Data Management and Analysis Project1

Team09 오태양, 김해찬, 하창우, 김민영

INTRODUCTION

온라인 소프트웨어 유통망 사이트 A는 개발자와 사용자의 교류 서비스를 제공하는 사이트로 소프트웨어를 출시, 구매할 수 있는 사이트이다. 관련 서비스를 제공하기 위해서는 소프트웨어, 사용자 등과 관련된 정보를 저장해야 하며 저장을 효율적으로 하기 위해 새로운 데이터베이스의 구축이 필요하다. 본 프로젝트에서는 A에서 원하는 조건(requirements)을 만족하는 Database(DB) 구축을 목표로 하였으며 사용자(User), 개발자(developer), 소프트웨어(item) 등의 용어로 바꾸어 사용하였다.

DB 구축을 위한 과정은 크게 2단계로 나누었다. 먼저 requirement를 만족하는 ER diagram을 도식화하여 구조를 명료히 한 후(PART I), 이를 relational model로 변환하여 constraints들을 설정한 후 Python 언어와 MySQL을 사용하여 DB를 구현화하는 것을 목표로 했다(PART II).

PART I: ER DIAGRAM 도식화

ER diagram 을 도식화할 때 중요한 것은 requirement를 충족하는 것과 data redundancy를 최소화 하는 것이다. 각 entity 의 저장할 attribute를 최소화 하기 위해 유도할 수 있는 attribute는 전부 derived attribute 으로 만듦으로써 requirement 를 충족시키며 redundancy 를 줄이려 하였다. 이를 위해서 다음과 같이 단계를 구분했다. 이를 바탕으로 그린 ER diagram 은 Fig.1와 같다.

단계 1: Defining entities and relationships
단계 2: Checking ratio of relationships
단계 3: Classifying attributes

단계 1: Defining entities and relationships

먼저 가장 필수적인 USER, ITEM, GENRE, BUNDLE, TAG라는 5가지 entity를 정의했다. USER와 ITEM사이에는 use와 review라는 2가지 관계가 존재하는데, review의 경우 사용자가 아이템을 리뷰함에 따라 타 사용자가 review를 추천할 수 있으므로 review 를 relationship 이 아닌 새로운 entity인 REVIEW로 정의하 였으며, USER와 REVIEW사이의 relationship인 Reviewing, ITEM과 REVIEW사이의 relationship인 Reviewed를 정의했다. 또한 requirement 에 따르면, 사용자는 타 사용자의 review에 대해 추천하는 행위를 할 수 있으므로, USER와 REVIEW사이에 RecReview relationship을 추가로 정의했다.

ITEM과 GENRE, ITEM과 BUNDLE, ITEM과 TAG사이에도 각각 Genre_of, Bundle_of, Tag_of relationship을 정의했다. 이 때, requirement 1-5 7에 의해 세 entity 모두 한 개 이상의 item을 포함해야하기 때문에 total participation of entity in relationship으로 정의했다. 추가적으로 TAG는 primary key가 존재하지 않기 때문에 weak entity type으로, Tag_of는 identifying relationship으로 정의했다.

마지막으로 번들에 속하는 아이템들이 어느 장

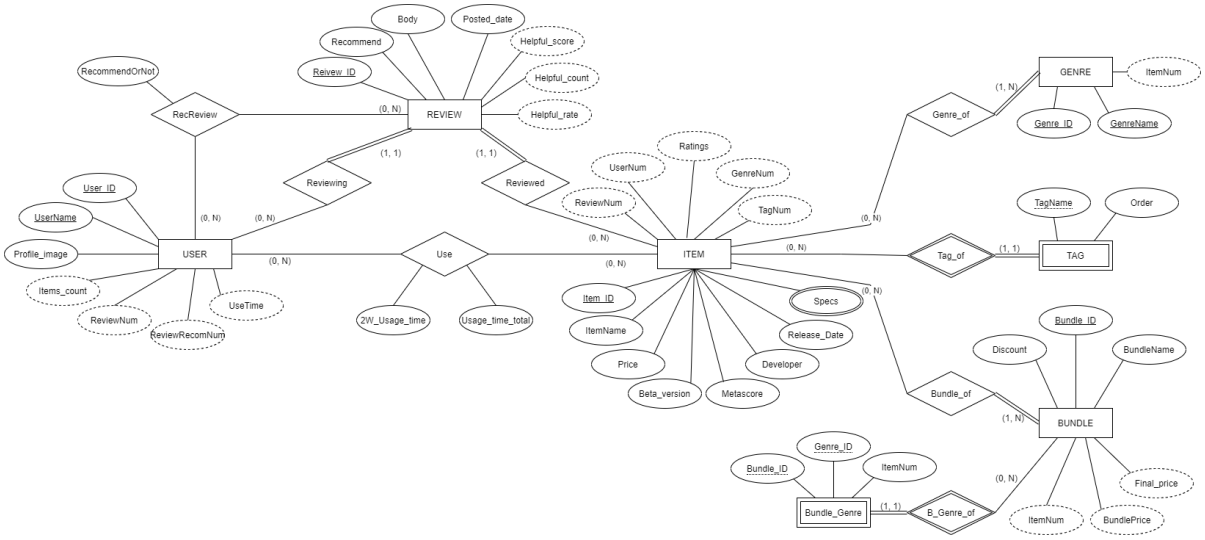

FIG. 1: ER diagram

르에 몇개나 속하는지에 대한 data를 저장해아하는데 만약 이를 attribute로 취급하면 이는 derived attribute이자 multivalued이다. 위와같은 문제를 방지하고 명료한 시각화를 위해 Bundle_Genre라는 weak entity를 정의하고, Bundle과의 identifying relationship을 B_genre_of으로 정의했다.

단계 2: Checking ratio of relationships

Use relationship에서 USER는 최소 0개, 최대 n개의 아이템을 사용할 수 있으며, ITEM도 최소 0명, 최대 n명의 사용자에게 사용될 수 있으므로 모두 (0,n)으로 표현했다. Reviewing relationship에서는 USER는 최소 0개, 최대 n개의 리뷰를 남길 수 있고, 모든 REVIEW는 하나의 USER에게서 남겨지므로 각각 (0,n), (1,1)로 표현했다. Reviewed relationship에서는 ITEM은 최소 0번, 최대 n번의 리뷰가 남겨지고 있고, REVIEW는 한 ITEM에 대해서만 남겨지므로 각각 (0,n), (1,1)로 표현했다. RecReview relationship에서는 타 USER는 REVIEW에 대해 최소 0번, 최대 n번의 추천여부를 남길수 있고, REVIEW는 최소 0명, 최대 n명의 타 USER에게 추천여부를 받을 수 있으므로 모두 (0,n), (0,n)로 표현했다. Genre_of relationship에선 ITEM은 최소 0개, 최대 n개의 장르에 포함되며, 장르는 최소 1개, 최대 n개의 item을 포함할 수 있으므로 각각 (0,n), (1,n)으로 표현했다. Bundle_of relationship와 Tag_of relationship도 마찬가지로 각각 (0,n), (1,n)으로 표현했다. B_genre_of relationship은 번들에 속하는 장르가 0 N으로 다양할 수 있기 때문에 (0,N)과 (1,1)로 표현했다.

단계 3: Classifying attributes

USER entity에는 사용자 고유번호를 key attribute로, 닉네임과 프로필 이미지 존재여부는 stored attribute로 정의했다. 그 외에는, 사용자의 아이템별로 총 사용시간과 2주간 사용시간을 Use relationship에 할당함으로써 USER entity의 사용자가 작성한 리뷰의 수는 와 작성한 리뷰들이 받은 총 추천수, 가입이래로 사용자가 이용한 아이템들의 이용시간의 총합을 derived attribute로 설정했다. 사용자 고유번호와 닉네임, 프로필

이미지 존재 여부, 사용자의 이용 아이템 개수는 derive가 불가능하기 때문에 USER entity의 stored attribute로 설정했다.

타 사용자의 리뷰에 대한 추천 여부는 RecReview relationship의 stored attribute로 정의하고 이로부터 리뷰가 받은 추천 수, 리뷰가 받은 평가 중 추천의 비율, 리뷰가 받은 총 평가 수를 derive하여 REVIEW entity의 derived attriute로 설정했다.

ITEM의 사용자 수는 Use relationship으로부터, REVIEW 수는 Reviewed relationship으로부터, 사용자들에게 받은 평가 점수는 REVIEW로부터, 속한 장르의 개수는 Genre_of relationship로부터, item의 태그 숫자는 Tag_of relationship으로부터 derive되기 때문에 derived attribute로 설정했다. 사이트에서 부여한 고유번호, 아이템의 이름, 가격 정보, 정식 출시 전 베타 버전의 유무, 외부에서 받은 평가 점수, 개발사 정보, 출시 일자, 스펙은 stored attribute로 정의했다.

장르에 속한 item 개수는 genre_of relationship, 번들과 관련된 장르에 속한 item 개수는 Bundle_Genre에는 Bundle_ID와 Genre_ID를 composite key를 할당 한 후 item_number attribute로 할당했다. 번들의 가격정보와 할인된 가격은 번들의 할인율 attribute와 item의 price attribute, bundle_of relationship으로부터 derive되므로 derived attribute로 설정했다. 장르와 번들별로 유일한 장르의 고유번호와 이름, 번들의 고유번호를 key attribute로 설정했다. Derive되지 않는 번들의 이름은 stored attribute로 설정했다.

태그는 weak entity type이기 때문에 key attribute가 존재하지 않는다. 하지만 한 아이템 내의 태그 이름은 유일하기 때문에, 태그의 이름을 partial key attribute로 설정했다.

Summary of ER diagram

본 프로젝트에서는 총 USER, REVIEW, ITEM, GENRE, BUNDLE, TAG, Bundle_Genre라는 7가지 entity를 정의했고 이를 바탕으로 entity 사이간에 Use, Reviewing, Reviewed, RecReview, Genre_of, Bundle_of, Tag_of, B_Genre_of 라는 8가지 relationship을 정의했다. 각각의 entity와 relationship의 의미는 다음 표와 같다.

Entity	의미	Relationship	의미
USER	사용자	Use	사용자가 아이템을 사용
REVIEW	리뷰	Reviewing	사용자가 리뷰를 남김
ITEM	아이템	Reviewed	아이템이 리뷰되어짐
GENRE	장르	RecReview	사용자가 리뷰를 (비)추천
BUNDLE	번들	Genre_of	아이템의 장르
TAG	태그	Bundle_of	아이템의 번들
Bundle_Genre	번들-장르	Tag_of	아이템의 태그
		B_Genre_of	번들과 장르사이 관계

FIG. 2: Entities and relationships used in ER diagram

PART II: DB 구현 및 데이터 입력

본 프로젝트에서는 DB의 구현화에 앞서 ER diagram을 바탕으로 relational model을 구현함으로써 constraint 조건과 domain을 확인하고자 했다. 이를 위해 강의자료의 ER-to-relational mapping step을 따라가고자 했다.

Relation name	Attribute name	Variable domain	key 여부	referential	특이사항
User	User_ID	VARCHAR(255)	primary key		
	UserName	VARCHAR(255)	unique- candidate key		
	Profile_image	TINYINT			
	Items_count	INT(11)			
Item	Item_ID	VARCHAR(255)	primary key		
	ItemName	VARCHAR(255)			
	Price	VARCHAR(255)			
	Beta_version	TINYINT			
	Ratings	INT(11)			
	Metascore	INT(11)			외부 평가
	Developer	VARCHAR(255)			null is possible
	Release_Date	DATE			null is possible
Item_specs	Item_ID	VARCHAR(255)	primary composite key	referential to ITEM(item_id)	
	Item_spec	VARCHAR(255)	primary composite key		null, multivalue is possible
Use(User_item)	User_ID	VARCHAR(255)	primary composite key	referential to USER(User_ID)	
	Item_ID	VARCHAR(255)	primary composite key	referential to ITEM(Item_ID)	
	Usage_time	INT(11)			
	Usage_time_total	INT(11)			
Review	Review_ID	VARCHAR(255)	primary key		
	User_ID	VARCHAR(255)		referential to USER(User_ID)	
	Item_ID	VARCHAR(255)		referential to ITEM(Item_ID)	
	Recommend	INT(11)			
	Body	INT(11)			
	Helpful_score	VARCHAR(255)			
	Helpful_count	INT(11)			
	Posted_date	DATE			
Genre	Genre_ID	VARCHAR(255)	primary key		
	GenreName	VARCHAR(255)	unique- candidate key		
Item_genre	Item_ID	VARCHAR(255)	primary composite key	referential to ITEM(Item_ID)	
	Genre_ID	VARCHAR(255)	primary composite key	referential to GENRE(Genre_ID)	
Bundle	Bundle_ID	VARCHAR(255)	primary key		
	BundleName	VARCHAR(255)			
	BundlePrice	VARCHAR(255)			
	Final_price	VARCHAR(255)			
	Discount	VARCHAR(255)			
Bundle_item	Bundle_ID	VARCHAR(255)	primary composite key	referential to GENRE(Genre_ID)	
	Item_ID	VARCHAR(255)	primary composite key	referential to ITEM(Item_ID)	
Bundle_genre	Genre_ID	VARCHAR(255)	primary composite key	referential to GENRE(genre_id)	
	Bundle_ID	VARCHAR(255)	primary composite key	referential to GENRE(bundle_id)	
	Genre_count	INT(11)			
Tag	Item_ID	VARCHAR(255)	primary composite key	referential to ITEM(item_id)	
	TagName	VARCHAR(255)	primary composite key		
	Order	INT(11)			

FIG. 3: relational model

Relational model

Relation

먼저 ER diagram의 entity였던 User, Item, Review, Genre, Bundle, Tag를 relation으로 정의했다. Relationship들에 대해선 Binary m:n 관계인 Use, Genre_of, Bundle_of relationship은 각각 Use, Item_Genre, Bundle_Item로 정의하였으며 RecReview 의 경우 relation을 만들어야 하지만 데이터가 존재하지 않아 생략하였다. 번들이 속한 장르의 개수를 알기 위해선 Bundle_of와 Genre_of를 경유해서 derive해야하는데, 이 또한 Binary m:n 관계이기 때문에 Bundle_Genre relation을 정의했다. Item의 spec은 multi-valued attribute이기 때문에 새로운 relation Item_specs을 정의했다.

Key constraints

Foreign key constraints Item_specs와 Item_genre, Bundle_item은 Item을 reference 해야하기 때문에 primary key인 Item_ID를 foreign key로 설정했다. Use(User_item)와 Review는 User와 Item을 reference해야하기 때문에 primary key인 User_ID와 Item_ID를 foreign key로 설정했다. Bundle_genre는 Bundle과 Genre를 reference해야하기 때문에 primary key인 Bundle_ID와 Genre_ID를 foreign key로 설정했다. Tag는 Item을 reference해야하기 때문에 primary key인 Item_ID를 foreign key로 설정했다.

Primary key constraints User relation에선 두 candidate key중 User_ID를 primary key로 설정했다. Item relation에선 Item_ID를 primary key로 설정했다. Review relation에선 Review_ID를 primary key로 설정했다. Genre relation에선 Genre_ID를 primary key로 설정했다. Bundle relation에선 Bundle_ID를 primary key로 설정했다. Tag relation에선, Tag가 key를 가지지 않는 weak entity type 이기 때문에 Item의 primary key 이자 Tag의 foreign key인 Item_ID와 Tag의 partial key인 TagName을 composite하여 primary key로 설정했다.

Use relation에선 User의 primary key이자 Use의 foreign key인 User_ID와 Item의 primary key 이자 Use의 foreign key인 Item_ID를 composite하여 primary key로 설정했다. Item_specs relation에선 Item의 primary key이자 Item_specs의 foreign key인 Item_ID와 Item_spec을 composite하여 primary key로 설정했다. Item_Genre relation에선 Genre의 primary key이자 Item_Genre의 foreign key인 Genre_ID와 Item의 primary key이자 Item_Genre의 foreign key인 Item_ID를 composite하여 primary key로 설정했다. Bundle_Item relation에선 Bundle의 primary key이자 Bundle_Item 의 foreign key인 Bundle_ID와 Item의 primary key이자 Bundle_Item 의 foreign key인 Item_ID를 composite하여 primary key로 설정했다. Bundle_Genre relation에선 Bundle의 primary key이자 Bundle_Genre의 foreign key인 Bundle_ID와 Genre의 primary key이자 Bundle_Genre의 foreign key인 Genre_ID를 composite하여 primary key로 설정했다.

Variable domain

기본적으로 attribute의 domain을 설정할 때, Foreign key로 reference 되는 경우 형식을 통일했다.

그 외의 attribute들의 domain 대해서는 Code section에서 다시 설명하겠다.

Code

사이트 A의 데이터에 대해 ER diagram으로 나타낸 뒤, 실제 주어진 데이터에 대해 relational schema로 표현하는 것까지 마쳤다. 그 다음 과정은 주어진 스키마에 맞는 DB 테이블을 실제로 생성하여 데이터를 입력하는 것이다. 주어진 조건대로 mysql-connector-python 라이브러리만을 이용하여 python 코드를 통해 MySQL의 서버에 DB를 생성하였다. 관련 코드는 따로 첨부하였다. MySQL에 데이터를 입력하는 과정은 총 4가지 과정으로 이루어졌다. 첫째, 전체 데이터가 들어갈 schema 생성, 둘째, schema내부에 테이블 생성, 셋째, 생성한 테이블에 데이터 저장, 넷째, 한 번에 foreign

key 제약을 걸어주기, 이렇게 4가지 과정이었으며 각각에 대해 설명하겠다.

1. 스키마 생성

relational model의 모든 테이블이 들어갈 범주인 schema를 'DMA_team09'라는 이름으로 생성하였다. 이 때 이미 존재할 경우 생성 과정을 다시 수행하지 않도록 'IF NOT EXIST'문을 넣어주었다. 스키마 생성은 이것으로 마무리되었다.

2. 테이블 생성

relational model을 주어진 데이터셋에 맞게 수정한 형태 그대로 테이블에 반영하면 되었다. Relation 테이블은 USER, ITEM, USER_ITEM, REVIEW, GENRE, ITEM_GENRE, BUNDLE, BUNDLE_ITEM, BUNDLE_GENRE, TAG, ITEM_SPECS 순으로 생성하였다. 각각의 테이블은 생성 시에 이미 존재할 경우 그 과정을 다시 진행하지 않도록 'IF NOT EXIST'문을 넣어주었다. 또한 이미 constraint 문제가 없다는 것이 확인된 데이터셋이 주어진 상황이기에 Foreign key 조건은 테이블 생성시에 입력하지 않고 데이터를 넣은 이후로 미뤄두었다. 이는 데이터 이식 시에 시간을 줄이기 위함이다. 각 테이블은 주어진 데이터셋의 자료 및 순서에 맞게 column을 생성하였으며 자료형은 Requirement에 맞게 생성하였다.

데이터 타입은, string 타입은 VARCHAR(255)로, int 타입은 INT(11)로 조건에 맞게 맞추었지만 데이터의 의미를 고려하여 예외를 몇 가지 설정하였다. 첫 째, 데이터 중 유일성을 띄는 key에 해당하는 User_ID, Item_ID와 같은 고유한 이름은 숫자로 되어있다 하더라도 숫자의 연산이 필요한 상황이 아니며 유일한 문자열이라는 의미를 띄기에 VARCHAR(255)로 선언하였다. 둘 째, 0 또는 1의 값을 가지는 데이터는 TINYINT(1)을 사용하여 저장공간을 절약했다. 셋 째, 날짜 자료는 DATE 자료형으로, $ 또는 %가 붙은 데이터는 주어진 데이터셋에 가공을 가하지 않기 위해서 그대로 VARCHAR(255)에 문자열 타입으로 저장했다. 이 외에, NULL 값이 가능하다고 명시된 값을 외에는 전부 NOT NULL을 선언해주었다. 한편, 기본값이 주어져있는 자료형에 대해선 NULL값 가능 여부를 선언되지 않아도 되었다.

3. 데이터 저장

데이터 저장은 지정된 경로에 있는 csv 파일에의 접근, 데이터셋을 줄 간격으로 읽은 뒤 각 줄을 column 별로 잘라낸 뒤 선언했던 relation 테이블의 자료형에 맞게 전처리를 해주는 과정으로 진행했다. 우선 데이터를 저장하기 이전에 USE DMA_team09 선언을 통해 기본 스키마를 지정해주었다. 데이터 저장은 테이블 생성과 동일하게 USER, ITEM, USER_ITEM, REVIEW, GENRE, ITEM_GENRE, BUNDLE, BUNDLE_ITEM, BUNDLE_GENRE, TAG, ITEM_SPECS 순으로 진행하였다. 코드의 가독성을 높이기 위함이다. 가장 먼저 데이터셋이 저장된 경로를 설정해줬다. 데이터셋이 지정된 filepath에 dataset이라는 폴더로 존재한다는 가정하에 경로를 지정하였다. 경로에서 csv파일을 찾아 연 뒤, 파일을 한 줄 씩 읽어가며 각 줄에 대해 반복문을 시행했다. 반복문 내에서는 먼저 연결된 스트링 타입으로 있는 데이터의 endline이나 공백을 제거해주는 strip() 메소드를 사용했

으며, ','문자를 기준으로 잘라 리스트로 만들기 위해 split(',') 메소드를 사용했다. 그 후 생성된 리스트에 enumerate로 인덱스를 달아서 인덱스별로 가공을 진행했는데, 빈 데이터가 있을 경우 "null"문자열로 대체했으며, 그 뒤 칼럼 경우별로 데이터 테이블에 맞게 INT(11), TINYINT(1)에 들어가야 하는 자료들은 자료형을 int로 바꿔주었다. 그 외 VARCHAR(255), DATE 타입은 문자열 그대로 저장해도 상관이 없었다. 자료형을 맞춰준 뒤엔 리스트 형태의 instance를 MySQL에 저장 가능하도록 tuple 자료형으로 바꿔주었으며 그렇게 생성된 튜플을 포함하여 MySQL에 이식하는 INSERT문으로 명령문을 선언, 실행하였다. 이 과정에서 "null" 문자열로 대체되어 있던 null값은 스트링이 아닌 실제 null 값으로 바꾸기 위해 replace 메소드를 사용했다. 추가로 각 테이블에 데이터를 저장할 때마다 cnx.commit을 선언했다.

4. Foreign Key 제약 설정

우선 앞의 함수에서 생성한 DMA_team09 스키마를 기본 스키마로 사용하도록 선언한 뒤, 생성되어 있는 테이블을 ALTER문으로 수정하여 Foreign Key 제약을 추가해주었다. 이 제약의 경우 Relational Model에서 정리해둔 대로 선언하였다. 총 12개의 Foreign Key 제약이 있었다.

5. 코드 테스트

코드 테스트에 앞서 코드 실행에 기본적인 변수인 host, user, password, directory를 각각 설정해주었다. host는 기본값인 'localhost', user는 따로 설정하지 않았으므로 'root', 비밀번호는 설정한 대로 두었으며 파일 경로는 dataset 폴더가 있는 폴더로 지정해주었다. 코드 실행 시 약 30분 정도의 시간이 소요되었으며, 코드 실행이 끝까지 완료된 뒤에 MySQL Workbench로 서버에 접속하여 스키마, 테이블, 데이터, Foreign Key 제약이 제대로 입력되었음을 확인할 수 있었다.