텍스트이(가) 표시된 사진

자동 생성된 설명

데이터베이스설계 실습

제 목

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2021년 12 월 12 일

학부 정보통신공학과

학년 3

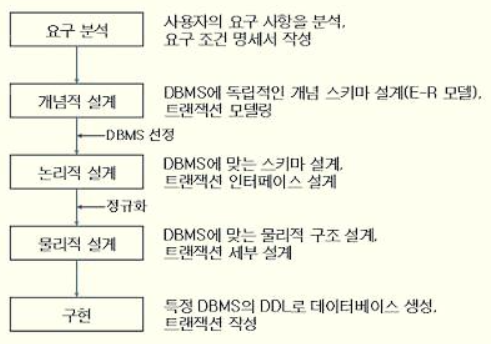
성명 고병찬

학번 12171748



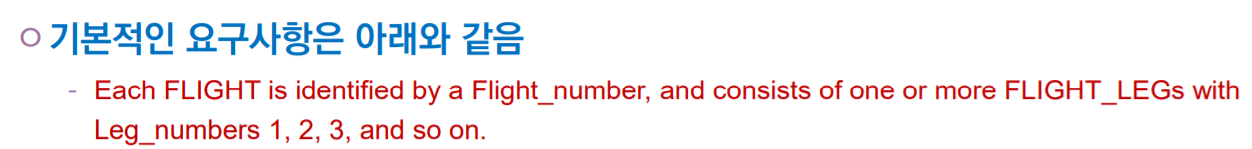
텍스트이(가) 표시된 사진

자동 생성된 설명

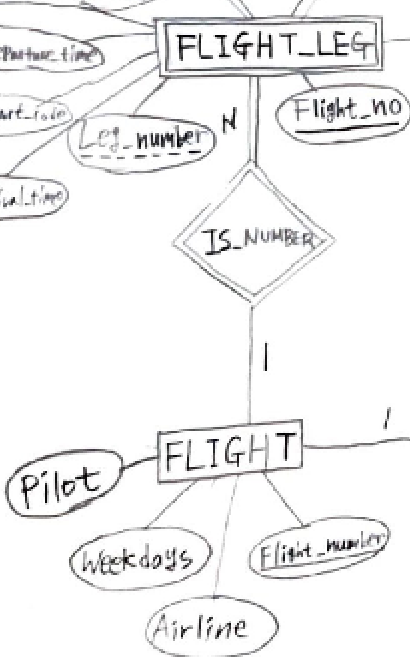


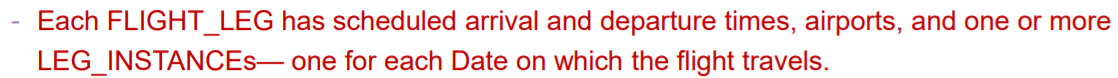
데이터베이스 설계 단계는 위와 같기 때문에 사용자의 요구사항 분석 및 요구 조건 명세서 작성을 해야 하지만, 이미 요구사항, 요구 조건 명세서와 DB 스키마가 주어졌기 때문에 먼저 요구사항 및 요구 조건 명세서를 분석하여 E-R모델을 그린 후, DB 스키마를 설계할 계획이다.

1. 요구사항 분석 + ERD 그림에 대한 설명



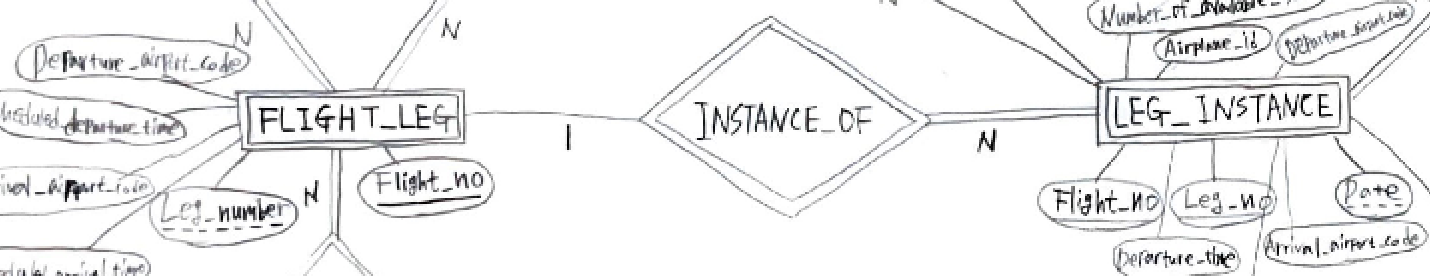
각 비행은 flight\_number로 구분(식별)된다. 그리고 하나의 FLIGHT는 FLIGHT\_LEGs 를 적어도 한 개 이상 가진다. FLIGHT table에 대한 설명: Flight\_number가 기본키. 하나의 FLIGHT에 여러 개의 FLIGHT\_LEGs가 있다는 말은 FLIGHT과 FLIGHT\_LEGs가 1:n관계로 볼 수도 있고, FLIGHT\_LEG를 다치로 볼 수도 있다. 비행 한 번에 구간이 1개 있으면 직항, 구간이 2개 있으면 한 번 경유, 구간이 3개 있으면 두 번 경유하는 것을 의미한다.

요구 조건에 따라 좌측 사진과 같이 FLIGHT\_LEG의 기본키는 Flight\_no로 설정하였고, 각 FLIGHT는 하나 이상의 FLIGHT\_LEG를 가지기 때문에 다치 속성으로 볼 수도 있지만 FLIGHT\_LEG를 다치 속성으로 보게 되면 1NF를 만족하지 않기 때문에 정규화를 위해 1 : N 관계로 설정하였다. FLIGHT table이 FLIGHT\_LEG의 개수를 가지기 때문에 관계 이름으로 IS\_NUMBER를 설정했다. FLIGHT\_LEG는 FLIGHT이 존재하지 않으면 존재할 수 없는 table이기 때문에 Weak\_entity로 정해줬다. 요구사항에 따라 이미 1 : N으로 관계를 설정하였지만, Weak\_entity이기 때문에 당연히 1 : N 관계가 될 것이고, 전체 참여가 된다. 그리고 말로 의미를 이해해도 비행은 항상 있는 것이 아니기 때문에 부분 참여, 비행 구간은 비행이 없어도 항상 존재해야 하기 때문에 전체 참여로 설정해주었다.

****

각 FLIGHT\_LEG는 예정된 도착 및 출발 시간, 도착 및 출발 공항, 및 하나 이상의 LEG\_INSTANCEs가 있다. LEG INSTANCE는 비행편이 이동하는 각 날짜에 대해 하나씩 있다.

FLIGHT\_LEG table은 scheduled arrival and departure times, arrival and departure airports가 attribute이고, LEG\_INSTANCE table과 1:n관계를 의미한다. 하나의 비행 구간에 여러 개의 비행 구간의 특정한 날짜가 존재할 수 있기 때문에 다치로 볼 수도 있다.

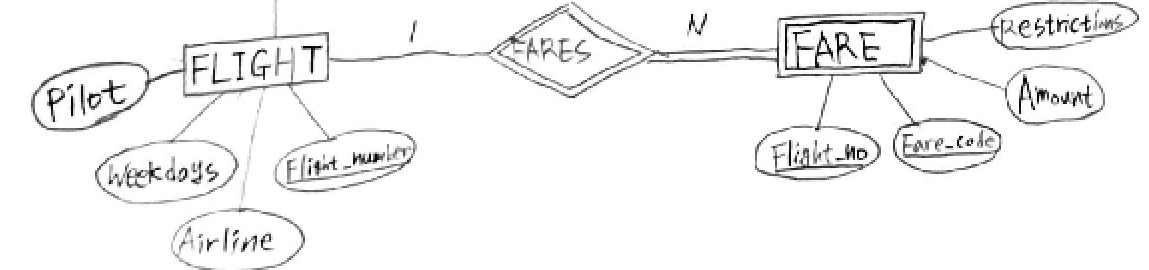


요구 조건에 따라 FLIGHT\_LEG table에 여러 속성들을 추가하고, LEG\_INSTANCE와 무슨 관계인지 나타내었다. 위에서 말했듯이, LEG\_INSTANCE table을 FLIGHT\_LEG의 다치 속성으로 볼 수도 있지만, 다치 속성으로 보게 되면 1NF를 만족하지 않기 때문에 정규화를 위해 1 : N관계로 나타내었다. 그리고 LEG\_INSTANCE table은 FLIGHT\_LEG table이 존재하지 않으면 존재할 수 없기에 Weak Entity로 표현하였다. LEG\_INSTANCE는 FLIGHT\_LEG의 Weak entity이기 때문에 전체 참여에 해당하고, FLIGHT\_LEG는 날짜에 대한 정보가 없어도 존재할 수 있기에 부분참여로 한다.



요금은 각 항공편에 보관된다.

비행의 입장에서는 한 번의 비행에 요금을 갖고 있다는 의미라서 FLIGHT의 다치 속성으로 볼 수 있다.

****

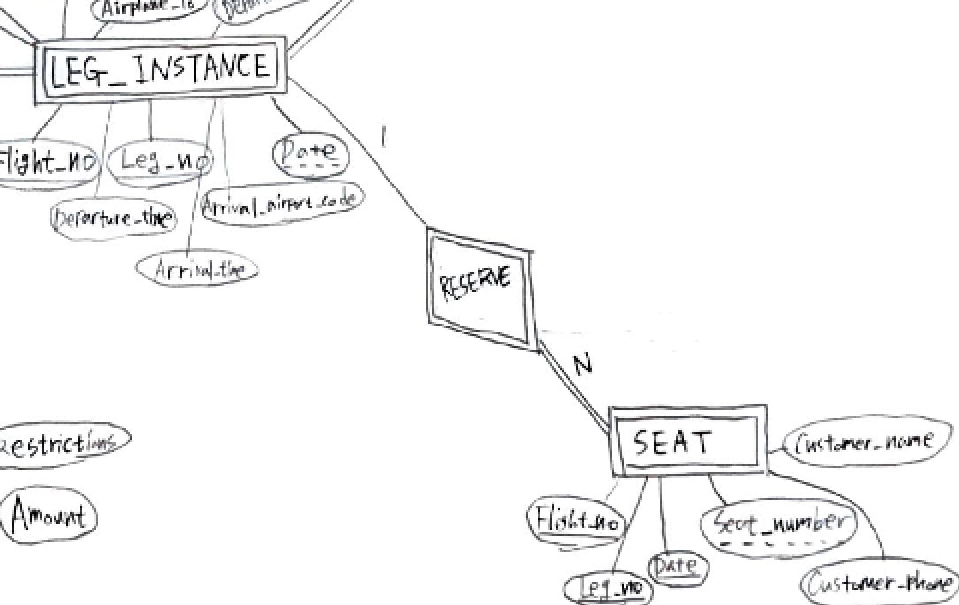
요구 조건에 따라 요금에 대한 정보인 FARE는 FLIGHT의 다치 속성이다. 하지만 다치 속성으로 정하면, 1NF를 만족하지 않기 때문에 1 : N 관계로 표현하였다. 게다가 FLIGHT가 존재하지 않으면 요금은 존재할 수 없기 때문에 Weak entity로 설정하였다. 비행은 요금이 없어도 존재할 수 있어서 부분참여로 표현했지만, 요금은 weak entity이기 때문에 전체 참여로 표현하였다.

****

각 FLIGHT\_LEG instance에 대하여, 구간에 사용된 항공기와 실제 도착, 출발 시간 및 공항과 마찬가지로 좌석 예약이 보관된다. Flight\_leg instance 테이블에 있는 attributes를 설명하는듯. 문장을 보면, SEAT\_RESERVATIONs만 따로 설명되어 있는데 이것은 SEAT\_RESERVATIONs가 weak entity임을 설명한다.

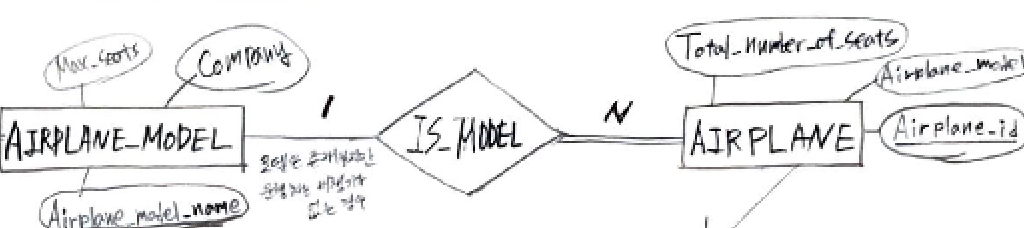
위 설명을 보면, FLIGHT\_LEG instance라고 되어 있어서, FLIGHT\_LEG table의 사례로 해석하여 FLIGHT\_LEG table에 대한 이야기인 줄 알았지만, 계속 고민하고 공부한 결과, LEG INSTANCE table에 대한 이야기임을 알아냈다.

LEG\_INSTANCE table에 좌측 사진과 같이 attribute에 대한 정보를 넣어줬고, 날짜에 대한 구간의 정보가 없으면, 좌석도 존재할 수 없기에 Weak entity로 표현하였다. Weak entity이기 때문에 전체 참여에 1 : N 관계로 표현하였다. 관계는 RESERVE로 표현하였다. FLIGHT\_LEG에서 예정된 출발 및 도착 시간을 보여줬다면, LEG\_INSTANCE에서는 실제 출발 및 도착시간을 보여준다.





비행기는 airplane\_id로 식별되며 특정한 airplane\_type에 속한다. Airplane\_id가 airplane의 기본키.



TYPE이라는 말보다는 MODEL이라는 말이 더 깔끔하기 때문에 MODEL로 수정하였다. 하나의 모델에 여러 개의 AIRPLANE이 존재하기 때문에 1 : N의 관계로 표현하였고, AIRPLANE은 LEG\_INSTANCE와 연결되어 있기 때문에 실제로 운항하는 비행기이다. 따라서 AIRPLANE\_MODEL은 존재하지만 운행하지 못하는 AIRPLANE이 있을 수 있기에 부분 참여가 된다. 하지만 AIRPLANE은 무조건 어떠한 비행기 모델일 수 밖에 없기 때문에 전체 참여이다.



CAN\_LAND는 착륙할 수 있는 공항과 PLANE\_TYPE을 관련시킨다. CAN\_LAND가 AIRPLANE table과 AIRPLANE\_TYPE table을 연결시킨다는 말. 즉, 관계라는 의미이다.



위 조건도 마찬가지로 TYPE보다는 MODEL이라는 말이 더 깔끔하기 때문에 AIRPLANE\_MODEL로 바꿨다. 여러 공항이 여러 모델의 비행기를 활용하기 때문에 M : N관계이다. M : N 관계는 table로 표현해야 하기 때문에 CAN\_LAND라는 table을 작성할 예정이다. 비행기가 무조건 공항에 착륙하지 않을 수 있기 때문에 부분 참여이고, 착륙하지 않는 비행기 모델도 존재할 수 있기 때문에 부분 참여이다.



AIRPORT table은 Airport\_code로 구분된다. AIRPORT table의 기본키가 Airport\_code라는 의미이다.

좌측 사진과 같이 AIRPORT table의 기본키를 Airport\_code로 설정하였다.



텍스트이(가) 표시된 사진

자동 생성된 설명

Query에 제공된 Flight number, leg number, arrival airport code, departure들은 FLIGHT\_LEG table에 존재한다. 만약 그렇지 않다면 FLIGHT LEG table에 먼저 삽입되어야 한다. FLIGHT\_LEG table의 속성들에 대해 이야기하고 있다.



Airplane\_id 값이 삽입된 대로 airplane id가 AIRPLANE relation 안에 있다. Airplane\_id가 AIRPLANE relation의 기본키이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Airplane\_id가 airplane type을 제공한다. 이 airplane type은 비행기가 착륙할 공항에 착륙할 수 있어야 한다. 다시 말해, AIRPLANE\_TYPE의 속성인airplane\_id가 CAN\_LAND table의 속성인

Airplane\_type\_name을 가리켜야 한다. 즉, FK, PK관계를 알려주고 있다. CAN\_LAND table의 Airplane\_type\_name, Airpot\_code는 둘 다 FK이자 PK이다. CAN\_LAND 관계로 이어져 있는 두 개의 table의 속성은 둘 다 PK이다. CAN\_LAND는 관계인데 table로 표현되어 있다. M:N 관계이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 요구사항에서 언급했듯이, AIRPORT table과 AIRPLANE\_MODEL table이 M : N 관계이기 때문에 CAN\_LAND 관계를 table로 표현하였다. 다른 속성이 없기 때문에 CAN\_LAND table은 AIRPORT table과 AIRPLANE\_MODEL table의 key로만 구성되어 있다. CAN\_LAND table은 두 속성 모두 PK이자 FK이다.



사용가능한 좌석 수는 AIRPLANE\_TYPE table의 좌석 수가 맞아 떨어져야 한다. 이 말을 이해해보면, 비행기 모델이 처음 만들어질 때, 최대 좌석 수를 정하고 비행기가 만들어졌고, 사용가능한 좌석 수는 LEG\_INSTANCE의 속성이기 때문에 사용자가 예약하는 과정에서 신경쓰면 되는 부분이다.



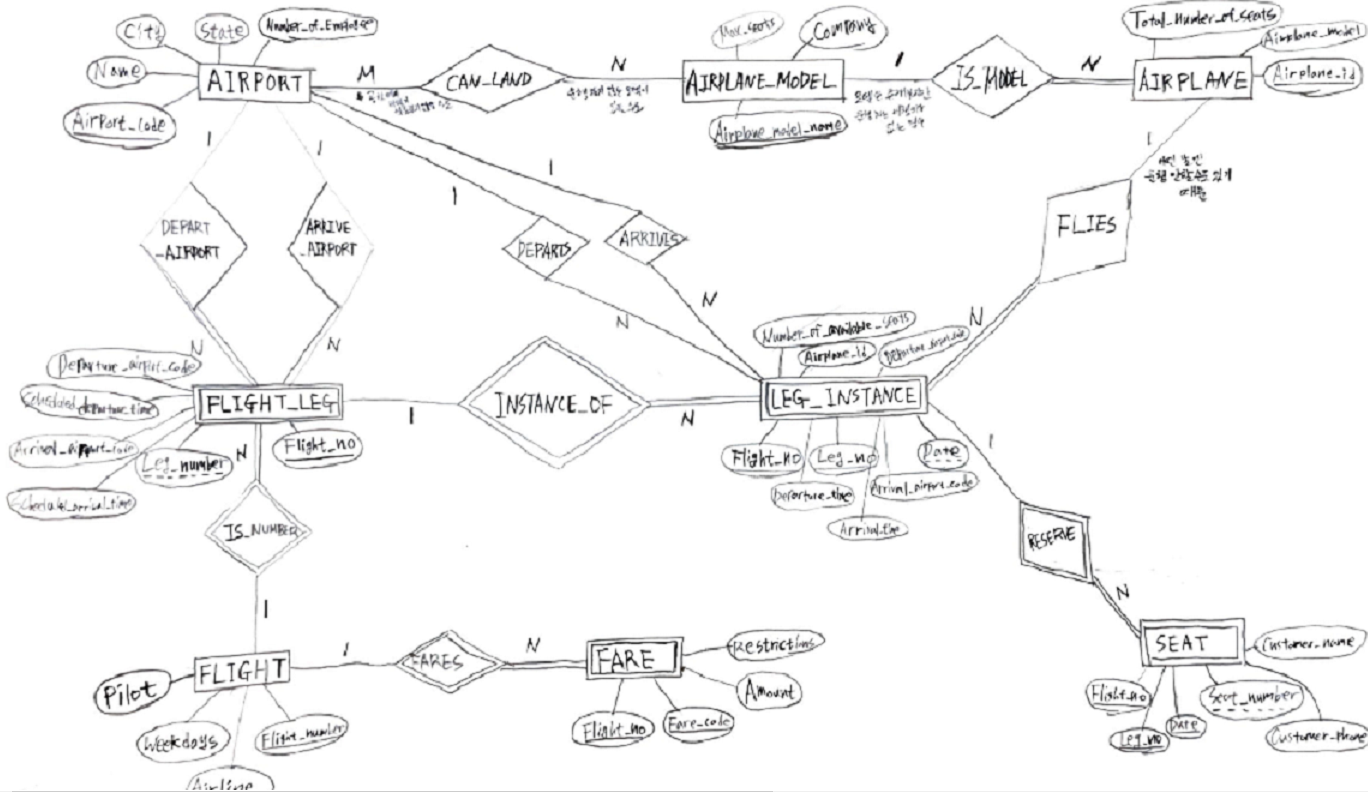
FLIGHT table에는 삽입된 비행 번호와 같은 비행 번호가 있다. 당연한 말이다.



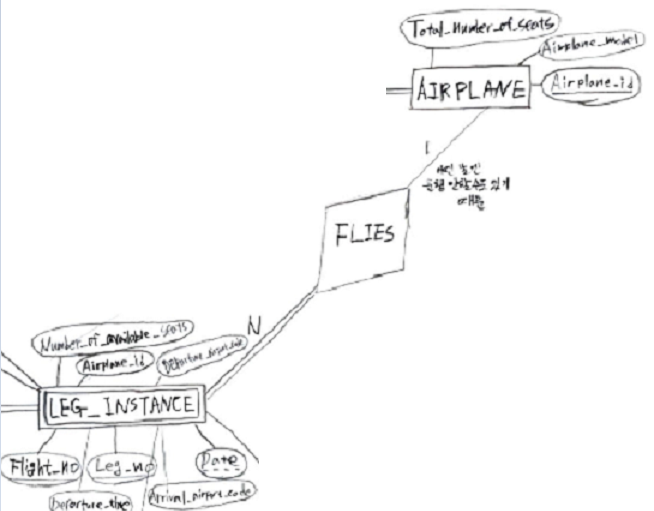
AIRPORT table에는 출발 airport code와 같은 airport code가 있다. FLIGHT\_LEG table의 departure airport code와 LEG\_INSTANCE의 departure airport code를 말하는 것이다. FLIGHT\_LEG table에 출발하는 공항이 존재하고, LEG\_INSTANCE table에도 출발하는 공항이 존재하기 때문에 모두 관계로 엮어줘야 한다.



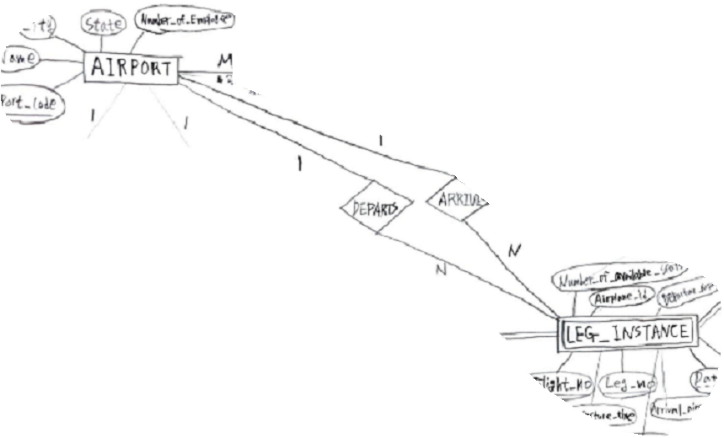
AIRPORT table에는 출발 airport code와 같은 airport code가 있다. FLIGHT\_LEG table의 arrival airport code와 LEG\_INSTANCE의 arrival airport code를 말하는 것이다. 바로 위 조건과 마찬가지로 arrival airport도 모두 관계로 엮어줘야 한다.



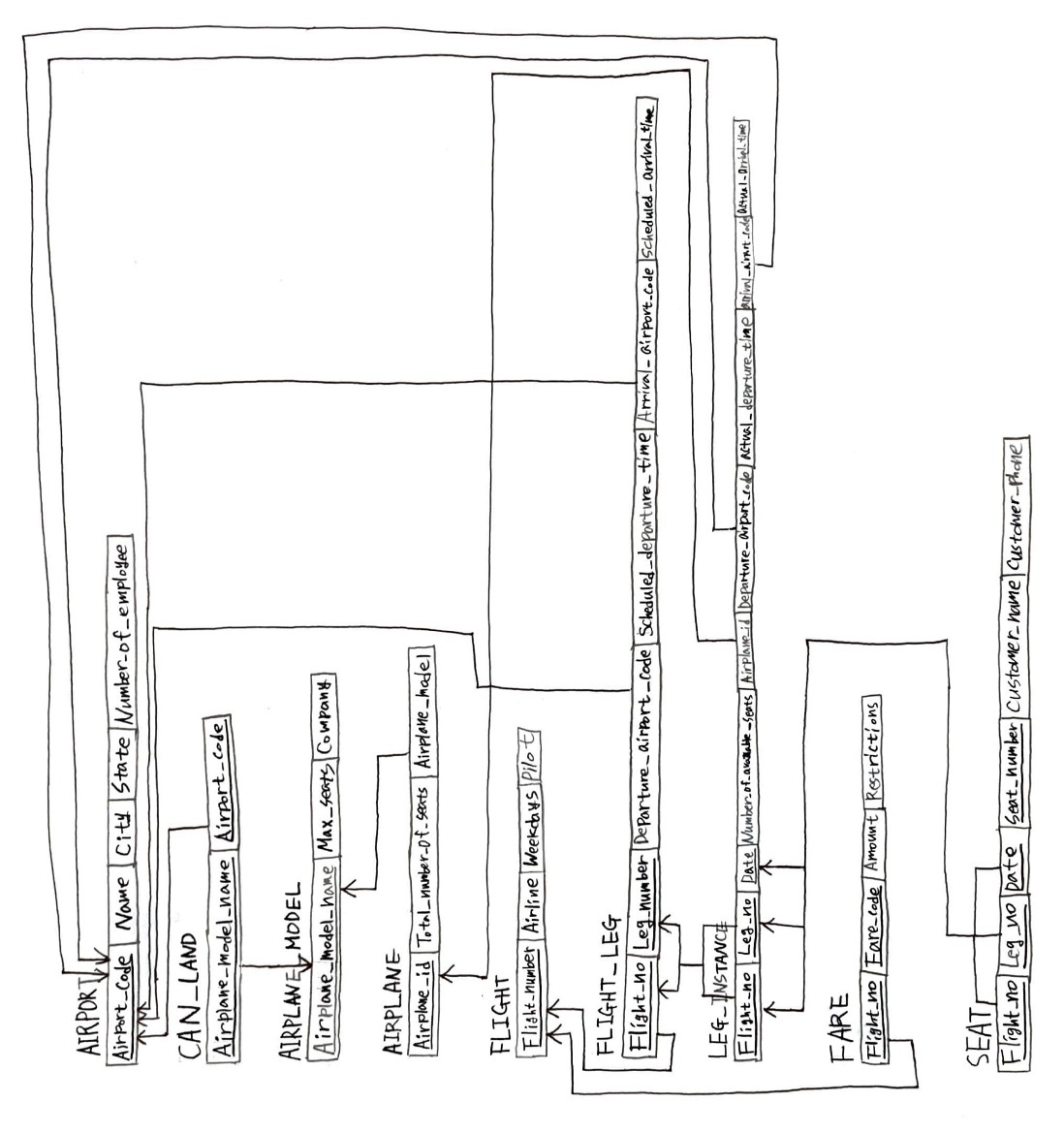
요구사항 분석을 통해 그린 ER Diagram이다. 위에서 거의 설명했지만, 아직 설명하지 못한 부분이 존재한다.



왼쪽 사진은 AIRPLANE table과 LEG\_INSTANCE table의 관계이다. 날짜에 대한 구간의 정보와 비행기 간의 관계다. 관계명은 알기 쉽게 FLIES로 표현했다. 하나의 비행기가 여러 날짜에 대해 운행할 수 있으니 1 : N 관계이다. 날짜에 대한 구간의 정보는 운행이 되어야만 존재하기 때문에 전체 참여로 설정하고, 비행기는 매일 운행을 하는 것은 아니기 때문에 부분 참여로 설정하였다.



이 부분도 위에서 설명하지 못한 부분이다. 날짜에 대한 구간 정보가 있으면 어디서 출발했는지 어디서 도착했는지에 대한 정보가 필요하기에 ARRIVES, DEPARTS 두 가지 관계를 활용하여 표현하였다.



위 사진은 내가 그린 ERD에 대한 DB 스키마이다. ERD에서 설명했던 내용으로 PK, FK를 고려하여 화살표를 그려주었다. 애초에 정규화를 고려하며 ERD를 그려서 정규화가 되어 있을 것이라고 생각했다. 하지만 정규화가 안되어 있을 가능성이 있기 때문에 모두 정규화가 되어 있는지 따져봤더니 원자성을 만족하여 1NF를 만족하고 있고, 부분 함수 종속이 없기 때문에 2NF도 만족한다. 마지막으로 이행 함수 종속이 있나 찾아봤는데, 이행함수 종속 역시 없기 때문에 3NF까지 만족하는 스키마이다. 스키마가 잘 설계된 것을 볼 수 있다.

정규화 과정 설명하기

텍스트이(가) 표시된 사진

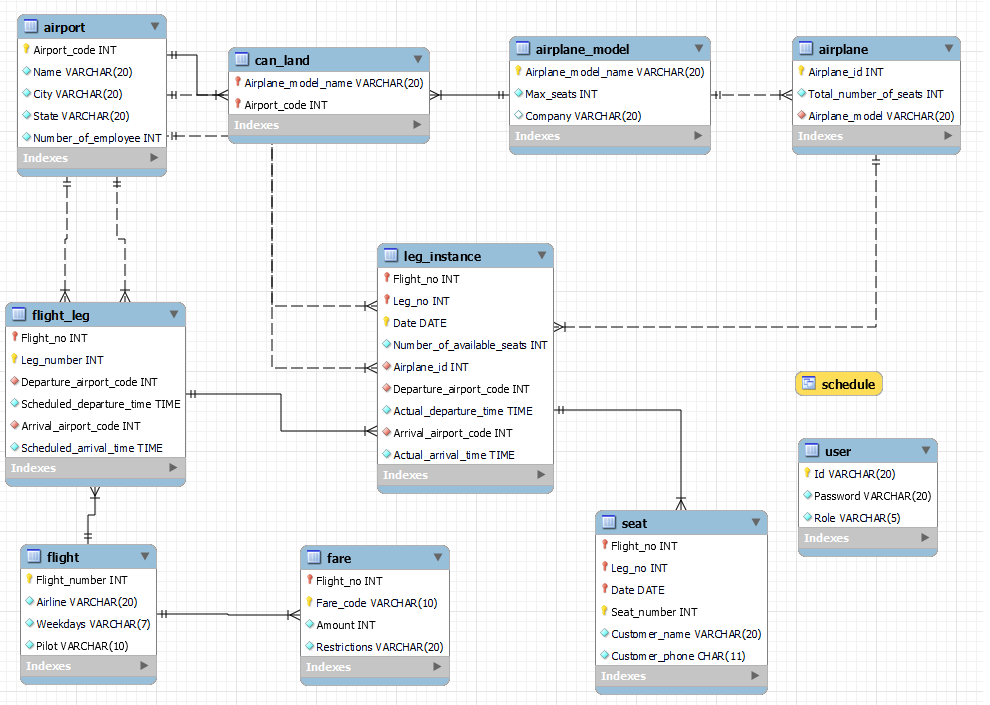
자동 생성된 설명

위와 같이 PowerShell 창에서 mysql에 접속한 후, table을 생성한다. CREATE문은 제약조건과 ERD를 통해 그린 스키마로 작성한다. 위 ERD를 통해 그린 스키마로 모두 설명이 되기 때문에 위 CREATE문과 나머지 create문의 설명은 생략한다. 나머지 모든 CREATE문은 txt파일로 제출할 예정이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 생성한 table의 조건에 알맞게 insert한 과정이다. Data type에 맞게 문제없이 insert된 모습을 볼 수 있다. 위와 같이 나머지 INSERT문도 txt파일로 제출할 예정이다.



Powershell에서 mysql에 접속하여 작성한 create문과 insert문, 그리고 create view문 등을 mysql workbench에 reverse engineering하여 나온 결과이다. 내가 그렸던 ERD와 같이 테이블과 관계가 출력된 것을 볼 수 있다. User는 따로 빠져 있는 데이터이기 때문에 관계가 없는 형태로 나타난 것이다. 노란색으로 보이는 schedule은 view이다. Create view를 할 때 이름을 schedule로 하였기 때문에 위와 같은 상태로 나타난 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 사진은 CREATE view를 해준 상황이다. 사용자가 예약을 하기 위해 필요한 정보들만 뽑아서 <http://localhost:3001/reserve> 예약페이지에 select되게 하였다. 내가 생각하기에 예약자에게 필요한 정보는 비행 번호, 경유 횟수, 비행기 번호, 비행 날짜, 출발 공항, 예정된 출발 시간, 도착 공항, 예정된 도착 시간, 좌석 등급에 따른 요금 정보, 수하물 무게 제한, 마지막으로 사용가능한 좌석수이다. 수업에서 배웠던 여러 가지의 join문을 활용하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

좌측과 같이 공항, 항공기, 항공편 정보를 입력/수정/삭제하기 위한 페이지를 만든다. Delete.js와 delete.hbs로 공항, 항공기, 항공편 세 가지 정보를 한 번에 삭제하는 페이지를 만들어서 버튼을 누르면 작동하게 만들고, insert, update도 모두 같게 만든다. Login.js와 login.hbs를 통해 로그인하는 화면과 로그인하는 동작까지 만들어준다. 내가 설계할 페이지는 다음 설명과 같다. 우선 로그인 페이지가 나오고, 관리자로 로그인하면, select 페이지가 뜬다. 현재 어떤 정보가 담겨 있는지 보여주는 화면이다. 만약 관리자가 정보를 쉽게 바꾸기 위해 select 화면에서 update, insert, delete 페이지로 바로 넘어갈 수 있도록 버튼을 만들었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Npm run start 명령어를 통해 페이지를 동작시키면 위와 같이 에러없이 작동이 잘 된다.

**3. 관리자 페이지 정보 수정**

텍스트이(가) 표시된 사진

자동 생성된 설명

Localhost:3001로 가게 되면, 위와 같은 창이 나온다.

관리자로 로그인하게 되면, select 페이지로 넘어가게 만들었다.

테이블이(가) 표시된 사진

자동 생성된 설명

위처럼 select 페이지가 뜬 것을 볼 수 있는데 여기서 위에 말했던 관리자가 정보를 바꾸고 싶을 때 다른 페이지로 넘어갈 수 있는 버튼을 만들었다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 버튼에서 delete를 눌러본다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 delete 페이지로 넘어온 것을 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Delete 페이지에도 위와 같은 버튼을 만들어서 서로 왔다갔다 할 수 있는 페이지를 제작했다.

위와 같이 update 페이지와 insert페이지도 왔다갔다 할 수 있게 만든다.

테이블이(가) 표시된 사진

자동 생성된 설명

관리자로 로그인하면, select되어 있는 공항 테이블, 항공기 테이블, 비행 테이블을 볼 수 있다.

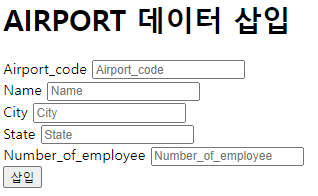
공항 정보를 건드리기 전 상태를 먼저 보여준다. 위와 같이 14개의 공항에 대한 정보가 담겨 있다.

1) 공항 정보 입력

공항 정보를 입력하기 위해 /insert 페이지로 넘어간다.

텍스트이(가) 표시된 사진

자동 생성된 설명



위와 같은 창이 뜨는 것을 볼 수 있다. 위 빈 칸에 값을 대입한다. 1번부터 14번까지의 airport\_code가 존재했다. 1~14값은 insert할 수 없다. 왜냐하면 airport\_code가 기본키이므로 중복될 수 없기 때문이다.

그래서 airport\_code가 15인 정보를 insert한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 정보를 입력한다.



확인하기 위해 select 버튼을 눌러서 페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 15번에 대한 정보가 insert된 것을 볼 수 있다.

2) 공항 정보 수정

현재 1~15의 공항의 값이 존재하는 것을 알 수 있다. 위 값을 수정해볼 계획이다.



위 버튼으로 update페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

상식적으로 공항에서 이름과 위치가 바뀌는 경우는 흔치 않기 때문에 위와 같이 그 공항에서 일하는 직원의 수를 추가해줬다. 15번 병찬공항에서 일하는 10000명을 5000명으로 수정한다.



수정 버튼을 누르고 select 버튼을 통해 확인한다.

테이블이(가) 표시된 사진

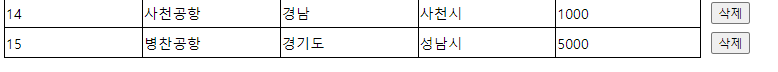
자동 생성된 설명

15번 공항의 직원수가 5000명으로 줄어든 것을 확인할 수 있다.

3) 공항 정보 삭제

현재 상태는 위에서 이어지는 상황이기 때문에 15번 병찬공항의 직원수가 5000명인 상황이다.

위와 같은 방법으로 delete페이지로 넘어간다.



우측의 삭제 버튼을 통해 15번 병찬 공항에 대한 정보를 삭제한다. 공항에 대한 정보를 삭제하게 되면 다른 table에 있는 공항에 대한 정보도 삭제해줘야 하기 때문에 삭제가 바로 안될 것이다. 참조 관계 때문이다. 내가 원하는 공항에 대한 정보를 한 번에 모두 삭제하고 싶다면 on delete cascade 명령어를 한 번에 삭제하고 싶은 table마다 추가해주면 된다. Update도 마찬가지이다. Cascade에 대한 설명은 여기서 다뤘기 때문에 다른 곳에서 설명은 생략할 예정이다. 하지만 위의 정보는 내가 삭제하기 위해 참조되지 않은 공항을 넣어놨기 때문에 삭제가 될 것이다. 다른 값들은 삭제가 되지 않는다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 삭제된 모습을 볼 수 있다.

4) 항공기 정보 입력

테이블이(가) 표시된 사진

자동 생성된 설명

수정하기 전 상태이다.

항공기 정보를 입력하기 위해 /insert 페이지로 넘어간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 창이 뜨는 것을 볼 수 있다. 위 빈 칸에 값을 대입한다. AIRPLANE TABLE이 AIRPLANE\_MODEL TABLE을 참조하고 있기 때문에 미리 Airplane\_model\_name값이 필요하다. 삭제해도 상관없는 값인 Airplane\_model\_name이 K-3인 AIRPLANE\_MODEL을 추가해둔다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Airplane 데이터를 삽입하기 위해 위와 같이 airplane\_model을 새로 추가해줬다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 정보를 입력한다.



확인하기 위해 select 버튼을 눌러서 페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 K-3에 대한 정보가 insert된 것을 볼 수 있다.

5) 항공기 정보 수정

현재 상태는 4)에서 마지막 사진이다.



위 버튼으로 update페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

비행기 번호나 Airplane\_model은 키로 설정되어 있기도 하고, 현실에서도 맘대로 바꾸기 힘든 값이기 때문에 좌석 수를 수정한다. 고장난 좌석이 수리되었다든지, 좌석이 고장났다든지 충분히 생길수 있는 상황이므로 위 값을 수정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

수정 버튼을 누르고 select 버튼을 통해 확인한다.

테이블이(가) 표시된 사진

자동 생성된 설명

Total\_number\_of\_seats가 600으로 줄어든 것을 확인할 수 있다.

6) 항공기 정보 삭제

현재 상태는 위에서 이어지는 상황이기 때문에 Total\_number\_of\_seats가 600인 상태이다.

위와 같은 방법으로 delete페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

위에서 설명한 것과 마찬가지로 leg\_instance table이 airplane table을 참조하고 있기 때문에 삭제가 안되어야 하지만 위에서 말했듯이, 이것 또한 삭제해도 상관없게 값을 insert해놨기 때문에 삭제가 된다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 삭제된 모습을 볼 수 있다.

7) 항공편 정보 입력

테이블이(가) 표시된 사진

자동 생성된 설명

수정하기 전 상태이다.

항공편 정보를 입력하기 위해 /insert 페이지로 넘어간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위에 보면 1~10까지의 정보가 있기 때문에 중복되면 안되서 다른 값인 11을 넣어준다.

위와 같은 창이 뜨는 것을 볼 수 있다. 위 빈 칸에 값을 대입한다. FLIGHT TABLE은 다른 값을

참조하고 있진 않고 참조를 받고만 있기 때문에 어떤 값을 삽입하는 데에는 지장이 없을 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 정보를 입력한다.



확인하기 위해 select 버튼을 눌러서 페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

위와 같이 flight\_number가 11인 flight table 정보가 insert된 것을 볼 수 있다.

8) 항공편 정보 수정

현재 상태는 7)에서 마지막 사진이다.



위 버튼으로 update페이지로 넘어간다.

테이블이(가) 표시된 사진

자동 생성된 설명

비행 번호, Airline은 쉽게 바꾸기 힘든 값이기에 새로 바꿀 값을 내가 추가해줬다. 그 운행에 대한 조종사의 정보를 수정할 예정이다. Flight\_number의 pilot값을 병일로 바꾼다.



수정 버튼을 누르고 select 버튼을 통해 확인한다.



Pilot이 병일로 바뀐 것을 확인할 수 있다.

9) 항공편 정보 삭제

현재 상태는 위에서 이어지는 상황이기 때문에 Pilot이 병일인 상황이다.

위와 같은 방법으로 delete페이지로 넘어간다.



이것 또한 FLIGHT\_LEG와 FARE가 참조하고 있어서 원래는 삭제되면 안되지만 위 값은 삭제해도 괜찮은 값이다.

테이블이(가) 표시된 사진

자동 생성된 설명

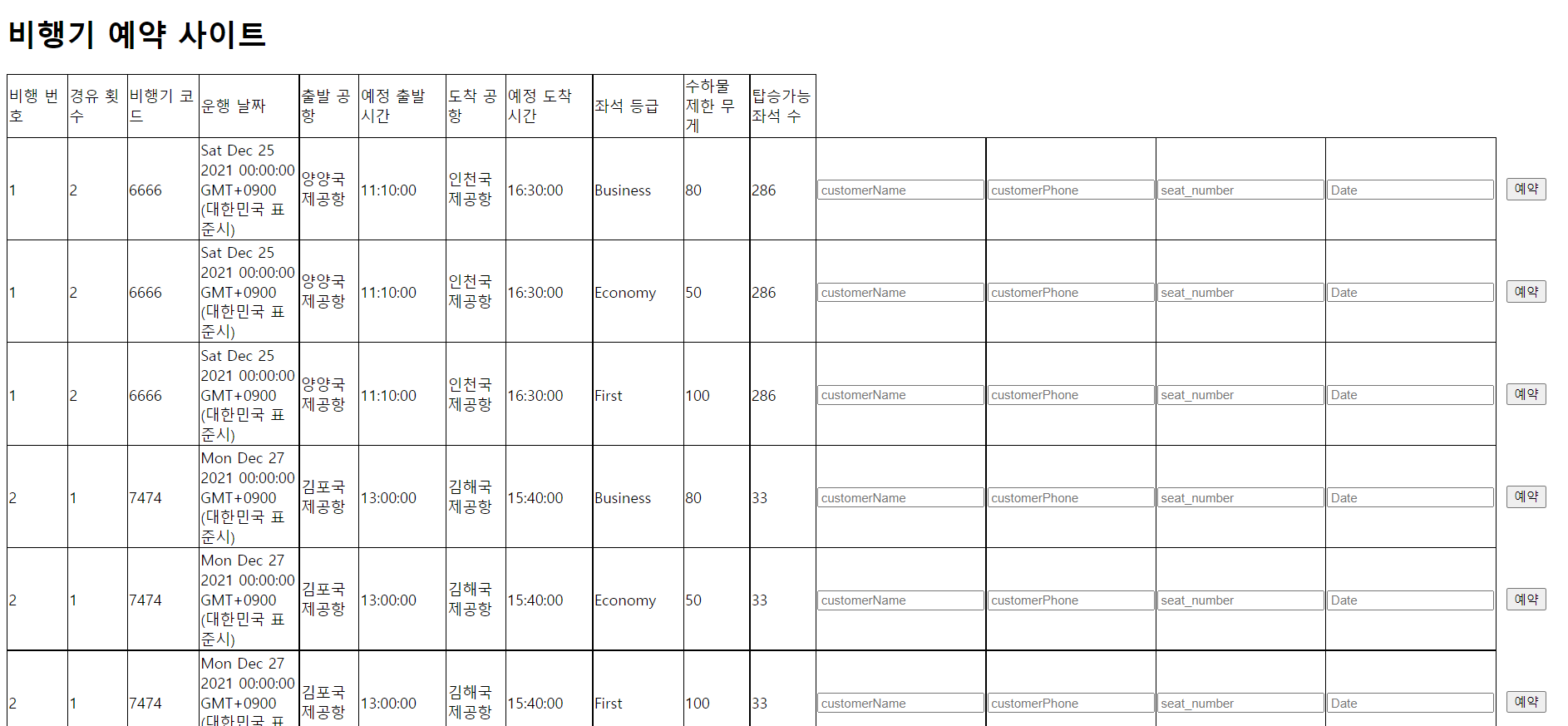
위와 같이 깔끔하게 삭제된 모습을 볼 수 있다.

**4. 사용자 페이지**

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로 위와 같이 로그인한다.



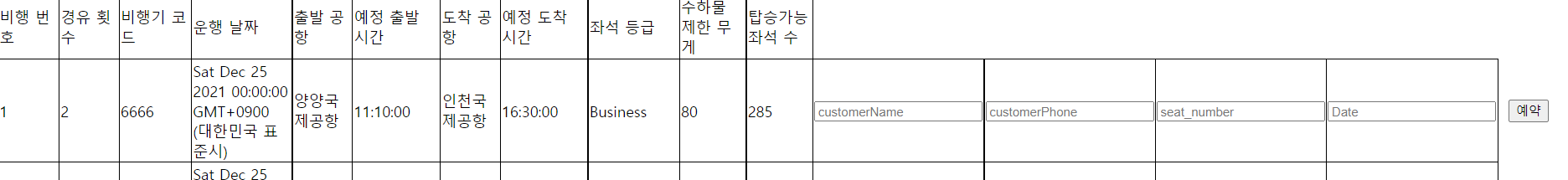
로그인하면 위와 같이 view로 select해준 정보들이 뜨면서 예약할 수 있는 사이트가 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 코드로 트랜잭션 기능을 추가하였다.

Start transaction을 하게 되면 autocommit이 off된다. commit하지 않은 상태에서 예약을 해봤는데 역시나 예약이 되지 않는 것을 확인할 수 있었다. 그래서 위 코드와 같이 commit을 해줌으로써 예약하면 탑승가능 좌석 수가 1 줄어들도록 코드를 만들어 주었다.



위 사진은 두 명의 예약자가 생긴 상태인데 2가 줄어든 것을 볼 수 있다.

텍스트이(가) 표시된 사진

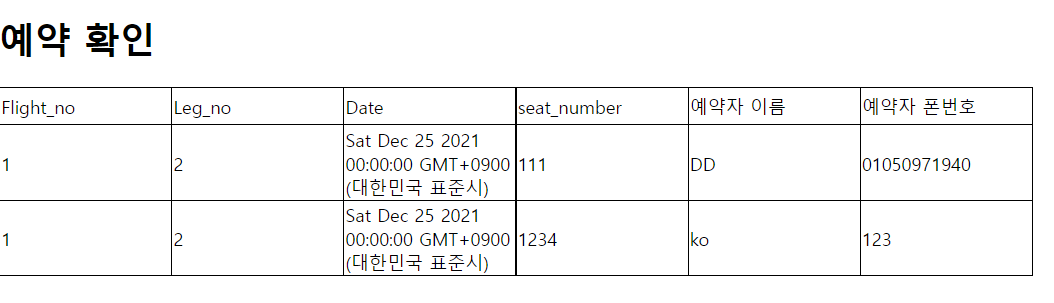
자동 생성된 설명

Reserve.hbs 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

한 명이 예약한 경우이다.



한 명 더 예약을 추가하자 위처럼 예약한 사람이 두 명인 것을 확인할 수 있었다.

예약을 취소하는 과정도 같은 방법으로 해주면 된다. 취소를 하면 예약 확인 사이트에서는 1명씩 지우고, 예약 사이트에서는 좌석이 1늘어나도록 트랜잭션을 설계해주면 취소하는 기능까지 완성할 수 있다.

**5. 고찰**

이번 학기에 데이터베이스 설계 수업을 들으며 데이터에 대한 관심이 정말 커진 듯 하다. 막연하게 빅데이터라는 말만 듣고 빅데이터를 해보고 싶다는 생각을 했었는데, 빅데이터에 대하여 조금 알게 되고, 데이터 관련 많은 자격증들이 있다는 것도 알게 되었다. 이 프로젝트를 하며 데이터베이스를 다루기보단 웹사이트를 다루는 과정을 더 많이 한 듯 하지만, 데이터베이스에 대한 전반적인 이해를 할 수 있었던 것 같다. 실습 과제에서 다뤘던 것들을 다시 반복하는 느낌이 많았지만 웹 프로그래밍은 겪어본 적이 없어서 코드가 매우 난잡한 느낌이 든다. 더 새로운 mini world를 자주 짜보는 경험을 해보고 싶다는 생각이 든다. 공항 mini world를 짜면서 느낀 점은 공항에 대하여 잘 알지도 못하는데 처음에 어떻게 해야할까 막연한 느낌이 컸지만 예전에 비행기 탔던 때를 떠올리고, 고민하고, 찾아보니까 어느 정도 공항에 대하여 이해할 수 있었다. 실제 세계와 완전히 일치할 순 없지만, 일치하기 위해 많은 이론들을 만들고, 여러 기능들을 만들어낸 데이터베이스를 설계한 사람들이 정말 대단하다는 생각이 다시 한 번 든다. 사용자 페이지를 만들어서 트랜잭션까지 적용해보는 과정은 실습에서 해보지 못한 기능이지만, 직접 해보며 수업에서 말씀해주셨던 autocommit의 역할에 대하여 자세히 알 수 있었다. 다른 데이터베이스에도 관심을 가지며 지속적으로 데이터베이스를 공부할 생각이다. 이 과목을 듣고 내가 갈 길을 DBA로 정하기도 하였다. 한 학기동안 유익한 수업을 들은 것 같아 뿌듯하다.