****

데이터관리와 분석

2021, Spring, 박종헌 교수님

**Project #1 Report**

ER diagram 도식화 및 DB 구현

|  |  |
| --- | --- |
| **Professor** | 박종헌 |
| **Team No.** | 1조 |
| **Team Member** | 2015-18278 김세찬  2015-17748 이건희  2018-12130 홍지현  2017-16495 추성민 |
| **Date** | 12th, April, 2021 |

**Part1**

1. 문제 정의

사이트 A는 온라인 소프트웨어 유통망이다. 개발자들은 소프트웨어를 사이트 A에 출시하고, 사용자들은 사이트 A를 통해 소프트웨어를 구매하고 이용한다. 사이트 A가 사용하는 DB에 대한 ER diagram을 requirement를 충족시키며 도식화 하는 것을 목표로 한다. 문제를 해결하기 위한 전반적인 과정은 다음과 같은 순서로 진행된다. 주어진 데이터와 Requirement를 분석하여 entity와 attribute를 도출한다. Attribute를 분석하여 Redundancy를 최소화할 수 있는 entity간 relationship을 설정한다.

1. ER diagram 도식화 과정
   1. Requirement에서 Entity와 Attribute 도출
2. USER entity

(R1-1)에서 USER entity를 생성한다. Attributes로 고유번호(Id), 닉네임(Name), 프로필 이미지 존재 여부(Profile), 사용자가 이용한 이력이 있는 아이템의 총 개수(Items\_count)를 stored attributes로 설정하였다. 이 때, 고유번호(Id), 닉네임(Name)은 고유한 값을 가지므로 key로 설정하였다. 사용자가 작성한 리뷰의 수(# of reviews), 작성한 리뷰들이 받은 총 추천 수(# of recommends), 가입 이래로 사용자가 이용한 아이템들의 이용 시간의 총합(Usage time of items)은 주어진 데이터에 저장되어 있지 않고 다른 데이터에서 유도할 수 있기 때문에 derived attributes로 설정하였다.

(R1-3)에서 USER entity에 attribute를 추가한다. 사용자 id(User\_id), 아이템 id(Item\_id), 사용자가 최근 2주 간 아이템을 이용한 시간(2Weeks), 사용자가 가입 이래로 아이템을 이용한 총 시간(Total)을 구성 attributes로 하는 아이템 이용 정보(Usage information)를 Composite attribute로 생성한다.

1. REVIEW entity

(R1-2)에서 REVIEW entity를 생성한다. 고유번호(Id), 작성자 id(User\_id), 아이템 id(Item\_id), 사용자의 아이템 추천 여부(Recommend), 본문의 길이(Body), 게시 일자(Posted\_date), 리뷰가 받은 추천 수(Helpful\_count), 리뷰가 받은 평가 중 추천의 비율(Helpful\_score)을 stored attributes로 설정하였다. 이 때, 고유번호(Id)는 고유한 값을 가지므로 key로 설정하였다. 리뷰가 받은 총 평가 수(# of evaluations)는 주어진 데이터에 저장되어 있지 않고 다른 데이터에서 유도할 수 있기 때문에 derived attributes로 설정하였다.

1. ITEM entity

(R1-4)에서 ITEM entity를 생성한다. 고유번호(Id), 아이템의 이름(Name), 가격 정보(Price), 정식 출시 전 베타 버전의 유무(Beta), 사용자들에게 받은 평가 점수(Ratings), 외부에서 받은 평가 점수(Metascore), 개발사 정보(Developer), 출시 일자(Release), 스펙(Specs)를 stored attribute로 설정한다. 스펙(Specs)의 경우 하나의 아이템이 여러 스펙을 가질 수 있기 때문에 multivalued attribute로 설정하였다. 아이템이 받은 리뷰의 수(# of reviews), 아이템을 사용한 이력이 있는 사용자의 수(# of users), 아이템이 속해 있는 장르의 수(# of genres), 아이템에 붙어 있는 태그의 수(# of tags)는 주어진 데이터에 저장되어 있지 않고 다른 데이터에서 유도할 수 있기 때문에 derived attributes로 설정하였다.

1. GENRE entity

(R1-5)에서 GENRE entity를 생성한다. 고유번호(Id), 장르 이름(Name), 장르에 속하는 아이템 id(Item\_id)를 stored attribute로 설정한다. 고유번호(Id), 장르 이름(Name)은 고유한 값을 가지므로 key로 설정하였다. 장르에 속하는 아이템의 수(# of items)는 주어진 데이터에 저장되어 있지 않고 다른 데이터에서 유도할 수 있기 때문에 derived attributes로 설정하였다.

1. BUNDLE entity

(R1-6)에서 BUNDLE entity를 생성한다 부여한 고유번호(Id), 번들의 이름(Name), 가격 정보(Price), 할인율(Discount), 할인된 번들의 최종 가격(Final\_price), 번들에 포함되는 아이템 id(Item\_id)을 stored attributes로 설정하였다. 번들에 포함되는 아이템 id(Item\_id)는 하나의 번들에 여러 아이템이 속할 수 있기 때문에 multivalued attribute로 설정하였고, 고유번호(Id)는 고유한 값을 가지므로 key로 설정하였다. 번들에 포함되는 아이템의 수(# of items)는 주어진 데이터에 저장되어 있지 않고 다른 데이터에서 유도할 수 있기 때문에 derived attributes로 설정하였다. 번들에 포함되는 아이템들이 어느 장르에 몇 개나 속해 있는지에 대한 정보는 Bundle\_id, Genre\_id, Genre\_count를 구성 attributes로 하는 composite attribute로 설정하였다.

1. TAG entity

(R1-7)에서 TAG entity를 생성한다. Key attributes가 없기 때문에 weak entity type이다. 태그가 붙어 있는 아이템 id(Item\_id), 태그의 이름(Name)과 태그가 아이템에 붙어 있는 순서(Order)를 stored attributes로 설정하였다. 하나의 아이템에 붙어 있는 태그는 모두 다른 이름을 가지므로 태그의 이름(Name)은 partial key로 설정하였다.

* 1. Attribute에서 Relationship 도출

하나의 entity type의 attribute가 다른 entity type를 참조하는 경우를 찾아 해당 entity 사이에 relationship을 설정하였다.

1. USES Relationship

USER entity에 사용자 id(User\_id), 아이템 id(Item\_id), 사용자가 최근 2주 간 아이템을 이용한 시간(2Weeks), 사용자가 가입 이래로 아이템을 이용한 총 시간(Total)을 구성 attributes로 하는 Composite attribute 아이템 이용 정보(Usage information)가 존재한다. 사용자 id와 아이템 id attributes가 USER와 ITEM를 참조하고 있기 때문에 아이템 이용 정보 attribute를 USER entity와 ITEM entity를 연결하는 USES relationship으로 전환하였다. Cardinality ratio는 하나의 USER 인스턴스가 다수의 ITEM 인스턴스와 대응될 수 있고, 하나의 ITEM 인스턴스가 다수의 USER 인스턴스와 대응될 수 있기 때문에 M:N으로 설정하였다. Participation은 both partial, attributes는 2Weeks, Total로 설정하였다.

1. WRITES Relationship

REVIEW entity에 작성자 id(User\_id), 아이템 id(Item\_id) attributes가 USER와 ITEM를 참조하고 있기 때문에 USER, ITEM, REVIEW를 연결하는 WRITES ternary relationship을 생성하였다. Cardinality ratio는 REVIEW의 인스턴스 하나와 ITEM의 인스턴스 하나의 짝에 대응될 수 있는 USER의 인스턴스 개수는 1개, REVIEW의 인스턴스 하나와 USER의 인스턴스 하나의 짝에 대응될 수 있는 ITEM의 인스턴스 개수는 1개, USER의 인스턴스 하나와 ITEM의 인스턴스 하나의 짝에 대응될 수 있는 REVIEW의 인스턴스 개수는 N개 이므로 1:1:N으로 설정하였다. Participation은 USER&ITEM은 partial, REVIEW는 total로 설정하였다.

1. BELONGS\_TO Relationship

GENRE entity에 장르에 속하는 아이템 id(Item\_id) attribute가 ITEM을 참조하고 있기 때문에 GENRE와 ITEM을 연결하는 BELONGS\_TO Relationship을 생성하였다. Cardinality ratio는 하나의 ITEM 인스턴스가 다수의 GENRE 인스턴스와 대응될 수 있고, 하나의 GENRE 인스턴스가 다수의 ITEM 인스턴스와 대응될 수 있기 때문에 M:N으로 설정하였다. Participation은 ITEM은 partial, GENRE는 total로 설정하였다.

1. INCLUDES Relationship

BUNDLE entity에 번들에 포함되는 아이템 id(Item\_id) attribute가 ITEM을 참조하고 있기 때문에 BUDNLE와 ITEM을 연결하는 INCLUDES Relationship을 생성하였다. Cardinality ratio는 하나의 ITEM 인스턴스가 다수의 BUNDLE 인스턴스와 대응될 수 있고, 하나의 BUNDLE 인스턴스가 다수의 ITEM 인스턴스와 대응될 수 있기 때문에 M:N으로 설정하였다. Participation은 ITEM은 partial, BUNDLE는 total로 설정하였다.

1. HAS Relationship

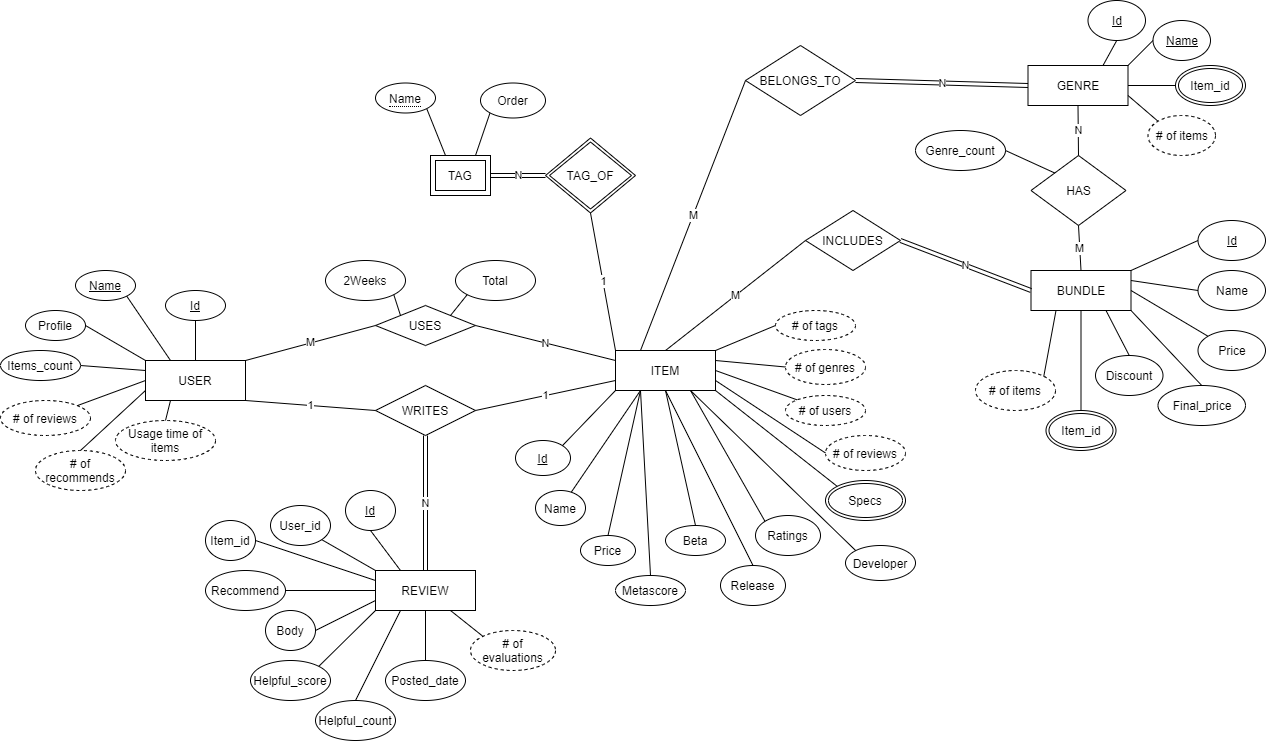
BUNDLE entity에 번들에 포함되는 아이템들이 어느 장르에 몇 개나 속해 있는지에 대한 정보는 bundle\_id, genre\_id, genre\_count를 구성 attributes로 하는 composite attribute로 설정하였다. bundle\_id, genre\_id가 BUNDLE과 GENRE를 참조하고 있기 때문에 번들에 포함되는 아이템들이 어느 장르에 몇 개나 속해 있는지에 대한 정보 attribute를 HAS Relationship으로 전환하였다. Cardinality ratio는 하나의 BUNDLE 인스턴스가 다수의 GENRE 인스턴스와 대응될 수 있고, 하나의 GENRE 인스턴스가 다수의 BUNDLE 인스턴스와 대응될 수 있기 때문에 M:N으로 설정하였다. Participation은 both partial, attributes는 Genre\_count로 설정하였다.

1. TAG\_OF Relationship

TAG entity에 태그가 붙어 있는 아이템 id(Item\_id) attribute가 ITEM을 참조하고 있기 때문에 TAG와 ITEM을 연결하는 TAG\_OF Relationship을 생성하였다. TAG는 weak entity이고 ITEM이 identifying entity이다. TAG\_OF는 identifying relationship이다. Cardinality ratio는 identifying relationship이기 때문에 1:N이 된다. Participation은 ITEM은 partial, TAG는 total이다. Attributes는 Name, Order로 설정하였다.

1. 최종 ER diagram

위의 과정을 거치며 도출된 최종 ER diagram은 아래와 같다.



**Part2**

1. **문제 정의**

Part 1에서 설계한 er diagram을 기반으로 실제 주어진 dataset을 이용하여 MySQL상에서 DB를 구축하는 작업을 수행한다. 주어진 dataset은 아래와 같았다.

user.csv: 사용자 정보

item.csv: 아이템 정보

user\_item.csv: 사용자가 아이템을 이용한 이력 정보

review.csv: 사용자가 아이템에 남긴 리뷰 정보

genre.csv: 장르 정보

item\_genre.csv: 아이템이 속한 장르 정보

bundle.csv: 여러 아이템을 묶어서 판매하는 번들 정보

bundle\_item.csv: 번들에 속한 아이템 정보

bundle\_genre.csv: 번들에 속한 아이템들이 가진 장르 정보

tag.csv: 태그 정보

item\_specs.csv: 스펙 정보

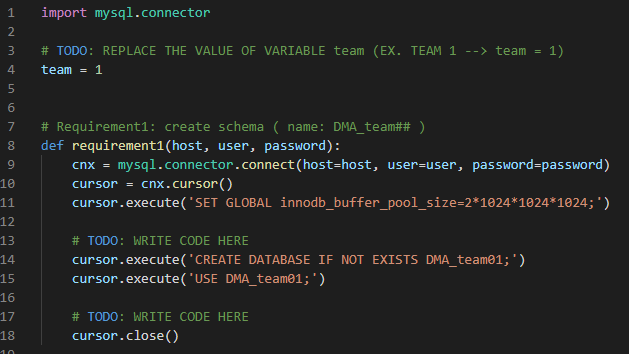
위 dataset들 중

item\_genre, bundle\_item, bundle\_genre는 각각 Part1의

BELONGS\_TO, INCLUDES, HAS relation에 대응되는 개념이다.

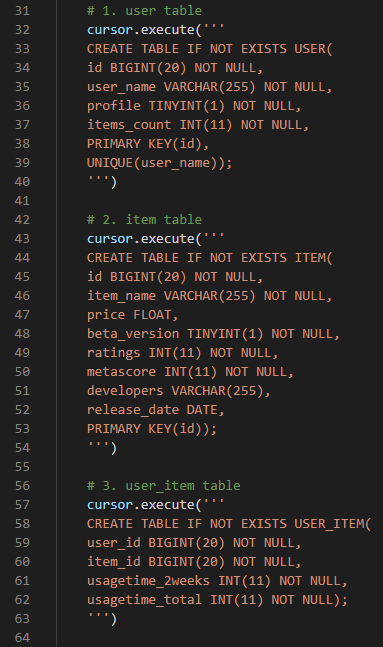
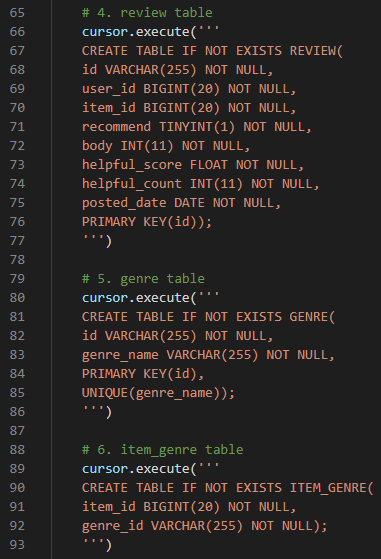
Part1에서는 USER, ITEM, REVIEW를 ternary relation으로 연결하였으나 part2에서는 주어진 dataset에 맞추어 user\_item relation을 이용하여 DB를 구축하였다.

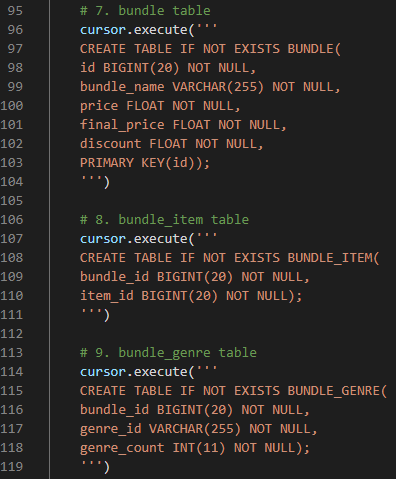
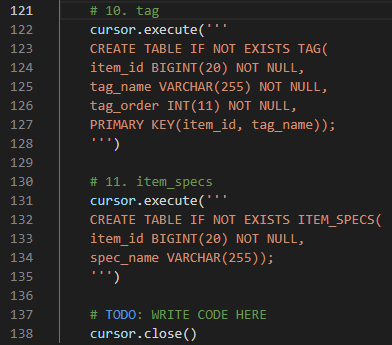
(R2-1)



주어진 조건에 맞추어 DMA\_team01의 이름을 가지는 schema를 생성하였다. Schema가 존재할 경우 생성 과정을 다시 수행하는 것을 방지하기 위해 CREATE DATABASE IF NOT EXISTS구문을 사용하였다.

(R2-2)

각 dataset에 해당하는 테이블이 존재할 경우 재생성을 막기 위하여 CREATE TABLE IF NOT EXISTS구문을 사용하여 TABLE을 생성하였다.

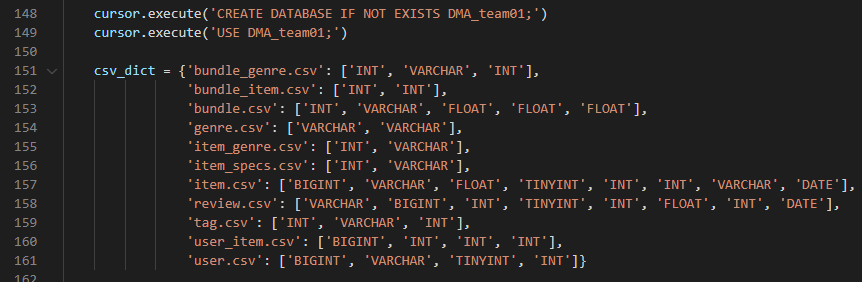
User id, item id, bundle id의 경우 정수형의 데이터들로 구성되고 매우 큰 값을 가지는 경우가 있으므로 자료형을 BIGINT로 설정하였으며, Review id, genre id의 경우 문자와 숫자가 혼합된 데이터로 구성되므로 VARCHAR자료형으로 지정하였다. 그 외의 name등의 text정보 또한 VARCHAR 자료형으로 지정하였으며, posted\_date와 같은 날짜의 경우 DATE자료형으로 지정하였다. Profile, beta\_version, recommend는 0과 1의 값만 가지므로 TINYINT 자료형으로 지정하였으며, 그 외의 정수형 자료는 INT로 저장하였다. Price, discount, helpful\_score등 소수점을 가질 수 있는 자료형들은 FLOAT으로 지정하였다.

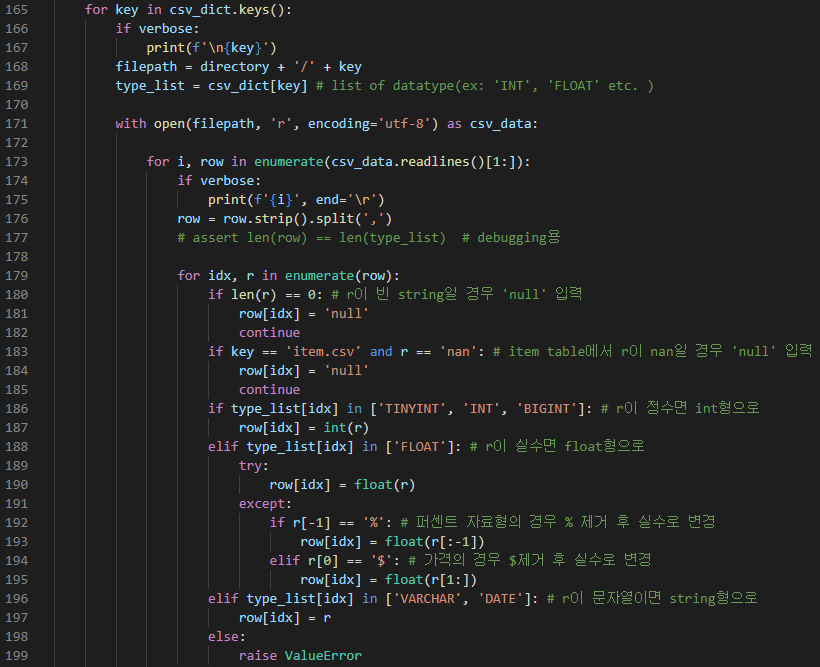
또한 requiremet에서 명시적으로 null값을 가질 수 있다고 언급한 item table의 price, developers, release\_date, 그리고 item\_spec table의 spec\_name을 제외한 모든 column에 not null 조건을 부여하였다.

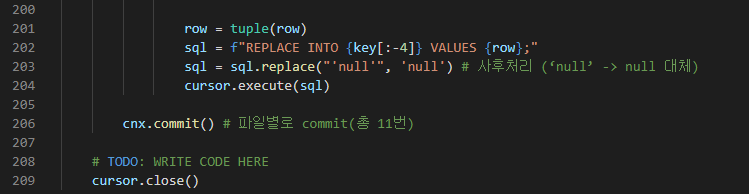
Relation이 아닌 entity들의 primary key 설정 정보는 아래와 같다.

1. User table: id
2. Item table: id
3. Review table: id
4. Bundle table: id
5. Tag table: (item\_id, tag\_name) > tag는 weak entity로 item과의 relation하에서 특정된다.

(R2-3)

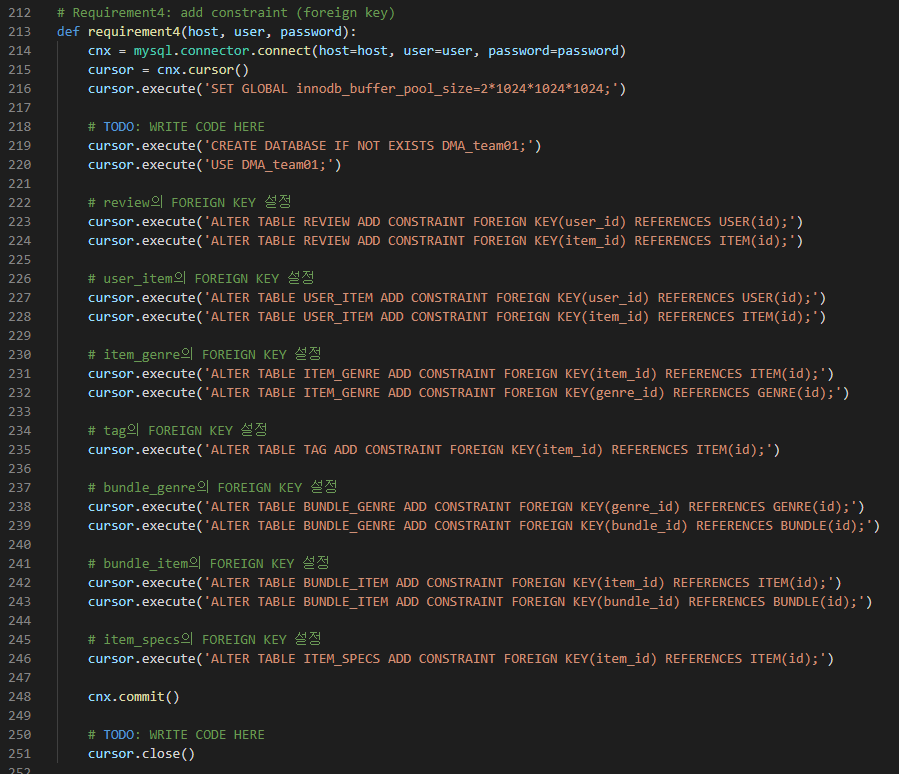






Requirement 3 파트에서는 주어진 dataset의 정보를 requirement2 파트에서 구축한 MySQL상 table에 입력하는 과정을 진행하였다. 우선 requirement2에서 지정한 각 table의 이름을 key로 가지고 column의 자료형들을 value list로 가지는 dictionary를 선언하였다. 이를 활용하여 key에 해당하는 dataset을 받을 때 ‘,’ 로 구분되어 있는 각 column의 자료형들을 각각 개별적으로 처리해 줄 수 있었다. 예외적으로 %자료형과 $기호가 포함되어 있는 가격정보의 경우 try-except구문을 이용하여 각각 기호를 제거하고 실수 단위로 저장할 수 있도록 처리하였다.

(R2-4)



part 4 에서는 테이블들 중 외부객체를 참조하는 column이 있는 경우 해당 정보를 foreign key로 반영해주었다. 각 dataset의 column중 외부키를 참조하는 column정보는 아래와 같다.

(전체 data column명들중 foreign key 설정부분은 형광색으로 표시)

**Bundle**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | bundle\_name | price | final\_price | discount |

**Bundle\_genre**

|  |  |  |
| --- | --- | --- |
| bundle\_id | genre\_id | genre\_count |

**Bundle\_item**

|  |  |
| --- | --- |
| bundle\_id | item\_id |

**Genre**

|  |  |
| --- | --- |
| id | genre\_name |

**Item**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | item\_name | price | beta\_version | ratings | metascore | developer | release\_date |

**Item\_genre**

|  |  |
| --- | --- |
| item\_id | genre\_id |

**Item\_specs**

|  |  |
| --- | --- |
| item\_id | spec\_name |

**Review**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | user\_id | item\_id | recommend | body | helpful\_score | helpful\_count | posted\_date |

**Tag**

|  |  |  |
| --- | --- | --- |
| item\_id | tag\_name | tag\_order |

**User**

|  |  |  |  |
| --- | --- | --- | --- |
| id | user\_name | profile\_image | items\_count |

**User\_item**

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id | item\_id | usagetime\_2weeks | usagetime\_total |

위의 데이터를 근거로 총 12개의 foreign key를 설정해주었다.