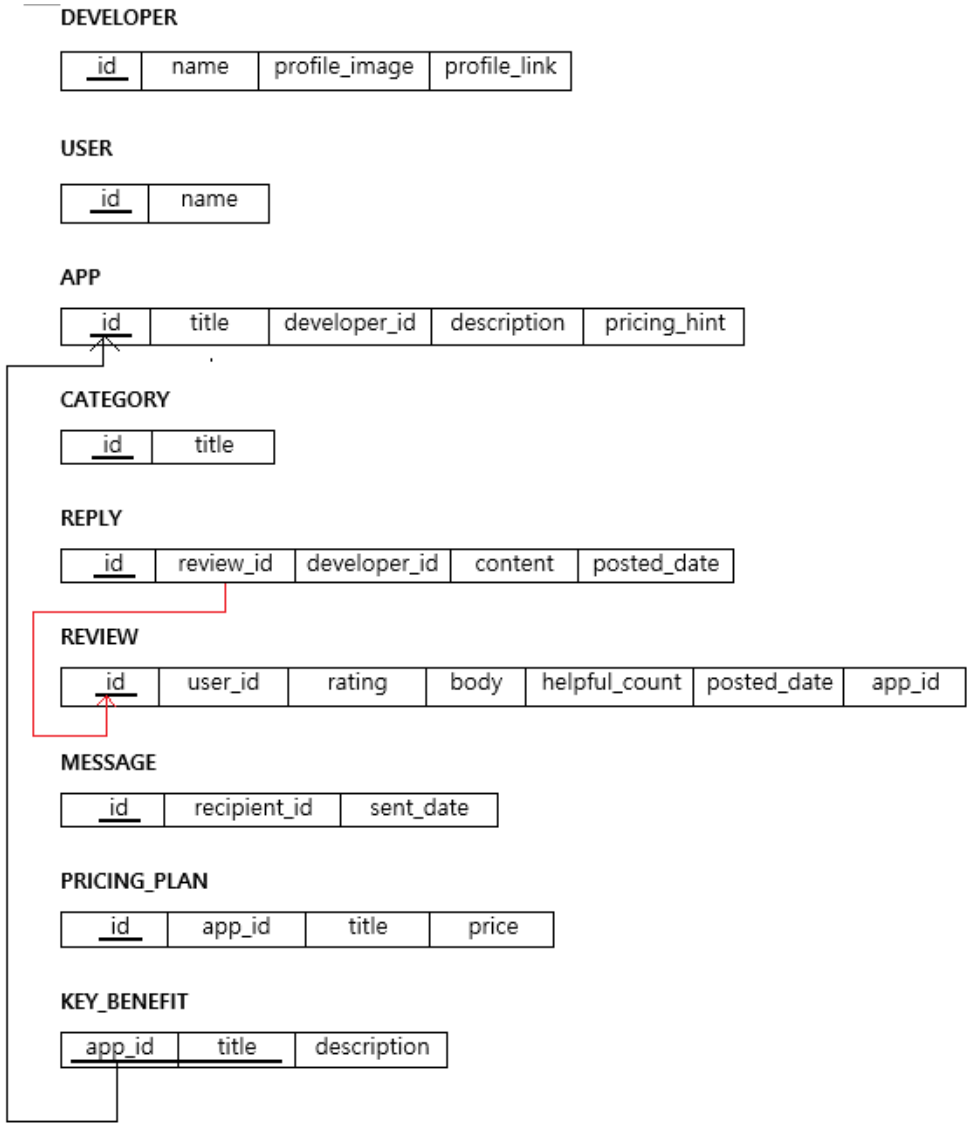
Weak Entity Type인 'KEY_BENEFIT'을 mapping 하는 과정을 나타냈다.

KEY_BENEFIT의 relation을 생성하고 'KEY_BENEFIT' Entity와 Identifying relationship 'INDICATES'으로 연결된 APP의 primary key를 refer한 foreign key 'app_id'와 partial key인 'title'의 combination이 KEY_BENEFIT을 unique하게 나타내므로 primary key로 설정하였다.



3) Binary 1:1 Relationship Type mapping

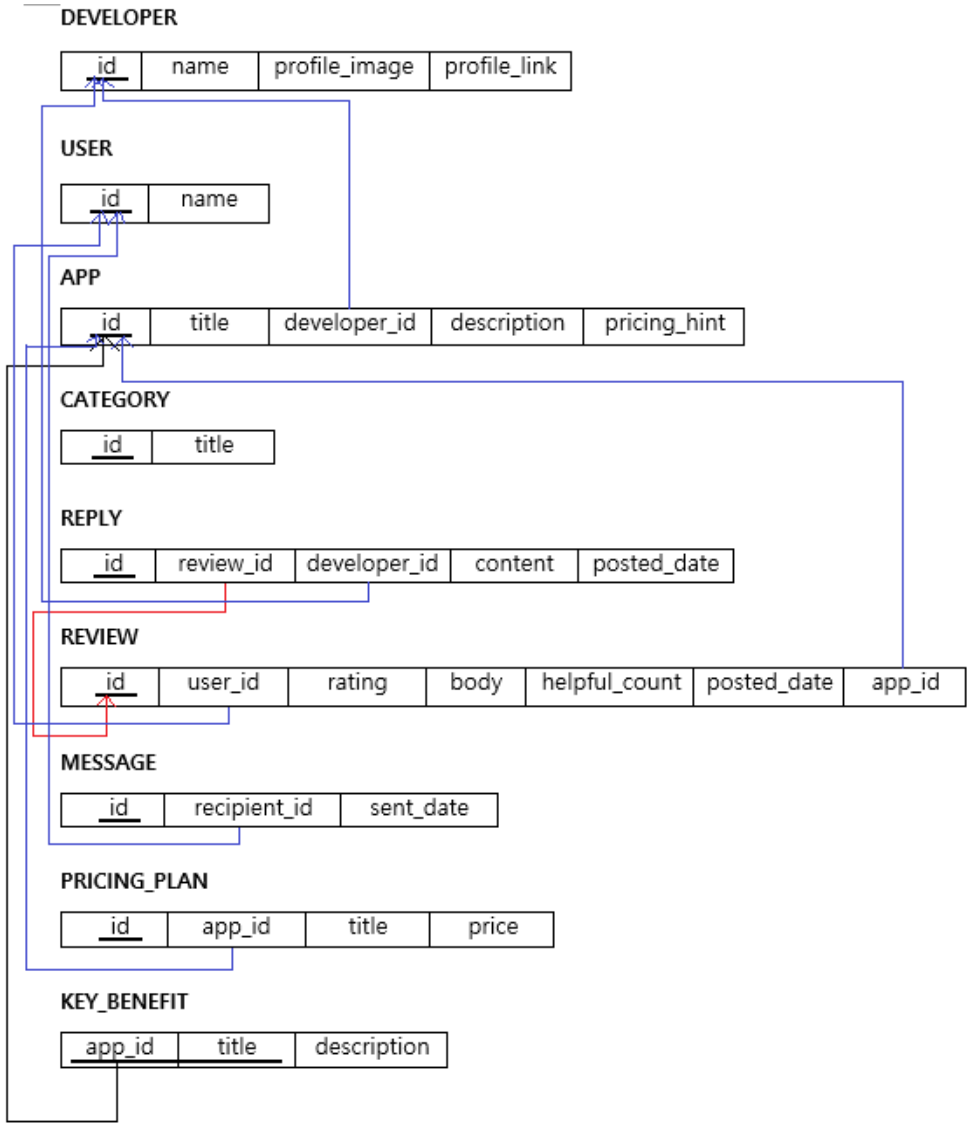
1:1 Relationship Type은 'TO' 1개 존재했고 'TO'의 attribute는 없었으므로 REPLY의 review_id가 REVIEW의 primary key id를 refer 하도록 아래의 그림과 같이 mapping 하였다. (빨간선으로 표현하였다.)



4) Binary 1:N Relationship Type mapping

1:N Relationship Type은 6개 존재하고 Relationship의 attribute는 존재하지 않았다. 1-side쪽 relation의 primary key를 N-side쪽 relation의 foreign key가 refer 하도록 아래의 그림과 같이 mapping 했다. (파란선으로 표현했다.)

relation	foreign key		relation	primary key
APP	developer_id	→ (refer)	DEVELOPER	id
REPLY	developer_id		DEVELOPER	id
REVIEW	app_id		APP	id
PRICING_PLAN	app_id		APP	id
REVIEW	user_id		USER	id
MESSAGE	recipient_id		USER	id

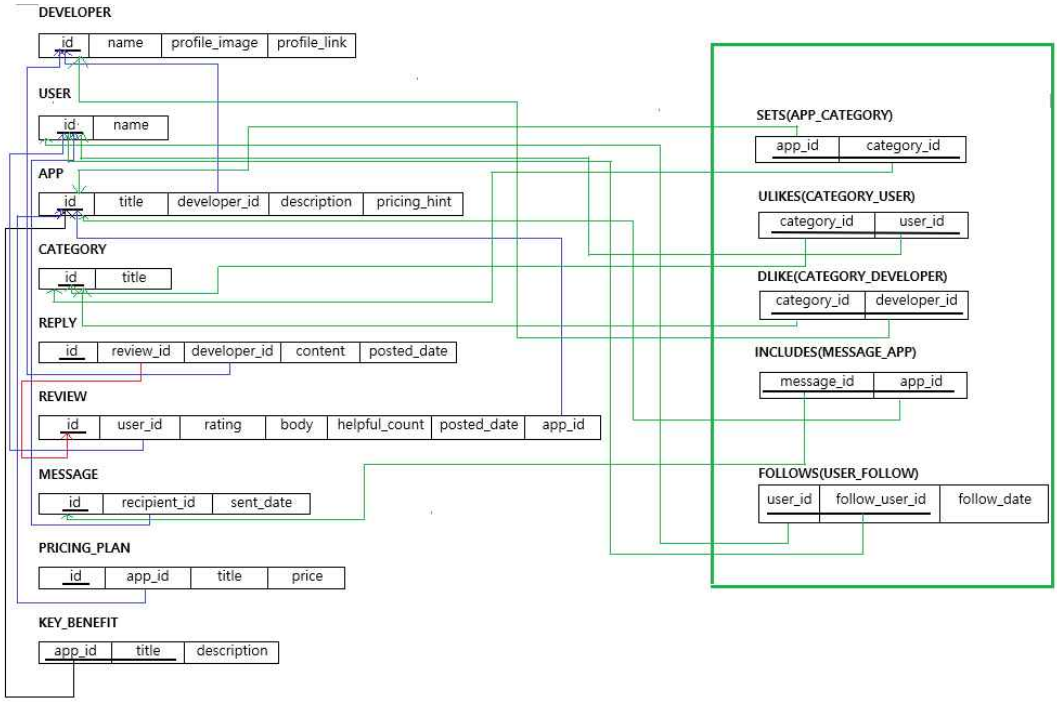


5) Binary M:N Relationship Type mapping

M:N Relationship Type은 5개 존재하고 각각의 해당 Relationship에 대해 새로운 relation을 생성하였다. 이때 relation의 이름은 데이터 엑셀파일 제목과 똑같이 설정하였다.

Relationship에 연결된 각 side의 primary key를 2개의 foreign key가 각각 refer 하도록 설정하고 두 개의 foreign key combination을 primary로 설정하였다. 이때 'FOLLOWS' Relationship은 재귀이므로 두 개의 foreign key가 같은 primary key를 refer 한다. Relationship에 attribute가 존재하는 경우는 'FOLLOWS' Relationship의 'follow_date' attribute 뿐이었고 이는 새로 생성한 relation에 포함시키도록 아래의 그림과 같이 mapping 했다. (초록색 박스에 포함된 relation과 초

록색 화살표가 **5)** Binary M:N Relationship Type mapping 과정이다.)



6) Multivalued Attributes mapping

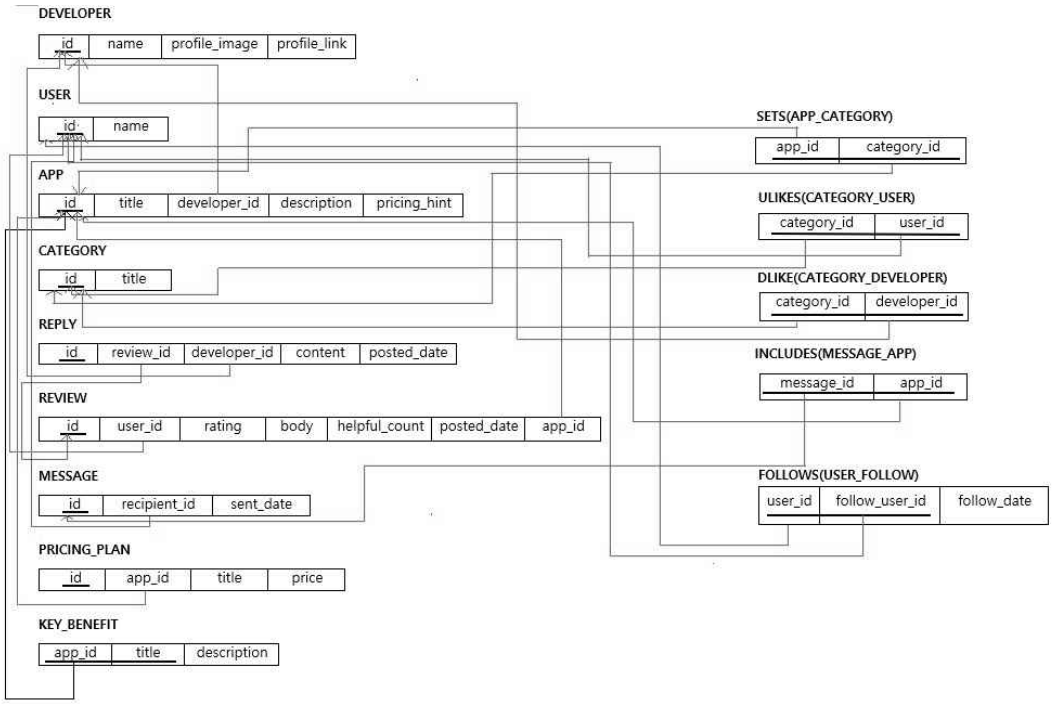
본 프로젝트에서 Multivalued Attributes는 존재하지 않았다.

7) N-ary Relationship Type mapping

본 프로젝트에서 N-ary Relationship는 존재하지 않았다.

8) 최종 Relation DB model

최종 Relation DB model은 아래의 그림과 같다.



III. 데이터 입력

1) 프로그램 준비 작업

Python과 MySQL을 사용하여 프로그램을 구현하기 위해
Python에서 mysql-connector-python 라이브러리를 가져와 mySQL과 연결하고,
팀 번호를 변수로 저장하는 과정이다.

```
import mysql.connector

# TODO: REPLACE THE VALUE OF VARIABLE team (EX. TEAM 1 --> team = 1)
team = 12
```

2) R2-1: Create Schema

MySQL 상에 'DMA_team12'의 이름을 가지는 schema(database)를 생성하는 과정이다. 이때, 해당 schema(database)가 존재할 경우 생성 과정을 다시 수행하지 않아야 하므로'~ IF NOT EXISTS ~'를 통해 존재하지 않는다면 해당 schema(database)를 생성하도록 한다.
- 각 요구사항에 대해서 함수를 호출하는 방식으로 코드를 작성하였기 때문에
각 함수(requirement 1,2,3,4)마다 cursor.execute('USE DMA_team12;')를 입력해 주었다.
-MySQL에서는 schema가 database와 사실상 같은 의미로 쓰이기 때문에 용어 간의 혼선을 빚지 않도록 병기하였다.

```
# Requirement1: create schema ( name: DMA_team## )
def requirement1(host, user, password):
    cnx = mysql.connector.connect(host=host, user=user, password=password)
    cursor = cnx.cursor()
    cursor.execute('SET GLOBAL innodb_buffer_pool_size=2*1024*1024*1024;')

    # TODO: WRITE CODE HERE
    cursor.execute('CREATE DATABASE IF NOT EXISTS DMA_team12;')
    cursor.execute('USE DMA_team12;')

    # TODO: WRITE CODE HERE
    cursor.close()
```

3) R2-2: Create Table

데이터를 저장하기 위한 table을 생성하는 과정으로, 다음과 같이 생성하였다.
- 앞서 도출한 최종 Relational DB에 따라 총 14개의 table을 생성하였다.
- 생성된 table과 column 이름 및 순서가 주어진 dataset과 일치하도록 생성하였다.
- table이 이미 존재하는 경우 생성 과정을 다시 수행하지 않아야 하므로
'CREATE TABLE IF NOT EXISTS ~' 형식의 MySQL 구문을 사용하였다.
- 'app'의 'pricing_hint'와 'pricing_plan'의 'title'은 null value를 허용하므로 제

외하고, 이외의 모든 attribute에 NOT NULL constraint를 적용하였다.
- 각 table의 PRIMARY KEY는 최종 Relational DB와 같게 설정하였다.
- 'category'의 'title'과 'reply'의 'review_id'에 UNIQUE constraint를 적용하였다.
- table을 실행하는 과정에서 데이터에 특수문자 혹은 다수의 따옴표가 포함되는 경우, 'incorrect string value' 형식의 Error가 발생하는 것을 확인하였다. 이는 MySQL에서 utf8가 3bytes까지만 지원하는 반면, 해당 형식의 데이터는 4bytes이므로 Error가 발생하는 것으로 파악하였다. 이에 Character Set을 utf8mb4로 설정하여 문제를 해결하였다. 따라서 각 table을 생성할 때 'Engine = InnoDB DEFAULT CHARSET = utf8mb4'과 같은 설정 구문을 통해 4bytes까지 허용하도록 하였다.

아래는 'app' table을 생성한 예시 화면이다. 나머지 table도 위와 같은 동일한 방식으로 생성하였으므로, 구체적인 생성 내용은 여기에서는 생략하였다.

```
# Requierement2: create table
def requirement2(host, user, password):
    cnx = mysql.connector.connect(host=host, user=user, password=password)
    cursor = cnx.cursor()
    cursor.execute('SET GLOBAL innodb_buffer_pool_size=2*1024*1024*1024;')

    # TODO: WRITE CODE HERE
    cursor.execute('USE DMA_team12;')
    cursor.execute('''
            CREATE TABLE IF NOT EXISTS app(
            id VARCHAR(255) NOT NULL,
            title VARCHAR(255) NOT NULL,
            developer_id INT(11) NOT NULL,
            description INT(11) NOT NULL,
            pricing_hint VARCHAR(255),
            PRIMARY KEY(id)
            )Engine = InnoDB DEFAULT CHARSET = utf8mb4;
            ''')
```

4) R2-3: Insert Data

위에서 생성된 table에 주어진 dataset의 데이터를 저장하는 과정이다. 다음과 같은 과정을 각 table에 대해 거침으로써 dataset에 주어진 모든 데이터를 대응하는 table에 각각 저장하였다.

- ‘f_(table name)’ 형식의 파일을 만들어 csv 형식의 dataset을 읽게 하였다.
- readlines()를 통해 읽은 데이터를 ‘data_(table name)’의 변수에 저장하였다.
- line별로 읽어진 csv 파일에서 ‘\n’을 제거해주고 ‘,’ 단위로 split하여 ‘line_(table name)’의 변수에 list 형태로 저장하였다.
- column 이름을 제외한 모든 row에 대해 DB에 데이터를 INSERT하였다.
- null value를 허용하는 column에 대해서는 if-else 구문을 활용하여 null value인 경우에 ‘NULL’, null value가 아닌 경우에는 해당 값을 데이터로 저장하도록 하였다.

아래는 ‘app’ table에 데이터를 저장한 예시 화면이다. 아래에서 “line_app[4]”에 해당하는 부분, 즉 ‘app’ table에서 ‘pricing_hint’에 해당하는 5번째 column의 데이터는 null value가 가능하므로 if-else 구문을 사용하여 데이터를 저장한 것을 확인할 수 있다. 나머지 table도 위와 같은 동일한 방식을 거쳐 데이터를 저장하였으므로, 구체적인 저장 내용은 여기에서는 생략하였다.

```
# Requirement3: insert data
def requirement3(host, user, password, directory):
    cnx = mysql.connector.connect(host=host, user=user, password=password)
    cursor = cnx.cursor()
    cursor.execute('SET GLOBAL innodb_buffer_pool_size=2*1024*1024*1024;')

    # TODO: WRITE CODE HERE
    cursor.execute('USE DMA_team12;')

    f_app = open(directory + '₩₩app.csv', 'r', encoding='utf-8')
    data_app = f_app.readlines()

    for i in range(1, len(data_app)):

        line_app = data_app[i].replace('₩n', '')
        line_app = line_app.split(',')

        if line_app[4]=='':
            sql='INSERT INTO app VALUES (%s,%s,%s,%s,NULL)'
            cursor.execute(sql, (line_app[0], line_app[1], line_app[2], line_app[3]))
        else:
            sql = 'INSERT INTO app VALUES (%s,%s,%s,%s,%s)'
            cursor.execute(sql,(line_app[0], line_app[1], line_app[2], line_app[3], line_app[4]))
    cnx.commit()
    f_app.close()
```

5) R2-4: Add Constraint Key (Foreign Key)

FOREIGN KEY constraint들을 반영하는 과정이다.

Relational DB Model을 mapping하는 과정에서 Weak Entity type, Binary 1:1 Relationship type, Binary 1:N Relationship Type, Binary M:N Relationship Type 등을 mapping할 때 발생한 Foreign Key Constraint들을 빠짐없이 DB에 반영해주었다.

```
# Requirement4: add constraint (foreign key)
def requirement4(host, user, password):
    cnx = mysql.connector.connect(host=host, user=user, password=password)
    cursor = cnx.cursor()
    cursor.execute('SET GLOBAL innodb_buffer_pool_size=2*1024*1024*1024;')

    # TODO: WRITE CODE HERE
    cursor.execute('USE DMA_team12')
    cursor.execute('ALTER TABLE app ADD CONSTRAINT FOREIGN KEY (developer_id) REFERENCES developer(id);')
    cursor.execute('ALTER TABLE reply ADD CONSTRAINT FOREIGN KEY (review_id) REFERENCES review(id);')
    cursor.execute('ALTER TABLE reply ADD CONSTRAINT FOREIGN KEY (developer_id) REFERENCES developer(id);')
    cursor.execute('ALTER TABLE review ADD CONSTRAINT FOREIGN KEY (app_id) REFERENCES app(id);')
    cursor.execute('ALTER TABLE message ADD CONSTRAINT FOREIGN KEY (recipient_id) REFERENCES user(id);')
    cursor.execute('ALTER TABLE pricing_plan ADD CONSTRAINT FOREIGN KEY (app_id) REFERENCES app(id);')
    cursor.execute('ALTER TABLE key_benefit ADD CONSTRAINT FOREIGN KEY (app_id) REFERENCES app(id);')
    cursor.execute('ALTER TABLE app_category ADD CONSTRAINT FOREIGN KEY (app_id) REFERENCES app(id);')
    cursor.execute('ALTER TABLE app_category ADD CONSTRAINT FOREIGN KEY (category_id) REFERENCES category(id);')
    cursor.execute('ALTER TABLE category_user ADD CONSTRAINT FOREIGN KEY (category_id) REFERENCES category(id);')
    cursor.execute('ALTER TABLE category_user ADD CONSTRAINT FOREIGN KEY (user_id) REFERENCES user(id);')
    cursor.execute('ALTER TABLE category_developer ADD CONSTRAINT FOREIGN KEY (category_id) REFERENCES category(id);')
    cursor.execute('ALTER TABLE category_developer ADD CONSTRAINT FOREIGN KEY (developer_id) REFERENCES developer(id);')
    cursor.execute('ALTER TABLE message_app ADD CONSTRAINT FOREIGN KEY (message_id) REFERENCES message(id);')
    cursor.execute('ALTER TABLE message_app ADD CONSTRAINT FOREIGN KEY (app_id) REFERENCES app(id);')
    cursor.execute('ALTER TABLE user_follow ADD CONSTRAINT FOREIGN KEY (user_id) REFERENCES user(id);')
    cursor.execute('ALTER TABLE user_follow ADD CONSTRAINT FOREIGN KEY (follow_user_id) REFERENCES user(id);')
    # TODO: WRITE CODE HERE
    cnx.commit()
    cursor.close()
```