Assignment2 - Relational Modeling

컴퓨터소프트웨어학부 2021088304 박현준

이번 과제는 첫 번째 과제에서의 Conceptual Design과 추가된 조건들을 바탕으로 Relational Diagram과 SQL Script를 작성하는 과제 입니다.

명세에 주어진 추가 조건들은 다음과 같습니다.

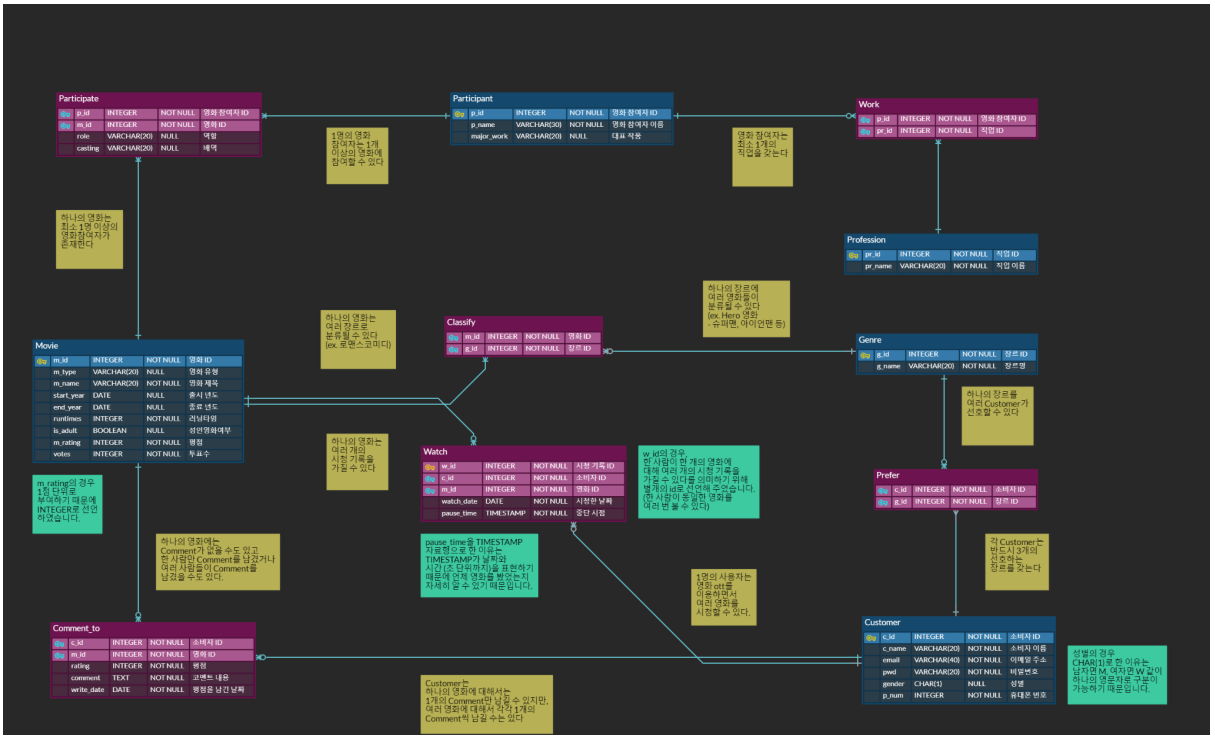
- 각 Customer는 0개 또는 여러 개의 선호 장르를 갖는다

  → 각 Customer는 반드시 3개의 선호 장르를 갖는다
- Customer가 삭제되면, Customer가 작성한 comment에 대한 정보도 삭제된다.

이 조건들을 보았을 때, 첫 번째 조건은 1대다 관계인데 3개로 고정되어 있으므로 Memo에 각 Customer는 3개의 선호 장르를 갖는다고 적었고, SQL Script에서는 이 부분에 대한 Constraint 조건을 부여 해야 한다고 생각을 했습니다.

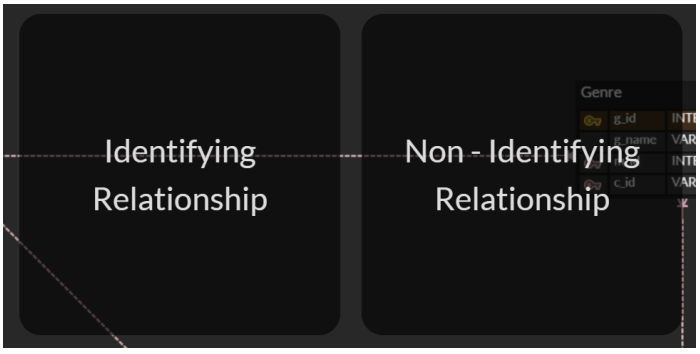
그리고 두 번째 조건인 Customer와 Comment간의 관계는 Comment가 Customer에 dependent한 관계이기 때문에 ERDCloud에서의 **파란색 실선 (Identifying relationship)**으로 표현을 하였습니다.

전체 Diagram



Crow-foot notation과 설명이 필요한 내용의 경우 ERD Cloud의 memo 기능을 이용하여 설명해 두었습니다. (Crow-foot : **노란색** / 설명이 필요한 내용 : **초록색**)

Relationship



ERDCloud에서는 두 Table을 연결하게 되면 Relationship에 대해 선택을 하도록 되어 있습니다.

1. **Identifying relationship**

   먼저 Identifying relationship의 경우 **파란색 실선**으로 표현이 되는데, 부모 table의 key를 자식 table이 자신의 기본키로 사용하는 관계를 의미합니다. 종속 관계 또는 상속 관계로 해석도 가능한데, 그 말인 즉슨 부모 table이 삭제 되면 자식 table도 삭제가 된다는 의미로 해석할 수 있습니다.

   따라서, 추가 조건 중 comment에 관한 조건이 이에 부합하기 때문에 Customer와 Comment는 Identifying relationship으로 선언해주었습니다.

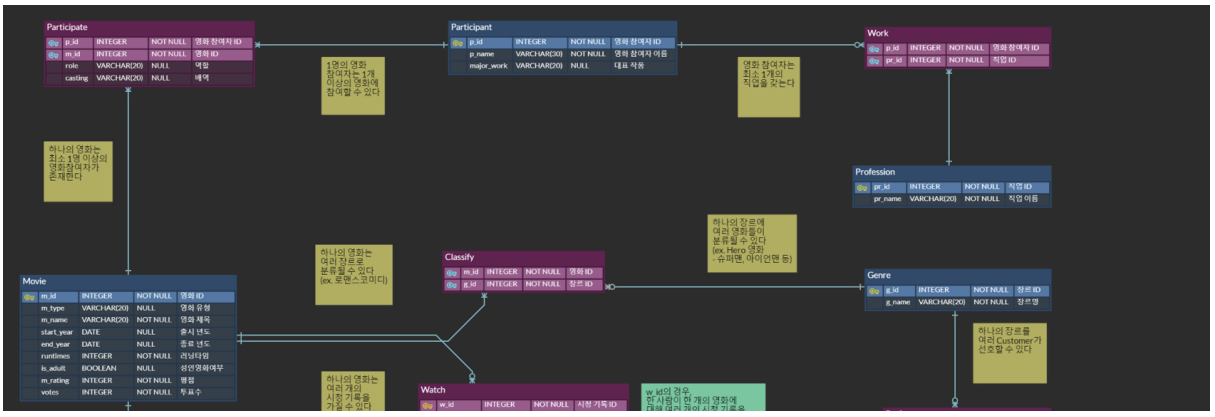
   추가적으로, Participant와 Profession의 경우도 Identifying relationship으로 선언을 하였는데 그 이유는 Profession이 Participant의 직업을 의미하기 때문에 Participant에 dependent하다고 생각해서 이렇게 선언을 하였습니다.

2. **Non-Identifying relationship**

   대부분의 경우는 Non-Identifyinh relationship에 해당되며, **분홍색 점선**으로 표시를 합니다. 이 경우는 부모 table의 key를 자신의 기본키가 아닌 외래키로 사용을 하는 관계를 의미합니다. 즉, Identifying relationship에서의 두 table의 관계는 dependent 관계였다면, Non-Identifying relationship에서는 independent한 관계임을 의미합니다.

여기서 부터는 각각의 relationship을 어떤 이유에서 Identifying / Non-Identifying으로 선언했는지 설명하도록 하겠습니다. 다만, Crow-foot notation의 경우 memo에 설명을 해두었기 때문에 이 부분에서는 relationship 위주로만 설명을 하도록 하겠습니다.

먼저 Diagram에서 Movie를 기준으로 위쪽부터 설명을 하면,



1. Movie - Participate - Participant

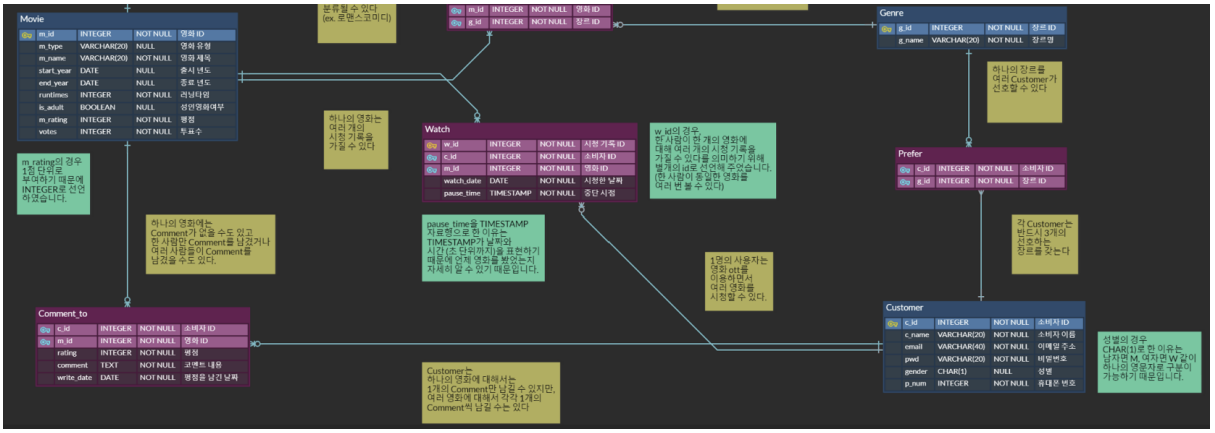
   영화 참여자는 영화 제작에 참여하는 모든 사람들이기 때문에 영화가 삭제되면 삭제된 해당 영화를 제작했던 참여자들에 대한 정보도 영화와 함께 삭제되어야 한다고 생각해서 **Identifying relationship**으로 선언을 하였습니다.

2. Participant - Work - Profession

   영화 참여자들의 직업의 경우 영화에 참여하는 배우, 감독, 작가 그리고 이에 해당하지 않는 의사, 변호사, 검사 등 여러가지가 존재합니다. 이 과제에 한정해서 제가 이해한 바는 영화에 참여한 사람들의 직업에 한정해서 relationship을 구현한 것이라고 생각을 해서 Participant가 삭제되면 Participant의 Profession에 대한 정보도 함께 삭제가 되는 종속관계로 판단을 해서 **Identifying relationship**으로 선언을 하였습니다.

3. Movie - Classify - Genre

   다대다 관계이고, Movie와 Genre는 Movie가 여러 Genre로 classify 될 수 있다는 관계를 갖고 있습니다. 예를 들어, `Matrix` 라는 영화를 놓고 봤을 때, `Matrix` 는 SF 영화이므로 `Matrix` 영화가 삭제 된다면 이 영화의 장르인 SF도 삭제가 된다고 생각을 해서 종속관계인 **Identifying relationship**으로 선언을 해주었습니다.



4. Movie - Watch - Customer

   Movie와 Customer 사이의 관계로는 comment를 작성하고 확인할 수 있다와 customer가 movie를 볼 수 있다 2가지 관계가 있습니다. 이 두 관계 모두 다대다 관계인데, **Customer가 삭제되면 Customer가 시청했던 영화의 기록도 함께 삭제가 되기 때문에** 종속관계라고 생각을 해서 **Identifying relationship**으로 선언을 하였습니다.

5. Movie - Comment_to - Customer

   이 관계 역시 Watch와 동일한 이유로 **Identifying relationship**으로 선언을 하였습니다.

6. Customer - Prefer - Genre

   Customer는 무조건 3개의 선호 장르가 존재한다는 조건이 존재하고, 4번과 5번에서의 이유와 동일하게 Customer가 선호하는 Genre이므로 Customer가 삭제되면 Customer가 선호하는 장르에 대한 정보 역시 삭제된다고 생각을 해서 **Identifying relationship**으로 선언을 하였습니다.

Null / Not Null

Null 여부의 경우, 살짝 단순하게 생각을 해서 식별자(ID)나 이름과 같이 하나의 Table에서 필수적인 정보의 경우 NULL이 되면 안되기 때문에 NOT NULL로 선언을 하였고, 반대로 성인 영화 여부나 개봉 년도와 같이 없어도 어떤 대상인지 판별이 가능한 경우에 대해 NULL로 선언을 해주었습니다.

SQL Script

SQL Script는 Assignment2.sql 파일에 있습니다. Modeling한 내용을 export한 내용과 동일하며, 명세에 주어진 추가 조건에 대해서 SQL Script를 수정한 부분에 대해서만 설명하도록 하겠습니다.

각 Customer는 반드시 3개의 선호 장르를 갖는다

```
301 CREATE OR REPLACE FUNCTION ensure_three_genres()
302 RETURNS TRIGGER AS $$
303 BEGIN
304     IF (SELECT COUNT(*) FROM "Prefer" WHERE c_id = NEW.c_id) <> 3 THEN
305         RAISE EXCEPTION '각 Customer는 반드시 3개의 선호 장르를 가져야 합니다.';
306     END IF;
307     RETURN NEW;
308 END;
309 $$ LANGUAGE plpgsql;
310
311 CREATE TRIGGER check_three_genres
312 BEFORE INSERT OR UPDATE ON "Prefer"
313 FOR EACH ROW
314 EXECUTE FUNCTION ensure_three_genres();
```

먼저 함수 정의 부분을 보면, `TRIGGER`를 `AS $$ $$` 사이의 내용으로 return을 하겠다는 의미입니다. `ensure_three_genres( )` 함수에 대한 선언 내용이고 TRIGGER가 호출이 되면 `ensure_three_genres( )`가 실행이 되고 `BEGIN`부터 `END` 사이의 code가 동작하게 됩니다.

조건문을 보면,

```
IF (SELECT COUNT(*) FROM "Prefer" WHERE c_id = NEW.c_id) <> 3 THEN
    RAISE EXCEPTION '각 Customer는 반드시 3개의 선호 장르를 가져야 합니다.';
END IF;
RETURN NEW;
```

`c_id = New.c_id` 즉, customer의 id가 새로운 customer의 id와 같은 사람 (새로 추가된 사람)의 "Prefer"에 대한 count를 세서 3이 아니면 선호하는 장르가 반드시 3개여야 한다는 명세 조건에 위배되므로 `EXCEPTION` message를 발생시키고 더 이상 선호하는 장르가 추가되지 못하도록 하였습니다. 이를 통해 각각의 customer가 무조건 3개의 선호하는 장르를 갖도록 하였습니다.

그 다음으로 `TRIGGER` 부분을 보면, `BEFORE INSERT OR UPDATE ON "Prefer"` 라는 문구에서 알 수 있듯이 해당 table에 새로운 record가 추가되거나 update가 되기 전에 "Prefer" table에서 `check_three_genres`를 호출하는 역할을 합니다. 그 뒤 각각의 행에 대해 `ensure_three_genres( )` 함수를 실행합니다.

Customer가 삭제되면, Customer가 작성한 정보도 삭제된다.

```
ALTER TABLE "Comment_to"
ADD CONSTRAINT fk_customer
FOREIGN KEY (c_id)
REFERENCES "Customer" (c_id)
ON DELETE CASCADE;
```

이 부분에 대한 구현입니다.
코드에 대한 설명을 하면 Customer가 작성한 정보의 경우, 영화에 대한 Comment가 존재합니다. 그러므로 Comment_to table에 Customer의 Primary Key `c_id` 를 참조하는 Foreign Key Constraint를 추가하고, `ON DELETE CASCADE` 를 선언해서 Customer에서 특정 row가 삭제될 때 해당 Customer와 관련이 있는 모든 Comment_to에 있는 정보들(rating, comment, write_date)도 함께 삭제가 되도록 하였습니다.

결론적으로, `ensure_three_genres( )` 함수와 `check_three_genres` TRIGGER로 각각의 고객이 Prefer table에서 정확히 3개의 장르만 선호하도록 constraint를 걸게 됩니다.

---

pgAdmin4에서 ERDCloud에서 export한 postgreSQL을 실행시켜보면 오류 없이 잘 실행이 된다는 것을 확인할 수 있습니다.