

ICE4016 데이터베이스설계

설계 프로젝트

AIRLINE 데이터베이스 구현 및 Web응용 작성

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 12월 12일

학부 정보통신공학과

학년 4

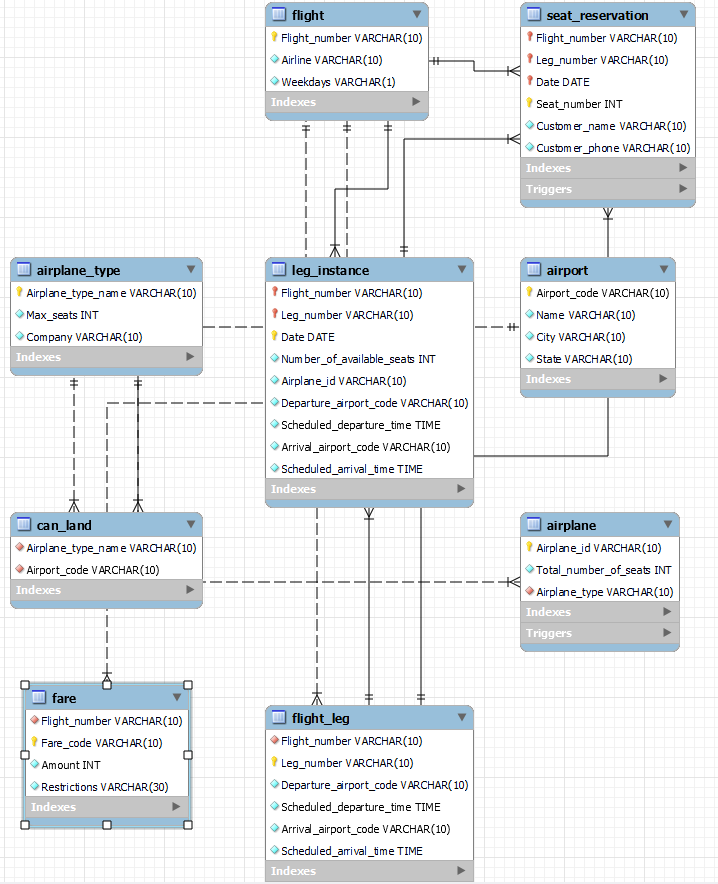
성명 이두열

학번 12151595



데이터베이스 과목의 설계 프로젝트로 AIRLINE, 즉 항공 시스템을 예약, 관리하는 데이터베이스를 만들고 Web에서 구현해 주었다. 항공 시스템은 퀴즈, 실습 등에서 진행한 COMPANY나 LIBRARY 같은 예와 같이 데이터의 삽입, 삭제, 관리가 중요한 시스템이라고 할 수 있다.

항공 시스템의 스키마와 속성 등을 먼저 보면, 관리자는 비행기 운항, 공항 등의 관리를 할 수 있고, 유저는 항공편에 대해 예약, 수정, 삭제를 할 수 있다. 이 과정에서 여러 테이블이 식별 관계와 제약조건들을 가지고 있다. 또한 예약 시스템 특성 상 사용자가 다른 사용자의 좌석을 예약할 수는 없을 것이며, 관리자에 의해 항공편이 취소되어도 예약할 수 없을 것이다.



MySQL의 스크립트를 기반으로 MySQL workbench에서 EER diagram을 작성해 주었다. 테이블간의 관계와 그린 순서를 설명하면, 먼저 AIRPORT 테이블과 FLIGHT 테이블의 경우 어떤 다른 테이블도 참조하지 않아 가장 먼저 생성해 주었다.

FLIGHT\_LEG 테이블은 FLIGHT 테이블의 Flight\_number를 외래키로 참조하며 Leg\_number가 함께 primary key가 되어야 한다. 왜냐하면 Leg\_number의 경우 주어진 항공 시스템에서 1, 2, 3의 방식으로 넘버링 되었는데, 다른 항공편일 경우 Leg\_number 혼자서는 메인키가 될 수 없기 때문이다.

다음으로 FARE 테이블을 설계했는데, FARE 테이블의 경우 Flight\_number를 외래키로 참조하며 이 경우에는 Fare\_code만 있어도 Flight\_number, Amount, Restrictions를 결정할 수 있다고 판단하여 Fare\_code만 primary\_key로 지정해 주었다. 여기서 Restrictions가 의아해서 찾아보았는데, 다른 항공권으로 바꿀 수 있는지 등의 제약사항을 적어 놓는다고 한다. 현재 설계에서는 크게 고려하지 않았다.

AIRPLANE\_TYPE의 경우 Airplane\_type\_name이 primary key가 된다. 나머지 정보들도 Airplane\_tpye\_name으로 결정될 수 있다고 보았다.

CAN\_LAND 테이블의 경우 Airplane\_type\_name과 Airport\_code 모두 외래키이자 primary key로, <Airplane\_type\_name, Airport\_code> 튜플이어야 착륙 가능 여부를 판단할 수 있다.

AIRPLANE 테이블은 Airplane\_id가 primary key이며, Total\_number\_of\_seats와 Airplane\_type 모두 AIRPLANE\_TYPE에서 참조할 수 있다. 그러나 Airplane\_type만 추가적으로 참조해 주었는데, 비행기 타입의 경우 비행기명과 같다고 두어도 되며 바뀔 일이 없다고 생각한다. 그러나 총 자리수의 경우 비행기 설계상 좌석수보다 수리 등의 이유로 적을 수 있다고 판단했다.

LEG\_INSTANCE 테이블은 Flight\_number와 Leg\_number를 외래키로 참조하며 추가로 날짜까지 합해 Primary key가 되며 이유는 위와 마찬가지이다. 여기서 LEG\_INSTANCE의 경우 이용 가능한 좌석 수를 AIRPLANE의 Total\_number\_of\_seats와 SEAT\_RESERVATION의 카운트로 계산해 주었다.

마지막으로 SEAT\_RESERVATION의 경우 Flight\_number, Leg\_number, Date를 참조하고 Seat\_number와 함께 primary key가 된다. 이 경우에도 비행기나 leg가 다를 경우 좌석번호 혼자 메인키가 될 수 없기 때문이다. 또한 Customer\_name과 Customer\_phone의 경우 Customer\_phone이 Customer\_name을 결정한다고 볼 수 있다. 고객명이 같을 수는 있어도 고객의 전화번호가 같을 수는 없기 때문이다. 이 경우에 정규화로 이행 함수 종속성을 없앨 수 있다고 생각하지만, 설계 설명 과정에서 user테이블을 분리하지 않아도 된다고 말씀하셔서 분리하지 않았다. 내 설계적으로는 seat\_reservation에서 고객명과 고객번호가 포함되는 것이 더 이상적이라고 생각하기 때문이다. 다른 테이블도 마찬가지로 원본에 가깝게 유지되었으며, 기본적으로 2NF, 3NF를 만족한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

생성 스크립트의 경우 사용할 테이블과 예시들을 위와 같이 넣어주었다. 예시로 든 예제의 경우에는 서울->인천->부산->제주 공항으로 이동하는 항공 라인이다. Leg에 대한 예시를 돕기 위해 위와 같은 예를 들었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블 선언만으로 불충분한 제약조건들의 경우 트리거를 통해 구현해 주었다. 먼저 가장 중요하게 생각한 부분은 사용자가 좌석을 예약하면 남은 좌석수가 줄어들어야 한다는 점이다. 이 경우 아래의 트리거와 같이 SEAT\_RESERVATION 테이블에 새로운 값이 insert 될 때 수행되며, 각 레그마다 Total\_number\_of\_seats에 count(\*) SEAT\_RESERVATION을 뺀 값을 이용 가능한 좌석 수로 계산해 주었다.

또한 앞서 설계한 과정에서 비행기 설계 좌석과 실제 총 좌석이 다를 수 있다고 이야기했는데, 다를 수는 있어도 비행기 설계 좌석보다 실제 총 좌석이 많을 수는 없을 것이다. 따라서 트리거를 통해 Total\_number\_of\_seats가 삽입될 경우 대소비교를 통해 Max\_seats보다 작은 값이 되도록 설정해 주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

색인 기능의 경우 사실상 항공 시스템 특성상 제한된 비행기의 수를 조회하는 경우보다 고객의 데이터를 조회하거나 수정하는 일이 훨씬 많을 것이다. 따라서 색인, view 등의 기능을 seast\_reservation 테이블에 대해서 구현해 주었다. 먼저 색인의 경우 seat\_number를 index삼아 탐색을 수행하도록 구현했다. B트리로 구현해 주었는데 Primary key들 중에서 가장 대소비교에 적합하기 때문이다. 위와 같이 구현해 주었으며 test2라는 view를 통해서도 확인할 수 있다. 사실 mySQL은 기본적으로 B트리로 구현되어 있어 index를 primary key로 지정했기 때문에 사실상 명시적 구현에 가깝다고 볼 수 있다.

START TRANSACTION;

INSERT INTO Seat\_reservation VALUES(“S2J”, “1”, “2021-12-01”,”51”,”LDK”,”33333”);

IF ((select \* from seat\_reservation where Seat\_number=”51”) != null) rollback;

COMMIT;

START TRANSACTION

INSERT INTO Seat\_reservation VALUES(“S2J”, “1”, “2021-12-01”,”51”,”LDK”,”33333”);

IF ((select \* from Flight\_number where Flight\_number=”S2J”) != null) rollback;

COMMIT;

트랜잭션의 경우 이 시스템에서 필수적으로 구현되어야 한다고 생각하는 경우는 2가지 case이다. 첫째로 사용자가 좌석 예약 중 다른 사람이 먼저 좌석 예약을 하게 될 경우 rollback해야 할 것이고, 둘째로 사용자가 좌석 예약 중 관리자가 항공편을 없애버리면 rollback해야 할 것이다. 위와 같은 방식으로 구현할 수 있을 것이며, 이 역시 mySQL에서 기본적으로 구현되어 있는 기능이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

기존의 코드를 활용하여 메인 페이지에 로그인 화면이 뜨게 된다. Admin계정을 통해 관리자 페이지, test 계정을 통해 사용자 페이지로 갈 수 있으나, 실질적으로 설계에 불필요한 user 테이블을 설계하기 때문에 주소창을 통해 이동하는 것이 바람직하다.

텍스트이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

본 설계에서는 관리자가 insert 페이지에서 Airport, Airplane, Flight 데이터를 모두 삽입할 수 있도록 만들어 주었다. Update 페이지와 delete 페이지도 마찬가지이다. 조회 페이지를 따로 만들지 않았기 때문에 삽입, 수정을 하게 될 경우 페이지는 다시 해당 페이지로 돌아오게 된다. Update 페이지의 경우에는 primary key가 되는 첫번째 인자들은 그대로 입력해 주어야 query문에서 해당 행을 인식하여 업데이트 할 수 있다. 또한 제약조건들이 있기 때문에 비정상적인 값들을 넣게 되면 삽입, 수정되지 않는다.

테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

Delete 페이지에서 삽입, 수정한 결과가 정상적으로 반영되었음을 확인할 수 있다. 또한 삭제 페이지에서도 삭제 버튼을 누르면 삭제가 정상적으로 이루어짐을 알 수 있다.

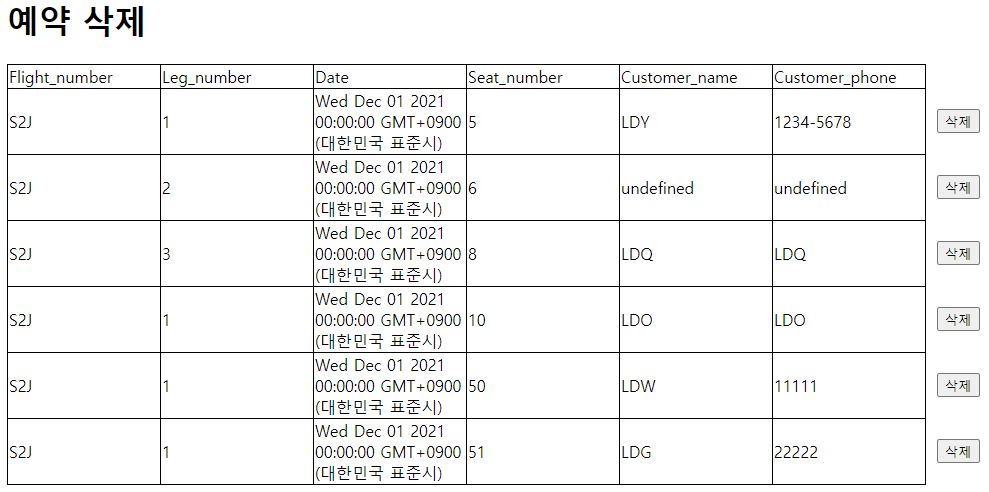
테이블이(가) 표시된 사진

자동 생성된 설명

고객의 경우에도 예약조회, 예약수정, 예약삭제가 가능해야 한다. 각각 userSelect, userInsert, userDelete 페이지에 구현되어 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

삽입 페이지와 삭제 페이지도 정상적으로 작동함을 확인할 수 있다.

이번 설계 프로젝트를 통해 총체적인 시스템을 처음부터 끝까지 구현했다고 볼 수 있다. 먼저 처음부터 고민했던 점은 생각보다 항공 시스템이 직관적이지 않은 요소들이 있다는 점이다. Restriction 같은 경우에는 비행기를 타본 경험이 있어도 신경 쓰지 않았었다. 이러한 특성상 설계 프로젝트가 은행 DB 설계, 보안 설계 등의 보다 복잡한 수식이나 개념을 요하는 설계였으면 난이도가 더 높았을 것이다. 설계상 아쉬운 점들은 Index, Transaction 등은 수업에서만 진행하고 실습의 기회가 없어 스스로 연습해보지 않아 미숙했던 것 같다. 또한 개개인마다 설계가 상당히 다를 것으로 예상되는데, field에서는 어떠한 방식으로 통일 되어있는지도 알아보아야 할 것 같다.

깃허브주소 : https://github.com/LEEDUYEOL/2021-02-database